A Graphical Query Language for Identifying Temporal Trends in Video Data

Stacie Hibino
Elke Rundensteiner
Department of EECS
University of Michigan
Ann Arbor, MI 48109
hibino@eecs.umich.edu, rundenst@eecs.umich.edu

ABSTRACT

Previous work in video annotation and analysis has concentrated more on the process of annotation than analysis. The focus of our research instead is to provide support for video *analysis* by developing an environment for the "casual user" to query the video data using spatio-temporal characteristics and to review visual results for trend analysis. In this paper, we present our approach for identifying trends in video data via examining relationships between video annotations. This approach allows users to characterize the video in terms of relationships between events (e.g., events of type B frequently follow events of type A). The focus of this paper is on *temporal* relationships and trends. We present a graphical query language for specifying relative temporal queries between sets of annotations. This query language builds on the notion of dynamic query filters and significantly extends them. It can be used in visual information seeking to discover temporal trends in the video data.

1 INTRODUCTION

1.1 Motivation

Movies, news, MTV, Channel 1, literature classics, sports, home videos, and observational data—from the home to the classroom to the research laboratory, video is becoming more and more commonplace. Indexing, re-using, reviewing, or analyzing this video, however, can be a cumbersome and time-consuming process. Few tools exist to aid in this process, and the few which do are typically limited in one way or another. Users need to be able to index, parse, and make personal comments about a video—they need to be able to make *annotations* to it. And once they have created annotations, they should later be able to easily retrieve and analyze the video data by exploiting these annotations.

Uses for Video Annotation and Analysis

Since video can be recorded in various settings for a variety of purposes, video annotation and analysis are not restricted to a particular domain. Video annotation could be used to describe, clarify, and highlight different parts of a video. It could be used by educational researchers to describe and interpret classroom video data; by communications or film experts to identify different film shots and transitions in a movie; by physics teachers to overlay abstract physics diagrams on top of real-life objects, etc.

In this research, we are examining the annotation and analysis of video that has been collected as video data. Video is becoming more and more a common form of data collection. For example, video is collected for classroom studies, lab studies (e.g., during usability testing), and workplace studies. Researchers can use video data to study social interactions, individual user errors during usability studies, etc. One of the advantages of video data is that it preserves a lot of detailed information which would be difficult to capture by other means of data collection. The data collected by video, however, is qualitative, and as such can be used to provide typical and atypical scenarios, or can be coded and analyzed to identify trends and formulate quantitative results. Our research focuses on coding and analyzing video data via the use of video annotations.

Consider the case where educational researchers use video annotation for describing and interpreting video data collected in a classroom setting. For example, they might be interested in studying teacher effectiveness in the classroom. By using the annotation system, they can link notes directly to the video rather than keeping binders of notes on paper or on a separate system. They can also use annotations to code the video data (e.g., to indicate places in the video when the teacher is speaking vs. when the student is speaking). Later, they can use the annotations to find relevant video segments, as well as to gain an overall understanding of the information being captured by the video. In addition, they can create annotations using different types of media, such as text, audio, graphic objects, and bitmaps. This allows them to create text labels to identify objects and people, use graphic objects such as circles to highlight interesting situations, etc. Furthermore, by storing annotations in a database, researchers can search for trends in the video data by specifying queries to locate occurrences of, and relationships between the annotations (e.g., do students generally speak for shorter periods of time than the teacher?).

The Need for a Video Annotation and Analysis System

While existing authoring, multimedia, or video editing systems could be used for annotating video, there would be several difficulties in doing so. In the case of video editing, the original video is not preserved, and tools for indexing and accessing video segments are usually not provided. In order to use a multimedia authoring environment, users must know how to *program* in the authoring environment. Users should not have to learn to program to work with video. They should be able to annotate and access video as easily as they can annotate and access text. They need technology tools to treat video as a primary document. And, they need an easy-to-use method for annotating these video documents as well as techniques for retrieving and analyzing their annotations in the context of the annotated data.

Video annotation systems have recently been given increased attention. A few have been developed (Mackay, 1989; Roschelle, et al., 1990; Davis, 1994; (see section 4)), but they focus more on the mechanism of synchronizing annotations to video. That is, these systems focus on temporally linking annotations to video, but do not deal with, or are limited in dealing with, the spatial dimensions of video. In addition, these systems focus more on the process of annotating rather than analyzing the video. True video annotation should allow users to annotate video in three dimensions—temporally and spatially. And, subsequently, retrieval and analysis of these annotations should also take advantage of the temporal and spatial characteristics of the video. Our video analysis support environment is specifically being designed to meet these needs. In particular, we are using temporal-spatio annotations, coupled with graphical query and presentation languages to support the user in visually analyzing the video data.

1.2 Terms and Definitions

This section defines the terms video annotation and video analysis as they are used in the context of this paper.

Video annotations. A *video annotation* is an object that is temporally linked to one or more video segments and spatially linked to a position on top of or next to the video (i.e., to a position on the display relative to the video window). An annotation can be a text, audio or graphical (e.g., square, circle, arrow, etc.) object. In hypermedia terms, an annotation can be thought of as an object containing a link to an anchor in the video. When reviewing all, or a set of retrieved annotations, users can click on any annotation instance to access the corresponding video segment. Links are bi-directional in that when the video is played, annotations can be played or displayed at the appropriate times and (spatial) places. In addition to spatial and temporal characteristics, each annotation may also have some descriptive characteristics, including name, category and description. In this paper, the term *event* refers to the real-world situation being captured by the annotation.

Video Analysis. In this paper, *video analysis* refers to the process of identifying trends and relationships between events in the video data. This use of the term video analysis is in contrast to bit-level video analysis such as that used for object extraction (see Section 4 on related work).

1.3 Overview

The remainder of the paper is organized into four additional sections. Section 2 states the problem description and presents our approach for addressing it. Section 3 presents a temporal graphical query language for specifying relative temporal queries. Section 4 discusses related work. Finally, Section 5 presents a summary of contributions and discusses future work.

2 PROBLEM AND APPROACH

2.1 Overall Problem Description and Approach

When analyzing video data, researchers typically code (i.e., annotate) the video, and then perform analysis through an iterative process of examining various relationships between different types of events. The overall problem then is to provide support for creating annotations and for analyzing video by supporting the process of identifying temporal and/or spatial relationships between temporal-spatio video annotations. More specifically, users need to be able to create annotations and then query the annotation collection to search for relationships and data trends. The results retrieved from relative queries need to be visually presented to facilitate this searching for trends. Also, to aid in the trend searching process, and to identify statistical significance, users may also need to review quantitative summaries and statistical information.

Figure 1 presents our overall approach to this problem. The approach consists of four components: query, presentation, navigation, and quantitative and statisticial analysis. *Annotations* are layered on top of the *video* and stored in a database. Users incrementally specify relative *queries* through *graphical query languages* (GQL). Dynamic *presentations* provide visualizations of subsets of the annotation collection (e.g., a view of results retrieved from a query). Users specify the format of a presentation via a *presentation language* (PL). A *statistics calculator* (SC) is provided to specify different types of quantitative information. Finally, *navigation controls* allow users to navigate within the temporal presentations.

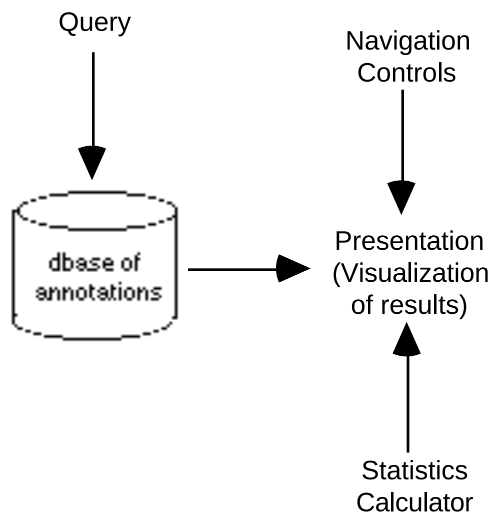


Figure 1. Overall Approach and System Description.

In this paper, we present our results of the specification of graphical query languages to support and simplify the analysis process for the users. More specifically, we will focus on a graphical query language for specifying *relative temporal* queries over video data.

2.2 Problem Statement

In general, a temporal graphical query language (TGQL) could be used in different contexts by various types of users. The language could be used by query language and system developers to extend existing languages or by interface designers to tailor query widgets to domain specific needs. A TGQL can also provide a simple query interface for application users. The goals of this paper are (1) to define a general TGQL that can be easily integrated into a variety of applications and (2) to design a TGQL interface that is easy and intuitive to use by application (i.e., video annotation and analysis) users who are typically query language novices.

Thus, the specific problem addressed in this paper is that of the users' need for a simple interface for specifying relative temporal queries to video data. *Relative* queries are necessary for examining relationships between events and thus for identifying trends. A *graphical* language is desired to correspond with the graphical nature of the objects in the database and is much more suitable for the type of novice query users we are targeting. In addition, we hypothesize that a temporal language is required to facilitate searching for temporal data trends.

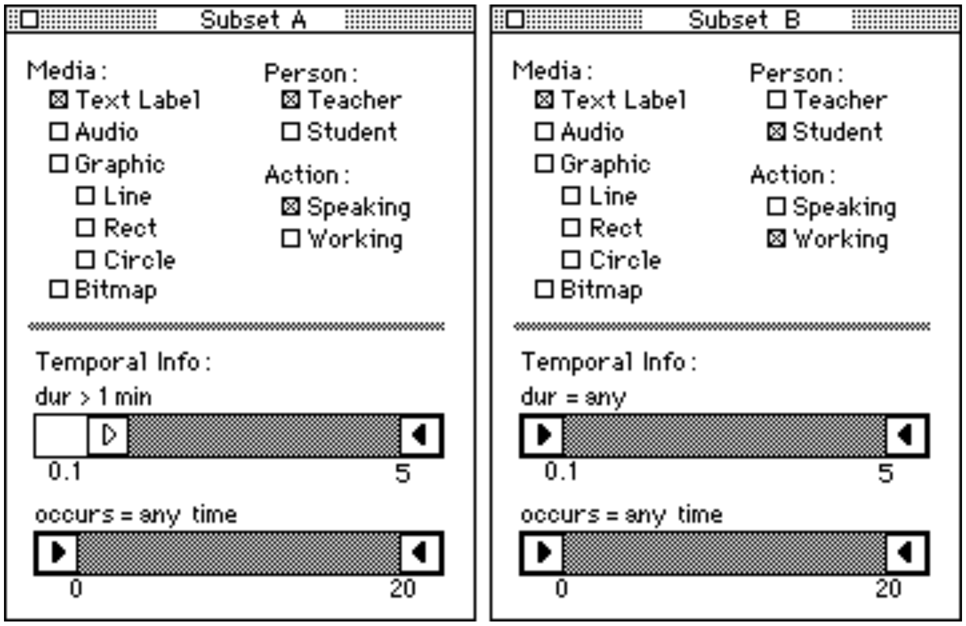
2.3 Background Paradigm

The process of analyzing video using video annotations involves incrementally and iteratively formulating queries and visually reviewing results for data trends. This process is similar to what Ahlberg and Shneiderman (1994) term visual information seeking (VIS). Ahlberg and Shneiderman describe VIS as a process for *browsing* database information. This is a process characterized by rapid filtering, progressive refinement, continuous reformulation of goals, and visual scanning to identify results. They have designed dynamic query filters to support rapid filtering, and starfield displays to support visual scanning. *Dynamic query filters* allow users to rapidly adjust query parameters via the use of sliders, buttons, etc. A *starfield display* presents a visual summary of query results. The starfield display is updated every time users make changes to any query filter. Thus, queries are expressed *incrementally* as users specify desired values for each record field. In addition to the dynamic links to the display, the query filters are also linked to one another to prevent the specification of null subsets. That is, when the user adjusts one query filter, the other filters are automatically updated to only include valid values with respect to the values of the current filter being adjusted. In this way, users can gain a sense of causality between adjusting a query filter and the corresponding changes presented in both the other query filters and the starfield display.

The characteristics of progressive refinement, continuous reformulation of goals, and visual displays make VIS particularly well suited for searching for trends in the video annotations. We thus propose to adopt this methodology as a framework for addressing our analysis problem. Note, however, that the methodology is limited in that it cannot handle relative queries. That is, while we can use the approach to identify and specify interesting *subsets* of data, we cannot use the approach to specify *relationships* between these subsets. Thus, while the methodology is suitable for analysis, it is not yet sufficient for the type of *relative* analysis desired. In this paper, we thus are extending the methodology to address this problem.

In order to better understand this problem, consider a query such as "show me all annotations where the student is working *while* the teacher is speaking." While we can use dynamic queries (DQs) to select subsets such as the "student working" and "teacher speaking" annotations, we cannot use DQs to specify a desired relationship between members of these subsets. In the case of the example, we need to be able to specify a relative condition that holds true for each (teacher-speaking, student-working) pair returned. Rather than specifying a range of *absolute* values, we need to specify *relative* values. In order to handle such queries, DQs need to be extended to handle variable binding as well as the specification of relationships (i.e., a temporal relationship for the example provided).

Video analysis of video data requires the specification of *relative temporal* and/or *spatial queries*. Extending the use of DQs to handle such relative queries of video data requires binding subsets of the data to variables and specifying temporal, spatial, or motion relationships between these subsets. Two subsets can be specified through the use of two sets of standard query filters. Specifically, we provide a query palette for specifying each subset. Parameters specified in the Subset A query palette automatically bind the subset formed to the variable A. The corresponding functionality is provided for binding the second subset to variable B. For example, we can use query filters in the Subset A palette to bind the subset "teacher talking" to variable A. We can then use query filters in the Subset B query palette to bind variable B to "student working" (see Figure 2). A temporal, spatial, and/or motion relationship *R* between sets *A* and *B* can then be specified with specialized spatial, temporal, or motion query filters (see Section 3 on temporal queries). Progressive refinement of queries is preserved by allowing the adjustment of query filters for *A*, *B*, or *R* at any time and in any order.



Subset A

Media:
- ☒ Text Label
- ☐ Audio
- ☐ Graphic
  - ☐ Line
  - ☐ Rect
  - ☐ Circle
- ☐ Bitmap

Person:
- ☒ Teacher
- ☐ Student

Action:
- ☒ Speaking
- ☐ Working

Temporal Info:

dur > 1 min

0.1 5

occurs = any time

0 20

Subset B

Media:
- ☒ Text Label
- ☐ Audio
- ☐ Graphic
  - ☐ Line
  - ☐ Rect
  - ☐ Circle
- ☐ Bitmap

Person:
- ☐ Teacher
- ☒ Student

Action:
- ☐ Speaking
- ☒ Working

Temporal Info:

dur = any

0.1 5

occurs = any time

0 20

Figure 2. Simple examples of potential subset query filters.
Note that these are only *simple* examples, used to illustrate how subsets can be formed. The actual subset query palettes allow the user to set more parameters and parameter values (e.g., to specify annotation names, categories, additional actions, etc.). The sliders at the bottom of each subset palette allow users to specify *absolute* temporal information for each corresponding subset. This is in contrast to the *relative* temporal information that will be specified using the graphical temporal query language (Section 3).

Notice that the query filters for specifying numerically valued ranges (e.g., the query filter for specifying duration) are double-thumbed sliders. Ahlberg and Shneiderman (1994) introduce this type of filter for specifying ranges of values. Each thumb has an arrow, pointing towards the range being specified. The thumb arrow is filled to indicate that the endpoint of a range is included, or empty to indicate that the endpoint of a range is excluded. Ranges are filled in the sliders for further clarification. The text above the slider indicates exact values. Figure 3 identifies each of these components of the double-thumbed slider.

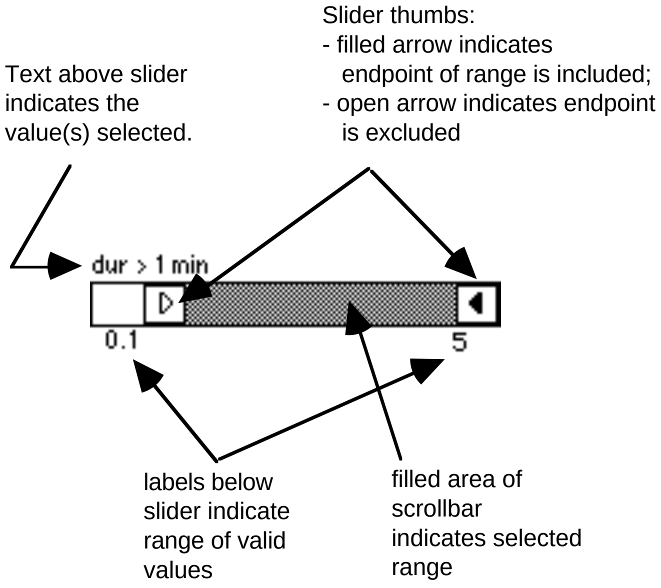

Slider thumbs:
- filled arrow indicates endpoint of range is included;
- open arrow indicates endpoint is excluded
Text above slider indicates the value(s) selected.
dur > 1 min
0.1
5
labels below slider indicate range of valid values
filled area of scrollbar indicates selected range

Figure 3. Description of a *double-thumbed slider* query filter.

In addition to double-thumbed sliders, Ahlberg and Shneiderman introduce a number of other interfaces for different types of query filters. For example, they use a single-thumbed slider (with no arrow) for specifying a single value, or a single-thumbed slider with an arrow to indicate a range towards one of the extremes of the slider. Rather than requiring the user to change the type of slider filter for different types of ranges, we are adopting a single type of slider filter (i.e., the one illustrated in Figure 3) to handle all range query specifications. It is as powerful as the specialized type of sliders, and the uniformity should simplify the user interface.

3 SPECIFYING TEMPORAL RELATIONSHIPS

This section presents a graphical query language for specifying relative temporal queries. The basic language is composed of temporal query filters, a disjunctive operator, and a macro operator. The approach to this work is motivated by the goal to gracefully integrate these temporal filters and operators with the standard DQs and VIS properties introduced in Section 2.

Relative temporal queries are composed of two components—relative duration and relative temporal position. We assume that absolute duration and position are specified with the corresponding subsets A and B. In Figure 1, for example, subset A is set to all “teacher speaking” annotations with duration greater than one minute and which occur at any time in the video, while subset B is set to all “student working” annotations with any duration and which occur at any time. Query filters for *relative* temporal duration and position are presented in Sections 3.1 and 3.2. The disjunctive operator is introduced in Section 3.2.

Temporal diagrams and descriptions used to enhance the clarity of a temporal query are presented in Section 3.3. We consider these aids to be unique augmentations to our temporal query language, increasing the utility of our approach to novice users. User-created macros (i.e., the macro operator) included for improving efficiency of query specification over time are presented in Section 3.4.

3.1 Relative Duration

Two events A1 and B1 can be temporally related in terms of their durations. Although relative duration may sometimes be dictated by relative temporal position (e.g., if event A1 “contains” event B1, then the duration of event A1 is longer than the duration of event B1), this is not the case for all temporal position relationships (see Section 3.2). In addition, users may wish to specify relative duration queries *independent* of relative temporal position. For example, users may wish to locate all events where student S1 works on a task for a longer period of time than student S2. To address this need, we propose a query filter for relative duration.

Given the absolute durations (dur) of events A1 and B1, the basic relationship for relative duration between the events can be described as follows:

$$\text{dur A1 } \theta \text{ dur B1}$$

where $\theta \varepsilon \{ <, >, = \}$. While this relationship can be used to *qualitatively* describe the relative duration between two events, it cannot be used to specify *quantitative* information about that relative duration. For example, if we are interested in cases where event A1 has a longer duration than event B1, then we can specify the constraint dur A1 > dur B1. This does not, however, specify the *amount of time* that A1’s duration is greater than the duration of B1. We can address this problem by redefining the relationship in terms of differences:

$$(\text{dur A1 - dur B1}) \ \theta \ 0$$

where $\theta \varepsilon \{ <, >, = \}$ and 0 denotes zero. If we let Δdur = dur A1 - dur B1, then we can say:

$$\Delta\text{dur } \theta \ 0,$$

where $\theta \varepsilon \{ <, >, = \}$. Now if we are interested in cases where the duration of A1 is one or more minutes longer than the duration of B1 (e.g., student S1 spends at least one minute more on a task than student S2), then we can set Δdur > 0 and more specifically, we can set Δdur $\geq$ 1 minute. This is in contrast to the first representation, where we would only be able to say dur A1 > dur B1.

In addition to allowing us to quantify the relative duration, the Δ representation facilitates specification of values from a continuous range. This allows us to use a slider query filter for specifying relative duration (see Figure 4).

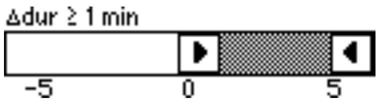
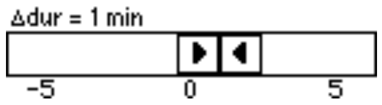
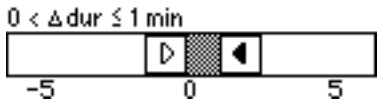
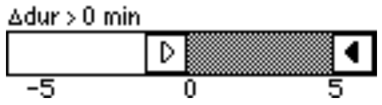


Figure 4. Query filter for specifying relative duration.
Example: Student S1 works one or more minutes longer on tasks than student S2. Given a subset A bound to "student S1 working" and a subset B bound to "student S2 working," we can specify the query "Show all situations where student S1 works one or more minutes longer than student S2" by setting Δdur > 0.

This now makes the relative duration specification compatible with the DQ interface we are designing for video analysis. Another advantage of this type of range selection is that it allows us to specify values at different levels of granularity. To illustrate this, consider the following examples:

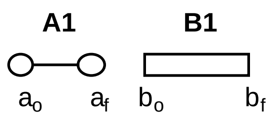
Example: **(Find all (Ai,Bj) pairs such that:)**	**Δ value**	**Query Filter**
Ai is one minute longer than Bj	Δdur = 1 min	
Ai is at most one minute longer than Bj	$0 < \Delta$dur ≤ 1 min	
Ai is longer than Bj	Δdur > 0	

3.2 Relative Temporal Position

Given two events A1 and B1, the relative temporal position between these events refers to the relationship between the temporal starting and ending points of the events. It describes information such as whether A1 starts before B1 or whether A1 and B1 finish at the same time. In order to be complete, a temporal graphical query language needs to be able to specify any primitive temporal relationship (Allen, 1983) as well as any combination of these primitives. This section describes how our temporal graphical query language can be used to accomplish this.

Individual temporal relationships

Allen (1983) describes thirteen primitive temporal relationships between two events: *before, meets, during, starts, finishes, overlaps,* the symmetric counterparts to these six relationships, and the *equals* relationship (see first column of Table 1). These temporal relationships can be described in terms of the relationships between the temporal starting and ending points of each of the events. Consider two events A1 and B1, with starting and ending points a_o,a_f and b_o,b_f, respectively:



There are four pairwise endpoint relationships between these events:

$a_o \ \theta \ b_o$,
$a_o \ \theta \ b_f$,
$a_f \ \theta \ b_o$,
$a_f \ \theta \ b_f$,

where $\theta \ \varepsilon \ \{ <, >, = \}$. We assume that the conditions $a_o < a_f$ and $b_o < b_f$ always hold true.

Similar to relative duration, these relationships can be redefined in terms of differences.

$$\begin{array}{lll}
\Delta_o\ \theta\ 0, & \Delta_o & = a_o - b_o; \\
\Delta_{of}\ \theta\ 0, & \Delta_{of} & = a_o - b_f; \\
\Delta_{fo}\ \theta\ 0, & \Delta_{fo} & = a_f - b_o; \\
\Delta_f\ \theta\ 0, & \Delta_f & = a_f - b_f;
\end{array}$$

where $\theta\ \varepsilon\ \{ <, >, = \}$. Recall that the benefit of describing these relationships in terms of differences is that it allows us to specify quantitative and continuous ranges of values, which is a natural specification to be handled by DQ sliders. All of Allen's thirteen temporal relationships can be uniquely defined by specifying one to three of these endpoint relationships. The last four columns in Table 1 summarize the minimum endpoint relationships required to uniquely specify a corresponding primitive temporal relationship (r_t). Double-lined boxes indicate which relationships are required, whereas the other relationships are automatically inferred.

Given four dynamic query filters for the four endpoint relationships, we can thus specify any *primitive* temporal relationship. Therefore, we incorporate these filters into our temporal graphical query language. Because Δ_o and Δ_f query filters can be used to uniquely define over half of the primitive temporal relationships, they are placed as the top two filters. Figure 5 illustrates what these filters look like.

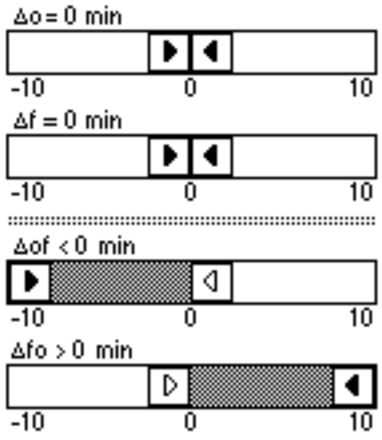
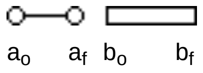
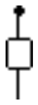
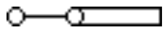
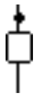
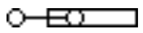
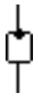
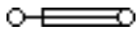
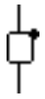
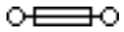
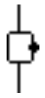
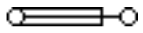
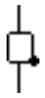
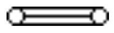
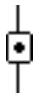
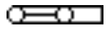
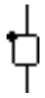
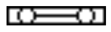
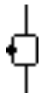
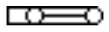
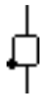
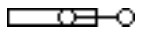
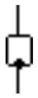
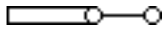
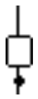
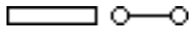
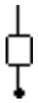


Figure 5. Query filters for specifying relative temporal position queries. Example: Students S1 and S2 start and finish working at the same time. Given a subset A bound to "student S1 working" and a subset B bound to "student S2 working," we can specify the query "Show all situations where students S1 and S2 start and finish working at the same time" by setting $\Delta_o = 0$ and $\Delta_f = 0$.

Note that the thumbs of the filters are always pointing inwards, thereby allowing the user to specify a *continuous* range. The significance of a continuous range is described in more detail below, in the section on single neighborhoods.

Table 1. Summary of Individual Temporal Relationships
[Double-line borders indicate minimum endpoint relationships required.
$\Delta_{of} = a_o - b_f$, $\Delta_o = a_o - b_o$, $\Delta_f = a_f - b_f$, $\Delta_{fo} = a_f - b_o$]

r_t	graphical description	Freksa's icons	Δ_{of}	Δ_o	Δ_f	Δ_{fo}
< before			< 0	< 0	< 0	< 0
m meets			< 0	< 0	< 0	$= 0$
o overlaps			< 0	< 0	< 0	> 0
fi finished by			< 0	< 0	$= 0$	> 0
di contains			< 0	< 0	> 0	> 0
si started by			< 0	$= 0$	> 0	> 0
= equals			< 0	$= 0$	$= 0$	> 0
s starts			< 0	$= 0$	< 0	> 0
d during			< 0	> 0	< 0	> 0
f finishes			< 0	> 0	$= 0$	> 0
oi overlapped by			< 0	> 0	> 0	> 0
mi met by			$= 0$	> 0	> 0	> 0
> after			> 0	> 0	> 0	> 0

As in the case of other query filters (i.e., those used to specify a subset), these four temporal query filters are bound to one another so that users cannot specify invalid queries. That is, when the user changes one query filter, the other three are automatically updated to exclude any invalid ranges, if necessary. In the example in Figure 5, the user needs to set both the Δ_o and Δ_f filters. However, only one filter can be set at a time. Suppose the user first specifies the Δ_o filter (i.e., the user sets $\Delta_o = 0$). As soon as this new value for Δ_o is set, the Δ_{of} and Δ_{fo} filters are automatically updated so that $\Delta_{of} < 0$ and $\Delta_{fo} > 0$. These latter two filters are updated because these are the only valid ranges for them when $\Delta_o = 0$ (see the last four columns of Table 1). Δ_f remains unchanged, because it can have any valid value when $\Delta_o = 0$. After the user sets $\Delta_f = 0$, neither of the other three filters are updated because they are already set to include valid values.

We can determine whether or not to automatically update a query filter by using Table 2 below. This table was derived from the information presented in the last four columns of Table 1. Consider the following examples regarding how to use this table. If the user sets $\Delta_{of} < 0$, then none of the other filters are updated, since *any* value is valid for each one. If the user sets $\Delta_{of} = 0$, however, then the other three filters are automatically updated to only include values > 0. If the user sets a filter to include more than one range of values (e.g., $\Delta_{of} \leq 0$) then the valid values for the other filters are determined by performing a union of each range. Thus, when the user sets $\Delta_{of} \leq 0$, no update is made to Δ_o since the union of any value (as allowed by $\Delta_{of} < 0$) and values > 0 is any value (i.e., (-) union (>0) = (-)). Similarly, no updates are made to Δ_f or Δ_{fo} when $\Delta_{of} \leq 0$.

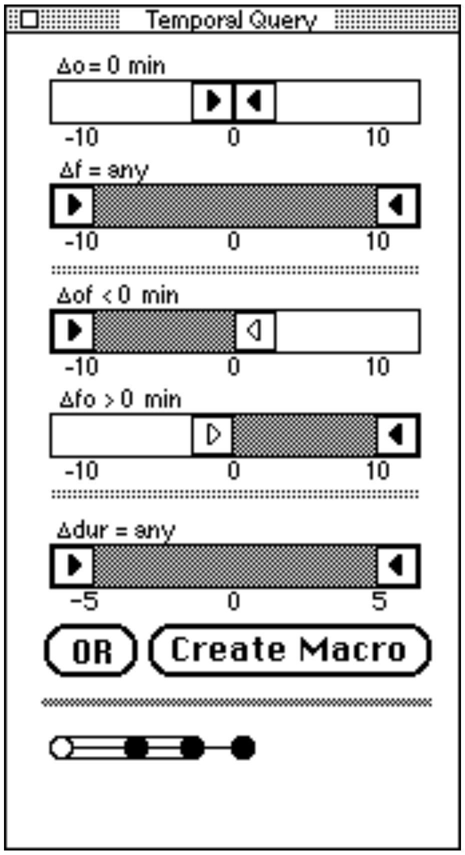
Table 2. Automatic updating of query filters
When a query filter is set as specified in the first column, the other query filters are automatically updated as indicated. [Key: *= current filter being set; - = any value is valid (i.e., no update necessary)]

Query filter and value being set		$\Delta$of	Δo	Δf	Δfo
Δ_{of}	<0	*	-	-	-
	=0	*	>0	>0	>0
	>0	*	>0	>0	>0
Δ_o	<0	<0	*	-	-
	=0	<0	*	-	>0
	>0	-	*	-	>0
Δ_f	<0	<0	-	*	-
	=0	<0	-	*	>0
	>0	-	-	*	>0
Δ_{fo}	<0	<0	<0	<0	*
	=0	<0	<0	<0	*
	>0	-	-	-	*

Neighborhoods of Temporal Relationships

Single neighborhoods

In addition to the primitive temporal relationships, we may also need to specify a combination of these primitives. For example, we may wish to find all events where student S1 starts working at the same time that student S2 starts working (i.e., they start at the same time, but may or may not end at the same time). Because the filters allow us to specify ranges of values for the endpoint relationships, we can use them to specify such a combination of temporal relationships. That is, we can set Δ_o to 0 and Δ_f to "any" in the case of the example (see Figure 6). Note that this one query palette now specifies a disjunctive combination of related primitive temporal relationships, corresponding to the disjunction of several primitive predicates in typically textual query languages (e.g., Snodgrass, 1987). That is, in the example, setting $\Delta_o = 0$ and Δ_f = any is equivalent to requesting events of type A which "start", "equal", or are "started by" events of type B.



Temporal Query

Δo = 0 min (-10, 0, 10)

Δf = any (-10, 0, 10)

Δof < 0 min (-10, 0, 10)

Δfo > 0 min (-10, 0, 10)

Δdur = any (-5, 0, 5)

Figure 6. Using dynamic query filters to specify a temporal neighborhood. Example: Student S1 starts working at the same time that student S2 starts working. Beginning with the query specified in Figure 5 (i.e., "student S1 and S2 start and finish working at the same time") and increasing the range of Δf to include all valid values, we can use the query filters to specify a conceptual neighborhood of (possible) temporal relationships. In this example, the query palette would retrieve all a,b pairs characterized by the *started by*, *equals*, or *starts* relationships. (Note: This figure presents the full temporal query palette. The use of the "OR" button is described at the end of this sub-section while the "Create Macro" button is described in Section 3.3. The temporal diagram at the bottom of the palette visually describes the query specified (see Section 3.3).)

Moreover, because the filters can specify ranges and because they are bound to one another to prevent invalid combinations, we can use the filters to specify single "neighborhoods" of related temporal primitives. Freksa (1992) defines two primitive temporal relationships between two events to be (*conceptual*) *neighbors* if a continuous change (e.g., shortening, lengthening, or moving of the duration of the events) to the events can be used to transform either relation to the other [without passing through an additional primitive temporal relationship]. Thus, the "overlaps" ( ) and "finished by" ( ) relations in Table 1 *are* neighbors, because we can move the ending point of A from the middle of B to the end of B without specifying any additional primitive relationship. On the other hand, the "overlaps" ( ) and "contains" ( ) relations are *not* neighbors. This is because we cannot move the ending point of A past the ending point of B without first passing through the "finished by" relation. A set or combination of temporal relationships between two events then forms a (*conceptual*) *neighborhood* if it consists of relations that are path-connected through conceptual neighbors. Thus, the use of continuous dynamic query filter *sliders* to specify temporal relationships allows us to capture meaningful disjunctions of the temporal primitives.

Multiple- or non-neighborhoods

Although we may prefer to specify single neighborhoods most of the time, we sometimes may also need to specify multiple- and non-neighborhood combinations of temporal relations. A multiple-neighborhood is a combination of two or more neighborhoods that together do not form one neighborhood. A non-neighborhood, on the other hand, combines a single primitive with either another primitive or a neighborhood, in such a way that the final result is *not* a single neighborhood. Consider the example where a user wishes to find all events where student S1 and student S2 start working at the same time, but finish working at different times. This example requires a discontinuous range for the same endpoint relation (i.e., for the Δ_f filter, since we need to set $\Delta_f < 0$ or $\Delta_f > 0$), thus making it impossible to specify with one set of query filters. To alleviate this problem, we incorporate an option to OR any set of temporal relationships together. In this way, the sliders and disjunctive operator form mechanisms which are sufficient to specify any disjunctive combination of temporal (position) relationships between two sets of events. Figure 7 illustrates the use of dynamic query filters and the disjunctive operator for the above example.

Temporal diagrams (described in Section 3.3) at the bottom of the query palette are used to indicate each part of the disjunctive OR query formed. An arrow to the right of the diagrams indicates the current part of the OR query that is being specified or edited. If users wish to edit a previously specified part, they can simply click on its corresponding diagram. Once this diagram is selected, the arrow is moved to indicate that that part of the OR query is being edited. In addition, the dynamic query filters are also automatically updated to match the part of the disjunctive temporal query being edited.

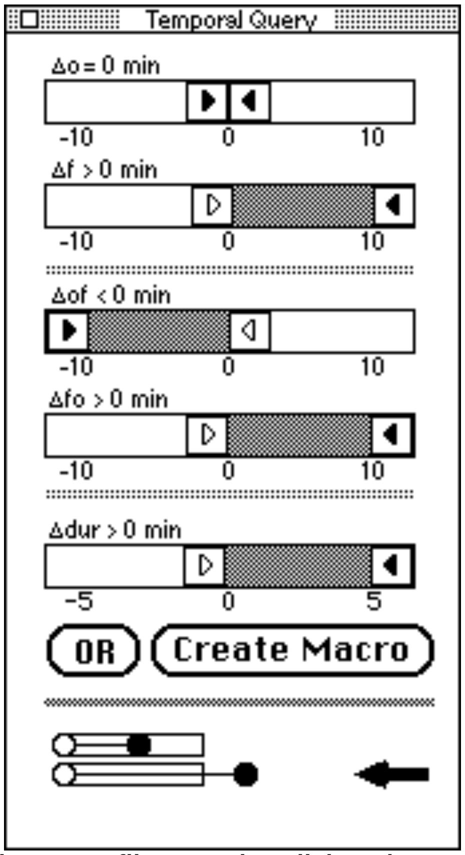


Figure 7. Using dynamic query filters and a disjunctive operator to specify a non-neighborhood.

Example: Students S1 and S2 start working at the same time but finish working at different times. The user specifies this query by forming the following disjunction: "students S1 and S2 start working at the same time but student S1 finishes before student S2" OR "students S1 and S2 start working at the same time but student S1 finishes after student S2." A user can specify the first part of the disjunction by setting $\Delta_o = 0$ and $\Delta_f < 0$, and then clicking on the OR button. Once the OR button has been clicked, the system creates a new temporal diagram (which is initialized as a copy of the previous diagram) at the bottom of the query filter. An arrow points to this newly created temporal diagram to indicate the part of the OR disjunction being specified. A user can then specify the remainder of this query by setting $\Delta_f > 0$. The new diagram is automatically updated to reflect the temporal primitives specified by the query filters (see Section 3.3 on temporal diagrams).

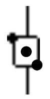
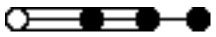
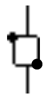
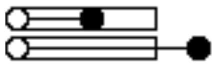
3.3 Clarifying the Temporal Query Specified

Although users can quickly adjust the query filters to specify any of the individual or any combination of the relative temporal position primitives, they may find difficulty in determining whether or not the query specified corresponds to the desired query semantics. Part of this ambiguity may be inherent in the finely-grained definitions of the primitives (e.g., three different primitives meet the specification that events “start” at the same time). In order to reduce ambiguity, we have thus incorporated two techniques for providing quick visual confirmation of the specified query—the use of temporal diagrams and textual descriptions. Each of these is described in more detail below.

Temporal Diagrams.

Note that Figures 6 and 7 include graphical representations of the specified temporal relationships at the bottom of the temporal query palette. In the diagrams, the box refers to B events, while the connected circles are used to describe potential relative positions of A events. The open circles represent possible starting points for set A, while filled circles represent possible ending points. Freksa (1992) uses similar diagrams to describe individual relationships but then uses different icons to describe combinations of temporal relationships (see columns 2 and 3 of Table 1). While the use of Freksa’s icons provides a compact graphical description, it is more difficult to decipher than the diagrams presented in the above figures. In other words, they are not visually intuitive. Part of the difficulty in deciphering Freksa’s icons is that individual endpoint relationships (e.g., Δ_o vs. Δ_f relationships) are obscured. That is, temporal queries model relationships over time, and this continuous temporal dimension is obscured in Freksa’s icons. In addition, Freksa did not intend to develop a query language for users, but rather to demonstrate some theoretical principles of temporal neighborhoods. Table 3 compares the use of Freksa’s icons to the temporal diagrams used in Figures 6 and 7 above.

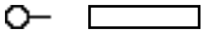
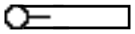
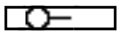
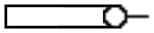
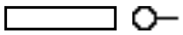
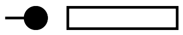
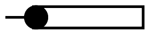
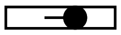
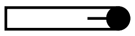
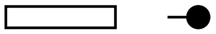
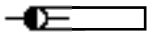
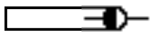
Table 3. Comparison of Freksa’s Icons to Temporal Diagrams

Freksa’s icons	Temporal Diagrams
	
	

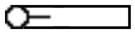
We have developed a transformation function to automatically generate the temporal diagrams from the relative temporal position query filters. The primitive operators of this transformation function are defined using Table 4.

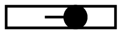
We introduce a half-filled circle (e.g., graphical results of (11) and (12)) to denote the overlap of starting and ending points of A. The transformation function has twelve (vs. thirteen, the number of temporal primitives) components because it evaluates potential *endpoints* specified rather than specific primitives selected. That is, it is used to determine which of the five starting points of A are specified, which of the five ending points of A are specified, and whether there is a case where a starting and ending point of A overlap at either the starting or ending points of B. The function is complete because it tests for each possible endpoint position.

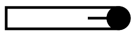
Table 4. Transformation function used to define temporal diagrams

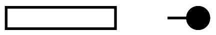
Δ Value(s)	Defines	
$\Delta_o < 0$		(1)
$\Delta_o = 0, \Delta_{fo} \neq 0$		(2)
$\Delta_o > 0, \Delta_{of} < 0$		(3)
$\Delta_{of} = 0, \Delta_f \neq 0$		(4)
$\Delta_{of} > 0$		(5)
$\Delta_{fo} < 0$		(6)
$\Delta_{fo} = 0, \Delta_o \neq 0,$		(7)
$\Delta_{fo} > 0, \Delta_f < 0$		(8)
$\Delta_f = 0, \Delta_{of} \neq 0$		(9)
$\Delta_f > 0$		(10)
$\Delta_o = 0, \Delta_{fo} = 0$		(11)
$\Delta_f = 0, \Delta_{of} = 0$		(12)

In general, the relative temporal position query filters pass Δ information to the transformation function. Each part (i.e., line) of the function evaluates as true or false. The graphical results of each valid part of the transformation function are then collapsed into one temporal diagram and a line is used to connect the endpoints. Consider the example of Figure 6: "student S1 starts working at the same time that student S2 starts working." In this example, we have $\Delta_o = 0$, Δ_f = any, $\Delta_{of} < 0$ and $\Delta_{fo} > 0$. Using the transformation operators depicted in Table 3, we see that the following parts evaluate to true:

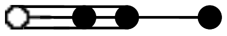
(2) 

(8) 

(9) 

(10) 

This leads to the following composite diagram:



As described in Section 3.2, query filters are linked together so as to permit the specification of only logically feasible situations. Hence, only valid relationships will be passed to the transformation function from the query filters. As a consequence, it is not possible to derive an invalid diagram where the only starting point of A comes after its ending point. In addition, because only continuous ranges are specified by the query filters, all valid starting and ending point combinations of the diagrams are included. A new temporal diagram is derived for each part of a query formed using the disjunctive OR operator.

The temporal diagrams presented in this section represent a valuable aid for confirming the specified query to the users in a visually intuitive manner (e.g., see Figure 7). This should increase the speed and certainty with which users modify or construct complex queries.

Textual Descriptions of Temporal Relations Specified

In order to provide further clarification of the specified temporal query, we also include an option for users to view corresponding textual descriptions. In addition, if the results of the trend analysis are printed in the form of a report, then these textual descriptions could be used as captions to accompany the visual results. Given the specification of Δ's, we can use a simple language to semantically describe the relationships between events A and B. This language is briefly described below.

Exp	**Translation**
$= 0$	at the same time as
< 0	after
> 0	before
≤ 0	at the same time or after
≥ 0	at the same time or before

Δ_o	A starts ____________ B starts
Δ_f	A finishes ___________ B finishes
Δ_{of}	A finishes ____________ the start of B
Δ_{fo}	A starts ___________ the finish of B

Thus, if we have a query filter specifying $\Delta_o < 0$, then our translation tool would generate the following textual description:

A starts after B starts.

3.4 User-Created Macros

Although users can easily specify temporal queries through the temporal query filters, they may have a tendency to use some settings over and over again. For this reason, we provide a mechanism for users to load "meaningful" groups of, or macros for, temporal relationships. While it may be difficult to define and enumerate all "meaningful" groups of relationships, we can identify some characteristics of such groups. For example, we can specify an "all starts" temporal relationship to describe the case where events start at the same time (e.g., Figure 6). In addition, we can provide a button (i.e., operator) to allow users to create and name their own macros for specifying temporal relationships. Users can then load any macro into the temporal query palette and manipulate the query filters starting from the settings specified by the macro.

4 DISCUSSION AND RELATED WORK

Research related to this work can be categorized into four areas: video annotation and analysis, video analysis, temporal queries, and multimedia databases. Work in video annotation and analysis has often focused more on *creating* annotations rather than analyzing them (e.g., Mackay, 1989; Weber and Poon, 1994). There is, however, some work involving storing annotations in an underlying database (Roschelle, et al., 1990). This then would allow for queries on the annotations as supported by the database. Unfortunately, this work focuses on *absolute* queries rather than *relative* ones. That is, users can search for events, but they cannot search for relationships between events without previously identifying and explicitly annotating the relationships themselves. Harrison et al. (1994) do not use a database to aid in the analysis process, but they use timelines for temporal analysis. They graph annotations over time, thereby indicating all of the occurrences of an annotation over time. By displaying all annotations in a parallel timeline, they present absolute as well as relative temporal information. This approach to finding relative temporal trends in the data is limited, however, in that users can only see a part of the timeline at a time. This makes it simple to find *individual* or a few instances of a temporal trend, but difficult to determine whether such a temporal relation holds true over *several* instances (i.e., over all time). In comparison to these approaches, we have presented a tool (in the form of a graphical query language) for directly retrieving absolute and relative temporal information. In addition, our use of dynamic queries allows users to specify queries *incrementally*, thereby giving them a sense of causality between adjusting a query filter and corresponding changes to the presentation of retrieved results.

In computer science, the term video analysis has been frequently interpreted as analysis of video at the *bit* level. Thus, work in this area tends to focus on issues such as scene and shot detection (Hampapur, et al., 1994), object extraction (Chua, et al., 1994), and some motion analysis (Dimitrova and Golshani, 1994). While these types of analyses are different from the type of analysis we propose to support, they are complimentary. In fact, we could use object extraction techniques to automatically create annotations of where people and objects occur in the video. Our analysis system could then be applied to the annotated video as before. In this way, bit-level video analysis can serve as a front-end to our more coarsely-grained analysis of relationships between video events.

Some of the work in temporal queries has focused more on the context of evolutionary databases rather than on databases of temporal information (e.g., Snodgrass, 1987). Evolutionary databases, such as those used to manage medical images (Chu, et al., 1992) are different from databases of temporal information in that they focus on *discrete* changes entered into the databases as an effect of changes made in the real world (e.g., John broke his leg on April 10, 1994). That is, these databases focus on discrete changes to static objects rather than changes in continuous, dynamic media. In addition to this focus on evolutionary context, work in temporal queries has focused on textual rather than graphical query languages. For the type of users we are targeting, an SQL type language would be unacceptable. We are exploring the development of a graphical query language to simplify query specification. We also hypothesize that our graphical query language and the VIS approach we are adopting and adapting will be more efficient for the purpose of searching for temporal trends in the video data. This hypothesis is supported by a study comparing the use of dynamic queries to a forms-based query language (Ahlberg, et al., 1992). In this study, the researchers found that users performed significantly better using dynamic queries over a forms-based approach for three out of five tasks, one of which involved looking for trends in the data.

Research in multimedia databases varies somewhat due to the interpretation of the term *multimedia*. Some researchers consider image databases (i.e., images + text databases) to be multimedia, but such databases do not deal with temporally-based media such as video or audio. On the other hand, databases which handle temporal media tend to focus on semantic or text-based queries as well as on *locating* information rather than *analyzing* it (e.g., Lenat and Guha, 1994; Chakravarthy, 1994). In these types of databases, media are typically images or short clips with textual captions or descriptions. Information is then located by semantic inferencing on these textual descriptions (e.g., Lenat and Guha, 1994). The drawback of this type of approach is that it does not take advantage of the temporal and/or spatial characteristics inherent in the media. While some image databases allow users to search using spatial information, such approaches have either not dealt with temporal media, or have focused on object extraction (e.g., Gevers and Smeulders, 1992). We distinguish our work from previous work in this area by using both temporal and spatial characteristics[1] of the media, as well as by addressing needs for both retrieval and analysis.

5 CONCLUSION

In this paper, we have presented a graphical query language for specifying relative temporal queries, including relative temporal position and duration between video data annotations. The four query filters used for relative temporal position enable users to specify any *primitive* temporal relationship as well as *conceptual neighborhood* combinations of these primitives. For completeness, we have provided support for the specification of disjunctive non- and multiple-neighborhoods through the introduction of a disjunctive operator. We have introduced the notion of temporal diagrams and descriptions to enhance the clarity of the specified query. Lastly, we have supported the reuse of redundant query specifications through user-created macros.

The primary contributions of this work include the graphical temporal query language for specifying *relative* temporal queries and for facilitating *temporal analysis* (i.e., searching for temporal trends in video data), as well as a transformation function for deriving temporal diagrams from endpoint relationships. In addition, we have discussed how our work can be linked to existing work such as object extraction. Finally, our work can be generalized to different media as well as different domains. By analyzing video through analysis of an annotation layer on top of the video, we have designed an approach which can be applied to other dynamic media (e.g., animation). We have also presented an approach which is not limited to the analysis of video *data* but one which can also be used to analyze other genres of video such as movies, sports, etc.

The work presented in this paper is just one component of the design of an integrated environment for video analysis. We are currently implementing our graphical temporal query language in a Windows-based multimedia pc (MPC) environment. We are also finishing work on graphical query languages for specifying relative spatial and

[1]Due to space limitations in this paper, we only describe the temporal aspect of our proposed interface.

relative motion queries. We plan to run usability tests to evaluate the completeness, clarity, and efficiency of the language for identifying temporal trends in the video data. We are in the process of developing a presentation language to allow users to specify views and parameters for displaying retrieved results.

REFERENCES

Allen, J.F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 832-843.

Ahlberg, C., Williamson, C., & Shneiderman, B. (1992). Dynamic Queries for Information Exploration: An Implementation and Evaluation. *CHI'92 Conference Proceedings*, (pp. 619-626). : ACM Press.

Ahlberg, C., & Shneiderman, B. (1994). Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *CHI'94 Conference Proceedings*, (pp. 619-626). : ACM Press.

Chakravarthy, A.S. (1994). Toward Semantic Retrieval of Pictures and Video. *AAAI'94 Workshop on Indexing and Reuse in Multimedia Systems*, 12-18.

Chu, W.W., Ieong, I.T., Taira, R.K., & Breant, C.M. (1992). A Temporal Evolutionary Object-Oriented Data Model and Its Query Language for Medical Image Management. *Proceedings of the 18th VLDB Conference*, (pp. 53-64): Very Large Data Base Endowment.

Chua, T.-S., Lim, S.-K., & Pung, H.-K. (1994). Content-Based Retrieval of Segmented Images. *ACM Multimedia'94 Proceedings*: ACM Press.

Davis, M. (1994). Knowledge Representation for Video. *Proceedings of the Twelfth National Conference on Artificial Intelligence.* (pp. 120-127): AAAI Press.

Dimitrova, N. & Golshani, F. (1994). Rx for Semantic Video Database Retrieval. *ACM Multimedia'94 Proceedings*: ACM Press.

Freksa, C. (1992). Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1992), 199-227.

Gevers, T. and Smeulders, A.W.M. (1992). Indexing of Images by Pictorial Information. Visual Database Systems, II (E. Knuth and L.M. Wegner, Eds.), North Holland: Amsterdam, 93-100.

Hampapur, A., Weymouth, T., & Jain, R. (1994). Digital Video Segmentation. *ACM Multimedia'94 Proceedings*: ACM Press.

Harrison, B.L., Owen, R., & Baecker, R.M. (1994). Timelines: An Interactive System for the Collection of Visualization of Temporal Data. *Proceedings of Graphics Interface '94.* Canadian Information Processing Society.

Lenat, D. & Guha, R.V. (1994). Strongly Semantic Information Retrieval. *AAAI'94 Workshop on Indexing and Reuse in Multimedia Systems*, 58-68.

Mackay, W. E. (1989). EVA: An experimental video annotator for symbolic analysis of video data. *SIGCHI Bulletin*, 21(2), 68-71.

Roschelle, J., Pea, R., & Trigg, R. (1990). VIDEONOTER: A tool for exploratory analysis (Research Rep. No. IRL90-0021). Palo Alto, CA: Institute for Research on Learning.

Snodgrass, R. (1987). The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, *12*(2), 247-298.

Weber, K. & Poon, A. (1994). Marquee: A Tool for Real-Time Video Logging. *CHI'94 Conference Proceedings*, (pp. 58-64). : ACM Press.