Composition and Search with a Video Algebra

Ron Weiss
MIT Laboratory for Computer Science

Andrzej Duda
CNRS-Imag and INRIA

David K. Gifford
MIT Laboratory for Computer Science

A new data model called algebraic video provides operations for composing, searching, navigating, and playing back digital video presentations. A prototype system helps find video segments of interest from existing collections and create new video presentations with algebraic combinations of these segments.

As digital video becomes ubiquitous and 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 in 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 video or of hierarchical relationships between video segments. Moreover, they do not support flexible, associative access based on the structure and hierarchy. For these reasons, we developed the idea of algebraic video to let users 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, and
- specify multistream viewing.

Algebraic video also integrates content-based access to video, allowing the user to

- associate content information with logical video segments,
- provide multiple coexisting views and annotations of the same data, and
- provide associative access based on the content, structure, and temporal information.

A new data model for algebraic video must integrate both content attributes and semantic structure of the video. It 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, algebraic video can employ other forms of information extraction. 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 and indexes these semantic structure attributes, along with content information, for content-based access.

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. Algebraic video preserves the correspondence between video segments so that all relevant segments and their neighbors can be found efficiently. The output characteristics of video expressions are media-independent, so the rendering can adjust to the available resources.

1070-986X/95/$4.00 © 1995 IEEE

Users can search algebraic video collections for relevant presentations with queries that describe the 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. In addition to content-based access, the model allows users to browse and explore the structure of the video expressions, which helps them understand the surrounding organization and context. For example, the user can find an interesting expression and then examine the neighboring video segments. Furthermore, users can create individual interpretations of existing video footage by composing new video expressions from the existing video components.

Video projects

The algebraic video data model offers the following important advances over previous digital video representations such as MHEG and HyTime (see sidebar):

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

We implemented our model as part of the Algebraic Video System project. The system allows users to compose algebraic video presentations. It extracts video attribute information and offers a query-based interface for searching, browsing, and playing back relevant video segments (see Figure 1, next page). 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.

Projects related to ours include video authoring and annotation tools, systems that provide content-based access to video, and tools for modeling unstructured video for content-based retrieval.

Authoring and annotation systems

Various tools provide facilities for composing

Hypermedia standards

When constructing our system, we examined two standards: MHEG and HyTime. The MHEG standard is intended for "coded representation of final form multimedia and hypermedia objects that will be interchanged across service and applications."[1] At the core of the standard are MHEG objects (represented in Figure A) 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, spatiotemporal, and close system synchronization.

The HyTime[2] hypermedia standard provides a mechanism for specifying hyperlinks and scheduling multimedia information in time and space. Based on the Standard Generalized Markup Language (SGML), HyTime uses architectural forms to express rules for hypermedia structuring information. These architectural forms and attributes of information objects are grouped into six modules: base, measurement, location address, hyperlinks, scheduling, and rendition. The scheduling module allows events, defined as occurrences of information objects, to be scheduled in finite coordinate spaces (FCS). The user expresses spatial and temporal positions of objects in FCS using coordinate axes or relationships. The rendition module maps the FCS representation to its real-world counterpart to play back the multimedia information.

As stated in their descriptions, both MHEG and HyTime are intended for final formatted documents and lack mechanisms for content-based access or editing and annotating the multimedia data.

References

1. R. Price, "MHEG: An Introduction to the Future International Standard for Hypermedia Object Interchange," *Proc. First ACM Int'l Conf. on Multimedia*, ACM, New York, 1993, pp. 121-128.
2. "International Standard: Information Technology Hypermedia/Time-Based Structuring Language (HyTime)," ISO/IEC 10743, Nov. 1992.

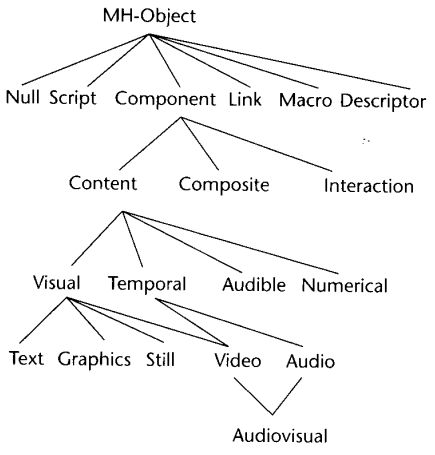

Figure A. The MHEG object inheritance tree.

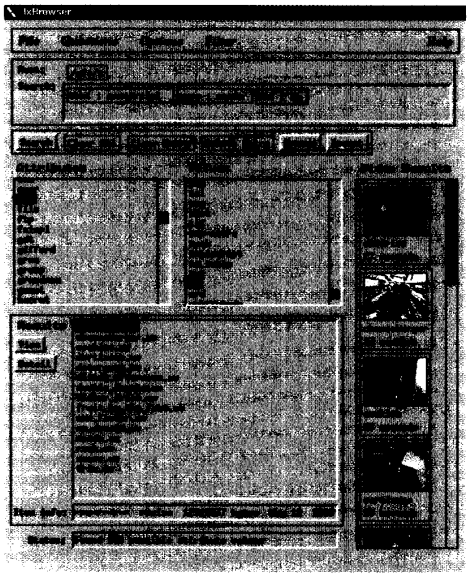

Figure 1. The Algebraic Video Query Interface lets users search, browse, and play back video segments based on various attributes.

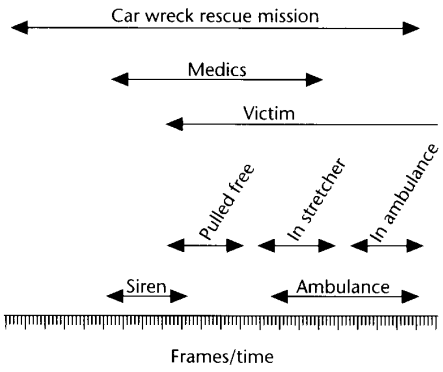

Figure 2. Simple stratification assigns descriptions to video footage, allowing easy retrieval by keyword. However, the strata lack context, since relationships are not preserved.

and annotating complex video presentations. Smith and Davenport[1] implemented a video annotation system that 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 2 shows an example of video footage annotated by strata. Strata are stored in files accessible by a simple keyword search. A user can find a sequence of interest, but cannot easily determine the context in which it appears because of the absence of relationships between the strata. Unlike simple stratification, the algebraic video model preserves the nested relationships between strata and allows users to explore the context in which a stratum appears.

Commercially available tools such as Adobe Premiere, Diva VideoShop, and MacroMind Director allow the user to create movies using audio and video tracks, and also to specify special effects during video segment transitions. These commercial systems are based on two distinct paradigms: *scripts* and *timelines*. The script (or flowchart) approach requires the author to explicitly program timing and placement information. 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 placed independently in the timeline. Normally, a direct-manipulation graphical editor similar to the one illustrated in Figure 3 (artificially created for the purposes of this discussion) presents the author with video and audio tracks, plus 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 marked by time indices that reflect the time elapsed since the beginning of the video presentation.

Such toolkits allow the user to edit video data in essentially the same manner as filmmakers 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 videotape, 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.

Multimedia authoring systems such as CMIFed[2] have rich structuring primitives for multimedia documents, but fail to address the structure of the video data itself. The video is still treated as an unstructured linear stream. Hamakawa and Rekimoto[3] propose a multimedia authoring system that supports editing and reuse of multimedia data. Their system, based on a hierarchical and compositional model of multimedia objects, 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 multilayered, iconic annotations of video content.[4] Icons denoting objects

and actions are organized into cascading hierarchies of 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 timeline. Currently, around 2,200 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 closed-captioned text.

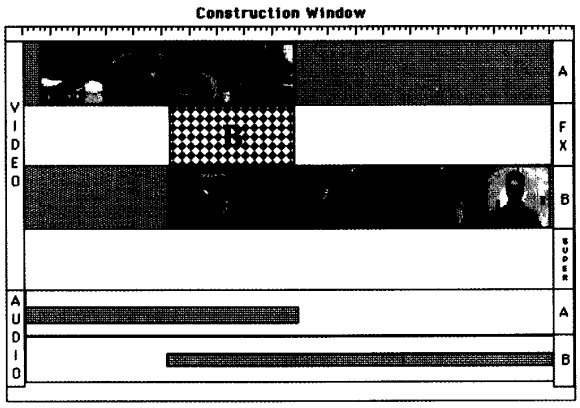

Figure 3. A timeline editor such as this artificially created example often provides tracks for drag-and-drop arrangement of video and audio clips, plus transitions with special effects.

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 not follow a strictly linear progression determined a priori. Commercial toolkits do not take advantage of this distinctive feature. Moreover, they lack methods for specifying the elaborate logical structure of video data and do not address content-based access. Our approach allows structured, multistream composition using video algebra operations and content-based access.

Content-based access systems

Content-based access systems provide facilities to discover video segments of interest. Little et al.[5] implemented a system that supports content-based retrieval of video footage. They define a specific data scheme composed of Movie, Scene, and Actor relations with a fixed set of attributes. The system requires manual feature extraction, then fits these features into the data scheme. It permits queries on the attributes of movie, scene, and actor. Having selected a movie or a scene, a user can scan from scene to scene. The data model and the virtual video browser are limited because descriptions cannot be assigned to overlapping or nested video sequences as in the stratification model. Moreover, the system is focused on retrieving previously stored information and is not suitable for users who need to create, edit, and annotate a customized view of the video footage.

Electronic Scrapbook is a system for home-video video annotation and editing,[6] where the annotations can later be used for content-based access. Users can attach descriptions to video clips and use a modified form of case-based reasoning to edit and create personalized video stories. They can query a database of video clips and filter, sort, or remove overlapping segments from the results. The system uses a small, special-purpose taxonomy usable in descriptions, but does not exploit the logical structure of video. For example, users cannot describe hierarchical relationships where video segments are nested.

Gibbs et al.[7] proposed 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. Because the database values (audio or video) are linear sequences of data elements, the logical structure is not represented. Also, the temporal composition mechanism is essentially equivalent to the timeline paradigm.

Unstructured video extraction systems

Video captured into digital format from an analog source initially exists as an unstructured sequence of video frames and audio segments. Several proposed systems extract information from these unstructured streams and provide a data model for content-based access. Swanberg et al.[8] defined such an architecture for parsing data semantics from the video stream. The system manages a fixed data scheme 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 related sequence of shots. 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 information about the segmentation of the video footage and the objects and features in the video, facilitating query optimization.

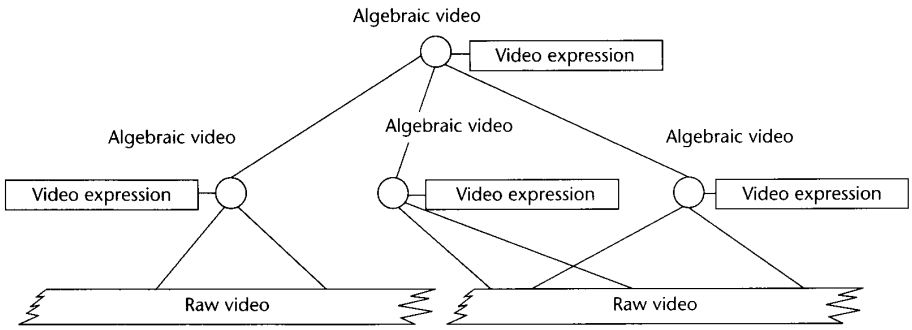

Figure 4. Algebraic video abstraction provides an efficient way to organize and manipulate video.

The rather inflexible data scheme used for this system is not suitable for free-form modeling of the complex relations between video segments.

Nagasaka[9] 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 acts 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.

Algebraic video model

The algebraic video data model consists of hierarchical compositions of video expressions with high-level semantic descriptions. The video expressions are constructed using video algebra operations. We introduce video algebra as a means of combining and expressing temporal relations, defining the output characteristics of video expressions, and associating descriptive information with these expressions.

In general, video is composed of different story units such as shots, scenes, and sequences arranged according to some logical structure. 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, such as closed-captioned text and key frames that characterize a shot. 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 through four activities: Edit and Compose, Play and Browse, Navigate, and Query. The operations that support playback, navigation, and content-based queries are grouped together as the *interface operations*.

In the algebraic video data model, the fundamental entity is a *presentation*, a multiwindow spatial, temporal, and content combination of video segments. Presentations are described by video expressions. The most primitive video expression creates a single-window presentation from a raw video segment. These segments are specified using the name of the raw video and a range within it (see Figure 4). Compound video expressions are constructed from simpler ones using video algebra operations. Video expressions can be named by variables, can be composed to reflect the complex logical structure of the presentations, and can 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.

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

We call our approach algebraic video because the operators used to construct video expressions have associated axioms. For example, many of our binary operations such as Union are associative. These axioms provide an algebraic video implementation flexibility on expression evaluation order and common subexpression processing.

Video algebra

The video algebra operations fall into four categories:

Table 1. Video algebra operations.

	Usage	Function
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	$E_1 \parallel_{\#} E_2$	Defines the presentation where E_1 and E_2 are played concurrently and terminate simultaneously
Conditional	(`test`) ? $E_1 : E_2 : \ldots : E_i$	Defines the presentation where E 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*	Sets the duration of the presentation equal to *factor* times duration of E_1 by changing the playback speed of the video expression
Limit	`limit` E_1 *time*	Sets the duration of the presentation 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 the bottom-left corner (x_1, y_1) and the right-top corner (x_2, y_2) 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, command overrides `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

1. *Creation* defines the construction of video expressions from raw video.

2. *Composition* defines temporal relationships between component video expressions.

3. *Output* defines spatial layout and audio output for component video expressions.

4. *Description* associates content attributes with a video expression.

Table 1 presents the video algebra operations. The arguments denoted by E_1, E_2, ..., 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 here, expressions of the form $(E_1 \oplus E_2 \oplus E_3 \oplus \ldots E_n)$, where $\oplus$ is any specific binary operation, denote an expression of the form $(\ldots ((E_1 \oplus E_2) \oplus E_3) \oplus \ldots E_n)$. Also note that the binary video algebra operations are inherently not commutative because they include a temporal component.

Composition. The composition operations can be combined to produce complex scheduling definitions and constraints. The Union operation allows the user to easily construct a nonrepetitive video stream from overlapping segments while preserving the temporal ordering of the component expressions. If these expressions do not contain overlapping segments, then Union is equivalent to Concatenation. The following pseudocode is an example of an algebraic video node that uses the Union operation to compose a video expression:

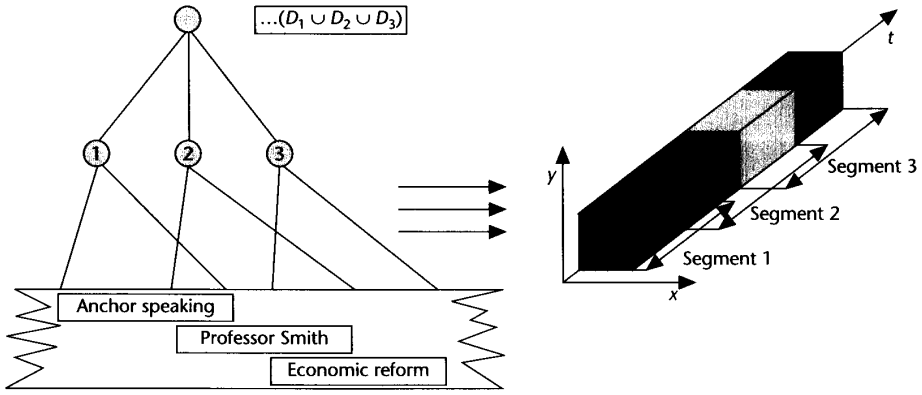

Figure 5. This example of the Union operation creates a nonredundant video stream from three overlapping segments.

C_1 = `create` Cnn.HeadlineNews.rv 10 30
C_2 = `create` Cnn.HeadlineNews.rv 20 40
C_3 = `create` Cnn.HeadlineNews.rv 32 65

D_1 = (`description` C_1 "Anchor speaking")
D_2 = (`description` C_2 "Professor Smith")
D_3 = (`description` C_3 "Economic reform")

($D_1 \cup D_2 \cup D_3$)

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, as Figure 5 shows.

The Intersection operation enables the user to construct from two arguments a new video presentation that includes only footage contained in both. The Parallel operation allows the user to compose multiwindow, concurrent video presentations. The Stretch operation changes the playback speed of the video presentation, but does not alter the playback speed of other presentations.

The Conditional operation can be used for personalized viewing or other viewings that external sources can affect. The test expression in the conditional operation must evaluate to an integer. However, it is easy to map noninteger test expressions, such as a user's environment variable, time of day, weather patterns, and input, 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, by logically structuring the video and allowing the user to choose segments based on interaction or prior specification.

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* set to 0.

The Contains operation permits the user to include the results of a query in 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 are explained in "Content-based access," below.

We are investigating other algebraic video composition operations, including those that will achieve overlay of video streams, synchronization on events, a general synchronization operator (similar to operators defined by Fiume et al.[10]), and nondeterminism.

Output characteristics. Because multiple video streams can be scheduled to play at any specific time within one video expression, 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. Since expressions can be nested, the spatial layout of any particular video expression is defined relative to the *parent rectangle*, 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. The following command sequence

```
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))
```

gives an example of an algebraic video node with nested window specifications. Figure 6 shows a snapshot of this example, captured during playback.

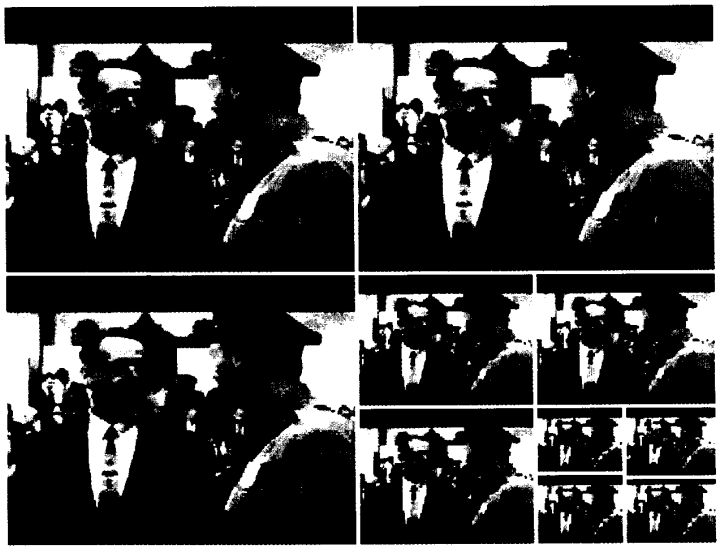

Figure 6. Output characteristics determine how video expressions will appear during playback, as this snapshot of the node with nested windows shows.

Window priorities (see Table 1) 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.

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 of Audio is defined in a manner analogous to the Priority parameter of the Window operation.

Descriptions. The model permits the association of arbitrary descriptions with a given video algebra expression. It allows textual descriptions, nontextual descriptions like key frames, icons, and salient stills, and image features like color, texture, and shape. The Description operation associates content information with a video expression.

The Content description of an expression is not fixed by our model. However, for the purposes of this article and our prototype, a Content is a Boolean combination of attributes that consists of a field name and a value. An example of an attribute is `title = "CNN Headline News"`. Some field names have predefined semantics—for example, `title`—while other fields are user-definable. 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 associated with a video expression. For example, a description may initially contain closed-captioned text. The user may add other attributes, such as actor, characters, and scene summary. The components of a video expression inherit descriptions by context, which implies that all the content attributes associated with some parent video node are also associated with all its descendant nodes.

Table 2. Interface operations.

	Usage	Definition
Content-based access		
Search	`search` *query*	Searches a collection of nodes for video expressions that match *query*
Browsing		
Playback	`playback` *video-expression*	Plays back the video expression
Display	`display` *video-expression*	Displays 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

The Hide-content operation defines a video expression E that does not contain any descriptions. The Contains and Search operations (see "Content-based access," below) on E do not recursively examine the components of E. The Hide-content operation provides a method for creating abstraction barriers for content-based access.

Interface operations

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

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 a collection of persistent algebraic video nodes. For querying within the collection, we chose a simple predicate query language. A query is a Boolean combination of attributes. When the Search operation applies a query to the collection, it searches the hierarchy of every node in the collection and returns the result set of nodes that satisfy the query. Note that the recursive Search does not examine subhierarchies 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.

The 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 to ensure that complete subhierarchies 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 7. 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 play back any of the expressions in the set or 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.

Browsing and navigation. Browsing operations enable the user to inspect the video expression and to view the presentation defined by the expression. The user can play back any expression or 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.

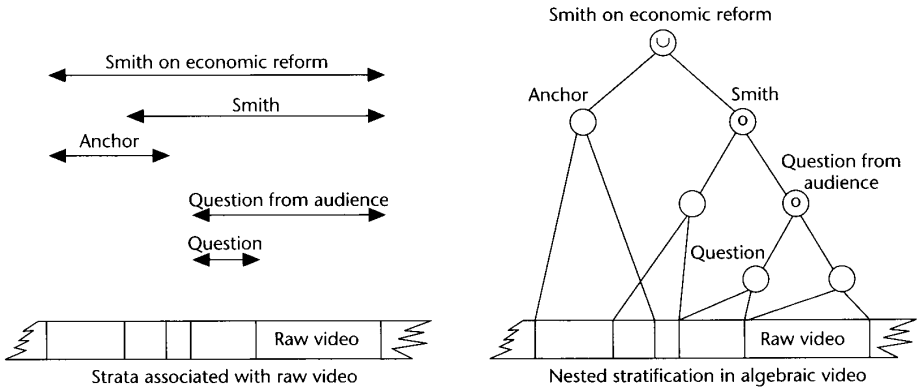

Figure 7. Nested stratification preserves the context of video expressions. It is integrated into the video algebra in a manner that prevents redundancy in playing back video presentations.

Nested stratification

Hierarchical relations between the algebraic video nodes allow nested stratification. Smith and Davenport[1] defined 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 provide multiple coexisting views and annotations and allow 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 (see "Video authoring and annotation," above), the algebraic video model preserves nested relationships between strata and allows the user to explore the context in which a stratum appears. The nested algebraic video nodes preserve the structural composition of the presentation while allowing various content and structural interpretations for overlapping footage to coexist (see Figure 7).

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, which combines overlapping nodes, guarantees that there will be no repetition of video footage during playback.

Algebraic Video System prototype

The Algebraic Video System is a prototype implementation of the algebraic video data model and its associated operations. The system provides support for algebraic video composition and content-based access. 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 segments. In the prototype, the units of storage and indexing are the algebraic video nodes. These nodes are textually represented by human-readable, semistructured algebraic video files. The system has a graphical user interface for managing a collection of raw video segments and algebraic video nodes; it 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 8 (next page) presents the architecture of the implementation.

The algebraic video system provides the following functions:

- acquiring video data from external sources such as TV broadcasts or other video collections,
- parsing the raw, unstructured video to algebraic video files,
- indexing algebraic video nodes,
- providing content-based access to the data,
- playing back and browsing the video expressions, and
- composing, reusing, and editing complex video expressions.

The implementation is built on top of three existing subsystems: the VuSystem,[11] the Seman-

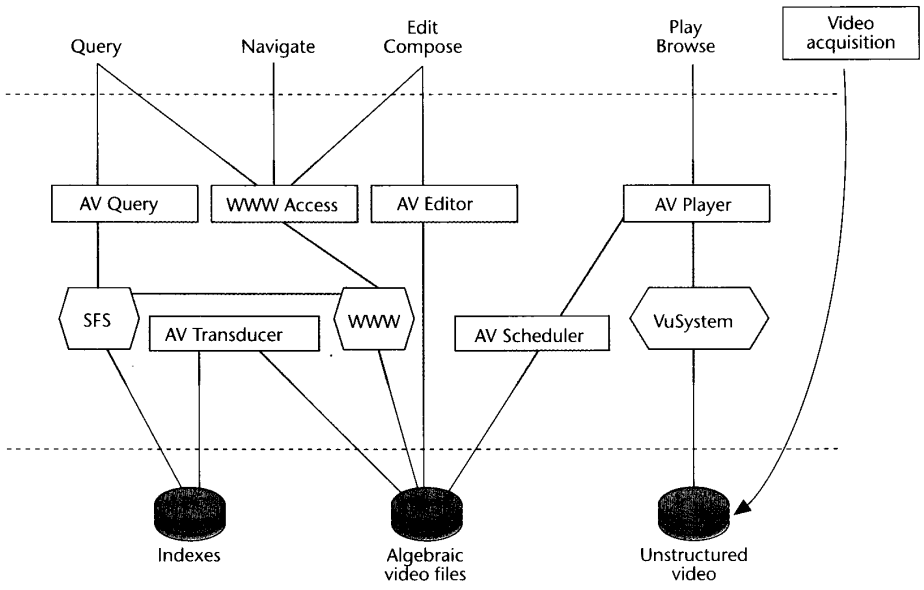

Figure 8. Our implementation incorporates the VuSystem, Semantic File System (SFS), and World-Wide Web (WWW) modules into the Algebraic Video System prototype.

tic File System (SFS),[12] and the World-Wide Web (WWW).[13] The VuSystem provides an environment for recording, processing, and playing video. A set of C++ classes manages basic functions such as synchronizing video streams, displaying in a window, and processing video streams. Tool command language (Tcl) scripts[14] control C++ classes and offer a programmable, customizable user interface. The VuSystem is used for managing raw video data and for its support of Tcl programming. The Semantic File System is used as a storage subsystem with content-based access to data for indexing and retrieving files that represent algebraic video nodes. The WWW server provides a graphical interface to the system that includes facilities for querying, navigating, video editing and compositing, and invoking the video player. The WWW access module includes static hypertext markup language (HTML) documents and a set of Tcl scripts that dynamically create HTML documents in response to user interaction.

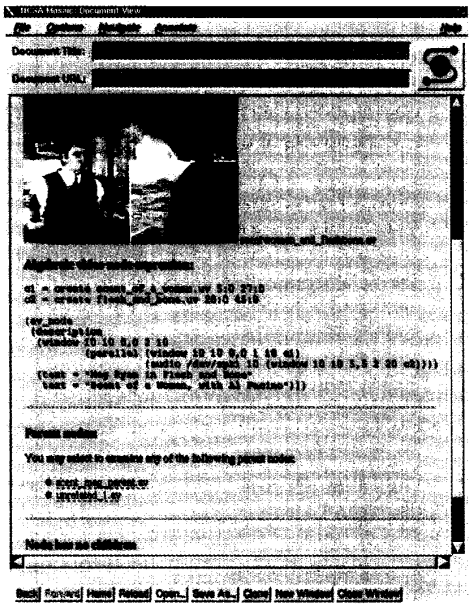

Figure 9. Mosaic and the WWW provide a graphical interface and access tools for the Algebraic Video System.

Figure 9 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 acquisition of video data and associated closed-captioned text, shot segmentation, and parsing use VuSystem support. Figure 10 shows 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 overlaid with a preview of a basketball game.

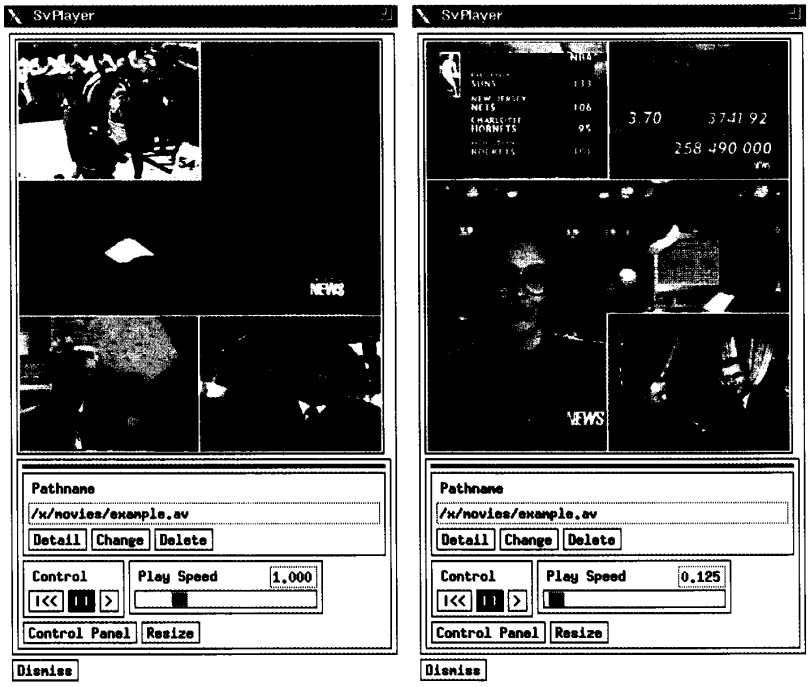

Figure 10. The Algebraic Video System browser displays two snapshots of a sample multiwindowed presentation.

Below the main windows are previews of two popular films. In the second snapshot, 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.

Content-based access

To support content-based access, the Algebraic Video System prototype extracts any closed-captioned text associated with the video stream and enters it into segment descriptions as the `text` attribute. The user can add more attributes such as `title`, `author`, and `actor`, and organize nodes into a hierarchy using video algebra operators. Since the algebraic video files are stored in a human-readable, semistructured file format, the user can edit and create algebraic video files using any available text editor. We are working on supporting the segmentation of raw footage using the VuSystem shot detection module and a priori knowledge of the video stream structure.

The system indexes the video files to create 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 associates attributes and values with the algebraic video files during the indexing process. Indexing this information allows efficient querying and retrieval of relevant video segments. Individual video nodes that overlap are indexed separately.

The AV Query (shown in Figure 1) and WWW modules, as well as the user, communicate with Semantic File System directly via the pathname interface. Semantic File System interprets a 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.

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. A schedule file is a Tcl script consisting 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 off-line. 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.

Experience

We created 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 closed-captioned text when available. Our Algebraic Video System provides rapid attribute-based access to the video collection and allows browsing of a video result set. Once users select an algebraic video node, they can examine the encompassing video context by following links to the node's parents and 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 for query access and video playback. For a result set of 25 video segments, the elapsed time between query invocation by the user and system response is less than five seconds. 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 ran the query client and the video player on a Sun SparcStation 10, the query and file server on a Silicon Graphics PowerSeries 4D/320S, and the hypertext transfer protocol (HTTP) server on a different SparcStation 10. The three machines communicated over an Ethernet local area network.

We also provided the prototype WWW interface to the Internet 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, using 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.

Conclusions

Our experience with the Algebraic Video System prototype suggests that algebraic video enables efficient access and management of video collections in interesting and diverse ways. From our experience so 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. **MM**

Acknowledgments

We are grateful to Chris Lindblad and David Tennenhouse for providing and supporting the VuSystem, and to readers James O'Toole, Mark Sheldon, and Franklyn Turbak. This work was supported by the Defense Advanced Research Projects Agency and the Department of the Army under contract DABT63-92-C-0012, and by an equipment loan from Forest Baskett at Silicon Graphics.

References

1. T.G. Aguierre Smith and G. Davenport, "The Stratification System: A Design Environment for Random Access Video," *Proc. 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, Springer Verlag, New York, 1992, pp. 250-261.
2. G. van Rossum et al., "CMIFed: A Presentation Environment for Portable Hypermedia Documents," *Proc. First ACM Int'l Conf. on Multimedia*, ACM, New York, 1993, pp. 183-188.
3. R. Hamakawa and J. Rekimoto, "Object Composition and Playback Models for Handling Multimedia Data," *Proc. First ACM Int'l Conf. on Multimedia*, ACM, New York, 1993, pp. 273-281.

4. M. Davis, "Media Streams: An Iconic Visual Language for Video Annotation," *Proc. IEEE Symp. on Visual Languages*, CS Press, Los Alamitos, Calif., 1993, pp. 196-202.
5. T.D.C Little et al., "A Digital On-Demand Video Service Supporting Content-Based Queries," *Proc. First ACM Int'l Conf. on Multimedia*, ACM, New York, 1993, pp. 427-436.
6. A.S. Bruckman, *Electronic Scrapbook: Towards an Intelligent Home-Video Editing System*, master's thesis, Massachusetts Inst. of Technology, Sept. 1991.
7. S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Audio/Video Databases: An Object-Oriented Approach," *Proc. 9th IEEE Int'l Data Eng. Conf.*, CS Press, Los Alamitos, Calif., 1993, pp. 381-390.
8. D. Swanberg, C.F. Chu, and R. Jain. "Knowledge Guided Parsing in Video Databases," *IS&T/SPIE Symp. on Electronic Imaging: Science & Technology*, SPIE, Bellingham, Wash., 1993.
9. A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances," in *Visual Database Systems II*, Elsevier Science Publishers, 1992, pp. 113-127.
10. E. Fiume, D. Tsichritzis, and L. Dami, "A Temporal Scripting Language for Object-Oriented Animation," *Proc. Eurographics 1987*, Elsevier, North Holland, 1987, pp. 283-294.
11. D.K. Tennenhouse et al., "A Software-Oriented Approach to the Design of Media Processing Environments," *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, CS Press, Los Alamitos, Calif., 1994, pp. 435-444.
12. D.K. Gifford et al., "Semantic File Systems," *Thirteenth ACM Symp. on Operating Systems Principles*, ACM, New York, 1991. Available in *Operating Systems Rev.*, Vol. 25, No. 5.
13. T. Berners-Lee et al., "World-Wide Web: The Information Universe," *Electronic Networking*, Vol. 2, No. 1, 1992, pp. 52-58.
14. J.K. Ousterhout, "An X11 Toolkit Based on the Tcl Language," *Usenix Assoc. 1991 Winter Conf. Proc.*, Usenix Assoc., Berkeley, Calif. , 1991, pp. 435-483.

Ron Weiss is a doctoral candidate in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. He received a BA in computer science and economics from Brandeis University in 1992 and an SM in electrical engineering and computer science in 1994 from MIT. His research interests include digital video, multimedia authoring, distributed systems, mobile computing, resource discovery, and distributed information systems. He is a member of Sigma Xi and Phi Beta Kappa.

Andrzej Duda is a Chargé de Recherche at the Centre National de la Recherche Scientifique in conjunction with the Imag institute in Grenoble, France. Duda is also with Bull-Imag Systémes and INRIA. He received the Dr.-Ing. degree from the Université de Paris-Sud in 1984. This work was done when the author was a visiting scientist at the MIT Laboratory for Computer Science. His current research interests include distributed multimedia systems, resource discovery, and distributed information systems.

David Gifford is a professor of electrical engineering and computer science at MIT. He received a BS from MIT and an MS and PhD from Stanford University. His research interests include multimedia information repositories, resource discovery tools, programming languages, distributed information systems, and computational aspects of molecular biology. He is a member of the IEEE Computer Society, ACM, and AAAS.

Readers can contact Weiss and Gifford at MIT, Laboratory for Computer Science, NE43-441, 545 Technology Square, Cambridge, MA 02139, e-mail {rweiss, gifford}@lcs.mit.edu. Duda can be contacted at CNRS-Imag, 2, rue Vignate, 38610 Gières, France, e-mail andrzej.duda@imag.fr.