Algebraic Video for Composition and Content-Based Access

Ron Weiss
Andrzej Duda*
David K. Gifford

Programming Systems Research Group
MIT Laboratory for Computer Science

Abstract

We introduce a new data model called *algebraic video* that provides operations for the composition, search, navigation and playback of digital video presentations. Video presentations are composed using a *video algebra* that consists of a set of basic operations on video segments to produce a desired video stream. The video algebra contains operations for temporally and spatially combining video segments as well as for attaching attributes to these segments. Algebraic video access methods also query and navigation operations. Query and navigation allow users to discover video presentations of interest by describing desired attributes and exploring a presentation's context. Unlike previous approaches, algebraic video permits video expressions to be nested in arbitrarily deep hierarchies. It also permits video segments to inherit attributes by context. Experience with a prototype algebraic video system suggests that algebraic video is an effective way to access and manage video. The prototype system is used to discover video segments of interest from existing collections and create new video presentations with algebraic combinations of these segments.

1 Introduction

As digital video becomes ubiquitous and as more video sources become available, applications will need to deal with digital video as a new data model. However, since video has both temporal and spatial dimensions, it places different requirements on applications than existing data types such as text. Moreover, the volume and unstructured format of digital video data make it difficult to manage, access and compose video segments into video documents. When creating new video presentations, it is essential to reuse existing video segments and presentations, because the sheer volume of the data makes copying prohibitive. Providing a new digital video data model with content-based access will alleviate these problems and motivate broader use of video resources.

Many existing digital video abstractions rely on the traditional view of video as a linear temporal medium. They do not take full advantage of either the logical structure of the video or of hierarchical relationships between video segments. Moreover, flexible associative access based on the structure and the hierarchy is not supported. For these reasons, *algebraic video* allows the user to:

- create video presentations that
  - model nested video structures such as shot, scene and sequence,
  - express temporal compositions of video segments,
  - define output characteristics of video segments,
  - specify multi-stream viewing.
- integrate content-based access to video:
  - associate content information with logical video segments,
  - provide multiple coexisting views and annotations of the same data,
  - provide associative access based on the content, structure and temporal information.

This work was supported by the Defense Advanced Research Projects Agency and the Department of the Army under contract DABT63-92-C-0012.

* Also with Bull-IMAG Systèmes and INRIA. This work was done when the author was visiting MIT Laboratory for Computer Science.

- playback video presentations

The algebraic video data model can provide the foundation for a digital video system where users efficiently access and manage video information. For example, consider a user that wants to compose a digital video presentation about the latest developments with economic reforms. First, the user searches a large collection of TV broadcasts for all video segments reporting on economic reforms and Smith, who is a notable economist. Then, the user may want to examine the context in which the segments have appeared: in the headline news or in a talk show. Finally, the user chooses some segments and combines them such that they form a new video presentation that can be played out, stored, or exchanged.

A new video data model that supports our example's operations must integrate both content attributes of the video and its semantic structure. A video data model may need to describe the people in a scene, the associated verbal communication for each video segment, and the relationships between segments. Automatic content extraction, such as image and speech recognition should be used when possible. Because these methods are not yet generally feasible, other forms of information extraction can be employed. Text captions or image features, such as color, texture, and shape, may be associated with the video footage. The user can also associate personalized descriptions with any component of the video presentation. Finally, the data model allows users to express temporal and structural relationships between video segments. Along with content information, these semantic structure attributes are also indexed for content-based access.

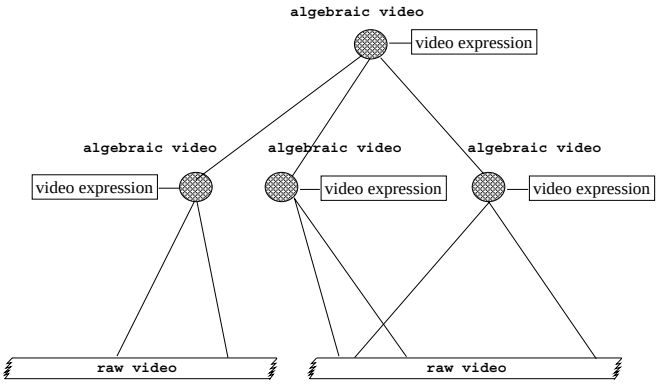

Figure 1: Algebraic Video

The *algebraic video* data model consists of hierarchical compositions of *video expressions* with high-level semantic descriptions. A video expression denotes a *video presentation*, which is multi-window, concurrent spatial and temporal combination of video segments (see Figure 1). The video expressions are constructed using *video algebra* operations. We introduce video algebra as a means for combining and expressing temporal relations, for defining the output characteristics of video expressions, and for associating descriptive information with these expressions. The algebraic video abstraction provides an efficient means of organizing and manipulating video data by assigning logical representations to the underlying video streams and their contents. The model also defines operations for flexible associative access to the video information. The algebraic video preserves the correspondence between video segments so that all relevant segments and their neighbors can be efficiently found. The output characteristics of video expressions are media-independent, and thus the rendering can adjust to the available resources.

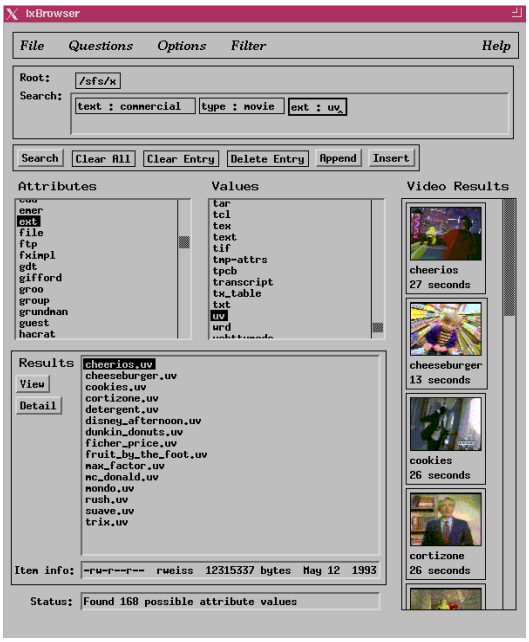

Figure 2: Algebraic Video Query Interface

Users access algebraic video via query, navigation, and playback of video presentations. These access methods to an algebraic video collection are grouped together as the *interface operations*. Users can search for relevant presentations with queries that describe desired attributes of video expressions. The invocation of a query results in a set of video expressions that can be played back, reused or manipulated by a user. In addition to content-based access, the interface operations allow users to browse and explore

the structure of the video expressions. These activities help users understand the surrounding organization and context. For example, the user can find an interesting expression and then examine the encompassing video segments. Furthermore, users can create their individual interpretations of existing video footage by composing new video expressions from the existing video components.

The algebraic video data model allows users to compose concurrent video presentations by structuring raw data into logical video segments and then describing the temporal relations between these segments. Hierarchical relations between the video expressions allow *nested stratification* — overlapping segments are used to provide multiple coexisting views and annotations of the same data and enable the user to assign multiple meanings to the same footage. Segments can be organized hierarchically so that their relationships are preserved and can be exploited by the user. In addition to simple stratification, the algebraic video model preserves nested relationships between strata and allows the user to explore the context in which a stratum appears (see discussion in Section 2).

The algebraic video data model offers the following important advances over previous digital video representations:

- It provides the fundamental functions required to deal with digital video: composition, reuse, organization, searching, and browsing.
- It models complex, nested logical structure of video using video algebra. The video algebra is a useful metaphor for expressing temporal interdependencies between video segments, as well as associating descriptions and output characteristics with video segments.
- The model allows associative access based on the content of the video, its logical structure and temporal composition.

The model has been implemented as part of the *Algebraic Video System* project. The system allows users to compose algebraic video presentations. It extracts video attribute information and supports content-based access, as well as video playback. It offers a query based interface for searching, browsing and playback of relevant video segments (see Figure 2). The algebraic video browser uses the logical representation of the video data to provide viewing methods based on the ascribed temporal characteristics of the video.

In the remainder of this paper we discuss related work (Section 2), the design of the algebraic video data model (Section 3), our prototype implementation (Section 4), and conclusions based on our experience (Section 5).

2 Related Work

Work related to ours includes video authoring and annotation tools (Section 2.1), systems that provide content-based access to video (Section 2.2) and modeling of unstructured video for content-based retrieval (Section 2.3).

2.1 Video Authoring and Annotation

Video authoring and annotation tools provide facilities for composing and annotating complex video presentations. Davenport *et al.* [3, 2] 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. Figure 3 shows an example of video footage annotated by strata. 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. This results from the absence of relationships between the strata. Our data model provides a full hierarchical organization of video footage that permits flexible browsing. Unlike simple stratification, the algebraic video model preserves the nested relationships between strata and allows to explore the context in which a stratum appears.

Commercially available tools such as Adobe Premiere [1], DiVA VideoShop [7] and MacroMind Director [14] allow the user to create movies using audio and video tracks, and also enables the user to specify special effects during video segment transitions. These commercial systems are based on two distinct paradigms: *timelines* and *scripts*. In the timeline approach, video and audio objects are placed on a line representing time flow. Video objects are normally a sequence of frames that may have an associated audio stream. Prerecorded audio streams can also be independently placed in the timeline. Normally, a direct manipulation graphical editor similar to the one illustrated in Figure 4 (artificially created for the purposes of this discussion), presents the video author with video and audio tracks, and a special effects track for combining the two video tracks. Synchronization between any two objects is achieved by carefully placing the video and audio objects on tracks that are marked

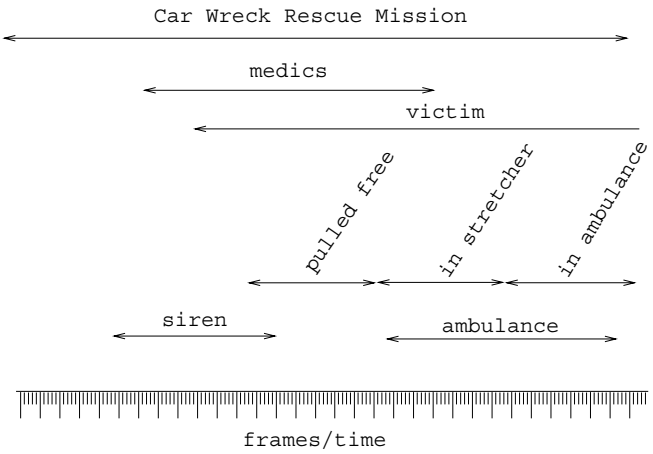

Figure 3: Annotation of Video Footage using Stratification

by time indices. The indices reflect the elapsed time since the beginning of the video presentation. The toolkits allow the user to edit video data in essentially the same manner as film makers edit movies. They arrange shots on a temporal linear axis by cutting, pasting and making transitions. The computer merely simplifies the previously mundane task of searching for a sequence of frames from a video source such as a video tape, and then copying the frames onto the video target. Any presentation created on a timeline can be easily mapped into an algebraic video presentation. However, some algebraic video presentations, such as ones that include choices, cannot be modeled using a simple timeline metaphor.

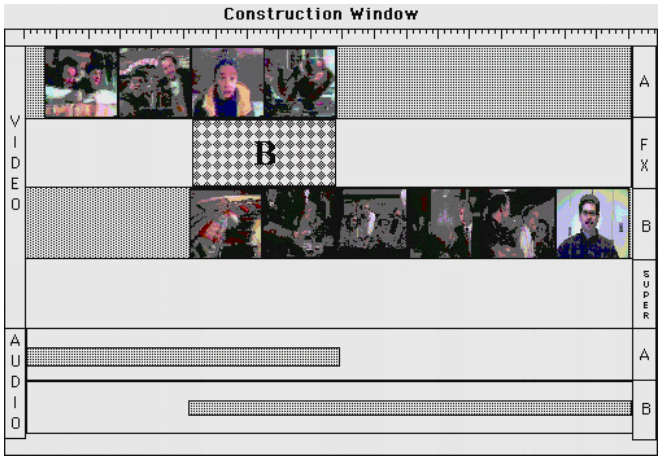

Figure 4: Timeline Editor

The script (or *flowchart*) approach requires the author to explicitly program timing and placement information.

Digital video is unique because it is not restricted by the linearity of traditional media. It possesses a dynamic element, where the video display may be determined during runtime and may not follow a strictly linear progression determined *a-priori*. The toolkits do not take advantage of this distinctive feature. Moreover, the toolkits lack methods for specifying the elaborate logical structure of video data and do not address content-based access. Our approach allows structured, multi-stream composition using video algebra operations and content-based access.

Multimedia authoring systems such as CMIFed [24] have rich structuring primitives for multimedia documents, but fail to address the structure of the video data itself. The video is still treated as unstructured linear stream. Hamakawa and Rekimoto [11] 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.

Media Streams is an iconic visual language that enables users to create multi-layered, iconic annotations of video content [6]. Icons denoting objects and actions are organized into cascading hierarchies from levels of generality to increasing levels of specificity. Additionally, icons are organized across multiple axes of descriptions such as objects, characters, relative positions, time or transitions. The icons are used to annotate video streams represented in a Media Time Line. Currently, around 2200 iconic primitives can be browsed. However, this user-friendly visual approach to annotation is limited by a fixed vocabulary. Also, it does not exploit textual data such as close-captioned text.

The MHEG [17] standard is intended for "*coded representation of final form multimedia and hypermedia objects that will be interchanged across service and applications*". At the core of the standard are the MHEG objects (represented in Figure 5) that play a federated role between interacting applications. MHEG defines the formats used at the interchange point between applications that want to exchange multimedia data. The objects are synchronized and composed to form complex presentations using four mechanisms: script, conditional activation, spatio-temporal, and close system synchronization.

The HyTime [19] hypermedia standard provides a mechanism to specify hyperlinks and schedule multi-

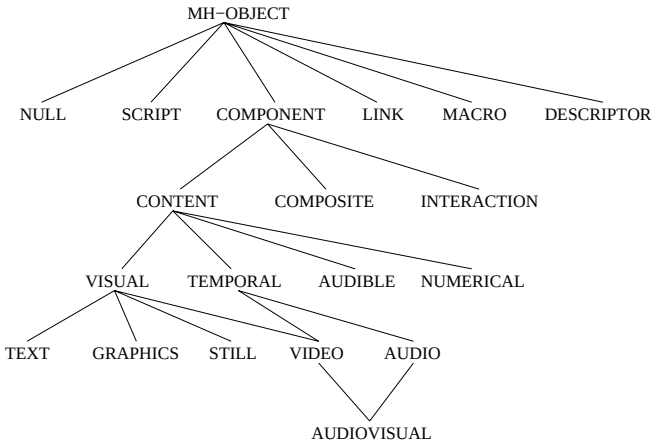

Figure 5: The MHEG Object Inheritance Tree

media information in time and space. It is based on the Standard Generalized Markup Language (SGML), using "architectural forms" to express rules for hypermedia structuring information. These architectural forms and attributes of information objects are grouped into six modules: Base module, Measurement module, Location address module, Hyperlinks module, Scheduling module and a Rendition module. The scheduling module allows events, which are occurrences of information objects, to be scheduled in "finite coordinate spaces" (fcs). The user expresses spatial and temporal positions of objects in fcs's using coordinate axes or relationships. The rendition module maps the fcs representation to its "real-world" counterpart to achieve playback of the multimedia information. As stated in their description, both MHEG and HyTime are intended for final formatted documents, and lack mechanisms for content-based access, editing and annotation of the multimedia data.

2.2 Systems with Content-Based Access to Video

Content-based access systems provide facilities to discover video segments of interest. Little *et al.* [12] 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 because descriptions cannot be assigned to overlapping or nested video sequences as is accomplished in the stratification model. Moreover, 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.

Electronic Scrapbook is a system for home-video video annotation and editing [5], where the annotations can later be used for content-based access. The user can attach descriptions to video clips and use a modified form of case-based reasoning to edit and create personalized video stories. The user can query a database of video clips and also filter, sort, or remove overlapping segments from the results. The system uses a small, special-purpose taxonomy that can be used in descriptions, but does not exploit the logical structure of video. For example, the user cannot describe hierarchical relationships where video segments are nested.

Gibbs *et al.* [9] proposes an object-oriented approach to video databases. An *audio/video database* can be viewed as a collection of *values* (audio and video data) and *activities* (interconnectable components used to process values). Two abstraction mechanisms, temporal composition, and flow composition allow aggregation of values and activities. The database values (audio or video) are linear sequences of data elements so that the logical structure is not represented. Also, the temporal composition mechanism is essentially equivalent to the timeline paradigm.

2.3 Modeling of unstructured video for content-based retrieval

When video is captured into digital format from an analog source, it initially exists as an unstructured sequence of video frames and audio segments. Several proposed systems extract information from these unstructured streams and then provide a data model that is used for content-based access. Swanberg *et al.* [20, 21] defines such an architecture for parsing data semantics from the video stream. The system manages a fixed data schema for representing information about the video stream, where a shot is defined as a sequence of frames without a scene change, and an *episode* is a sequence of shots that are somehow related. The system provides tools and models to aid in the analysis of a video stream, including support for identification of *shots* and *episodes*. A knowledge module maintains the information about the segmentation of the video footage, information about the objects and features in the video, and information to facilitate query optimization. The data schema used for this system is not

sufficiently flexible and therefore not suitable for free form modeling of the complex relations between video segments.

Nagasaka [15] implemented a system that automatically indexes video by detecting cuts and associating a small icon of a representative frame with each subpart. The list of icons is used as an index of the video. Additionally, the system supports full-video searches for frames in which a specified object appears. Queries are accomplished using an image of the reference objects.

3 Algebraic Video Model

In general, video is composed of different story units such as *shots, scenes and sequences* arranged according to some logical structure [13]. Frames recorded sequentially form a *shot*. One or several related shots are combined in a *scene* and a series of related scenes forms a *sequence*. The logical structure is defined by a screenplay that organizes story units and provides detailed descriptions of scenes and sequences. Video also contains complex content information that can be extracted and associated with the video story units. For example, attributes such as close-captioned text and key frames that characterize a shot can be associated with video. Also, descriptive information from screenplays can be added to video descriptions. The design goal of algebraic video is to provide a high-level abstraction that models complex information associated with digital video data and supports content-based access.

Interaction with algebraic video is accomplished via four activities: *edit* and *compose* video presentations, *play* and *browse* existing algebraic video presentations, *navigate* the video hierarchy and *query* to find relevant video. The operations that support playback, navigation and content based access are grouped together as the *interface* operations.

In the algebraic video data model, the fundamental entity is a *presentation*. A presentation is a multi-window spatial, temporal, and content combination of video segments. Presentations are described by *video expressions*. The most primitive video expression involves the creation of a single-window presentation from a raw video segment. These segments are specified using the name of the raw video and a range within the raw video. Compound video expressions are constructed from simpler ones using *video algebra* operations. Video expressions can be named by variables, composed to reflect the complex logical structure of the presentations, and share the same video data. A video expression may contain composition information, descriptive information about the contents, and output characteristics that describe the playback behavior of the presentation. Video expressions can be played back, searched and browsed.

An *algebraic video node* provides a means of abstraction by which video expressions can be named, stored and manipulated as units. An algebraic video node contains a single video expression that may refer to children nodes or raw video segments (see Figure 1).

The remainder of this section introduces the algebraic video operations for editing and composition (Section 3.1), presents the interface for managing and accessing video presentations (Section 3.2), and describes nested stratification as an effective means for content-based access (Section 3.3).

3.1 Video Algebra

The video algebra operations are classified into the following categories:

- **Creation**: defines the construction of video expressions from raw video.
- **Composition**: defines temporal relationships between component video expressions.
- **Output**: defines spatial layout and audio output for component video expressions.
- **Description**: associates content attributes with a video expression.

Table 1 presents the video algebra operations. The arguments denoted by E_1, $E_2, \ldots, E_k$ are video expressions. The result of a video expression is a presentation. A video expression defines the temporal and spatial composition of its *presentation* arguments using the operators defined in the table. For the examples given in this chapter, expressions of the form: $(E_1 \odot E_2 \odot E_3 \odot \ldots E_n)$, where $\odot$ is any specific binary operation, denote an expression of the form: $(\ldots((E_1 \odot E_2) \odot E_3) \odot \ldots E_n)$. Also note that the binary video algebra operations are inherently not associative because they include a temporal component.

3.1.1 Composition

Complex scheduling definitions and constraints can be expressed using the composition operators. The composition operations can be combined to produce complex scheduling definitions and constraints. The *union*

Creation	
create	`create` *name begin end* creates a presentation from the range within the identified raw video segment
delay	`delay` *time* creates a presentation with empty footage for duration *time*
Composition	
concatenation	$E_1 \circ E_2$ defines the presentation where E_2 follows E_1
union	$E_1 \cup E_2$ defines the presentation where E_2 follows E_1 and common footage is not repeated
intersection	$E_1 \cap E_2$ defines the presentation where only common footage of E_1 and E_2 is played
difference	$E_1 - E_2$ defines the presentation where only footage of E_1 that is not in E_2 is played
parallel	$E_1 \parallel E_2$ defines the presentation where E_1 and E_2 are played concurrently and start simultaneously
parallel-end	$E_1 \wr\wr E_2$ defines the presentation where E_1 and E_2 are played concurrently and terminate simultaneously
conditional	(`test`) ? $E_1 : E_2 : \ldots : E_k$ defines the presentation where E_i is played if `test` evaluates to i
loop	`loop` E_1 *time* defines a repetition of video expression E_1 for a duration of *time* (can be *forever*)
stretch	`stretch` E_1 *factor* the duration of the presentation is equal to *factor* times duration of E_1. This is achieved by changing the playback speed of the video expression.
limit	`limit` E_1 *time* the duration of the presentation is equal to the minimum of *time* and the duration of E_1, but the playback speed is not changed.
transition	`transition` E_1 E_2 *type time* defines *type* transition effect between expressions E_1 and E_2; *time* defines the duration of the transition effect
contains	`contains` E_1 *query* defines the presentation that contains component expressions of E_1 that match *query*
Output	
window	`window` E_1 $(x_1, y_1) - (x_2, y_2)$ *priority* specifies that E_1 will be displayed with *priority* in the window defined by (x_1, y_1) as the bottom-left corner, and (x_2, y_2) as the right-top corner such that $x_i \in [0, 1]$ and $y_i \in [0, 1]$
audio	`audio` E_1 *channel force priority* specifies that the audio of E_1 will be output to *channel* with *priority*. If *force* is true, override `audio` specifications of the component expressions.
Description	
description	`description` E_1 *content* specifies that E_1 is described by *content*
hide-content	`hide-content` E_1 defines a presentation that hides the content of E_1

Table 1: Video Algebra Operations

```
C1 = create Cnn.HeadlineNews.rv 10 30
C2 = create Cnn.HeadlineNews.rv 20 40
C3 = create Cnn.HeadlineNews.rv 32 65

(description
  (C1 ∪ C2 ∪ C3)
  (title = "CNN Headline News"
   text = "Smith proposes economic reform ..."))
```

Figure 6: Union Operation in an Algebraic Video Node

operation allows the user to easily construct a non-repetitive video stream from overlapping segments. It preserves the temporal ordering of the component expressions. If these expressions do not contain overlapping segments, then *union* is equivalent to *concatenation*. Figures 6 and 7 present an example of an algebraic video node that uses the *union* operation in the composition of a video expression. In this example, one raw video file is annotated by three overlapping nodes. The *union* of the three overlapping nodes yields one video stream with no redundancy in the playback.

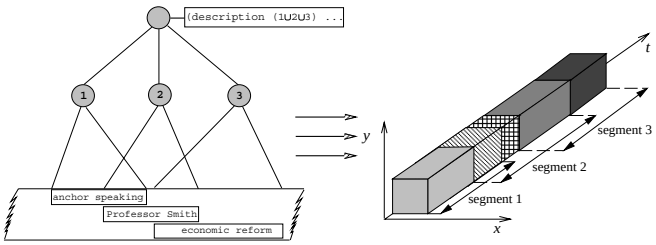

Figure 7: Graphical Representation of Union

The *intersection* operation enables the user to construct a new video presentation that includes only footage that is contained in both of the arguments. The *parallel* operation allows the user to compose multi-window, concurrent video presentations. The *conditional* operation can be used for personalized viewing or other viewing that can be affected by external sources. The test expression in the conditional operation must evaluate to an integer. However, it is easy to map non-integer test expressions, such as a user's environment variable, time-of-day, weather patterns, and user interaction, to valid integers. The *conditional* operation can be used in the domain of interactive movies where a user creates 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.

The *stretch* operation changes the playback speed of the video presentation, but does not alter the playback speed of other presentations. The *transition* operation combines two video expressions using a transition effect of duration *time*. The transition *type* is one of a set of transition effects, such as dissolve, fade, and wipe. Note that *concatenation* is a simple *transition* with $time = 0$. The *contains* operation permits the user to include the results of a query on a video expression argument. The operation combines the subexpressions that match the query into one video expression, while preserving the hierarchical relations of the video expression argument. The syntax and semantics of the *query* argument in a *contains* operation is explained in section 3.2.1.

We are investigating other algebraic video composition operations. These include operations that will achieve overlay of video streams, synchronization on events, a general synchronization operator (similar to operators defined by Fiume *et al.* [8]), and non-determinism.

3.1.2 Output Characteristics

Because multiple video streams can be scheduled to play at any specific time within one video expression, the playback may require multiple screen displays and audio outputs. Therefore, video expressions include output characteristics that specify the screen layout and audio output for playing back children streams.

All video expressions are associated with some rectangular screen region in which they are displayed. A video expression constrains the spatial layout of its components. As expressions can be nested, the spatial layout of any particular video expression is defined relative to the parent rectangle. The parent rectangle is the screen region associated with the encompassing expression. The *window* operator defines a rectangular region within the parent rectangle where the given video expression is displayed. The rectangular region is specified by two points in a relative coordinate system, the top-left (x_1, y_1) and bottom-right (x_2, y_2) corners, such that $x_i \in [0, 1]$ and $y_i \in [0, 1]$. By default, a video expression is associated with a square that fits in the parent rectangle. Figures 8 and 9 give a simple example of an algebraic video node and illustrate the playback characteristics of this node using spatial and temporal coordinates. Figures 10 and 11 give a more elaborate example of an algebraic video node with nested window specifications and a snapshot captured during playback.

```
C1 = create Cnn.HN.127.Intro.rv 35 70
C2 = Cnn.HN.314.Anchor.av
C3 = Bosnia-7-14-93.av

P1 = window C1 (0,0) - (.7,1) 10
P2 = window C2 (0,0) - (1,1) 20
P3 = window C3 (.7,0) - (1,1) 30

(P1 || P3) ∘ P2
```

Figure 8: An Algebraic Video Node with Parallel and Concatenation Operations

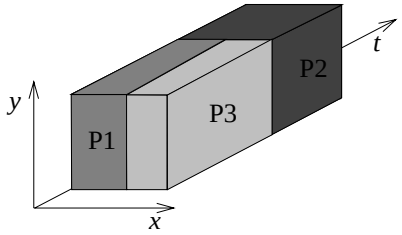

Figure 9: Playback of the Algebraic Video Node

Window priorities are used to resolve overlap conflicts of screen display. The *window* operation establishes the video priority of the associated window region with the *priority* parameter. The window with the higher priority overlaps the window with the lower priority. For example, assume that the two windows W_{c1} and W_{c2} are children of the same parent window region. If the priority of W_{c1} is greater than the priority of W_{c2}, then W_{c1} and all its video subexpressions will overlap W_{c2} and all its video subexpressions. 192z The *audio* operation directs the audio output of the video expression to *channel*, which can be any logical audio device. If the *force* argument is true, then the *audio* operation overrides any *channel* specifications of the component video expressions. The *priority* parameter is defined in a manner analogous to the *priority* parameter of the *window* operation.

3.1.3 Descriptions

The model permits the association of arbitrary *descriptions* with a given video algebra expression. It allows textual, as well as non-textual descriptions such as key frames, icons, salient stills [23], and image features like color, texture, and shape. The *description* operation associates *content* information with a video expression.

```
C1 = create hoffa.rv 30:0 50:0

P1 = window C1 (0,0) - (0.5,0.5) 10
P2 = window C1 (0,0.5) - (0.5,1) 20
P3 = window C1 (0.5,0.5) - (1,1) 30
P4 = window C1 (0.5,0) - (1,0.5) 40
P5 = (P1 || P2 || P4)
P6 = (P1 || P2 || P3 || P4)

(P5 ||
  (window
    (P5 || (window P6 (0.5,0.5) - (1,1) 60))
    (0.5,0.5) - (1,1) 50))
```

Figure 10: An Example of Nested Windows

Figure 11: A Playback Snapshot of the Node with Nested Windows

The *content* description of an expression is not fixed by our model. However, for the purposes of this paper and our prototype implementation, a *content* is a boolean combination of *attributes*. Each attribute consists of a *field* name and a *value*. An example of an attribute is `title = "CNN Headline News"`. Some field names have predefined semantics, e.g., `title`, and other fields can be defined by the user. Values can assume a variety of types, including strings and video node names. Field names or values do not have to be unique within a description. Therefore, a description can have multiple titles, text summaries, and actor names that are associated with a video expression. For example, a description may contain recorded closed-captioned text. The user may add other attributes,

Content-Based Access	
search	`search` *query* searches a collection of nodes for video expressions that match *query*
Browsing	
playback	`playback` *video-expression* playback the video expression
display	`display` *video-expression* display the video expression
Navigation	
get-parents	`get-parents` *video-expression* returns the set of nodes that directly point to *video-expression*
get-children	`get-children` *video-expression* returns the set of nodes that *video-expression* directly points to

Table 2: Interface Operations

such as actor, characters, and the scene summary. The components of a video expression inherit descriptions by context. This implies that all the content attributes associated with some parent video node are also associated with all its descendant nodes.

The *hide-content* operation defines a video expression E that does not contain any descriptions. The *contains* and *search* operations (described in section 3.2.1) on E does not recursively examine the components of E. The *hide-content* operation provides a method for creating abstraction barriers for content-based access.

3.2 Interface Operations

The interface operations on video expressions fall into three main categories: content-based access, browsing and navigation. Table 2 defines these operations. They are also discussed in the following subsections.

3.2.1 Content-Based Access

Associative access to video expressions is accomplished with the *search* operation, in which the user specifies desired properties of the expressions. A *search* is performed within the context of a collection of persistent algebraic video nodes. For querying within the collection, we have chosen a simple predicate query language. A *query* is a boolean combination of attributes. When a query is applied to the collection in the *search* operation, the hierarchy of every node in the collection is searched recursively and the operation returns the *query result set* of nodes that satisfy the query. Note that the recursive search does not examine sub-hierarchies of the components of expressions constructed by the *hide-content* operation. Also, a node that can be revealed in more than one way is not searched more than once.

A description of a video expression is implicitly inherited by its subexpressions (which can be descendant nodes). The scope of a given algebraic video node description is the subgraph that originates from the node. Matching a query to the attributes of an expression must take into account all of the attributes of that expression, including the attributes of its encompassing expressions. In the case where the expression is an algebraic video node, this also includes the attributes of the ancestors. However, if a node's ancestor is in the result set, the descendant node is removed from the result set. This is done to ensure that complete sub-hierarchies of algebraic nodes are not returned in the result set of a query that matches some ancestor node. For example, consider the query `text:smith and text:question` applied to a collection that contains the node described in Figure 12. The result of the query is the node with the description `text:"question from audience"`, because this node implicitly contains the description `text:smith`. The node with the description `text:question` is not returned because it is a descendant of a node already in the result set.

Once a query result set is generated, the user can then playback any of the expressions in the set or browse and explore the video context and composition using the operations described in the previous section. For example, the user can inspect the encompassing video segment by examining the parent nodes.

3.2.2 Browsing and Navigation

The browsing operations enable the user to inspect the video expression and to view the presentation as defined by the expression. The user can *playback* any expression. For any given expression, the user can browse and traverse the organizational hierarchy with the *get-parents* and *get-children* operations. Notice that *get-parents* of a video expression that is not a node will yield an empty result set. Finally, the user can *display* the expression associated with a video expression. As discussed above, the expression includes description, composition, and output characteristics. If the argument is not a node, the operation is simply the identity function.

3.3 Nested Stratification

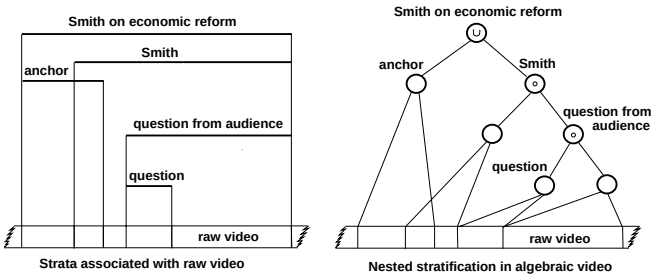

Figure 12: Nested stratification with algebraic video

Hierarchical relations between the algebraic video nodes allow *nested stratification*. Davenport *et al.* [3, 2] defines a *stratification* mechanism, where textual descriptions called *strata* are associated with possibly overlapping portions of a linear video stream. In the algebraic video data model, linear strata are just algebraic video nodes. To create a simple strata as in the Davenport model, a user specifies the raw video file and the sequence of relevant frames with the *create* operation.

Nodes that refer to the same video data are used to provide multiple coexisting views and annotations, and enable the user to assign multiple meanings to the same footage. Moreover, algebraic video nodes can be organized hierarchically so that their relationships are preserved and can be exploited by the user. In addition to simple stratification, the algebraic video model preserves nested relationships between strata and allows the user to explore the context in which a stratum appears. Figure 12 presents an example of algebraic video that utilizes *nested stratification*. The nested algebraic video nodes preserve the structural composition of the presentation, and also allow coexisting content and structural interpretations for overlapping footage.

The nested stratification is used primarily for annotation and editing purposes, however it can also be used when browsing, searching or playing back video. The *union* operator used for combining overlapping nodes guarantees that there will be no repetition of video footage during playback.

4 Implementation

The *Algebraic Video System* is a prototype implementation of the algebraic video data model and its associated operations. The system provides support for composition and content-based access to algebraic video. The creation of video expressions involves the specification and combination of raw video segments. Video expressions also serve as repositories for attribute information extracted from the video segments. In the prototype, the units of storage and indexing are the algebraic video nodes. These nodes are textually represented by human-readable, semi-structured, algebraic video files. The system has a graphical user interface for managing a collection of raw video segments and algebraic video nodes, and includes query and browsing tools. The storage subsystem includes raw, unstructured video, a representation of algebraic video, and indexes to support content based access. Figure 13 presents the architecture of the implementation.

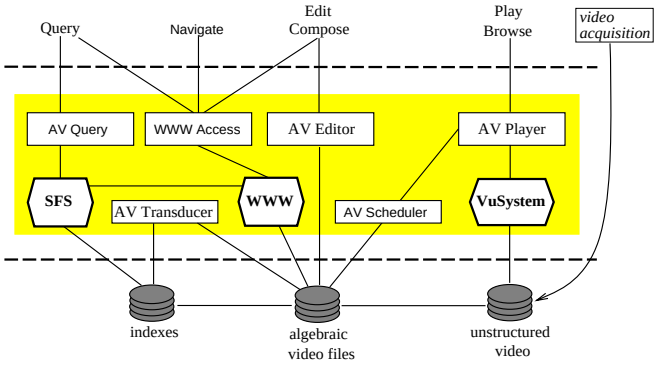

Figure 13: Algebraic Video System Implementation

The algebraic video system provides the following functions:

- acquisition of video data from external sources (such as TV broadcasts, or other video collections),
- parsing the raw unstructured video to algebraic video files,
- indexing of algebraic video nodes,
- content-based access to the data,
- playback and browsing of the video expressions,
- user composition, reuse and editing of more complex video expressions.

The implementation is built on top of three existing subsystems: the VuSystem [22], the Semantic File System (SFS) [10], and the World-Wide-Web (WWW) [4]. 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 [16] scripts control C++ classes and offer a programmable user interface that can be customized. The VuSystem is used for managing raw video data and for its support for TCL programming. The Semantic File System is used as a storage subsystem with content-based access to data for indexing and retrieving files, which represent algebraic video nodes. The WWW server provides a graphical interface to the system that includes facilities for query, navigation, editing and composing video, and invoking the video player. The *WWW access* module includes static HTML documents and a set of TCL scripts that dynamically create HTML documents in response to user interaction. Figure 14 illustrates how a user examines an algebraic video node using Mosaic and the WWW interface. The *HTML* document shown includes a snapshot of the playback of the node, the associated video expression, and links to other nodes with ancestral relationships. Currently, the parsing of raw video to algebraic video nodes is carried out manually.

All video algebra operations in Tables 1 and 2 except *delay*, *limit*, *parallel-end*, *transition* and *hide-content* have been implemented. The temporal attributes have not been implemented yet. The acquisition of video data, the associated close-captioned text, shot segmentation and parsing uses the VuSystem support. Figure 15 illustrates two different snapshots of the browser playing the same algebraic video file. The first snapshot contains the main window with a segment from CNN Headline News which is overlayed with a preview of a basketball game. Below the main windows are previews of two popular films. The second snapshot is taken some time later. The original main window has disappeared and the configuration of some of the windows has changed. However, the basketball preview and an excerpt from the movie "Hoffa" are still present.

4.1 Content-Based Access

To support content-based access, close-captioned text associated with the video stream is extracted if available and entered into segment descriptions as the `text` attribute. The user can add more attributes such as `title, author,` and `actor`, and organize nodes into a desired hierarchy using video algebra operators. Since the algebraic video files are stored in a human-readable, semi-structured file format, the user can edit and create algebraic video files using any available text editor. We are working on supporting the segmentation of the raw footage using the VuSystem shot detection module and *a-priori* knowledge of the video stream structure.

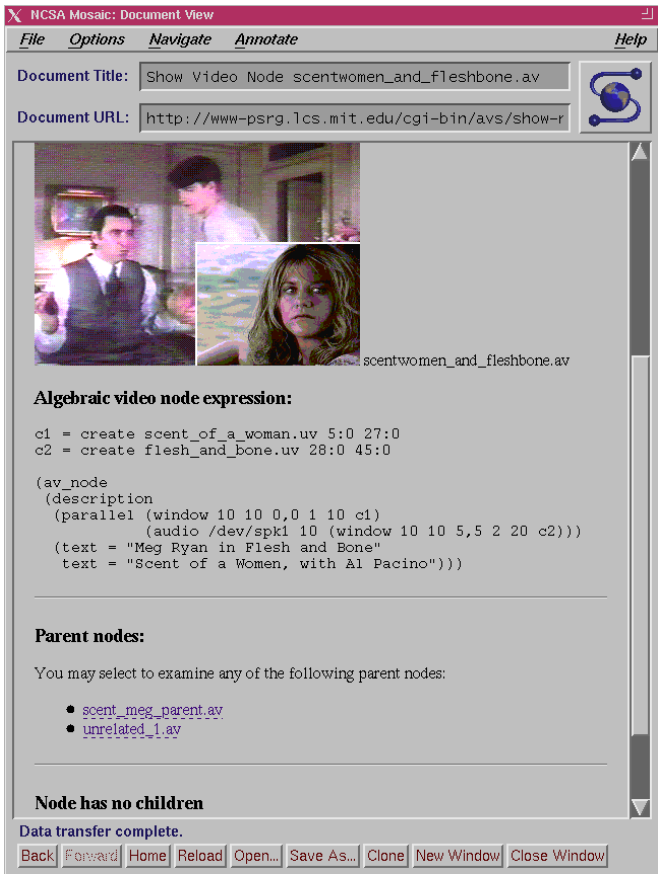

Figure 14: Examining a node using Mosaic and the WWW

The system indexes the video files to create the correspondence between the attributes and the algebraic video nodes. We implemented an *algebraic video transducer* in the Semantic File System to extract attributes from the descriptions stored in algebraic video files. The transducer is used in the indexing process to associate attributes and values with the algebraic video files. Indexing this information allows efficient querying and retrieval of relevant video segments. Individual video nodes that overlap are indexed by the system separately.

The *AV Query* (shown in Figure 2) and *WWW* modules, as well as the user, communicate with SFS directly via the pathname interface. SFS interprets a given file pathname as an attribute query. It then returns the result in a dynamically created *virtual directory* that contains the set of matching algebraic video nodes.

Figure 15: Browser Snapshots

4.2 Playback

Once the user selects a nested hierarchy of algebraic video nodes for playback, the system recursively parses the nodes and compiles a schedule file for each node. A schedule file is as a TCL script that consists of window and audio output declarations and a segment activation script. The system implements *playback* by dynamically interpreting the schedule files to produce streams of digital video. The streams are transmitted to the VuSystem, which then displays the digital video on the client workstation. Some of the playback information can be determined offline. For example, the *concatenation* and *parallel* operations on raw video segments will always result in the same video stream. However, other composition alternatives, such as *conditional* or a live video feed coupled with the *union* operation, require the system to dynamically modify the playback characteristics.

4.3 Experience

We have experimented with our prototype system to gain insight into the algebraic 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. The indexing process also included associated close-captioned text when it was available. Our Algebraic Video System provides rapid attribute-based access to the video collection and allows browsing a video result set. Once an algebraic video node has been selected, the user can examine the encompassing video context by following links to the node's parents and its children. New video presentations can be created from the existing video collection with query based discovery and algebraic combination of video segments of interest. The user can edit an existing video node, or interactively enter a video expression and preview its presentation.

The prototype delivers reasonable performance of query access and video playback. For a result set of twenty-five video segments, the elapsed time between query invocation by the user and system response is less than five seconds. Our experience is from running the query client and the video player on a SparcStation 10, the query and file server on an SGI PowerSeries 4D/320S, and the HTTP server on a different SparcStation 10. The three machines use an ethernet local area network for communication. The sys-

tem 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 have also provided the prototype WWW interface to the community. Users with WWW clients that support HTML forms can now edit and compose algebraic video nodes, and immediately view the resulting presentations. Note that in the current implementation, the HTTP server executes the algebraic video player locally, and uses X Windows to display the video stream at the client screen. Future versions of the prototype will support local execution of the video player that uses client side caching of the video presentation.

5 Conclusions

We define a video algebra for expressing unique compositions of temporal relationships between component video expressions, defining output characteristics of the video streams, and associating content descriptions with the video. We also describe how semantic information about video can be structured and used for content-based access. The semantically rich model of algebraic 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.

We have built a prototype system that we have used to retrieve video segments and to browse our video collection. Our experience suggests that algebraic video enables efficient access and management of video collections in interesting and diverse ways. From our experience thus far, we believe that the algebraic video data model is an adequate abstraction for representing digital video and supporting content-based access.

The algebraic model can be extended to multimedia documents that temporally and spatially combine text, pictures, audio and video. The multimedia document algebra must also model asynchronous user actions that can affect the playback of the user presentation. These documents may also include *hypermedia links* that can be instantiated in video expressions. A user may traverse these links to related video nodes that exist in different collections. We are extending the algebraic video system to provide an Internet Video Server with content-based access. We also plan to examine object-oriented database support for algebraic video storage. Another area of future research is the exploration of interactive movies and home video editing.

Acknowledgments

We are grateful to Chris Lindblad, David Tennenhouse for providing and supporting the VuSystem, and to our readers: James O'Toole, Mark Sheldon and Franklyn Turbak.

References

[1] Adobe Systems Incorporated, Mountain View, CA. *Adobe Premiere User Guide*, first edition, 1991.

[2] 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.

[3] 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.

[4] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-wide web: The information universe. *Electronic Networking*, 2(1):52–58, 1992.

[5] A. S. Bruckman. Electronic scrapbook: Towards an intelligent home-video editing system. Master's thesis, Massachusetts Institute of Technology, September 1991.

[6] M. Davis. Media Streams: An iconic visual language for video annotation. In *Proc. IEEE Symposium on Visual Languages*, pages 196–202, Bergen, Norway, 1993.

[7] DiVA Corporation, Cambridge, MA. *DiVA VideoShop User's Guide*, 1991.

[8] E. Fiume, D. Tsichritzis, and L. Dami. A temporal scripting language for object-oriented animation. In *Proc. Eurographics 1987*, pages 283–294, Amsterdam, Netherlands, August 1987.

[9] S. Gibbs, C. Breiteneder, and D. Tsichritzis. Audio/Video databases: An object-oriented approach. In *Proc. 9th IEEE Int. Data Engineering Conference*, pages 381–390, 1993.

[10] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. 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.

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

[12] T.D.C Little et al. A digital on-demand video service supporting content-based queries. In *Proc. First ACM International Conference on Multimedia.*, pages 427–436, Anaheim, California, August 1993.

[13] W. E. Mackay and G. Davenport. Virtual video editing in interactive multimedia applications. *Communications of the ACM*, 32(7), July 1989.

[14] MacroMind. *Director Version 2.0*, 1990.

[15] Akio Nagasaka and Yuzuru Tanaka. Automatic video indexing and full-video search for object appearances. In *Visual Database Systems, II*, pages 113–127. Elsevier Science Publishers, 1992.

[16] 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.

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

[18] R. Snodgrass. The temporal query language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[19] International Standard. Information technology hypermedia/time-based structuring language (hytime). ISO/IEC 10743, November 1992.

[20] 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.

[21] 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.

[22] D. K. Tennenhouse et al. A software-oriented approach to the design of media processing environments. In *Proc. IEEE International Conference on Multimedia Computing and Systems.*, Boston, MA, May 1994.

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

[24] G. van Rossum et al. CMIFed: A presentation environment for portable hypermedia documents. In *Proc. First ACM International Conference on Multimedia.*, pages 183–188, Anaheim, CA, August 1993.