DYNAMIC OBJECT-ORIENTED BAYESIAN NETWORKS FOR FLEXIBLE RESOURCE-AWARE CONTENT-BASED INDEXING OF MEDIA STREAMS

Ole-Christoffer Granmo, Frank Eliassen and Olav Lysne

Department of Informatics
University of Oslo
Gaustadalléen 23, Postbox 1080 Blindern
N-0316 Oslo, Norway
{olegr, frank, olavly}@ifi.uio.no

ABSTRACT

To effectively utilize the growing number of digital media sources in todays electronic world, easy to use computational flexible content-based access is a necessity. In contrast, content-based access must often be based on interpreting results from low-level computationally expensive feature extraction algorithms. We propose a new approach based on dynamic object-oriented Bayesian networks for encapsulating low-level feature extraction algorithms in integration with probabilistic interpretation models, within combinable high-level domain concepts. This approach opens up for content-based access in terms of high-level domain concepts at the end-user level. Furthermore, we suggest how the feature extraction algorithms can be activated on a per need basis (hypothesis driven activation). This opens up for trading off feature extraction resource usage against the reliability of the content-based access, which in turn may allow real-time content-based access in a greater range of processing environments.

1. INTRODUCTION

The technical ability to generate volumes of digital media data is becoming increasingly "main stream" in today's electronic world. To utilize the growing number of media sources, both the ease of use and the computational flexibility of methods for content-based access must be addressed; e.g. an end-user may want to access live content in terms of high-level domain concepts under a variety of processing environments ranging from complex distributed systems to single laptops.

In *content-based indexing* of media streams a media stream is segmented into scenes, and each scene is described or classified. The media stream content can then be accessed from the generated index. Recent work [3, 4, 11] demonstrate how Hidden Markov Models (HMMs) can be used for content-based indexing of video/media streams. By using HMMs for content-based indexing, a video/media stream can be segmented and classified in one step in contrast to previous approaches which segment the video/media stream before each segment is classified. This limits the latter methods mainly to video/media streams in which scenes are separated by hard-cuts [4].

A problem with HMMs is that the large number of parameters to be specified makes defining an indexing task very demanding. In fact, the number of parameters generally grows exponentially with the number of entities to be modeled, making even a learning based approach unmanageable. This may be a problem when indexing multiple interrelated media streams or tracking multiple objects.

Dynamic Bayesian networks (DBNs) [8] generalize HMMs by representing the hidden state (and observed states) in terms of state variables related in a directed acyclic graph (DAG). The DAG encodes conditional variable independencies, effectively reducing the number of parameters to be specified. Thus DBNs seem more appropriate than HMMs for specifying complex media indexing tasks.

Another problem governing previous approaches is that when indexing in real-time, they are limited to media features which can be extracted within the time frame of a media sample (video frame) as the features are extracted indiscriminately in every media sample. This severely restricts the set of feature extraction algorithms which can be used; e.g. face detection- and recognition algorithms may demand to much resources to be executed in every video frame.

In [7] it is suggested that extraction of image features in still image classification should be hypothesis driven such that image features are extracted only when the expected information gain relative to the extraction cost is significant. Such an approach supports activation of resource demanding feature extraction algorithms on a per need basis.

In this paper we introduce a new approach based on dynamic object-oriented Bayesian networks (DOOBNs) [1] addressing the identified problems. Firstly, the specification of media indexing tasks at the end-user level is simplified by

This research is supported by the Norwegian Research Council under grant no. 126103/431

- using object-orientation to encapsulate the low-level media features and the DBN parameters within *high-level domain concepts*,
- basing the specification of media indexing tasks on these high-level domain concepts.

Secondly, the activation of feature extraction algorithms is controlled by our novel real-time inference algorithm. The algorithm is based on the particle filter [10] and extends the approach suggested in [7] to DBNs/DOOBNs. Empirical results indicate that the inference algorithm allows

- hypothesis driven extraction of media features from media streams on an expected information gain/extraction cost basis,
- more flexible use of feature extraction algorithms under resource restrictions,
- real-time indexing.

The research is done in the context of the DMJ project [5] where the overall goal is to establish a framework supporting the creation of self-contained analysis units representing high-level combinable domain concepts, capable of executing under a variety of processing environments. The main focus of this paper is the novel inference algorithm for hypothesis driven feature extraction in DBNs/DOOBNs.

In section 2 we briefly describe how dynamic Bayesian networks can be used to index media streams. Then, in section 3 we introduce our DOOBN based approach to specifying media indexing tasks. In section 4 we present the hypothesis driven inference algorithm. Empirical results are presented in section 5. We conclude in section 6 and discuss further work.

2. DYNAMIC BAYESIAN NETWORKS FOR INDEXING OF MEDIA STREAMS

In this section we briefly describe how DBNs can be used to model and reason about high-level domain concepts, including their correspondence to low-level media features. We also describe how such a model can be used to index media streams.

As roughly sketched, a *DBN* is a graphical probabilistic model for compact description of causal relationships between the state of variables in a domain, and for effective inference about the state of some variables given the state of other variables. Qualitatively, a DBN consists of a directed acyclic graph (DAG) where each node represents a variable and a directed edge between two nodes represents a direct causal relationship. By the notation $P(X|Y)$ we mean a conditional probability distribution which assigns a probability to each possible state $X = x_i$ given each state $Y = y_j$. For each node Q the DAG is annotated with a conditional probability distribution $P(Q|R_1, \ldots, R_n)$ which describes the causal relationship between the variable Q and its parents $R_1, \ldots, R_n$ in the DAG quantitatively. When a node Q has no parents, the conditional probability distribution is reduced to an unconditional distribution, $P(Q)$. The annotated DAG is sectioned into a sequence of structurally and parametrically equivalent *time slices* representing the state at a particular point or interval in time. A DBN is called dynamic as it defines a dynamic (possibly infinite) sequence of time slices.

Fig. 1 gives an example of a DBN modeling the indexing task of detecting "when a *person* occurs in media stream '1' *within* the interval a *person* occurs in media stream '2"'. The variables $Person_1^t$, $Person_2^t$ and $Within^t$ in DBN

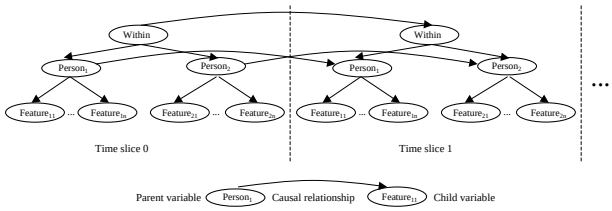

Fig. 1. A sequence of DBN time slices modeling a media indexing task

time slice t represent the three emphasized high-level domain concepts from the the previous sentence, in time interval t. The variables $Feature_{11}^t$, ..., $Feature_{1n}^t$, $Feature_{21}^t$, ..., $Feature_{2n}^t$ represent results from low-level feature extraction algorithms in the same interval. Without going into details, $Person_1^t$ and $Person_2^t$ can be in the states *{"yes", "no"}*) and $Within^t$ can be in the states *{"no person in media stream 1", "a person in media stream 1 and no person in media stream 2", "'a person both in media stream 1 and media stream 2' after 'a person in media stream 1 and no person in media stream 2"'}*).

The set of probability distributions relating the variables in two consecutive time slices is hereafter called the *transition model*. The transition model in Fig. 1 consists of $3 \times 3 + 2 \times 2 + 2 \times 2$ parameters; $P(Within^t|Within^{t-1})$, $P(Person_1^t|Person_1^{t-1})$ and $P(Person_2^t|Person_2^{t-1})$. In contrast, the corresponding transition model in a HMM requires $(3 \times 3) \times (2 \times 2) \times (2 \times 2)$ parameters; i.e. exponential growth in the number of variables as opposed to linear growth.

The set of probability distributions relating the state of the variables representing high-level domain concepts to the state of variables representing results from low-level feature extraction algorithms, within a given time slice, is called the *observation model*. In Fig. 1 this set corresponds to the probability distributions $P(Feature_{11}|\ Person_1)$, ..., $P(Feature_{1n}|Person_1)$, $P(Feature_{21}|\ Person_2)$, ..., $P(Feature_{2n}|\ Person_2)$, $P(Person_1|Within)$, and $P(Person_2|\ Within)$.

Based on the transition model and the observation model,

the *a posteriori distribution* of the variables representing high-level domain concepts given the state of the variables representing low-level media features (e.g. $P(Within^t | Feature^0_{11}, \ldots, Feature^t_{2n})$), can be inferred. The expected indexing error rate is minimized by indexing each time slice with the most probable state of each variable as given by the a posteriori variable distributions.

3. SPECIFICATION OF MEDIA INDEXING TASKS IN DOOBNS

Even if DBNs generally require fewer parameters than HMMs, it seems inappropriate to specify these parameters at the end-user level when defining media indexing tasks. Meaningful specification of the parameters require knowledge about low-level feature extraction algorithms as well as probabilistic modeling and inference. In contrast, an end-user may want to specify a media indexing task in terms of high-level domain concepts. E.g., an end-user may want to specify the task of detecting "when a person occurs in media stream '1' within the interval a person occurs in media stream '2"' in an abstract notation such as "Detect when $'\mathbf{Person}'_1$ occurs $'\mathbf{Within}'$ $'\mathbf{Person}'_2$ occurs". However, it is necessary to interpret extracted low-level media features such as color histograms and picture differences in order to solve the given task. In this section we present an approach for encapsulating the low-level media features and the DBN parameters within *high-level domain concepts*, such that a media indexing task can be specified simply by relating the high-level domain concepts. The approach is based on the framework of dynamic object-oriented Bayesian networks (DOOBNs).

3.1. Dynamic object-oriented Bayesian networks

A *DOOBN* as described in [1] can be seen as a hierarchical top-down specification of a DBN. Each *DOOBN time slice* consists of a set of *DBN objects* with *input/output* variables and a set of *references* relating each input variable to at most one output variable in the *current* or *preceding* time slice. In short, an input variable act as a placeholder for an output variable. The *internals* of a DBN object define how the state of the input variables influences the state of the output variables in terms of a *DBN time slice part* or another set of related DBN objects. Note that an input variable in a DBN time slice part cannot have parents within that part. As in ordinary DBNs cycles are not allowed.

Fig. 2 gives an example of a DOOBN time slice consisting of three DBN objects where each object encapsulates a part of a DBN time slice. Note that some of the edges have been reversed in the "$Within$"-object compared to the DBN in Fig. 1. This changes the semantics slightly and is necessary to accommodate for the input variable restrictions of the DOOBN framework. The input variables "Event 1" and "Event 2" of the "$Within$"-object act as placeholders for the output variables taking part in the within relation. The input variables having no reference from an output vari-

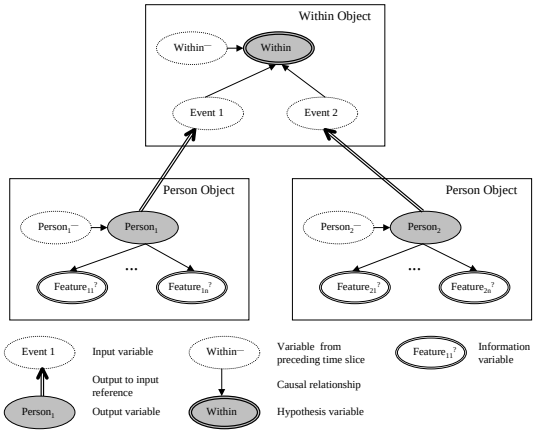

Fig. 2. A DOOBN time slice specifying a media indexing task

able, marked with a minus sign ('-') in Fig. 2, relate the state of the given DOOBN time slice to the state of the previous DOOBN time slice. Hence, a chain of time slices specifying a dynamic process can be created as illustrated in Fig. 3. Note that some of the variable types in Fig. 2 and 3 will

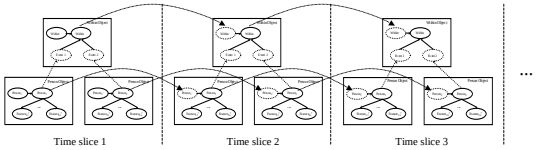

Fig. 3. A DOOBN constructed by chaining the time slice from Fig. 2

be explained later.

3.2. Specification of media indexing tasks

Our approach is to represent high-level domain concepts for specifying media indexing tasks as a set of *DBN object classes* from which DBN objects can be instantiated. Input variables represent possible concept contexts, output variables represent the concept characteristics, and the object internals encode how the characteristics of a concept depend on its context and how the concept can be recognized from extracted media features. The concept 'Within' may for example be represented as a DBN object class (see Fig. 2) with three input variables representing the concept state in the previous time slice and the state of the two concepts (events) taking part in the within relation. The output variable represents the state of the concept in the current time slice (e.g.*{"not event 1", "event 1 and not event 2", "'event*

1 and event 2' after 'event 1 and not event 2"'}). The object internals encode a stochastic state machine capable of recognizing the within relation.

The main advantage of this approach is that the specification of media indexing tasks can be performed in two steps:

- A domain expert specifies manually or by training appropriate domain specific DBN object classes which encapsulates the feature extraction algorithms as well as the DBN parameters.
- An end-user specifies media indexing tasks by simply relating the input/output variables of instances of the DBN object classes.

For example, the task of detecting "when a person occurs in media stream '1' within the interval a person occurs in media stream '2"' may be specified as outlined in Fig. 2.

In addition to standard DBN variables, the specification also includes *information variables* representing extracted media features, and *hypothesis variables* representing the content to be detected. The role of these variables are discussed in more detail in the next section.

4. PARTICLE FILTERS FOR HYPOTHESIS DRIVEN INDEXING OF MEDIA STREAMS

There exists many algorithms for exact and approximate inference in DBNs/DOOBNs [2, 8, 10]. Particle filtering (PF) [10] is a technique for *real-time* approximate filtering in which the *a posteriori* distribution is approximated with a set of weighted samples ("particles"). In our context the technique can be used to approximate the a posteriori distribution of the variables in a DOOBN time slice given the state of past and current information variables. In this section we first describe a standard PF algorithm. Then we suggest how the PF algorithm can be extended in order to support hypothesis driven extraction of media features from media streams on an expected information gain/extraction cost basis.

4.1. A particle filter algorithm

Let I^t denote the information variables, X^t all other variables, and o^t the observations $I^t = i^t$ currently made in time slice t. Furthermore, let all observations up to and including time slice t be denoted by O^t. Finally, let s_j^t denote a state sample $X^t = x_j^t$ and w_j^t the weight (relevance) of this sample.

The PF can be used to obtain a set of weighted samples $[s_1^t, w_1^t], \ldots, [s_n^t, w_n^t]$ which approximates $P(X^t|O^t)$ - the a posteriori probability distribution of X^t given the observed states O^t - from

- a set of weighted samples $[s_1^{t-1}, w_1^{t-1}], \ldots, [s_n^{t-1}, w_n^{t-1}]$ approximating $P(X^{t-1}|O^{t-1})$,
- the state transition model $P_{tm}(X^t|X^{t-1})$, including $P(X^0)$, defined in the DOOBN,
- the observation model $P_{om}(I^t|X^t)$, also defined in the DOOBN,
- the observations o^t.

The algorithm in Fig. 4 implements a PF. A weighted sample set $[s_1^t, w_1^t], \ldots, [s_n^t, w_n^t]$ approximating $P(X^t|O^{t-1})$ is the starting point for each iteration t. First the state of the information variables, $o^t \leftarrow I^t = i^t$, is observed. Then the weight w_j^t of each sample is updated by multiplying w_j^t with the conditional probability $P_{om}(o^t|s_j^t)$ of observing o^t given the sampled state s_j^t. The updated sample set can be used to approximate $P(X^t|O^t)$ by normalizing the weights of the set. Let $\hat{P}^t \sim [s_1^t, w_1^t], \ldots, [s_n^t, w_n^t]$ denote this normalization and approximation operation.

Before moving on to the next time slice $(t+1)$ a new sample set is drawn randomly (with replacement) from the set of samples $[s_1^t, w_1^t], \ldots, [s_n^t, w_n^t]$ according to the sample weights. Generally the new sample set will provide a better starting point for approximating $P(X^{t+1}|O^{t+1})$ as samples representing less likely scenarios are equivalently

---

Procedure DBN_PF(n)

% Initalization.
for $j = 1, \ldots, n$
$s_j^0 \leftarrow$ **sample from** $P(X^0)$;
$w_j^0 \leftarrow 1$;
for $t = 0, \ldots$
% Processing of features in time slice t.
$i^t \leftarrow \mathbf{observation}(I^t)$;
$o^t \leftarrow I^t = i^t$;
for $j = 1, \ldots, n$
$w_j^t \leftarrow w_j^t \times P_{om}(o^t|s_j^t)$;
$\hat{P}^t \sim [s_1^t, w_1^t], \ldots, [s_n^t, w_n^t]$;
% Processing of time slice t finished.
% Preparations for next time slice.
for $j = 1, \ldots, n$
$[s_j^t, w_j^t] \leftarrow$ **draw from**
$([s_1^t, w_1^t], \ldots, [s_n^t, w_n^t])$;
$s_j^{t+1} \leftarrow$ **sample from** $P_{tm}(X^{t+1}|s_j^t)$;
$w_j^{t+1} \leftarrow w_j^t$;

Fig. 4. Particle filter algorithm

less likely to be included in the sample set. Thus the samples are concentrated on the likely scenarios and will not spread out across an exponential large set of possible but very unlikely scenarios [10]. For each sample s_j^t in this refined sample set, a sample s_j^{t+1} for time slice $t+1$ is generated based on the transition model $P_{tm}(X^{t+1}|s_j^t)$. The new sample set can be used to approximate $P(X^{t+1}|O^t)$.

4.2. The hypothesis driven particle filter algorithm

We have extended the PF algorithm to support hypothesis driven extraction of media features in DBNs/DOOBNs on an expected information gain/extraction cost basis. The algorithm is shown in Fig. 5.

Essentially, the algorithm makes observations sequentially within each time slice until the observation cost outweighs the expected information gain. The observation with the largest expected information gain/extraction cost ratio is always made first.

Let $H^t \subseteq X^t$ denote the hypothesis variables, V^t an information variable in I^t, and $i_j^t[V^t = v]$ a sampled state of I^t where the state of variable V^t is set to v. Furthermore, recall that o^t denotes the observations currently made in time slice t. We use the *expected hypothesis distribution energy*

$$\sum_{\substack{h \in H^t \\ v \in V^t}} P(H^t = h|V^t = v, O^{t-1}, o^t)^2 P(V^t = v|O^{t-1}, o^t)$$

as a measurement to establish the expected information gain of information variable V^t. Let $\#H$ be the number of possible hypothesis states. The expected hypothesis distribution energy ranges from 1 when the state of the hypothesis variables is certain to $1/\#H$ when the hypothesis distribution is uniform.

In order to estimate the expected hypothesis distribution energy, we approximate the a posteriori probability distribution $P(X^t, I^t|O^{t-1}, o^t)$. This distribution relates the state of the information variables to the state of the hypothesis variables. Thus, in contrast to the algorithm in Fig. 4, the information variables are also sampled.

For each time slice the goto loop in the algorithm

- estimates the current hypothesis distribution energy $e_{current}$ based on the approximation of $P(X^t, I^t|O^{t-1}, o^t)$ denoted by $\hat{P}^t$.
- estimates the expected hypothesis distribution energy e_{V^t} for each information variable V^t. The variable with the largest expected energy divided by the extraction cost is denoted V_{max}^t.
- observes the information variable V_{max}^t and updates the weighted sample set with the observed state, if the expected increase in hypothesis distribution energy, $e_{V_{max}^t} - e_{current}$, divided by the extraction cost $c_{V_{max}^t}$ is larger than the given $threshold$.

Procedure DBN_HDPF($n, threshold$)

% Initalization.

for $j = 1, \ldots, n$

$(s_j^0, i_j^0) \leftarrow$ **sample from** $P(X^0, I^0)$;

$w_j^0 \leftarrow 1$;

for $t = 0, \ldots$

% Sequential processing of features in time slice t.

loop: $\hat{P}^t \sim [(s_1^t, i_1^t), w_1^t], \ldots, [(s_n^t, i_n^t), w_n^t]$;

$e_{current} \leftarrow \sum_{h \in H^t} \hat{P}^t(H^t = h)^2$;

% Identifies "best" feature.

for each $V^t \in I^t$

$e_{V^t} \leftarrow \sum_{h \in H^t, v \in V^t} \hat{P}^t(H^t = h| V^t = v)^2 \hat{P}^t(V^t = v)$;

$V_{max}^t = \mathbf{maxarg}_{V^t \in I^t} \frac{e_{V^t}}{c_{V^t}}$;

% Feature observed if "significant".

if $\frac{e_{V_{max}^t} - e_{current}}{c_{V_{max}^t}} > threshold$ **then**

$v_{max} \leftarrow \mathbf{observation}(V_{max}^t)$;

for $j = 1, \ldots, n$

$w_j^t \leftarrow w_j^t \times P_{om}(V_{max}^t = v_{max}|s_j^t, o^t)$;

$i_j^t \leftarrow i_j^t[V_{max}^t = v_{max}]$;

$o^t \leftarrow o^t \cup V_{max}^t = v_{max}$;

goto loop;

% Processing of time slice t finished.

% Preparations for next time slice.

for $j = 1, \ldots, n$

$[(s_j^t, i_j^t), w_j^t] \leftarrow$ **draw from** $([(s_1^t, i_1^t), w_1^t], \ldots, [(s_n^t, i_n^t), w_n^t])$;

$(s_j^{t+1}, i_j^{t+1}) \leftarrow$ **sample from** $P_{tm}(X^{t+1}, I^{t+1}|s_j^t, i_j^t)$;

$w_j^{t+1} \leftarrow w_j^t$;

Fig. 5. Hypothesis driven particle filter algorithm

This procedure is repeated in the given time slice until the expected increase in the hypothesis distribution energy falls below the given threshold.

5. EMPIRICAL RESULTS

Our initial experiments indicate that we can construct media indexing tasks from high-level domain concepts represented as DBN object classes, and then trade off media indexing reliability against media indexing resource usage by varying the introduced threshold. The experiments were conducted by simulating the media indexing task specified in Fig. 3 with one cheap and one expensive feature extraction algorithm for each media stream. The expensive feature extraction algorithm is three times as accurate as the cheap algorithm and uses twice the resources. The simulations were run across 300 000 time slices and the detection error rate (number of false negatives/positives divided by the total number of detections) and relative media indexing cost (amount of expended resources divided by the maximum amount of resources possible to spend) were estimated for various threshold values with our extended particle filter. Fig. 6 reports the found correspondence between detection error rate and relative media indexing cost. Each point represents the detection error rate and the relative cost for a given threshold. The strategy of classifying each time

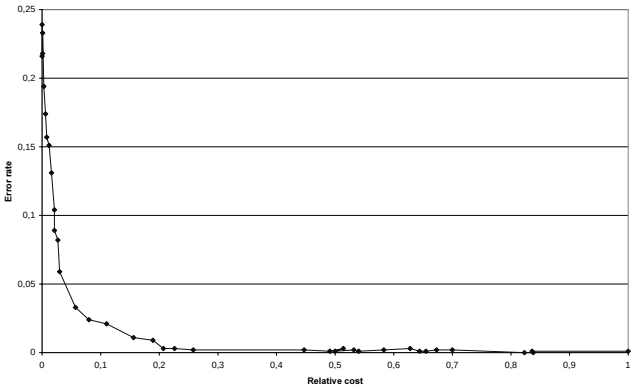

Fig. 6. Relative cost and error rate for different thresholds

slice as the a priori most probable event (cost 0) resulted in an error rate of 0.25. The strategy of executing the media processing algorithms continuously achieved an error rate of 0.001. As seen, the extended particle filter was able to achieve a similar error rate by using only 25% of the resources.

To examine the activation pattern of each feature extraction algorithm for different relative cost levels, we plotted the actual state of the 'Within'-object output variable, the tracked state of the output variable and the activated feature extraction algorithms across 30 000 time slices. As outlined in section 3 the output variable has three states:

State 1 represents the event "no person in media stream 1",

State 2 represents the event "a person in media stream 1 and no person in media stream 2",

State 3 represents the event "a person both in media stream 1 and media stream 2 after 'State 2"'.

Fig. 7 displays the plot generated at relative cost 0.856. As seen in this plot, both the expensive and the cheap algorithms are used extensively which results in an accurate track (i.e. low detection error rate).

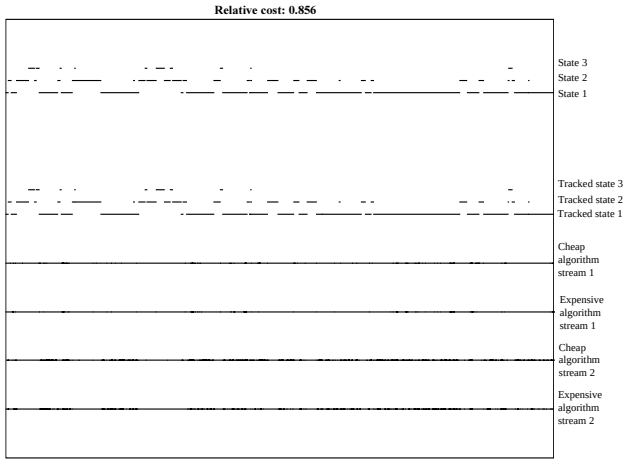

Fig. 7. Tracking simulation 1, relative cost = 0.856

When reducing the relative cost to 0.474, as seen in Fig. 8, the feature extraction algorithms are only activated on stream '2' when the extended filter is tracking state 2 and state 3. This can be explained by the semantics of the within

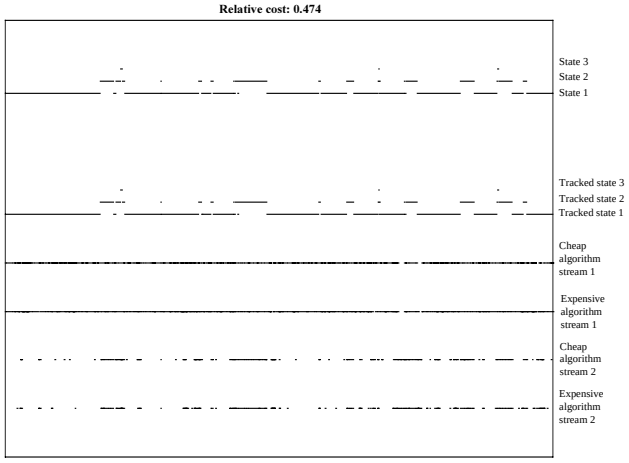

Fig. 8. Tracking simulation 2, relative cost = 0.474

relation; as long as no person is in media stream '1', it is not very significant whether there is a person in stream '2'.

When reducing the relative cost further to 0.194, as displayed in Fig. 9, the activation of the expensive feature

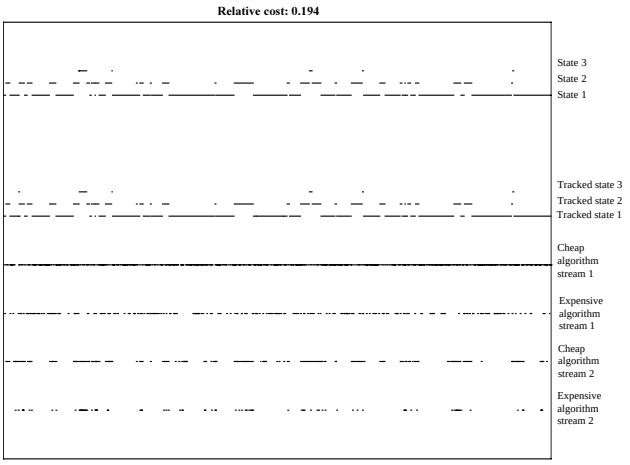

Fig. 9. Tracking simulation 3, relative cost = 0.194

extraction algorithms is reduced to a minimum. Still, the track is quite accurate, although the segment boundaries drift slightly.

Finally, a further reduction of relative cost to 0.104 is achieved by only activating the cheap feature extraction algorithms sporadically, as seen in Fig. 10. At this resource

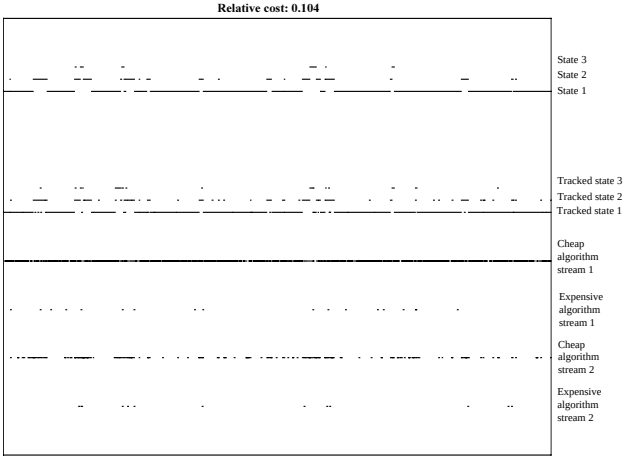

Fig. 10. Tracking simulation 4, relative cost = 0.104

consumption level, a significant number of false positive and false negative classifications occurs, in addition to the increased segmentation border drift. This degradation continues until every segment is classified as the most common event at detection error rate 0.25.

Given a media stream of 25 samples per second the simulated media indexing tasks executed on average 20 times faster than real-time (500 time slices per second) on a Pentium III 650Mhz laptop computer. This excludes the execution time of the feature extraction algorithms.

6. CONCLUSIONS AND FURTHER WORK

In this paper we have presented an approach for specification of media indexing tasks based on DOOBNs in integration with a real-time resource-aware inference method based on the particle filter. Preliminary results indicate that our approach supports the creation of self-contained indexing units which

- can be combined to create complex media indexing tasks,
- are capable of executing under a variety of resource consumption levels.

Thus our approach addresses both the ease of use and the computational flexibility which is necessary to make content-based indexing of media streams a generic tool. Although our experiments are based on a DOOBN with only three objects, recent work such as [9] show that the particle filter algorithm scales well. It is reasonable to believe that this also applies to our extended particle filter algorithm.

A number of areas for further research have been identified. Roughly stated a DBN integrates Bayesian networks [6] and finite-state machines. We are investigating how more expressive languages, such as context-free grammars, can be integrated with Bayesian networks to achieve more flexible specification of media indexing tasks. We are also investigating approaches for effective creation and tailoring of the DBN object classes. This includes the construction of a high-level domain concept library. Furthermore, we are working on a framework for distributed processing and control of the DOOBNs and the feature extraction algorithms under resource restrictions. The status of this work is described in [5]. In that context, adaption to changing resource restrictions is an issue. Thus, an effective approach for mapping the appropriate resource consumption level to the threshold used in the extended particle filter algorithm must be found. We are also extending the resource model and the greedy algorithm used in section 4 to handle a distributed processing environment by addressing the aspects of communication delay and multiple processing resources.

7. REFERENCES

[1] O. Bangsø et al. Top-down construction and repetitive structures representation in bayesian networks. In *FLAIRS'00*, pages 282–286. AAAI Press, 2000.

[2] X. Boyen et al. Tractable inference for complex stochastic processes. In *UAI'98*. Morgan Kaufmann, 1998.

[3] S. Boykin et al. Machine learning of event segmentation for news on demand. *Communications of the ACM*, 43(2):35–41, Feb 2000.

[4] S. Eickeler et al. Content-based video indexing of tv broadcast news using hidden markov models. In *Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 2997–3000. IEEE, 1999.

[5] V. S. W. Eide et al. Distributed journaling of distributed media. In *NIK'2000*. Tapir, 2000.

[6] F. V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

[7] F. V. Jensen et al. Bayesian methods for interpretation and control in multi-agent vision systems. In *Applications of Artificial Intelligence X: Machine Vision and Robotics*. SPIE, 1992.

[8] U. Kjærulff. dHugin: a computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11(1):89–113, 1995.

[9] J. Kwon and K. Murphy. Modeling Freeway Traffic using Coupled HMMs. Technical report, Department of Computer Science, U. C. Berkeley, 2000.

[10] J. Liu et al. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93:1032–44, 1998.

[11] M. R. Naphade et al. Probabilistic multimedia objects (multijects): A novel approach to video indexing and retrieval in multimedia systems. In *ICIP'98*, pages 536–540. IEEE, 1998.