Link Management Framework for Hyper-media Documents

Dragos-Anton Manolescu
National Center for Supercomputing Applications
405 N. Mathews Ave., Urbana, IL 61801
daman@ncsa.uiuc.edu

Klara Nahrstedt
Department of Computer Science
1304 W. Springfield Ave.,Urbana, IL 61801
klara@cs.uiuc.edu

Abstract

This paper presents a framework for link management within hyper-media documents. The framework includes: (1) an object-oriented document architecture with a consistent interface for different media types; (2) a transition model between multimedia objects founded on content-based static links for atemporal media, and content/time-based dynamic links for temporal media; and (3) intrinsic support for content-based access which requires a reduced amount of indexing. A prototype system has been implemented to verify the feasibility of the paradigm. The results confirm that the framework is viable and back up the formalism.

1. Introduction

Recent advances in multimedia technology have made multimedia documents commonplace. With the availability of many different media types, applications require new access methods and extended functionality, which are not possible with the text-only counterparts. Existing systems (e.g., the World-Wide Web) already integrate heterogeneous multimedia objects within one document. However, the available exploration methods, originally developed for text documents, do not take into account the additional dimensions associated with multimedia information.

The aim of this paper is to provide a framework for advanced link management within a refined multimedia document architecture. The primary contributions are the following:

1. Based on multimedia objects, we introduce an object-oriented architecture for hyper-media documents which allows for non-linear transitions between multimedia objects in a consistent manner.

2. We formalize the relationships (links) between multimedia objects. Static links parameterize transitions only by contents and work on any type of objects. They are already widely used in systems like the World-Wide Web. For objects with a temporal dimension (e.g., audio, video) we introduce dynamic links, which parameterize transitions by contents and time. To represent a hyper-media document with dynamic links, we extend the tree-like hierarchical structure to accommodate the additional temporal dimension and introduce planar nodes and planar document structures.

3. We describe the intrinsic support our framework provides for content-based access and identify different query types.

Section 2 describes the problem and presents a survey of other systems which handle non-linear transitions between different media types, content-based access and hyper-links. Section 3 introduces the terminology, notation and semantics within our framework. The set of operations on multimedia objects is described in Section 4. We summarize in Section 6.

2. Problem description

To make our objectives easier to understand, we begin with a short example:

> Assume you are watching a video which shows the northern part of the campus at the University of Illinois at Urbana-Champaign. The camera pans slowly over the Beckman Institute and while the building is visible, a message window signals that a video about the Institute is available and a transition to this video object is possible. You click on the image at this time and the playback of the new video starts. The camera enters the building and you can see the Beckman instruments, which are on display at the South entrance. While the images display these instruments, the message window shows that a text document about them is available. While this tran-

sition is possible, you click on the output window again. The video presentation pauses and a window displaying this text pops up. When you are done reading, you close the text window and the Beckman video resumes playback at the point where it was interrupted. You see images from the main lobby, and the message window shows that a sound file is available. You decide you have seen enough of the Beckman Institute for now and select the return button. The video with the North campus resumes from where it was interrupted. The camera is now over the Computer and Systems Research Laboratory. The message window signals that an audio-video presentation of this laboratory is available. Next you see the Civil Engineering Building, and then the Digital Computer Laboratory. You decide that you would like to see all the videos that are available about DCL. You send a query to search for these objects and start watching the ones that have been found.

This paper describes a framework that allows for presentations like the previous one. We abstract a multimedia document as a hierarchy of multimedia objects, which could be text, graphics, images, video, audio, etc. We focus on two aspects: *links*[1] between related multimedia objects; and *content-based access*.

We use an object-oriented architecture which offers a high-level abstraction and is easy to integrate within existing multimedia document standards, like MHEG [9, 13]. The superclass of all document objects, *multimedia object*, is an abstract class. This is illustrated in Figure 1, where we use the notation from [5] and the abstract classes appear in italics. The two subclasses correspond to different requirements. The *temporal* subclass includes all objects which are intra-domain time dependent (e.g., audio, video, etc.). The *atemporal* subclass includes objects which do not have intra-domain time dependencies (e.g., text, static images, graphics, etc.).

Based on the architecture from Figure 1, we start by describing other systems that deal with hierarchical visual information, non-linear content-based transitions and hyperlinks.

For each system, we focus on two characteristics: the *media types* of entities the system operates with, and the *transitions* between related entities. In each case, the behavior is illustrated in a transition diagram which represents different entities as filled circles and the transitions between them as arrows.

- The Slovenian Virtual Gallery [12] offers a virtual walk through several galleries, organized as a hierarchy of images. The user controls the presentation by clicking on different parts of an image. This determines the next image. The system handles just atemporal information, and the only possible type of transition is given in Figure 2.

[1] The abstractions we use are independent of the media. A link relates several objects which could have different media types. For this reason, there is no need to use the term "hyper-link."

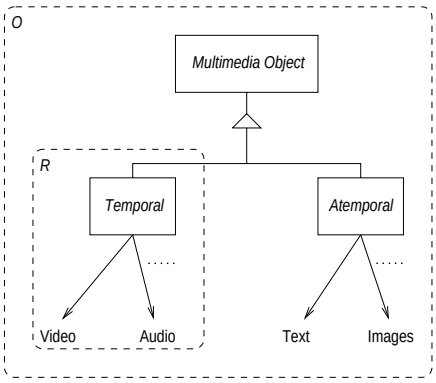

Figure 1. Multimedia document architecture. We denote the set of all multimedia objects with $\mathcal{O}$, and the set of temporal multimedia objects with $\mathcal{R}$.

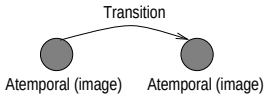

Figure 2. Transition diagram for the Slovenian Virtual Gallery.

- The Advanced Video Information System (AVIS) [1] organizes video data to facilitate efficient querying. A video is first broken down into short sequences and several kinds of entities (objects, activities, and events) are associated with them. The system defines a 9-tuple database [3] which supports different types of queries. Possible transitions are from the atemporal domain (queries) to the temporal domain (video sequences)—Figure 3.

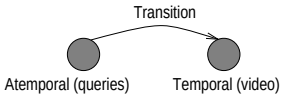

Figure 3. Transition diagram for AVIS.

- The Query by Image Content (QBIC) [4] is another content-based image retrieval system (CBIR) [6] which takes a different approach. During an initial

phase, called "population," images and videos are processed to extract features describing their contents. These could be colors, textures, shapes, camera and object motions. Queries are then composed graphically and their features are extracted as well. A matching engine finds the videos or images with similar features from the database. This is illustrated in Figure 4.

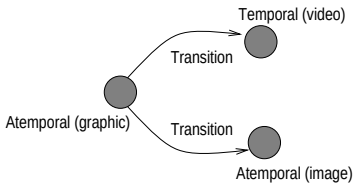

Figure 4. Transition diagram for QBIC.

- Vosaic [2] is a World-World Web browsing system extended to support continuous video media. Although not a CBIR system, it introduces the notion of "video hyper-links" which allow a user to click on an image in a video stream and pull up another video stream. The corresponding transition diagram includes transitions between any combination of video and atemporal media—Figure 5.

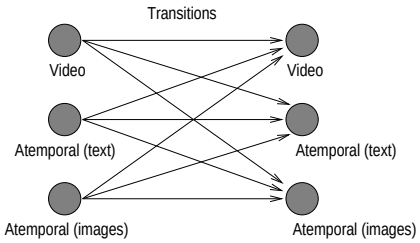

Figure 5. Transition diagram for Vosaic.

- For our framework, all objects in a document are subclassed from the multimedia object abstract class—Figure 1—and *transitions are regarded in a consistent way*, independent of the media type. Therefore, transitions between any combination of temporal and atemporal objects are possible. Additionally, for temporal objects, we extend the notion of links and *parameterize transitions by both contents and time*. This is illustrated in Figure 6, where the dashed arrows represent transitions that depend on both contents and time.

The following two sections set up the framework. In Section 3, we introduce the terminology, state the assumptions and define several concepts. Then in Section 4 we describe the operations for hyper-media document manipulation.

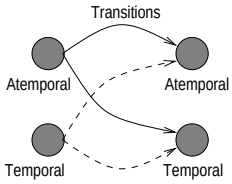

Figure 6. Transition diagram with dynamic links.

3. Definitions and assumptions

We define the class of multimedia objects according to a composite model. An entity within our hyper-media document model can be either a single multimedia object (Figure 1) or a collection of multimedia objects.

The basic unit of a temporal object is a *frame*. A set of frames creates a *strand*, a set of strands creates a *ring* and a set of rings tied together by synchronization information creates a *rope*. In the rest of this paper, we use the terminology given in Table 1.

Dynamic links include a temporal dimension. They parameterize the relationships between objects by contents and time. Assume $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ the set of temporal multimedia objects, where $\mathcal{R} \subseteq \mathcal{O}$ (these sets are figured as dashed boxes in Figure 1) and $T = \{\tau;\ \tau = [t_i, t_j]\}$ the set of time intervals corresponding to these objects. The function $d_l : \mathcal{R} \times T \rightarrow \mathcal{O}$ models the links between objects in $\mathcal{R}$ and objects in $\mathcal{O}$.

For example, if $r \in \mathcal{R}$ is a multimedia object with $\tau \in T,\ T = [0, 200)$ seconds: $d_l(r, \tau_1) = o_1,\ \tau_1 = [0, 25)$, $d_l(r, \tau_2) = o_4,\ \tau_2 = [80, 100)$ and $d_l(r, \tau_3) = o_{10},\ \tau_3 = [100, 200)$. This defines a dynamic link from r to o_1 for the first 25 seconds; no link for the next 55 seconds; from r to o_4 for the next 20 seconds; and from r to o_{10} for the last 100 seconds.

Static links are a specialization of dynamic links, where the space is atemporal and the transition depends just on contents. Formally, they are modeled by $s_l : \mathcal{O} \rightarrow \mathcal{O}$.

We represent dynamic and static links with association maps [1]. An association map λ specifies which objects correspond to which strands when they take part in links. To formally define the association map, we make the following assumptions:

1. The set of frames (raw data) is the set $\{1, 2, \ldots, n\}$ for some arbitrarily chosen, but fixed, integer n.

2. A strand represents the set of all frames between i (inclusive) and j (exclusive), where $1 \leq i \leq j \leq n$ are frame indices. The strand is then denoted by the pair $[i, j)$. i is the start of the strand and j is the end of the strand.

Medium	Any form of information that is manipulated by multimedia systems and can be broken down into frames (e.g., audio, video, control information for servo systems and mechanical actuators, etc.).
Frame	Basic unit of continuous medium. Operations on frames are *read* and *write*.
Strand	Immutable sequence of ordered frames—the immutability of strands means that the frames inside a strand can not be referenced as stand-alone entities from the outside. Operations on strands are *playback* (which include fast forward and rewind, etc.), *record*, *set* and *reset* the reference.
Ring	A collection of $n \geq 1$ strands. The collection is ordered and the order of the composing strands is fixed—can not be modified once the ring is created. Another attribute is meta-data.
Rope	A collection of $n \geq 2$ rings (of the same or different medium) tied together by synchronization information. Ring synchronization could be either forced (by means of a clocking device) or automatic, and is handled at lower levels.
Meta-data	Additional information about a ring (e.g., name, description, keywords, etc.). Some fields are read-only and can not be modified (e.g., frame rate, etc.).
Link	A discrete function that formalizes the relationships between objects. In this context, by discrete we mean that the function values are references to individual objects. Depending on the type of argument, a link is either *dynamic* or *static*.
Composite object	A hierarchy of objects tied together by links—a multimedia document is a composite object as well.

Table 1. Terminology.

3. We define a partial ordering $\sqsubseteq$ on the set of all strands such that $[i_1, j_1) \sqsubseteq [i_2, j_2)$ if $i_1 < j_1 \leq i_2 < j_2$. If $j_1 \neq i_2$ and $[i_1, j_1) \sqsubseteq [i_2, j_2)$, then $[i_1, j_1) \sqsubset [i_2, j_2)$.

4. The set of multimedia objects is the set $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$ for some arbitrarily chosen, but fixed, integer m.

Under the assumptions 1–4, we define the following:

DEFINITION 1
A set of strands $\mathcal{S}$ is well ordered if:

1. $\mathcal{S} = \{[i_1, j_1), [i_2, j_2), \ldots, [i_s, j_s)\}$ *for some integer* s *($\mathcal{S}$ is finite);*
2. $[i_1, j_1) \sqsubseteq [i_2, j_2) \sqsubseteq \ldots \sqsubseteq [i_s, j_s)$.

DEFINITION 2
A set of strands $\mathcal{S} = \{[i_1, j_1), [i_2, j_2), \ldots, [i_s, j_s)\}$ is solid if:

1. *$\mathcal{S}$ is well ordered;*
2. *There is no strand in the form $[i_1, i_2)$ and $[i_2, i_3)$.*

DEFINITION 3 (ASSOCIATION MAP)
An association map is a function $\lambda : \mathcal{O} \rightarrow \mathcal{R} \times \mathcal{S}$ such that for each multimedia object $o_i \in \mathcal{O}$, the strands in $\lambda(o_i)$ are a solid set.

For example,

$$\lambda(o_i) = \{(r_k, [i_3, j_3)), (r_l, [i_7, j_7)), (r_m, [i_9, j_9))\}$$

means that o_i occurs in all strands $[i_3, j_3)$, $[i_7, j_7)$ and $[i_9, j_9)$—the first member of a pair $(r_k, [i_x, j_y))$ is the ring that contains $[i_x, j_y)$.

The advantage of using association maps to model links is twofold:

- Association maps correspond to line segments on the x-axis of the Cartesian plane. From a database perspective, they can be efficiently stored by any method for storing collinear line segments [1].
- For any given object, the association map determines all the referencing strands. This allows discarding of non-referenced objects with reference-based garbage collection mechanisms.

A document containing just static links is organized in a tree-like hierarchy. Each node has a coordinate on the contents axis. A static link between two objects establishes a connection between their corresponding coordinates—see Figure 7a. However, if dynamic links are also present, this model can not represent the additional dimension, time. We define *planar nodes*, which have two dimensions and therefore correspond to line segments in the contents-time plane. For any given node, the contents coordinate is constant, like in the previous case. A dynamic link between objects establishes a connection between a time segment and a point on the contents axis—see Figure 7b.

Figure 8 illustrates a tree with nodes corresponding to static and dynamic links. For each node, we show the projection of the coordinates on the time axis. This yields a single point for static links and line segments for dynamic

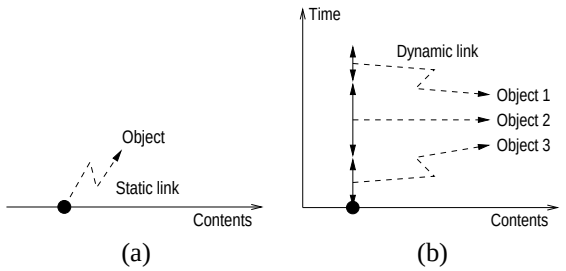

Figure 7. Tree nodes for static links (a) and for dynamic links (b).

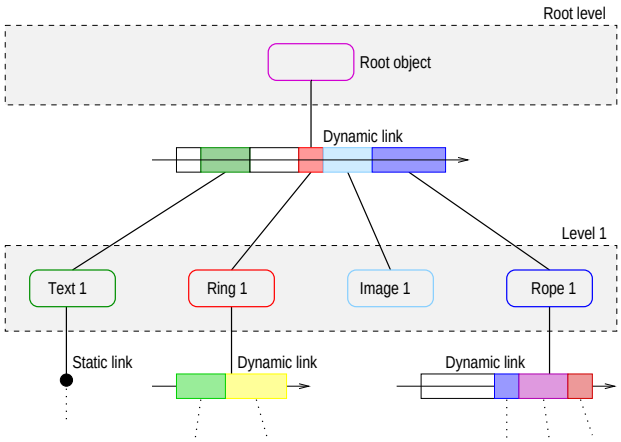

Figure 8. Tree with 1-D (contents) and 2-D/planar (contents, time) nodes.

links. The time segments which are associated with other objects are shaded, while the ones with no associated objects are left white.

Therefore, a hierarchical multimedia document with dynamic and static links can be represented as a tree with planar nodes (i.e., planar document structure). In the case of dynamic links, nodes store the information corresponding to the additional dimension (time) and represent points in a 2-dimensional space. The structure with unidimensional nodes is compatible only with static links and can not express the temporal information required by dynamic links.

The links (dynamic or static) model relationships between objects in a top-down manner. For a given object, they determine the complete hierarchy of associated objects, allowing for resource allocation and reservation, prefetching and caching.

Structuring the media as frames, strands, rings and ropes (see Table 1) not only offers a high-level abstraction for the raw data, but also allows for good performance from a low-level storage perspective. File systems designed to support continuous media (digital video and audio) by providing facilities for creating, editing and retrieving multimedia objects [11] use an identical structure and therefore no intermediate conversion layer is required.

4. Operations

We classify hyper-media document management operations into two categories: *low-level* operations which are available at the application level, and *high-level* operations available to application users.

4.1. Low-level operations

The low-level operations include basic operations for manipulating strands, such as Record, Playback, Stop, Fast Forward and Fast Rewind. They also include operations to establish or remove links associated with a certain strand. The definitions of these operations are listed in Table 2.

Record	Begins recording $n \geq 1$ multimedia strands. Recording continues until a subsequent Stop operation is issued.
Play	Retrieves a previously recorded set of $n \geq 1$ strands and processes each of them according to the medium type.
Fast Forward	A variant of Play that retrieves and processes every other k frame.
Fast Rewind	A variant of Play that retrieves every other k frame, starting with the last frame of each strand and progressing towards the first frame.
Stop	Issued to interrupt a previous command (Record or Play variant).
Set Reference	Establishes a link from a strand to a given multimedia object.
Reset Reference	Removes the existing link to a multimedia object.

Table 2. Low-level operations.

4.2. High-level operations

The entities available at the application level are temporal (e.g., strands, rings and ropes) and atemporal (e.g., text, graphics, static images, etc.) multimedia objects. Applications allow operations such as creation, composition and presentation. A document (a set of objects and a set of links—relationships—between them) can also be regarded

Link	Establishes a link between two objects. Static links require just the objects. Dynamic links require the time coordinate as well.
Unlink	Removes a previously established link.

Table 3. High-level composition operations.

as a relational database [3]. Therefore, support for content-based access operations is intrinsic. The entity types correspond to each object's media data and the meta-data associated with it.

High-level operations are divided into two groups: *general operations* which allow overall manipulation of multimedia objects (such as creation and presentation), and *content-based access operations* which allow different types of queries.

General Operations.

Creation defines the construction of high-level multimedia objects. In the case of temporal objects, the basic building blocks are strands. First they are recorded with Record and then assembled together to form high-level objects with Concatenate. Atemporal objects are imported from other applications—their creation is beyond the scope of this framework.

Composition combines simple objects (temporal or atemporal) into composite objects—Table 1. The user constructs the hierarchy by specifying links between objects. The corresponding operations (Link and Unlink) are defined in Table 3.

Presentation defines the operations that are available for exploring a composite multimedia object, such as Playback, Jump, Return, Plunge, meta-data manipulation and Query. Once the hierarchy has been constructed through composition, the user can interact with it in various ways. These operations are described in Table 4.

For example, assume that a video ring r_1 consists of three strands—see Figure 9. The first strand AC is related to an HTML document `doc.html`, and the third strand DI to another ring r_2 (audio). The user adds a link to `doc.html` in AC and a link to r_2 in DI. Therefore, the dynamic link of r_1 is defined as follows:

$$d_l(r_1, t) = \begin{cases} d_l(r_1, [\text{AC}]) & = & \text{doc.html} \\ d_l(r_1, [\text{DI}]) & = & r_2 \end{cases},$$

and the association map:

$$\begin{aligned} \lambda(\text{doc.html}) &= \{(r_1, [AC])\} \\ \lambda(r_2) &= \{(r_1, [DI])\} \end{aligned}.$$

Playback	Sequentially retrieves and presents the strands corresponding to a high-level object. Whenever a strand is associated with another object, feedback is given to the user. This operation includes all variants available at object level (fast forward and fast rewind).
Jump	Changes the context (following a link) from the object being played back to the one associated with it, advancing one level down in the hierarchy. Whenever the lower level object ends, the playback of the referencing object is resumed.
Return	Changes the context from the object being played back to the one that referenced it, advancing one level up in the hierarchy. This is not possible for the root object, which is not referenced by any other object.
Plunge	Variant of playback which does not return to the referencing object.
View data	Displays the meta-data for a high-level object.
Change data	Allows the user to change the meta-data fields which are not read-only.
Query	Provides services for content-based access.

Table 4. High-level presentation operations

When r_1 is played back, the application provides feedback about the links within AC and DI. Assume Jump is selected at $t = B,\ A \leq B < C$. The first link is taken and the presentation changes from r_1 to `doc.html`. Whenever Return is selected (because this object does not have a temporal dimension, its presentation needs to be terminated by the user), the playback of r_1 resumes at B. Next, assume Jump is selected at $t = E,\ D \leq E < I$. The second link is taken and the audio corresponding to r_2 is played back. If Return is selected at $t = G,\ F \leq G < H$ (before completion), then the presentation of r_2 ends and r_1 resumes at E and finishes at I.

Content-based Operations. Queries can be classified according to different criteria. We include examples for several query types.

Type of information involved in a query. Depending on their type, some queries require an additional processing step for all objects in a hierarchy, similar to the indexing stages described in [6].

Queries on object information take into account just

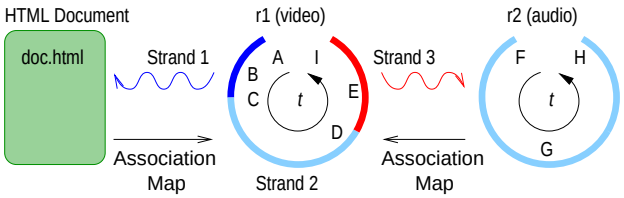

Figure 9. Example of hierarchy with 3 objects (two rings and a document).

the media data. Unlike other systems [6], they do not require additional meta-data information and therefore work on any document structure.

```
QUERY 1
Return all objects that contain a
given strand
```

Queries on meta-data require objects to contain valid information (no `NULL` values[2]) in the meta-data fields that are selected.

```
QUERY 2
Return all objects created after
August 29, 2000
```

Queries on mixed information take into account all the information about objects. Again, they require meta-data to be present.

```
QUERY 3
Return all objects that contain a
given strand and were
created after August 29, 2000
```

Granularity of results. Object-level queries return entire objects (e.g., text document, rings, ropes, etc.). Sub-object-level queries return parts of objects (e.g., a section of a text document, strands, etc.).

5. Partial verification and results

In this section we describe a partial implementation of the framework. First we argue that, although partial, the verification does not restrict the generality and demonstrates that the framework is feasible. Then we cover several implementation-specific details. We conclude with an experiment and results.

[2]If an object does not define a particular meta-data field, then the value is considered `NULL`.

5.1. Multimedia objects

As stated in Section 2, our framework regards the multimedia objects in a consistent way, independent of the media type. However, one of the novel aspects introduced in the previous sections is dynamic links, which are defined for temporal objects—Section 3. Existing systems already implement static links (e.g., World-Wide Web) and therefore their verification is not required. Consequently, in order to implement and test dynamic links, we use just temporal objects.

We restrict the verification to video information. Besides supporting dynamic links, this choice also allows testing video to video transitions, which just a few other systems support [2]. Restricting to one media type does not decrease the generality. Rather, it simplifies the implementation. The details associated with manipulating other types of media (e.g., audio, text, etc.) are left aside.

5.2. Design patterns

An essential framework attribute is a high degree of flexibility, such that the framework is usable for a wide range of applications. Design patterns help to achieve this objective, emphasizing the reuse of successful designs and architectures [5].

Our document architecture is an instance of the *Composite* pattern: each element of the hierarchy could be either a simple object or a composite object. *Composite* ensures an uniform document interface and enables for any combination of media types. It also allows to apply the *Visitor* pattern, which decouples the document's structure from the high-level operations—Section 4.2. Consequently, defining new operations does not require changes on the document side. This flexibility is important in the context of large document databases, where recreating the entire database is an expensive operation.

5.3. Implementation

For our prototype system, we have chosen an object-oriented architecture and design. A high-level toolkit like Motif [7] is the best candidate for the user interface, which focuses on ease of programming and event-driven processing. Likewise, media data processing requires fast and reliable code. C++ and the Standard Template Library [10] are an excellent choice for these requirements.

The strands are motion-JPEG compressed video sequences. The absence of temporal prediction allows for frame-level granularity.[3] For our implementation, we use

[3]The framework does not depend on strand granularity. In case temporal prediction is used (e.g., MPEG), the granularity is higher (GOP for MPEG). Because the strands are immutable, this does not affect higher-level objects.

this format because of the availability of hardware-assisted operations,[4] which allow for fast processing.

Processing of the media data associated with digital video has to take place in a timely manner, according to the soft deadlines typical to multimedia systems [13]. The scheduling solution used in our prototype relies on the X timer callbacks [7, 14]. Although not optimal, this approach has the advantage of being portable and has been adopted by other multimedia applications [8]. However, our implementation is generic and could be easily modified to take into account real-time services, in case they are provided by the system.

5.4. Results

To verify the framework, we performed a series of experiments with the system described in 5.1 and 5.3. The experiments involve three 5 minute video rings, r_1, r_2 and r_3. r_1 consists of 4 strands: $r_1 = \{s_1, s_2, s_3, s_4\}$ and its dynamic link is: $d_l(r_1, [120, 180]) = r_2,\ d_l(r_1, [240, 300]) = r_3$ (time intervals are in seconds). The image size corresponding to each video is 320 by 240 pixels, at a rate of 20 frames/second.

We were interested in observing how intuitive the dynamic link between different video objects is for the user and how fast our system can descend in the hierarchy (change the context from one video ring to another). The program executed on an HP workstation and the video objects were served by an HP multimedia server, via NFS. We ran about 50 experiments and used as metrics the delay for Jump and Return operations. The average measured values are 2.5ms and 1.2ms respectively.

The results show that the context changes are fast (millisecond range), hence the transitions from one ring to another are smooth and satisfy the user's visual perception. The difference between the two measured times are caused by the different low-level operations associated with each action (i.e., file open, file close) and are implementation-dependent.

6. Summary

We presented a framework for advanced link management in hyper-media documents. The framework is based on an object-oriented document architecture, supports non-linear transitions between multimedia objects and has intrinsic support for content-based access. Non-linear transitions are formally represented by *dynamic links* and *static links*. To represent the additional temporal information associated with the dynamic links, hyper-media documents are structured as trees with planar nodes. A prototype system shows that: (1) the framework is viable for hyper-media documents which contain temporal objects; (2) the dynamic link abstraction is feasible; and (3) the overall system performance satisfies the user's visual perception.

[4] We use hardware from Parallax Graphics, Inc.

References

[1] V. S. Subrahmanian at al., "*The Advanced Video Information System: data structures and query processing,*" ACM-Springer Multimedia Systems Journal, 1996.

[2] Roy H. Campbell et al., "*Real Time Video and Audio in the World Wide Web,*" Fourth International World Wide Web Conference, 1995.

[3] Ramez Elmasri, Shamkant Navathe, "*Fundamentals of Database Systems,*" Addison-Wesley 1994.

[4] Myron Flickner et al., "*Query by Image and Video Content: the QBIC System,*" IEEE Computer, September 1995.

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "*Design Patterns—Elements of Reusable Object-Oriented Software,*" Addison-Wesley 1995.

[6] Venkat Gudivada, Vijay Raghava, "*Content-Based Image Retrieval Systems,*" IEEE Computer, September 1995.

[7] Dan Heller, Paula M. Ferguson, "*Motif Programming Manual,*" O'Reilly & Associates Inc., 1994.

[8] Christopher J. Lindblad, "*A Programming System for the Dynamic Manipulation of Temporally Sensitive Data,*" MIT/LCS/TR–637, 1994.

[9] Thomas Meyer-Boudnik, Wolfgang Effelsberg, "*MHEG: An Interchange Format for Interactive Multimedia Presentations,*" Computer Science Technical Report, University on Mannheim, 1994.

[10] David Musser, Atul Saini, "*STL Tutorial and Reference Guide,*" Addison-Wesley 1996.

[11] P. Venkat Rangan, Harrick M. Vin, "*Designing File Systems for Digital Video and Audio,*" Proc. of the 13th ACM Symposium on Operating Systems Principles, 1991.

[12] Franc Solina et al., "*Slovenian Virtual Gallery,*" `http://razor.fri.uni-lj.si:8080/gal`.

[13] Ralf Steinmetz, Klara Nahrstedt, "*Multimedia: Computing, Communications & Applications,*" Prentice Hall 1995.

[14] Douglas A. Young, "*The X Window System Programming and Applications with Xt,*" Prentice Hall 1990.