

# Comparison of Exact Static and Dynamic Bayesian Context Inference Methods for Activity Recognition

Korbinian Frank\*<sup>‡</sup>, Matthias Röckl\*, María Josefa Vera Nadales<sup>†</sup>, Patrick Robertson\* and Tom Pfeifer<sup>‡</sup>

\**Institute of Communications and Navigation*

*German Aerospace Center (DLR)*

*Oberpfaffenhofen, Germany*

*Email: firstname.lastname@dlr.de*

<sup>†</sup>*E.T.S. de Ingeniería de Telecomunicación*

*Universidad de Málaga*

<sup>‡</sup>*Telecommunications and Software Systems Group (TSSG)*

*Waterford Institute of Technology*

**Abstract**—This paper compares the performance of inference in static and dynamic Bayesian Networks. For the comparison both kinds of Bayesian networks are created for the exemplary application activity recognition. Probability and structure of the Bayesian Networks have been learnt automatically from a recorded data set consisting of acceleration data observed from an inertial measurement unit. Whereas dynamic networks incorporate temporal dependencies which affect the quality of the activity recognition, inference is less complex for dynamic networks. As performance indicators recall, precision and processing time of the activity recognition are studied in detail. The results show that dynamic Bayesian Networks provide considerably higher quality in the recognition but entail longer processing times.

**Keywords**-Probabilistic Inference, Context Inference, Activity Recognition, Activity Estimation, Acceleration Sensors

## I. INTRODUCTION

Combining input from different, homogeneous as well as heterogeneous sources to create context information has become well-accepted for building ubiquitous systems. Data fusion approaches are being investigated to manage the heterogeneity.

Inferring new information from factual sources (i.e. physical sensors) can be considered as a new, virtual source in the field of data fusion. With an increase of factual and inferred sources, systems can become inherently complex, their scalability needs to be considered and managed.

With the exemplary usage of context data for activity recognition in mind, this paper addresses the complexity issue and compares the performance of different approaches of inference. In particular, the objective of this paper is to study the benefits and drawbacks of using two different probabilistic inference approaches, on the one hand neglecting, on the other hand explicitly dealing with the temporal domain of causal influences. In this work we will give an introduction to Bayesian Networks (section III), Hidden Markov Models (section V) and probabilistic inference (section IV), apply these methods to a realistic

scenario from activity recognition (section II) and compare the outcome (in sections VII and VI) with respect to (1) execution time and (2) correctness compared to the recorded ground truth.

## II. ACCELERATION DATA BASED ACTIVITY INFERENCE

For our comparison of inference algorithms, we chose a very relevant and realistic environment, namely activity recognition. In order to organize operations of police, first responders or e.g. firefighters, knowledge about the current or previous motion related activities is very helpful, as mission coordinators can react faster like that on unforeseen events or upcoming difficulties. Equally the current activity is relevant for indoor positioning and navigation (see for instance Widyawan in [1]).

In all use cases, the requirement of real-time recognition without additional infrastructure becomes obvious. Therefore suitable inference methods have to be investigated and compared. To compare them we use the following settings:

We focused on a set of seven important motion related activities that are useful for multi-sensor indoor positioning, as well as for emergency missions. Activities comprise repetitive ones, such as *walking* and *running*, the static activities *standing*, *sitting* and *lying*, as well as short-time activities *falling* and *jumping*.

The measurements to infer the activities are coming from a single Inertial Measurement Unit (IMU) that is providing data about the magnetic field (in three dimensions), the turn rates in three dimensions and acceleration in three dimensions. Having compared the performance of this sensor for different parts of the body, we decided to position it on the belt, as at this location it provides information about both the upper and the lower part of the body.

From the measurements we calculate eighteen features from the norm of the angular velocity, the attitude, vertical and horizontal acceleration, the angle between local gravity axis and the global vertical axis, as well as the norm of

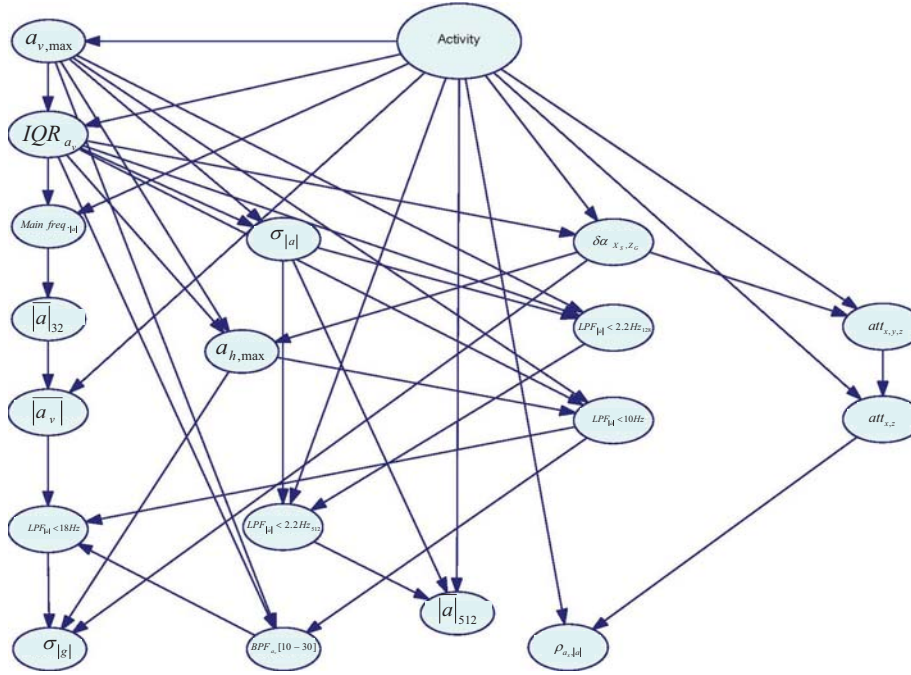


Figure 1. This Bayesian network was learnt from a data set recorded from 16 different persons. It shows the dependencies of the current "Activity" on 18 features calculated from an Inertial Measurement Unit including magnetometer, a 3D gyro and 3D accelerometers.

acceleration and jerk. These features were calculated for a data set we recorded from 16 persons (6 female, 10 male) over in total more than 4.5 hours. They were quantized and fed into a learning algorithm using a K2 algorithm by Cooper and Herskovits [2] that produced the network shown in Fig. 1.

### III. BAYESIAN NETWORKS

A *Bayesian Network (BN)* [3] is a probabilistic model consisting of a Triplet  $(V, E, P)$ , with a set of *Random Variables (RVs)*  $V = \{A_1, A_2, \dots, A_n\}$ , a set of dependencies  $E = \{(A_i, A_j) | i \neq j, i, j \in V\}$  between these RVs and a *joint probability distribution (JPD)*  $P(V) = P(A_1 \cap A_2 \cap \dots \cap A_n)$ .  $P$  is the product of the *Conditional Probability Distribution (CPD)* of every RV  $P(A_i) \forall A_i \in V$ . A BN must not contain directed cycles. This model subsumes a great variety of other stochastic methods, such as Hidden Markov Models or stochastic dynamic systems [4]. It allows for inference of knowledge being able to deal with missing or uncertain data (as for erroneous sensors or uncertain data links) and can be built or modified either by machine learning algorithms or by human expertise.

RVs represent sets of events. Thereby they can be continuous or discrete, which has consequences on CPDs. In the case of a continuous value range  $R$ , the CPD is a function  $CPD(A_i) : R \rightarrow [0, 1]$ , in the case of a discrete value range,  $R$  consists of a finite number of states, that are assigned a probability depending on the state of the nodes,  $A_i$  is

depending on.

A particular view on BNs are *Causal Networks*, where dependencies are interpreted as causal influence. This model makes understanding of such a network very intuitive, in particular with a graphical representation of the BN. A BN can be drawn as a *directed acyclic graph (DAG)* like the one in Fig. 1. These graphs take advantage of the fact, that with its explicit dependencies, a BN exploits the conditional independence to represent a JPD more compactly [5]. Every RV represents a node or vertex in the graph, every dependencies  $(A_i, A_j)$  a directed edge from node  $A_i$  to node  $A_j$ . This representation imposes the understanding of the set  $pa(A_j) = \{A_i | \forall i \in V \wedge (A_i, A_j) \in E\}$  as the *parents of*  $A_j$ . The definition of *children of*  $A_j$   $ch(A_j)$  is analogous. Dependencies and therefore parents help to represent the JPD more compact and by that to compute it faster:

$$P(V) = P(A_1 \cap A_2 \cap \dots \cap A_n) = \prod_{i=1}^n P(A_i | pa(A_i)) \quad (1)$$

With the structure (RVs and their dependencies) and the CPDs these networks contain the already known information about a specific domain represented by the BN. They are a knowledge representation and maintenance format. To incorporate current observations about the domain, these can be introduced as *evidence* into the corresponding RV. The observation that RV  $A_j = a_{j,1}$  sets  $P(A_j = a_{j,1}) = 1$  and  $P(A_j = a_{j,x}) = 0 \forall x \neq 1$ . In the case of discrete RVs, this can be interpreted as "switching" the probability tables of

children nodes to the observed columns.

An important concept for BNs is *d-separation* with the "d" standing for dependence. It helps to reduce the network to relevant parts of it for given observations and a specific target RV whose state is queried. If two variables are d-separated relative to a set of variables  $Z$ , then they are independent conditional on  $Z$  in all probability distributions of its BN. Roughly, two variables  $X$  and  $Y$  are independent conditional on  $Z$  if knowledge about  $X$  gives you no extra information about  $Y$  once you have knowledge of  $Z$  [6].

More precisely: a *path* is a sequence of consecutive edges including one or more nodes. A path is called blocked or *d-separated* if a node on the path blocks the dependency. This is the case if the path  $p$  and the set of observed nodes  $Z$  are in a constellation in which

- "p contains a chain  $i \rightarrow m \rightarrow j$  or a fork  $i \leftarrow m \rightarrow j$  such that the middle node  $m$  is in  $Z$ , or"
- "p contains an inverted fork (or collider)  $i \rightarrow m \leftarrow j$  such that the middle node  $m$  is not in  $Z$  and such that no descendant of  $m$  is in  $Z$ ."

The d-separation criterion can be summarized by: "a node is conditionally independent of its non-descendants, given its parents" or "a node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents – that is, given its Markov blanket" [7]. This means that the Markov blanket of a node is the only knowledge needed to predict the behaviour of that node [3]. The values of the parents and children of a node evidently give information about that node. However, its children's parents also have to be included, because they can be used to explain away the node in question. For the node `Activity` the Markov Blanket is shown shaded in Fig. 2.

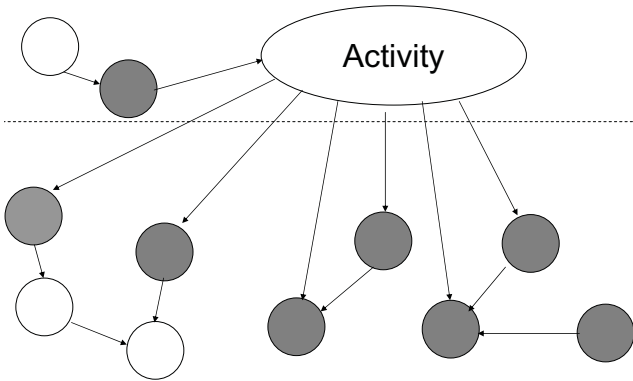


Figure 2. The Markov Blanket of the node `Activity` in a simplified example BN is shown in gray. It contains the node's parents, children and the parents of the children without the node itself.

#### IV. EXACT INFERENCE IN STATIC BAYESIAN NETWORKS

Exact Inference in BNs or *belief updating* [8], i.e. calculating the  $P(X|E)$  of a set  $X$  of RVs given evidence  $E$ ,

is in general a NP-hard problem as Cooper has shown in [9]. Only structural limitations of the network or knowledge about observed nodes allow for efficient solutions.

Hence a lot of research has gone into finding faster algorithms for inference. The pure calculation of the joint probability, as shown in Eq. 1, is often not an option if the network structure is very complex and contains undirected cycles. It is however the fastest option if the Markov Blanket of the queried node is observed. Hence it is always useful to reduce the network as far as possible taking advantage of the d-separation constellations. A well known algorithm for this task is the Bayes Ball algorithm developed by Shachter in [10]. Moreover research focused on the one hand on exact inference methods and on the other hand on approximated inference (which can also be proven to be NP-hard, see Dagum in [11]) and real-time inference, where not only the accuracy, but also the timeliness of the computation has to be taken into account. After a computation deadline, the utility of the result degrades, see [12] for details.

A universally usable and very popular inference algorithm is *Probability Propagation in Trees of Clusters (PPTC)* by Lauritzen and Spiegelhalter [13]. PPTC takes advantage of proven fast inference algorithms in tree-like structured BNs. It therefore transforms the possibly multiply connected network into a tree of supernodes, so called *cliques* or *clusters* [14], in which the *message passing* algorithm can be used for propagation of probabilities. To form these trees of clusters, the DAG of the original BN has to be moralized (linking all parent nodes of a node) and triangulated (connecting all nonadjacent nodes in cycles with length greater or equal to four). Clusters are then maximal and complete undirected subgraphs, that represent original nodes with their related nodes. They are assigned the potential of the contained nodes. A *Sepset* is representing an edge in this supernode tree and assigned with the potential of the intersection of the two clusters linked by it. Propagation of the probabilities is twofold, it is separated in a *collect-evidence* and a *distribute-evidence* step, each passing and processing once every cluster. To finally obtain the queried result, we have to marginalize a containing cluster potential into the queried RV and normalize it by the probability of the evidence. This approach works efficiently for sparse networks, but has problems with dense networks, as its complexity is exponential in the size of the largest clique [8].

For our use case, we restrict ourselves to exact inference methods, in particular as we can build on an important factor: we know that always all features will be observed. This is a valid assumption in our case, as the complete raw data are transferred to the processing unit and either every or no (if sensor connection is lost) feature can be computed. In case no feature could be computed, we were stuck with the prior distribution for the node `Activity`. Hence under the given circumstances we can limit inference

to the computation of the posterior based on the nodes in the Markov Blanket, which are always observed.

The prior distribution of the node `Activity` was not learnt to avoid overfitting with our data set. Using human expertise it was set to the following values representing normal daily distributions:

| Activity | Probability |
|----------|-------------|
| Sitting  | 0.195       |
| Standing | 0.245       |
| Walking  | 0.295       |
| Running  | 0.1         |
| Jumping  | 0.01        |
| Falling  | 0.005       |
| Lying    | 0.14        |
| Up/Down  | 0.01        |

## V. EXACT INFERENCE IN DYNAMIC BAYESIAN NETWORKS WITH A GRID BASED FILTER

Under the assumption that successive activities are independent of each other, decisive knowledge is wasted. In most cases the last activity you performed influences the current activity you are doing. For instance, if you are *lying*, the most probable activity you will perform after it is getting *up* or still *lying*, but not *falling*. This knowledge can provide valuable input for activity recognition.

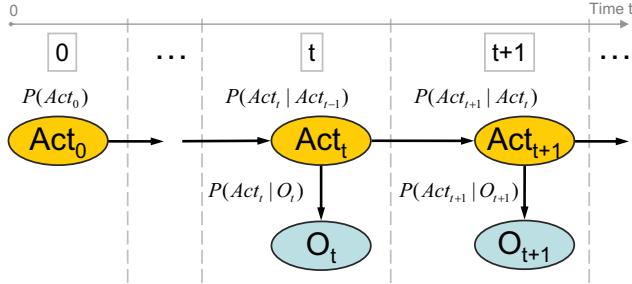


Figure 3. Hidden Markov Model scheme.

Activity recognition can be considered as the estimation of a hidden process which discloses periodic evidence by sensor measurements. A respective first-order *Hidden Markov Model (HMM)* [15] for the activity recognition (see Figure 3) can be characterized by the following:

- $N$ , the number of states of the hidden variable which correspond to the different activities considered in the approach. The individual states at time  $t$  are  $Act_t = \{act_{1,t}, act_{2,t}, \dots, act_{N,t}\}$ . Activities are considered as hidden because they cannot be observed directly.
- The physical output which can be observed is called the observation symbols. An observation symbol for a single point in time is given by a vector with values for all features  $f_1, \dots, f_M$  computed from the raw sensor data. The vector of features is common for all the hidden states and can be denoted  $O_t = (f_{1,t}, f_{2,t}, \dots, f_{M,t})$

where  $f_{i,t}$ ,  $1 \leq i \leq M$  is the value of the feature  $i$  at time  $t$ .

- The state transition probability distribution or transition model  $A = \{a_{ij}\}$  where  $a_{ij} = P(act_{j,t+1} | act_{i,t})$ ,  $1 \leq i, j \leq N$ .  $A$  will be represented as a matrix and is given in table I.
- The observation symbol probability distribution in state  $j$ ,  $B = \{b_j(k)\}$ , where  $b_j(k) = P(f_{k,t} | act_{j,t})$ ,  $1 \leq j \leq N, 1 \leq k \leq M$  also called measurement model in Bayesian algorithms. This probability is given by the Bayesian network.
- The initial state distribution  $\pi = \{\pi_i\}$  where  $\pi_i = P(act_{i,0})$ ,  $1 \leq i \leq N$  is the prior probability. In our evaluations we assumed a startup with the activity *Standing*. Therefore the prior assigns a 100% probability to this state and zero to the other activities.

Once the first-order HMM is defined and denoted as  $\lambda$ , the objective is to estimate the most probable hidden state at time  $t$  given the past and current observations  $O_{1:t} = O_1, O_2, \dots, O_t$  as well as the model  $\lambda$ . It is given by  $\arg \max_i P(act_i | O_{1:t}, \lambda)$ .

As in our system the hidden state space has a finite number of states (i.e. activities), grid-based methods can be applied providing an optimal estimation of the posterior probability  $P(act_t | O_{1:t}, \lambda)$ . The so called *Grid-based Filter* [16], as any recursive Bayesian filter, consists of two successive stages: *prediction* and *update*.

- In the prediction stage, the transition model defined in the HMM is used to predict the current probability distribution over the hidden state space given the past observations or  $P(act_t | O_{1:t-1}, \lambda)$ .
- In the update stage, the latest observation is used to correct the predicted probability distribution. With the update, the posterior probability distribution  $P(act_t | O_{1:t}, \lambda)$  is obtained.

According to [17] the prediction and update equations for the Grid-based Filter are defined by:

Prediction Equation:

$$P(act_t | O_{1:t-1}, \lambda) = \sum_{i=1}^N w_{i,t|t-1} \delta(act_t - act_{i,t}), \quad (2)$$

$$\text{with } w_{i,t|t-1} \triangleq \sum_{j=1}^N w_{j,t-1|t-1} P(act_{i,t} | act_{j,t-1}, \lambda) \quad (3)$$

Update Equation ( $\delta(\cdot)$  is the Dirac delta measure):

$$P(act_t | O_{1:t}, \lambda) = \sum_{i=1}^N w_{i,t|t} \delta(act_t - act_{i,t}), \quad (4)$$

$$\text{with } w_{i,t|t} \triangleq \frac{w_{i,t|t-1} P(O_t | act_{i,t}, \lambda)}{\sum_{j=1}^N w_{j,t|t-1} P(O_t | act_{j,t}, \lambda)} \quad (5)$$

| $act_t \backslash act_{t+1}$ | Up       | Sitting  | Standing | Walking  | Running  | Jumping  | Falling  | Lying    | Down     |
|------------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| Up                           | 0.052629 | 0.421030 | 0.526288 | 0        | 0        | 0        | 0.000053 | 0        | 0.000001 |
| Sitting                      | 0.025860 | 0.948194 | 0        | 0        | 0        | 0.000043 | 0.000043 | 0        | 0.025860 |
| Standing                     | 0        | 0        | 0.560735 | 0.420551 | 0        | 0.004673 | 0.000023 | 0        | 0.014018 |
| Walking                      | 0        | 0.043375 | 0.433745 | 0.433745 | 0.065062 | 0.002169 | 0.000217 | 0        | 0.021687 |
| Running                      | 0        | 0        | 0        | 0.486510 | 0.398054 | 0.079611 | 0.000442 | 0        | 0.035383 |
| Jumping                      | 0        | 0        | 0.353534 | 0.050505 | 0.252525 | 0.303030 | 0.000003 | 0        | 0.040404 |
| Falling                      | 0        | 0.266667 | 0        | 0        | 0        | 0        | 0.400000 | 0.333333 | 0        |
| Lying                        | 0.1      | 0        | 0        | 0        | 0        | 0        | 0        | 0.900000 | 0        |
| Down                         | 0.000094 | 0.660004 | 0        | 0        | 0        | 0        | 0.000471 | 0.282859 | 0.056572 |

Table I  
STATE TRANSITION PROBABILITY MATRIX  $A$ . EACH CELL DEFINES THE TRANSITION PROBABILITY  $P(act_{t+1}|act_t)$ .

For the evaluations the respective observation likelihoods  $P(O_t|act_{i,t}, \lambda)$  were learned from the recorded data sets and the state transitions  $P(act_{i,t}|act_{j,t-1}, \lambda)$  were manually configured by expert knowledge (see table I).

Once the posterior probability is estimated, the most probable activity is given by the state with maximum probability.

## VI. RESULTS

This section gives the inference results for the static approach multiplying the conditional probabilities and the dynamic approach with the Grid-based Filter. They were obtained from the data of a female (Sinja) and a male (Emil) subject who were not included in the training set of the BN. Both performed the activities considered in the system during 200 seconds following a fixed schedule, but with freedom to interpret every activity in an individual style.

The data processed were recorded as described in section II with the IMU at the belt of the subjects. The sample frequency of the sensor was set to 100  $Hz$  and features are computed at a frequency of 4  $Hz$  giving a new estimation of `Activity` every 0.25  $s$ . The data were processed using an Intel Core Duo microprocessor at 3.00  $GHz$  with 3 GB of RAM.

The following figures show the results during the 200 seconds of evaluation. A constant line on top of the figure depicts the ground truth, colours identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.

Figure 4 shows the evaluation of the activities *standing*, *sitting* and *standing* for Emil. The evaluation of the system during the transitions is not taken into account, although they were modelled in the system to implement the dynamic model.

*Jumping*, *walking* and *running* are shown in Fig. ?? for Sinja.

Finally, the results of the inference for the activities *falling* and *lying* are shown in Fig. 6.

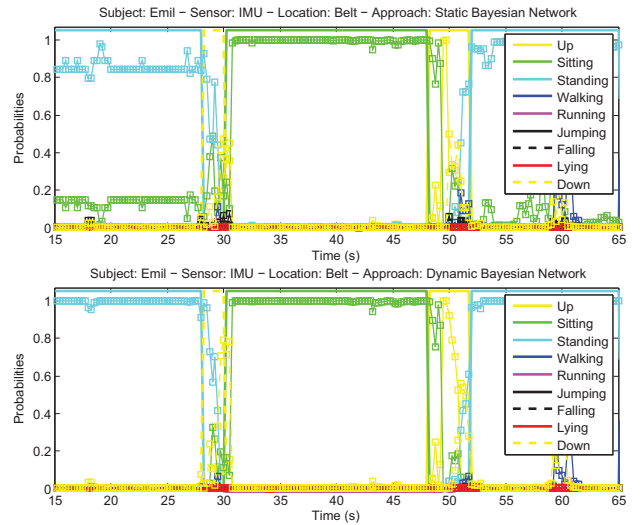


Figure 4. Evaluation results for the activities *standing*, *sitting* and *standing* using static (above) and dynamic (below) Bayesian inference. The top line represents ground truth, the curves below the estimated probabilities.

## VII. EVALUATION

### A. Inference Quality

In this section, the quality of both inference methods will be evaluated compared to the ground truth using precision and recall.  $Precision = \frac{TP}{TP+FP}$  and  $Recall = \frac{TP}{TP+FN}$  are defined like in [18], with  $TP$  being *True Positives*,  $FP$  being *False Positives* and  $FN$  being *False Negatives*.

|           | SIT  | STA  | WLK  | RUN  | JMP  | FAL  | LYG  | UpD  |
|-----------|------|------|------|------|------|------|------|------|
| Recall    | 0.98 | 0.9  | 0.99 | 0.82 | 0.56 | 0.75 | 0.97 | 0.24 |
| Precision | 0.84 | 0.92 | 0.86 | 0.84 | 0.76 | 0.66 | 0.96 | 0.9  |

Table II  
RECALL AND PRECISION FOR EVERY ACTIVITY FOR THE STATIC APPROACH.

As can be seen in tables II and III, recall is improved by the dynamic approach. With respect to precision, the dynamic approach also improves the results except for the short time activities *falling* and *jumping*.

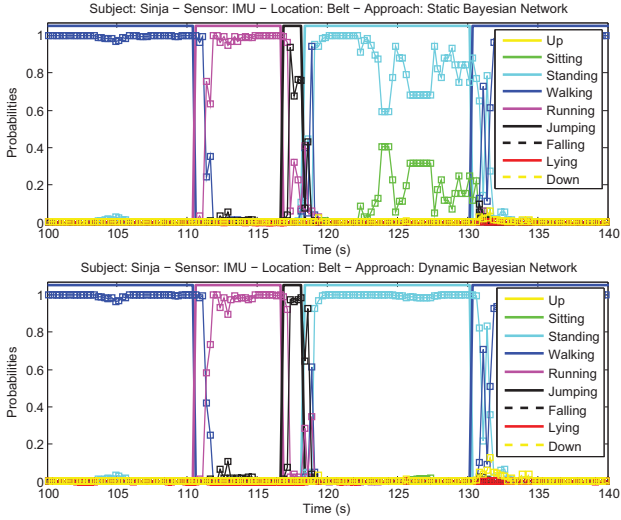


Figure 5. Evaluation results for the activities *standing*, *walking*, *running* and *jumping* using static (above) and dynamic (below) Bayesian inference. The top line represents ground truth, the curves below the estimated probabilities.

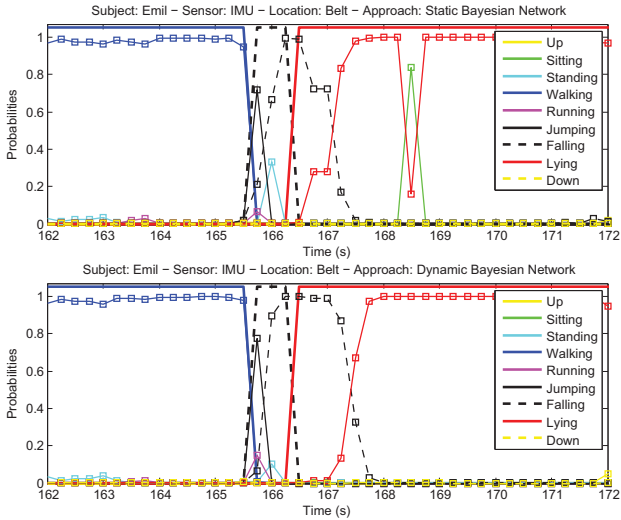


Figure 6. Evaluation results for the activities *walking*, *falling* and *lying* using static (above) and dynamic (below) Bayesian inference. The top line represents ground truth, the curves below the estimated probabilities.

|           | SIT  | STA  | WLK  | RUN  | JMP  | FAL  | LYG  | UpD  |
|-----------|------|------|------|------|------|------|------|------|
| Recall    | 0.97 | 0.91 | 0.99 | 0.85 | 0.7  | 0.75 | 0.96 | 0.3  |
| Precision | 0.89 | 0.92 | 0.88 | 0.88 | 0.66 | 0.5  | 0.97 | 0.77 |

Table III  
RECALL AND PRECISION FOR EVERY ACTIVITY FOR THE DYNAMIC APPROACH.

Even if precision decreased for *jumping* using the dynamic approach, recall improves significantly. Regarding *falling*, precision decreased due to the inertia given to this activity in the transition model. It can be corrected decreasing the

probability of *going on falling* in the transition probability model.

Worst results are obtained for the transitions *up or down* as they were not the objective of this work and were not regarded throughout the feature identification process, but only necessary now to improve the results of the dynamic filter. In the current configuration the dynamic approach gives worse results, which could be improved modifying the transition model probabilities.

### B. Inference Duration

Execution time (in milliseconds) for static and dynamic inference is shown in table IV. For comparison of the complexity, the table also gives the length of feature computation in our implementation. As it can be observed, the computation of the features is not time-consuming. Inference based on the network from Fig. 1 takes about 3 milliseconds for the multiplication of the conditional probabilities in the static BN (see section IV) and about 6 milliseconds for the Grid-based Filter (see section V). Though still very short, dealing with the HMM roughly doubles the inference time.

| Operation           | Execution time (ms) |
|---------------------|---------------------|
| Feature computation | 2.2 – 2.3           |
| Static inference    | 2.5 – 3.5           |
| Dynamic inference   | 5 – 6.5             |

Table IV  
EXECUTION TIME IN MILLISECONDS FOR FEATURE COMPUTATION, STATIC AND DYNAMIC INFERENCE.

As can be seen, the time and time difference for a single evaluation is almost negligible. For the decision, if to apply a dynamic or a static model depends rather on other factors like the inference frequency, but also the update frequency of the raw data (compare [19]). In our specific case, the sensor is sending with  $100Hz$ . Given rounded  $10ms$  processing time for data transport and parsing, feature computation, dynamic inference and eventual storage, this means that the CPU is loaded 100%. Even if we take into account our reduced inference frequency of  $4Hz$ , this considerable load of the CPU should only be performed if the application really consumes the data with this frequency.

If the data is only queried once in a while, on-demand inference would make more sense – which on the other hand then does not benefit from the last known state, i.e. the HMM. In this case static inference would be preferable, whereas dynamic inference is preferable for quality reasons, whenever activity is monitored continuously.

## VIII. CONCLUSIONS AND OUTLOOK

In this paper we have given an overview over context inference techniques with Bayesian methods for activity recognition. The comparison of applying dynamic or static inference to our use case 'activity recognition' has shown,

that the transition model used by the HMM helps improving the outcome considerably, already with a first-order HMM. On the other hand inference time has doubled, which is not a problem in the current case with all input RVs being observed.

Considering scheduling of inference, it was shown, that dynamic inference models should be used in the case of a continuous monitoring of a variable or high-frequent querying with important short-time activities that must not be missed. In the case of less frequent on-demand inference, a dynamic model does not add value, but cause computational costs and should be neglected.

A hierarchical model of inference depending on the information type seems to be an option worth considering. The closer information is to raw sensor data, the higher is the update rate and the need for inference (e.g. acceleration data, features at 100 Hz). A level higher (e.g. activity, as in our example) already needs lower frequencies, the shortest activity is in the range of 1 s or 1 Hz. The highest level of information, e.g. the current situation comprising a set of activities then already has much lower frequency of change. A normal duration may lie within minutes. These different durations imply different inference techniques and an hierarchical model to transfer results between levels where necessary.

#### ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 215098 of the *Persist* (PERSONAL SELF-IMPROVING SMART SPACES) Collaborative Project and from the Irish HEA through the PRTL cycle 4 project "Serving Society: Management of Future Communication Networks and Services". We also want to thank all persons helping to record the data set.

#### REFERENCES

- [1] Widyawan, M. Klepal, and S. Beauregard, "A backtracking particle filter for fusing building plans with PDR displacement estimates," in *Proceedings of the 5th Workshop on Positioning, Navigation and Communication, 2008, WPNC08*, Hannover, Germany, Mar. 2008.
- [2] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 09, no. 4, pp. 309–347, October 1992. [Online]. Available: <http://www.ingentaconnect.com/content/klu/mach/1992/00000009/00000004/00422779>
- [3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, 1988.
- [4] K. P. Baclawski, "Bayesian network development," in *International Workshop on Software Methodologies, Tools and Techniques, pages 18-48. Keynote address.*, Sept 2004.
- [5] D. Pennock, "Logarithmic time parallel bayesian inference," in *Proc. 14th Conf. Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1998, pp. 431–438.
- [6] E. Charniak, "Bayesian networks without tears," *AI Magazine*, vol. 12, no. 4, pp. 50–63, 1991.
- [7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 1995.
- [8] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *In the joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, 2002.
- [9] G. F. Cooper, "Probabilistic inference using belief networks is NP-hard," Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, CA, Tech. Rep. KSL-87-27, May 1990.
- [10] R. D. Shachter, "Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams)," in *Proceedings of the Fourteenth Conference in Uncertainty in Artificial Intelligence*, 1998, pp. 480–487. [Online]. Available: <http://jmvidal.cse.sc.edu/library/shachter98a.pdf>
- [11] P. Dagum and M. Luby, "Approximating probabilistic inference in bayesian belief networks is np-hard," *Artif. Intell.*, vol. 60, no. 1, pp. 141–153, 1993.
- [12] A. Garvey and V. Lesser, "A survey of research in deliberative real-time artificial intelligence," *Real-Time Syst.*, vol. 6, no. 3, pp. 317–347, 1994.
- [13] S. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of the Royal Statistical Society series B*, vol. 50, pp. 157–224, 1988.
- [14] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *International Journal of Approximate Reasoning*, vol. 15, no. 3, pp. 225–263, 1996.
- [15] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.
- [16] I. D. Coope and C. J. Price, "On the convergence of grid-based methods for unconstrained optimization," *SIAM J. on Optimization*, vol. 11, no. 4, pp. 859–869, 2000.
- [17] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.
- [18] B. Lev, "Book reviews," *Interfaces*, vol. 39, no. 1, pp. 91–96, 2009.
- [19] K. Frank, N. Kalatzis, I. Roussaki, and N. Liampotis, "Challenges for context management systems imposed by context inference," in *MUCS '09: Proceedings of the 6th international workshop on Managing ubiquitous communications and services*. New York, NY, USA: ACM, 2009, pp. 27–34.