Structured Video: A Data Type with Content-Based Access

Andrzej Duda[†]
Ron Weiss

September 1993

MIT/LCS/TR-580

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

[†]Also with Bull-IMAG Systèmes, and INRIA

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract DABT63-92-C-0012.

Keywords: video indexing and searching, video databases, content-based retrieval, video algebra

Abstract

We describe *structured video*, a general video data model allowing free form annotation, composition, and content-based access to video segments. The structured video abstraction provides an efficient means of organizing and manipulating video data by assigning logical representations to the underlying video streams and their contents. We introduce a video algebra as a means for combining and expressing temporal relations between structured video segments. Key aspects of the model have been implemented as part of the *Structured Video System* project. The system offers a query based interface for browsing and playback of relevant video segments. The structured video browser uses the logical representation of the video data to provide viewing methods based on the ascribed temporal characteristics of the video. The structured video system provides interesting and diverse ways to efficiently access and manipulate large video collections. Experience from the prototype implementation is reported.

1 Introduction

The volume and unstructured format of digital video data make it difficult to manage, access, search and browse collections of digital video. The problem is becoming more acute as digital video becomes ubiquitous and as more video collections become available. Providing content-based access to digital video can alleviate these problems and motivate broader use of video resources.

Content-based access requires extracting descriptive information from video data. Unlike textual data, digital video does not support automatic extraction of semantic information for indexing: image and speech recognition is hardly feasible. Instead, other forms of information such as text captions or structural and temporal relationships between video segments can be extracted from the raw video and used for indexing. Additionally, manually entered information can describe video segments. The descriptive information must be associated with raw video data in a rich, structured, semantic representation. This representation provides the means for efficient content-based access, organizing video data into collections and supporting hypermedia links to video data.

Consider for example a large collection of TV broadcasts. A user may want to find and view all video segments reporting on Hillary Clinton and health care reform. Moreover, the user may want to see the context in which the segments have appeared: in the headline news, in a talk show or on Saturday Night Live. Once a given segment is found, the user can also view the surrounding video context: the news anchor's introduction, follow-up audience questions in the talk show, or the next comedy skit. A system that supports such requests must both understand the semantic structure of the video and contain indices that allow fast discovery and retrieval of relevant information. For the above example, the system needs to store and efficiently locate attributes such as the people in a scene, the associated verbal communication for each video segment in the collection, and the relationships between segments.

Another example is the domain of interactive personalized movies where a user can create her own story by choosing to explore different possible plot threads. This can be accomplished by logically structuring the video and allowing the user to choose segments based on interaction or an *a priori* specification.

This paper introduces a system based on the *structured video* data model that supports high level semantic descriptions of video footage. The system

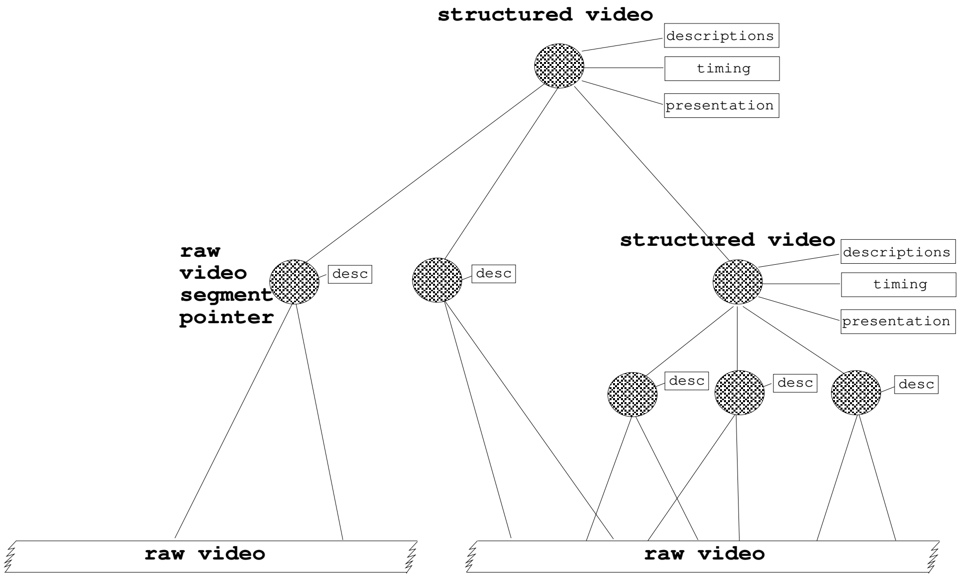

Figure 1: Structured Video Data Model

also provides flexible associative access to video segments. The structured video abstraction provides an efficient means of organizing and manipulating video data by assigning logical representations to the underlying video streams and their contents. This data model is composed of structured video nodes and pointers to video segments. The nodes can nest and form a directed acyclic graph. They include descriptive information, temporal scheduling information, and a semantic structure (see Figure 1). The structured video is indexed in a manner that preserves the correspondence between video segments so that all relevant segments and their neighbors can be efficiently found.

1.1 Related Work

In related work, Davenport, Aguierre Smith and Pincever [6, 7] implemented a video annotation system. It uses the concept of *stratification* to assign descriptions to video footage, where each stratum refers to a sequence of video frames. The strata may overlap or totally encompass each other. Strata are stored in files and can be accessed using simple keyword search. A user can find a sequence of interest, but cannot easily determine the context in which the video sequence appears. A drawback of the stratification mechanism is the absence of relationships between the strata. Our system provides a full hierarchical organization of video footage that permits flexible browsing.

Little, et al. [3] implemented a system that supports content-based retrieval of video footage. They define a specific data schema composed of *movie, scene* and *actor* relations with a fixed set of attributes. The system requires manual feature extraction, and then fits these features into the data schema. Queries are permitted on the attributes of movie, scene and actor. Once a movie or a scene is selected, a user can scan from scene to scene beginning with the initial selection. Their data model and Virtual Video Browser are limited for several reasons. First, descriptions cannot be assigned to overlapping or nested video sequences as is accomplished in the stratification model. Second, the system is focused on retrieving previously stored information and is not suitable for users that need to create, edit and annotate a personally customized view of the video footage. For example, users cannot create a new movie from the collection of scenes that are returned as a result of a query. Moreover, the browser does not support queries based on the temporal ordering of scenes.

Swanberg, et al. [8, 9] defines an architecture for parsing data semantics from the video stream. It uses a fixed set of video elements (including a *shot* and an *episode*), and is not suitable for free form modeling of the complex relations between video segments.

Multimedia authoring systems such as CMIFed [12] have rich structuring primitives for multimedia documents, but fail to address the structure of the video data itself.

Hamakawa and Rekimoto [2] propose a multimedia authoring system that supports editing and reuse of multimedia data. Their system is based on a hierarchical and compositional model of multimedia objects. It allows the user to mark objects with a title at a certain point in time. However, it does not support a fully functional free form annotation mechanism that enables subsequent content-based access.

The MHEG [5] multimedia and hypermedia standard is intended for final formatted documents, and lacks mechanisms for content-based access, editing and annotation of the multimedia data.

In our approach, the context of video segments is represented in the organization of and hierarchical relations between the structured video nodes. As in the stratification model, segments may overlap and encompass each other. However, we generalize this concept by enabling strata to be nested and thus have hierarchical organization. The result of a query may form new structured video that can be played back or manipulated by the user. In

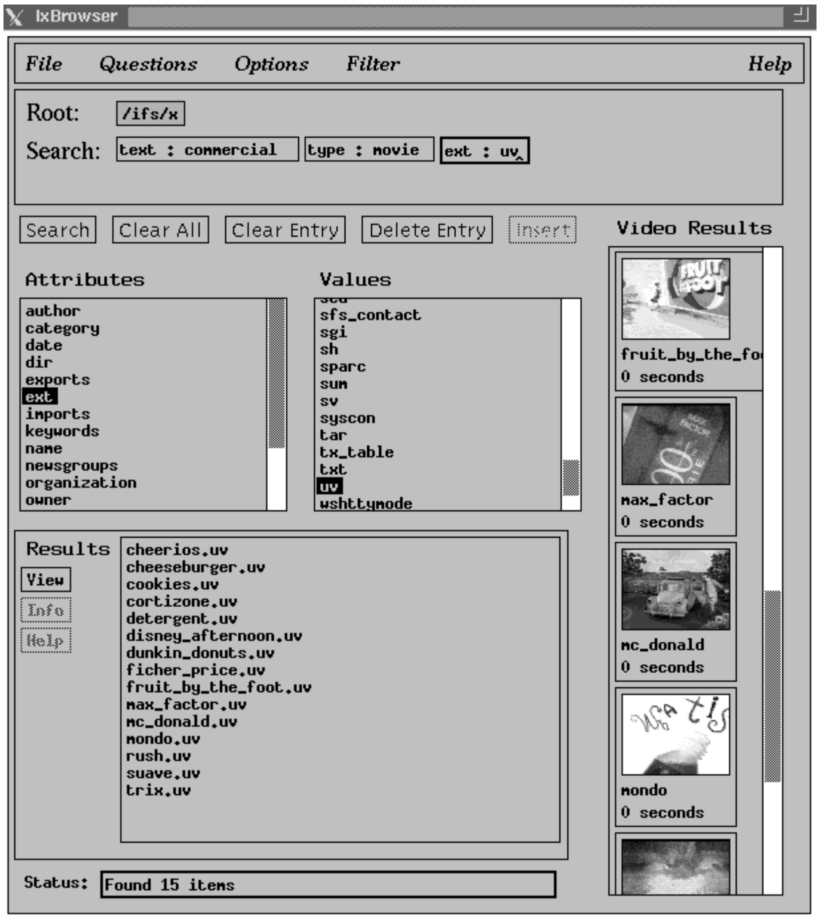

Figure 2: Structured Video Query Interface

addition, the system allows an extensible schema for representing video: the user can define new attributes and annotations that the system will index.

Our data model allows users to compose logical video streams by structuring raw data into video segments and then describing the temporal relations between the segments. In addition to content-based access, the system allows browsing. The user can explore the hierarchical structure to understand how the segments are organized. For example, a user can find an interesting segment and then look at the encompassing sequence that contains it. Furthermore, users can create their individual interpretations of existing footage by defining new structured video nodes. Hypermedia links can be instantiated in structured video nodes so a user may traverse links to other related video segments that exist in different collections. The structured video data model can also be a base for interactive movies by providing conditional pointers to other video segments.

We have implemented a structured video system that extracts video attribute information and supports content-based access, as well as video playback. The system offers a query based interface for searching and playback of relevant video segments (see Figure 2). Our implementation is based on two subsystems: VuSystem [10] - a toolkit for recording, manipulating and playing video, and the Semantic File System [1] - a storage subsystem with content-based access to data.

In the remainder of this paper we discuss the data model and its semantics (Section 2), the motivating design (Section 3), our prototype implementation (section 4), and conclusions based on our experience (Section 5).

2 Structured Video Model

The Structured Video System (SVS) is based on the *structured video* data model. This model can be conceived as a directed graph of structured video nodes and raw video segment pointers (see Figure 1). The structured video nodes contain pointers to other structured video nodes or to raw video segments (which are always the leaf nodes). The nodes contain descriptive information about themselves, information which is later used in the indexing process. These nodes also contain presentation and timing information that describes the playback characteristics of their children nodes. The user can playback or edit any structured video node independently of its parent nodes. SVS allows arbitrary nesting of the structured video nodes, as well as multiple views of the same data.

2.1 Descriptions

The model permits the association of arbitrary attributes and values with a given structured video node. For example, the nodes pointing to an unstructured video segment may contain close-captioned text recorded by the system. The user may add other attributes, such as title, author, characters, and the scene summary. Descriptions can also be non-textual such as key frames, icons, and salient stills [11]. The scope of a given structured video node description is the subgraph that originates from the node.

2.2 Video Algebra

A structured video node controls the temporal playback characteristics of its children. There are two methods of expressing these characteristics: relative time coordinates and video algebra. Relative time coordinates specify the starting point and the duration of the segments (or alternatively, end point marker) with respect to the parent node. This allows the expression of any temporal combination of children segments.

Our video algebra defines rules for combining one or more structured video nodes to form new video streams. A node can express the relative ordering and playback schedule of its children using the following operations:

- *concatenation*: for any child segments S_1 and S_2, $S_1 \circ S_2$ defines a schedule where S_2 follows S_1.
- *parallel*: for any child segments S_1 and S_2, $S_1 \parallel S_2$ defines a schedule where S_1 and S_2 are played concurrently.
- *union*: for any child segments S_1 and S_2, $S_1 \cup S_2$ defines a schedule where video footage of S_1 and S_2 is played (common footage is not repeated).
- *intersection*: for any child segments S_1 and S_2, $S_1 \cap S_2$ defines a schedule where only common video footage of S_1 and S_2 is played.
- *difference*: for any child segments S_1 and S_2, $S_1 - S_2$ defines a schedule where only video footage of S_1 that is not in S_2 is played.
- *conditional*: for any child segments $S_1, S_2, \ldots, S_k$, `(expression)` ? S_1 : S_2 : ... : S_k, defines a schedule where S_i is played if `expression` evaluates to i.
- *loop*: for child segment S_i, `loop` S_i *time* defines a repetition of the segment for a duration of *time* (which can be *forever*).

The above operations can be combined to produce complex scheduling definitions and constraints. The *union* operation allows the user to easily construct a non-repetitive video stream from overlapping segments. The *intersection* operation enables the user to construct video that incorporates only the overlap of multiple segments. Figure 3 illustrates the use of the above operations to form a playback schedule.

The *conditional* operation is used for interactive viewing, personalized viewing, and other viewing that can be affected by external sources. The expression in the conditional operation must evaluate to an integer. However, it is easy to map non-integer expressions, such as a user's environment variable, time-of-day, weather patterns, and user interaction, to valid integers.

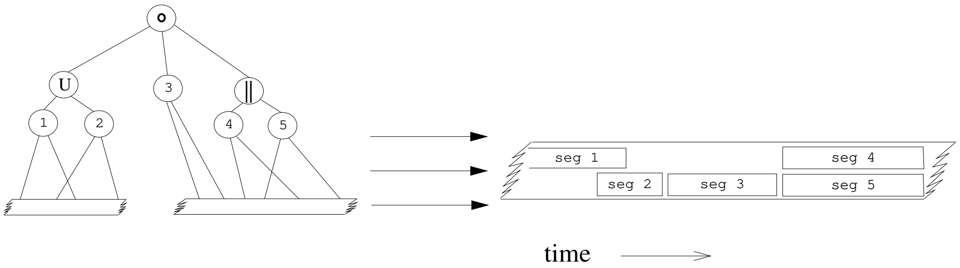

Figure 3: Playback Schedule

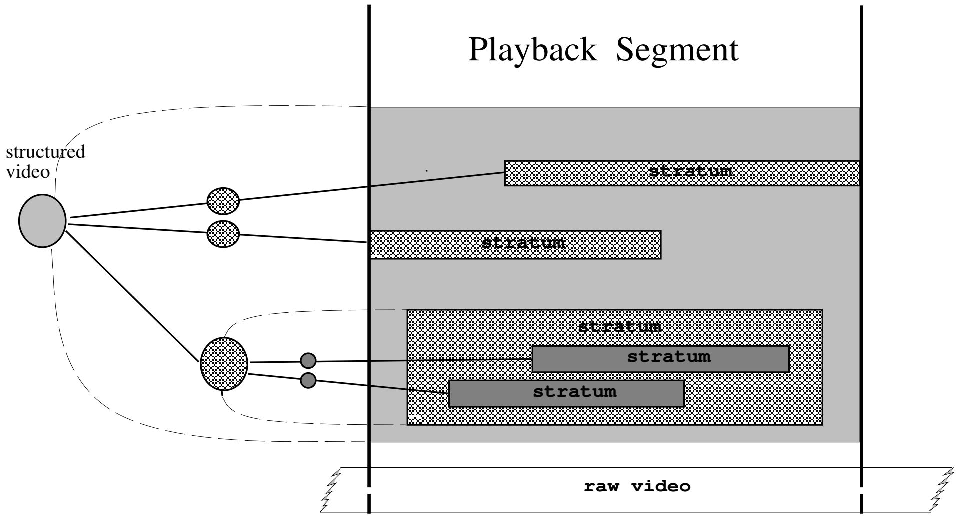

Figure 4: Nested stratification with structured video

Nested Stratification Figure 1 illustrates segments of a structured video node that point to overlapping portions of video data. This overlap enables the user to assign multiple meanings to the same footage, and serves the same purpose as the *stratification* mechanism described in [6]. Individual video segments that overlap are indexed by the system separately. A query may yield a structured video result that incorporates multiple segments when the query matches more than one segment. Other queries can result in structured video that corresponds to only one of these segments. Figure 4 gives an example of structured video that provides nested stratification. With this hierarchical organization, strata (which are just video segments) may be descendants of larger, encompassing strata. This stratification is used primarily for annotation and editing purposes (but can also be used when browsing or playing back video). When *union* is the timing operator for combining these overlapping segments, there will be no repetition of video footage during playback.

2.3 Presentation

A structured video node defines one or several continuous video streams to display in a window screen (the number of streams depends on the temporal information described above). The presentation information specifies the screen layout for displaying the streams and the overlap priorities between streams. Children nodes are either displayed relative to the region defined by the parent node window, or otherwise mapped independently.

3 Design

The Structured Video System is designed to manage the following processes: acquisition of video data from external sources (such as TV broadcasts, or other video collections), parsing the raw (i.e. unstructured) video to structured segments, user composition and editing of more complex video nodes, indexing of the structured video, content-based access to the data, and a platform for the playback and browsing of the video. At the core of the system is the *structured video* data model. To manage the data structures associated with the abstraction, the system stores the internal representation of structured video in UNIX files. The representation is decoupled from any particular portion of the system because any module may individually manipulate this data structure.

3.1 Structured Video Files

Structured video files contain semi-structured, textual specifications of the video nodes. There is one such file per node. The structured video file has

description	type	news
	title	CNN Headline News
	text	Today in Bosnia, ...
timing	$(1 \parallel 3) \circ 2$	
segment 1	raw video	Cnn.HN.127.Intro.rv
	presentation	(0, 0) - (.7, 1)
segment 2	structured video	Cnn.HN.314.Anchor.sv
segment 3	structured video	Bosnia-7-14-93.sv
	presentation	(.7, 0) - (1, 1)

Figure 5: A Structured Video File

the following parts:

- **Description**: associates descriptive attributes with the structured video node.
- **Timing**: specifies temporal relationships between children nodes.
- **Presentation** (optional): defines presentation characteristics of the screen layout for children nodes.
- **Segment descriptors**: identify the children structured video nodes.

Figure 5 and 6 give an example of a structured video file and illustrate the playback of this file using spatial and temporal coordinates.

Descriptions A description is composed of a set of attributes, where each attribute has a name and a value. An example of an attribute is `title = "CNN Headline News"`. Attribute names or values do not have to be unique within the file. Therefore, a description can have multiple titles, text summaries, and actor names that are associated with a video node. Attribute names are not fixed and can be defined by the user.

Timing Timing specification is either relative or using video algebra as described in 2.2. The relative specifications use time coordinates with respect to the parent node to determine the start point and duration of a child node. The video algebra specifications make use of the composition operators on

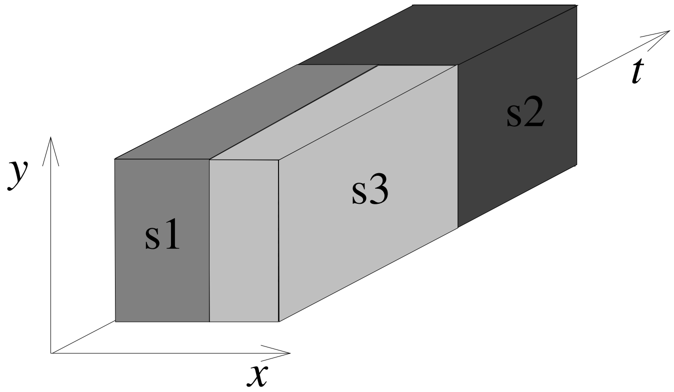

Figure 6: Playback of the Structured Video File

children nodes. Figures 7 and 8 illustrate an example of using the *union* operation in the composition of a structured video node. In this example, one raw video file is annotated by three overlapping segments. The *union* of the three overlapping segments yields one video stream with no redundancy in the playback.

Presentation As multiple structured video nodes can be scheduled to play at any specific time, their playback may result in concurrent video streams that need simultaneous screen display. To resolve conflicts and allow greater flexibility in video stream positioning, the presentation information of a node defines its children's screen layout (see 2.3). When two or more segments are scheduled to play simultaneously and also intersect in some portion of the screen, the structured video file defines overlap precedence. Thus, a segment with the higher precedence overlaps segments with lower precedence. When the file supplies no overlap information, the system arbitrarily chooses the precedence.

Naming Structured video files and raw video files are named using content-based access. Thus, each system defines a name space, and each structured video node within that name space has a unique name. For example:

```
structured_video = /svs/name:/CnnHeadlineNews.11256.sv
```

and

```
raw_video = /svs/name:/CnnFootage.12253.rv
```

description	title	CNN Headline News
	title	Clinton, Health Care
	summary	First lady proposes ...
timing	$(1 \cup 2) \cup 3$	
segment 1	structured video	Cnn.HN.AnchorTalk.sv
segment 2	structured video	Cnn.HN.Hillary.12.sv
segment 3	structured video	Cnn.HN.Healthcare.sv

Figure 7: Union Operation in Structured Video Files

where **/svs** denotes the structured video system name space.

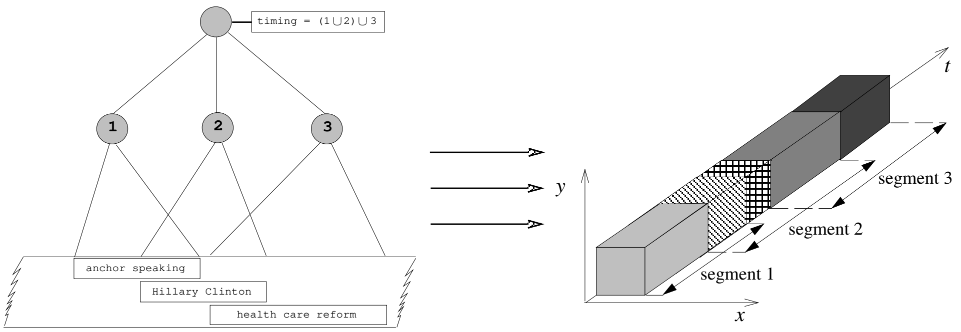

Figure 8: Graphical Representation of Union

3.2 Parsing

The system is designed to create structured video files from video footage using a VuSystem scene change detection module. The raw footage is parsed into scenes, which are then used as the basic building blocks of more complex structured video. Close-captioned text associated with the video stream is extracted and entered into descriptions as the `text` attribute. Then, the user can add more attributes such as `title`, `author`, and `actor`, and organize files into a desired hierarchy.

3.3 Structured Video Playback

Once the user has defined a nested hierarchy of structured video nodes, the system recursively parses the files and constructs a flattened 2-dimensional schedule for all the raw video segments (see Figure 3). One dimension is time and the other dimension relates which segments should be played at specific time intervals. A schedule file is associated with each structured video file and it defines the playback of raw video file segments. Temporal operations on children, such as *concatenation, parallel, conditional, difference* and *intersection* are straightforward to implement. The implementation of *union* is more difficult. It requires examining the video footage, scanning the schedule for overlapping video file segments, and determining how to join the overlapping segments. The playback module inspects the schedule file for a given structured video file and plays successive video frames from different raw video files as defined in the schedule.

3.4 Content-Based Access

To support content-based access, the system is designed to create structured video representations by parsing video footage and extracting temporal and content information. Indexing this information allows efficient querying and retrieval of relevant video segments.

3.4.1 Indexing

The user and the system associate descriptive, temporal and hierarchical information with structured video. Attributes such as scene title, actors or summary are added to nodes to further describe the contents of the video (also see discussion in 2.2). Also, timing relations between nodes are defined. The system then indexes the representation to create the correspondence between the attributes and the structured video nodes.

3.4.2 Querying

The query language allows the user to specify attributes describing desired properties of structured video nodes. Queries are boolean combinations of attributes, where each attribute is a *field-value* pair describing the desired value of a particular field. For example, the query $(person : clinton)\&(text : tax)$ will find all video segments that include Clinton and mention the word tax. An interesting feature that aids users in formulating queries is the ability to enumerate the admissible values for a given attribute.

A query returns a set of structured video nodes that can be played back or browsed to explore the video context and composition. For example, the user can follow a pointer to any parent node to inspect the encompassing video segment.

Temporal queries are handled using three predefined relations: *before, after*, and *overlap*:

- *before:* the relation holds for segments sharing the same parent immediately preceding the current segment.
- *after:* the relation holds for segments sharing the same parent immediately following the current segment.
- *overlap:* the relation holds for segments that have common video footage.

4 Implementation

We have built a prototype implementation of a Structured Video System that provides rapid content-based access to structured video. For our system, we use files containing textual specifications to represent structured video nodes. We implemented a special SV Transducer in the Semantic File System (SFS) to extract attributes from the semi-structured descriptions stored in these files. The transducer is used in the indexing process to associate attributes and values with the structured video files. The indices built during the indexing process allow rapid content-based access.

For finding structured video files that correspond to a particular query, the Structured Video System relies on the SFS pathname interface. SFS interprets a given UNIX pathname as an attribute query, and returns the result in a dynamically created *virtual directory*. For example, the query `ls /svs/type:/news/text:/clinton` will find all news footage that mentions Clinton. A virtual directory that contains the set of matching structured video will be created. Since the structured video files are stored in a human-readable, semi-structured file format, the user can edit and create structured video files using any available text editor.

The SV Player module consolidates the scheduling information from the structured video files with the raw video stored on the file system to produce a stream of digital video. This stream is transmitted to the VuSystem,

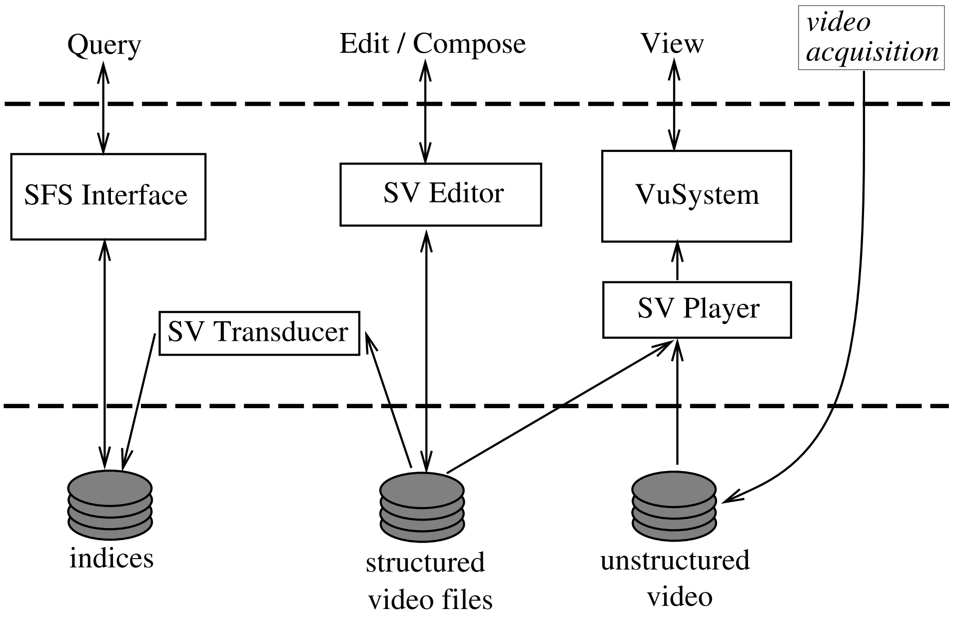

Figure 9: Structured Video System Architecture

which then displays the digital video on the client workstation. The VuSystem provides an environment for recording, processing and playing video. A set of C++ classes manage basic functions such as synchronizing video streams, displaying in a window, and processing video streams. TCL [4] scripts control C++ classes and offer a programmable user interface that can be customized. For example, our structured video browser is written as a TCL script. Figure 9 presents the architecture of the implementation.

4.1 Graphical User Interface

The Structured Video System user interface consists of a query interface and a video browser. A prototype query interface to the Structured Video System (see Figure 2) has been implemented. It provides a user friendly interface for searching within a structured video system and viewing the set of resulting segments. Currently, the browser can only play back nodes that use relative timing as the scheduling mechanism.

4.2 Experience

We are experimenting with our prototype system to gain more experience on the structured video data model and its support for content-based access. We acquired and indexed a collection of video segments: TV broadcast news, commercials, and movie trailers. When available, the indexing process also included the associated close-captioned text. Our Structured Video System provides rapid attribute-based access to the video collection, allows browsing a video result set, and delivers reasonable video playback performance once relevant video footage has been found. Typically, for a result set of twenty-five video segments, the elapsed time between query invocation by the user and system response is around five seconds. This experience is from running the client on a SparcStation 10 and the server on an SGI PowerSeries 4D/320S. The system response includes enumerating and displaying the first frame of the matching video segments. Once the user selects a video node to play, typically three seconds elapse until the browser begins to play the video stream.

We are working on extending the system to provide an Internet Video Server that can be mounted remotely as a NFS server. This requires incorporation of some video playback performance enhancement techniques, such as controlled quality degradation for networks with limited bandwidth, video compression, and data caching.

5 Conclusions

We have described how semantic information about video can be structured and used for content-based access. The semantically rich model of structured video provides an efficient means of organizing and manipulating video data by assigning logical representations to the underlying video streams and their contents. It supports nested stratification for powerful descriptions of video footage. Structured video also uses video algebra to express unique compositions and temporal relationships between nodes.

We have built a prototype based on the Semantic File System and the VuSystem. We have successfully used the prototype to retrieve interesting video segments and to browse our collection. The results of these experiments suggest that structured video enables efficient access and management of video collections in interesting and diverse ways. From our experience thus far, we believe that the structured video data model is the adequate abstraction for representing digital video and supporting content-based access.

We plan to enhance our prototype by adding a graphical editor for structured video and by investigating mechanisms to automate the parsing of video footage. Also, a richer query language is needed for expressing temporal relationships. We also plan to examine object-oriented databases support for structured video storage. Another area of future research is the exploration of interactive movies and home video editing.

Acknowledgments

We would like to acknowledge the support of the Defense Advanced Research Projects Agency and the Department of the Army under contract DABT63-92-C-0012. Special thanks go to Chris Lindblad and David Tennenhouse for providing and supporting the VuSystem. We would also like to thank James O'Toole for supporting our use of the Semantic File System. Thanks also to our readers: David Gifford, James O'Toole and Mark Sheldon.

References

[1] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, and James W. O'Toole. Semantic file systems. In *Thirteenth ACM Symposium on Operating Systems Principles*. ACM, October 1991. Available as *Operating Systems Review* Volume 25, Number 5.

[2] R. Hamakawa and J. Rekimoto. Object composition and playback models for handling multimedia data. In *Proc. First ACM Int'l Conference on Multimedia.*, pages 273–281, Anaheim, CA, August 1993.

[3] T.D.C Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh. A digital on-demand video service supporting content-based queries. In *ACM, First Multimedia Conference.*, Anaheim, California, August 1993. ACM.

[4] J.K. Ousterhout. An x11 toolkit based on the tcl language. In *USENIX Association 1991 Winter Conference Proceedings*, pages 105–115, Dallas, TX, January 1991.

[5] Roger Price. Mheg: An introduction to the future international standard for hypermedia object interchange. In *Proc. First ACM Int'l Conference on Multimedia.*, pages 121–128, Anaheim, CA, August 1993.

[6] T.G. Aguierre Smith and G. Davenport. The stratification system: A design environment for random access video. In *Proc. 3rd International*

Workshop on Network and Operating System Support for Digital Audio and Video., La Jolla, CA, November 1992.

[7] T.G. Aguierre Smith and N.C. Pincever. Parsing movies in context. In *Proc Summer 1991 Usenix Conference.*, pages 157–168, Nashville, Tennessee, June 1991.

[8] D. Swanberg, C.F. Chu, and R. Jain. Architecture of a multimedia information system for content-based retrieval. In *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video.*, La Jolla, CA, November 1992.

[9] D. Swanberg, C.F. Chu, and R. Jain. Knowledge guided parsing in video datbases. In *IS&/SPIE's Symposium on Electronic Imaging: Science & Technology*, San Jose, CA, January 1993.

[10] David K. Tennenhouse. Telemedia, networks and systems group annual report. Technical Report MIT/LCS/AR-001, MIT Laboratory for Computer Science, June 1992.

[11] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. In *Proc. First ACM Int'l Conference on Multimedia.*, pages 39–46, Anaheim, CA, August 1993.

[12] G. van Rossum and et al. Cmifed: A presentation environment for portable hypermedia documents. In *Proc. First ACM Int'l Conference on Multimedia.*, pages 183–188, Anaheim, CA, August 1993.