# System Design for Structured Hypermedia Generation

Marcel Worring[1], Carel van den Berg[1], Lynda Hardman[2], Audrey Tam[3]

[1] Intelligent Sensory Information Systems
Department of Computer Science
University of Amsterdam, The Netherlands
worring@wins.uva.nl
[2] Interoperable Multimedia Systems Group, CWI, Amsterdam, The Netherlands
[3] Computer and Mathematical Sciences, Victoria University of Technology, Australia

**Abstract.** In this contribution we consider the design of a hypermedia information system that not only includes standard functionality of storage and presentation, but also the automatic generation of hypermedia presentations on the basis of a domain dependent knowledge base. We identify and describe the type of knowledge required and the processes involved.

## 1 Introduction

The usefulness of multimedia information systems hinges on the ease with which the information can be retrieved and on the speed and quality of the presentation of the information to the user. The most convenient way of interacting with multimedia information is through a hypermedia interface, where the user is guided in navigating through the large set of media items. This requires the definition of links relating the different pieces of information. This is a well known concept in hypermedia systems, but in many such systems, including HTML on the Internet, the links are embedded within the media. Recent research on hypertext models [13], hypermedia models [15] and open hypertext systems such as MICROCOSM [8] have introduced the concept of link databases in which links are stored separately from the media, using the notion of anchors. In such open environments it becomes feasible to integrate multimedia information systems and hypermedia interfaces.

With the large variety of user platforms and user requirements it is virtually impossible for an information provider to anticipate the full set of hypermedia presentations one is likely to encounter. Therefore, rather than trying to generate all possible hypermedia presentations beforehand we aim at a multimedia information system providing tools to generate presentations automatically when they are requested in a certain context by the user. To do so requires explicit knowledge about the domain. Examples of applications where such domain knowledge is present are medicine, weather, sports and news. In all of these domains, a large part of the domain knowledge is fixed whereas the media items are changing constantly.

Let us first consider an example of interaction with the proposed system. A person is consulting a database on animals and is looking at a multimedia presentation about the South Pole. At a certain point in a video on the animals living at the South Pole, the user sees a penguin, decides that more information on penguins would be interesting and clicks with the mouse on the visual representation of the penguin in one of the frames. If a link has already been created by an author, the user can follow it; however, there might be no link from the penguin to other components or anchors. Now, if the object in the video had an attribute stating that it is a penguin, we could retrieve all media items from the information system that are in some way related to penguins. This requires that we have a knowledge base describing the domain of penguins and their habitat. Now, retrieving relevant information gives us a collection of media items which are related via the domain knowledge description and possibly some stored hyperlinks. To provide for proper presentation this collection of media items should be structured automatically, combining them into coherent groups, e.g., all information on different species of penguins and one presentation on their diet. This results in a new hypermedia document that can be played at the user's hardware.

A system incorporating functionality related to the above is presented in [3]. The system is capable of automatically producing text- and/or graphics-based presentations tailored to the user's expertise, language and presentation hardware. However, both text and graphics are generated as needed; future work is planned to incorporate stored media items.

We are currently working on a system to achieve the latter functionality based on the extensible database system Monet [5] and the CMIF presentation environment [22]. In this paper we will consider some topics in the design of the proposed system. We will illustrate most concepts using video as it is the most complex and data-intensive media type. In section 2 the data model is introduced. Section 3 describes the processing steps used in the system and finally in section 4, a design for the architecture of the proposed system is presented.

## 2 Data model

The development of a system that automatically generates multimedia presentations is a complex task, drawing on the expertise of many disciplines. In an attempt to standardize terminology, functionality and architecture, Ruggieri et al. [24] have proposed a *reference model* for intelligent multimedia presentation systems, arguing that development, analysis and comparison of systems benefit from agreement on a reference model. In addition to basic terms, they define an *intelligent multimedia presentation system* (IMMPS) as a system that exploits knowledge sources to design multimedia presentations to achieve goals.

The reference model leaves open the data model to use. As data model, we use the Amsterdam Hypermedia Model (AHM) [15], which can be viewed as an extension of the Dexter model [13]. In the AHM, a hypermedia system consists of three layers: the *within-component layer* stores details of the content and internal structure of the components; the *storage layer* stores the hypermedia structure; the *runtime layer* stores information used for hypermedia presentation and handles user interaction. The IMMPS reference model fits into the AHM's runtime layer.

Given the complex functionality of the system, it is important to have precise definitions for the terms used for different types of information acted upon in the steps of the process. *Media items* are the raw pieces of data, e.g., a piece of video or sound. The Dexter model introduced the concepts *component* (both *atomic* and *composite*), *link* and *anchor*. Each component has a unique identifier. A composite component is a collection of atomic and/or composite (child) components. An atomic component contains *content* (a media item), *attributes* (semantic information [1]), anchors (objects embedded in the media item, such as an object in a picture) and a *presentation specification*, which describes how the component should be displayed by the system (for dynamic data, this includes the *duration*). A link connects anchors and/or components and can also be specified as a database query. (Note that anchors are *not* encoded within the media items and links are stored separately from components.)

The AHM incorporates and extends both the Dexter hypertext model and the CMIF multimedia model [6]: each atomic component is assigned a *logical channel* as presentation specification, which is an abstraction of a *physical channel* capable of playing the associated media item. Although the CMIF model allows the specification of timing constraints between components, the hierarchical structuring implicitly imposes a particular form of constraint, i.e., playing in parallel or serially. In the AHM, timing can be specified for child components relative to the parent or to a sibling component, e.g., start or end at the same time, or start a specific time after the other starts or before it ends.

We note here that the MPEG-4 project [18], if successful [2], may make it easier to identify and manage audiovisual (AV) objects. MPEG-4's goal is to establish universal, efficient coding of AV objects, of natural or synthetic origin, by defining a set of coding tools for AV objects and a syntactic description of the coding tools and coded AV objects. In the proposed syntax, each AV object has a local coordinate system (3D+Time). An AV object can be placed in a scene by the encoder or end-user via a coordinate transformation from its local coordinate system into the scene's (global) coordinate system. This coordinate transformation is part of the scene, not part of the AV object. AV objects may be composites of other AV objects.

The information on presentations in the above described format must be stored in four logical databases: the media database, the knowledge base, the link database and the component database. To achieve independence in the processing steps presented later, these databases do not have symmetric relations.

[1] Derivation of the attributes is outside the scope of the AHM. In our system, this will be handled by the annotation phase (see section 3.1): each component will receive a *semantic annotation* that is an instantiation of one or more concepts in the knowledge base.

[2] MPEG-4 is scheduled to become international standard by November 1998.

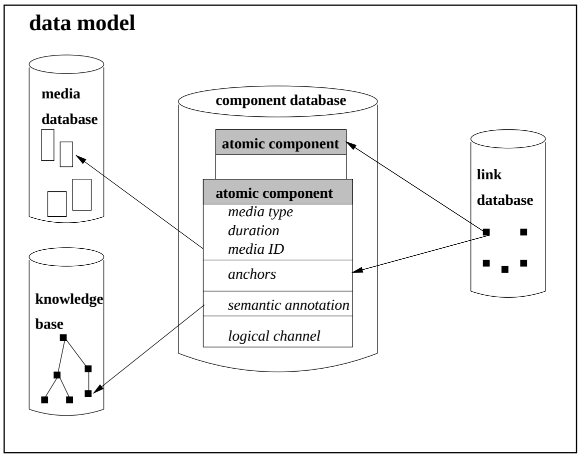


**Fig. 1.** *The logical databases used in the system and their relations.*

The exact relations are shown in figure 1.

# 3 Processing steps

Let us go back to the example in the introduction as it indicates the information processing steps carried out by the system. In general we can divide the processing into three main steps. In the *hypermedia creation* step basically all steps are done that are carried out prior to, or at the time of entering the media items into the system. To be precise, it involves the creation of the knowledge base, insertion of the media items and giving annotations, and finally the entering of links into the link database. The other two steps are performed at run-time and will be performed in an alternating sequence. These steps are the *hypermedia presentation* (playing) of hypermedia documents and the *hypermedia generation* performed whenever the user wants to follow links other than those foreseen by the author. The different processing steps are illustrated in figure 2 and will be described in the following sections.

## 3.1 Hypermedia creation

**Knowledge representation** Domain knowledge needs to be represented in different ways for different purposes. Our interest is limited to the task of finding media items similar to another item. That is, we are concerned with the similarity of different pieces of information and are not concerned with interpreting what those pieces of information actually mean or represent. Kashyap et al. [16] approach the problem of semantic correlation of information from different media types by proposing an architecture with three levels: ontologies, metadata

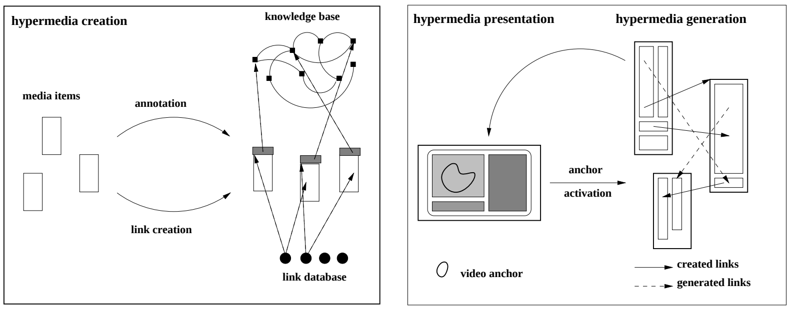


**Fig. 2.** *Overview of the different steps.*

and databases. Ontologies contain both domain-specific and domain-independent terms that characterize the (semantic) content of the databases, irrespective of media type. Metadata are classified as content-dependent, -descriptive or -independent information about the data (text, image, audio, video) in the databases. The ontology and metadata levels are evident in AHM components.

Porter notes that the types of features used for processing the semantics of items are of crucial importance [21]. In particular, superficial features that are about form and are independent of context and goal-of-use, e.g., size, color and material, should be distinguished from abstract features that are about function and are dependent on the context and goal-of-use, e.g., "hammer" (a tool for hitting). Superficial features can be derived from the media items through data analysis. Abstract features are much harder to obtain but might often be more useful in matching. An example of a system based purely on superficial features is the QBIC system [11] whereas in e.g. CORE [30] the distinction between the two types of features is made explicit.

Two ways of obtaining abstract features are by hand or via a domain knowledge representation. Our approach is to first create a domain knowledge representation by hand, and to assign aspects from this to the anchors within the media items by hand. This can be combined with an analysis of the raw data of the media items to obtain superficial features.

As concerns domain based annotation, work has been carried out by Davis [9] for the particular case of video. His task was slightly different to our own, but sufficiently similar to form a basis for this work. His chosen representation for domain knowledge of a collection of video sequences is based on knowledge frames, to be more specific the Framer system [12]. In broad terms this is a hierarchical frame-based structure, allowing multiple values for slots, where any node in the structure (leaf or interior) can be used for describing the persons and objects in the video as well as the activities performed by the subjects. For example, a video showing a penguin walking on an ice-floe would have the description "penguin" as a semantic attribute, which is a specialization of "bird", which is a specialization of “warm-blooded creature” etc. Another attribute used would be “standing”, which is a specialization of “pose” and so on. The representation in Davis is very broad. In [27], narrower domain models are used, geared towards documentary video and news programs.

The knowledge base should also be capable of providing a similarity measure for two items. This is done by considering the hierarchical organization of the knowledge. Each semantic annotation consists of a number of attributes. The similarity between two attributes of different annotations can be defined as the number of steps one has to make in the hierarchy when moving from one concept to another. Hence, when given two annotations, a set of values is returned one for each attribute of the annotation. A zero value indicates that the match is exact and a positive value indicates that the match is inexact. Going back to the previous example, consider a picture showing a penguin lying on a beach. Both the video and the picture would have the semantic attribute “penguin”, so this attribute matches exactly. The semantic attributes “lying” and “standing” have a distance of 2: one from “lying” to “pose” and one from “pose” to “standing”.

Given a set of annotations corresponding to components or anchors we can define a similarity matrix or graph giving the similarity measures among all items in the set. Such a similarity matrix will be used in the hypermedia generation step. In the next section we will first discuss the annotation step.

**Video annotation** The task of annotating a video begins with *parsing* it into “meaningful” segments, both temporally (e.g., shots) and spatially (e.g., objects). Each of these segments can then be annotated, as the semantic content of a video is usually closely related to its temporal structure and significant moving objects. Fully automated annotation is probably too difficult, except for specific, well-defined domain areas; our system aims for semi-automatic annotation, presenting candidate segments to human annotators for confirmation, and attempting to control the variation among annotations, e.g., by using icon palettes [12]. A secondary reason for video parsing is pragmatic: in most cases, a hypermedia user would prefer to view only the relevant segment(s) of a video rather than the entire video.

Our system will be based on the following spatial and temporal segments:

- object: a connected region of arbitrary shape (which may vary over time) that appears to move as a coherent entity, e.g., a penguin. Annotating individual objects gives a more direct association of the objects with their semantics. It should furthermore be noted that the information in the knowledge base is not restricted to video, but can also be applied to related media items such as textual descriptions or audio fragments.
- shot: a sequence of pictures that appear to have been continuously filmed, e.g., consecutive pictures showing a penguin walking across the ice. Each shot will also be annotated with the type(s) of camera work (pan, zoom etc.) it contains, not only to answer queries that request a specific type of camera work, but also as semantic hints to aid in object and motion detection and in shot abstraction (see section 3.3).

- scene: a sequence of shots with the same location, e.g., a shot of diving penguins followed by shots of the penguins underwater;
- story-unit: a sequence of shots with common semantic content (a scene is a type of story-unit), e.g., shots of feeding female Emperor penguins interleaved with shots of their mates (100 km away) incubating the eggs. Most queries for specific semantic content will be answered by story-units.
- thread: a collection of scenes and/or story-units with common or related semantic content, e.g., story-units about penguins separated by story-units about seals, whales, scientists in the Antarctic etc. Threads are useful for maintaining the temporal relationships among related story-units and could be used to answer a query that is not specific enough to isolate a single story-unit.

To parse videos into these different types of segments, we will explore the use of existing methods, and also develop new techniques. Methods for detecting shot boundaries automatically are plentiful (e.g., [27, 14, 20]); work on clustering shots into story-units is well-advanced; efficiently detecting objects remains a hard problem. The segmentation problems and our proposed approaches to solving them are discussed in more detail below, but first, we briefly describe the video annotation process.

The process of annotating a video will begin by detecting shot boundaries. Each shot will be analyzed for camera work, moving objects and background, and its key frame(s) will be identified. (Key frames and other video-previewing tools are discussed in section 3.3.) Shots will then be clustered into story-units, which will be presented to the annotator for confirmation or amendment. During and after this clustering step, the annotator will be able to assign attributes to story-units, either individually or in groups (i.e., threads). To reduce variation in annotation, the attributes will be selected from a hierarchical palette, which the annotator can modify during the annotation process. Shots and objects in a story-unit will initially inherit its attributes; in the next step, the annotator will modify attributes of shots and objects, as needed. Finally, the system may suggest thread (re)groupings, based on story-unit annotations. The video annotation process is illustrated in figure 3.

*Shot boundary detection:* Shot-boundary detection can be done as reliably by computers as by humans [1]. Yeung et al. [31] detect shot boundaries using only reduced (approximate) DC images[3], i.e., only partially decoding the MPEG pictures, and reducing the size of the images by a factor of 64. Although methods to detect progressive shot transitions (fade, dissolve, wipe) are generally less reliable than those to detect abrupt cuts, it can be concluded that accuracies of 90-95% can be achieved with the available methods. In an interactive environment, this

[3] MPEG pictures are encoded in 8x8-pixel blocks. A discrete cosine transform (DCT) is applied to each block, resulting in one DC coefficient and 63 AC coefficients. The DC coefficient is just the average value of the original 64 values. A DC image contains only DC coefficients, i.e., one pixel for each 8x8-pixel block in the full-size picture.

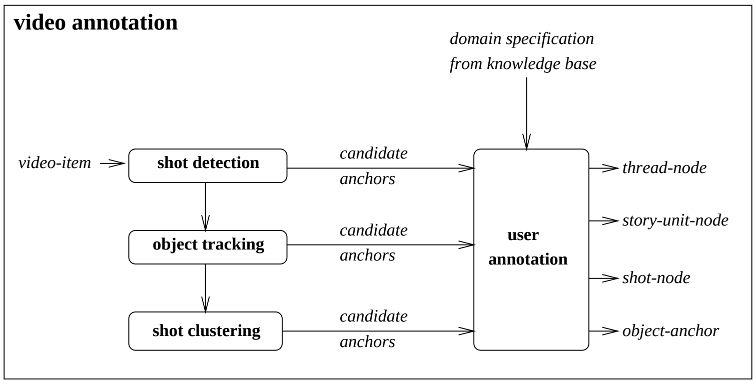


**Fig. 3.** *Overview of the annotation process.*

will generally be sufficient. In particular, a camera pan or zoom sequence might be misinterpreted as a gradual transition, so all detected gradual transitions should be checked during the camera work analysis step. Existing commercial and prototype software MPEG encoders use shot boundary information to optimize the allocation of bits to pictures [10], e.g., the first picture after an abrupt or gradual transition is encoded as an intra-frame (a single image, with no reference to past or future pictures). We anticipate that this practice will become commonplace in the near future; inspection of (DC images of) intra-frames will then be sufficient to detect shot boundaries during the MPEG *decoding* process. Until this happens, however, we will adopt the methods of [31].

*Scene analysis:* After segmenting the video into shots, sequences of shots can be grouped into semantic units - story-units and threads (although the Informedia project [28] first detects *video paragraphs*, then segments each into its component shots). Yeung et al. observe that a few minutes of video typically contain hundreds of shots. They therefore cluster sequences of shots (related by a common locale or dramatic event) into more meaningful story-units. Similarity measures on DC images and temporal heuristics enable semi-automatic detection of story-units. Image similarity measures and annotations may suggest candidates for inclusion in a thread, but inspection by the (human) annotator will be needed to ensure correctness.

*Camera work analysis:* Apparent motion between consecutive frames can be caused by motion of an object in the picture or by camera work (or both). Camera work analysis therefore involves distinguishing camera work from object motion; this information will be useful in the object detection step. Panning and zooming sequences can produce histogram patterns similar to gradual shot transitions, so camera work analysis is also needed to eliminate false transitions.

The basic approach to detecting a camera pan or zoom is analysis of the optical flow field: panning and zooming produce distinctive patterns of motion vectors; gradual transitions and moving objects do not (unless the object is very large relative to the image size). [2] presents a computationally efficient method for detecting camera work, based on tomography: the "X-ray" projections are the average values of each row and column in successive images. We plan to investigate the possibility of using this method in the partially-decoded domain, e.g., DC images. [25] generated X-rays from JPEG-compressed images, using only low-order DCT coefficients. Meng and Chang [19] also detect panning and zooming in the compressed domain.

*Object detection and tracking:* Detecting objects in a fully interactive way is a tedious task as first shots have to be defined (probably using a hierarchical magnifier as described in [27]) and then objects have to be outlined in every subsequent frame. Using computer vision techniques, the user can be aided in these two tasks. Rowley et al. [23] have trained a neural network to detect human *faces* in images containing frontal views of faces (with both eyes visible and open). In [7] the use of video objects as anchors is proposed. Their method of defining anchors aids the user in defining objects by using interpolation. However, this is purely based on computer graphics and no use of the video data itself is made. Hence, the resulting video objects have an inaccurate representation.

We intend to use an object tracking framework similar to [4]. In this framework, a contour is parameterized with a small set of parameters using B-splines. Based on the video data found locally around the contour, the object is tracked through the sequence using predictive filtering. For our purposes, we will extend the framework to take into account color edges rather than intensity edges. This might be combined with the estimation of dominant and multiple motion models to separate the fixed (background) layer from moving objects and to differentiate camera motion from object motion [11]. In [19], moving objects are detected in some types of MPEG-encoded video, using motion vectors and DC coefficients; however, DC image analysis is inadequate for detecting small object movements, as each DC coefficient is the average value of an 8x8 block of pixels.

**Link database creation** Although we aim at automatic generation of links, authors have the option of creating or adding to a link database. For links based on information not explicitly coded in the knowledge base, this is even essential. Adding links that can be derived from the knowledge base should be done in close conjunction with the hypermedia generation step described in the next section. Adding those to the link database might speed up the processing, but adding them to the link database explicitly is not essential.

### 3.2 Hypermedia generation

Whenever the user selects a subject of interest, by selecting an anchor in a media item, for which more information should be provided, the semantic attributes associated with the selected anchor are recorded. A call is made to the database to select items that are similar to the given set of attributes [4]. This will yield a set of media items as well as a similarity matrix describing their relations (see section 3.1).

The set of media items should be presented to the user in a structured way as a hypermedia presentation. Here we have to decide which items will be grouped into composite components and how they will be connected by links. This is done on the basis of a set of heuristics. Heuristics for semantic grouping can be based on the matching criteria in [17]:

- goal-directed: group components that involve the same goal;
- salient-feature: group components that match most important features or largest number of important features;
- specificity: group components that matches features exactly over those that match features generally;
- frequency preference: group components that are matched frequently;
- recency preference: group components that are matched recently;
- ease-of-adaptation: group components for which the features are easily adapted to new situations.

The above criteria can be mapped directly to the distances defined in the similarity matrix or their dynamic behavior. Example composition and linking heuristics not based on the semantics are:

- create the smallest number of composites;
- create the smallest number of links between composites
- don't allow incompatible media types (e.g. 2 videos together).

The weighting for the different criteria depends on how the different approaches work in practice, and may even be put in the hands of the end user e.g. to give preferences for an overview of a subject area, or an in-depth search.

### 3.3 Hypermedia presentation

In the above generation step, a complete hypermedia presentation description is derived. At this point it does not yet include the actual media items, but a logical channel has been assigned to each media item. As indicated before, a logical channel represents a physical channel capable of playing the media item. Hence, when the hypermedia presentation is sent to the client, logical channels have to be mapped to the physical channels available. When this mapping is performed, the client informs the database server of the physical channel properties, e.g., resolution for a picture channel.

---

[4] In the ideal situation there would be some sort of memory, so that the user's previous selections can be stored and the associated attributes used to contribute to the information used in the database search.

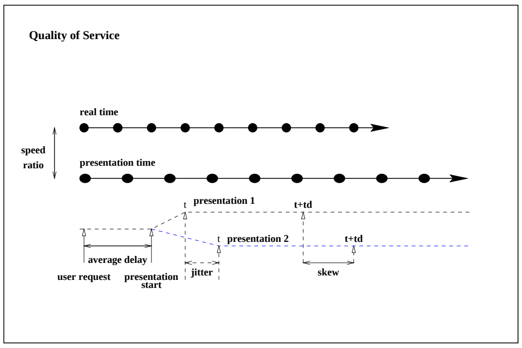


**Fig. 4.** *Quality of service.*

A key issue for acceptance of a multimedia information system is the quality of service provided by the system. The parameters that determine the quality of service are depicted in figure 4. Ideally real time equals the presentation time. The *ratio* between real time and the presentation time is an important quality factor. *Average delay*, *jitter* and *skew* represent relative timing delays between user requests and simultaneous presentations. The last factor, the *utilization*, describes the ratio between the data volume used for the presentation and the data volume available.

From this description it is clear that the quality of service is timing related. Consequently, this aspect has a great impact on the system design as a whole. As the multimedia data used for a presentation may be too voluminous to be stored in the client, the database must be designed to offer the quality of service. In other words, it must have a real-time kernel.

To keep a specific quality of service it might be required to reduce the utilization such that all of the above measures are kept within acceptable limits. This can for example be achieved by sending images at a reduced resolution. This adaptation can also be initiated by the client when for example a window in which a media item is presented is resized. So apart from sending information on the properties of the physical channels also the required quality of service should be communicated. On the basis of these measures the database can start sending the actual media items to the client in the appropriate format and with the highest quality possible within the constraints. The whole process is shown in figure 5.

**Video presentation** For delivery of video from the database server to a client workstation, users will be offered two options: (1) the server decodes and sends the pictures to the client or (2) the server sends the MPEG stream to the client for local decoding, either in hardware or software. These options can be combined with quality-of-service parameters, such as reduced picture rate or resolution. In addition, the client and/or server might buffer sequences of decoded pictures (in physical or virtual memory) if there is enough storage.

The first option requires a very high bandwidth network [5] to provide the highest quality of service and places a heavy load on the server's processor(s) and/or storage (depending on the number of pictures buffered in the server). The second option requires a client workstation with reasonable processing and storage capabilities.

All three - network bandwidth, processing power and storage capacity - are increasing rapidly and becoming more affordable. However, network bandwidth fills up almost as soon as it becomes available, so the first option cannot always be depended on for timely delivery of decoded pictures to the client. On the other hand, individual users often have access to workstations with fast processors and large amounts of storage, and workstations are more readily upgraded than are networks or database servers. Hence, the second option may prove to be more viable. The client software for accessing the video database server would include the MPEG decoder. There would be a switch to use the client workstation's hardware MPEG decoder, if it has one.

*Video abstraction and representation:* Many queries will retrieve several relevant video clips (shots, story-units or threads). These should be presented to the hypermedia user in a compact, yet informative, format that enables them to decide which clips to view in full. Several techniques have been proposed for automatically deriving an abstract of the content of a video or video segment, and presenting this information visually and concisely. Initially, we will evaluate the following video representations for computational efficiency and conciseness.

- Key frames: one or a few pictures are chosen to represent the video or video segment; these can be displayed as thumbnail (e.g., DC) images to further reduce the amount of data transferred. The selection algorithm might be purely positional - e.g., the first, middle and last pictures in a shot - or it might be based on similarity measures and frequency - e.g., this segment contains many pictures similar to this - or on camera work - e.g., the first and last pictures of a zoom. The Informedia project [28] combines clues from image analysis and keyword prominence analysis to identify a key frame for each shot in a video paragraph (story-unit), and to select one of these to represent the entire paragraph.
- Mosaic (salient still) [11]: several pictures in a shot are combined to create a single still image that retains much of the original content and context while dramatically reducing the amount of data (e.g., a 64KB mosaic of a 22MB panning shot of Yosemite Valley is available from QBIC's WWW page *http://www.almaden.ibm.com/cs/video/*). Mosaics are easily created from the results of dominant motion estimation, which we plan to use for object detection (see section 3.1). Because this technique separates moving

[5] Typical video picture rates are about 30 pictures per second. A decoded full-color 256x256 pixel picture occupies 256 KB. Sending 30 decoded 256x256 full-color pictures per second to a client workstation requires a network bandwidth of 60 Mbps.

objects from the background, it can also be used to create a *dynamic* mosaic, with the objects moving against a completely static background mosaic. A mosaic is an alternative to a key frame for a shot; although it may be possible to create mosaics of some *scenes*, most story-units would require more than one mosaic.

- Scene transition graphs [31]: the story-line of an entire video is represented as a directed graph. each node is a cluster of visually similar and temporally close shots, represented by a key frame; edges indicate the temporal flow of the story. *Cut edges* partition the graph into story-units. Since our system is most concerned with representing story-units, we will be interested in the degree of clustering possible, i.e., the ratio of the number of shots to the number of nodes in a story-unit. For some types of video (e.g., situation comedies), this ratio can be quite high.
- Skim video [26]: "significant" images and words are selected from the video segment and accompanying audio to create a compact synopsis (e.g., 1100 frames of the documentary "Destruction of Species" can be represented by a 78-frame skim video). Natural language processing is applied to the transcript to identify the most important keywords and phrases and the time-corresponding frames are examined for scene changes, camera work and relevant objects. Additional clues are obtained from numerous heuristics based on widely-used film production practices. Skim videos provide an efficient filtering mechanism to the hypermedia user, if the static representations are inadequate for differentiating the retrieved video clips.

# 4 System architecture

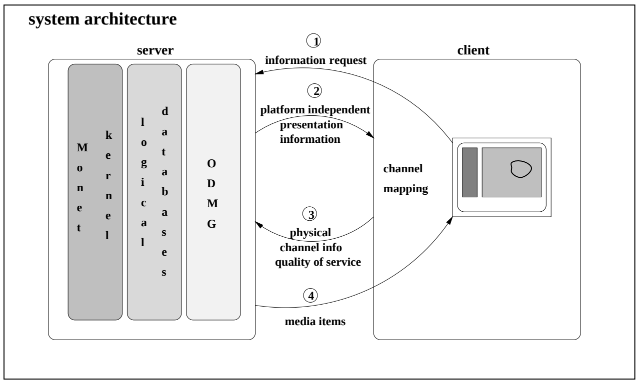


**Fig. 5.** *Overview of the layered architecture of the multimedia information system (or server) and the communication with the client in the presentation of hypermedia.*

The limitations for storing multimedia information in relational systems are well known. The kernels of these systems simply do not provide the hooks for achieving the fine control required in meeting the real time constraints. Object-oriented systems have the advantage that they enable the expression of multimedia data operations, such as image and video algebra [29]. Unfortunately, real-time behavior is not achievable with these systems.

Extensible databases provide the required level of control to implement multimedia databases. They allow extension of a small fixed database kernel with application specific data types and operations. Our system is based on Monet [5], an extensible main-memory database system. Monet uses a flexible and efficient decomposed storage model and offers database triggers, a type extension mechanism and a set of binary relational algebra operations. The latter have predictable performance, which is important in achieving a certain quality of service.

The basic media types can be implemented using the type extension mechanism of Monet. These types include video, audio, images, links, anchors and the spatial and temporal relations used for presentation. At the application interface level an Object Oriented interface to these types is provided based on the ODMG data model. This has the advantage that a seamless interface is provided for applications written in any language. Currently, bindings for C++ and Java exist.

The image, video and temporal types have been implemented and the audio type is under construction. Video pictures can be converted into full-size or reduced DC images; images can be processed by image processing operations, similarity measures and detectors. Work has begun on the object tracking system for video annotation. The annotation and retrieval mechanisms will be implemented as ODMG applications.

## 5 Conclusion

We have considered the design of a multimedia information system in which hypermedia presentations can be generated automatically. In the system, we have identified four logical databases: a knowledge base, a media database, a component database and a link database. The processes acting upon this information can be divided into three steps. In the *hypermedia creation* step, the logical databases are populated. In this step, each component is semi-automatically assigned a semantic annotation. The *hypermedia generation* step is based on heuristics using a similarity function for grouping and linking components. Finally, the design of the hypermedia system architecture is such that real-time adaptive *hypermedia presentation* can be achieved.

## References


1. P. Aigrain, H.J. Zhang and D. Petkovic, Content-based representation and retrieval of visual media: A state-of-the-art review. *Multimedia Tools and Applications*, 3, pages 179–202, 1996.
2. A. Akutsu and Y. Tonomura, Video tomography: An efficient method for camerawork extraction and motion analysis. *Proc. ACM Multimedia Conference*, 1994.
3. E. André and T. Rist, Coping with temporal constraints in multimedia presentation Planning. *Proc. ECAI 96*, 1996.
4. A. Blake et al., A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, 1993.
5. P. Boncz and M.L. Kersten, Monet: an impressionist sketch of an advanced database system. *Proc. BIWITT'95*, 1995.
6. D.C.A. Bulterman, G. v. Rossum and R. v. Liere, A structure for transportable, dynamic multimedia documents. *Proc. USENIX*, pages 137–155, 1991.
7. V. Burrill, T. Kirste and J. Weiss, Time-varying sensitive regions in dynamic multimedia objects: a pragmatic approach to content based retrieval from video. *Information and Software Technology*, 36(4):213–223, 1994.
8. H. Davis, W. Hall, I. Heath, G. Hill and R. Wilkins, MICROCOSM: An open hypermedia environment for information integration. *Proc. ECHT*, 1992.
9. M. Davis, *Media streams: representing video for retrieval and repurposing*. PhD thesis, MIT, 1995.
10. A. Dev, Personal communication, 1996.
11. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: the QBIC system. *IEEE Computer*, 28(9), 1995.
12. K. B. Haase, FRAMER: a persistent portable representation library. *Proc. ECAI 94*, 1994.
13. F. Halasz and M. Schwartz, The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.
14. A. Hampapur, R. Jain and T. Weymouth, Digital video segmentation. *Proc. Second ACM International Conference on Multimedia*, pages 357–364, 1994.
15. L. Hardman, D.C.A. Bulterman and G. v. Rossum, The Amsterdam hypermedia model: Adding time and context to the Dexter model. *Communications of the ACM*, 37(2):50–62, 1994.
16. V. Kashyap, K. Shah and A. Sheth, Metadata for building the MultiMedia Patch Quilt. In *Multimedia Database Systems: Issues and Research Directions*, S. Jajodia and V.S. Subrahmaniun, Eds., Springer-Verlag, 1995.
17. J. Kolodner, Judging which is the "best" case for a case-based reasoner. *Proc. Case-based reasoning workshop*, 1989.
18. MPEG Convener, Description of MPEG-4, *http://drogo.cselt.stet.it/mpeg/mpeg-4_description.htm*.
19. J. Meng and S.-F. Chang, Tools for compressed-domain video indexing and editing. *SPIE Conference on Storage and Retrieval for Image and Video Database*, Vol. 2670, 1996.
20. K. Otsuji and Y. Tonomura, Projection detecting filter for video cut detection. *Proc. First ACM International Conference on Multimedia*, pages 251–257, 1993.
21. Porter et al., Concept learning and heuristic classification in weak-theory domains. *AI Journal*, 1990.

22. G. van Rossum, J. Jansen, S. Mullender and D. Bulterman, CMIFed: a presentation environment for portable hypermedia documents. *IEEE Multimedia*, 1993.
23. H. Rowley, S. Baluja and T. Kanade, Human face detection in visual scenes. CMU Technical Report *CMU-CS-95-158R*, 1995.
24. S. Ruggieri, M. Bordegoni, G. Faconti, T. Rist, P. Trahanias and M. Wilson, The Reference Model for Intelligent Multimedia Presentation Systems: 1st Draft. Proc. ECAI 96 Workshop *Towards a Standard Reference Model for Intelligent Multimedia Systems*, 1996.
25. F. Salazar and F. Valéro, Analyse automatique de documents vidéo. IRIT rapport de recherche *95-28-R*, Université Paul Sabatier, 1995.
26. M.A. Smith and T. Kanade, Video Skimming for Quick Browsing Based on Audio and Image Characterization. CMU Technical Report *CMU-CS-95-186*, 1995.
27. S.W. Smoliar and H. Zhang, Content-based video indexing and retrieval. *IEEE Multimedia*, pages 62–72, 1994.
28. H.D. Wactlar, T. Kanade, M.A. Smith and S.M. Stevens, Intelligent access to digital video: Informedia project. *IEEE Computer*, 29(5), pages 46–52, May 1996.
29. R. Weiss, D.D. Andrzej and D.K. Gifford, Content-based access to algebraic video. *Proc. International Conference on Multimedia Computing and Systems*, pages 140–151, 1994.
30. J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam and Y.J. Gao, CORE: a content based retrieval system for multimedia information systems. *Multimedia systems*, 3:25–41, 1995.
31. M. Yeung, B.L. Yeo and B. Liu, Extracting Story Units from Long Programs for Video Browsing and Navigation. *Proc. International Conference on Multimedia Computing and Systems*, 1996.