Software Tools for Content Based Hypervideo

João Martins, Nuno Correia, Nuno Guimarães
{jota, nuno.correia, nuno.guimaraes}@ inesc.pt
INESC, R. Alves Redol 9 - sala 636, 1100 Lisboa, Portugal

ABSTRACT

In most current hypermedia systems, video and audio data are considered still as navigationally opaque data, without an accessible internal structure, and where no hyperlinking is performed. This article describes an object model, realized as a library of C++ classes, for the development of annotation and navigation applications in video spaces. The object model considers not only aspects related to the structuring of these materials, but ways in which they can be integrated with others (images, text, etc), enriched with automatically extracted information, as well as presentation and interaction aspects.

Keywords: video hyperlinking, video navigation, movie-only spaces, automatic content extraction.

INTRODUCTION

Current hypermedia systems allow the integration of heterogeneous materials like text, images, video and audio. Only for the first two cases, however, are the interaction mechanisms and presentation forms well studied, and is taken into account the material's internal structure. The World Wide Web illustrates this fact well. Video and audio are, on the other hand, still viewed and manipulated as a whole, as blocks of opaque data, without an accessible internal structure. This is due to several problems, not only related with the larger complexity of these types of information, but also with process, storage, and access problems, and largely with its intrinsically dynamic nature.

This work focus the creation of a set of objects capable of making the internal structure of this information (with a special emphasis to video materials) visible, as well as the study of ways to interact with this data, the presentation's aesthetics, and guiding this interaction by the materials' contents. All this is supported by mechanisms that give an accessible structure to the video, allowing references to it as a whole, in temporal terms (an image or block of images of a sequence) or in spatial terms (a region in an image or block of images of a sequence). Moreover, we sought to make it easier to build a context to the video, enriching it with the result of automatic content extraction algorithms like the ones described in [Olive97] and [Marti97]. The temporal nature of the material is, however, the essential difference between other types of media and video or audio, and the most important aspect to consider.

RELATED WORK

The processing and navigation in video spaces has already been studied in technical literature, and some of the related work is described below.

The Amsterdam Hypermedia Model [Hardm94] is a framework that extends the Dexter model [Halas90], but adds the notions of time, presentation attributes, and link context. The introduction of time in what was a static model allowed the development of the notions of *collection*, in which its components are presented simultaneously, and *synchronisation*, in which the order of presentation of several elements is defined. The presentation attributes define the values of some global attributes of the system, e.g., the volume of the audio pieces, that no longer need to be specified individually for each element. Finally, the link context solves the problem of deciding what happens when a link is followed, defining what will happen to the elements being displayed, and what elements are presented after following the link.

In another fundamental work [Liest94], the aesthetic and rhetorical aspects of integrating video in

hypermedia were considered. The articles stresses the importance of achieving a smooth integration of the different kinds of materials, and specifically, text and video, where the first is "actively" read, and the second "passively" viewed. Some rules were derived from the basic idea that the techniques used in movies to achieve continuity are somehow usable in this environment, which in turn lead to using icons to signal the presence of links in video, and giving dynamic characteristics to text, changing it into a video.

In the ConText system [Daven95], a new way of displaying information, the Evolving Documentary, is described, where the objective is to support a narrative approach to browsing a multimedia database, with a partnership between the viewer and the presentation engine. The browser suggests narratives basically by taking advantage of the descriptive associations, thus changing an author's task to the selection of those materials and the construction of their descriptions. This system satisfies users who do not know either what is in the database, or what they want to see, and to whom a retrieval-by-query approach is not applicable. During browsing, the user interaction is not required, as a dynamically-attributed weight system allows the browser to determine the order in which the elements in the database shall be presented.

Navigation in movie-only hypermedia, or hypermovies, is considered in [Giess95]. The common navigational concepts of browsing, maps, history links and tours are explored and enhanced for this new space, and the importance of traditional film theory and of reducing user disorientation is once again stressed. The author uses the basic concept of track of a movie, where a track provides information about one of its specific aspects, as a structuring concept, and links are attached not to movies or segments of movies, but to segments of tracks. Several tools where implemented to illustrate the ideas mentioned. The importance of the path mechanism in hypermedia, somewhat overlooked in this article, is extensively developed in [Zellw89], where paths are considered to be first class citizens of hypertext systems. A path is defined to be the presentation of successive entries, ordered in a way that all or most of the navigation decisions are made by the author in advance. These entries can be simply nodes in the hypermedia graph, or active nodes, where active nodes might be used to animate pictures, perform computations, etc. Three sequencing models are introduced: the sequential path, the branching path, and the conditional path, where a condition is evaluated to decide which way to go, and three types of playback control : single-stepping, automatic, and browsing.

Marc Davis, in Media Streams [Davis96], presents an extensive iconic annotation language, while arguing that keyword annotations are not enough to maintain a consistent and scalable representation of the salient features of video content, and don't allow the creation of reusable video archives. He also presents a minimal set of properties with which the media should be annotated, and proposes that the segmentation of movies into smaller units - clips - [Zhang97] shouldn't be fixed and hardwired, like in most current approaches, but rather obtained through a series of multi-layered annotations with precise time indexes, allowing for different segmentations of the material. The concepts are illustrated with several applications.

The HyperCafe [Sawhn96] is an experimental hypermedia prototype, developed to illustrate what a complete hypervideo system can be like, exploring forms of display the information, how to represent links opportunities, etc. The user is placed in a virtual café, composed primarily of video clips of actors involved in fictional conversations, and is left free to navigate in that space, following links and visiting nodes in the narratives. The HyperCafe is not a development toolkit, but a closed environment to illustrate a set of concepts.

In the WebStage [Yamag97], a active media enhanced WWW browser, the television metaphor is used to display Web pages, in an attempt to encourage passive (reading, as mentioned in [Liest94], is an active operation) users to access the WWW. The main features identified in the TV metaphor were: a visually and auditory enhanced appearance; simple operation; continuous output (this element also having been explored in [Sawhn96]), and these principles led to a system that converts conventional HTML pages into multimedia presentations. The three benefits desired were reducing the cognitive overload of reading text, improving the comprehension of abstract information in a glance, and improving the ability to distinguish between different types of information.

In the field of automatic content extraction, relevant work was done in [Zhang94], where television news reports were parsed with the aid of a *a priori* model of the programs' structure. The news video was first partitioned into shots, using cut detection algorithms, and then the shots were classified into anchorperson or news shots. These techniques were further refined and studied in works like [Zhan97] and [Oliv97], where more processing algorithms for video are presented, allowing a better detection of cuts (defined to happen when the difference between consecutive frames is above a given threshold), the detection of gradual transitions between shots (fades, dissolves, and wipes), motion based segmentation, camera

operation (still images, pans, tilts, zooms) and object motion analysis, key frame extraction (frames that can be used as representatives of an entire shot), characterization of the lighting conditions (outdoors or indoors scene), scene segmentation (decomposing of an image into its main components), which includes caption extraction, edge detection, etc.

There is also some work on the field of automatic audio content analysis audio. [Pfeif96] introduces several frequency-domain algorithms to aid in the detection of violence. These are able to recognize silences, speech, music, noise, and sounds indicative of violence, like gun shots, explosions, and cries. [Marti97] does similar work, distinguishing between music, silences, and speech, but using only time-domain algorithms.

HYPERVIDEO MODEL

Our model for Hypervideo includes several of the concepts mentioned above. We want to be able to represent the media in our programs, preferably hiding data format details from the user while still compromising efficiency as little as possible, and manipulate them uniformly. Using this media representation, we wish to be able to specify links or link opportunities from this information to other data nodes, be it dynamic or static media. This will imply a reference mechanism, which should be able to specify both temporal and spatial link opportunities. Also, we wish to use the WWW as a potential data source, and as such include some support for this medium. Simple textual/keyword annotations must also be supported, in what should be an extensible structure. The annotations and links from a given media object are to be considered as the meaning, or context, of this object, and as such be stored in the same structure. Also, for each media object there must be the possibility of having several different contexts.

As to the viewing, it must be possible to display all the different kinds of media, and the two kinds of link mentioned above must be included. These views should implement video-audio synchronization, and as such are the most platform-dependent module. These views will also be able to display the context of a media (all its annotations and links), and a path, a journey through a set of the stored media. The notion of navigation history is to be implemented as a particular case of a path, but is structure which is created based on the user's actions.

Finally, it should be possible to augment media context's by supporting the creation of content-based annotations, extracted by automatic processing algorithms, and perform media conversion, the smooth integration of different kinds of media into one same navigational and aesthetic environment.

All of these concepts are included and available in our application-development toolkit.

PROGRAMMING MODEL

Several C++ classes were created with the purpose of implementing the model described above. The four basic modules are described below.

Locators

Locators are the simplest and most basic element of the environment, and are used to address the materials. There are two main kinds of locators, Region Locators and Time Locators.

Region locators allow for references to spatial areas in the materials, be it words or sentences in text, regions in images/ videos, frames in audio, etc. Time Locators qualify the Region Locators, and are used only when representing data with temporal characteristics, like video and audio. These allow for references to point time locations or to interval locations, and are an extension of the MADE (Multimedia Application Development Environment) Project synchronization model (see [MADE92] and [Corre94]).

Annotations

Now that we have the locator objects to make references, we can add the annotations, which are used to associate information to the materials. There are several classes of annotations, but the two most elementary ones are Label Annotation and Action Annotation. Objects of the first are used annotate the material with strings of text, with subclasses of this being used to refine the associated data. Action Annotations are used to denote actions that must be performed when the material is accessed. Two examples of this are *play* and *preview*.

Annotations can be grouped in the Context class, which is also associated with a Media Object. The Context of a Media Object gives the meaning of that Media Object, and for the same Media Object there can be several Contexts, built according to the individual goals of different applications or uses, corresponding to different meanings or points of view. This class effectively simultaneously implements the concepts of timeline, path, and hypermedia graph through a node. The context contains, additionally, the specification of what

should be done when the display of the media object ends: loop back to beginning, jump to another media object/context, or just stop.

Media Objects

The Media Object classes are used to model the materials to manipulate. Again, two elementary subdivisions were made, into Simple Media Objects and Stream Media Objects. We consider Stream Media Objects to consist of sequences of Simple Media Objects. In this view, an image in a stream of video is a simple object, as is a word in a text, a pixel in an image, or a frame in an audio stream. The full hierarchy for the Media Objects in depicted in the following diagram (Fig. 1).

The hierarchy is divided in two layers. The top layer classes are platform independent, and completely generic. The bottom layer classes, however, deal with specific formats, and as such are platform dependent. These classes provide an easily extendible interface for the I/O operations, and "convert" these formats into the object model.

The Stream and Simple Media Objects sub-hierarchies are interconnected by another set of classes, the Iterators. Each of this classes is associated with one class in the Stream hierarchy, and gives the application access to the individual elements they contain.

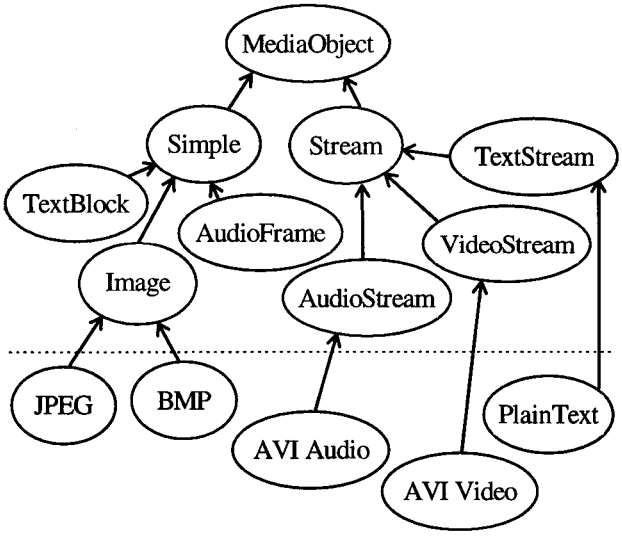

Figure 1 - Simplified class inheritance diagram of the Media Object hierarchy. The classes above the dotted line are platform independent, the classes below the line are format dependent.

The last of the class in this section is the Media Factory. The Media Factory objects create Media Objects from *Uniform Resource Locators*, or URLs [W3C], after identifying its type. The use of URLs to identify objects allows a simple integration with Internet-based applications, and are presently a de-facto standard. The Media Factory provided is capable of dealing with all the formats and classes initially implemented, and is easily extended to deal with extra formats.

Views

The Views are the most complex of all the classes implemented, and are mostly platform dependent. The main hierarchy is shown in Fig. 2 and described below. Generally, views display information, represented in the application as Media Objects, and each view understands a specific type of media information. Based on this assumption, we created an hierarchy of views that closely mirrors that of Media Objects.

The top-most class is the abstract class View. This class provides the general characteristics to be implemented and existent in all the subclasses, and contains auxiliary methods susceptible of being useful in the subclasses. Below this are four classes, used to display the major four kinds of materials considered in this work: text, audio, video, and image. Each of these classes has subclasses that implement specific ways of viewing the information. For example, an image can be displayed in colour or black and white, a video segment can be shown in the "movie paradigm", where images are displayed superimposed and in rapid succession, or we can see a movie of contours, or, more generally, a movie with the results of applying some algorithm to the images before displaying. All the views must implement several common navigational-related methods: *play*, *stop*, *pause*, and *resume*.

The bottom-level classes of the hierarchy are the Preview classes. In this classes, the *play* method starts the display of a "short" version of the Media Object, a kind of abstract. In a video, this might correspond to a small sequence of reduced frames shown repeatedly [Brøn91], or the first few words of a text document.

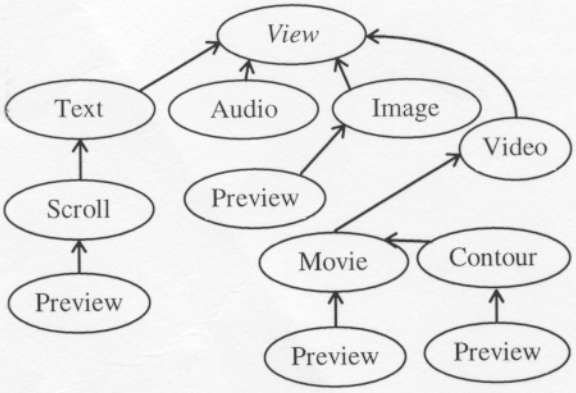

Figure 2 - The View class inheritance hierarchy, simplified. The names of the classes were shortened for clarity. For example, the full name of the Preview class under Scroll is actually ScrollTextPreview, etc.

The View classes are, generally, assigned the function of displaying the information represented by the media object they contain. An extra element is, however, relevant in these classes. In our object model, we defined one of the views that can coexist in an application as the Master view, the other being slave views. This object is then responsible for the coordination of the whole synchronisation and navigational processes. It is this view that responds to user interaction with the application, reads the timeline represented by the Context, assigning and/or creating other views (or previews) to windows in the display area, and keeping the navigation history. The master-slave concept is here used not only in its low-level meaning, where it is used to synchronize, e.g., audio and video (the audio being the master stream), but also in a high-level, or semantic, meaning, where a master view displays the most relevant information, the slaves corresponding to "accessory" information.

These classes must have additional data, namely an object of the Geometry Manager class, used to manage the screen locations of the information being displayed, and the already mentioned Context. To finish, these objects are also responsible for the last of the important element in the system, described below.

Paths / History

A path is, as mentioned in [Zellw89], the ordered traversal of links. Our model implements the sequential and branching paths by extending the concept of Context. Whereas the Context defines a full navigation graph passing through a specific Media Object, the path defines a subset of this graph, with less (if any) link opportunities, and a predefined content, using parameterized play annotations to represent the entries in the path.

The user's interaction with the system, together with the initial Context or Media Object which was assigned to the view, defines a navigation history. This simple record of the user's actions, performed on request by the view, is easily converted to sequential paths, and can be later replayed. This makes an author of every user.

The concept of *path* allows the implementation of yet other relevant concept in systems of this kind: the Guided Tour (further developed in [Halas87] and [Hammo88]), a kind of virtual journey through a subset of all the information stored in a database or multimedia system, where all the nodes to visit were defined in advance by the author.

SAMPLE APPLICATION

A sample application was implemented to allow users to navigate in video spaces. This application, as well as the programming model described above, were implemented using Microsoft Visual C++ 4.2, under Microsoft Windows 95. The general look of the application is shown in Fig.3.

The application allows navigation in a hypermedia space where video is the most important media, but audio, text, and still images are also possible nodes. The contexts for these spaces were built using algorithms implemented either under Windows 95 or Sun Sparc Unix environments, using the Context's I/O platform- independent interface, and some hand-made textual annotations were also added. All these materials are stored in a local filesystem.

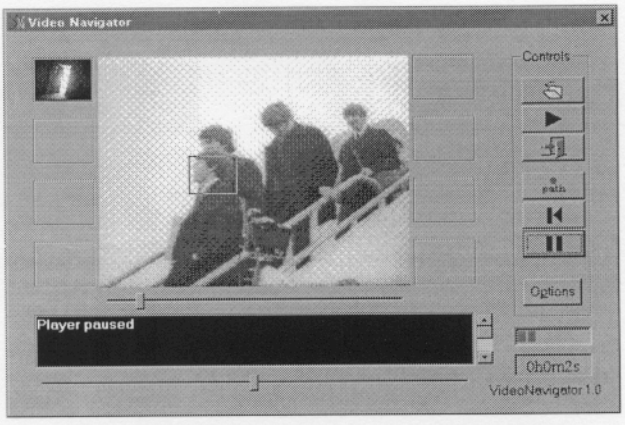

Figure 3 - The Video Navigator.

The player's window is divided into several regions. (This region distribution is not fixed, as a geometry is not imposed on the applications). The main region is currently showing a video about the *Beatles*, assigned to a MovieView object, with a rectangle labeled "1"

over one of its members. On the sides there are eight regions, all of them empty except for the one on the top left, region one, which is showing a preview of another movie (a door opening and closing), and represents a link opportunity corresponding to the region drawn over the image. The eight side regions are Preview regions (corresponding to Preview objects, in the case of region one, a MoviePreview). The application dynamically creates or assigns an appropriate Preview object to each of the regions. Clicking this window, or the rectangle in the main area, follows the link.

In the bottom there is a larger area, which is destined to display textual annotations (a TextView object). Finally, there are two sliders, one of them indicating the progression on the current main movie, and the one on the bottom indicating the progression on the context. Both of these can be manipulated to advance or rewind the movie or context, respectively.

The navigational controls are in the top right, and include *open*, *play*, *exit*, *record path*, *go back*, and *pause*. Below these there's still a progress indicator, and a text label indicating the time progress of the movie. The *options* button leads to an extra window that allows the user to access a series of application options, and the author mode of the application - the annotator -. The open button allows the user to select an AVI file, a Context, or a Path.

This application allowed us to study the feasibility of the implementations, and do some preliminary considerations about the interface's user friendliness. The player has been evaluated by several users, and although considered easy to use, some complained about the high level of mental concentration required in deciding about whether or not a link should be followed. This finding lead us to impose a minimum time length for the links over video.

CONCLUSIONS AND FUTURE WORK

This paper describes the main components of our programming model for hypervideo, and shows how it can be used to build hypervideo annotation and navigation applications. A video navigator application, coded to demonstrate the concepts, is also briefly described. This application was tested with videos that are presently annotated only with the results of video processing algorithms, which extract information about cuts, gradual transitions, camera movements, scene's main components, and captions [Olive97], and also some simple audio processing techniques, which extract information about silences and performs very accurate speech-music discrimination [Marti97]. Although there is a lot of work done in the field of audio processing and specially speech recognition, these imply complex and heavy computations, and are out of our research scope, which focus mainly in video.

Real hypervideo systems are, in our view, still in early stages of development, and require a high cognitive effort on the users, as well as high-bandwidths for transmission and storage. For these reasons, the authors consider that further work must be done before this kind of applications reaches the easy-of-use and integration level of the World-Wide Web, limiting it to professional production environments and/or expert users.

We are currently working on finishing the annotator module of the player, and designing a structure visualization application. The former will allow for easy human-based annotation of the materials and will integrate the automatic augmenting of annotations with the results of extra video processing algorithms. The later will permit the user/author to have a general view of the hypermedia network, and use it as a navigational aid. A query mechanism is also under consideration. Preliminary work has also been done on porting the player to the Java programming language, allowing for its demonstration on the WWW.

REFERENCES

[Brønd91] Brøndmo, H, Davenport, G: Creating and Viewing the Elastic Charles - a Hypermedia Journal, in McAleese, R and Green, C (eds.) *Hypertext: State of the Art*, Oxford:Intellect, pp. 43-51, 1991.

[Corre94] Correia, N, Guimarães, N: Time and Synchronization Objects in Multimedia Application Construction. In *Proceedings of the Fourth Eurographics Workshop on Object-Oriented Graphics*, Sintra, Portugal, 1994.

[Daven95] Davenport, G, Murtaugh, M: ConText : Towards the Evolving Documentary. In *Proceedings of ACM Multimedia*, pp. 381-389, 1995.

[Davis96] Davis, M: Media Streams: An Iconic Visual Language for Video Representation. Originally published in *Readings in Human-Computer Interaction: Toward the Year 2000*, available online at *http://web.interval.com/papers/mediastreams/.*

[Geiss96] Geissler, J: Surfing the movie space: advanced navigation in movie only hypermedia. In *Proceedings of ACM Multimedia'95,* San Franscisco, USA, 1996.

[Halas87] Halasz, F, Moran, T, Trigg, R, NoteCards in a nutshell, in *Proceedings of the ACM CHI+GI'87 Human Factors in Computing Systems and Graphics Interface Conf.*, pp 45-52, Canada, April 1987.

[Halas90] Halasz, F, Schwartz, M: The Dexter Hypertext Reference Model. In *Proceedings of the Hypertext Standardisation Workshop*. National Institute of Standards and Technology, USA, January 1990.

[Hammo88] Hammond, N, Allinson, L: Travels around a learning support environment: rambling, orienteering or touring? In *Proceedings of the ACM CHI'88 Conference*, Washington DC, pp. 269-273, April 1987.

[Hardm94] Hardman, L, Bulterman, D: The Amsterdam Hypermedia Model Adding time and context to the Dexter Model. *Communications of the ACM*, 37(2):50-62, February 94.

[Liest94] Liestøl, G : Aesthetic and Rhetorical Aspects of Linking Video in Hypermedia. In *Proceedings of ECHT'94*, pp.217-223, September 1994.

[MADE92] MADE (EEC funded Esprit Project). MADE 1 (EP 6307): Technical Annex, March 1992.

[Marti97] Martins, J, Goulão, M, Oliveira, I: An Experiment in Television , In *Proceedings of RECPAD'97, The 9th portuguese conference on Pattern Recognition*, pp. 209-212, Coimbra, Portugal, 1997.

[Olive97] Oliveira, I, Correia, N, Guimarães, N: Image Processing Techniques for Video Content Extraction. In *Proceedings of the Fourth Delos Workshop*, San Mineato, August 1997.

[Pfeif96] Pfeiffer, S, Fisher, S, Effelsberg, W: Automatic Audio Content Analysis, University of mannheim, Department of Computer Science, *Technical Report TR-96-008*, 1996. Available online at *http://www.informatik.uni-mannheim.de/~lienhart/papers/tr-96-008.ps.gz* .

[Sawhn96] Sawhney, N, Balcom, D, Smith, I: Hypercafe: Narrative and aestethic properties of hypervideo. In *Proceedings of Hypermedia '96*, 1996.

[W3C] Extensive information about the WWW and URLs is available online on the Internet at *http://www.w3.org/*, the WWW Consortium pages.

[Yamag97] Yamaguchi, T, Hosomo, I, Miyashita, T: WebStage: An Active Media Enhanced World Wide Web Browser. In *Proceedings of the ACM CHI'97*, available online at *http://www.acm.org/sigchi/chi97/proceedings/paper/ty.htm* .

[Zhang94] Zhang, H, Yihong, G, Smoliar, S, Yong, T.C.: Automatic Parsing of News Video, in *Proceedings of the IEEE ICMCS'94 Conference*, Boston, USA, 1995.

[Zhang97] Zhang, H: Video Content Analysis and Retrieval, in *Handbook on Pattern Recognition and Computer Vision*, World Scientific Publishing Company, 1997.

[Zellw89] Zellweger, P: Scripted Documents: A Hypermedia Path Mechanism. In *Proceedings of ACM Hypertext '89*, November 1989.