# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**UMI**

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600


Université d'Ottawa · University of Ottawa

# Media Processing and Retrieval Model for Multimedia Documents Databases

by

**Naël B. Hirzalla, M.A.Sc.**

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

**Ph.D. in Electrical Engineering**

Ottawa-Carleton Institute of Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
December 1997

To my mother, father and two sisters, Sana and Maha .....

To my wife, daughter and son .....

# Contents

# List of Figures

# List of Tables

# Abstract

*Typically, multimedia applications involve thousands of hours of video, images, audio, text and graphics that need to be stored, retrieved and manipulated in a large multimedia database. There is therefore an important need for novel techniques and systems which provide an efficient retrieval facility of the voluminous information stored in the multimedia database. Such a facility will consist of a query language, a query user interface, a data structure, indexing and finally content searching algorithms. Our interests focus on the first four components of such system.*

*In this thesis, we propose a multimedia query specification language that can be used to fully describe a multimedia segment that needs to be retrieved from a database. We designed an object-oriented database schema that can integrate the logical, temporal and spatial information of a multimedia document. We also proposed an algorithm to index a video by detecting camera breaks and coarse camera operations which will then be used as indices. Moreover, we discussed how to index interactive multimedia documents where undetermined user actions are involved, by proposing a temporal model that has the capability to represent interactive multimedia document scenarios. Indices that represent possible user actions and reactions are then extracted from the model and used to index such documents. Finally, we designed a simple and friendly query user interface on top of the proposed language, in which users can fully describe graphically a multimedia segment.*

# Acknowledgments

I would like to express my appreciation to my thesis supervisor Dr. Ahmed Karmouch for his guidance during my Ph.D. Program. I would also like to thank Dr. Abdel Obaid from U.Q.A.M., Dr. Dorina Petriu, and Dr. Ekow Otoo from Carleton University, and Dr. Luis Orozco-Barbosa from the University of Ottawa for their valuable comments.

My special thanks are due to the university of Ottawa staff in general, and Ms. Lucette Lepage and Ms. Michele Roy at the Department of Electrical Engineering and Ms. Johane Lalonde at the School of Graduate Studies and Research in particular.

I am truly grateful to my parents and sisters for their consistent support without which this work would not have been possible for me.

Finally, millions of thanks and appreciation's go to my wife, Sana, my daughter, Dana, and my son, Adam for their patience and support at all times.

# 1. Introduction

## 1.1 Motivation

It is obvious that the evolution of the digital version of multimedia has facilitated the development of multimedia applications and servers in fields including among many education, medicine and military. The multimedia application boost gave no sufficient time for database developers and designers to enhance the existing traditional facilities or develop new ones that have the sufficient capability to store, manipulate, retrieve and play back a large volume of multimedia data. A typical multimedia application involves thousands of hours of video, images, audio and text leading to an overwhelming need to efficiently store, retrieve and manipulate these data in a large multimedia database. There is

therefore an impending need for novel techniques and systems which provide a systematic and efficient access to this voluminous information. From the database point of view, such a system is referred to as a Multimedia Database Management System. A Multimedia Database Management System must provide a facility to retrieve the voluminous textual as well as non-textual data. This facility includes a query language, a user interface, a model of complex objects such as the spatio-temporal relationships among media components, and an index mechanism for objects like video and user actions in active multimedia documents.

The growing number of multimedia applications requires obviously a query specification language offering detailed views of multimedia document contents. Browsing multimedia documents through visual representations, such as Hypermedia, involves traversing numerous levels of tree-like structures. This task may find the user lost, especially when a detailed view of the contents is required. Therefore, it is more convenient in this case to use a retrieval facility that allows query-based retrieval activities directed towards extracting individual multimedia portions of documents.

The introduction of multimedia has raised questions such as ***"how to describe a piece of a multimedia document to be retrieved?"***. If only text is involved, one can simply type in keywords or just a pattern of characters to be matched in the paragraph to locate. However, the data is not limited to text. Therefore, answers to questions like; how to describe a video clip, audio segment, or image for instance, are needed. Should we use tagging keywords or fill in attributes, like date of creation or names of the objects, to be matched or compared with the corresponding information of each object? Even if we assumed that it is possible to

tag each video clip or image in the database with specific information and keywords, we will have to use the same keywords and format that were used for tagging.

Another question is ***how to organize and structure multimedia objects?*** As mentioned in many occasions so far, multimedia documents differ significantly from traditional documents that are composed of text and possibly geometric graphics. The introduction of continuous media such as audio and video imposes new requirements on document representations. Therefore, questions like how to structure and model multimedia documents that facilitates browsing and searching techniques and how to integrate the document's temporal and layout structures in one schema need answers.

Furthermore, to be able to design a database schema and retrieve information from any database, the information itself should be indexed. Indexing is an important component which has to be considered by any retrieval system. Since video has become the most important element in multimedia, let us then ask ourselves: ***how do we index a video stream?*** To be able to answer this question, we need first to segment the stream into its basic units. Segmenting a video stream is a process of detecting camera breaks (cuts) between consecutive units (shots or scenes). Hence, the information about the cuts such as their positions, types, and lengths are then used in creating handling units and indices for video browsing and visual summary. However, video segmentation is a difficult process when considering various types of camera breaks and operations like zooming and panning. A typical algorithm will result in detecting false cuts or missing true cuts if a static (single value) threshold or an inappropriate one is selected. Therefore, "how to segment a video

stream as accurate as possible without missing any true cut or detecting false cuts?" is the question.

Multimedia applications may include not only passive multimedia objects or documents but also active ones. *Active multimedia objects* implies that there are hypermedia-type choices presented to users during the playback of a multimedia document which allow the user's interaction to "drive" the playback. It is unlikely for a multimedia retrieval system to ignore Active multimedia. Therefore, when including it, the first question that may come to one's mind is: ***how do we describe an active multimedia clip to the system, and what features or indices to consider in an active multimedia document?***

## 1.2 Contribution

Our work aimed to answer all the questions mentioned in section 1. The contribution of this thesis can be summarized as follows:

1. Proposing a multimedia query specification language that describes a multimedia document in terms of its spatial, temporal, and contained media information. A new video data specification is also proposed in which camera operations and cuts information can be described.

2. Designing an object-oriented database schema that has the capability to integrate the logical structure, the temporal structure, and the layout structure information of a multimedia document. The temporal structure can either specify the temporal

relationships between individual portions of the document representing *chapters*, *sections* (referred to by the *logical-temporal structure),* or define the relationships between the media components of the document (referred to by the *media-temporal structure*).

3. Proposing a novel video cut detection algorithm that has the following goals: (i) relaxing the problem of threshold selection (with which the inter-frame difference values are compared), (ii) not missing true cuts, and (iii) avoiding detecting false cuts. The first two goals are achieved by selecting a threshold that is as low as 0% with which inter-frame differences are compared. The third goal is achieved by working on understanding the causes of the inter-frame changes, whether they resulted from camera operations, object movements, or cuts.

4. Proposing a novel temporal model to represent interactive multimedia documents scenarios. It depicts possible user action as well as their corresponding reactions, and has the power to express the relationships between the various user actions in time prior to the time they actually occur. The information corresponding to these possible actions can be extracted from the model and used to index such active multimedia documents.

5. Designing a visual user interface to be developed on top of the proposed query language. It is characterized to be easy and accomplish visual interaction in a user-friendly manner. Using this interface, the user can formulate his/her query by describing the multimedia segment of interest.

## 1.3 Outline

Various attempts were made based on monomedia retrieval facilities where only one media is involved in the query process. Although query languages have been proposed for specific types of documents (e.g., documents having only text and image content [CAK93]), few attempts have been made to address a wide range of multimedia documents. In chapter 2 we will review one of these attempts.

In chapter 3, we propose a multimedia query specification language that can be used to describe the multimedia contents portion to be retrieved from the database. Such a contents portion is termed a multimedia segment (***multisegment*** for short). This language permits retrieval of potentially relevant multisegments from the multimedia database. The user formulates a query describing the content of the segments of interest. Relevant multisegments that contain all aspects of the query are retrieved by analyzing the full content of all documents in the database. The query language provides means by which the user can specify the information on the media as well as the temporal and spatial relationships among these media.

In chapter 4, we focus on designing an object-oriented database schema that is based on the multimedia document structures. A multimedia document contains a logical structure, a temporal structure, and a layout structure. The temporal structure can either specify the temporal relationships between individual portions of the document representing *chapters*, *sections* or *paragraphs,* in terms of SEQUENTIAL and CONCURRENT (the *logical-*

*temporal structure),* or define the relationships between the media components of the document, in terms of STARTS, FOLLOWS, EQUALS, OVERLAPS, and AFTER, (the *media-temporal structure*). The proposed database schema supports all these structures and facilitates document browsing and retrieval techniques.

In chapter 5, we propose an algorithm that detects video cuts and basic camera operations: zoom-in, zoom-out, tilt-up, tilt-down, pan-left, and pan-right. The algorithm has three goals: (i) to relax threshold selection problem, (ii) to detect true cuts, and (iii) not to detect false cuts. The first two goals are achieved by selecting a low threshold value with which inter-frame differences are compared. The third goal is achieved by understanding the various inter-frame changes, whether they resulted from camera operations, object movements, or cuts. The algorithm generates a ***Detection Graph***, that plots the type of inter-frame change in the associate video clip versus the frame number, which provides relevant information about the video. This information is used to index the video segment in order to facilitate later retrieval from the database.

The language introduced in chapter 3, allows the users to delineate interactive multisegments. This is achieved by specifying user actions within the query as *External Information.* In order to index interactive multimedia documents, we propose a temporal model in chapter 6 which provides a new representation for asynchronous and synchronous temporal events. Our emphasis is to represent the user actions and key events. At the end of this chapter, we contrast and compare similar models to ours.

Chapter 7 contains a design of a visual user interface that is easy to understand and accomplishes visual interaction in a user-friendly manner. It can be applied on top of the proposed query language. Using this interface, a user may formulate his/her query to describe the segment of interest. The interface provides means by which the user can specify the information on the media contained in a multimedia segment as well as the temporal and spatial relationships among these media. To follow the decomposition of information contained in a passive multimedia segment that we will propose in chapter 3, the menu driven user interface will be composed of three main windows: *Temporal Specification Window, Spatial Specification Window,* and a set of *media specification windows.* Finally, we draw our conclusions in chapter 8.

# 2. Multimedia Query Languages

## 2.1 Introduction

Conventional Information Retrieval systems focus on keywords based searching techniques which depend on data being tagged with descriptive words. These methods depend widely on the keywords provided automatically by the system or manually by the users for every piece of information. Very few of these methods can retrieve complete or accurate information. Too general a query may yield a lot of items and too specific a query may retrieve no items.

A few query languages have been proposed for multimedia databases. Some of them are extended from the standard SQL language [ROU88]][JOS88], while others are based completely on new approaches such as the one introduced in chapter 3, and [KAU94], and [BIN93]. Many other languages are media specific query languages, i.e. they are defined to specify one medium in isolation, mainly image media type. In chapter 3, we will discuss few examples on existing media-specific query languages such as QBIC, [BAR93]. However, in this chapter, we will discuss the query language introduced by Kau and Tseng from Taiwan, MQL, [KAU94], that is defined to support multimedia databases.

## 2.2 A Query Language for Multimedia Databases: MQL

MQL specifies and manipulates various types of multimedia information. The language is divided into two parts: definition language and manipulation language. The data definition language, DDL, is a C++-like language which achieves extendibility and flexibility by employing an object-oriented framework for multimedia information management. The data manipulation language, DML, is an SQL-like language which retrieves multimedia elements from database in several ways.

The syntax of the MQL-DDL is described as follows:

*Create CLASS <class_name>*
*{SUPER <superclass_name>;*
*[<data_type><attribute_name> (default_value)|(alternative<media_type>)];*
*[<method_name>()]; (<constructor>); }*

The characteristics of the class is included in braces in the above syntax. The second line "SUPER" specifies the superclass if this class has one, otherwise it can be ignored. The third line specifies the attributes belonging to this class. The brackets identify repeatness, and the last line defines the instruction routines which will be invoked when an object is created.

The syntax of the MQL-DML which is of more interest to us is described as follows in the BNF notation:

*<S> ::= SELECT <A> <V> FROM <R> WHERE <C>*
*<A> ::= [<attribute_name>] | **
*<V> ::= ((<previous> | <media_type>) version)*
*<R> ::= <class_name>*
*<C> ::= (<C1> (AND | OR |) <C> | <C1>)*
*<C1> ::= ("("<C>")" | <E>)*
*<E> ::= <L>("=" |"<" | ">" | IN | CONTAINS) <L>*
*<L> ::= (<constant> |<A>|<M>|<S>)*
*<M> ::= "("<A> WHERE <C> ")"*
*<S> ::=<string>*

In summary, this syntax is basically a SELECT statement where <A> is a list of attributes to be retrieved, <V> is the result of version including historical version, <R> is the domain class, and <C> is a condition.

The following is a query example taken from [KAU94]:

*Retrieve the previous image house which costs less than 8,000,000 and has a kindergarten around.*

Its query syntax is as follows:

***SELECT*** *image_version_class_instance* ***PREVIOUS***

***FROM*** *House_class_instance*

***WHERE*** *price_text_instance<8000000*

***AND*** *Evironment_video_instance* ***CONTAINS*** *"kindergarten"*

Kau and Tseng claim that by supplying such a DML, MQL can satisfy the following multimedia database query requirements:

| | |
|---|---|
| *1.* | *Complex object query: completed by retrieving the whole components rooted at the wanted object which is indicated in <A>.* |
| *2.* | *Pattern matching query: Provided by specifying CONTAINS in the syntax. MQL will compare the given condition with the attributes of object to find the possible results. Thus, it can serve the pattern matching on image, voice or text.* |
| *3.* | *Version query: Provided when users specify a version name in <V>.* |
| *4.* | *Nested query: Users can filter results by according to "IN" between two SELECT statements.* |

However, we chose to address the problem of finding an expressive and simple query language for multimedia databases using a different approach. In chapter 3, we will propose a language in which the *condition* on the multisegment to be retrieved is specified in a less abstract way. As will be shown in the next two chapters, it explicitly describes the temporal and the layout structure of the multisegment as well as its content which is based on the media type.

# 3. A Multimedia Query Specification Language

## 3.1 Introduction

In this chapter, we present a multimedia query specification language that could be applied to different types of multimedia documents. This language permits retrieval of potentially relevant segments of multimedia documents (*multisegments*) from a multimedia database. The user may, for instance, want to retrieve a multisegment in which video shots of the 1995 G7 meeting are displayed, the voice of an anchorperson discussing the role of Canada is played out, and a text summarizing the G7 events is also displayed on the screen. To

specify this query, the user must provide selective descriptions of the multisegment using a query language. These descriptions are written provided that the user has previously seen the multisegment.

This chapter is organized as follows: in section 2, we discuss and decompose the information contained in a multisegment. Then we define the Spatio-temporal operators that are used in the query language. Next we review existing independent mono media specifications for audio, image and text, respectively, that we propose to use as part of our multimedia query language. We then introduce a new video specification in which camera operations and video shots are considered. At the end of section 3, we present the complete syntax of our multimedia query specification language.

## 3.2 Multimedia Information Content

In a multisegment retrieval activity, the user formulates a query describing the multisegment of interest and the document(s) (or the entire database) from which the multisegment should be retrieved. The syntax of the query is defined as follows:

*FIND X IN Doc-list*
*WHERE Multisegment-Specification*

It is a FIND-WHERE query where X is a variable referring to the retrieved multisegment(s), Doc-list is a list of documents selected from the database, and Multisegment-Specification is a description of the multisegment(s) of interest written in a multimedia specification

language which will be described later. The multisegment retrieval algorithm is responsible for extracting all the multisegments pertaining to Doc-list that match the specification.

Multisegment-Specification consists of a description of the content of the multisegment to be retrieved. Consider the following user query example using a free-text-like syntax:

> *a multimedia segment including video shots on G7 meeting 1995,*
> *audio explaining the role of Canada,*
> *and there is a text summarizing the G7 events..*

The process of segment retrieval can be made easier by allowing the user to formulate a description that is similar to what he/she has seen previously. For instance, the user can describe the time relationship between two media objects. The layout on the display screen may also be used to specify a segment. For example the user may add to the above example the following:

> *the video shots and the audio were played back simultaneously, and*
> *the text window was displayed at the bottom of the screen*

Hence, the multimedia information perceived by viewers can be decomposed into three different types:

| | |
|---|---|
| o | Temporal information *that specifies the relationships in time between the various media objects.* |
| o | Spatial information *which specifies the locations of the media objects in space.* |
| o | Media information *describing the individual media elements composing the multisegment.* |

In addition, interactive multimedia documents may also include unpredictable time events, such as user interaction, reaching a particular program or document state, and programs with

unpredictable execution time. These behaviors are usually remembered by the viewers and could be used to enrich the description of a multisegment. Therefore for such interactive documents, a new type of information exists, External information, that describes the user actions or program states. It follows that multimedia information will be divided into four separate types: Media information, Temporal information, Spatial information, and External information, (Figure 3.1). However, in this chapter we will only be concerned with non-interactive (passive) multimedia documents that do not encounter any unpredictable time events. Later in chapter 6, we introduce interactive multimedia documents and the associated External information that needs to be considered in our multimedia query language. Hence, only Temporal, Spatial, and Media information are considered in this chapter.

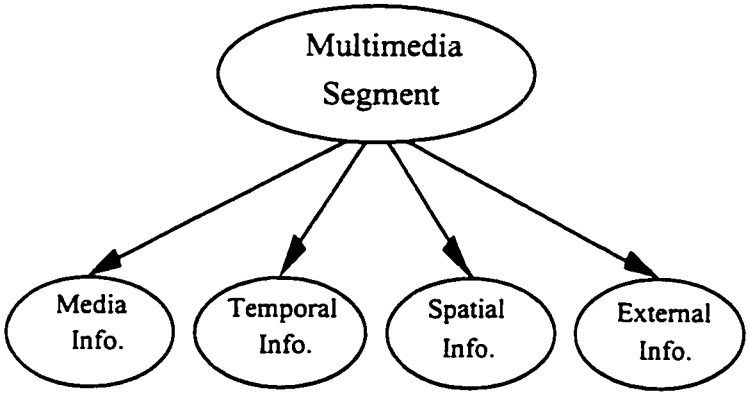



***Figure 3-1 Multisegment Information***

### 3.2.1 Media Information

Media information is the most important information in a document and is the most difficult to outline. It describes the individual Media objects in a multisegment. Different media

types have different characteristics and could be described differently. For instance, an image could be described by the objects it contains and their relative positions to each other. A video clip could be described as an image with the addition of object motions or camera operations. Therefore, the Media information could also be decomposed into:

| | |
|---|---|
| *o* | Media-Content *information that describes the content of the media, such as the energy spectrum of an audio, the words of a text, and the objects in an image or a video frame.* |
| *o* | Media-Spatial *information that provides the relative positions of objects within the media, such as describing a 'person' to be in front of a 'table' in an image or in a video frame, and* |
| *o* | Media-Temporal *information which describes any type of change that happens over time, such as those resulting from camera operations (e.g. zoom-in) in a video clip or the average power change in an audio segment.* |

Obviously, it is not necessary for each media type to have all these information (still images do not have Temporal information for instance). Table 1 shows the various media types and the information types they include.

| | Media-Content Info | Media-Spatial Info | Media-Temporal Info |
|---|---|---|---|
| Text | X | | |
| Graphics | X | X | |
| Moving Graphics | X | X | X |
| Image | X | X | |
| Video | X | X | X |
| Audio | X | | X |

***Table 3-1 Media information***

Therefore, a multimedia query is formulated by specifying the various information on the multisegment(s) of interest, namely the Temporal, Spatial, Media, and External information (Figure 3.2).

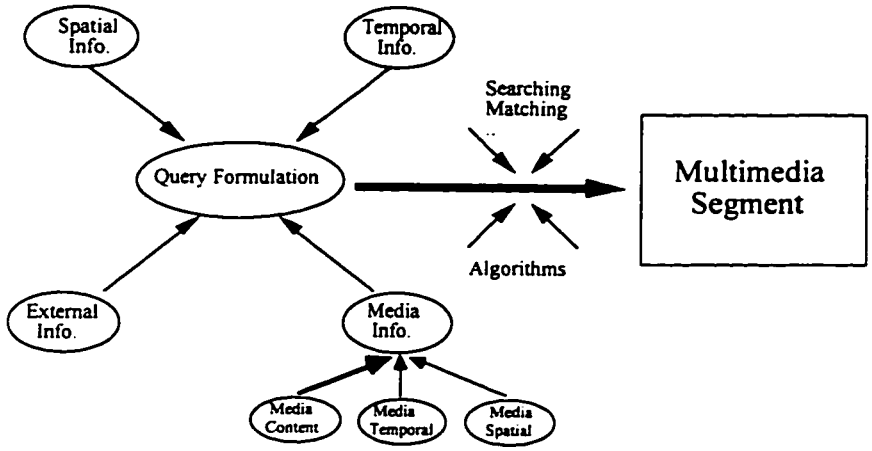



***Figure 3-2 Query formulation***

## 3.3 Multimedia Query Language

In this section, we introduce a multimedia specification language based on the information decomposition discussed above. The syntax skeleton of a query is defined as follows:

***FIND*** *X* ***IN*** *Doc_List*
***WHERE BEGIN***
*[Temporal_Specification]* *and/or*
*[Spatial_Specification]* *and/or*
*[Media_Content_Specifications]* *and/or*
*[External_Specifications]*
***END***

Using this language, multisegment specification is performed by describing the Media Specification, Spatial Specification, Temporal Specification, and/or External Specification.

Spatial and Temporal specifications are responsible for describing the structure of the multisegment using spatio-temporal operators. Media Specification is responsible for describing the media components using specific languages. Finally, External Specification is responsible for describing the user interactions which may contain different information for each user. Again in this chapter, we limit our discussion to the first three information types.

The question raised at this point is the accuracy of the multisegment information perceived and specified back to the system by viewers. In reality, the amount and accuracy of the multisegment specification given by a user depends on the level of his/her attention during rendering and on the way these information are presented (e.g., the presentation rate and the layout). For example, the user may not be aware of the exact positions of various media objects on the screen. Therefore, most likely the information provided by the user would represent a small part of the full multisegment information.

### 3.3.1 Temporal Specification

When we think of documents in a database we usually think of books. A book involves writing a linear storyline which describes events which happen in time. Multimedia presentation is considerably more complex. Unlike a book, which we may call uni-media and linear, multimedia presentation may have media which must occur simultaneously or in some related way; all these relations must be specified by the author in multimedia documents scenarios.

Using the representation of [HAL91, ALL83] temporal composition can be achieved through the reversible seven operators shown in Figure 3.3. These operators, BEFORE, MEETS, OVERLAPS, STARTS, DURING, FINISHES, and EQUALS, determine how two media objects relate in time.

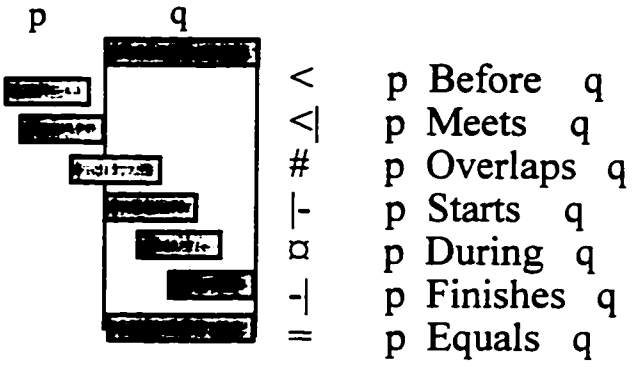



***Figure 3-3 Graphical Representation of Temporal Operators***

| | | |
|---|---|---|
| *Temporal_specification* | | |
| *BEGIN* | | |
| *Video2* | *'-\|* | *Video1;* |
| *Video2* | *'\|-* | *Image1;* |
| *Video2* | # | *Audio1;* |
| *Video2* | # | *Text1;* |
| *Video1* | *'#* | *Image1;* |
| *Video1* | ¤ | *Audio1;* |
| *Video1* | # | *Text1;* |
| *Image1* | # | *Audio1;* |
| *Image1* | # | *Text1;* |
| *Text1* | -\| | *Audio1;* |
| *END* | | |

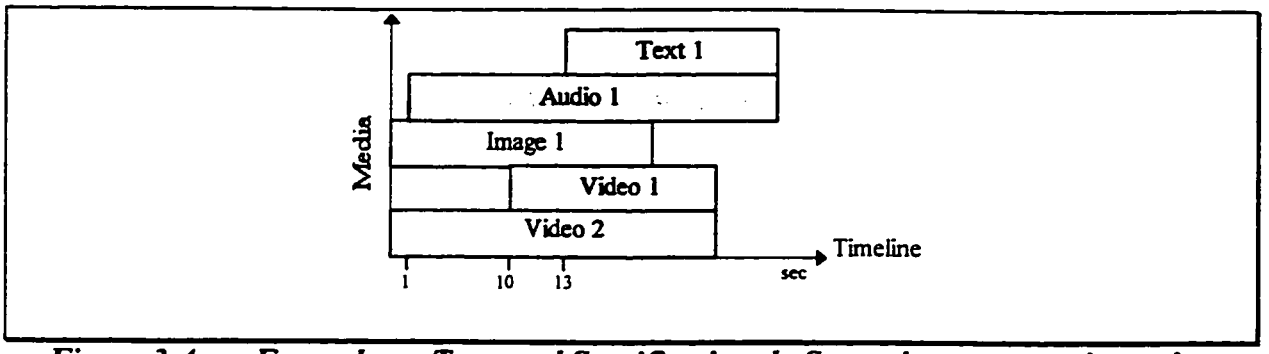



***Figure 3-4 a. Example on Temporal Specification b. Scenario representation using timeline model***

Temporal formula, T, describes the relationships between media time intervals. It is expressed as follows:

$$T := p; \mid p <O_{pt}> q;$$

Where p and q are media objects, and $O_{pt}$ is a temporal operator that relates object p to object q in time. We will refer to the inverse of $O_{pt}$ by $\acute{O}_{pt}$. Therefore, if (p $<O_{pt}>$ q) then (q $<\acute{O}_{pt}>$ p). For example, consider a multisegment scenario shown in Figure 3.4b, the associated Temporal Specification would be as follows:

### 3.3.2 Spatial Specification

Using spatial representations similar to that used by Bimbo et al [BIM93], we introduce the following spatial relationships between two media objects based on their projections on one axis at a time, the x or the y-axis.

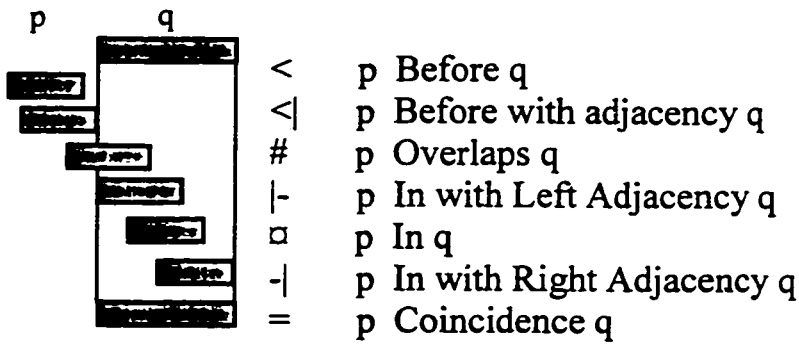



***Figure 3-5 Graphical Representation of Spatial Operators on Axial Projections***

As shown in Figure 3.5, these operators define the relative position between two visual objects. It is clear that the spatial relationships defined by these operators are more or less the same as those used by the Temporal Specification. However, the interpretations are different. A spatial formula considers the intervals originated by the media objects projections on an axis, x or y, is expressed as follows:

$$S := p; \mid p <[O_{px}|?],[O_{py}|?]> q;$$

Where p and q are media objects and $O_{px}$ and $O_{py}$ are spatial operators that relate object p projections on x and y axis, respectively, to object q projections on the same axis. '?' refers to unknown.

*Spatial_specifications*

*BEGIN*

| | |
|---|---|
| *Video2 ( '<, < )* | *Image1;* |
| *Video2 ( '-\|, <\| )* | *Text1;* |
| *Video2 ( '<, '-\| )* | *Video1;* |
| *Video1 (¤, <)* | *Image1;* |
| *Video1 ( <, <\| )* | *Text1;* |
| *Image1 ( <, '#)* | *Text1;* |

*END*

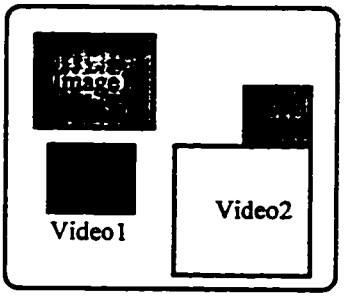



***Figure 3-6 a. Example on Spatial Specification b. Spatial Layout on a display screen***

### 3.3.3 Media Specification

As explained previously, Media Specification describes the media components within a multisegment. It may be decomposed into three types of information: (i) Media-Content information, (ii) Media-Spatial information, and (iii) Media-Temporal information. Not all Media have all three types of information. However, Media-Content Information may include also Media_Spatial information, as we will see later in this section.

To allow Media_Content Specification, we propose to use existing specification languages for text, image, video, and audio content types. A different content specification language is used for each media type. Furthermore, we propose a new Video-Temporal specification that will be discussed later in this section. In the following, we will give a brief overview on the common approaches used for Media_Content Specifications, focusing on Image, Video, and Audio media types.

#### 3.3.3.1 Query on images

There are numerous retrieval systems proposed for multimedia databases that use different query specifications based on images. Image media type has no temporal information. Thus, image query specification is based on the *Image_Content* and the *Image_Spatial* information. Most of the defined query specifications for images are based on one or a combination of the following approaches:

***Text-based query:*** Most of the searching techniques in image databases are based on the use of keywords associated with the images or the video segments, [LUC93]. Each image will have a set of attributes to describe relevant information or specific objects contained in the image. The information may include both Image_Content and Image_Spatial information. Ideally most attributes would be automatically assigned by the system during indexing process; however, current automatic methods to identify objects within an image are not sufficiently robust. Therefore, in almost all systems this is allowed to be assigned manually or semi-automatically. Queries are formulated using either standard query language, such as an extended Structured Query Language (SQL), or a free-text query, from which the system

extract the relevant terms to be used for retrieval. Attributes may be combined in a query using conjunctions (ANDs), disjunction's (ORs), and negations (NOTs). In addition, functions using synonyms, thesaurus support, and logical semantic hierarchies (e.g. Husband IS-A Man IS-A Human-being) can be built to allow the user to navigate within a set of images based on the semantic hierarchy. A simple example on the syntax which can be used for such queries is as follows:

```
Image_specification
        BEGIN
        Date = 'February 1995';
        Title = 'G7 Meeting'
        Keywords = ('France' and 'Canada') or 'USA';
        END
```

These methods are exclusively based on the initial text information associated with each image and do not directly capture the visual or image properties. As a result, there are several problems associated with these methods. (i) The search is dependent solely on the keywords, so if the current query refers to image properties that were not initially described, the search will most likely fail. (ii) Some visual properties are difficult to describe with text such as certain shapes. (iii) Even if all useful characteristics of an image are described by text, there is no common vocabulary for describing image properties or contents, so that a 'curvy' item may not match a 'wavy' one.

***Icon-based query:*** The pictorial objects (named objects such as 'house', 'tree', 'man', etc.) are represented by symbols or icons. These icons are used in a visual interface query

language and are defined through a computer system by the user, thus the name icon-based query.

The image information in these systems is described in terms of image objects (i.e., Image_Content information) and their spatial relationships (i.e., Image_Spatial information). A symbolic description of a scene is a set of formulas expressing mutual relationships between pairs of objects with reference to a coordinate system. Two types of formulas may be considered, [BIM93]. The first deals with the positional relationships between the objects, and the second with the directional relationships between the objects in a scene. The positional formulas take into account the relationships between the minimum enclosed parallelepiped projections of each object over an axis of the reference system (similar to the multisegment Spatial Specification described in section 3.2). An example of a formula is "object A is (completely to the left, just left, overlaps, etc.) object B". The directional formulas, on the other hand, consider relationships between axial planes of the object. Two objects might be collinear, parallel, or not parallel. An example on the syntax used for such queries is as follows, where O(x), O(y), and O(z) refer to the spatial operators being applied between the objects projections on the x, y, and z-axis, respectively.

```
Image_specification
Content_specification
        BEGIN
        man;
        window;
        car or truck;
```

```
        END
Spatial_specification
        BEGIN
        Object_1 [before(x), in (y), overlaps (z), parallel] object_2;
        Object_1 [in_with_adjacency (x), before(y), after(z), not parallel] object_3;
        object_2 [after(x), overlaps(y), coincidence(z), not collinear] object_3;
        END
```

There are some issues that should be taken into consideration for such image representations. (i)The selection of the reference coordinate system, where two distinct approaches may be followed: using object-centered or observer-centered scene descriptions. (ii)The determination of scene descriptions, whether it is automatic or manual. The query in this approach may be formulated using visual interface or a lower level symbolic-based query language.

***Content-based query:*** This is the most challenging approach of all, in terms of indexing, pattern matching or searching, and its access structure. This approach allows images to be retrieved by a variety of image content descriptors (typically combining the Image_Spatial and the Image_Content information) including color, texture, and shape. These properties may describe not only the image as a whole but also the individual objects in it. It also offers virtually unlimited set of queries rather than having the system automatically classifies and organizes samples into a small number of predefined classes.

The properties of color, texture and shape are the basic image content properties which most content based queries are built on. There are various ways to extract these features. One of

the key aspects of content based image retrieval is extraction by shape. Shape similarity has proven to be a difficult issue [MUM87] in the model based vision applications and it remains unsolved in the content based image retrieval applications.

The query interface, for instance, may allow the user to specify color, texture, and/or shape properties visually by example. Systems, such as those described in [BAR93,HIR93], provides facilities to sketch the content of an image to users. The retrieval is then performed on the reduced version of each image, [HIR92], by comparing and selecting images that are relevant to the query.

The following query syntax can be used for such queries:

```
Image specification
        Begin
        feature_i = feature_value;
        sketch matches sketch_object_id;
        End;
```

### 3.3.3.2 Query on Video

A video stream may contain thousands of frames. The main difference between video and image media types is the presence of motion elements in the former. The motion is the source of the Temporal information for the video. Video-Temporal information includes

information on camera breaks (cuts), camera operations, speed of the camera, length of the individual shots (a shot is a sequence of frames with no camera breaks), etc. To extract these information each video stream should be first segmented into shots by locating the cuts. A cut may be sharp or gradual. A sharp cut takes place between one frame and the next; whereas, a gradual cut takes place over a sequence of frames. Gradual cuts, such as fade-in, dissolve, and special effects, complicates the detection process.

In chapter 5, we will propose an algorithm that detects cuts and basic camera operations such as zoom-in, zoom-out, tilt-up, tilt-down, pan-left, and pan-right. The algorithm generates a so-called *detection graph* which provides relevant information about the video. This information is used to index the video segment in order to facilitate later retrieval from the database. Figure 3.7, shows a sample of a *detection graph* where type of inter-frame change is plotted versus frame number. The type of change domain is {Zoom-In (ZI), Zoom-Out (ZO), Pan-Left (PL), Pan-Right (PR), Tilt-Up (TU), Tilt-Down (TD), Cut/Transition (T)}.

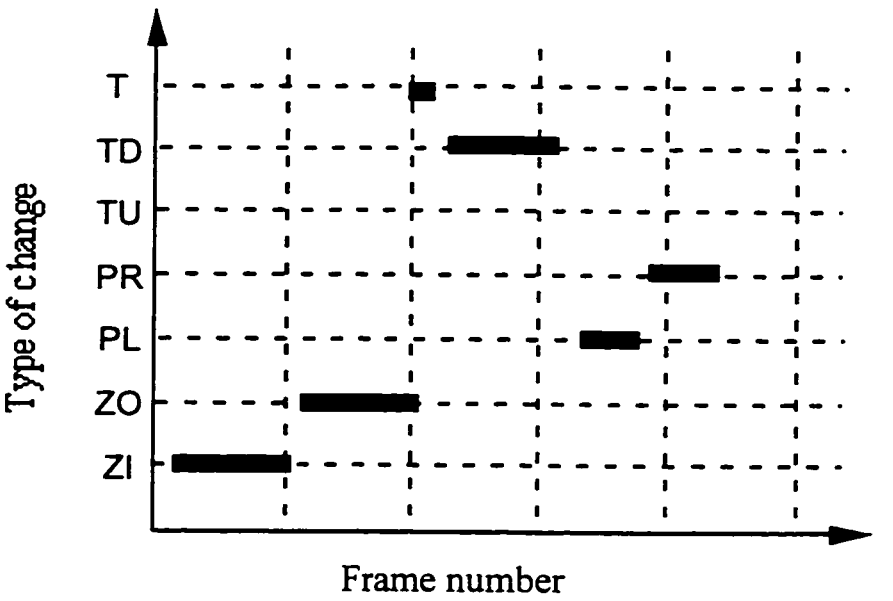



***Figure 3-7 Detection graph***

The syntax of the proposed Video_Temporal Specification is described in section 3.4. The block at the end of this section shows a close example on Video_Temporal Specification associated with the video clip depicted in Figure 3.7.

Each shot in a video should be represented by at least one frame which will be used for indexing. Several techniques to select shot representing frames were proposed that are based on pixel properties, [SMO94]. The Average Frame is one technique where a frame for each shot is calculated by averaging the pixels values of all the frames in the shot at every grid point. The resulting frame is called the Average frame. Then the frame that is most similar to the Average Frame is extracted from the shot as the representative frame.

Another approach involves averaging the histograms of all the frames in a shot and selecting the frame whose histogram is the closest to the average histogram as the representative

frame. Even though, neither of these two approaches involves semantic properties, they are sufficient for many applications.

After selecting at least one representative frame per shot, the query problem on video (i.e., Video_Content Specifications and Video_Spatial Specifications) becomes similar to that of the selected image.

```
Video_temporal_specification
BEGIN
Sequence = (ZOOM_IN, ZOOM_OUT, CUT, TILT_DOWN, PAN_LEFT, PAN_RIGHT);
Operation_1
        BEGIN
        LENGTH > 4;
        START_OFFSET >2;
        END
Operation_3
        BEGIN
        LENGTH =2;
        DISSOLVE;
        END
Shot_2;
        BEGIN
        LENGTH > 100;
        SPEED = 30;
        Avg-Red >100;
        MATCH sketch_object_id;
        END
```

#### 3.3.3.3 Query on Audio

Auditory data can be grouped into categories such as music, human voice, and sounds. Audio provides a very rich information that would be used to understand the content of a video. Kageyama and Takashima, [KAG92], proposed a method by which a user hums a tone and the system retrieves the associated melody from the database. Unfortunately, this technique cannot be applied on other types of data than a melody. Sounds of musical instruments can be specified by the spectrum pattern and the power envelope pattern. These patterns can be automatically extracted and used as indexes during the storage of audio in the database. A user query may be formulated by the use of a microphone, a tape recorder, or a musical instrument. The system then matches this query with the sound-indexes stored in the database and presents the results. The syntax of an audio query is as follows:

```
Audio_specification
        Begin
        Spectrum_index = S;
        Power_index = P;
        Keywords ("Prime Minister" and "G7"and ("Canada" or "Chrétien"))
        Spectrum_change or Power_change
                Offset_position >5;
                Spectrum_index > S;
        END
```

Speech recognition techniques deal with another type of data converting it to text. Such techniques will not be as reliable and as fast as human speech for some time, [ZUE91].

Human aided indexing, on the other hand, which includes most types of auditory data, helps in assigning keywords to each audio segment to undergo string matching during retrieval process. However, problems on keywords assignments similar to those discussed in section 3.3.1 for text-based query on images hold here.

### 3.3.4 BNF Representation of the Query Language

We use BNF (Backus Naur Form) notation, where nonterminal symbols are shown in angled brackets <...>, optional parts are shown in square brackets [...], repetitions are shown in braces {...}, and alternatives are shown in parentheses (...|...|....).

| | | |
|---|---|---|
| **<Query_expression>** | ::= | FIND X IN <Doc_List><br>WHERE <multisegment_specification> |
| **<Doc_List>** | ::= | ALL [EXCLUDE <Doc_List>] \| ( <doc_name> {,<doc_name>}) |
| **<multisegment_specification>** | ::= | [NOT] <single_specification> \| (<complex_specification>) |
| **<complex_specification>** | ::= | <multisegment_specification> <logic_op><br><multisegment_specification> |
| **<single_specification>** | ::= | (TEMPORAL_BEGIN<Temporal_specification>TEMPORAL_END) \|<br>(SPATIAL_BEGIN <Spatial_specification> SPATIAL_END ) \|<br>(MEDIA_BEGIN <Media_specification> MEDIA_END ) \|<br>(EXTERNAL_BEGIN<External_Specification> EXTERNAL_END ) |
| **<Temporal_specification>** | ::= | [{<object_id> <Spatio-Temporal_op> <object_id>} ]<br>[{<object_id> <Temporal_description> }] |
| **<Temporal_description>** | ::= | {<Temporal_att> <numeric_op> <value>} |
| **<Temporal_att>** | ::= | LENGTH \| START_OFFSET |
| **<Spatial_specification>** | ::= | [{<object_id> ({<Spatio_Temporal_op>}$^2$) <object_id>} ]<br>[{<object_id> <Spatial_description> }] |
| **<Spatial_description>** | ::= | {<Spatial_att> <numeric_op> <value>} |
| **<Spatial_att>** | ::= | POSITION \| WIDTH \| LENGTH |
| **<Spatio_Temporal_op>** | ::= | [!] (<) \| (<\|) \| (#) \| (\|-) \| (¤) \| (-\|) \| (=) \| (?) |
| **<Media_specification>** | ::= | [NOT] <Media_single> \| (<Media_omplex>) |

| | | |
|---|---|---|
| **<Media_complex>** | ::= | <Meddia_specification> <logic_op> <Media_specification> |
| **<Media_single>** | ::= | <object_id> |
| | | (<Text_spec> \| <Image_spec> \| <Video_spec> \| <Audio_spec>) |
| **<Text_spec>** | ::= | TEXT <Text_content> END |
| **<Text_content>** | ::= | INCLUDE {<Complex_Keyword> } |
| **<Complex_Keyword>** | := | (<keyword> <logic_op> <Complex_Keyword>) \| <keyword> |
| **<Image_spec>** | ::= | IMAGE [<feature_name> <numeric_op> <value> ] [<Text_content>] |
| | | [ MATCH <sketch_object_id> ] END |
| **<Video_spec>** | ::= | VIDEO [ <Video_Temporal_spec> ] [<Shot_description>] END |
| **<Video_Temporal_spec>** | ::= | SEQ <ordered_camera_op> { [<Operation_description>] |
| | | [<Shot_description>] } |
| **<ordered_camera_op>** | ::= | (<sequence> <logic_op> <ordered_camera_op>) \| <sequence> |
| **<sequence>** | ::= | ANYOPERATION \| (<Camera_operation> {,<sequence>}) |
| **<Camera_operation>** | ::= | [NOT] <single_operation> \| (<complex_operation>) |
| **<complex_operation>** | ::= | <Camera_operation> <logic_op> <Camera_operation> |
| **<single_operation>** | ::= | ZOOM-IN \| ZOOM-OUT \| PAN-LEFT \| PAN-RIGHT \| TILT-UP \| |
| | | TILT-DOWN \| BREAK |
| **<Operation_description>** | ::= | <operation_seq_no> ({<video_att> <numeric_op> <value>} \| <type>) |
| **<type>** | ::= | FADE \| WIPE \| DISSOLVE \| SPECIAL_EFFECT \| OTHERS |
| **<Shot_description>** | ::= | (<shot_seq_no> \| ANYONE ) |
| | | [{ <video_att> <numeric_op> <value> }] |
| | | [<Image_spc> ] [<Text_content>] |
| **<video_att>** | ::= | SPEED \| LENGTH \| START_OFFESET |
| **<Audio_spec>** | ::= | AUDIO [<Audio_content_spec>] [<Audio_Temporal_spec>] END |
| **<Audio_content_spec>** | ::= | [ {<audio_att> <numerical_op> <value>}] [<Text_content>] |
| **<audio_att>** | ::= | SPECTRUM_INDEX \| POWER_INDEX |
| **<Audio_Temporal_spec>** | ::= | {<change_type><change_description>} |
| **<change_type>** | ::= | [NOT] <single_change> \| (<complex_change>) |
| **<complex_operation>** | ::= | <change_type> <logic_op> <change_type> |
| **<single_change>** | ::= | SPECTRUM_CHANGE \| POWER_CHANGE |
| **<change_description>** | ::= | {<Change_att> <numerical_op> <value>} |
| **<Change_att>** | ::= | OFFSET_POSITION \| DURATION \| <audio_att> |
| **<logic_op>** | ::= | (OR \| AND \| XOR ) |
| **<numeric_op>** | ::= | (=) \| ( ) \| (>) \| (<) \| (<=) \| (>=) |

The following is an example on the syntax where CITIZEN-FEB-15 refers to a multimedia document in the database.

```
FIND X IN (CITIZEN-FEB-15)
WHERE       ((TEMPORAL-BEGIN                    * Temporal specification
            Video1 (¤)      Audio1;
            Video1 (#)      Text1;              * implicit AND
            Text1   (-|)    Audio1;
            TEMPORAL-END)   AND
      ((SPATIAL-BEGIN                           * Spatial specification
            Video1 ( < <| )     Text1;
            SPATIAL-END)        AND
      (MEDIA-BEGIN                              * Media specification
            (Text1 TEXT                         * Text specification
                  INCLUDE                       * Text includes keywords
                   ("G7 1995" and ("Canada" and ("Events" or "Agenda")));
                  END)          AND
            ((Video1 VIDEO                      * Video specification
                  SEQ (ZOOM_IN,                 * Video includes sequence
                        ZOOM_OUTCUT,TILT_DOWN,PAN_LEFT, PAN_RIGHT);
                  Operation_1                   * refers to ZOOM-OUT operation
                        LENGTH > 3;             * units are typically in seconds
                        START_OFFSET >2;
                  Operation_3
                        LENGTH =2;
                        DISSOLVE;     * DISSOLVE is a type of a gradual cut
                  Shot_0                * refers to the first shot in the sequence
                        LENGTH > 10;
                        SPEED = 30;   * typically, in frames per second
                        IMAGE         * describing the representative frame
                              Avg-Red >70;
```

```
                                MATCH sketch_object_id;
                                Title = "G7 1995";
                                INCLUDE ("First Day" or "Anthem");
                                END
                        END) AND
        (Audio1 AUDIO
                SPECTRUM_INDEX = S1;
                POWER_INDEX = P1;
                INCLUDE    ("Prime Minister" and ("G7"and ("Canada" or
"Chrétien")))
                SPECTRUM_CHANGE or POWER_CHANGE
                        OFFSET_POSITION >5;        *when the change takes place
                        SPECTRUM_INDEX > S2;
                END ))
        END)))
```

# 4. A Database schema For Multimedia Databases

## 4.1 Introduction

In this chapter, we provide an integrated and homogeneous way to describe, organize, and structure multimedia objects [KAR96] A major challenge in structuring and modeling multimedia documents one should consider is the way the document scenario and layout (the temporal and spatial) information is represented in the model. Another is to have this model facilitate and allow queries including spatial and temporal specification to be formulated and used to find the desired document or segment.

The chapter is organized as follows: In the following section, we discuss the multimedia document structures. In section 3, we present our system architecture. The database schema is described in section 4 where we show how it can support various document structures. Then a brief explanation on the browsing method in section 5. Finally in section 6, we have the summary for this chapter.

## 4.2 Multimedia Documents

Multimedia documents differ significantly from traditional document that are composed of text and possibly geometric graphics. The introduction of continuous media such as audio and video imposes new requirements on document representations. A multimedia document architecture can be defined as an integrated and homogeneous method to describe and structure multimedia objects and to represent their temporal and spatial relationships in a single entity. In the Multimedia Information Research Laboratory (MIRLab) we have designed a document architecture that supports the creation of multimedia documents by means of their logical structures, spatial structures, and rendering scenarios, [KAR96].

The ***logical structure*** is used to describe and organize the information contained in a document. Consider the example of a logical structure for the document entitled "Circuits 101", (Figure 4.1). This document contains four sequential (SEQ) first level document segments or chapters, entitled "Introduction", "Components", "Currents" and "Summary".

"Introduction" has only one media component; whereas, chapter two, "Components", contains two sections that will be rendered in a sequential fashion at the time the document,

or Chapter two, is played back. Section one, entitled "Resistors", contains three media components which will be rendered concurrently at some instant of time (CON). Section 2 contains two paragraphs, etc.

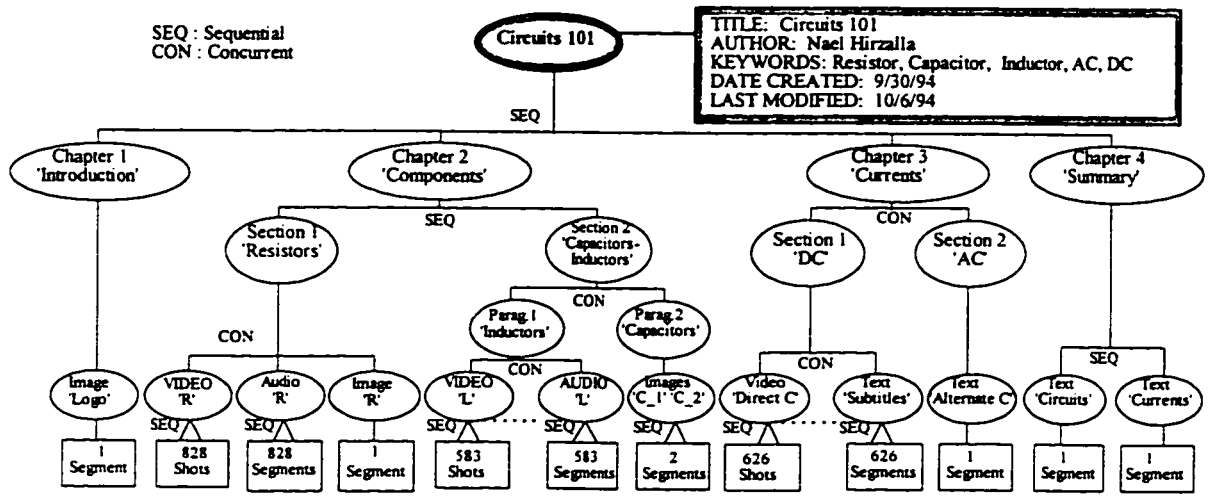



***Figure 4-1 Logical Structure for "Circuits101" document example***

Therefore, a document can be structured into a number of hierarchical levels. Each level will contain a number of segments. At the lowest level, segments will represent media components. In a given level, each group of segments, that belong to the same upper level segment, is either sequentially or concurrently related to one another. The sequential operator (SEQ) specifies that the segments will be rendered in a sequential fashion starting from the left-most segment. The concurrent operator (CON) specifies that at some instant of time the segments should be played back simultaneously.

The ***spatial structure*** of a document describes the spatial properties of the media objects. It specifies the spatial relationships between media objects as well as the physical location of each media on the display screen in terms of the horizontal-vertical coordinates and the

width-height measurements. The spatial structure identifies also the layer number (for each media) which defines the visibility priority among the overlapping media objects during the playback.

Finally, the ***temporal structure*** or the rendering scenario defines the temporal synchronization that coordinates the real-time presentation of a multimedia document. It also maintains the time-ordered relations among the media components. Scenarios are essential for multimedia documents.

## 4.3 System Architecture

The system architecture is described in Figure 4.2. It provides a step-wise document query process to enable efficient retrieval of multimedia documents from the database. Starting from the top, we have a *Query Interface,* [HIR95,CHE96], that allows the user to graphically describe the segment he/she wants to retrieve The *Query Interpreter*, besides translating the tabular or graphical representation of a query to its syntactical form, may also perform some feature extractions from the media objects specified in the query. For example, the Interpreter may extract the color or outline features from a sketch that the user drew to specify the image in the segment to be retrieved.

In order to improve the retrieval process, the query will be hierarchically segmented into query components. The query language structure (section 3.3), allows the Query Decomposer to segment the query into four components, Document Query, Temporal Query, Spatial Query, and Media Query component, when possible. This allows filtering the

database one step at a time satisfying first the most general specification of the query down to the more specific one. The document query component is extracted from the query specification using the Knowledge Model; whereas, the other components are extracted straightforward from the query syntax (see section 3.3). The Document Knowledge Model provides information on the document of interest that may be extracted from the query itself, such as from the Media Query component. This information may include the category the document belongs to, or appropriate keywords on the subject of the document. This information is then passed back to the Query Decomposer to form the Document Query component.

The Document Query module filters the database into a smaller list of documents that satisfy the document specification provided by the Query Decomposer, such as the document title, document subject, document category or other general attributes about the document from which the segment will be retrieved. The document list is then passed to the next module to be further filtered into a list of concurrent segments.

The Temporal Query module contains the temporal specification which defines part or all of the temporal relationships among the media objects contained in the desired segment. The Query module tries to match the given temporal specification by searching through the temporal data model (section 4.4.2) of the database schema. This search is conducted on the filtered document list and will provide a smaller list of concurrent segments (in terms of number of objects it contains). The resulting list will then be referenced by the Spatial Query module and passed to the Media Query module.

The Spatial Query module contains the spatial specification part of the query, which describes the physical relationships among the media objects in the segment. This module considers only the concurrent segments list provided by the temporal component, since spatial relationships exist only between media objects that are concurrently rendered at some instance of time. Therefore this module deals with the spatial data supported by the schema for the segments that are in the concurrent list.

The Media Query module is the last in the hierarchy. It performs matching among the corresponding media instances provided by both the spatial and the temporal components. It may also consider all the media instances that belong to the filtered document list provided by the Document Query Component. Pointers to the media objects that satisfy the media component query and the segments they belong to are then passed to the Display Module.

The Display Module provides the query interface with the resulted objects representatives. It also collects the temporal and spatial information needed to display the corresponding media segments, document segments, or the entire document. A document segment is a logical component of the document defined in the logical model of the schema by the author and stored along the document in the database. This component represents for instance a section or a paragraph of the document (as discussed in the previous section).

Finally, the proposed database schema (discussed fully later) is designed using the object oriented paradigm. It can support the document structures: *Temporal structure, Spatial structure and Logical structure*. Each of the query modules will then communicate with the data module and search for the desired objects. For instance, *the Document query module*

will search for the desired documents list through the *Document Object Identification* data module, and the *Temporal query module* will communicate with the *Temporal Model* searching for the matching list of concurrent segments, etc.

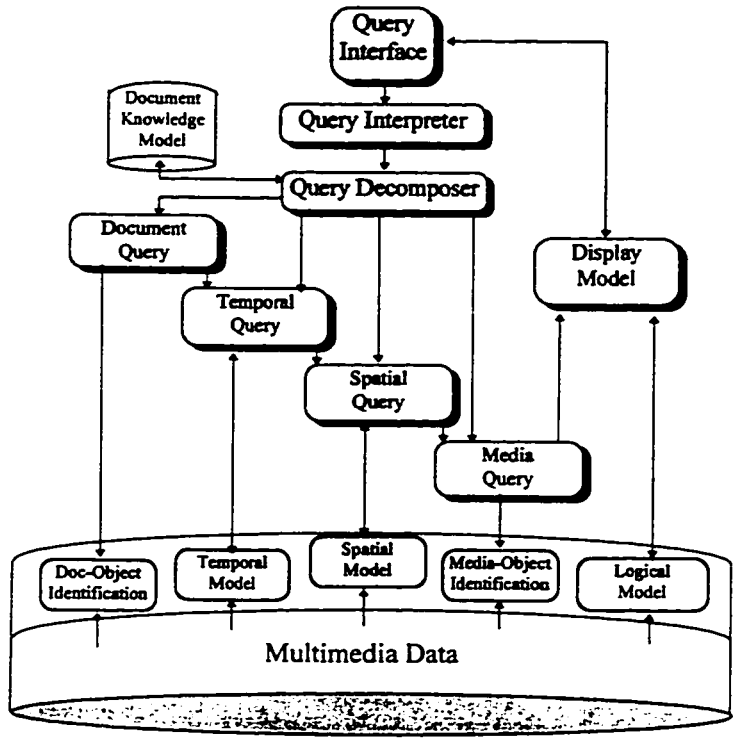



***Figure 4-2 System Architecture***

## 4.4 Database Schema

In this section we present the database schema which is based on the object-oriented paradigm. We start first by modeling the logical structure of the document, then we explain how we modeled the temporal structure and finally the spatial structure. In section 4.4.4, we also show how all these sub models can be integrated into one global database schema.

### 4.4.1 Logical Model

Using the modified OMT notation, the schema provides aggregation and generalization. An instantiation relationship exists between a class and all its derived objects which are referred to by instances of that class. Other relationships between objects (whether classes or instances) and attributes and between objects and methods are expressed by "HAS-ATTRIBUTE" and "HAS-METHOD" respectively. Furthermore, an object may refer to another object through a reference pointer.

As shown at the top left of Figure 4.3, a class is represented by a rectangular box. Class A is said to be ***abstract*** if *A* is in italics in the box, and **concrete** otherwise. An instance of class *A*, may contain one (single head arrow) or many (double heads arrow) instances of another class (**aggregation**). It may also refer to one or many instances of another class, through the **object reference** arrow. Based on the class and **generalization** (represented by a triangle) definition, all attributes, methods **implementations**, and data structures defined in a class are commonly owned by all instances of that class and inherited to its subclasses.

Figure 4.3, shows how the logical structure is modeled. A multimedia document (a *MainDocument* instance) is composed of one or more segment instances of either *ConDocument* or *SeqDocument* classes. *ConDocument* instances, of the same parent, are said to be rendered simultaneously at some instance of time; whereas, *SeqDocument* instances will be rendered in a sequential fashion. At the very lower level, each segment will contain a *GroupOfMedia* instance pointing to one or more Media component.

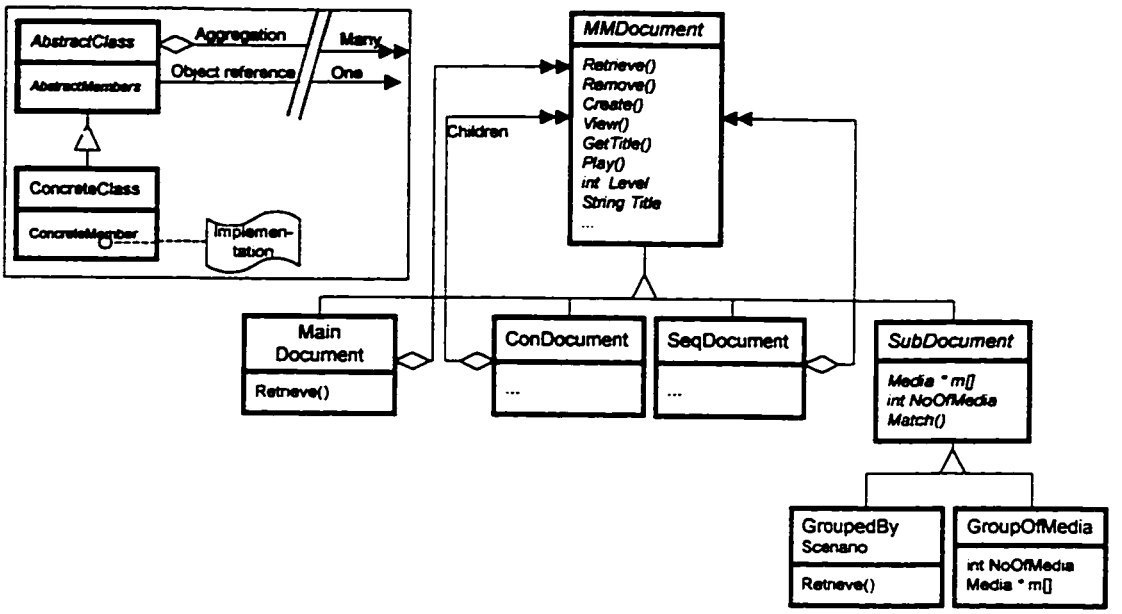



***Figure 4-3 Logical Model***

If for example the document (or a segment) is not logically structured (i.e. mixing CON and SEQ relationships together) then the document or the segment will be composed of media components that are related in time in some way specified by a scenario (SC). Thus, the *MainDocument*, *SegDocument*, or *ConDocument* instance will be composed of an instance of type *GroupedByTemporal* instead, that will group the media components with respect to their temporal relations. Figure 4.4 shows the entire "Circuits101" document without the logical structure. In this case, the *MainDocument* instance, "Circuits101", will not have neither *SegDocument* nor *ConDocument* instances, instead it will contain an instance of type *GroupedByTemporal.* An instance of type *GroupedByTemporal* will be used to model the scenario of the document and can exist in conjunction with instances of type *SegDocument*, or *ConDocument* within the document to model the same or different segments as will be discussed in the next section.

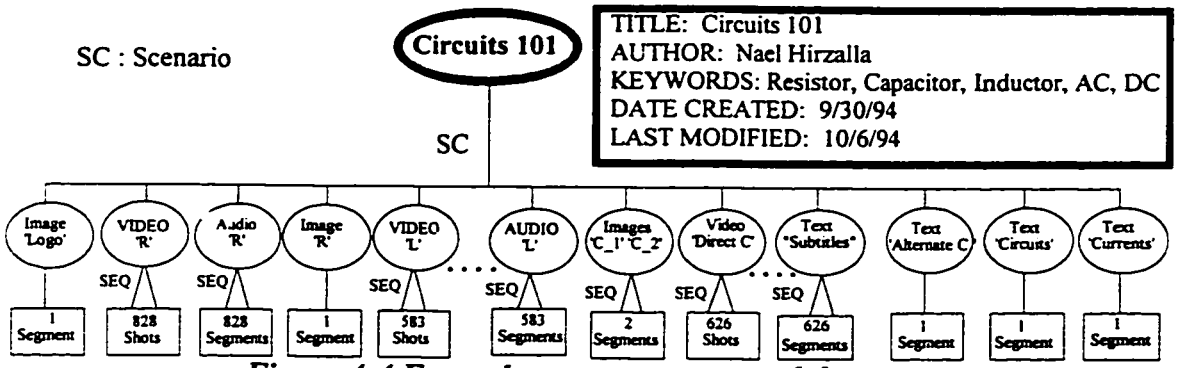



***Figure 4-4 Example on an unstructured document***

### 4.4.2 Temporal Model

To understand the *GroupedByTemporal* class, which models the temporal structure, consider the scenario depicted in Figure 4.5 using the time-line model [GIB91], which represents also the scenario of section two of "Circuits101" document (Figure 4.1). This section of the document contains: three images, a video, an audio, and a text objects. The scenario will be decomposed into a number of phases. The first phase, *phase*1, begins and ends with the longest media among those that begin first *in the scenario*. The second phase begins and ends with the longest media among those that begin first *after the end of phase1*, and so on. All media that starts *within* (after the begin and before the end of) a given phase will form a *sub-scenario* and will be further decomposed into sub-phases of that phase. That is, *sub-phase1* (of *phase1*) begins and ends with the longest media among those that start first *in the sub-scenario.*

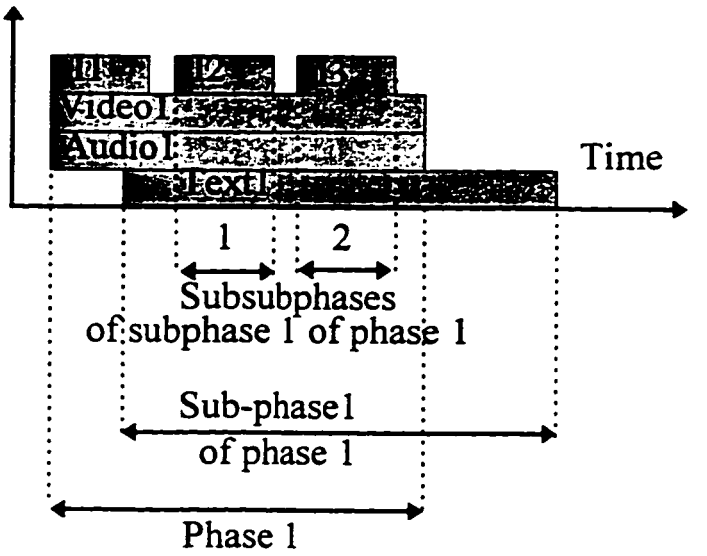



***Figure 4-5 Section 2 scenario***

In the example, *phase1* begins at the same time as *Audio1*, *Video1*, and *I1*, and ends when *Audio1* or *Video1* ends. *I2*, *I3*, and *Text1* form a sub-scenario of *phase1*, since their start times are within *phase1*. *Sub-phase1* begins and ends at the same time *Text1* does, since it begins before *I2* and *I3*. *I2* and *I3* then form a sub-scenario of *sub-phase1*, and so on so forth.

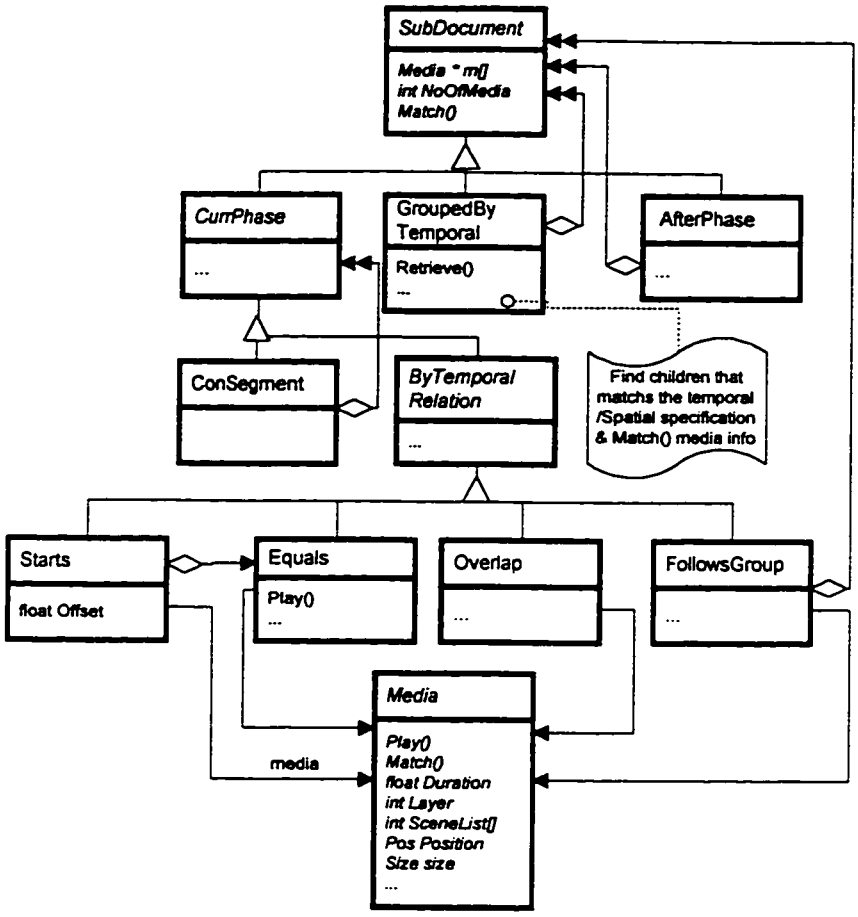



***Figure 4-6 Temporal Model***

A *GroupedByTemporal* instance composed of a *ConSegment* (a subclass of *CurrPhase*) instance to refer to the first phase, *phase 1*, and an *AfterPhase* instance to refer to phase 2, (if any). The *ConSegment* instance is composed of (i) an instance of type *FollowsGroup* that refers to sub-phase1 of phase1, (ii) an instance of type *Overlap* that refers to those media objects that have started before phase 1 and end within it, and (iii) an instance of type *Starts* that has an *Offset* value equals to the start time of *phase1* (or the offset from the end of the previous phase) and points to each media that starts with phase1 but do not have equal duration. The Starts instance is also composed of a number of instances of type Equals that points to those media that start with phase1 and have the same duration. The FollowsGroup

and *AfterPhase* instances may also be composed of a *ConSegment* and *AfterPhase* instances each as well, and so on so forth.

### 4.4.3 Spatial Model

The *GroupedBySpatial* abstract class represents the spatial models of the document. Since the Spatial structure defined specifies the spatial relationships between the media objects in terms of the horizontal-vertical coordinates, two concrete subclasses will emerge *GroupedBySpatial-X* and *GroupedBySpatial-Y* (or *BySpatial-Y* and *BySpatial-X* for short) classes. These two classes will then model the spatial relationships among the media objects with respect to the horizontal and vertical axis, receptively. Each is more or less described and modeled in the same way as the *GroupedByTemporal* class. The classes are called differently, though, to distinguish them from those of the temporal model, (Figure 4.6).

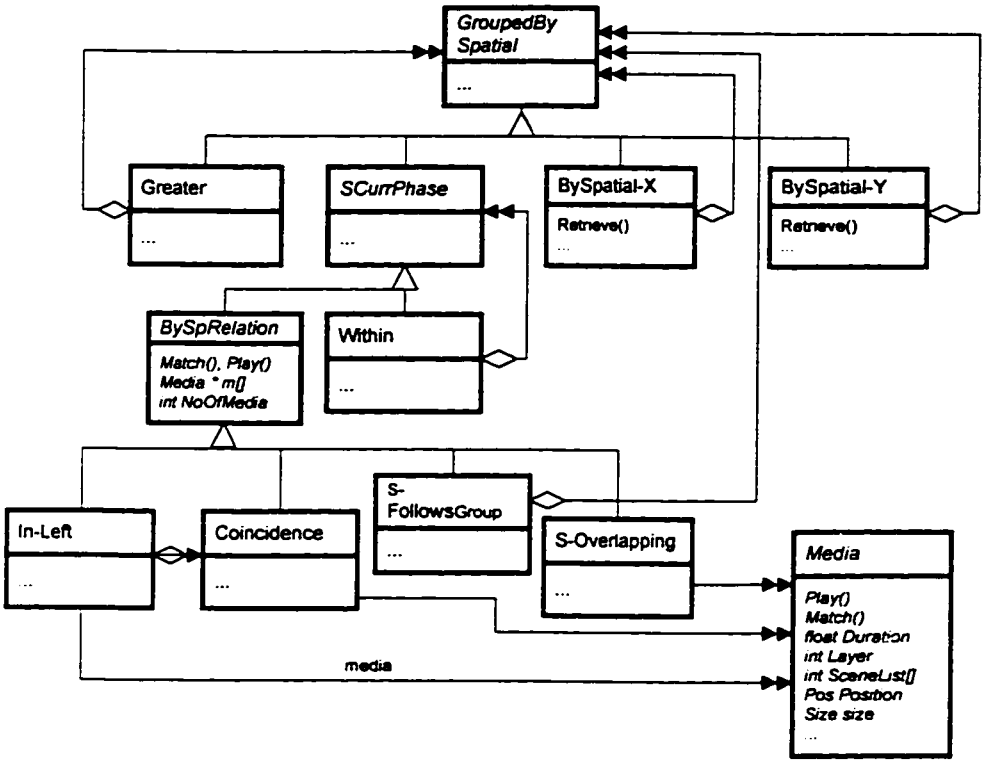



***Figure 4-7 Spatial Model***

Modeling the spatial structure of the document is somehow more difficult than that of the temporal specification. Let us first define a *scene* to represent the presentation layout of the document on a display screen at some instant of time. The scene changes only when another physical media object starts being rendered. Unlike the temporal structure, the spatial structure of a multimedia document consists of many of such scenes. For modeling the spatial structure of a document, we have two options:

1. To model each individual scene

2. Project all the scenes into one global scene and model only that scene

The first option results in a lot of redundancy information being modeled which results in more instances being created and thus longer searching process. The second option results in unreal global scene or media relationships that do not exist between media objects that are never rendered concurrently. Moreover, the global scene will include all the media objects contained in the document resulting in a complex and condensed spatial structure.

Our data model considers an approach that is between the previous two options. To model the spatial structure, we first segment the temporal scenario into phases as described previously in modeling the temporal structure, and choose a *phase* to represent the basic unit (called also segment). Next, the global scene per each phase is found by projecting all the different scenes contained in each phase. Then we model the Spatial-x structure and the Spatial-y structure of this global scene exactly as we modeled the temporal structure for the same phase. However, to distinguish between real spatial relationships and unreal ones during the retrieval process, each media object contains a list of the individual scene numbers which the media object exists in. Real spatial relationships exist only between media objects that have at least one common scene.

Consider for example the same scenario in Figure 4.5, which consists of only one phase. By looking at the scenario, we can extract four different scenes, scene 1 to scene 4 as shown in Figure 4.8. If we assumed that the layout of scene 1 to scene 4 are shown in Figure 4.9a, then the global scene which represents the summation (projection) of all the four scenes is shown if Figure 4.9b. Note that I1, I2, and I3 media objects will not have a common scene. In other words, the intersection of their scene list is an empty set. From the global scene we

then form an x-projection and y-projection spatial relationships among the media objects. Finally, as described previously, we model the spatial structure by dealing with the media projections like temporal scenarios with a slight difference in relationships interpretations.

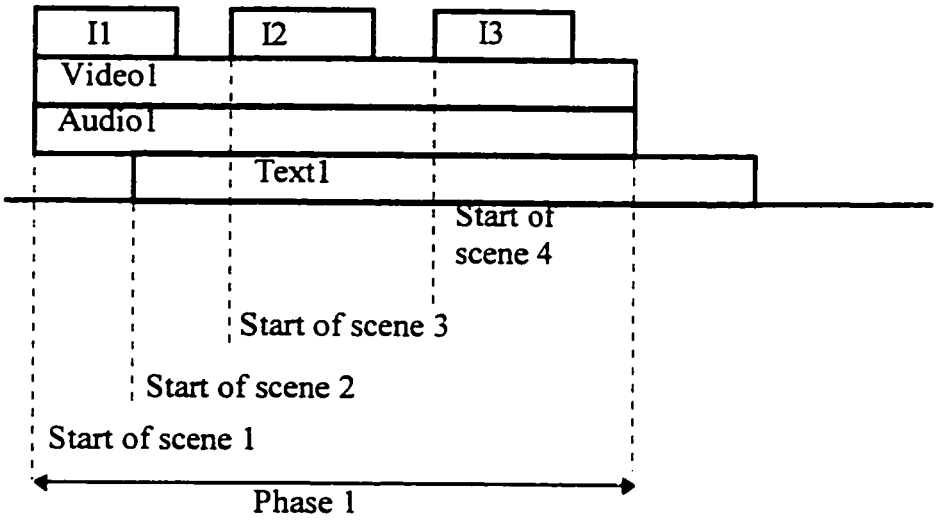



***Figure 4-8 Segmenting the scenario into scenes***

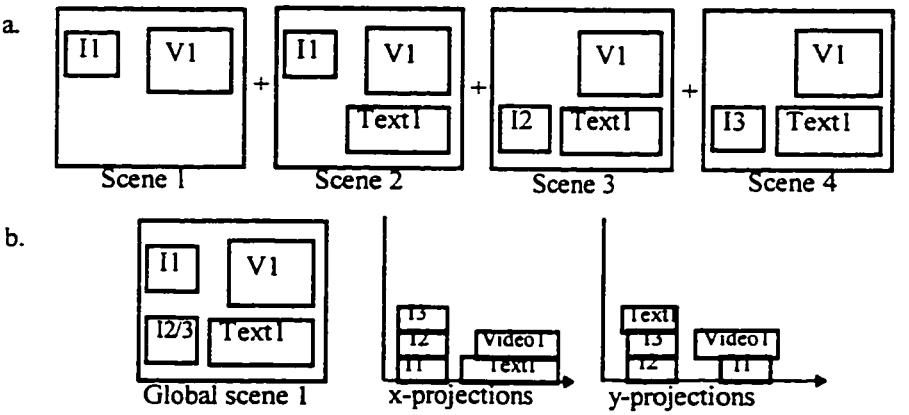



***Figure 4-9 Global Scene***

### 4.4.4 Temporal and Spatial Relationships

With respect to the temporal or spatial relationships between media objects, unlike the proposed query language which allows the temporal description to be formulated using the

seven reversible operators defined originally by Allen, [ALL93], (described in Figure 4.10), the introduced database schema supports only five of them. Figure 4.10 shows the temporal relationships from three different points of view, Allen's, ours, and a more coarse perception. We believe that a viewer of a multimedia document can perceive the temporal relationship between two media objects, A and B, with more precision than just concurrent and sequential, to refer to whether or not A and B are rendered concurrently at some point in time. However, still he/she may not remember for instance whether a media object starts just after the end of another one (MEET) or after a short period of time (AFTER). So supporting our five relationships (Figure 4.10) simplifies the model and at the same time facilitates the retrieval process. In the same manner, the spatial relationships are modeled using only the corresponding five relationships, Figure 4.11.

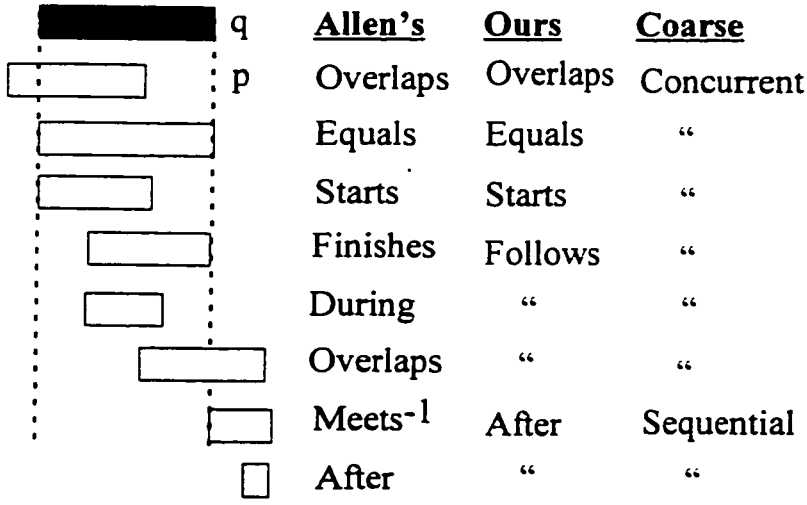



***Figure 4-10 Temporal relationships***

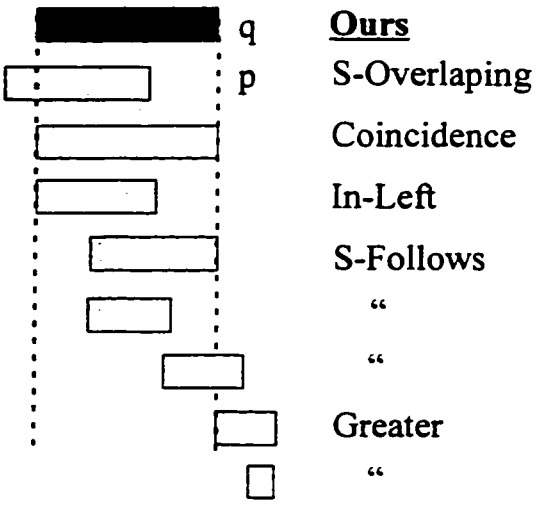



***Figure 4-11 Spatial relationships***

It is worth also to note that due to the way the temporal or the spatial structures are modeled within our proposed model the reverse of Allen's OVERLAPS relationship is different using our set of relationships, Figure 4.10. To explain this, consider for example a media object B that starts after the start time of object A, and lasts after A's end time. We say A Overlaps B. Furthermore, since object B follows the starting of object A, we also say B Follows A, as well. However, we found that it is useful to distinguish the OVERLAPS when reversing the FOLLOWS relationship. Therefore our model, Figure 4.6, also refers to the media object A through the OVERLAPS relationship, as well.

### 4.4.5 Global model: The big picture

Since Spatial relationships exist only between media that are rendered simultaneously or concurrently, i.e. belong to the same phase, both classes *BySpatial-Y* and *BySpatial-X* in the spatial model will be subclasses of the abstract class *CurrPhase* and will be contained in an instance of type *ConSegment*. Moreover, since an instance of type *GroupedByTemporal* will

contain references to media that are grouped by their temporal relationships as well as their spatial relationships, we will be calling this class by *GroupedByTemporal/Spatial*.

Figure 4.12, shows an integrated database schema that supports the logical, temporal and spatial structures of a multimedia document. The class *Category* will be explained in the next section.

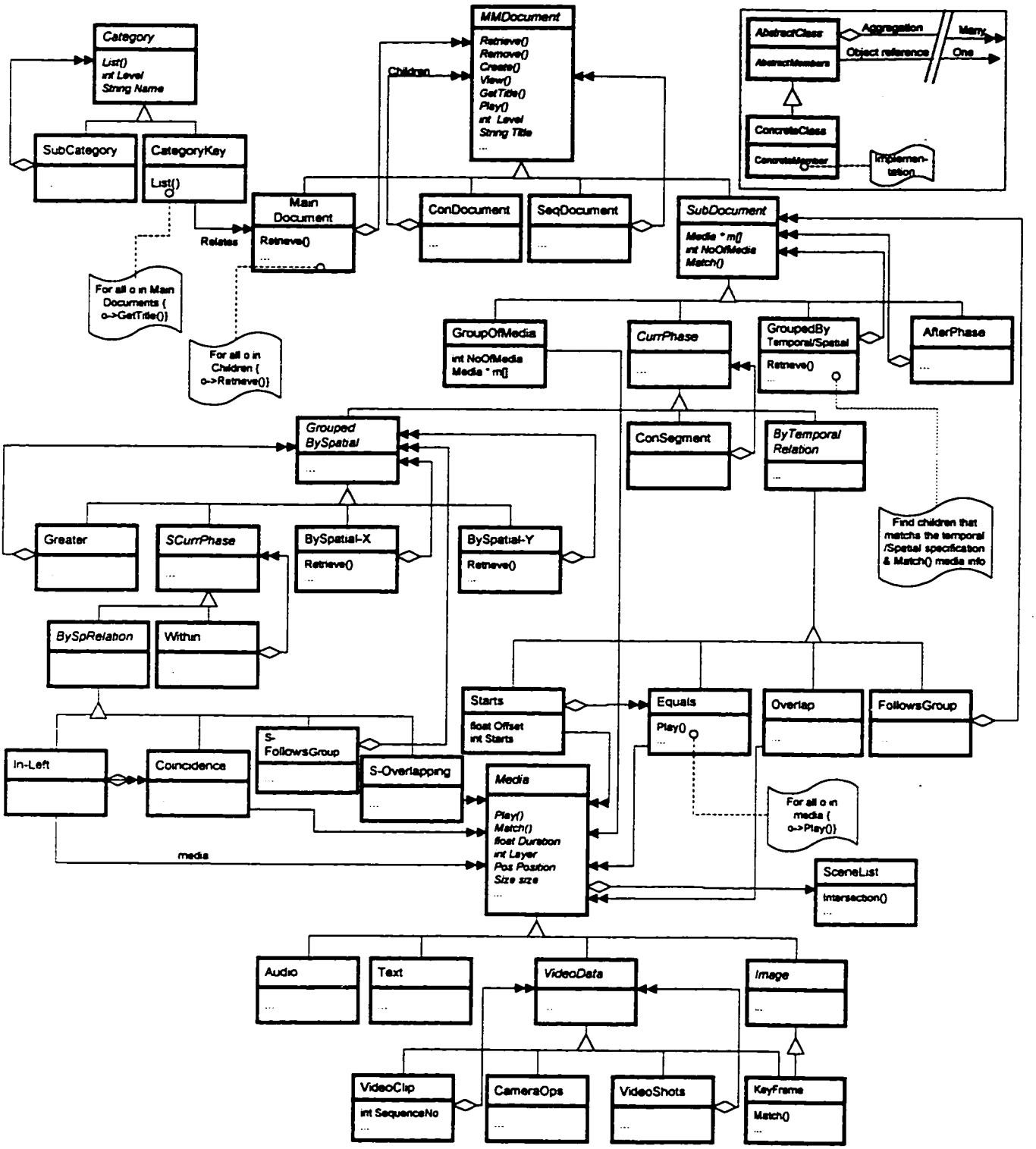



*Figure 4-12 Database Schema.*

Figure 4.13, shows a portion of the "Circuits101" example using the object-oriented approach. The document has a logical structure that differs from the one shown in Figure 4.1 in that "Section 2" is not logically structured and contains directly the media components that are related together by the scenario shown in Figure 4.5, and the spatial structure shown in Figure 4.9.

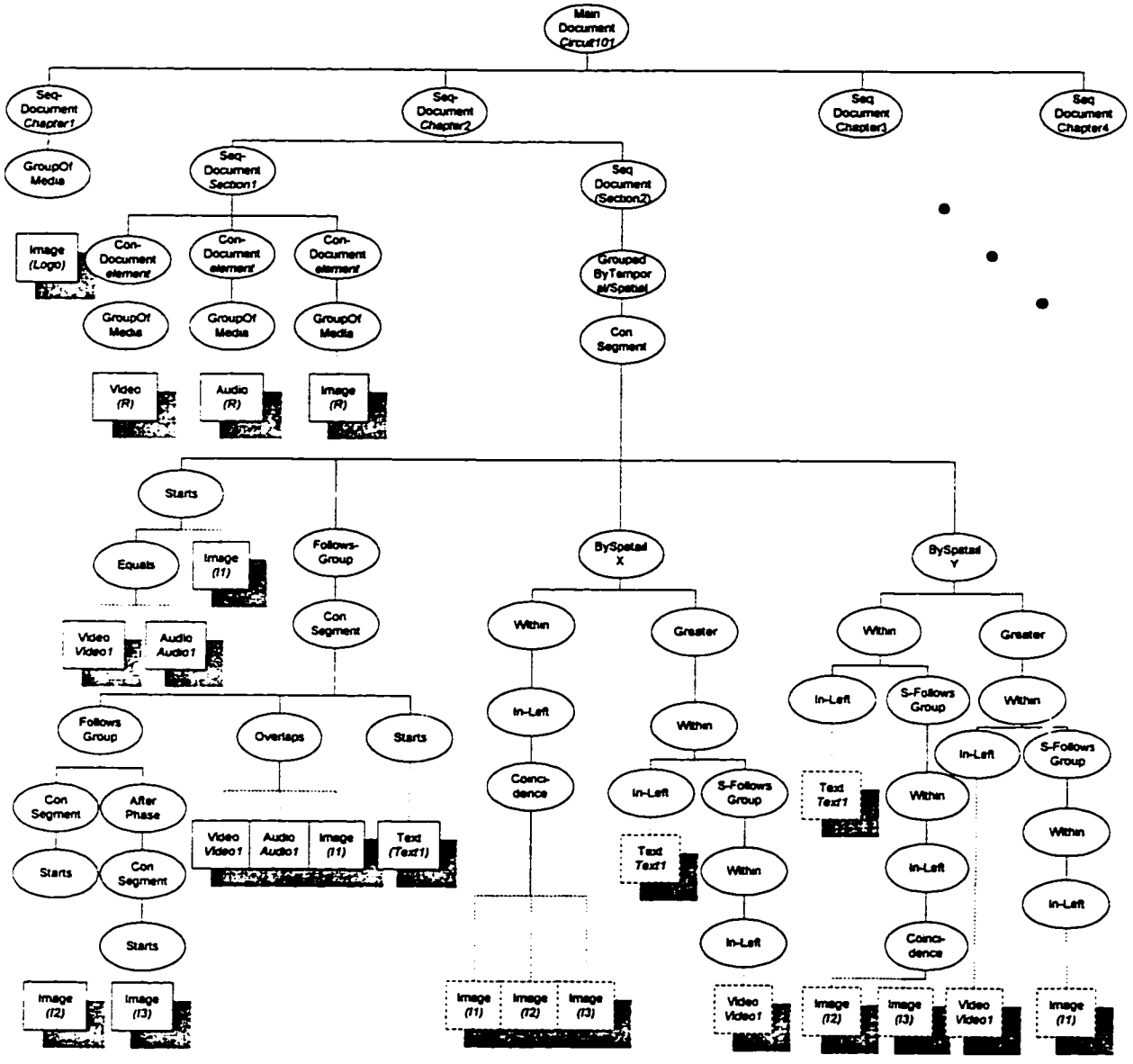



*Figure 4-13 Instances of the database schema classes of the document "Circuits101"*

In summary, the database schema presented supports the documents spatial, logical, and temporal structures. It also gives the option to model a document with or without being logically structured into levels. Thus, it supports two types of temporal structures, logical-temporal structure and media-temporal structure. The former specifies the temporal relationships (in terms of SEQ and CON) among the document segments, whereas, the latter specifies the temporal relationships (in terms of *Starts, Equals, Follows*, *overlap*, and *after*) among the media components. The spatial model supports *Inleft, Coincidence, S-Follows*, *s-overlap*, and *Greater* relationships among the media objects projections on the horizontal and the vertical axes at a time.

Logically structured documents, facilitates document segments play-back and browsing processes by rendering the segments using the logical-temporal structure. Two features can be derived from the media-temporal structure (scenario) provided by the schema. First, the temporal model preserves the playback time-order of the media components of a document or a document segment. That is, for each scenario phase, the "level" attribute associated with each Starts instance (that reflects the hierarchy level of the media-temporal structure) represents the playback time-order of that media component. Second, the five temporal relationships specified above which, as we mentioned earlier, the most perceived temporal relationships between media components by viewers, are preserved in the media-temporal structure, and will be used by retrieval and searching algorithms. Same for the five spatial relationships that will also be used for searching for a specific document segment given its spatial specification.

## 4.5 Query Modules

As shown in Figure 4.2, the query is decomposed into: document, temporal, spatial, and media query components. Consider the query example of section 3.3.4 provided by the interpreter and written in the language syntax proposed in the previous chapter. Let us now visit each of the modules shown in Figure 4.2.

### 4.5.1 Document Query

The decomposer will first communicate with the knowledge model to get more information about the wanted document. This information may include the category which the document belongs to, keywords, date of creation, etc. Consider the provided query example, the knowledge model may conclude that the document may be found under "news" or "G7" category. Furthermore, since one of the media query refers to the date "1995" the knowledge model may also conclude that the document may be created or modified after that date (however, it might not always be the case, the 1995 may refer to a future date). This component of the query will then search for the document through the Document Object identification.

For instance in our example the documents instances belonging to the category "newspapers" will be investigated for the given attributes values and keywords, such as the *creation date > 1994* and subject *keywords* include *"G7" and "Canada"*. This task of the Document Query Component may be summarized in the following pseudocode.

| | |
|---|---|
| *Input :* | *Database (or list of MainDocument instances)* |
| *Investigated class:* | *Category* |
| *Algorithm:* | *Get or filter MainDocument instances for a specific Category* |
| | *Filter MainDocument instances further by matching the given attributes and keywords* |
| *Output:* | *Filtered MainDocument instances* |

The filtered document list is then passed to the Temporal Component.

### 4.5.2 Temporal Query

As the user specifies the temporal structure of the segment he/she wants to retrieve, by representing it graphically using the Timeline temporal model for instance, the query interpreter translates that using the seven temporal operators supported by the language, Chapter 3. The temporal specification section in the query is well defined between TEMPORAL-BEGIN and END statements. The temporal component will then reduce these temporal relationships specified using the five that are supported by the database schema, (EQUALS, STARTS, *OVERLAPS*, FOLLOWS, and *AFTER*). Next a search through the temporal database schema is conducted to find the media instances of interest. The main problem we may encounter is trying to find the media instances that satisfies the *after* relationship. This is due to that this relation is not bounded. This may lead to a lengthy search by traversing the whole scenarios of all the documents in the list. To reduce the processing time we start the search by finding the media objects that satisfy the other relations such as EQUALS.

The five supported relations and how they could be matched are as follows:

| | |
|---|---|
| A EQUALS B: | A and B belongs to one instance of type "equals" |
| A STARTS B: | A and B belongs to one instance of type "starts" |
| A OVERLAPS B : | A and B belongs to one instance of type "overlap" |
| A FOLLOWS B: | A and B should be concurrent. That is, A should belong indirectly to the first instance of type "ConSegment" that B indirectly belongs to.<br>Second, A should also belong indirectly to the "FollowsGroup" instance which belongs to that "ConSegment" instance |
| A AFTER B: | case I: A should not belong directly or indirectly to the same "ConSegment" instance B indirectly does, but to any of the "AfterPhase" instances.<br>case II: A may belong indirectly and through a "FollowsGroup" instance to the 1st "ConSegment" instance B indirectly belongs to if and only if B belongs directly to an "overlap" instance. |

The Algorithm to find the Media instances that satisfies the specified temporal relationships is shown in the following pseudo-algorithm:

| | |
|---|---|
| *Input:* | *MainDocument instances* |
| *Investigated class:* | *GroupedByTemporal/Spatial* |
| *Algorithm:* | *Reduce the seven temporal relations to the five supported by the model*<br>*Filters the ConSegment instances by matching the given EQUALS relationship through traversing all the equals instances belonging to the corresponding ConSegment instances.*<br>*With reference to the found equals instances, filter the media instances by matching the given STARTS and OVERLAPS relationships.*<br>*With reference to the matched "ConSegment" instances work on matching FOLLOWS then AFTER.* |
| *OUTPUT:* | *Media instances, corresponding ConSegment instances* |

### 4.5.3 Spatial Query

It is obvious that in order for two or more media objects to have a spatial relationship, they should be concurrent. Therefore, the spatial search is only performed among ConSegment instances that are provided as a result of the temporal search and refer to *phases*. To simplify the argument for the spatial search, and because the spatial and the temporal model are more or less the same with the only difference in relationships interpretations, the same search discussed in the previous section will be conducted for the Spatial-x and the Spatial-y projections of the media objects. Moreover, to have a spatial relationship between object A and B, A and B should have at least one common scene number in their scene list.

| | |
|---|---|
| *Input:* | *"ConSegment" instances, "Media" instances* |
| *Investigated class:* | *corresponding Spatial instances to ConSegment instances* |
| *Algorithm* | *Reduce the seven spatial relations to the corresponding five relations supported by the model* |
| | *Filters the spatial Within instances by matching the given COINCIDENCE relationship through traversing all the coincidence instances belonging to the corresponding Within instances.* |
| | *With reference to the found coincidence instances, filter the media instances by matching the given INLEFT and S-OVERLAPS relationships.* |
| | *With reference to the matched Within instances work on matching S-FOLLOWS then s-after.* |
| *OUTPUT:* | *Media instances (spatial) intersect Media instances(temporal)* |

### 4.5.4 Media component

Each media has different specification. The Media Query Component first classifies its query part based on the type of media specified. The searching technique depends on the searching algorithm provided. A full-text search on text media type or full image search on image media type may be provided for instance. The model also supports video segments and camera operations which is also supported by the query language. This task of the Document Query Component may be summarized in the following pseudocode.

| | |
|---|---|
| *Input :* | *Media instance* |
| *Start class:* | *corresponding media types.* |
| *Algorithm* | *Classify the media specifications based on the media types. Filter the Media instances further by matching the given attributes and keywords, and running the corresponding matching algorithms.* |
| *Output:* | *Media instances* |

The filtered media list is then passed to the Display Model to be represented to the user and communicate with the database schema to get information on how the spatial and temporal structures of the corresponding document segment these media instances belong to.

## 4.6 Multimedia Document Browsing and Retrieval

A well-designed database schema allows simple and efficient retrieval and navigational facilities within the database. Our retrieval system provides, besides the database schema, facilities to browse and query multimedia databases. In browsing mode, the user can navigate through a hierarchy of categories. Each category refers to a number of multimedia documents as well as sub-categories. When a user chooses a specific category, all referenced documents will be selected. Moreover, each document has a list of keywords that relate

semantically to other documents. At the time a document is created, the author assigns a list of keywords to it. These keywords will be passed to the Document Knowledge Model that will return the appropriate categories for the document. References to the document will be added to each associated category, if the category exists in the hierarchy. Otherwise, a new category instance will be created and inserted in the hierarchy. In the same manner we can browse the content of the document, by navigating through its components in the database schema until reaching the lowest level. Users can also directly access all media objects starting from the top level of the hierarchy, without having to go through the whole hierarchy. For instance, from the video-root, one can browse through all video objects in the database.

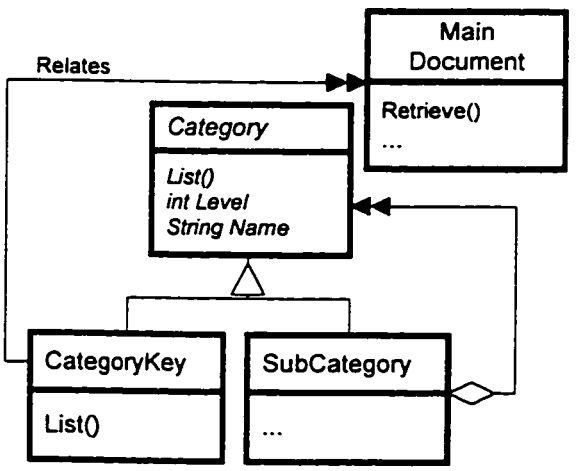



***Figure 4-14 Hierarchy of categories***

Browsing multimedia documents in a database searching for a specific document segment or media through a keyword hierarchy or visual representations such as hypermedia, involves traversing numerous levels of tree-like structures. This task may find the user lost, especially when a detailed view of content is required. Therefore, a growing number of multimedia applications require a retrieval facility that offers detailed views of multimedia

document contents. In the previous chapter we proposed a multimedia query specification language that could be used by multimedia databases. This language permits retrieval of potentially relevant *multisegments* (segments or portions of multimedia documents) from a multimedia database. The proposed query language takes into account the spatio-temporal specifications as well as the media specifications contained in a multimedia document or segment.

# 5. Detecting Cuts by Understanding Camera Operations for Video Indexing

## 5.1 Introduction

In order to facilitate retrieval services for video information one must consider how to build indices from video data. *We* might preview a whole video, select significant shots, and manually assign some tag information to every one of them. The first step in many automatic video indexing techniques is to break down a given video stream into basic units to be used in identification and indexing. Partitioning a video stream is a process of detecting the cuts between consecutive units. The information about the cuts such as their positions, types, and lengths are then used in creating handling units and indices for video browsing and creating a visual summary to a video.

Video segmentation has been recently thought to be important for video computing. Various cut detection methods have been proposed, [TON94][NAG92][OTS93]. Various changes

may happen within a single shot such as those related to camera operations, object movements, flashes, etc. Various similarities between different shots may also occur. Therefore, techniques based on inter-frame comparisons may result in detecting false cuts or missing true cuts if the appropriate threshold has not been chosen. In this chapter we propose a detection mechanism that understands and detects various changes within a scene in order to detect the cuts between different scenes. These cuts may represent sharp or gradual transitions. This mechanism depends on the hue value flow generated as a result of camera operations, object movements, or scene transitions.

This chapter is organized as follows: in section 2 we briefly review basic cut detection and threshold selection techniques. Then in section 3, we discuss our cut detection technique. In section 4 we present and discuss our simulations results. Finally in section 5 we draw our conclusions. But first let us define few terms that will be used frequently in this chapter

- ***Scene**: a scene consists of one or more frames representing a continuous action in time (i.e. no camera breaks) and space forming a piece of video.*
- ***Cut**: defined to be a discontinuity between scenes. A cut is sharp if it can be located between two frames, and it is gradual if it takes place over a sequence of frames.*
- ***Histogram**: A histogram of a frame or an image reflects the number of pixels in that frame having the same parameter value for all possible values of that parameter.*
- ***Hue, Value, and Chroma**: Image parameters forming three dimensional space that represents tri-attributes of psychological human color perception. The hue of a color refers to its 'redness', 'greenness' and so on.*
- ***Frame size**: the number of pixels in one frame of the video clip, and equals to its width times its height.*

## 5.2 Basic cut detection methods

### 5.2.1 Pixel or Pair-wise comparison

Many of the developed algorithms base their scene boundaries detection on a particular video parameter inter-frame difference, [NAG92,OTS91]. Video parameters include intensity, red-green-blue (RGB), hue-value-chroma (HVC), and motion vector. A basic approach to detect a boundary is to compare a specific video parameter, such as intensity, of the corresponding pixels in a pair of consecutive frames. If the total absolute difference exceeds a certain threshold, a cut is detected. This approach is called pair or pixel-wise comparison. It also refers to the technique in which instead of summing up the differences in the intensity values, it counts the number of pixels whose intensity values have changed from one frame to the next. If the number of changed pixels exceeds a certain percentage of the total number of pixels, a cut is detected. A problem associated with these approaches is that camera operation often misleads the algorithm, provoking false cut detections. Furthermore, it is difficult to distinguish a large change in a small area and a small change in a large area. Therefore, false cuts might be detected in the case of rapid change in a small area.

### 5.2.2 Histogram differences

Another approach is to compare the histograms of a pair of consecutive frames, [TON91]. The difference between two histograms gives the sum of the absolute differences in the number of pixels for each value in the two frames. A cut will be detected whenever the

difference exceeds a certain threshold. The idea behind using the overall difference in the parameter distribution is that it is less sensitive to camera or object movements than the pixel-wise comparison. For instance, the intensity distribution is related to the frame, if the objects within the frame are totally changed, the intensity distribution will most probably change along with it. However, if an object in a frame moves, the change in the intensity distribution will be small. As a result, this approach gives slight dispersion within a scene. On the other hand, camera flashes and object movements just in front of the camera may result in false cut detections.

### 5.2.3 Threshold optimization

Threshold selection with which the inter-frame differences are compared is an important factor in detecting true cuts and not missing any. The main issue in cut detection methods may be considered sometimes to be setting the appropriate threshold. In [OTS93], Otsuji and Tonomura proposed a filter to be applied over a cut detection mechanism. The filter separates the distribution of the inter-frame difference values within a scene and those at cuts. In other words, the filter reduces the camera operation and object movement effects. This gives more tolerance in selecting a threshold. They reported an improvement in their detection rate. However, cuts with low difference values will be missed. This is because the filter may further lower these values and exclude them from being considered as cuts. In addition, the filter considers cuts to represent sharp cuts, and acts to reduce the original difference values for gradual cuts, making it more difficult to detect such cuts.

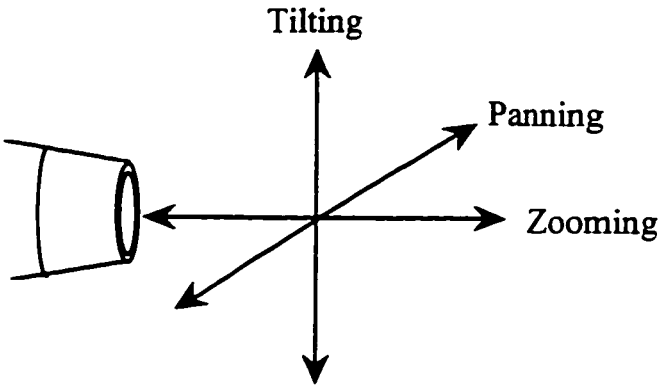



*Figure 5-1 Basic Camera Operations*

## 5.3 New detection mechanism

Frames within a scene may vary widely from one frame to another. Changes occur as a result of camera operations, object movements, and flashes between frames help in augmenting the inter-frame difference values. On the other hand, similarities between consecutive scenes help in lowering the inter-frame difference values at cuts. Therefore, no threshold value will perfectly distinguish between all scene differences and cut differences. Figure 5.2 shows the difference value distribution for a typical video sample. Most inter-frame differences within a scene are expected to have low values, i.e., in the range "a". On the other hand, inter-frame differences at sharp cuts are expected to have high values, i.e. in the range "d". Unfortunately, due to various changes within a scene and various similarities between scenes the inter-frame differences will range widely, making it difficult to select an appropriate threshold. As shown in Figure 5.2, the higher the value of the threshold selected, the lower the false detection rate and the higher the missing rate become, and vise versa.

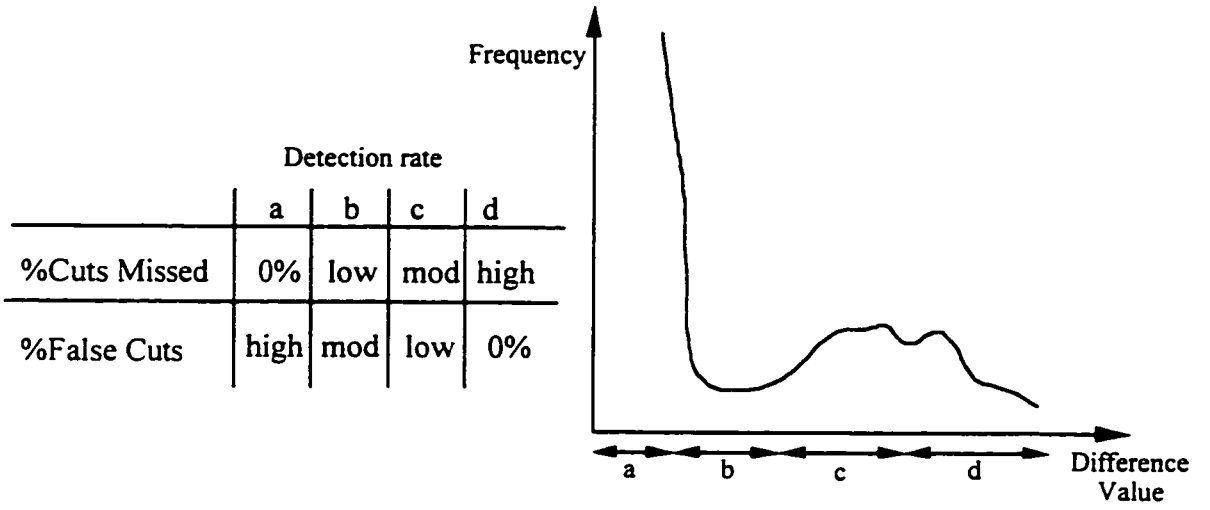



***Figure 5-2 Inter-frame intensity histogram difference distribution for a sample of video.***

We propose a new cut detection method with the following goals:

- Relaxing threshold selection problem,
- Cuts should not be missed,
- False cuts should be avoided.

The first two goals are approached by selecting a low threshold value with which inter-frame differences are compared. The third goal is approached by understanding the various inter-frame changes, whether they resulted from camera operations, object movements, or cuts.

### 5.3.1 Detection Algorithm

An important issue in cut detection algorithms, besides threshold selection, is to select a criterion with which comparisons between frames will be performed. Since the HVC data has shown to mirror the human color perceptions [MIY88], we have selected the hue distribution as the video parameter for performing comparisons. By choosing the hue

distribution, H, instead of the intensity distribution, variations in the intensity values due to light changes and flashes are reduced to some extent.

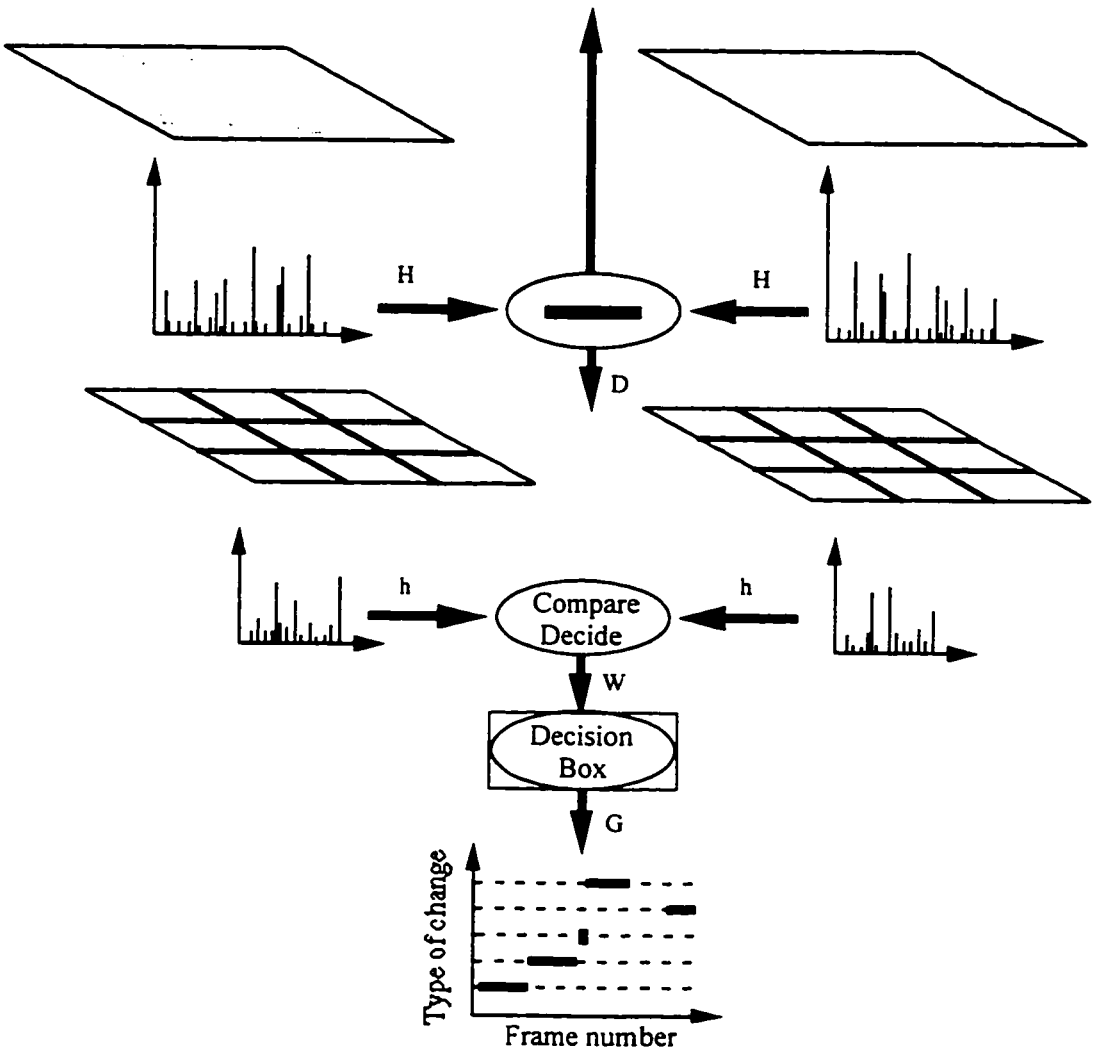



***Figure 5-3 Detection Mechanism.***

In our algorithm, we compare the hue histograms of each pair of consecutive frames. We sum up the absolute differences between corresponding pairs of bins in the hue histograms. The result is then divided by the Max value, twice the frame size, and referred to by the

difference ratio, D. If $H_1(x)$ represents the number of pixels in the first frame having hue components equal to 'x', then the difference ratio D is:

$$D = \frac{1}{2} \frac{\sum_i |H_1(i) - H_2(i)|}{frame\_size} \tag{1}$$

Each difference ratio, D, which represents the size of the area (relative to the size of the frame) where its hue components have been changed, is then compared to a threshold selected in the range *"a"* to ensure 0% missed cuts (Figure 5.2). If D exceeds the threshold, then the two consecutive frames are taken for further processing. The choice of a threshold is a tradeoff between the magnitude of the change information needed and the processing time required. For example, if slow speed camera operations are needed to be detected, then one should select a very low threshold. This, however, will decrease the performance speed of the algorithm causing every pair of frames in the video to be checked for cuts and camera operations.

Once the algorithm detects a change between two frames, it works on detecting the type of that change, whether it is a result of a camera operation or a cut. If neither type is detected, the change is said to be caused by object movements within the frames. To achieve this, each frame is divided equally into *nxn* sub-areas (see Figure 5.3 for n equals to 3), where *n* is a function of the inter-frame difference:

$$n = \left\lfloor \frac{1}{\sqrt{D}} \right\rfloor_{odd} \tag{2}$$

where $\lfloor . \rfloor_{odd}$ is the closest lower odd integer. This gives sub-areas of sizes equal to or larger than the portion of the frame whose hue components have been changed. However, for high inter-frame difference ratios, the minimum value of n is bounded by 3.

The hue histogram, h, for each sub-area is then constructed. These hue histograms along with the sub-area numbers are passed to an algorithm that checks on various camera operations and scene cuts. The algorithm returns initial decisions on the type of change occurred and their corresponding weights, W. These values are then passed to a decision box shown in Figures 5.3 and 5.4, which in turn creates or updates the *detection graph*, G.

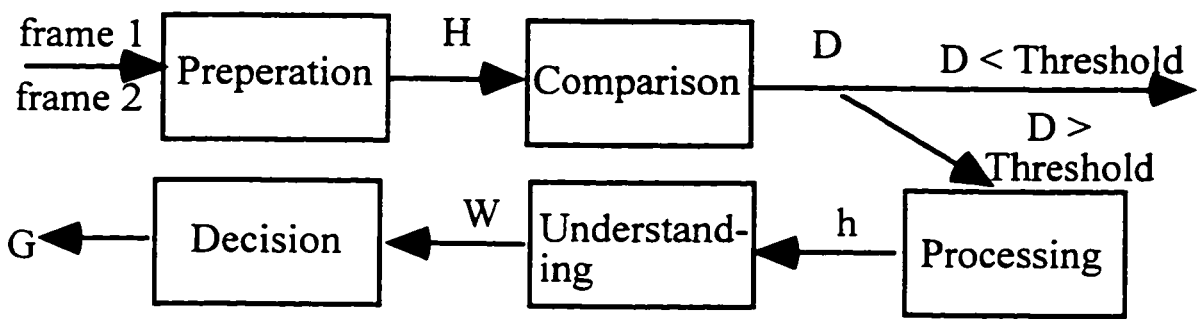



***Figure 5-4 Block diagram***

### 5.3.2 Change Understanding Algorithm

Detection of camera operation is based on field motion analysis. Figure 5.1 shows different camera operations. When a camera moves, the captured objects show a global change. For example, if the camera is panned to the right, background and unmoving objects move to the left within a camera frame. Therefore, each operation has its own characteristic pattern of motion vectors. Working on the motion vectors provided by MPEG and H.261

compression techniques to analyze camera operations and object motions might reduce processing time and is currently under research. However, these vectors do not always represent the accurate optical flow [ZHA94,AKU92].

Initially the algorithm will not distinguish between objects movements that are due to camera operations and those that are due to moving objects while the camera is still. Camera operations will be distinguished in the final stage of the algorithm. The algorithm traces the pixels flow generated by camera operations or object movements, if any, by performing comparisons among the hue histograms of the sub-areas of consecutive frames. This can be further clarified as follows: the most frequent *m* hue values for each sub-area are found from the corresponding hue histograms. Consider 'x' to be a dominating hue value in sub-area *A* of frame 1 (*A1*) with $h_{A1}(x)$ pixels and *B* to be an adjacent sub-area to *A*. If the change in the number of pixels whose hue value is 'x' in *B* from frame 1 to frame 2 is equal to the number of pixels lost from *A1* minus the number of pixels lost from *B1*, then objects in *A* are said to be moved fully or partially to *B* (Figure 5.5). Unfortunately, it is not easy to find the exact numbers of pixels due to pixels values changes (of the same object) from one frame to another. Besides, how could the number of pixels lost or gained be known for sub-areas at the boarders of a video frame. Therefore, we substitute the above statement by simply stating: if

$$\text{Case I}: h_{A1}(x) - h_{B1}(x) \geq \Delta h_B(x) \quad \text{when} \quad \begin{matrix} h_{A1}(x) \geq h_{B1}(x) \\ \text{and} \\ h_{B2}(x) \geq h_{B1}(x) \end{matrix} \tag{3}$$

*or*

$$\text{Case II: } h_{A1}(x) - h_{B1}(x) \le \Delta h_B(x) \quad \text{when} \quad \begin{matrix} h_{A1}(x) < h_{B1}(x) \\ \text{and} \\ h_{B2}(x) < h_{B1}(x) \end{matrix} \tag{4}$$

where $\Delta h_B(x) = h_{B2}(x) - h_{B1}(x)$ and $h_{B1}(x)$ & $h_{B2}(x)$ are the number of pixels with hue components equal to 'x' in sub-area B of frame 1 and 2, respectively, then objects in *A* are said to be "most probably" moved fully or partially to *B*.

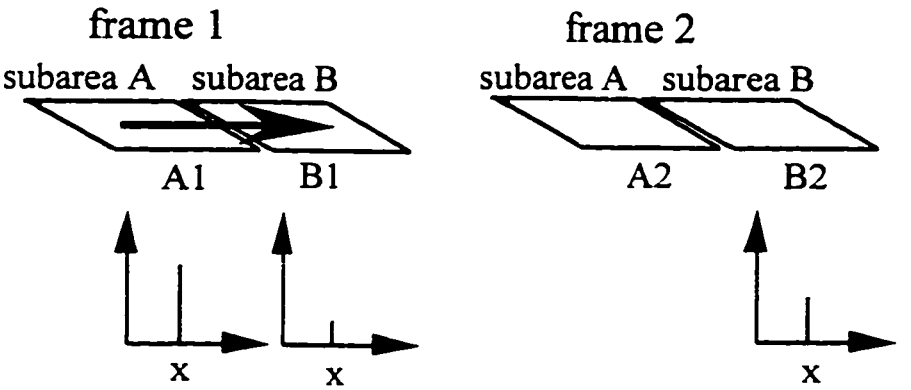



***Figure 5-5 Object moves to the right***

To show the above, consider sub-areas A and B in Figure 5.5. Assume that there is a movement of pixels in the direction from A to B. If the movement is due to a camera operation such as Pan-Left (other than zoom-in or out), we would expect the same number of pixels to move from A to B and from B to its right adjacent sub-area. This is because the whole frame will move as one entity in the direction opposite to the camera movement. Let P refers to this number. By looking at one hue value 'x' and assuming that most pixels in the same scene do not change their values but their locations from one frame to another, we can say "pixels gained whose hue components equal to 'x' by B plus those found already minus

those lost by B equals what remains in B". If we let $p_{A1}(x)$ to represent the probability of the first moving pixel from A1 to B to have a hue component equals to 'x', then $p_{A1}(x)$ equals to:

$$p_{A1}(x) = \frac{h_{A1}(x)}{size(A1)} \tag{5}$$

If we roughly assumed that this probability remains fixed for the other moving pixels then we would have on the average:

$$p_{A1}(x).P + h_{B1}(x) - p_{B1}(x).P \cong h_{B2}(x) \tag{6}$$

$$\therefore \frac{h_{A1}(x)}{size(A1)}P - \frac{h_{B1}(x)}{size(B1)}P \cong h_{B2}(x) - h_{B1}(x) \tag{7}$$

but $S = size(A1) = size(B1)$ ,then

$$\frac{P}{S}[h_{A1}(x) - h_{B1}(x)] \cong \Delta h_B(x) \tag{8}$$

By splitting the frame into large sub-areas (Eq. (2)), we have $0 < \frac{P}{S} \leq 1$. If we substitute P/S with the largest value, we would get:

$$h_{A1}(x) - h_{B1}(x) \geq \Delta h_B(x) \tag{9}$$

The above gives Eq(3). However, Eq(9) is valid iff both sides are positive. If both sides are negative, the comparison sign should be reversed (by looking at Eq(8) we cannot have different signs in Eq(9)). Thus, Eq(4) is reached. ¤

However, there are situations where *B* has lost or gained more pixels whose hue value is 'x' due to movements of objects within a camera frame than what it might gain from *A*, in case I or II respectively. Moreover, the gain or loss in *B* may not necessarily be from or to the left or right adjacent sub-areas. Figure 5.6a represents the former situation for both cases, where the movement detection is missed; whereas, Figure 5.6b represents the latter situation where a false movement is detected.

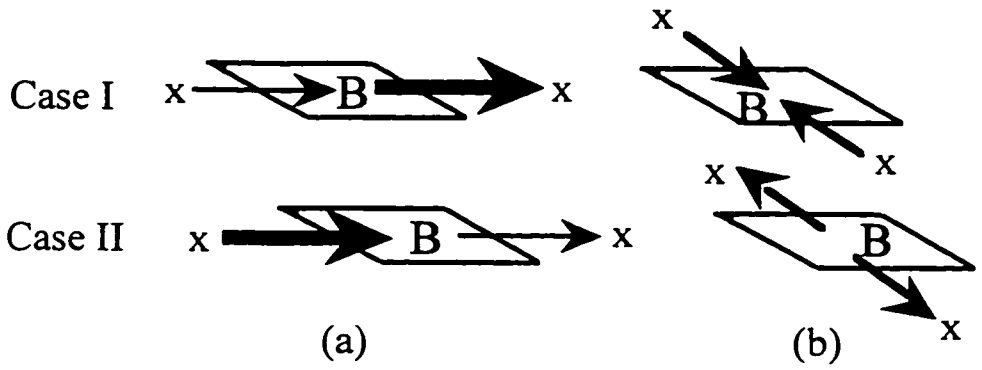



***Figure 5-6 a- Missed right movement detection. b- False right movement detection.***

The other dominating hue values in sub-area *A* are also checked for movement detection. The decision, whether objects in *A* have moved towards *B*, is assigned a weight that is set initially to zero and increased upon each finding by a specific amount, depending on the number of pixels having the same hue value. The size of the sub-area is an important parameter in the algorithm which should be related to the speed of the camera operation. As

mentioned in the previous section, the sub-area size is set equal to the portion of the frame that has been changed, which is proportionally related to the speed of the camera operation.

```
Algorithm I:
Goal: to detect movement from sub-area A to an adjacent sub-area B
Find MA1[m], an ordered array of the most frequent m hue values in A of frame 1
sub_weight=0
for i=1 to m
        if case I:
                if ( hA1(MA1[i])-hB1(MA1[i]) ≥hB2(MA1[i])-hB1(MA1[i]) ) then
                        sub_weight += incA1(i)
        if case II:
                if ( hA1(MA1[i])-hB1(MA1[i]) ≤ hB2(MA1[i])-hB1(MA1[i]) ) then
                        sub_weight += incA1(i)
end-for
```

Where:

$$inc_{A1}(x) = \max(sub\_weight).\frac{h_{A1}(x)}{\sum_{i=1}^{m} M_{A1}(i)} \quad : \tag{10}$$

### 5.3.2.1 Camera Operation Detection

Objects are initially assumed to be moved within a camera frame. If the motion flow pattern found is similar to a pattern formed as a result of a camera operation, then that operation is detected. Only the following operation are selected to be checked on: Panning-Left-Right, Tilting-Up-Down, and Zooming-In-Out. Complex operations are left for future work.

In panning, objects in a column of sub-areas are expected to move to the adjacent column to the reversed direction of the camera operation. In tilting, objects in a row of sub-areas within a frame are expected to move to the upper or lower row depending on the direction of the camera operation. However, for Zoom-in, we would expect the objects in the center sub-area (remember *n* is an odd value Eq (2)) to be partially moved to the four or eight adjacent sub-areas in the next frame. If the motion flow pattern matches the pattern shown in Figure 5.7a or 5.7b, then a zoom-in is detected. For Zoom-out camera operation, the motion flow pattern would match the pattern shown in Figure 5.7a or 5.7b with the arrows reversed. For every finding a weight is added to the decision.

```
Algorithm II:
Goal: to detect Pan_Left (Right movements within a frame)
total_weight=0;
for i = 1 to number of sub-area rows
        for j = 1 to number of columns-1
                detect movement from sub-area(i,j) to sub-area(i,j+1)
                total_weight += sub_weight
        end
end
W[Pan_Left]=total_weight
```

Where *sub_weight* is a value returned by algorithm I. If *total-weight* is given a maximum value, max(*total-weight*), then it would be fair to distribute it uniformly among the sub-areas to check. That is, if we are checking on Pan-Left, then:

$$\max(sub\_weight) = \frac{\max(total\_weight)}{n.(n-1)} \qquad (11)$$

Where *n.(n-1)* is the number of sub-area tests performed. Substitute Eq(11) in (10) to get the following:

$$\therefore inc_{A1}(x) = \frac{\max(total\_weight).h_{A1}(x)}{n.(n-1).\sum_{i=1}^{m} M_{A1}(i)} \tag{12}$$

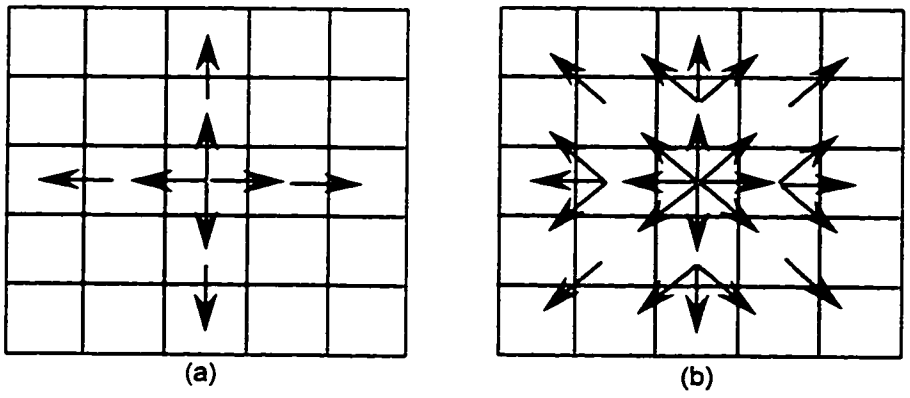



***Figure 5-7 Zoom-in motion flow pattern. a-Four adjacency. b-Eight-adjacency***

### 5.3.2.2 Cut detection

Once cut detection algorithm is invoked, the current frame has been already checked for various camera operations (Figure 5.8). Only frames that final decisions could not be made regarding camera operations are passed through a cut detection algorithm. A decision becomes *final* when the associated W is equal to its maximum value.

In cut detection, instead of summing up the weights for each finding as in the other detection algorithms, an initial weight is assigned to the cut decision. This initial weight is a function of D and W's, camera operations decision weights. Next, a sub-area-wise comparison on the dominating hue values is performed. For each similarity finding, a

specific weight representing the similarity is *subtracted* from the cut decision weight. Similarity comparisons are based on the number of pixels per each dominating hue value as well as their values. As a final step, the different decisions and their corresponding weights are passed to a decision box, where the final decision is made.

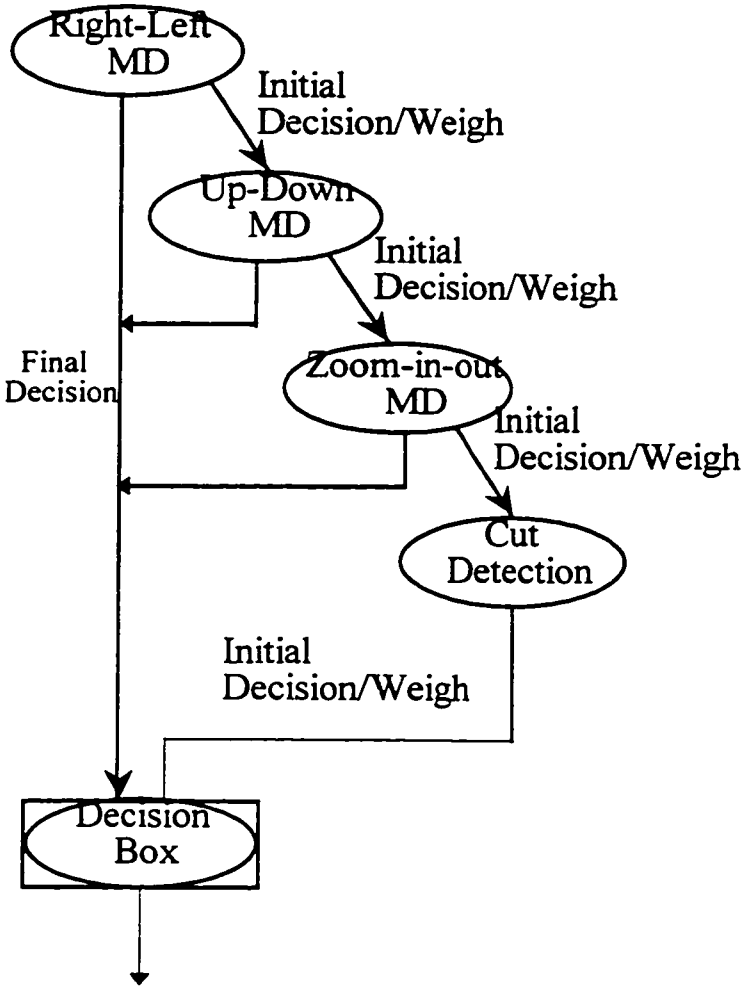



***Figure 5-8 Detection Algorithms flow chart.***

*Object movements detection* is a difficult process due to the wide range of object sizes. At this time, we are less interested in the details associated with object movements than in

camera operations and cuts. However, a video clip is said to include object movements if a frame change is detected and neither cut nor camera operations could be detected.

#### 5.3.2.3 Decision box

This final stage of the algorithm compares the weights associated with each initial decision. As shown in Figure 5.8, there should be at most one final decision (that its associated weight reached its maximum). If one was found, the algorithm simply updates the *detection graph* and keeps this decision and its associated weight in memory in a so-called *decision-weight history*. If no final decision was found, first, percentages of the recent weights from the *decision-weight history are* added to the current weights, such that 20% of the most recent decision weight is added to the corresponding current decision weight, then 10% of the previous one is added to the corresponding current decision weight, and so on. Second, the algorithm finds the maximum weight among the current updated weights, and stores it in the *decision-weight history*. If this weight exceeds 75% of the maximum possible value (*max(total-weight)*) then the associated decision will be final and he *detection graph* will be updated

Two questions may arise: (i) What does happen when the algorithm misses a camera operation or a cut? (ii) What effects do flashlights within scenes have on the final decision? In some cases, when the decision about the type of inter-frame change is not *final* when it is supposed to be, the decision comes few frames later. Camera operations and gradual cuts last for few frames. Due to the *decision-weight history*, the decision weight in question will keep building up and eventually exceed the weight threshold few frames later. This might

not be seen as a problem; however, sharp cuts in this case will be missed. For that reason, the initial weight assigned to the cut decision are relatively high, i.e. the frame change is initially assumed to represent a cut unless proven otherwise.

The answer to the second question is as follows: as Eq(9) will be tested *m* times on every sub-area of the frame (incrementing the decision weight every time it is successful) and by knowing that flashlights first do not often affect more than half the size of the frame (every successful similarity test between sub-areas of consecutive frames will decrement the cut decision weight), and second they do not last long (*decision-weight history* affects every decision making), the effect of flashlights on the weights values is minimized. Finally, flashlights have less effects on the pixels hue components than on the intensity components.

### 5.3.3 An example

Consider for example a video that contains only three frames of size 50x50 pixels each. The frames are numbered from frame 0 to frame 2. Assume now that from frame 0 to 1 there is no remarkable object movements neither due to the objects being moving within the camera frame nor due to a camera operation. Whereas, from frame 1 to 2 the camera pans to the left.

The algorithm will start by comparing the hue histogram of frame 0 with that of frame 1. Since the difference will be very small, close to 0, both frames will be ignored. When the algorithm finds that the difference ratio, D, (say D equals to 3%) for the next pair of frames, frame 1 and 2, exceeds a preselected low threshold, both frames will be equally divided into 25 (Eq(2)) sub-areas as seen in the figure below.

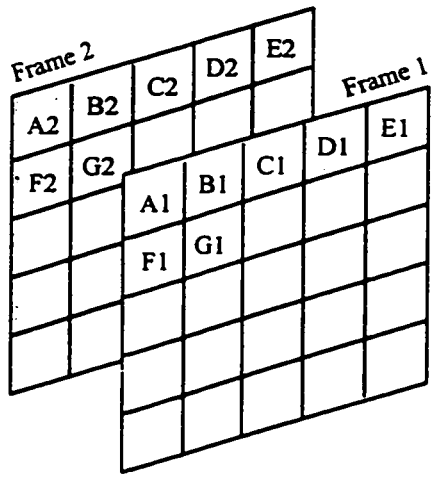



***Figure 5-9 Dividing each of frame 1 and 2 into 25 sub-areas***

Next the algorithm finds out the hue histograms for each sub-area and use them to trace the pixels flow within the frame. It tries to match any of the six patterns of motion vectors generated by the camera operations: Pan-Left, Pan-Right, Tilt-Up, Tilt-Down, Zoom-In, and Zoom-Out, in order to provide the final stage of the algorithm with the matching or decision weights.

Consider Pan-Left camera operation depicted in Algorithm II. Assume that the algorithm parameter *m* (the number of the most frequent hue values for each sub-area) is set to 2 and max(total-weight) is set to 100. If 'redness R' and 'greenness G' are the two most frequent hues in sub-area A1, where $h_{A1}(R)$=40 pixels and $h_{A1}G)$=30 pixels, then Eq(12) gives us $inc_{A1}(R)$=2.86 and $inc_{A1}(G)$=2.14. Algorithm II basically runs Algorithm I on every pair of adjacent sub-areas such as (A,B), (B,C), (C,D), (D,E), (F,G), and so on testing 20 such pairs. Algorithm I checks on pixels movements from one sub-area to the adjacent one, such as from A to B. Two test will be run: one for R and another for G. If both tests where successful the sub-weight will have a value of 5.0 ($inc_{A1}(R)$+$inc_{A1}(G)$). This gives a

maximum value to the total-weight of 100 (20 x 5). All the matching (or the decision) weights are passed to the final stage which will basically compare them together finding the maximum weight. The associated decision will be used to update the *decision-weight history* and possibly update the *detection graph* if its weight exceeds the weight threshold.

## 5.4 Results

We ran the algorithm on four samples of video that differ in (i) types of cuts, (ii) magnitude of action, and (ii) amount of camera operations. The first sample contains only sharp cuts. It is used to study the performance of the algorithm in detecting these cuts. The second sample contains gradual cuts, little of action and no camera operations. It is used to show the advantage of the algorithm in detecting gradual cuts. The third one is selected from a news video to discuss the performance of the algorithm in recognizing camera operations. Finally, the last sample contains a lot of action and camera operations as well as both types of cuts. This sample is used to visually compare the detection graph with the inter-frame difference graph. The algorithm is coded in C language and runs on Sparc 20 workstation. The program reads an Mpeg video stream, decompresses one frame at a time, and converts the pixels values to hue values using a lookup table before running the algorithm. The results showed that the mechanism is a promising one for detecting cuts and coarse camera operations. The cut detection rate (detected true minus false detections over true cuts) was found to exceed 91% .

Figure 5.10 shows a plot of D, the inter-frame difference ratio, and W[T], the weight of the cut decision, versus frame number. The video sample we have selected, contains 100 frames, 93 sharp cuts, and 85 different frames. The inter-frame difference threshold selected is equal to 5%. The hue histograms are constructed using 200 separate hue values. For this test, the decision-weight history was deactivated, since the video sample contains scenes of mostly 1 frame each (for the sake of testing). If the weight threshold is set to 75% to finalize a decision, then a few cuts would be missed due to false camera operation detections. These missed cuts are shown by the small circles in Figure 5.10 at frames 8, 24, 72, 84, 86. Note that the *decision box* returns only the max W and reset the others to zero (this explains why most circles are on the 0% axis). This happens at certain combinations of frame 1 and frame 2. When frame 1 contains a certain pattern of colors, (such as those shown in Figure 5.11), followed by frame 2, a multicolor one, a false camera operation might be detected depending on the pattern.

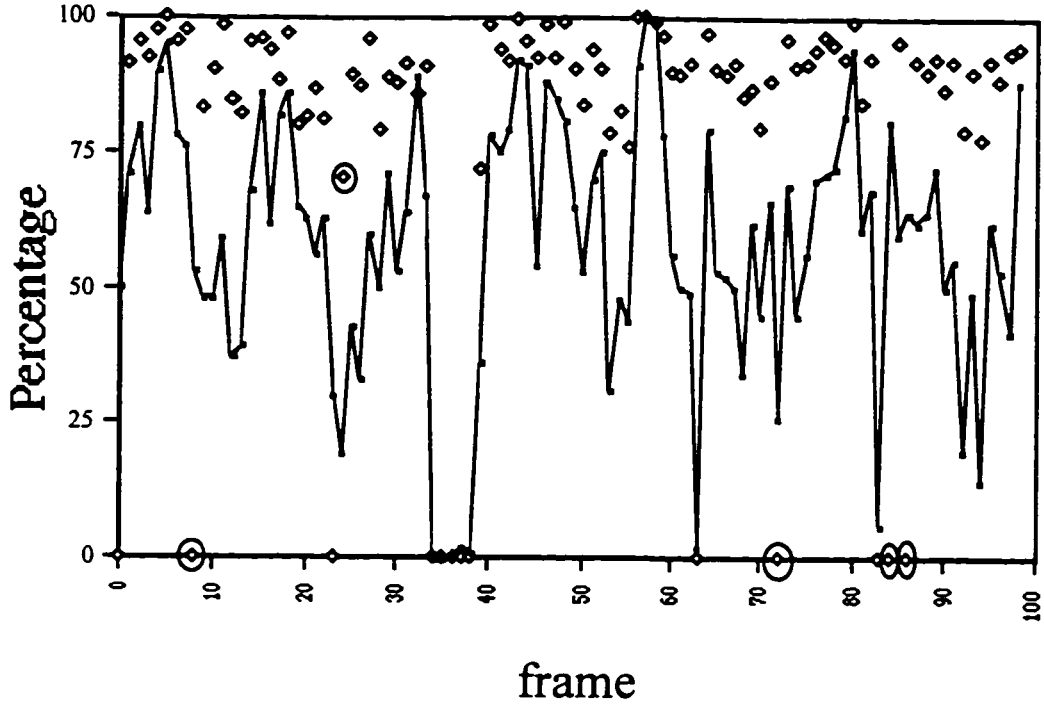
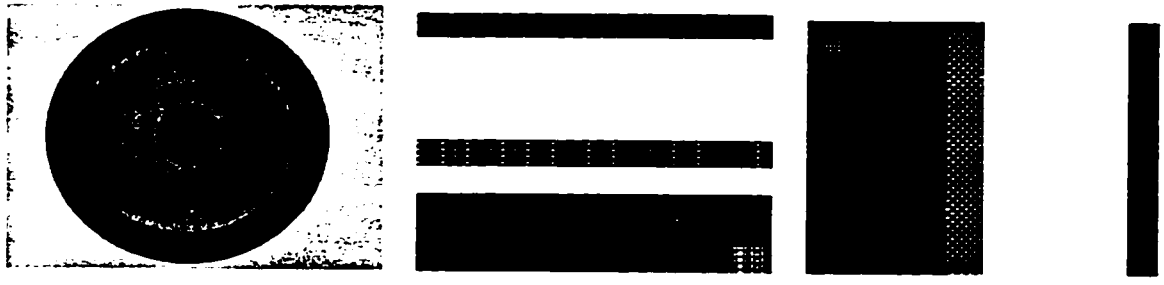



*Figure 5-10 D (continuous line) and W[T] (dotted line) for sample 1 video.*



a b c

*Figure 5-11 a- ZI or ZO false detection. b-TU or TD false detection. c-PL or PR false detection*

To explain this, let frame 1 be Figure 5.11c and frame 2 be any multicolor frame. In this case Eq(3) is most probably satisfied for Pan-Left or Right check. This is because A1 may be a sub-area with one hue value 'x' and B1 be an adjacent sub-area with another hue value. With $h_{B1}(x)$ equals zero and $h_{A1}(x)$ equals or very close to the size of a sub-area, we have

$h_{A1}(x) \geq h_{B2}(x)$ giving Eq(3). To reduce this problem, W's are averaged over a reversed check. For instance, for Pan-Left decision weight, W is averaged with the reversed Pan-Right decision weight:

*W[Pan_Left, frame1=>2]=avg.(W[Pan_Right, frame2=>1], W[Pan_Left, frame1=>2])*

Figure 5.12 shows the improved algorithm compared to the previous one, where the number of missed cuts is reduced to one (at frame 72). On the other hand, by looking at D (represented by continuous line) in Figure 5.10, it is difficult to select an appropriate threshold that detects the cuts, knowing that at frame 23 there is no cut, whereas at frames (24,72,83,92,94) there are cuts.

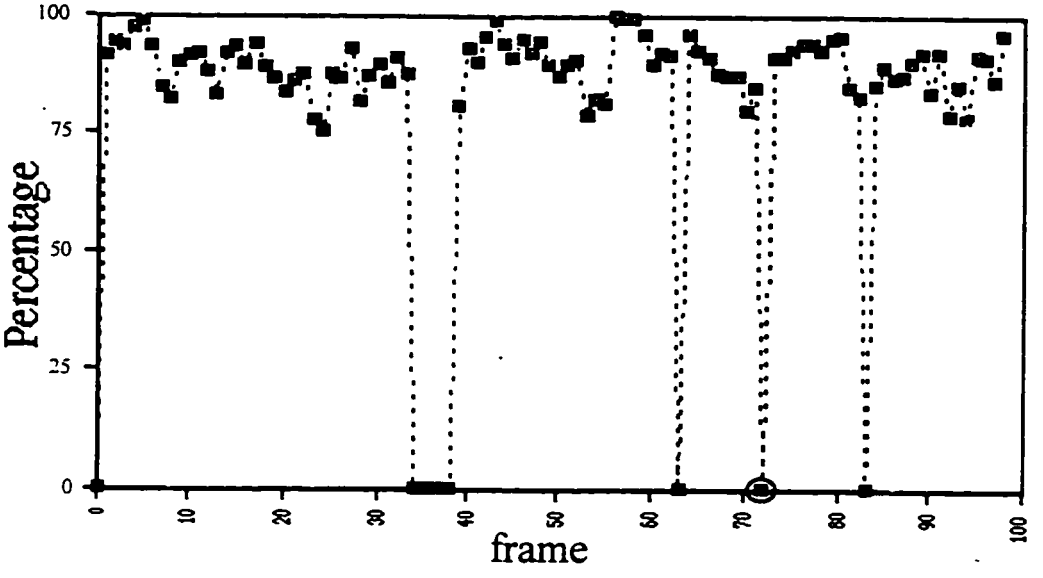



***Figure 5-12 W[T] for the modified algorithm for sample I.***

Figure 5.13 shows another video sample, a commercial that includes three fade in-out (f), one dissolve (d), and one special effect (S) gradual cuts. It also includes a sudden object

movement (m). The inter-frame difference threshold is left at 5%, but the decision-weight history is reactivated. If D (the continuous line) is used to detect cuts, a threshold value between 10% and 20% needs to be selected. This is because if the threshold is lower than 10% the inter-frame change due to object movement will be detected as a cut point, whereas if the threshold is higher than 20% the cuts will be missed. Two things worth to be mentioned here: (i) thresholds lower than 20% are relatively low and will result in detecting false cuts especially if the video contains camera operations, (ii) even if the threshold is selected between 10 and 20%, this will detect most cuts twice, since the value of D increases and then decreases crossing the threshold twice (since they are gradual cuts). On the other hand, the algorithm detects these cuts once and succeeds to distinguish object movements from true cuts.

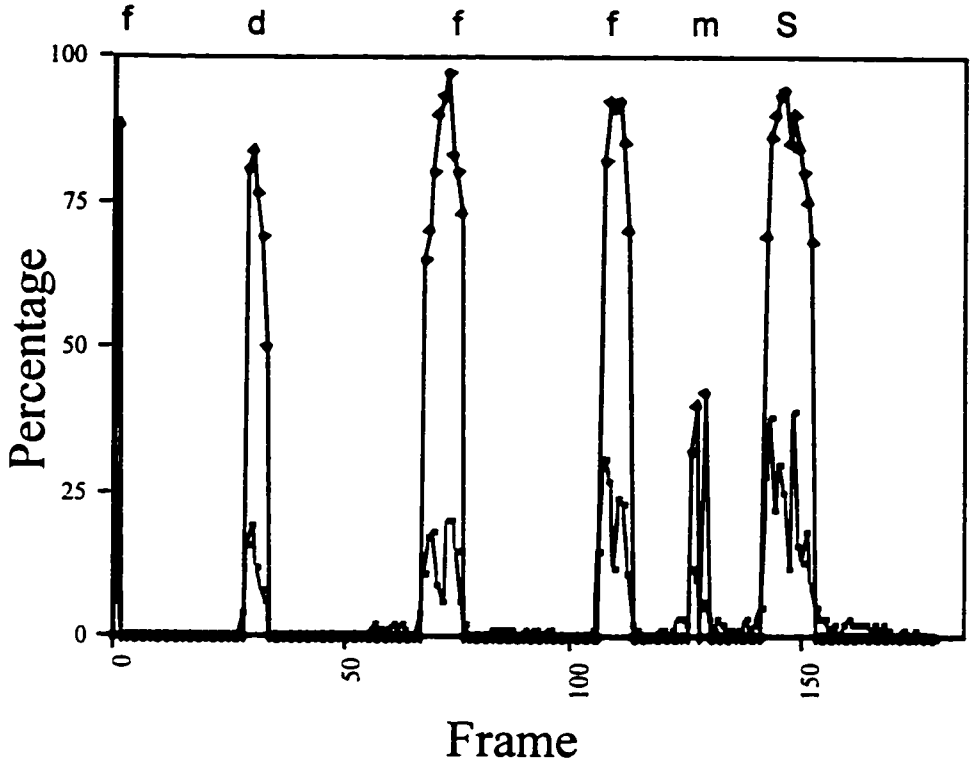



*Figure 5-13 D (continuous line) and W[T] (dotted line) for a commercial sample 2 video*

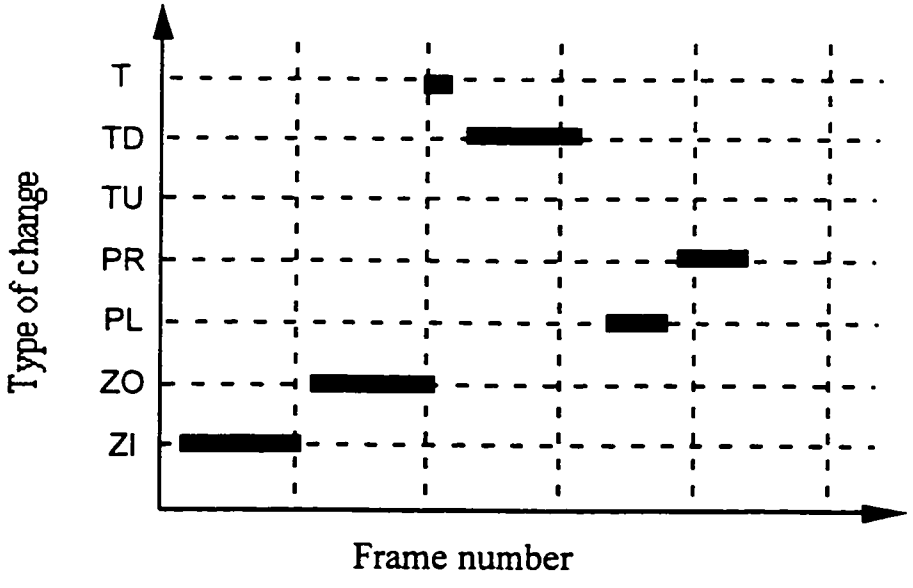



*Figure 5-14 Detection graph*

In Figure 5.14, we show a sample of a *detection graph* where *type of change* is plotted versus frame number. The *type of change* domain is {Zoom-In (ZI=1), Zoom-Out (ZO=2), Pan-Left (PL=3), Pan-Right (PR=4), Tilt-Up (TU=5), Tilt-Down (TD=6), Cut/Transition (T=7)}. Figure 5.15 represents the Detection Graph of that shown in Figure 5.10.

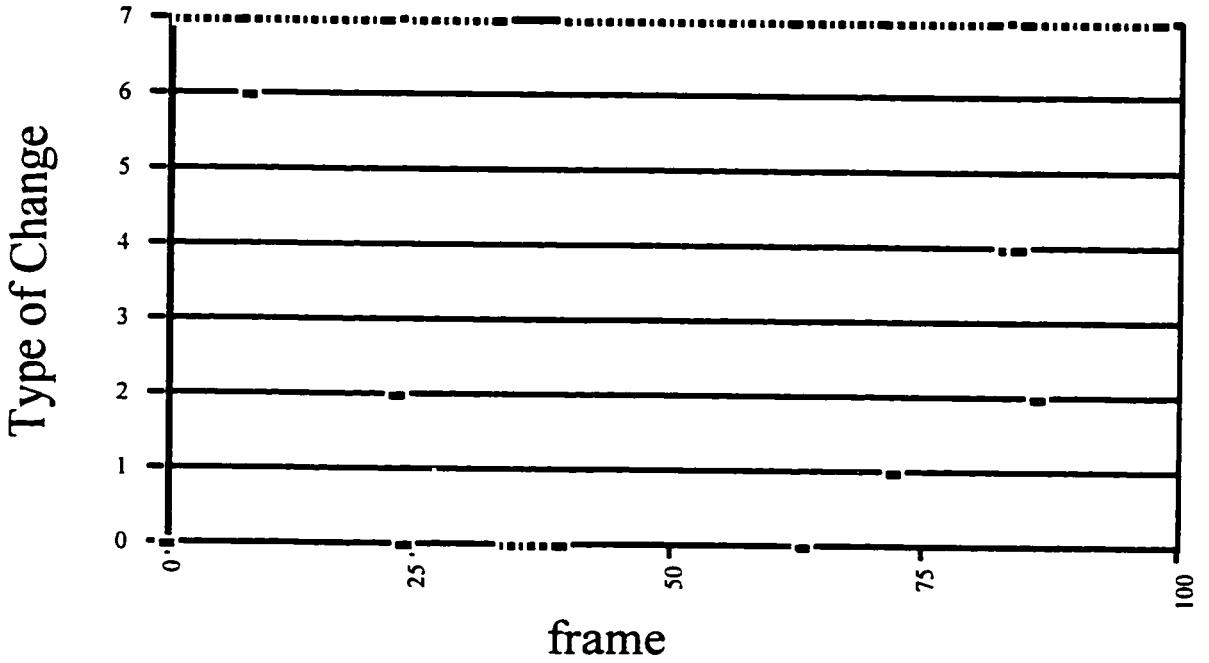



***Figure 5-15 Detection Graph for sample video***

We chose our third video sample to represent a news clip. The clip starts with the anchorperson introducing a subject followed by a few video scenes on that subject. The video was captured at a rate of 8 fps. In a typical news clip, one or more scenes of an anchorperson (especially when there are more than one anchorperson) precedes every video clip about a particular subject or news. In these scenes, there are usually no camera operations but sharp cuts when the camera switches from one anchorperson to another.

Therefore, by detecting at least one camera operation in any scene, one can conclude that this scene might represent a news or a commercial scene. However, if no camera operation is detected in a scene, one cannot conclude that this scene represents an anchorperson scene. In Figure 5.16, a detection graph of the news sample is shown. Up to frame 59, no camera operation or cut was detected. This, however, does not mean that this represents a scene of an anchorperson. Consider for instance the scene from frame 108 to 153, no camera operation was detected even though that this scene is about the subject introduced. The first camera operation is detected in the second scene which is part of the news clip about the subject introduced. The algorithm has a low camera operation detection rate (which does not contradict with the main goal of the algorithm) especially when the video contains composite camera operations (such as pan-left and zoom-in at the same time) which are frequent in most video scenes. Figure 5.17 shows the average value of the six camera operations weights, as well as the maximum weight. These low values are due to composite camera operations. Thus, for this video sample the decision threshold for camera operations is set to 70% instead of 75%.

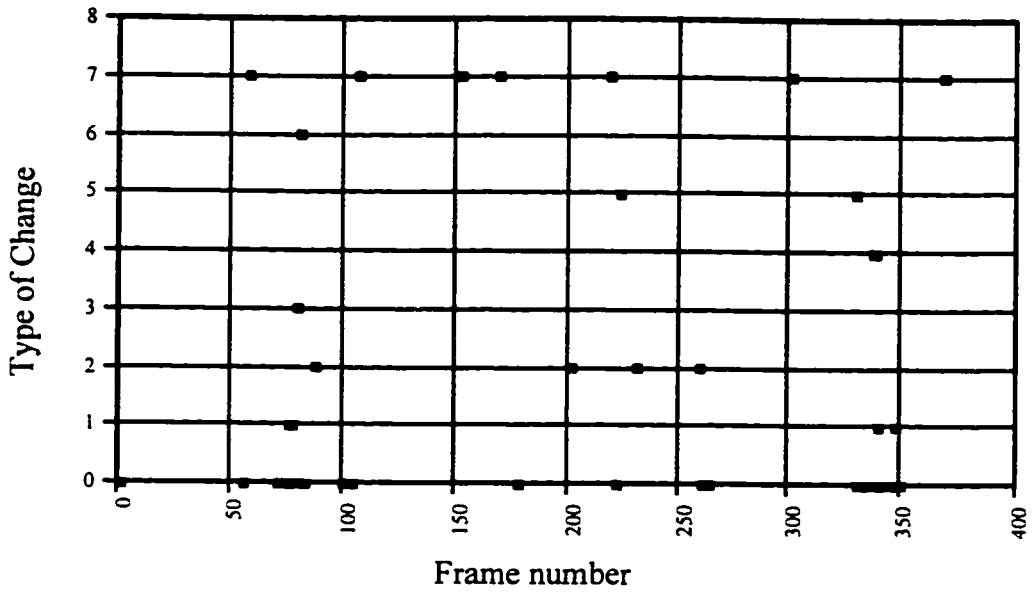



***Figure 5-16 The Detection Graph of the news clip***

Figures 5.18 and 5.19 shows the inter-frame difference ratio, D, versus frame number, and the detection graph of fourth video sample. This video sample represents a clip from Universal Pictures Movie Preview for Carlito's Way and We're Back. Points where an algorithm using an inter-frame difference threshold only might detect as cuts are found to be camera operations using our algorithm. On the other hand, points representing true cuts that could be missed are detected as sharp or gradual cuts.

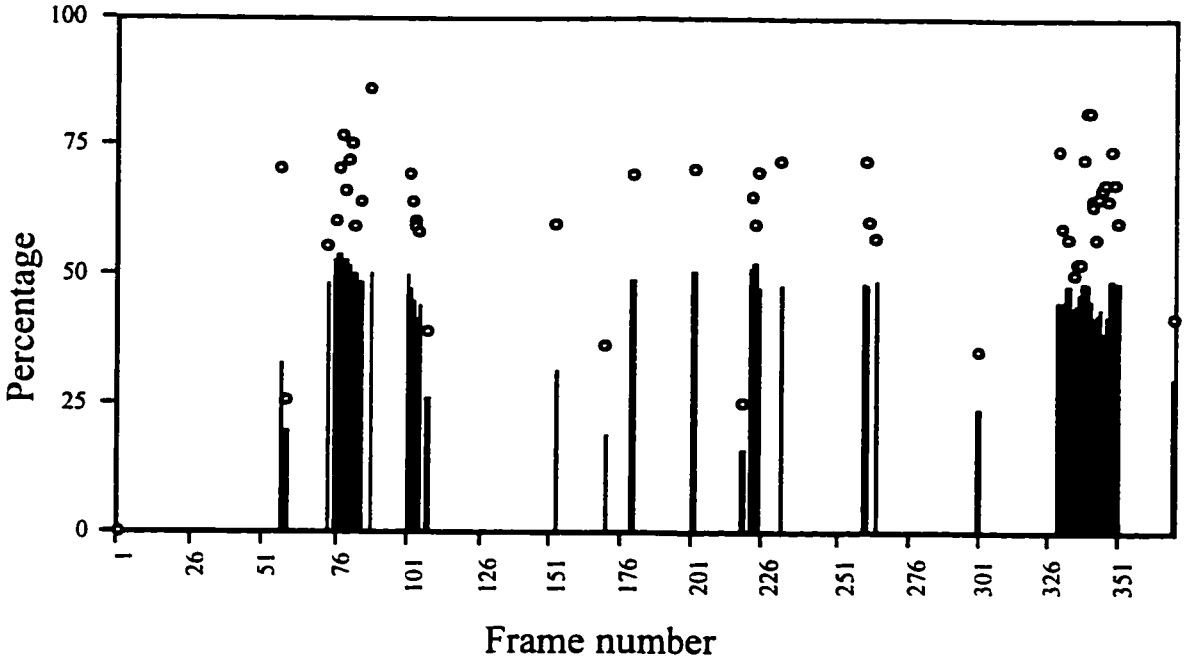



*Figure 5-17 Average weight of camera operations (bars), maximum weight (circles)*

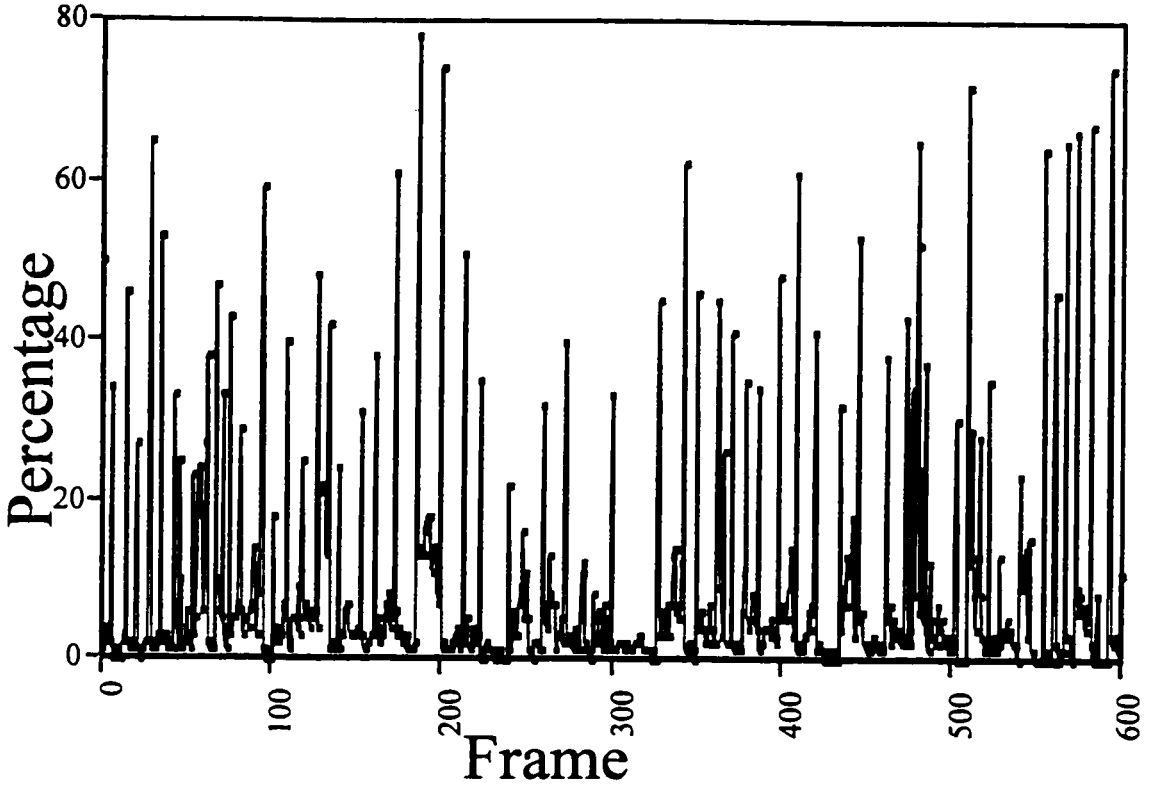



Figure 5-18 Inter-frame difference ratio vs. frame number for movie preview sample.

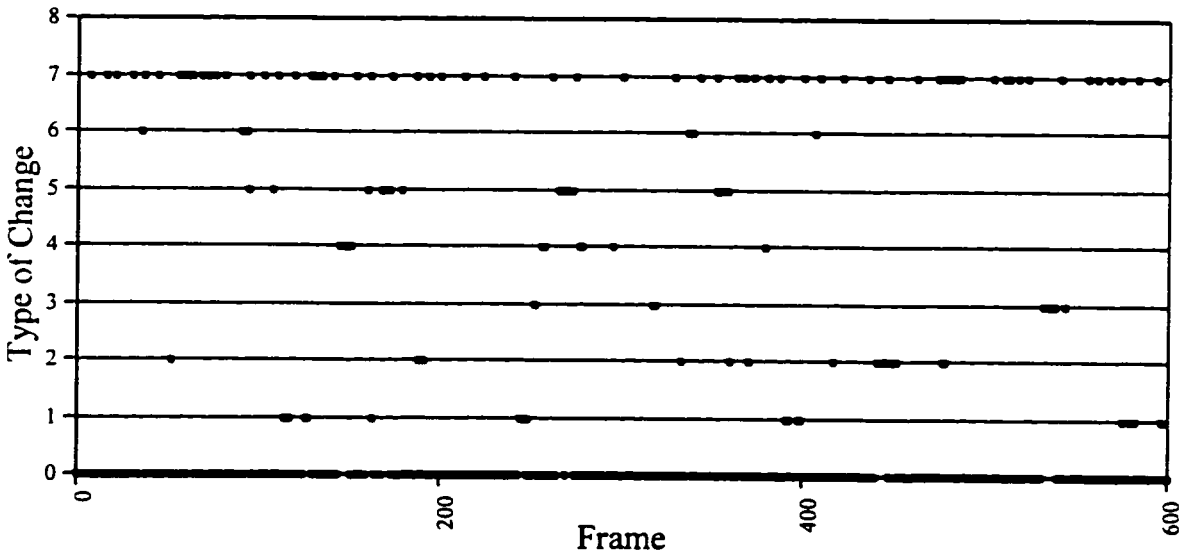



***Figure 5-19 Detection graph of movie preview***

The detection graph constructed as a result of the mechanism would be very useful for video browsing and indexing techniques, and would also give sufficient summary on the video if combined with a so-called clip window that shows thumbnail size representative frames for each scene, [ZHA94b]. This, we believe, gives much better understanding to the content of the video than showing a clip window alone.

# 6. Interactive Multimedia Specification

## 6.1 Introduction

One should distinguish between hypermedia, passive multimedia, and active multimedia presentations. *Hypermedia* implies store-and-forward techniques where user actions, typically mouse-selections on hot-spots, cause the system to retrieve a new "page" of data which could be an image, text, video etc. There are usually no temporal relationships

between media. *Passive multimedia* implies a fully synchronized document that "plays itself back" in time, synchronizing all media objects together. *Active multimedia* implies that there are hypermedia-type choices presented to users during the playback of a multimedia document which allow the user's interaction to "drive" the playback.

As retrieval facilities are needed for databases containing passive multimedia documents there is also a potential need for facilities that manipulate and retrieve active multimedia documents. Our target in this chapter is to offer a specification language for interactive multimedia documents. As explained in chapter 3, multimedia information can be decomposed into four types: Media information, Temporal information, Spatial information and External information. The External information is related to interactive multimedia documents and hence will be discussed in this chapter.

## 6.2 External Information

There are enormous amount of information that could be used to describe interactive documents. We choose to consider the information that are related to user actions that are either document-specific or user-specific. Document specific information includes possible user actions, range of possible time of occurrence, activation time, and reactions. In this chapter, we will propose a temporal model that is capable of representing interactive multimedia documents scenarios. The model depicts information about possible user actions as well as their corresponding reactions, such as starting a new media, stopping a current media, etc. The model has also the power to express the time relationships between the

various user actions, in spite of the fact that their time of occurrences are not known at authoring time, i.e. prior to the time they actually take place. Such information, which can be extracted from the model, can also be used to index such interactive documents.

User-specific information, on the other hand, describes actual user actions that have already taken place. It includes information that are specific to the user who did the action, such as WHO is the user, WHERE was the user, WHAT was the action, and WHEN did the action take place. These information will be considered in the specification language to query on such documents. These information are dependent on what the system records during the playback of the interactive multimedia document.

Figure 6.1 shows a typical user model where the client monitors the user's actions during the document play back. When a user requests to play back a specific document, the client first loads the corresponding temporal information of the documents (document scenario), which will include (besides the synchronization constraints of the various media to be displayed) information about possible user actions. The client then monitors user's actions at only these points where user actions are expected and filters them to actual actions that have taken place. It then stores the corresponding information (discussed in the previous paragraph) down in the user's local disks. After the user is finished, the recorded files are moved to a public disk which may move back to the user's local disk during retrieval.

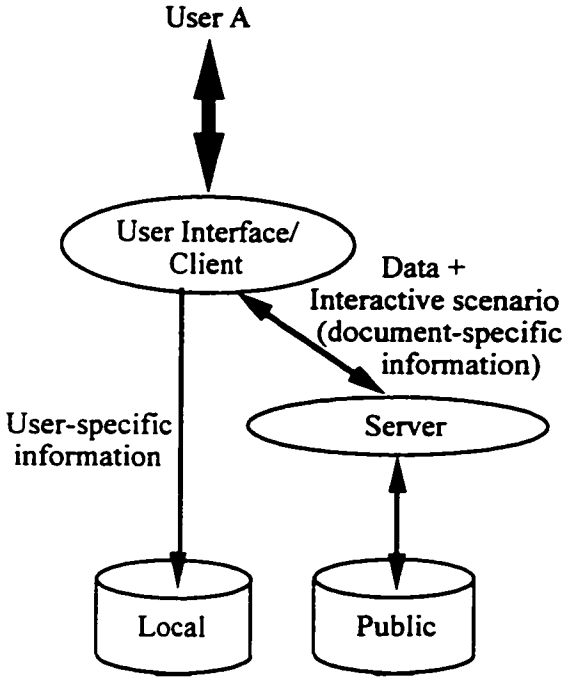



***Figure 6-1 User Model***

In the following, we will present our temporal model that represents passive and active multimedia documents scenarios. The scenario representation using the model contains all the document-specific information that can be used to describe the document. User-specific information on WHAT and WHEN are subset of the document-specific and thus can be represented in the model. Hence, the proposed model will have the capabilities to represent all the key events that are expected during the active document play back. These events are depicted in the model as well as with the letter 'C' which stands for a user's choice.

## 6.3 Representing Interactive Multimedia Scenarios

One of the main issues in temporal models of scenarios is the flexibility the model has to express different temporal relationships. In this chapter, we are mainly concerned with the

temporal behavior of scenarios; other attributes of the document including its layout, quality, and playback speed are not the primary focus of our investigation. Of main issue to us was representing and modeling multimedia scenarios (i.e. fully specified temporal entities involving multiple media) which "play themselves back" (that is, multiple media are rendered automatically before the users' eyes) *as well as* letting the user interact with the running presentation, "driving" it in a custom direction. We provide a new representation for asynchronous and synchronous temporal events.

In this chapter we will make use of definitions originating from [BUC92]. They are summarized as follows:

- ***events**, points at which the display of media objects (text, video, etc.) can be synchronized with other media objects. In our work, we focus on start and end events but we can generalize to internal events.*
- ***synchronous events**, those with predictable times of occurrence - e.g. their temporal placement is known in advance.*
- ***asynchronous events**, those with unpredictable times of occurrence and durations - e.g. their time of occurrence cannot be known in advance.*
- ***temporal equality**, a synchronization constraint requiring that two events either occur simultaneously or that one precedes the other by a fixed amount of time.*
- ***temporal inequality**, a synchronization constraint requiring for example that for two events, A and B, they occur such that A precedes B by an unspecified duration, by* at least *some fixed time, or by* at least *some fixed time and* at most *another fixed time.*

### 6.3.1 A Temporal Model for Active Multimedia

Perhaps the most prevalent model is the timeline [BLA91][GIB91], a simple temporal model that aligns all events (start and end events of media objects) on a single axis which

represents time. Since the events are all ordered in the way they should be presented, exactly one of the basic point relations, *'before (<)', 'after (>)', or 'simultaneous to (=)'*, holds between any pair of events on a single time line (Figure 6.2). The timeline model, though simple and graphical, lacks the flexibility to represent relations that are determined interactively, such as at run-time. For example, assume a graphic (say a mathematical graph) is to be rendered on the screen only until a user-action (say a mouse-selection) dictates that the next one should begin to be rendered. The start time of the graphic is known at the time of authoring. The end time of the graphic depends upon the user-action and cannot be known until presentation-time, hence the scenario cannot be represented on a traditional timeline which requires a total specification of all temporal relations between media objects. Consequently, there is a need for a model which accommodates partial specifications or interactive multimedia scenarios.

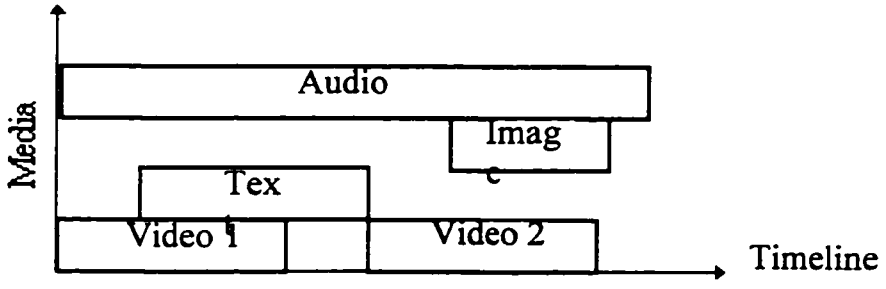



***Figure 6-2 The basic timeline model***

We should distinguish between hypermedia, passive multimedia, and active multimedia presentations. *Hypermedia* implies store-and-forward techniques where user actions, typically mouse-selections on hot-spots, cause the system to retrieve a new "page" of data which could be an image, text, video etc. There are usually no temporal relationships

between media. *Passive multimedia* implies a fully synchronized document that "plays itself back" in time, synchronizing all media objects together. *Active multimedia* implies that there are hypermedia-type choices presented to users during the playback of a multimedia document which allow the user's interaction to "drive" the playback.

In the remainder of this section we will enhance the traditional timeline model by including temporal inequalities between events. We strive to model *active multimedia* documents graphically.

### 6.3.2 Background Work in Active Multimedia

We found that modeling user actions as media objects was an early, effective way of representing active multimedia scenarios. That is, traditionally, the vertical axis of the timeline is reserved for media such as text, graphics, audio and video. By creating a new type of media object called *choice*, we can accomplish much. Firstly, this new object is associated with a data structure with several important fields. The key fields will be: *user_action, region,* and *destination_scenario_pointer*. User_action completely describes what input should be expected from the viewer of the presentation; for instance *key-press-y*, or *left-mouse*. Region describes what region of the screen (if applicable) is a part of the action; for instance *rectangle(100,100,150,180)* may describe a rectangle in which if a user clicks with the mouse, some result is initiated. Destination_scenario_pointer is a pointer to some other part of the scenario, or in fact a different scenario; for instance the author may specify that if the user clicks on a region of the screen at a certain time, the presentation "jumps" to a new chapter.

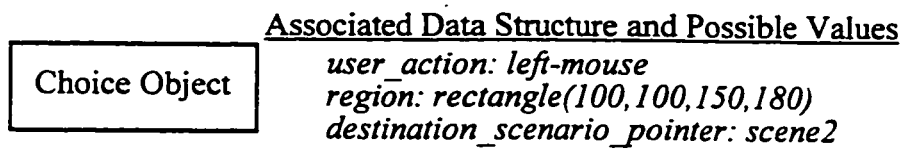



***Figure 6-3 The choice object and its data structure***

This media object "choice" may be placed directly on the traditional timeline. Let us work with a running example to clarify. Suppose there is a scenario wherein a video of American presidents is being rendered, along with an audio track. The video serves to introduce us to three American presidents, Clinton, Bush, and Reagan. Suppose again we have text boxes which are rendered displaying the name and the age of each president as they are introduced in the short video clip. The scenario on the traditional timeline looks like Figure 6.4.

Now suppose the authors wish to create some additional timelines, one for each president. In this way a user might make some selection during the playback, the result of which would be "jump" to a more in-depth presentation of the president currently being introduced - i.e. active multimedia. To do this each of the in-depth scenarios must be authored. Then we must add three choice objects to the original timeline whose data structures contain: user_action=left-mouse, region=the appropriate layout location on the screen, and destination_scenario_pointer=the appropriate scenario. The choice objects are added in Figure 6.5.

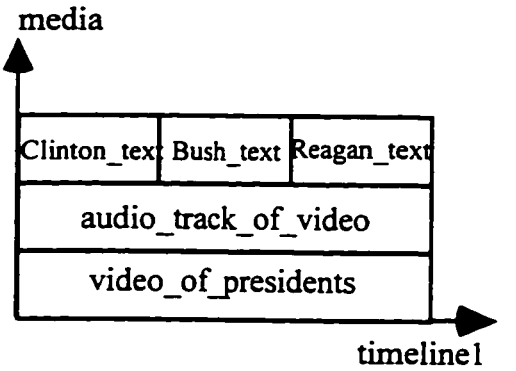



***Figure 6-4 A Simple Timeline***

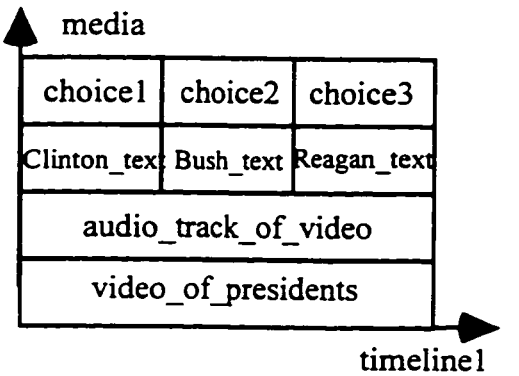



***Figure 6-5 Adding choice objects***

In effect the active multimedia scenario is finished. Since the choice data structures are completed, each such object refers to some other sub-scenario as in Figure 6.6.

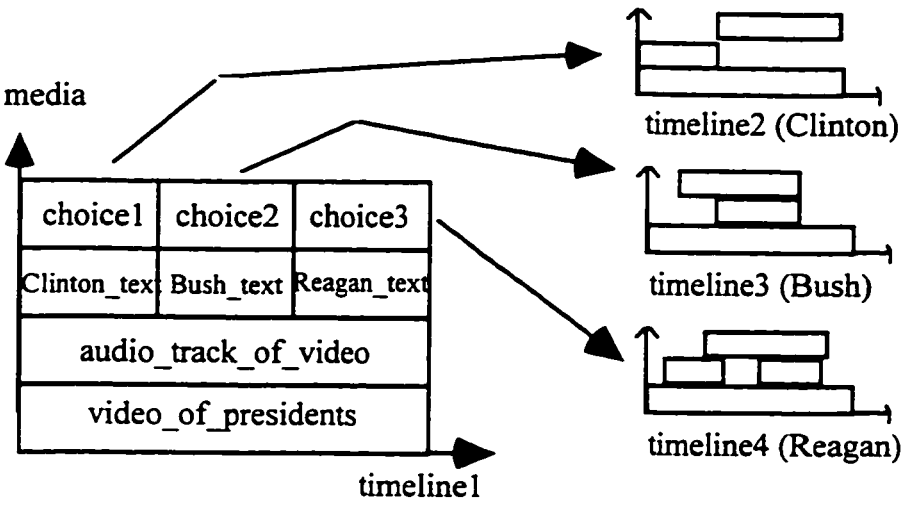



***Figure 6-6 The choice objects and the destination scenarios they are associated with***

The resulting scenario is indeed an interactive one. Note firstly that choice objects, like other media objects, have a duration. This means that the user has a "window of opportunity" to make the action that initiates the choice. If the associated action is not made during this time, the user loses the chance to make it. That is if the user does not make the mouse-selection while the text object Clinton_text is being rendered, he/she will continue to see the rendering of timeline1 and a new choice (the one associated with President Bush) will be offered to the user. If the appropriate choice is made, rendering continues from the destination timeline and the original scenario is terminated.

We also developed a way to represent the active multimedia scenario graphically so that authors can visualize their work. We discovered that a tree of timelines is effective. The interactive scenario described above may be represented in the timeline tree shown in Figure 6.7.

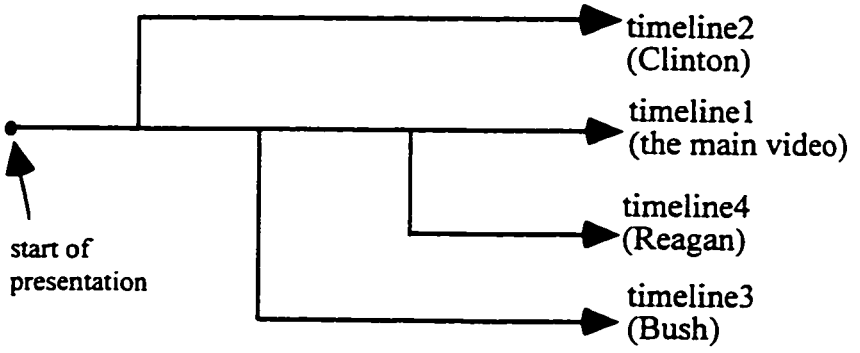



***Figure 6-7 A timeline tree representation of the interactive scenario in Figure 6-6***

Our earlier work focused on representing choice objects and interactive documents graphically in order for authors to visualize their work. We modeled non-halting user interactions and choices. That is, interactions were modeled as having temporal longevity; as a result we could model choices which were transparent in that they did not force the presentation to halt and wait for input, rather they were integrated seamlessly into the presentation.

Active multimedia presents many of the same difficulties as hypermedia. In [HAR94] the authors discuss the Amsterdam model which addresses issues relevant here such as "link context" (i.e. when a user choice is made, what happens to the media which were being rendered at the source? Do they stop or continue in the new context?). Another question which arises in active multimedia is, "how to represent events whose start or end times are not known at authoring-time?" These are issues that we confront in the next subsection.

### 6.3.3 The Enhanced Temporal Model

In our enhanced model, the traditional rectangle which represents a media on a timeline is split into three basic units as shown in Figure 6.8a. The edges of these units, which represent start or end events, are either straight or bent lines. Straight lines represent synchronous events and bent lines represent asynchronous events. The actual time of an asynchronous event can only become known at run-time, and will be shifted by $\delta$ seconds to the right on the time axis, where $\delta$ is a non-negative number representing the user response delay (Figure 6.8b). Again, at authoring-time the value of $\delta$ is unknown.

The author may specify a limit to the value of $\delta$ where if the user does not respond to some interactive prompt a default response may be assumed (that starts rendering the media object in the example shown in Figure 6.8b). Therefore, defining the maximum value for $\delta$ is useful and necessary. The length of the unit from the left sharp point to the right straight line represents the maximum value of $\delta$ (see Figure 6.8b).

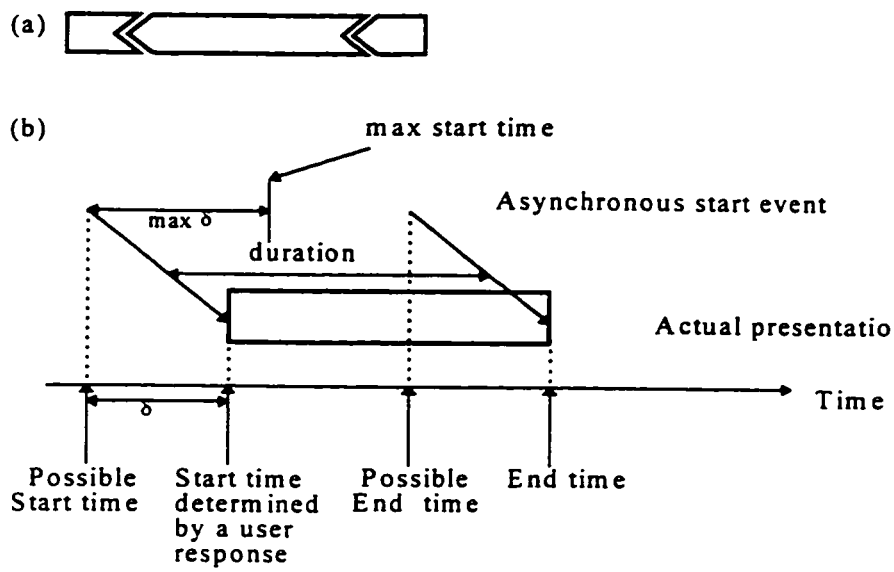



Figure 6-8 . a- Basic representation units. b- User response delay

Based on the three basic units shown in Figure 6.8a, many different shapes could be formed, but only six of them are applicable to interactive scenarios. These six shapes and their descriptions are shown in Figure 6.9. The assumptions used in forming the shapes are:

| | |
|---|---|
| *1.* | *An event that is temporally related to an asynchronous event is itself asynchronous. A simple example is that if the start time of a media that has a specific duration is unknown, then its end time is also unknown.* |
| *2.* | *∂, the user response delay, is a non-negative value.* |

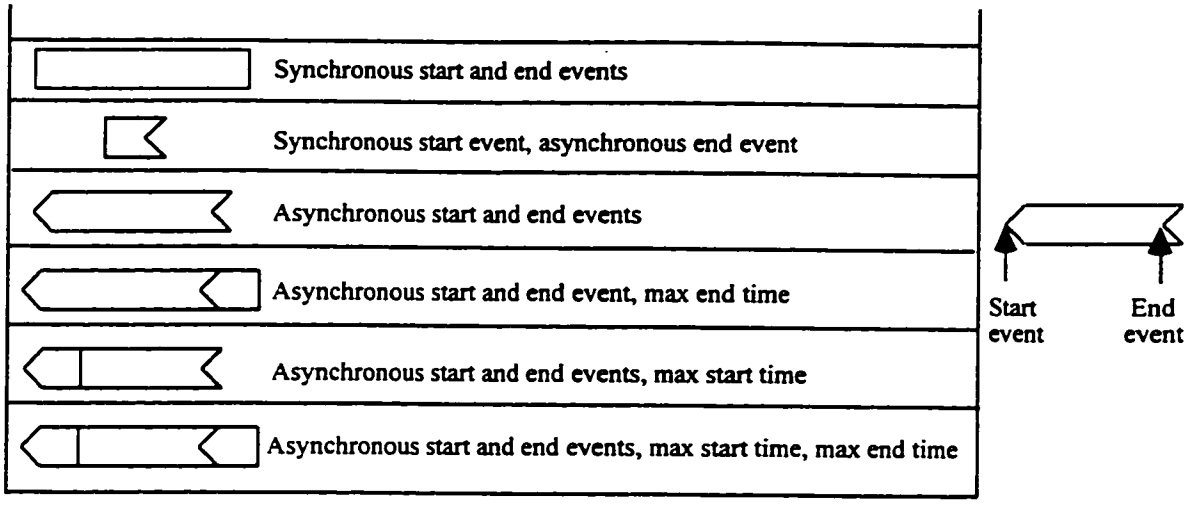



***Figure 6-9 Applicable units***

#### 6.3.3.1 Timeline Tree

Because of Assumption 1, we introduce a symbol that is associated with each asynchronous event to distinguish between temporal equalities and temporal inequalities with other asynchronous events. Events that have temporal equalities between one another, i.e., do not admit terms such as "at least" or "at most", carry the same symbol; otherwise the symbols are different. We use the symbol $Choice_i$ (abbreviated $C_i$) where $i$ refers to timeline(i), $i \geq 0$. Timeline(i) is a timeline that "branches" from timeline(j) ($j<i$), because of the unknown user response time. $\delta i$ will then refer to the user delay in responding to the first event with symbol $C_i$. Associated with $C_i$ is a data-structure which determines the user-action which will initiate that object (for instance, pressing the key "k", or clicking the mouse in some rectangular region of the playback space as mentioned in section 3.2 for the choice objects). If we defined $time_i(j)$ to be the instant of time when j first appears with respect to timeline(i), then we have the following formulae:

$$time_i(Origin_j) = time_i(C_j) + \sum_{n=i}^{j} \delta_n \quad \text{where i \_ j and i} \geq 0 \tag{1}$$

$$Minimum\ duration\ of\ a\ presentation = time_0(possible\ end\ of\ presentation) \tag{2}$$

$$Length\ of\ the\ actual\ presentation = time_0(possible\ end\ of\ presentation) + \sum \delta \tag{3}$$

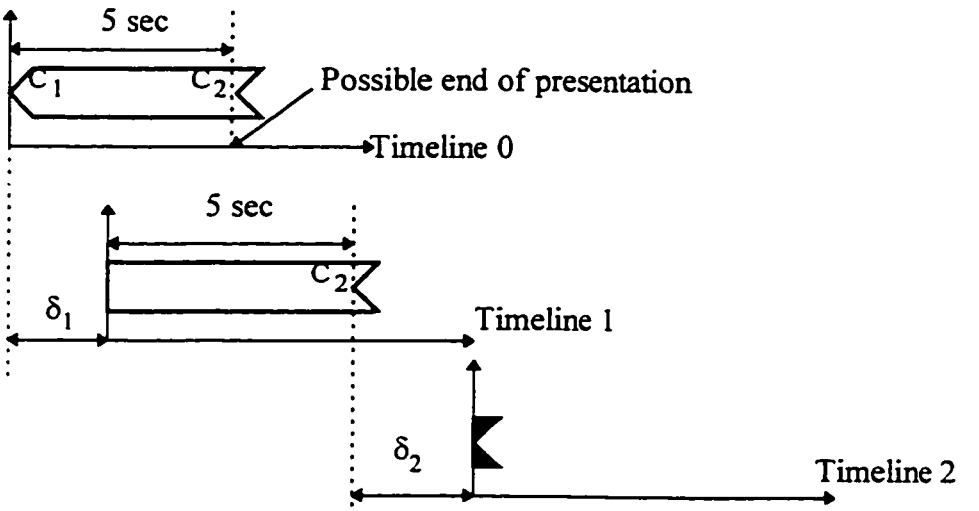



***Figure 6-10 Illustrating multiple timelines depending on point of reference***

To illustrate further, Figure 6.10 represents a scenario in which a media object is rendered immediately after a user action (say a mouse-selection) and the object ceases to be rendered after a second user action. Furthermore, the second user action will only be effective at least five seconds after the first action. The start and end times of the object involved are hence unknown at authoring-time. Since there is no temporal equality between the two events such as, "end event occurs exactly five seconds after start-event", both must have different symbols, $C_1$ and $C_2$. As the scenario is rendered at run-time, the start time will become known when the user performs some action to initiate the object. Since the time it takes the

user to respond is not known in advance, a different timeline (timeline(1)) which "clips" out the user delay is used to represent the result of this user action. Similarly, the object's end time only becomes known when the second user action takes place. From Figure 6.10 we may write the following:

$$time_0(Origin_1) = time_0(C_1) + \delta 1$$
$$= \delta 1$$
$$time_0(Origin_2) = time_0(C_2) + \delta 1 + \delta 2$$
$$= 5 + \delta 1 + \delta 2$$
$$time_1(Origin_2) = time_1(C_2) + \delta 2$$
$$= 5 + \delta 2$$

*Minimum duration of a presentation = 5 sec*

*Duration of the actual presentation* $= 5 + \delta 1 + \delta 2$

Note here that we are closer to active multimedia. Consider for example the following presentation: the user is presented with a graphic of a car. Embedded onto the presentation screen (actually, modeled in the scenario as a combination of a user-action (mouse click) and a region on the screen) are three hot spots: the hood, the door, and the background of the car. The user then has one of three options to choose from by clicking on one of the hot spots, or he/she may choose not to make a choice at all. Depending on the choice, either a text explaining the engine features of the car is rendered, a video clip along with an audio showing and explaining the interior of the car is rendered, or the car image disappears, respectively. If the user does not respond within a certain time frame the car image ceases to be rendered. If the user chooses the engine and gets the text object, he/she might then choose to listen to the sound of the engine by making a further interactive selection from the playback area; after the audio, the presentation ends. Figure 6.11a shows the enhanced

temporal model representation of this scenario. Note that, since there is a temporal equality between the end events of "Text" and "Audio 2" objects and is determinate to be at the same time, both events have the same symbol $C_4$. This point will be further explained in section 6.3.4.

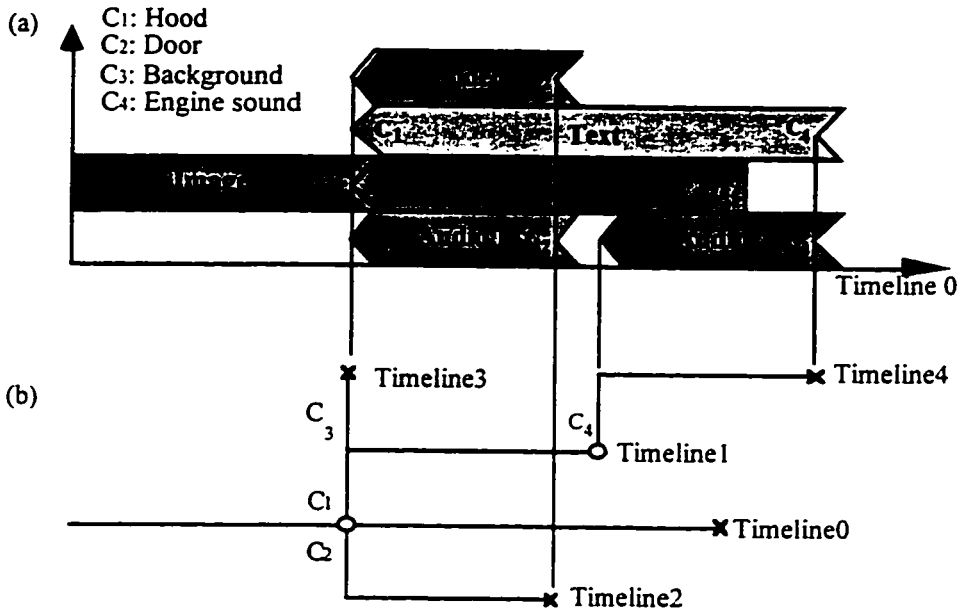



***Figure 6-11 Choices representation in a Timeline-Tree model***

Our timeline-tree model represents interactive scenarios using a tree-like structure. Figure 6.11b shows the tree corresponding to the interactive scenario in 11a. The small circles are abstractions which represent "branches" where user-actions may change the course of the scenario. If choices are not made the current timeline simply plays itself out, otherwise the user may effectively traverse the timeline tree, viewing a "custom" presentation, depending on his/her choices. X's, on the other hand, represent possible ending points of the scenario.

Note that at most one choice, $C_i$, will be selected at a time. Consequently, the presentation-flow will branch to timeline(i). Therefore, we define the *timepath* by the set of timelines the actual presentation follows. For example, if the user in the above scenario selected the hood of the car ($C_1$), the presentation-flow will branch from timeline(0) to timeline(1). Then, if he/she chose to listen to the sound of the engine ($C_4$) the presentation-flow will then branch to timeline(4) until the presentation finishes. The *timepath* as a result will be {0,1,4,X}. All possible *timepath* sets will start with timeline(0) and end with an X which refers to the end of the presentation. Therefore, equations 1 to 3 have to be changed to refer to a certain *timepath* set.

$$time_i(Origin_j) = time_i(C_j) + \sum_{n=i,\, n \in timepath}^{j} \delta_n \quad \text{where i, j} \in timepath \tag{4}$$

$$Minimum\ duration = time_0(X) \qquad X \in timepath \tag{5}$$

$$Length\ of\ the\ actual\ presentation = time_0(X) + \sum_{n \in timepath} \delta_n \quad X \in timepath \tag{6}$$

The shortest duration over all the possible *timepath* sets could be found by examining the timeline tree:

$$Minimum\ duration = time_0(\ closest(X)) \tag{7}$$

In the tree model, the circles represent the times that asynchronous events (corresponding to the symbols shown at the circle) become activated. Those events will be deactivated only when the presentation flow branches to another timeline.

### 6.3.4 Example

The following example, also used by [WAH94] (but without the Logo Graphic), illustrates the use of different timelines. Figure 6.12 shows how an author can create a scenario with the following properties: any slide starts being rendered three seconds before its corresponding talk to allow a silence period to give viewers a first impression of the slide, after a talk is finished the current slide continues to be rendered until a viewer responds with some input. A graphic logo is also rendered for the whole duration of the scenario. Figure 6.12a shows the above scenario representation. The start event of Slide2 has a temporal equality with the end event of Slide1, as does Talk2; thus they all utilize the same symbol, namely $C_1$. The end event of Slide 2, however, has a temporal inequality with its own start event (since it ends interactively), as do Slide1 and Slide3. Thus different symbols are utilized.

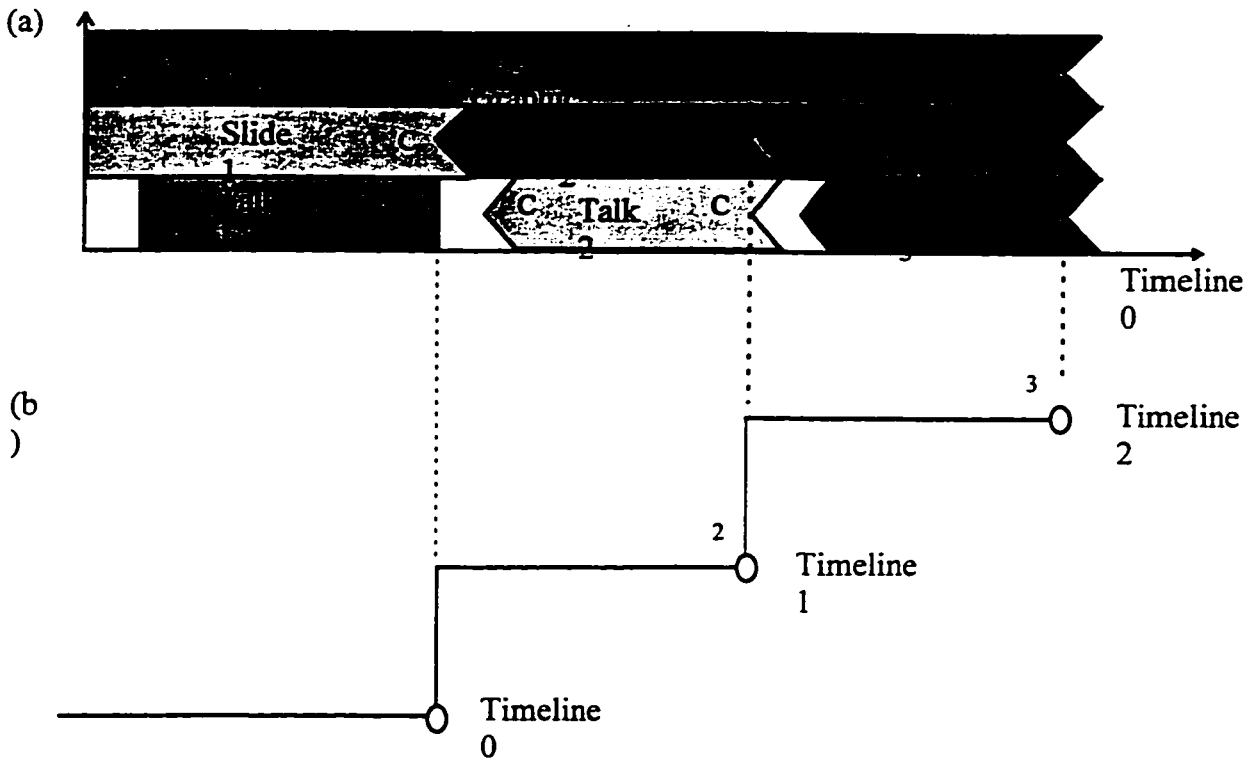



*Figure 6-12 Slide show representation using Timeline-Tree model*

For visualization, it is possible to think of viewing the scenario with respect to a certain timeline from the timeline-tree. Figure 6.13. shows the view of the scenario from timeline1's point of view (after $C_1$ has been made). Thus Slide1 is not shown, nor is Talk1, and the Logo Graphic is depicted as being continued (i.e. already in progress).

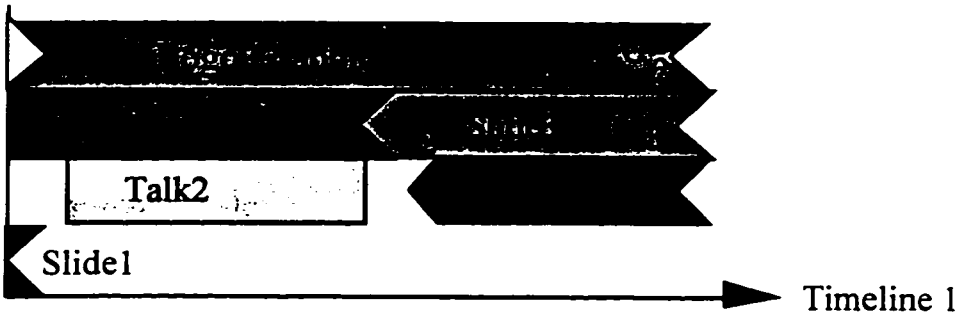



*Figure 6-13 Viewing a specific timeline, timeline(1)*

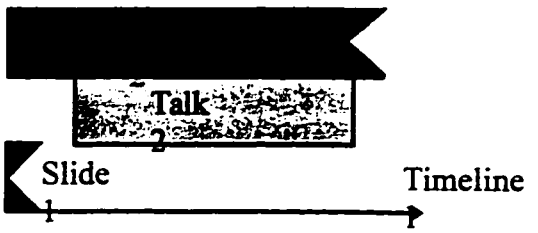



***Figure 6-14 Viewing a specific timeline and specific choice list, timeline(1) AND {C1}- i.e. "what media objects are initiated when action C1 is made?"***

Furthermore, for complex documents that may involve many choices and many user interaction points, it is important to be able to select a subset of the scenario for inspection; for example a specific timeline and a specific set of choices may be inspected visually. Figure 6.14, shows only media objects which are initiated by the action $C_1$.

## 6.4 Related Work

Other work has illustrated a wide range of ways to represent temporal scenarios [EME93][HAR94][ROS93]. In this subsection we examine six of these models.

The reason we adopt the concept of the Timeline temporal model is because, in our opinion, it is simple, graphical, and easy to grasp. This makes them attractive as an authoring tool, particularly to non-technical authors who do not wish to have to learn concepts of programming languages in order to, for example, author documents using scripts. However, the model is not flexible enough to include any indeterminism that results from, for example, user interactions.

In a petri-net model originally proposed in [LIT90], media intervals are represented by "places", relations by "transitions". As shown in Figure 6.15, each of the basic point relations, 'before', 'simultaneous to', and 'after', can be modeled by a transition in conjunction with a delay place δ (or • in the figure). The delay place has a non-negative value which represents an idle time. If the delay is not known at authoring-time, temporal inequalities can be expressed, allowing user interactions. However, unlike the timeline model the graphical nature of the petri-net model can become complex and difficult to grasp when the document becomes relatively large. Figure 6.15 shows a possible petri-net representation of the scenario shown in Figure 6.2.

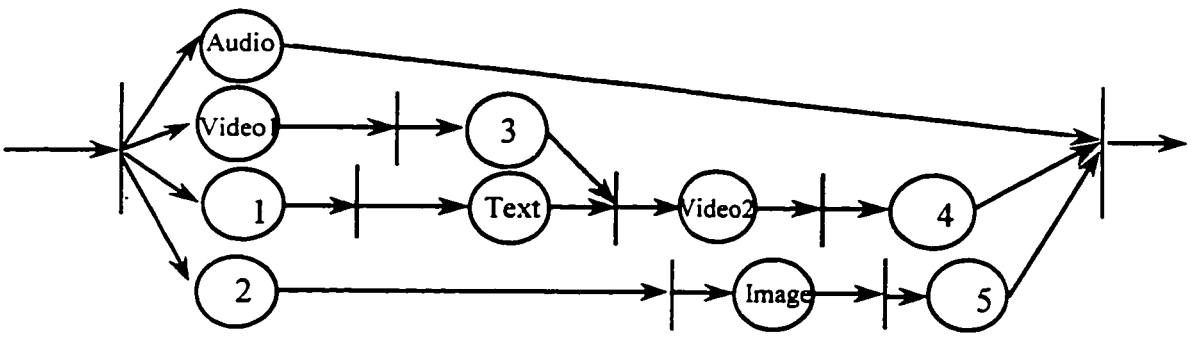



***Figure 6-15 Petri-net model for the scenario shown in Figure 6.2***

The CMIFed multimedia authorer [ROS93] is a tool which provides users with a novel graphical way of visualizing and representing multimedia scenarios. CMIFed offers the traditional timeline-type visualization, called the "channel-view", as shown in Figure 6.16. Synchronizing media objects can be achieved by using CMIFed "sync-arcs". As shown in Figure 6.16, sync-arcs synchronize the start of the audio track to the end of the logo graphic, as well as synchronizing the start of the video to the start of the audio. The hierarchy-view is a novel way of visualizing both the structure of the scenario and the synchronization

information using nested boxes. Boxes that are placed top-to-bottom are executed in sequential order, while those placed in left-to-right order are executed in parallel.

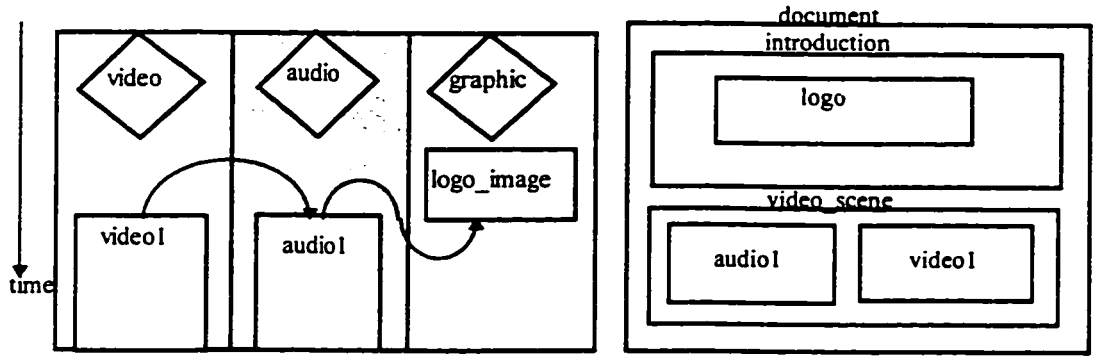



***Figure 6-16 CMIFed channel-view (left) and hierarchy-view (right)***

Buchanan and Zellweger, [BUC92], presented a powerful system, Firefly, that supports and models synchronous as well as asynchronous behaviors. Each media object is modeled by two connected rectangular nodes representing start and end events. Any other event that would be used for synchronization (such as a frame in a video) is called an internal event and represented by a circular node that is placed between the start and end events. In Firefly, asynchronous events contained in a media item are represented by circular nodes that float above the start event. Finally, temporal equalities between events are represented by labeled edges connecting these events. However, in complex interactive documents, we see things start to get messy and hard to trace especially when the edges are labeled with time delays. Figure 6.17 shows the car example, depicted in Figure 6.11, using Firefly temporal view. However, although the data structures in Firefly permit any event in a media item to activate or deactivate asynchronous events, *the way* to represent the lifetime and the activation time of an asynchronous event is not yet clear. Two questions may arise: "Are the four

asynchronous events in the car image activated after the start event?", and, " What is the difference or relationship between 'No response' and 'Background' asynchronous event?" Therefore, the scenario representation shown in the figure is not complete. In other words, unlike our temporal model, the temporal relationships representations between the activation and deactivation of asynchronous events with other events are not supported in their temporal view.

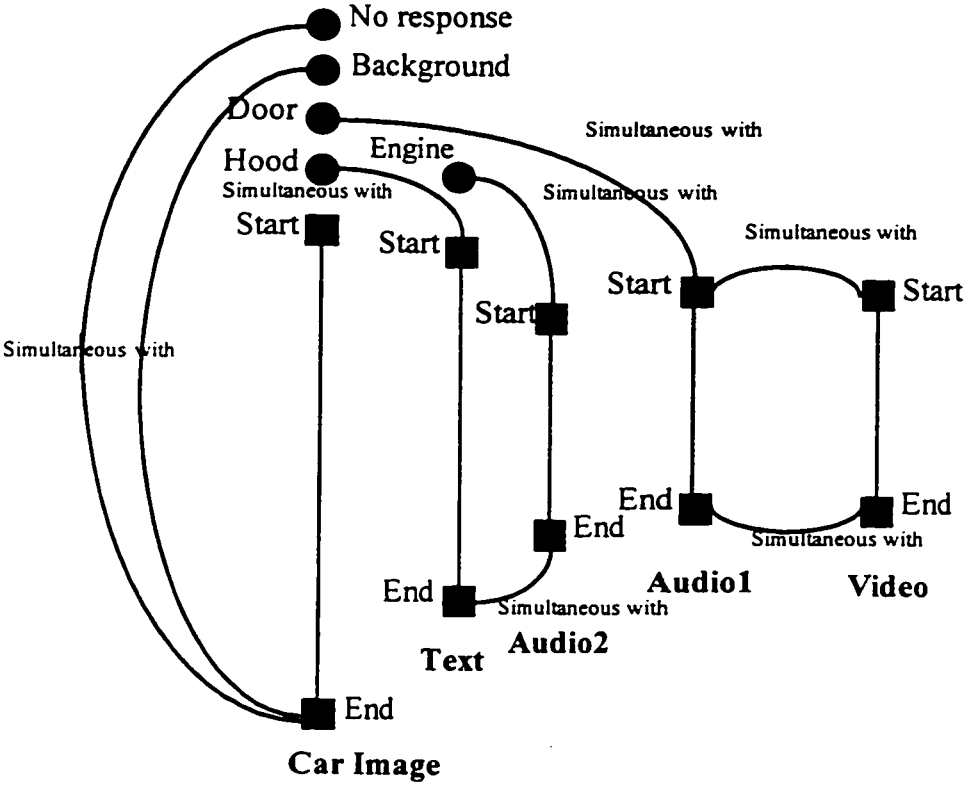



***Figure 6-17 Firefly temporal view of the car example***

Wahl and Rothermel [WAH94] have presented a high-level interval-based temporal model by defining powerful operators that include both temporal equalities and inequalities between events. Each operator is represented by a certain pattern of edges that are labeled with delay values. As before, in complex interactive documents, we see that scenario

representations are hard to trace. In addition, although the operators they define can be applied to asynchronous events (whose start-times are not known in advance but exist in the actual presentation), the operators cannot be applied to those whose start-times may not exist at all in the actual presentation. Figure 6.12 shows a slide show example taken from [WAH94] in which every event *will* definitely occur.

In [WEI94], multimedia documents are automatically presented based on parsing and translation. Grammar-rules map content to the look-and-feel of a spatially and temporally layed-out document. Our system does not address spatial layout but neither does [WEI94] address user interaction with an active document as opposed to a hypermedia (store and forward) document, as ours does. Also our system is concerned mainly with scenarios which are interactive and thus allow more than one temporal play-back while [WEI94] mainly addresses documents with alternate spatial layouts.

## 6.5 Query Specification and the Proposed Temporal Model

As described above, interactive multimedia documents include unpredictable time events, such as user interactions. These events are usually remembered by the viewers and can be used to describe a certain multimedia segment. In section 3, we have proposed a temporal model that is capable of representing interactive multimedia documents scenarios. From the scenario representation using this model, the system can extract information about users' actions. These information can be used as indices and to search for a specific interactive multimedia segment that includes some user actions from the database.

Therefore, to complement the multisegment specification proposed in chapter 3 where we were left out of the external specification, we propose the following syntax on the external specification language that considers these information.

The syntax of the query language on the external specification is as follows

| | | |
|---|---|---|
| **<External_specification>** | ::= | (<event_id><br>[ {<attribute_name> <numeric_op> <value>; } ]<br>[<Text_content>;]<br>[ACTION <action>;]<br>[RESULT <Reaction>] [<object_id>;] |
| **<numeric_op>** | ::= | (=) \| (◡) \| (>) \| (<) \| (<=) \| (>=) |
| **<attribute_name>** | ::= | WHO \| WHEN \| WHERE \| WHAT \| WHY |
| **<Action>** | ::= | MOUSE_BUTTON \| KEYBOARD_BUTTON \| SCREEN_TOUCH \| OTHER |
| **<Reaction>** | ::= | START \| END \| SIMULTANEOUS_TO |

Consider the following example:

*Find the segment where the user named 'Dana',*
*clicks on the left mouse button to start an anonymous image object.*

The syntax of this query would be:

*FIND X IN DATABASE*
*WHERE*
*(EXTERNAL_BEGIN*
*action1*
*ACTION MOUSE_BUTTON-left;*
*RESULT START IMAGE_*;*
*EXTERNAL_END)*

# 7. User Interface for the Multisegment Query Language (MsQL)

## 7.1 Introduction

In chapter 3, we have proposed a Multimedia Query Language that allows the specification of a multimedia content portion (termed a multimedia segment, multisegment for short) not only in terms of its contained media but also in terms of the spatial and temporal relationships that exist among these media objects. Although the language proposed is

powerful, a user might not know how to translate a multisegment description from his/her own words to the syntax of the query language. Thus, a graphical user interface is needed on top of the query language to relieve the user from tedious translation work while constructing the query. In this chapter, we introduce a visual user interface that is developed on top of this query language. Although, the interface contains components that corresponds to the structure of the proposed language, it is intended to be independent and used on top of other query languages for multimedia databases. The main features of the interface are that it is easy to understand by users and accomplishes visual interaction in a user-friendly manner. Therefore, the user may not need to learn a specific query language, instead he/she can use visual methods to describe the information to be retrieved from the database.

The chapter is organized as follows: In section two we study the multimedia query and discuss the various information contained in a multisegment. We then present the query user interface and discuss the Temporal and the Spatial Specification windows, the Media Specification window, and the Result window in section three. Finally in section four we draw our conclusions.

## 7.2 A View At A Multimedia Query

Using the query interface, a user can formulate a query describing the multisegments of interest. This description could be written if the user has previously seen the document. Let us first study the various information types contained in a multisegment..

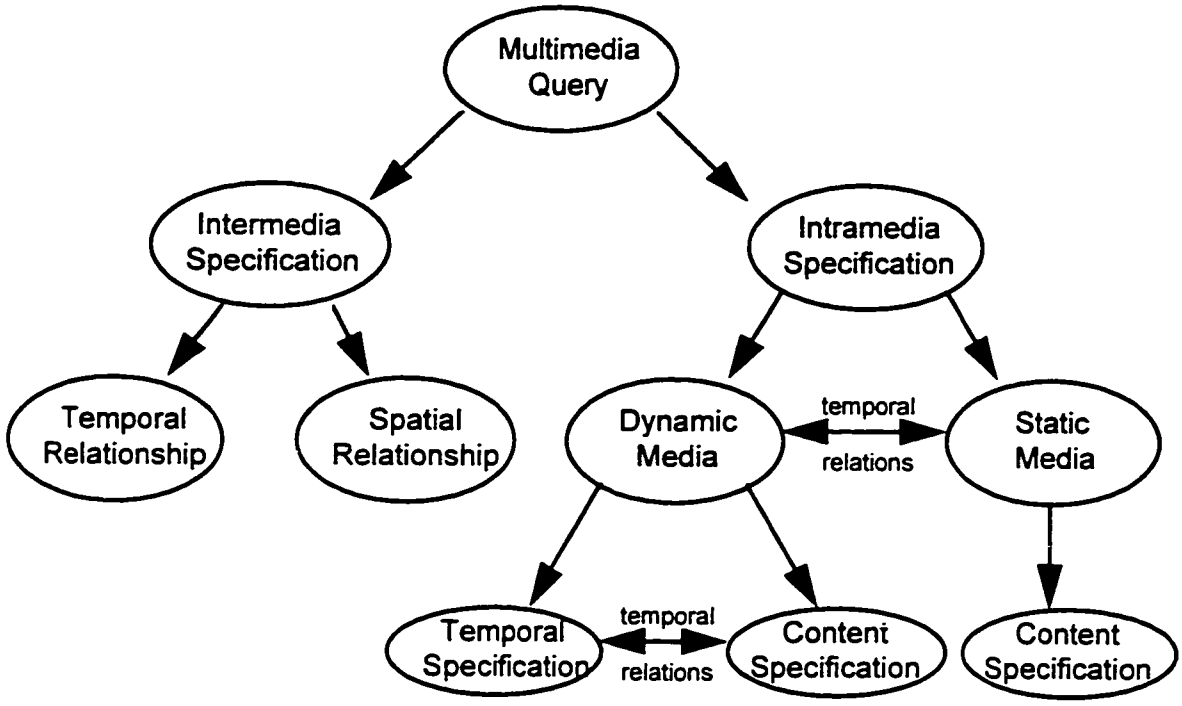



***Figure 7-1 Query decomposition***

Figure 7.1 illustrates how our query interface views a multimedia query. A query is decomposed into two major components: intermedia specification and intramedia specification. The intermedia specification deals with the temporal and spatial relationships among different media types. For instance, how long does a video stream last in terms of seconds and what is the physical location of an image on the display screen. The intramedia specification provides the means to define the contents of a specific media. Based on the properties of the media, it may belong to one of the following two categories: dynamic media and static media. The dynamic media refers to those whose characteristics change with time, such as video and audio. Whereas, the properties of the static media are not functions of time. Image, text, and graphics belong to this category.

From the multimedia information point of view, and as defined in chapter 3, the information can hence be decomposed into three types (Figure 3.1) :

1. Temporal Information that specifies the relationships in time between the various media objects,

2. Spatial Information which specifies the locations of the media objects in space, and

3. Media Information describing the individual media elements composing the multisegment.

Thus, Spatial and Temporal specifications define the structure of the multisegment using spatio-temporal operators. Media Specification describes the media components based on their types.

## 7.3 Query Interface

The interface provides means by which the user can specify the information on the media as well as the temporal and spatial relationships among these media. Relevant multisegments that contain all aspects of the query are retrieved using searching algorithms that are beyond the scope of this chapter. The main window that appears to the user is shown in the figure below.

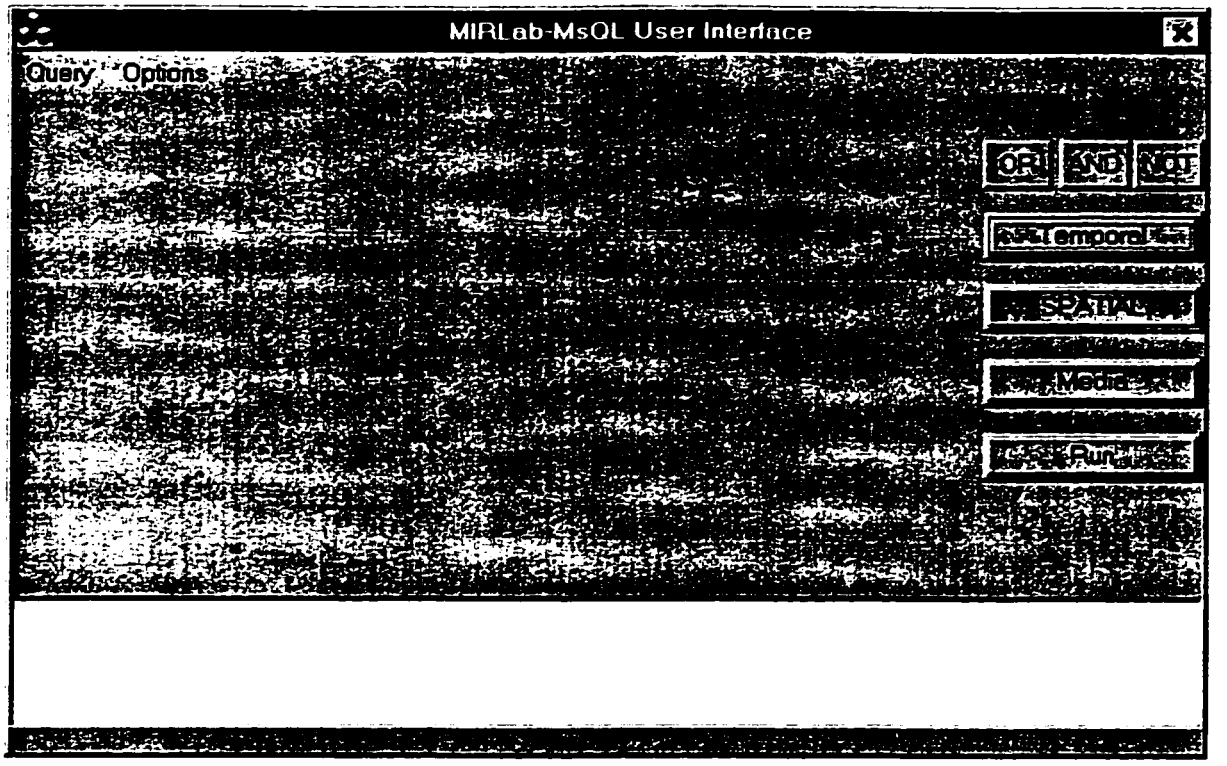



*Figure 7-2 Main Query Window*

### 7.3.1 Intermedia Specification

The temporal specification is used to illustrate the time relationships between the various media objects within a segment. There are many temporal models that represent these relationships. Timeline temporal models, [BLA91]], are simple and easy to grasp. They are attractive to visualize and construct by non-technical as well as technical users. For these reasons, we selected the timeline model to describe the multisegment Temporal information. The temporal model aligns all events (start and end events of media objects) on a single axis which represents time. However, one of its shortcomings is that it can only specify qualitatively the temporal relationship but lacks the ability to show the quantitative

information. The Figure below shows the Multimedia query window, the Temporal Specification window in the middle and the Result window at the bottom.

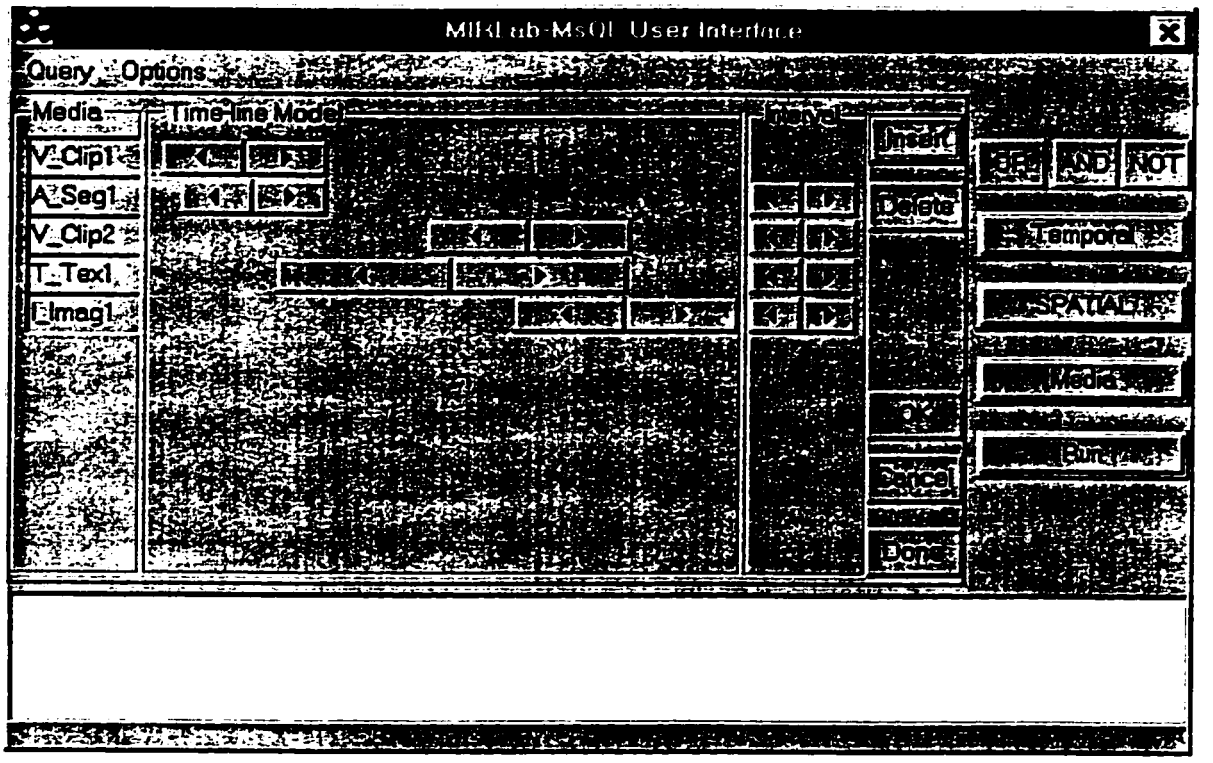



***Figure 7-3 Temporal query window***

From the *Temporal Specification window* (in the middle of the figure), the user can *insert* media representing objects, *delete, set*, and *position* or *relate* selected object(s) in time to other objects. The user can also *display* syntactically the temporal relationships of selected object(s). Using all these options, the user can describe a sub-scenario (using the timeline model) of the multimedia segment of interest to represent the multisegment Temporal Specification.

Another important information that exist among different media types is the spatial relationships. What we are concerned with here is the spatial layout of a media object with

respect to others. In the spatial query window, we use rectangles to represent media objects. The user can place rectangles at different positions in the window to represent different media types. By putting the rectangles on desired positions on the query window and adjusting the size of each rectangle, the user can clearly express the spatial layout in a visual means of the segment to be retrieved. The mechanism in the system to calculate the spatial relationships between each pair of objects is based on the projections of the rectangles on the horizontal and vertical axis, chapter 3, of the display screen.

### 7.3.2 Intramedia specification

As we have explained before, the media can be categorized into static and dynamic media. The static media, such as text and image, contain information that is relatively easier to describe than their counterparts, such as video and audio.

#### 7.3.2.1 Static media

The text deals with alphanumeric data. It is the most commonly used data type by the traditional database systems. There has been a suite of well-developed mechanism related to it. We use the conventional keyword searching techniques to query on text. Though these techniques have some drawbacks, such as keyword mismatching, it is still the most straight forward and widely used method. We will not discuss it in detail in the chapter, instead, we will focus on the image media type.

Although image could be thought of as a static medium, there are lots of visual features associated with it, either in the low-level( color, texture, shape, etc.) or in the semantic level ( visual object in a picture), making the query complicated. It is almost impossible to use query based on keywords to specify the visual features effectively. Our approach is to use visual query methods to retrieve the information in which the user is interested. The contents of image involved in the query refers to the image features obtained by image analysis and processing, either automatically or manually. The features include color, texture, visual objects, and subjects of domains. The system allows the user to construct his query on the basis of these features.

#### 7.3.2.1.1 Query on Images

*Icon-based Query:* we use the icon-based method in the image objects query for the user to construct queries. The iconic query method is considered to be intuitive and user-friendly, since it gives the user the sense as if he is directly interacting with the reality. The user describes the image in terms of the objects it contains and their spatial relationships. Only those salient objects with prominent visual effect in the image give deep impressions to the user. To construct the query, the user unnecessarily needs to see the real image in advance, but having watched it will certainly help him better constructing the query. Using these objects to describe the image, the user can semantically specify his query without difficulties. In the object query window, we have a hierarchy of icons that are classified using the object-oriented paradigm. Each icon maintains a symbolic relationship with the corresponding object in the real world. The user can easily navigate through the hierarchy.

When a category of icons in the hierarchy is selected, the icon instances belonging to that category will automatically be displayed to the user. The user can then choose the icon he/she wants and put it to the appropriate location on the query space. The size of the icon in the query space is adjustable so that the user can specify the spatial information of the object as well. By arranging the relative locations of the selected icons and resizing their sizes in the query window, the user can clearly express the content of the image he is interested in. The user has also the option to pop up a dialog window, named "additional information", associated with each selected icon. It allows him to add more information to the icon, since sometimes the visual icon can not convey the exact object the user is looking for. For example, assume that a user is interested in a scene that contains the Parliament building. Unfortunately, he may not be able to find an icon that represent that object. However, the user can select an icon which represent a building, and then add the word "Parliament" in the dialog box. Hence, we can add more semantic information to the icons.

*Color query*: In the color query window, the user is offered with a color to select the desired color. Once a color is selected, it is displayed in a patch, and the user can adjust it into a more desired degree using a set of R, G, B sliders. The user can divide the query space into different areas in order to assign different colors to each of them to formulate more complex queries. For example, if a user wants to specify the grass land under a sunny sky, what he needs to do is to divide the query space into two parts. The upper area is assigned to blue color, and the lower part is assigned to green. In this approach, the query space can be divided into sub-areas on the user's demand.

#### 7.3.2.2 Dynamic Media

The dynamic media distinguishes itself from the static media with by possessing temporal properties and some other unique features. Video and audio belong to this category. In this chapter, we emphasize on the video.

Video is considered to be the most complicated medium type in terms of its content and properties. It is an "information rich" medium, possessing all the properties contained in image as well as temporal information and hierarchical structure. A video stream can be parsed and segmented into shots. A shot can be further represented by a keyframe or a number of keyframes, depending on the complexity of the shot. The keyframe could be thought of as a static image. Thus, we can use the query methods for image to query on the keyframes. Perhaps the most important video features is the temporal information that includes object movements, camera operations, and the duration of the video. Based on the information embedded in a video clip, we divide the video query into two major steps: video-temporal query and video-content query.

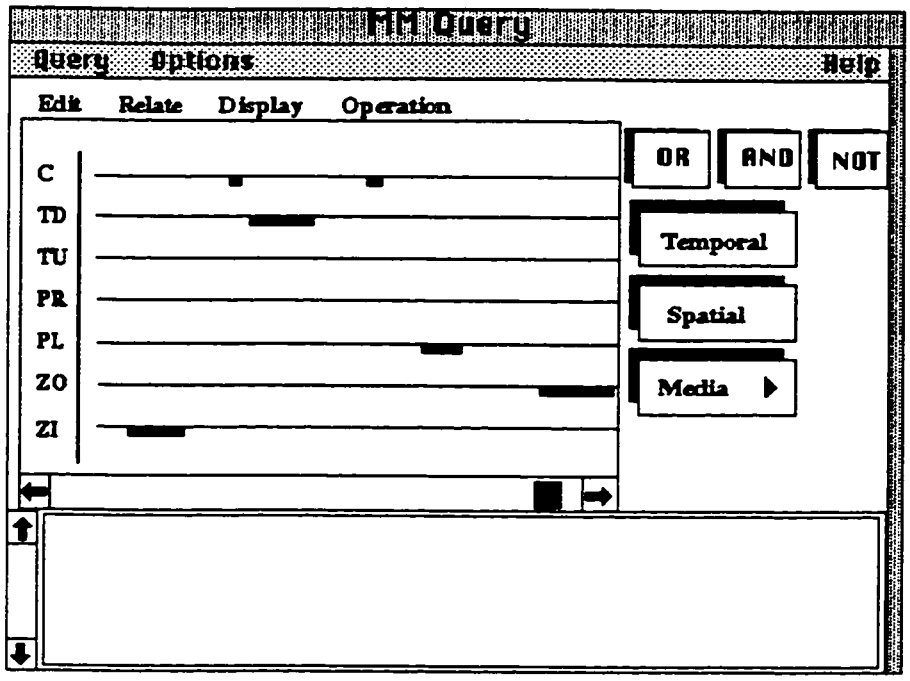



***Figure 7-4 . Video Temporal Specification window (in the middle)***

We introduced a new *Video-Temporal specification* which includes information on camera breaks (cuts), camera operations, speed of the camera, object movements, length of the individual shots, etc. The middle part of Figure 7.5 shows the *Video Temporal Specification window*. Similar to the Temporal Specification, the Video_Temporal information is described using a *Detection Graph*, Chapter 5 section 4. The Detection Graph plots the basic camera operations and cuts (or transitions) versus the frame number. However, in the interface as the frame number is not perceived by users compared to time, we replace the frame number axis with the time axis. The basic camera operations considered are: Zoom-in, Zoom-out, Pan-left, Pan-right, Tilt-up, and Tilt-down. Therefore, instead of inserting media types representing objects as in the Temporal Specification window, the user inserts objects

to represent camera operations and cuts. The user may insert more that one camera operation during the same period to represent a logical OR combination.

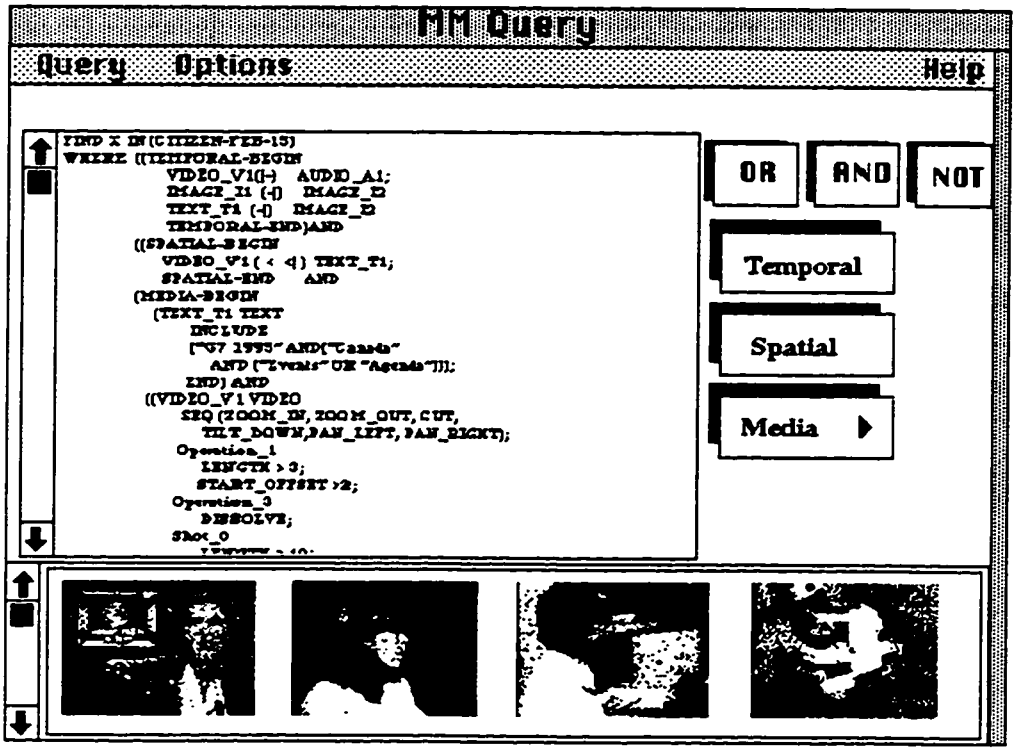



*Figure 7-5 Display of current query syntax & results as shots representatives*

### 7.3.3 Result window

When a user finishes formulating his/her query and selects *Run* option to run the query, multisegments representatives will be displayed in order from the best match to the n'th best match in the Result window (shown at the bottom of the MM Query window) Figure 7.6. If image media type is chosen to represent a multisegment, then a thumbnail size of the image specified by the user and contained in the multisegment (or any image contained in the multisegment) will be displayed. If Video media type is selected then a thumbnail size of the

representative frame of the first shot of the video specified by the user and contained in the multisegment will be displayed instead. Then it is up to the user to select the one he/she refers to. In other words, this approach uses the pattern matching capabilities of the user for selecting the best match multisegment. When the user selects a multisegment representative to view, the corresponding multisegment or the media is played back on the screen.

The bottom window of Figure 7.6 (the Result window) shows few reduced size images that represent the first shot of each video contained in all the resulted multimedia segments. The user can select anyone to display the corresponding multimedia segment if he/she wants. However, the middle window represents the current query syntax that includes the temporal and spatial specifications of the described multimedia segment, as well as the media specifications. Note that the user can select either one or all of the specifications to display syntactically.

Having the auditory data the key for retrieving a multisegment is a challenging area of research. Kageyama and Takashima, [KAG92], proposed a way in which a user can hum a tone and the system retrieves the desired melody using this data. Unfortunately, this technique is inapplicable for other types of data than a melody. Sounds of musical instruments can be specified by the spectrum pattern and the power envelope pattern. When we input the auditory data, the system automatically extracts the spectrum pattern and the power envelope patterns as a media-index, or a sound-index. A user query may be formulated by the use of a microphone, a tape recorder, or a musical instrument. The system

then matches this query with the sound-indexes stored in the database and presents the candidates.

If multisegments representatives are chosen to be of audio type, then the user should understand this presentation to make his/her desired selection. Two methods have been addresses but yet to be devised, [HIR93]. The first method is to visualize the candidates by icons to be displayed in the Result window. The second method is to play back the candidates simultaneously using different channels. The user in the latter method moves closer towards one of the speakers which is judged to be the desired. The former method saves time but the latter is more user-friendly.

# 8. Conclusion and Open Issues

Multimedia databases require retrieval facilities to extract individual multimedia portions from the documents. Retrieval systems require a specification language with which the requested multimedia data are described. In chapter 3, we proposed a multimedia query language that describes the data in terms of their spatial, temporal, and content information. A new video data specification is also proposed in which camera operations and cuts information can be described.

Next, we have described an object-oriented database schema based on the multimedia document structures. A multimedia document can have a logical structure, a temporal structure, and a layout structure. Moreover, the temporal structure can specify the temporal relationships between individual portions of the document representing chapters, sections or

paragraphs, thus, the temporal structure is referred to by the logical-temporal structure. On the other hand, the temporal structure can define the relationships between the media components of the document and hence referred to by the media-temporal structure. The database schema we presented supports these structures and facilitates document browsing and retrieval techniques.

We also proposed a new mechanism to detect cuts between scenes. These cuts may be sharp cuts between scenes or special effect (gradual) cuts, such as dissolves, fade-ins, etc. The problem of selecting an appropriate threshold is relaxed in this mechanism. This is because cut detection is based on the understanding of various changes that might occur from one frame to another. These changes include those resulting from camera operations, object movements, and scene cuts. Basically, the inter-frame difference threshold is selected to be very low, assuring that no cuts would be missed. The algorithm then acts to filter those changes that represent camera operations or object movements. The algorithm also takes the recent inter-frame changes into consideration in making its detection. This confirms the current decision as well as helps in detecting gradual cuts.

The initial tests we performed showed detection rate of over 91% for boundaries between scenes, achieving a rate of over 78% of gradual cut between scenes. However, more work needs to be done for detecting complex camera operations where a combination of two or more of the basic operations considered in this thesis may be combined. For example, if the camera is panning to the left and at the same time zooming in a particular object is

considered a complex camera operation and requires more investigation to increase the cuts and camera operations detection rate.

In chapter 5, we have proposed new ways of using the traditional timeline model to create active multimedia presentations. Our research had two approaches; in the first phase we modeled the user-interaction with the document as a media type with an associated data structure which completely described the interaction, and we represented the interactive scenario as a basic tree of timelines. In the next phase we focused on giving the timeline representation itself more functionality by changing the look and semantics of the media objects placed on the timeline. Media objects whose start and end times were dependent on some user-interaction (and hence could not be placed on the traditional timeline) were distinguished both geometrically (by the rectangle with bent end-edges) and textually (by the text marker placed on the object itself). We feel that modeling scenarios using our timeline approach is superior to doing so using the traditional timeline since we can model fully interactive, transparent user-interactions with the scenario. Thus a much wider and more interesting range of scenarios may be modeled such as interactive books, encyclopedias, and movies, with interaction that is modeled as a part of the scenario.

Our model for defining interactive documents can be used to create these types of documents in many different settings. In our opinion using the model to create interactive instructional course material is probably the best application. Not only can fully synchronized multimedia courses be created but users will not be limited to simply watching the presentation. They may interact with it, browse and explore. Other examples

of useful applications are in creating interactive news articles for multimedia news on-demand or creating interactive entertainment for video-on-demand and interactive cinema. The proposed temporal model can be used to extract external information of the multimedia passage it represents, as well as to create indices.

One of the main problems within today's database systems is the lack of easy ways for the user to specify the complex queries. It results from the difference between the user's means of thinking and the query language embedded in the systems. Our query interface is built on top of the multimedia query language we propose in chapter 3. The user doesn't need to translate what he is thinking about the query into the query language specific to the database system. Instead, we provides the user with a straight forward and easy to use query interface, assisting the user specifying the query on the basis of what he is thinking in his mind. We divide the query specification into intermedia and intramedia query. The intermedia query deals with the temporal and spatial relationships among different media types, and the intramedia query is concerned about the internal content of a specific medium type. By combining different query steps together, the user could accomplish complex queries. Part of the query interface was implemented using MFC library.

Some open issues may lie in introducing more visual methods to support more complicated query, such as motion properties in video streams, as well as developing algorithms for image and video indexing. Audio is considered to be an important component of video. Thus, how to use vocal information to efficiently index the video is also an important and hot area of research.

# 9. Publications


[1] Nael Hirzalla and Ahmed Karmouch, "A Multimedia Query Specification Language," International Workshop on Multimedia Database Management Systems, NY, August 1995.

[2] Nael Hirzalla and Ahmed Karmouch, "A Multimedia Query User Interface" IEEE ICCCE, Montreal Canada, September 1995.

[3] Nael Hirzalla and Ahmed Karmouch, A. "Automatic Cut and Camera Operation Detection for Video", International Conference on Consumer Electronics, June 1995

[4] Nael Hirzalla, Benjamin Falchuk, and Ahmed Karmouch, "A Temporal Model for Interactive Multimedia Scenarios," IEEE Multimedia Mag., Fall 1995

[5] N. Hirzalla, Omar Megzari and Ahmed Karmouch "An Object-Oriented Data Model for Multimedia Databases," IEEE ICECS, Amman, Jordan, December 1995

[6] N. Hirzalla and A. Karmouch, "Detecting cuts by Understanding Camera Operations for Video Indexing" Journal of Visual Languages and Computing, 1995.

[7] N. Hirzalla and A. Karmouch, "A Multimedia Query Specification Language", Chapter in a book on Multimedia Database Management Systems, to be published by Kluwer Academic Publishers.

[8] N. Hirzalla and Ahmed Karmouch "An Object-Oriented Data Model and a Query Language for Multimedia Databases," submitted for review in Multimedia Systems, ACM Press.

# 10. References


[AKU92] A.Akutsu et al. (1992) Video Indexing Using Motion Vectors. *Proc. SPIE Visual Comm. and Image Processing,* SPIE, Bellingham, Wash., pp. 1,522-1,530.'

[ALL83] J.F. Allen, "Maintaining Knowledge about Temporal Intervals", Comm. ACM Vol. 26, 1993, pp 832-843.

[ALL93] J.F. Allen, "Maintaining Knowledge about Temporal Intervals", Comm. ACM Vol. 26, 1993, pp 832-843.

[BAR93a] R.Barber, W.Equitz, C.Faloutsos, M.Flickner, W. Niblack, O.Detovic, and P.Yanker, "Query by content for large on-line image collections," Research report RJ9408 (82660) June 1993, IBM Research Division, NY.

[BAR93b] R.Barber, W.Equitz, M.Flickner, W. Niblack, D.Petovic, and P.Yanker, "Efficient query by image content for very large image databases," Compcon Spring '93, San Francisco, California, Feb 1993, pp 17-20.

[BAR93c] W. Niblack, R.Barber, W.Equitz, C.Faloutsos, M.Flickner, O.Detovic, and P.Yanker, "The QBIC project: Querying images by content using color, texture, and shape" SPIE proc. Vol. 1908, pp 173-186, 1993

[BIM93] A.Bimbo, M.Campanai, P. Nesi, "A three-dimensional Iconic environment for image database querying," IEEE trans. on Soft. Eng. Vol. 19, No 20, Oct 1993 pp.997-1011.

[BIN93] Binbo, A., Campanai, M. , and Nesi, P. "A Three-Dimensional Iconic Environment for Image Database Querying", IEEE Transactions on Software Engineering, Vol.19, No.20, October 1993

[BLA91] G.Blakowski, J.Huebel, and U.Langrehr, "Tools for specifying and executing synchronized multimedia presentations," 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video, Nov. 1991

[BUC92] M.Buchanan and P.Zellweger, "Specifying temporal behavior in hypermedia documents." Proceedings of the ACM Conference on Hypertext, ACM Press, NY, Dec. 1992, pp.262-271.

[CAK93] D. Cakmakov, D. Davcev (1993) Experiments in Retrieval of Mineral Information, First ACM International Conference of Multimedia, Anaheim, Calif, pp. 57-64. '

[CHE96] J. Cheng and A. Karmouch, "A Query Interface for Multimedia Databases," to appear in the International Conference on Consumer Electronics'96, Chicago, Illinois, June 1996.

[EME93] J. Emery and A.Karmouch, "A Multimedia Document Architecture and Rendering Synchronization Scheme", *Proc. of 2nd Int. Conf. on Broadband Islands*, June 1993, pp59-69

[GIB91] Gibbs, S. "Composite Multimedia and Active Objects," Proc. Of OOPSLA'91, pp. 97-112, 1991.

[HAL91] J. Halpern and Y. Shoham, "A propositional model logic of time intervals," J. Ass. Comput. Mach., vol. 38, pp. 935-962, Oct. 1991'

[HAR94] L. Hardman, D. Bulterman, G.V. Rossum, "The Amsterdam Hypermedia Model", *Comm. of the ACM*, Vol.37, No. 2, Feb. 1994

[HIK94] N.Hirzalla and A.Karmouch "Multimedia Content Retrieval System." Technical report, Electrical Eng. Dept., University of Ottawa, Nov. 1994

[HIR92] K.Hirata and T.Kato, "Query by visual example" Advances in Database Techn. EDBT'92, Vienna, Austria, March 1992.

[HIR93] K.Hirata, Y. Hara, N.Shibata, and F.Hirabayashi, "Media-based navigation for hypermedia system," Proc. Hypertext'93, November 1993 pp.159-173.

[HIR95a] Nael Hirzalla and Ahmed Karmouch, "A Multimedia Query User Interface" IEEE ICCCE, Montreal Canada, September 1995.

[HIR95b] Hirzalla, N and Karmouch, A. "Automatic Cut and Camera Operation Detection for Video", International Conference on Consumer Electronics, June 1995

[JOS88] Joseph, T. and Cardenas, F. "PICQUERY: A High Level Language for Pictorial Database Management Systems", IEEE Transactions on Software Engineering, Vol.14, No.5, May 1988

[KAG92] T. Kageyama, and Y. Takashima, "Music retrieval with hummed melody," 8th Symposium on Human Interface, 1992, pp.195-200

[KAR96] A. Karmouch and J. Emery, "A Playback Schedule Model for Multimedia Documents," IEEE Multimedia,. Spring 1996, pp.50-61.

[KAU94] S.C. Kau and J.Tseng, "MQL-- A Query Language for Multimedia Database," Second ACM International Conference on Multimedia, 1994, ACM Press, pp 511-516.

[LIT90] T.D.C. Little and A. Ghafoor, "Synchronisation and storage Models for Multimedia Objects," IEEE JSAC 8(3):413-427, Mar 1990.

[LUC93] D. Lucarella, S. Parisotto, and A. Zanzi, "MORE: Multimedia Object Retrieval Environment," Proc. Hypertext'93, November 1993 pp.39-50.

[MIY88] M.Miyahara and Y.Yoshida (1988) Mathematical Transform of RGB Color Data to Munsell (HVC) Color Data. *Proc. SPIE Visual Comm. and Image Processing,* SPIE, Bellingham, Was. Vol 1001, pp 650-657.

[MUM87] D.Mumford, "The problem with robust shape descriptions," First Int. Conf. On Coput. Vis. pp 602-606, London, England, June 1987.

[NAG92] A. Nagasaka and Y. Tanaka (1992) Automatic Video Indexing and Full-Video Search for Object Appearance. In: *Visual Database Systems*, Vol. II, Amesterdam, pp. 113-127.

[OTS91] K. Otsuji, Y. Tonomura, and Y. Ohba (1991) Video Browsing Using Brightness Data. *Proc. SPIE Visual Comm. and Image Processing 91*, SPIE Bellingham, Washington. Vol. 1606, pp 980-989.

[OTS93] K. Otsuji, Y. Tonomura (1993) Projection Detection Filter for Video Cut Detection. *Proc. ACM Multimedia*, ACM Press, NY, pp 251-257.

[ROS93] R. Rossum, J. Jansen, K. Mullender, D. Bulterman, "CMIFed: A Presentation Environment for Portable Multimedia Documents", *Proc. of ACM Multimedia '93*, June 1993

[ROU88] Roussopoulos, N. and Faloutsos C. "An Efficient Pictorial Database System for PSQL", IEEE Transactions on Software Engineering, Vol.14, No.5, May 1988

[SAL93] G. Salton, J. Allan, C. Buckly (1993) Approaches to Passage Retrieval in Full Text Information Systems, Proc. of Sixteenth Int. ACM/SIGIR Conference on Research and Development in Information Retrieval, 1993.'

[SMO94] S.Somoliar, and H.Zhang, "Content-based video indexing and retrieval," IEEE Multimedia, Summer 1994, pp.62-72.

[TON91] Y.Tonomura (1991) Video handling based on structured information for hypermedia systems. *ACM Proc. of Int. Conf. on Multimedia Information Systems*, ACM Press. pp. 333-344.

[TON94] Y. Tonomura, A. Akutsu, Y. Taniguchi, and G. Suzuki (1994) Structured Video Computing. *IEEE Multimedia* Vol. 1, No. 3. Fall 94, pp. 34-43.

[VIL86] M.Vilain and H.A Kautz, "Constraint Propagation Algorithms for Temporal Reasoning", AAAI-86 Philadilphia, 1986, pp132-144'

[WAH94] T.Wahl and K. Rothermel, "Representing Time in Multimedia Systems," *Proc. of Int. Conf. on MM Comp. Sys.*, 1994, pp538-543.

[WEI94] L.Weitzman and K. Wittenburg, "Automatic Presentation of Multimedia Documents Using Relational Grammars", *Proc. of ACM Multimedia '94*, November 1994

[WOE86] D. Woelk, W. Kim, and W. Luther, "An Object-Oriented Approach to Multimedia Databases", Proc. of the ACM-SIGMOD 1986 Conf. on Management of Data (Washington, DC, May 1986), ed. C. Zaniolo, SIGMOD Record, Vol. 15, No 2, pp. 311-325.

[ZHA93] H. Zhang, A. Kankanhaalli, and S. Smoliar (1993) Automatic Partitioning of Full-Motion video. *ACM Multimedia Systems*, ACM Press, NY, Vol. 1, No 1, pp 10-28.'

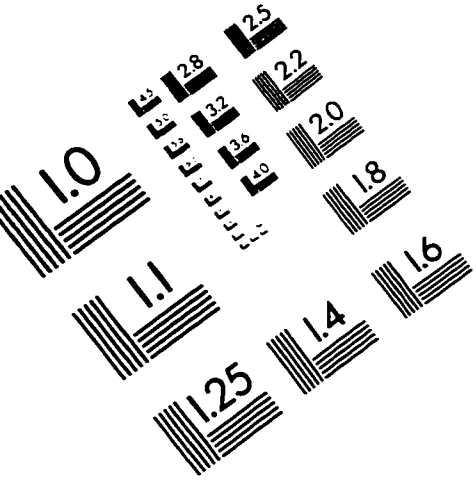
[ZHA94] H. Zhang and S. Smoliar (1994) Developing Power Tools for Video Indexing and Retrieval. *Proc. IS&T/SPIE Symp. Electronic Imaging: Science and Technology,* pp. 140-148.

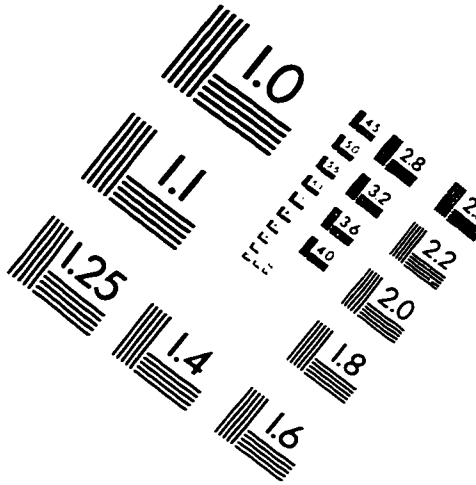
[ZHA94b] H. Zhang, Y.Gong, S. Smoliar, and S. Tan, (1994) Automatic Parsing of News Video. *Int conf. on Multimedia Computing and Systems,* IEEE Computer Society Press, pp. 45-54.

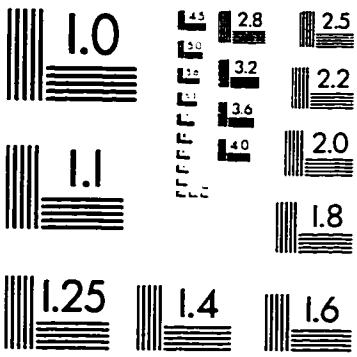
[ZHA94c] H. Zhang, S. Smoliar, and Y. Gong, (1994) Video Parsing Using Compressed Data. *Proc. SPIE Conf. on Image and Video Processing II,* San Diego.'

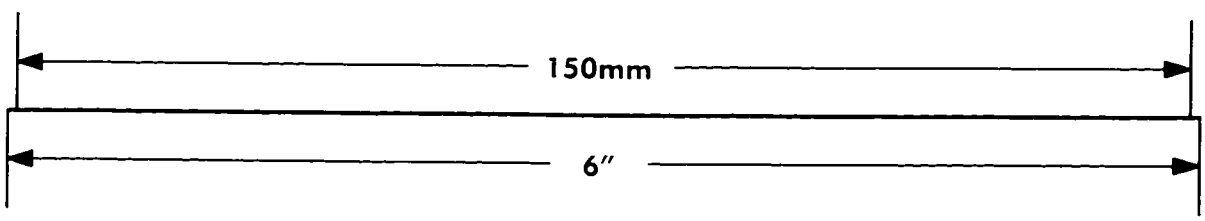
[ZUE91] V.W. Zue, "From signals to symbols to meaning: On machine understanding of spoken language." Proc. of the 12th Int. Congress of Phonetic Sciences, 1991.

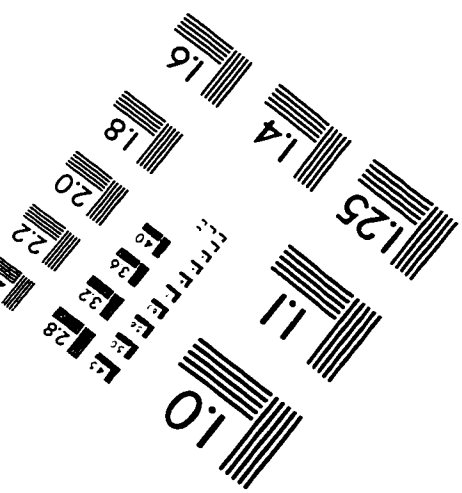
# IMAGE EVALUATION TEST TARGET (QA-3)

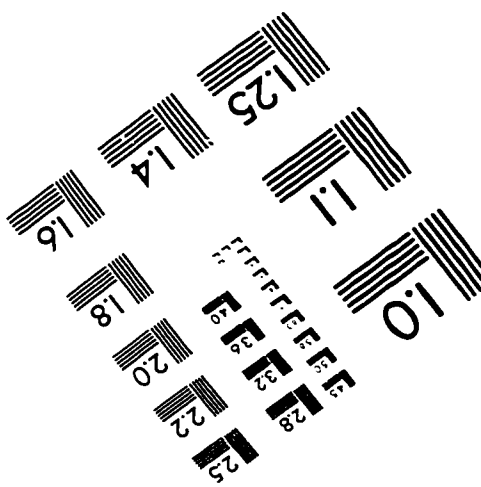



APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989