Distributed Journaling of Distributed Media

Viktor S. Wold Eide, Frank Eliassen,
Ole-Christoffer Granmo and Olav Lysne *
Department of Informatics
University of Oslo
Gaustadalléen 23
P.O. Box 1080 Blindern
0314 Oslo, Norway.

Abstract

There are many challenges in devising solutions for on-line automatic content analysis, indexing and annotation of live networked multimedia sessions. These include content analysis under uncertainty (evidence of content are missed or hallucinated), the computational complexity of feature extraction and object recognition, and the massive amount of data to be analysed under real-time requirements.

In this paper we propose an architectural framework for on-line media content analysis targeted as an approach to the above challenges. This includes an approach for formulating and transforming high-level media content queries into distributable configurations of media content analysis algorithms. Our framework allows the system to guide the user in trading off between the reliability and latency of the content analysis. The uniqueness of our architecture lies in the combination of probabilistic knowledge-based media content analysis with QoS and distributed resource management to handle real-time requirements. This is all packaged into a generic framework that is tailorable to specific application domains.

1 Introduction

In this paper we introduce "media journaling" as a blanket term to refer to a broad class of applications in which users have the capability to capture multimedia content and related information of real-world activities or sessions as they happen, as well as the ability to process the media so as to create a segmented synchronized multimedia record of events. Media journaling will add value to applications such as video conferencing, distributed/virtual class rooms, telemedical applications, and monitoring and surveillance systems such as traffic surveillance.

The focus of our research is the development of a framework for on-line (real-time) processing of networked multimedia sessions for the purpose of indexing and annotating the data being analysed. An advantage of a carefully designed framework is that new subtechnologies of relevance for this kind of task can be plugged into the framework as they become available.

*Authors are listed alphabetically.

Issues of media journaling are similar to those found in the area of content-based multimedia indexing and retrieval. We may think of a user specification of journaling requirements as a complex content query specification. However, rather than performing the content analysis off-line on stored data, on-line analysis requires real-time processing of (live) networked sessions.

The two common approaches to the problem of content-based multimedia indexing and retrieval has been attribute-based indexing and retrieval of multimedia content, and a combination of feature extraction and object recognition. In comparison to the former, the latter has the advantage that it does not require manual entry of keyword descriptions which is a very time consuming operation. On the other hand the automatic indexing and annotation approach is computationally complex and only works satisfactory limited to specific domains.

The above reasoning suggests that a knowledge-based approach combined with open distributed processing technology, might be a promising approach to the challenges of media journaling. First of all, for feature extraction/object recognition to work well, specific domain knowledge is needed during analysis. Secondly, a distributed solution is required to cope with the computational complexity of feature extraction and object recognition, the massive amount of data to be analyzed in real time, and the scalability of the system with respect to the complexity of the session to be journaled and the content to be detected.

One promising approach to integrating domain knowledge (semantics of user domains) into the content-analysis process is to combine (in domain specific ways) low-level quantitative content querying (i.e. query by colour histogram, texture, pitch, etc.) into higher-level qualitative content querying. Furthermore, a probabilistic approach to detection and recognition of content is needed, since evidence of content extracted from media streams can be either missed or hallucinated (false positives) by the involved algorithms.

Our approach to the above challenges is to build upon the methodology of dynamic object-oriented Bayesian networks (DOOBNs) [6] [3]. In DOOBN, uncertainty can be handled explicitly and domain concepts may be effectivly represented as dynamic Bayesian network objects. Furthermore, we believe that a framework should be continuously adaptable and scalable to the available resources. Such an approach allows the system to guide the user in trading between the reliability and latency of the content analysis. This requires that a framework for on-line content analysis must be resource aware based on an open distributed resource model.

In this paper we propose an architecture for a distributed journaling framework, and an approach for formulating and transforming high-level media content queries into distributable configurations of media content analysis algorithms. The contribution of the architectural framework lies in the unique combination of probabilistic knowledge-based media content analysis with QoS and resource awareness to handle real-time requirements.

The rest of the paper is organized as follows. In Section 2 we present an overview of the architecture of our distributed media journaling framework. In Section 3 we describe the requirements of content analysis in media journaling, and show how these requirements can be met by dynamic object-oriented Bayesian networks. Section 4 discusses architectural requirements to on-line media content analysis and suggests a corresponding architecture for the distributed content analysis. This section also discusses requirements to resource management. In Section 5 we discuss some related work, while we in Section

6 offer some conclusions and outlook to future work.

2 Architectural overview

Figure 1 shows an architectural model of a generic distributed media journaling system. The components, shaded gray, represent the architectural focus of our work. The architecture is generic in the sence that it makes minimal assumptions about the networked sessions to be journaled and the nature of the content to be detected, indexed and annotated.

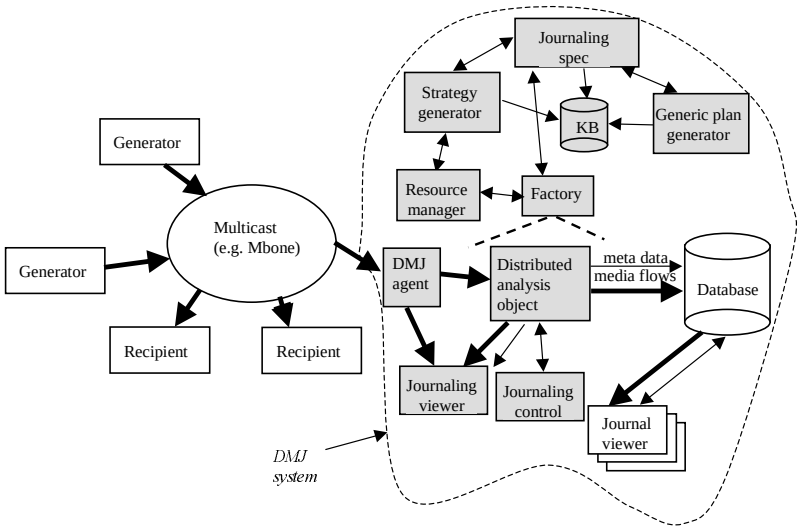

Figure 1: *Coarse architecture of media journaling.*

A DMJ system supports the capture, indexing, annotation, storage and retrieval of distributed networked multimedia sessions such as the media streams of a video conference or a surveillance system. Media input to the DMJ system is provided by DMJ agents that are responsible for capturing a media stream of the networked session to be journaled.

The journaling specification component validates abstract content queries specifying journaling tasks, and validates queries syntactically and semantically. The generic plan generator translates the user content query into a configuration of media processing functions according to a hierarchical plan. The hierarchical plan is generated from the abstract user request and from domain knowledge in the knowledge-base. A media processing function denotes a certain class of algorithms such as edge detection, for which there may exist many different implementations (i.e. algorithms with different cost and reliability characteristics).

The strategy generator investigates alternative media processing strategies that can be applied to a given hierarchical plan. The goal is to find an optimal strategy that trades off the qualitative and quantitative requirements of the user against the available resources of the distributed processing enviroment (DPE). This may involve direct negotiating with the user for selecting which strategy to use when there are several possibilites. The knowledge base (KB) defines a set of domain profiles. Domain knowledge is essentially information about combining low-level quantitative analysis into higher-level content recognition.

The resource manager manages the computing and communication resources of the DPE. Depending on the particular resource management services of the DPE, it may

provide dynamic load information to the strategy generator and the distributed analysis object, make resource reservations on behalf of the strategy generator and make resource allocations on behalf of the factory. The factory object takes as input a media processing strategy representing a user request and instantiates with the help of the distributed resource management systems, a distributed analysis object that is deployed in the DPE as a collection of interacting software components. The distributed analysis object is responsible for analyzing the media streams for the purpose of detecting and annotating content as specified by the user.

The journaling control component performs the run-time management of the journaling process. This includes actions for adapting the configuration of media processing algorithms in response to changing user requirements and/or resource availability. The journaling viewer component is a tool for viewing the journal in real-time, including presentation of content annotations and associated content. Other components are a database for storage and retrieval of recorded sessions, and rendering components for off-line viewing of media journals.

3 Content analysis

In order to make DMJ a generic tool, the notion of domain semantics as well as the aspects of performance must be addressed. At the *generic plan generation* level, an end-user may prefer to specify media analysis tasks in terms of high-level domain concepts. At the *strategy generation* level, it may be necessary to generate strategies for executing analysis tasks under a variety of resource usage- and reliability demands.

In this section we present an approach for handling the above challenges based on dynamic object-oriented Bayesian networks (DOOBNs), a generalization of the hidden Markov model (HMM). We also present preliminary empirical results indicating the suitability of our approach.

3.1 Dynamic object-oriented Bayesian networks

Over the past decade, dynamic Bayesian networks (DBNs) [8] have established themselves as an effective and principled framework for knowledge representation and reasoning under uncertainty in dynamic systems. While initially inadequate for representing large and complex domains, recent work introducing dynamic object-oriented Bayesian networks (DOOBNs) [6] [3] have to a large degree alleviated this.

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 casual relationship. For each node v the DAG is annotated with a conditional probability table $P(v|w_1, \ldots, w_n)$ which describes the causal relationship between v and its parents $w_1, \ldots w_n$ quantitatively. This annotated DAG is sectioned into structurally and parametrically equivalent *time slices* representing the state at a particular point or interval in time.

A *DOOBN* as described in [3] 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. The *internals* of a DBN object defines how the

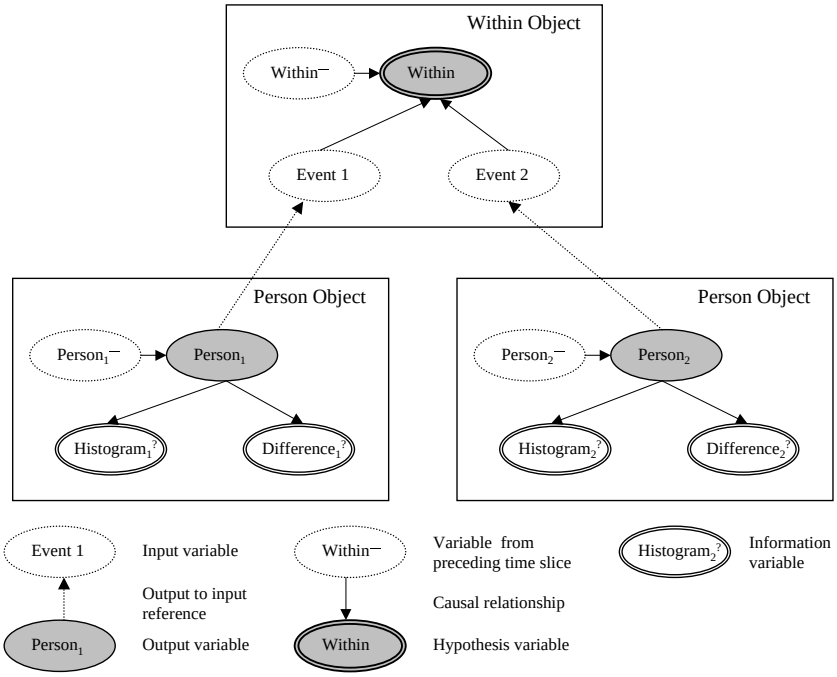

Figure 2: *A DOOBN time slice specifying a media analysis task.*

state of the input variables influences the state of the output variables in terms of a DBN time slice part with input/output variables or another set of related DBN objects. As in ordinary DBNs cycles are not allowed. Figure 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. The input variables having no reference from an output variable, marked with a minus sign ('-') in figure 2, relate the state of the given DBN time slice to the state of the previous DBN time slice. Hence, a chain of time slices specifying a dynamic process can be created as illustrated in figure 3. Note that some of the variable types in figure 2

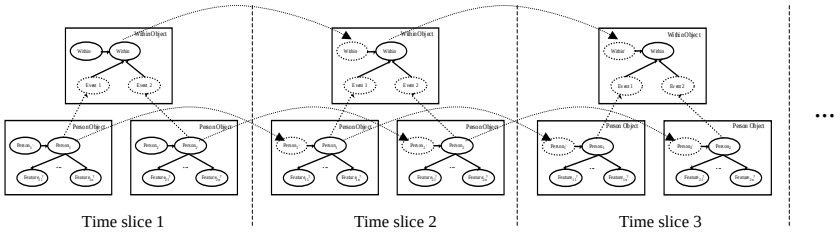

Figure 3: *A DOOBN constructed by chaining the time slice from figure 3.*

and 3 will be explained later.

3.2 Specification of media content analysis tasks

Even if DBNs generally require fewer parameters than HMMs, it seems inappropriate to specify these parameters at the end-user level when defining media analysis tasks. Mean-

ingful specification of the parameters require knowledge about low-level quantitative media processing algorithms as well as probabilistic modeling and inference. In contrast an end-user may want to specify a media analysis 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 video stream '1' within the interval a person occurs in video stream '2"' in an abstract notation such as "$Detect\ when\ '\mathbf{Person}_1'\ occurs\ 'Within'\ 'Person}_2'\ occurs$", when it in fact is necessary to interpret results from media processing algorithms such as color histograms and picture differences.

Our approach to the above problem is to represent high-level domain concepts 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 media processing algorithm results. The concept 'Within' may for example be represented as a DBN object class (see figure 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. *{"no events", "event 1", "event 2", "'event 1 and event 2' after event 1", "event 1 after 'event 1 and 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 analysis tasks can be performed in two steps:

- A domain expert specifies manually or by training appropriate domain specific DBN object classes and stores them in the DMJ *Knowledge base*. The classes encapsulates the media processing algorithms as well as the DBN parameters.
- An end-user specifies media analysis 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 video stream '1' within the interval a person occurs in video stream '2"' may be specified as outlined in figure 2. In addition to standard DBN variables the specification also includes information variables representing results from media processing algorithms and hypothesis variables representing the content to be detected.

We are currently investigating the flexibility of the object-oriented framework described in [3] when applied to parameter-free construction of DBNs from general building blocks.

3.3 Media processing strategies

For each time slice, results from low-level quantitative media processing algorithms are fed into the DOOBN and based on these results the most likely state of the hypothesis variables are estimated. By executing the media processing algorithms in every time slice in every media stream, optimal media analysis reliability is achieved given the media processing algorithms at hand. Unfortunately this brute force approach may require a lot of resources. On the other hand, by not executing any media processing algorithm at all a minimum of resources is used, but obviously the media analysis will be useless. Often more sophisticated media processing strategies for trading off analysis reliability against resource usage can be found.

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 image processing algorithms on a per need basis.

We have combined the *particle filter* [9] and the approach suggested in [7] to support real-time hypothesis driven extraction of media features in continuous media streams. By executing the media processing algorithms within a time slice sequentially, the results from already executed media processing algorithms can be used to determine which algorithm to execute next. In a greedy approach, the algorithm which maximizes the expected information gain/execution cost ratio is chosen. In order to save resources, our extended particle filter moves on to the next time slice when the expected information gain/execution cost ratio in the current time slice falls below a given threshold.

The above strategy implicitly defines a classification tree for each time slice. A *classification tree* is a tree-structured classifier in which the data to be classified is passed down the tree from the root node. At each interior node a given feature of the data is evaluated to determine which child node to visit. When the data is passed to a leaf node it has been classified [12]. Figure 4 gives an example of a classification tree for a time slice from figure 3. The media features to be extracted are interior nodes and a leaf node identi-

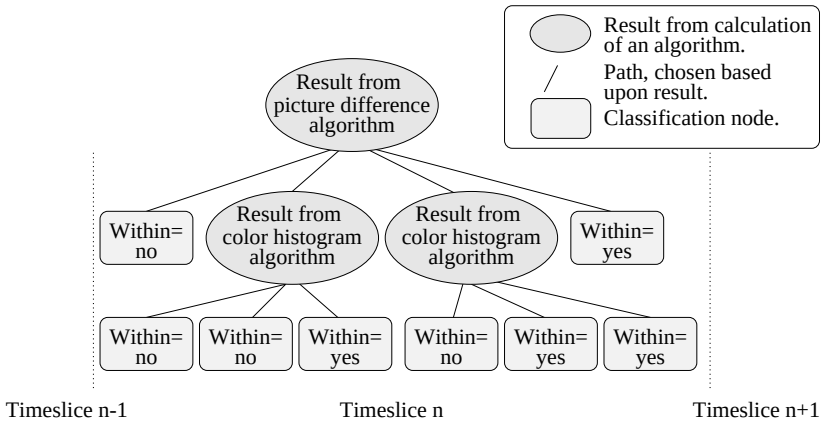

Figure 4: *A classification tree representing a media processing strategy.*

fies the most likely state of the hypothesis variables given the extracted media features represented by the branch.

Our initial experiment indicates that we can trade off analysis reliability against analysis resource usage by varying the introduced threshold. The experiment was conducted by simulating the process defined by the DOOBN in figure 3 across 300 000 time slices and estimating the detection error rate and relative analysis cost with our extended particle filter for various threshold values. Figure 5 reports the found correspondence between detection error rate (number of false negatives/positives divided by the total number of detections) and relative analysis cost (amount of expended resources divided by the maximum amount of resources possible to spend). The strategy of classifying each time slice as the most common 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.

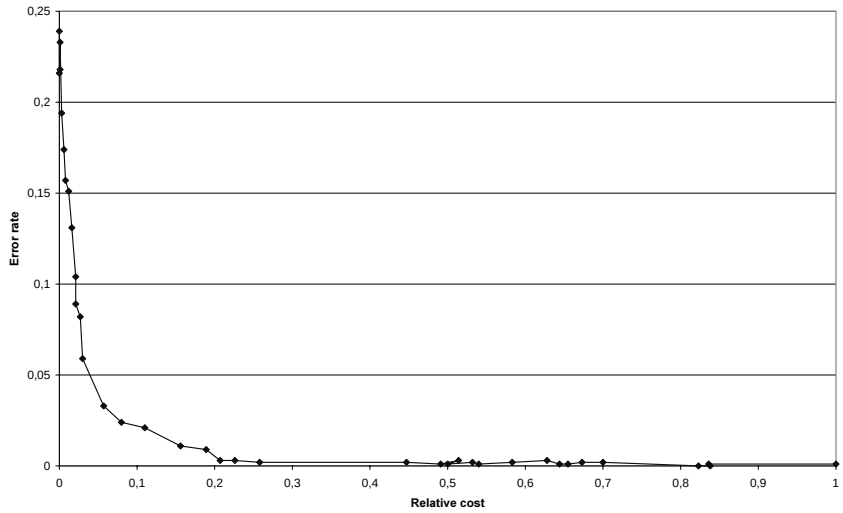

Figure 5: *Relative cost and error rate of different media processing strategies.*

Given a media stream of 25 samples per second the analysis task executed on average 20 times faster than real-time (500 time slices per second).

In the next section we describe how this approach to generic plan and strategy generation relate to the *Distributed analysis object*.

4 Distributed analysis object

As stated earlier, a distributed solution is required to handle the computational complexity of DOOBNs and low level quantitative media processing algorithms in a scalable way.

4.1 Units of distribution

As explained in the previous section and visualized in figure 4, a DOOBN during each time slice implicitly builds a classification tree used for analysis of results. This activity we may associate with a component which may execute on any host. We hereafter refer to such a component as a *composite detector, CD*, a component composing results. For the classification task, a CD hence requires results calculated by other components, possibly executing on other hosts in the distributed environment. We also define another kind of component, the *primitive detector, PD*, which operates directly on a media stream and corresponds to a low-level quantitative media processing algorithm, such as color histogram or picture difference calculation. The result from a PD consists of certain quantitative characteristics calculated from one or more media samples, such as when a picture difference exceeds a certain threshold value. Thus a CD consumes results produced by other PDs and/or CDs and produces its own result whenever certain conditions on the consumed results are satisfied.

The arrangement of PDs and CDs form a hierarchy where each level uses the results produced at the level below. The hierarchy is organized corresponding to the media processing strategy of the journaling task, where each component encapsulates content detection algorithm(s) that performs a particular part of the content recognition task.

4.2 Interaction model

From the above follows that the distributed analysis object consists of a collection of interacting components where some components monitor other components and react to particular changes in their state.

An interaction model that enables multiple components to share results generated by other components is desirable from a resource consumption viewpoint, for scalability reasons. Components sharing results from other components may, however, introduce a problem with respect to conform usage.

Another interesting aspect regarding both interaction model and resource consumption is *push* versus *pull* style communication. Some of the components executing detection algorithms, typically those generating results used close to the root of the classification tree, will be running continuously, producing results possibly interesting to a large number of other components. Such producer components may best fit a push style communication model. On the other hand, components generating results, which are used further down in the classification tree, must rely on the consuming components for activation since the producing components do not have the control logic embedded inside. Such producer components may fit a pull style communication model better. A hybrid solutions is also possible, where a component working in pull mode is instructed to enter push mode for some time. Some algorithms may be very complex, time consuming and require a lot of computational resources. These algorithms do not seem particular well suited for real time analysis, but we believe such algorithms can be handled uniformly by components having a pull mode communication pattern.

A further refinement of pull mode interaction is the distinction between *eager* and *lazy* components. A lazy component may save host computer resources by not doing any calculation until explicitly requested. An eager pull mode component on the other hand, is able to deliver a result immediately upon request, but at the expense of increased resource consumption.

To support adaptation due to changing resource availability, the interaction model must be flexible, allowing arbitrary reconfigurations of analysis components. To achieve the latter, it would be advantageous that the different components do not need to be aware of each other, but communicate indirectly.

The outlined requirements and communication patterns fit very well with the publish/subscribe interaction paradigm, leading to an event based model. Event based systems relies on some kind of notification services, like an event (message) broker. The responsibility of the event broker is to propagate events (messages) from the event producer to event consumers residing in different computers, generally in a one-to-many manner. To achieve scalability, the event broker itself must be realized as a distributed service introducing a minimum of event propagation delay.

4.3 Computational architecture

The resulting computational architecture of distributed analysis objects is shown in figure 6. Each PD operates directly on a media stream and the calculated results are then delivered to other components through the event broker. Each CD consumes results produced by other PDs and/or CDs and deliver its own result to other components through the event broker. A filter may remove, convert or preprocess the information in a stream so as to simplify the task for other entities. The filtering process performed by a filter object is controlled by messages received from the event broker. The storage entity is not part of

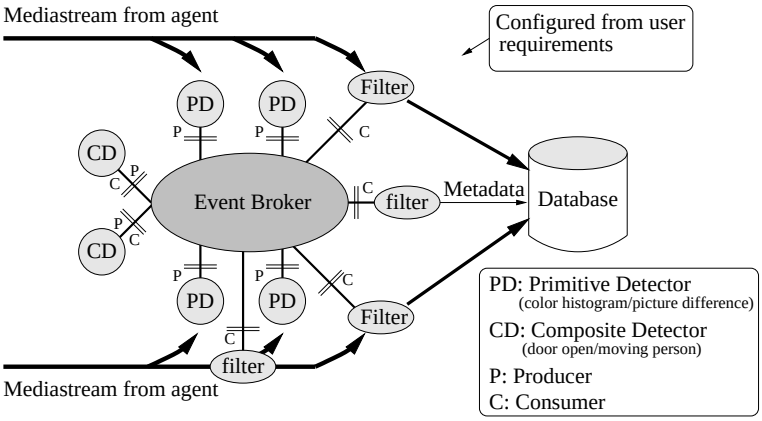

Figure 6: *The computational architecture.*

the distributed analysis object, but may be the sink of the media data as well as associated indexing and annotation information.

The configuration of an analysis object is the result of a chosen media processing strategy in which qualitative and quantitative requirements of the user are traded off against the available resources of the DPE. This requires knowledge about the resources of the execution environment and the performance and quality characteristics of each analysis algorithm.

4.4 Resource management

The characteristics of different DPEs vary with respect to their resource management capabilities. In general, dynamic selection of media processing strategy, and hence analysis algorithms, requires quantitative information about the cost of each strategy and information about the runtime environment.

At one extreme, there are DPEs that do not support any kind of resource reservation. A component running on a loaded computer in such an environment may experience overflow situations when it is not able to perform the calculations within the limited time frame determined by the real-time properties. Similarly, underflow situations may occur if the network is overloaded, causing starvation at components. In order to satisfy user requirements, such situations should be detected by monitor objects and result in adaptation in individual components or a reconfiguration of the analysis object. As we have seen in section 3, different media strategies have different tradeoffs between annotation reliability and resource usage. In a situation where available resources are insufficient, the system may change to a strategy with lower relative cost (c.f. figure 5).

At the other extreme, there are DPEs where it is possible to reserve resources, such as CPU, memory, disk and network bandwidth. This kind of DPE makes it possible for a DMJ system to provide some level of quality guarantee beyond best-effort. However, resource reservation is commonly not available in current DPEs and is an ongoing field of research.

Ideally, a distributed analysis object should be deployable in different kinds of environments, taking advantage of resource reservation whenever available and required by the user.

5 Related work

The current trend in image and video annotation is to use simple context-free feature extracts based on shape, color, or texture to describe content in an image (or video) [11]. This is evident from major work such as [5]. The computer vision community has demonstrated that context or domain knowledge is a necessity for inferring even the simplest concepts [11]. The potential benefit of integrating context in the annotation process is shown in recent multimedia applications such as [10] which implicitly take advantage of domain knowledge. A few recent annotation systems support explicit use of domain knowledge [4] [2], but none of these take explicitly into account the aspects of performance. Thus the handling of real-time requirements is limited.

6 Conclusions and future work

In this paper we have presented an architecture for a Distributed Media Journaling (DMJ) framework for on-line content analysis of networked multimedia sessions. The framework allows for formulating and transforming high-level content queries into distributable configurations of low-level media content querying algorithms.

The main challenges of DMJ are on-line automatic indexing and annotations of media content under uncertainty, and the computational complexity of content analysis. We argued that these challenges can be handled by an approach based on decomposable object-oriented Bayesian networks and an adaptable open distributed processing environment for executing content queries. A salient feature of our architecture is the combination of probabilistic knowledge-based media content analysis and resource awareness to handle uncertainty under real-time requirements

Currently we are concentrating on developing a first prototype of the DMJ framework. This prototype is based on Java Media Framework, [1], which performs much of the low level media tasks like capture, transport, (de)multiplexing, (de)coding and rendering. JMF also provides a pluggable architecture for integration of custom media processing algorithms. The primitive detectors developed for this prototype are pluggable into JMF.

In our future work we will conduct experiments to identify areas with potential for improvement. Some limitations we are already aware of includes the complexity of domain profile construction/tailoring, the disadvantages of the greedy media processing strategy generation approach used in the prototype and the difficulty of determining appropriate strategy generation threshold values.

References

[1] Java Media Framework, API Guide, 1999.

[2] W. Al-Khatib et al. An approach for video meta-data modelling and query processing. In *ACM Multimedia'99*, 1999.

[3] O. Bangsø et al. Top-down construction and repetitive structures representation in bayesian networks. In *FLAIRS-00, AAAI Press*, 2000.

[4] P. Castro et al. Using context to assist in multimedia object retrieval. In *First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.

[5] S. Chang et al. VideoQ: An automated content-based video search system using visual cues. In *ACM Multimedia'97*, 1997.

[6] N. Friedman et al. Structured Representation of Complex Stochastic Systems. In *15th National Conference on Artificial Intelligence, AAAI Press*, 1999.

[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*, 1995.

[9] J. Liu et al. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 1998.

[10] S. Mukhopadhyay et al. Passive capture and structuring of lectures. In *ACM Multimedia'99*, 1999.

[11] B. Perry et al. *Content-based Access to Multimedia Information - From Technology Trends to State of the Art*. Kluwer Academic Publishers, 1999.

[12] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.