University of Alberta

Query Languages in Multimedia Database Systems

by

John Z. Li, M. Tamer Özsu, Duane Szafron
Laboratory for Database Systems Research
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada T6G 2H1
{zhong,ozsu,duane}@cs.ualberta.ca

Technical Report TR 95–25
December 1995

DEPARTMENT OF COMPUTING SCIENCE
The University of Alberta
Edmonton, Alberta, Canada

Query Languages in Multimedia Database Systems

December 26, 1995

Abstract

Declarative query languages are an important feature of database management systems and have played an important role in their success. As database management technology enters the multimedia information system area, the availability of special-purpose query languages for multimedia applications will be equally important. In this report[1], we survey multimedia query languages and query models. Particularly, we look at those systems from the point of view of well-defined queries, fuzzy queries, visual queries, and query presentations. Several research issues, such as generic multimedia query languages, incremental queries, fuzzy queries, spatio-temporal queries, feature storage and organization, are also identified. In our opinion these are vital issues for the success and development of a multimedia query language.

[1]This research has been supported by a grant from the Canadian Institute for Telecommunication Research (CITR), a Federal Network of Cetre of Excellence funded by the Government of Canada.

Contents

List of Figures

List of Tables

1 Introduction

In the past few years, we have seen a growing trend in the computer industry to provide support for multimedia data. Many of the new computer applications involve multimedia data. Although multimedia has been a buzzword for quite a few years, it is actually difficult to give it a formal definition. In fact, in the most general setting, *multimedia data* could mean arbitrary data types and data from arbitrary sources [Kim95]. These data include, in addition to traditional data types like numeric data and character strings, graphics, images, audio and video.

Multimedia query languages are important since query languages are an integral feature of database management systems (DBMS). We focus on the latest development of many multimedia query languages and query models. In particular, we discuss how those systems deal with the spatial and temporal relationships among multimedia data, uncertainties in exact-match and similarity-match queries, and the presentation and synchronization issues for the delivery of queries. We also discuss some pending research issues, including the generality of multimedia query languages, incremental queries, fuzzy queries, spatio-temporal queries, feature classification and organization. These issues are vital to the success of a multimedia query language.

The report is organized as follows. The rest of this section provides an introduction to multimedia database systems and multimedia queries. We give the motivation for having multimedia query languages and the basic features of a multimedia language. In Section 2 we discuss different functionalities and facilities provided by different systems. Query languages are classified into four categories: well-defined, fuzzy, visual, and presentational. Section 3 contains our ideas about what are the most important research problems in multimedia query languages at the present time. We give our conclusions in the last section.

1.1 Why Database Techniques Are Necessary

Many multimedia systems or media servers are implemented using *multimedia file systems*. These systems leave to the user the responsibility of formatting the file for multimedia objects as well as the management of large amounts of data. The development of multimedia computing systems can benefit from traditional DBMS services such as data independence (data abstraction), high-level access through query languages, application neutrality (openness), controlled multi-user access (concurrency control), fault tolerance (transactions, recovery), and authorization (access control).

Another important reason for using a DBMS is that the existence of temporal and spatial relationships and the related data output synchronization requirements complicate data management and delivery. These

relationships need to be modeled explicitly as part of the stored data. Thus, even if the multimedia data are stored in files, their relationships need to be stored as part of the meta-information in a DBMS.

An object-oriented approach is an elegant basis for addressing all data modeling requirements of multimedia applications [WK87, CK95]. It accommodates various data types while providing a uniform interface to access them.

1.2 Multimedia Query Languages in DBMSs

One of the basic functionalities of a DBMS is to be able to process declarative user queries. This is achieved by defining a query language as part of the DBMS. The complex spatial and temporal relationships inherited in the wide range of multimedia data types make a multimedia query language quite different from its counterpart in traditional database systems. Many have proposed object-oriented technology as a promising tool for dealing with multimedia data. As a result almost all multimedia database systems are directly or indirectly (by extending relational models into object-oriented models) based on object-oriented technology.

A powerful query language significantly helps users to manipulate a multimedia DBMS. It also helps to maintain the desired independence between the database and the application. Effective query languages must be user friendly for both naive and expert users. This is especially important in multimedia systems because of the many different kinds of media and the large volume of data. Furthermore, the presentation of query results is another key issue in multimedia query languages since it usually requires the synchronization of continuous media.

The query languages of traditional DBMSs only deal with exact-match queries on conventional data types. This might be sufficient to deal with queries posed against metadata and annotations of multimedia data. These queries are definitely important. However, content-based information retrieval requires non-exact-match (fuzzy) queries which go beyond the traditional approaches. Further research is necessary to deal with fuzzy queries. Some research [HK95, PS95b] has been done in supporting multimedia content specification and retrieval in the design of a multimedia query language.

There is a common misconception that WWW browsers, such as Netscape or Mosaic, are sufficient to query multimedia repositories available on the Internet. A browser enables the user to look for useful data, but the user has to know where the data is located. In order to facilitate the search, many search engines have been created, such as Yahoo, Lycos, WebCrawler etc. These search engines occasionally scan the WWW and construct indexes of interesting keywords. These indexes are useful in locating information by using browsers or other automatic tools. However, they are fundamentally different from

query languages. This is because in WWW systems the arrangement of multimedia objects is *static*. Documents are generated and linked together before a user queries them and the whole WWW system is based on a structureless Internet organization. A browser usually treats multimedia data as Binary Large OBjects (BLOB). Therefore, content-based information retrieval is not possible. Furthermore, a user is not allowed to issue sophisticated queries because queries are restricted to certain patterns. On the other hand, database query systems usually deal with multimedia objects in a dynamic way in the sense that objects are not hard-wired to other objects by static hyperlinks.

2 Multimedia Query Models and Languages

Retrieving multimedia information requires powerful query languages that support semantic data retrieval. That is, information retrieved from multimedia databases should be obtained not only by means of keyword search or indexes on keywords, but also by the contents of multimedia objects. In terms of properties, queries can be classified into two general categories: *well-defined* and *fuzzy*. A query is *well-defined* if the properties of objects being queried belong to some well defined set of labels, and the conditional operators are also well defined. In these queries, users have perfect knowledge of the underlying database and what they want from the system. They want their queries to be executed with exact matches. The query "*Show me all the video clips from the G7 Summit'95 database in which president Clinton appears*" is a good example of a well-defined query.

A query is *fuzzy* if the properties of objects being queried cannot be certain (like *grey hair*) or the comparison operators in the query cannot provide exact matches. For example, the query "*What are the most interesting places in Canada*" is a fuzzy query. If the system has to derive all the interesting places from other features, the result is uncertain since different people may have different opinions about interesting places. A person who likes hiking may interpret interesting places differently than a person who likes camping. Although novice users are a source for fuzzy queries, knowledgeable users may use fuzzy queries too. "*Find all the pictures* `like` *this one*". The operator `like` is a fuzzy operator because it does not require exact matches.

Three different multimedia query methods have been studied: keyword querying, visual querying, and semantic querying. Usually keyword querying only handles well-defined queries. Both visual querying and semantic querying are designed primarily to deal with fuzzy queries although they can also be used in handling well-defined queries. In order to aid users' access and retrieval, some systems incorporate a combination of the above methods.

Keyword querying is the easiest method. It requires each object to carry some tags (or keywords) and entire objects (BLOBs) are retrieved. For any given video or audio, we can store many keywords to accurately describe the content of the video or audio. For example, if we have a video describing the interesting traveling sites of Canada, then *Canada, Rocky Mountains, Jasper, Banff, Toronto, Vancouver, Montreal,* etc. could be the keywords. It is very efficient to do keyword searching and matching. Most of the searching techniques in multimedia databases are based on the use of keywords associated with the images or the video segments [LPZ93, OT93]. Part of the reason for using keyword querying is that such queries can be easily formulated using a standard query language, such as SQL.

Two major drawbacks of keyword querying are the limitation of keywords and information loss. In many cases there are no proper words to describe an image or video, so we are restricted by the keywords and our knowledge of a language (say English). Alternately, there may be too many words to describe an object and there may be no consensus as to which are better or the best. The second problem is information loss. Keywords cannot preserve all the spatial and temporal relationships, which are critical, among the media, such as the shape of an object or the space an object occupies in an image (the old saying "A picture is worth a thousand words" is relevant to this context). Furthermore, it may not be possible to enumerate all the features in a picture. The unavoidable information loss may be intolerable.

Despite these disadvantages, keyword querying is still present in most systems. It is usually coupled with other query facilities because of the simplicity and efficiency of keyword retrieving.

Visual query languages are helpful for users to formalize complicated queries. Querying by pictorial example is a natural extension of relational database's query by example. The query system provides some typical samples (say pictures, video frames, or audio sampling) extracted from the database. A user is asked to select one of those samples and then the system will try to retrieve all the possible matches from the database. A typical system is Query By Image Content (QBIC) [NBE+93], which is an image management system. Another form of visual querying is querying by icon [BCN93]. In an iconic query system, the sample objects are simplified to very few objects. For example, an image icon represents just a car; a video icon represents just a person; an audio icon represents a period of lullaby. For this type of querying, the underlying system must be able to support common features in multimedia data, such as shape, texture, size, and color. It is possible to express spatial and temporal relationships in query by pictorial example. However, visual query systems are usually domain-dependent because of the difficulty of generalizing sample pictures and icons.

Semantic querying (or content-based querying) is the most challenging approach of all, in terms of indexing, pattern matching or searching, and its access structure [HK95]. The presentation of information

from a database requires inclusion of semantics, either explicitly or implicitly [Jai92]. Explicit semantics can be introduced by declarative knowledge representation techniques. We may also specify procedures to assign semantics to data. The semantics or content of an object is usually expressed by its features, which include the object's attributes and relationships between objects. Therefore, the functionalities of such features depend on many techniques of multimedia data processing. These techniques include image processing, pattern recognition, speech recognition, motion detection etc., which are being studied broadly by many researchers. One of major goals of those techniques is to make accurate feature extraction from input data. In an image database, for example, semantic querying allows images to be retrieved by a variety of image content descriptors (both spatial and content information) including color, texture, and shape. These attributes may describe not only the image as a whole but also the individual objects in it. It also offers the opportunity to pose virtually an unlimited set of queries rather than having the system automatically classify and organize sample queries into a small number of predefined classes.

Another type of feature in multimedia data is *metadata*, i.e., the data about data. Metadata has received considerable attention in multimedia research [KS94, JH94, AS94]. Typical metadata are media type description, image size, playing speed, service quality requirements, statistical data (e.g., how frequently has this type of query been asked, how many images include this kind of car etc.). Metadata assumes more importance when managing multimedia data than when managing traditional structured data [KS94]. Some of the reasons are the inability to do exact matches in many cases, the inability to do content-based search in some cases with large data sets that are hard to analyze, and the greater role of derived data, interpreted data, context, as well as semantics when dealing with audio-visual data. For example, *content-descriptive metadata* [BR94] is ubiquitous in multimedia systems. This metadata is determined intellectually or by means of semi-automatic or automatic methods. In the last two cases, these methods are media-type-specific. Examples of content-descriptive metadata are a list of persons or institutions having some relation to a particular multimedia document's content.

2.1 Well-defined Queries

Exact match queries are ubiquitous in traditional DBMSs. Since multimedia database systems must also support traditional DBMS functionalities, multimedia query languages have to accommodate well-defined queries. We will use Figure 1 as our running example throughout the paper. In Figure 1, US President Bill Clinton is in the middle of the image, Canadian Prime Minister Jean Chretien is at the left, and British Prime Minister John Major is at the right. Figure 2 shows a data definition of an image in a C++-like format. The well-defined query in this subsection is "*Is there any image such that Jean Chretien is at the*

left of Bill Clinton".

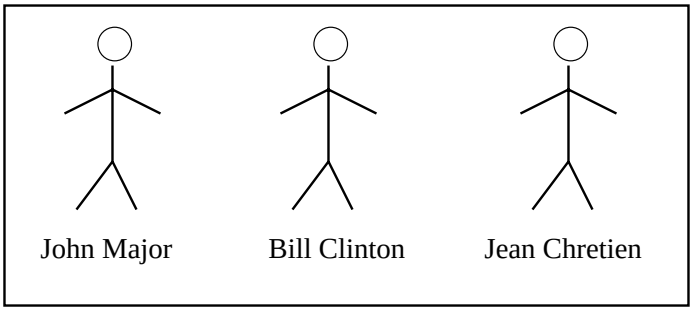

Figure 1: People in the G7 Summit

EVA [DG92] is one of the early query languages designated for multimedia information systems. Based on functional language features, EVA has its roots in conventional set theory. It is formally defined using the mathematical framework of many sorted algebra. In EVA the sample query can be expressed as follows:

{ (title(PIC) | (Image(PIC) **and exists** PERSON PERSON2 **and**
PERSON **isin** object(feature(PIC)) **and**
PERSON2 **isin** object(feature(PIC)) **and**
name(PERSON) **is** "Bill Clinton" **and**
name(PERSON2) **is** "Jean Chretien" **and**
left(PERSON2, PERSON))}

where left(OBJ, OBJ2) is a feature predicate which detects whether object *OBJ* is at the left of object *OBJ2*. **isin** and **is** are built-in functions. EVA is object-oriented and supports objects, classes, and relationships between objects. It also supports set operations like **isin**, **union**, **intersection** etc., which are useful for many well-defined queries. However, EVA does not support any spatial queries (see next section) and currently does not support video data.

Garlic is an object-oriented multimedia middleware and object-based management system developed at IBM Almaden Research Center [CHN+95]. Query By Image Content (QBIC), which is an image retrieval system (see Section 2.3 for details), has been successfully integrated into Garlic. The Garlic data model is a variant of the ODMG-93 object model [Cat94]. Query processing and data manipulation services, especially for queries where the target data resides in more than one place, are provided by the Garlic Query Services and Runtime System. Queries are broken into pieces, each of which can be handled by a single media manager. Garlic extends standard SQL with additional constructs for traversing paths composed of inter-object relationships, for querying collection-valued attributes of objects, and for invoking methods

```
class Person {
    public:
        char sin[9]; // Social Insurance Number
        char name[40];
        char address[80];
        date birthdate; // suppose "date" is already defined
        char sex;
        int age();
}
class Image {
    public:
        char title[40];
        int format; // format of the image: bitmap, GIF, JPEG, TIFF etc.
        int resolution;
        date date; // creation date
        ImageFile *content; // a pointer pointed to the image file
        Feature feature; // features extracted from an image
        void compress();
        void resize();
        void move();
        void display();
        void hide();
}
class Feature { // only some spatial features
    public:
        CollectionOfObject object; // interesting objects as the features
        Keyword keyword;
        Boolean left();
        Boolean front();
        Color color();
}
```

Figure 2: Query Data Definition

with queries. Most of these are actually adopted from ODMG-93 OQL (Object Query Language). The same query is expressed in Garlic's SQL as follows:

select PIC.title
from Image PIC, Person PERSON PERSON2
where PIC.feature.object(PERSON) **and**
PIC.feature.object(PERSON2) **and**
PERSON.name("Bill Clinton") **and**
PERSON2.name("Jean Chretien") **and**
PIC.feature.left(PERSON2, PERSON)

Since the Garlic query language is intended for querying databases that contain data in a variety of repositories, including multimedia repositories with associative search capabilities, Garlic's SQL extension must also take into account the needs of such repositories.

OMEGA (Object-Oriented Multimedia Database Environment for General Application) [Mas91] is an object-oriented database system for managing multimedia data that is under development at the University of Library and Information Science, Japan. In this model an *acceptor* is defined to allow a user to communicate with the system. Such an acceptor can recognize and describe a real world entity by using any symbol system that is allowed in specified media. A collection of designer's impression about the real world in terms of objects is organized as a *kernel object base* for integration. This kernel object base is used as a glue to integrate various conceptual models in different media. An object is defined by three properties: *identifier, protocol*, and *state*. Some extensions are made in OMEGA for representing part hierarchies and temporal and spatial information about multimedia data. OMEGA has some facility for supporting three types of user language interfaces: an object-oriented interface, an SQL-like interface and a Graphic interface. Syntactically the query example expressed in OMEGA's SQL is very similar to Garlic's SQL.

Marcus and Subrahmanian [MS93] have proposed a formal theoretical framework for characterizing multimedia information systems. It also provides a logical query language that integrates diverse media. This is a first attempt at mathematically characterizing multimedia database systems. The model is independent of any specific application domain and provides the possibility of uniformly incorporating both query languages and access methods based on multimedia index structures. A special data structure, called a *frame*, is defined in this model for data accessing. The corresponding query language is called *Frame-Based*

Query Language (FBQL). Under FBQL, the example query is:

(∃ IMG) Image(IMG), Person(PERSON), Person(PERSON2)
name(PERSON)="Bill Clinton" & name(PERSON2)="Jean Chretien"
PERSON ∈ IMG.feature.object() & PERSON2 ∈ IMG.feature.object() &
IMG.feature.left(PERSON2, PERSON))

A similar approach has been used in AVIS (Advanced Video Information System [ACC+95]). A video database is defined by a 9-tuple in AVIS

$$(RVD, OBJ, EVT, \lambda, ACT, \mathcal{R}, \mathcal{P}, ROLE, PLAYERS)$$

where RVD is a set of integers $\{1, \ldots, n\}$; OBJ is a set of objects; ACT is a set of activity types which describe the subject of a given video frame-sequence (for instance, car chasing, holding a party); EVT is a set of events which could be considered instances of activity types; λ is a function which maps each element from $OBJ \cup EVT$ into a set of frame-sequences; $\mathcal{R}$ is a set of roles which is a description of certain aspect of an activity (roles may involve objects: drivers are roles in the activity *car chasing*); $\mathcal{P}$ is a function which maps an event to an activity type; $ROLE$ is a function which maps a frame-sequence $EVT \cup (\bigcup_{o \in OBJ} \lambda(o))$, an activity in $A \in \mathcal{P}$, and a role in $\mathcal{R}(A)$ into an element of OBJ; $PLAYERS$ is a function which maps an event and its activity type into a mapping from the roles of the activity to the entities in the database and to strings.

The clear separation of salient objects and events in a video helps to model these two entities because they have very different structures. Furthermore $ROLE$ and $PLAYERS$ are introduced to describe a real world event. This kind of generalization is very close to a human's perception of the real world. The events that are of the same activity type are differentiated from each other by the set of objects involved in them. However, the conditions placed on segmenting a video many be too restrictive to capture all events. Although the authors have claimed that their frame data structures, designed to process different types of queries, are good and that there exist polynomial-time algorithms for traversing these structures, they do not define a clear query model or query language. A procedural querying facility is supported in AVIS in contrast to a declarative one.

A knowledge-based object-oriented query language, called PICQUERY+, is proposed in [CIT+93]. PICQUERY+ is a high-level domain-independent query language designed for image and alphanumeric database management. It allows users to specify conventional arithmetic queries as well as evolutionary

and temporal queries. The main PICQUERY+ operations include panning, rotating, zooming, superimposing, color transforming, edge detecting, similarity retrieving, segmenting, and geometric operations. A template technique has been used in PICQUERY+ for user accesses. *Query templates* in PICQUERY+ are used to specify predicates to constrain the database view. The example query is shown in Table 1. Five

Object	RO	Object Value	LO	Group
PERSON	IN	IMG.feature.object	AND	1
PERSON2	IN	IMG.feature.object	AND	1
PERSON.name	=	"Bill Clinton"	AND	1
PERSON2.name	=	"Jean Chretien"	AND	1
IMG.feature.left	IS	(PERSON2, PERSON)		

Table 1: Template Querying in PICQUERY+

columns are defined in Table 1. The *Object* column defines the interesting objects or objects' attributes. The *RO* column defines relation operators which show how data should be correlated with the object value template in order to satisfy the query conditions. Typical relation operators are arithmetic comparison operators (=, <, > etc.), temporal interval comparison operators (OVERLAPS, DURING, PRECEDES etc.), and fuzzy operators (SIMILAR_TO etc.). The *Object Value* column defines what kind of values could be assigned to the object so that the relation operator evaluates to be true. The *LO* column defines logical operators which are standard, such as AND, OR, and NOT. The *Group* defines parenthesized clauses in complex predicates. As an example of using the *Group* column, let us change the query slightly "*Is there any image such that Jean Chretien is at the left of Bill Clinton or John Major is at the right of Bill Clinton*". Then, two more rows must be inserted into the template: one describes PERSON John Major and another one is for predicate *right*. Also the *Group* column value must be set to 2 for newly added two rows.

OVID (Object-Oriented Video Information Database) [OT93] is an object-oriented video model. It introduces the notion of a *video object* which can identify an arbitrary video frame sequence (a meaningful scene) as an independent object and describe its contents in a dynamic and incremental way. However, the OVID model has no schema and the traditional class hierarchy of OODB systems is not assumed. An inheritance based on an *interval inclusion relationship* is introduced to share descriptional data among video objects. Intuitively if a video frame sequence V_1 includes another video frame sequence V_2, then

some of V_1's attributes and attribute values are automatically inherited by V_2. This means that instances, not types, inherit attributes. Therefore, the hierarchical structure of a video object would be described by a series of derivations, and not by composition. Queries are processed in OVID by VideoChart and VideoSQL. VideoChart is a bar-chart type, pictorial (visual) query interface. VideoSQL facilitates retrieval of a collection of video-objects that satisfy a given condition. VideoSQL supports queries to retrieve video-objects that contain a specified video-object as their attribute value. For sophisticated queries, both VideoSQL and VideoChart must be used.

Since VideoSQL in the OVID system only deals with video media, we are going to change the query slightly to "*Is there any video such that Jean Chretien is at the left of Bill Clinton*". Here is the query in VideoSQL:

select anyObject
from G7 Summit Database
where "Jean Chretien" **is at left** "Bill Clinton"

Note that VideoSQL is quite different from standard SQL. The **select** clause specifies only the category of the resulting video-objects. Video-objects are classified into three categories: **continuous**, **noncontinuous**, and **anyObject** (either continuous or noncontinuous). Continuous video objects consist of a sequence of video frames while noncontinuous video objects have just one video frame. The **from** clause specifies only the name of the database, in which the query will be posted. A unique feature of the OVID system is that users are allowed to have the ability to identify a meaningful scene at any time and to define the meaningful scene with its descriptional data as a video-object. This is because the video object data model does assume the existence of a database schema. As the OVID model depends heavily on the *video description*, it has no strong support for content-based video retrieval. Another drawback of the OVID system is the lack of a powerful query model or query language because not many query primitives (e.g., query synchronization operators) are defined in OVID. The general usage and the implementation efficiency of interval inclusion inheritance needs further verification.

2.2 Fuzzy Queries

One of the unique properties of multimedia database systems compared to traditional text-based database systems is the necessity of fuzzy queries. In practice, exact matches on features rarely produce useful output [Gro94]. Users may have to use fuzzy queries either because of the difficulty in describing some scene or because of an intention to relax search conditions in order to make sure that all requested objects are present. Therefore, each feature should have its own tunable notion of what a close (or similarity)

match means. The closeness of two multimedia objects should be calculated based on a weighted average of the closeness of their respective components.

Now let us look at another query example. Suppose we have a query "*Find any video that includes Jean Chretien shaking hands with Bill Clinton with an airplane in the background that looks like a Boeing 767*". This is a very difficult query even for a specialized video DBMS. This query is fuzzy because of the fuzzy operator *looks like* and uncertain attributes *shaking hands* and *background*. Most query systems discussed above, especially those that are SQL-based, cannot answer this query because such kind of scenes (background, shaking hands) are difficult to describe precisely by just using words. The fuzzy nature means that they are usually easier to process by visual query language systems, such as QBIC [NBE+93], assuming that appropriately similar images or icons have been defined. The concept of similarity has received significant attention in multimedia research. It is generally understood that a mathematical definition of distance provide a good model of human similarity perception. Some quite sophisticated techniques have been developed for similarity measurement [BW92, LP94, SJ95].

It is common to post a query such as "*Find all the images that look like this one*" to an image database if a face has been sketched by an artist. First a number of features must be extracted from the sketch and they are used to assess the similarity to other images. Some feature transformations are necessary because two faces may have been observed under different conditions. The group of invariant transformations dictates the geometry of the space where we measure similarity [SJ95]. In this scenario, the object we want to retrieve from the database is not the same as the object in the query, but something that it is *similar in a perceptual sense*. Most query languages deal with this problem by providing a *SIMILAR* operator to compute the similarity between multimedia objects -- `SIMILAR_TO` in PICQUERY+, `imageSim` and `audioSim` in EVA, and `Sim` in SCORE.

SCORE (a System for COntent based REtrieval of pictures [ATY+95]) is a visual query interface for image databases. SCORE makes use of a *refined* E-R model to represent the content of images. The refined E-R model is introduced to handle the traditional E-R model's type mismatches, i.e., the ambiguity among its components: entities, attributes, and relationships. For example, it is sometimes difficult to state whether a subject is of type entity, attribute, or relationship. The associations among the objects are classified into two types: actions which describes action relationships (typically this includes verbs or verb phrases) and space which describes the relative positions of two objects. A set of graphical input primitives has been provided. Fuzzy queries are allowed by using fuzzy matching of attribute values, imprecise matching of non-spatial relationships, and a controlled process of deduction (and reduction) of spatial relationships. A unique feature of SCORE fuzzy query processing is its novel definition of object

similarity. Two objects are similar if both of the following points are satisfied

- the names of the objects are either the same, or they are synonyms, or the two objects appear in an IS-A relationship hierarchy, and
- the attribute values of the two objects do not conflict.

Let us look at the comparison of similarity values for the attribute position within a picture. The position is relative to the image. An image may be divided into 9 regions, such as *northWest, SouthEast, Center* etc. If two attribute values indicate the same region or neighboring regions, they are similar. Otherwise, they are in conflict.

The extended FBQL [MS95] has a special feature for supporting fuzzy queries. It provides a method of *relaxing a query* when the original query does not have an answer (i.e, returning empty). Intuitively, if there are two features f_1 and f_2, then $f_1 \leq f_2$ indicates that feature f_1 is considered to be a subfeature of f_2. Here, $\leq$ is a partial order defined on the set of features. A vice president and a deputy prime minister are typical subfeatures of a president and a prime minister respectively. A function (`RPL`) is used to determine what constitutes a "possible" answer to a query. For example, assume that we want to search for an image in which President Clinton is signing a document on a particular day, and there is no such image in the database. However, assume that the vice president was actually signing the document on that day because the president was unavailable. If the vice president is a subfeature of president, the query system may retrieve an image with the vice president signing the document on that day instead of returning no image. The same idea applies to object attributes and that is exactly what the `SUBST` function does in extended FBQL.

2.3 Visual Queries

Data visualization is vital in multimedia database systems because of the complex structure and spatio-temporal relationships inherent in multimedia data. Intuitively, a *visual query* is a query that includes not only alphanumeric expressions, but also some other non-alphanumeric expressions, such as icons, pictures drawn by users, sample audio etc. A visual language is one that allows users to post visual queries. Since visual queries have to be transformed into lower level query primitives (such as query algebra, query calculus), visual languages can be seen as an interface between query models and users.

Fuzzy queries can be easily expressed in a visual language by partial or approximate drawings. Graphical tools are necessary to capture and sketch images. Mapping functions must be provided by query systems to map the images into internal query representations. The same approach applies to audio data. Therefore,

visual query models are more difficult to implement than non-visual query models since visual query systems have to take the responsibility of translating user requests into lower level query primitives.

QBIC (Query By Content of Image [NBE+93]) is an image retrieval system that uses the content of images as the basis of queries. The content used by QBIC includes colors, textures, shapes, and locations of user-specified objects (e.g., a person, flower, etc.) or areas (e.g., the lake area) in images, and/or the overall distribution and placement of colors, textures, and edges in an image as a whole. Queries are posed visually, by drawing, sketching, or selecting examples of what is desired. A typical QBIC query is "*Find images where President Bill Clinton is wearing a black suit and sitting at a round table*". The image predicates (President, black, round, ...) are specified graphically using person icons, color wheels, and drawing tools, by selecting samples, and so on.

The Pictorial Query- By-Example (PQBE) [PS95a] language is aimed at the retrieval of *direction relations* from symbolic images. A *symbolic image* is an array representing a set of objects and a set of direction relations among them. As in the case of relational Query-By-Example, PQBE generalizes from the example given by the user. However, instead of having queries in the form of skeleton tables showing the relation scheme, PQBE has skeleton images which are themselves symbolic images.

Figure 3 shows some symbolic images which could correspond to visual scenes, geographic maps or other forms of spatial data. PQBE supports nine primitive binary, pairwise disjoint direction relations in symbolic images: {`NorthWest, RestrictedNorth, NorthEast, RestrictedWest, SamePosition, RestrictedEast, SouthWest, RestrictedSouth, SouthEast`}. These primitive relations correspond to the highest resolution that can be achieved using one symbol per object in symbolic images [PS94]. The query "*Is there any image such that Jean Chretien is at the left of Bill Clinton*" could be posted as shown in Figure 4.

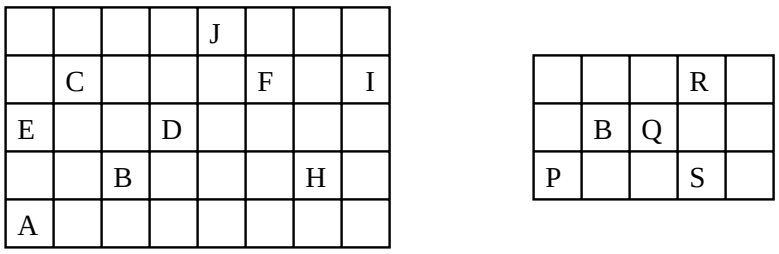

Figure 3: Symbolic Images

From Figure 3 and Figure 4 we can see that symbolic images are just a way of representing direction constraints and they do not preserve absolute metric information but only spatial order. As a consequence, the information preserved in an image does not change if we add or remove empty rows and columns. In

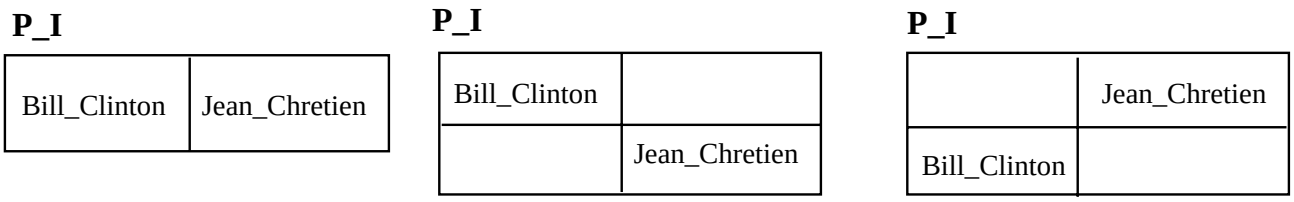

Figure 4: Skeleton Images

Figure 3, symbol characters, such as A, B, C denote *reference* objects. The objects to be located are called *primary* objects. Primary objects are preceded with "P" as shown in Figure 4. The symbolic images used for retrieving other images are called *skeleton images*. We can formally describe the example query in the PQBE internal representation by

$$Jean\_Chretien \in \texttt{O(I)} \wedge Bill\_Clinton \in \texttt{O(I)} \wedge \texttt{West}(Bill\_Clinton, Jean\_Chretien)$$

where `O(I)` is the set of interesting objects in image `I`, $\texttt{West}(X,Y) = \texttt{NorthWest}(X,Y) \vee \texttt{RestrictedWest}(X,Y) \vee \texttt{SouthWest}(X,Y)$. Actually, PQBE has the ability to express negation, independent sub-conditions in the form of multiple skeleton images, union, intersection, and join operations, image and relation retrieval, and to perform inference. It is not difficult to apply PQBE to some multimedia applications, such as GIS and image databases.

However, there are some drawbacks in PQBE approach. First, no quantitative spatial operations are supported, although there is a rich set of qualitative spatial operations. Users cannot ask queries like "Find an object that is 10 meters above this one". Secondly, no computational functions are supported. Typical computational functions are region area, line length etc. Thirdly, no object similarity match functions are supported.

Little et al [LAF+93] have developed a video-on-demand (VOD) system to model a real-world video rental store. A VOD system allows the viewing preferences of each individual to be tailored and adapted to the available programs (for example, one viewer's interest in classic movies, another's restriction to children's programs). A prototype system, called Virtual Video Browser (VVB), is designed to support *temporal access control operations* (such as fastforward, reverse playback, loop, middle-point suspension, middle-point resumption etc.) based on data structures achieved by using a *domain-specific model* for motion pictures. With this domain-specific model, the time-based representation is translated to a conceptual schema in the form of a temporal hierarchy representing the semantics of the original specification. Leaf elements in this model typically represent base multimedia objects (audio, video, text, etc.); only audio and video (and subtitles) elements are used for the video-on-demand application. Timing information is

also captured with node attributes, allowing the assembly of component elements during playback. The VVB application characterizes motion pictures and their attributes to the level of movie scenes by using a movie-specific data schema. This schema interfaces a temporal model to provide temporal access control operations such as fast-forward and reverse playing.

After selecting a video category, users are directed into a *virtual shelf screen* which is a virtual display of the available movies as patterned after a video rental store. Each of the movies on the virtual shelf screen is represented by a rectangular icon which resembles the shape of an actual VHS video box. When the query button is pressed, a query input screen appears. This screen allows users to input queries that identify scenes or identify desired movies. The interface permits users to make queries to the database on any content within the realm of movies, scenes, or actors. In addition to the iconic visual representation of the selected movies, the VVB can display any textual data from the database by using available attributes of title, director, actors, year, synopsis, etc.

The VVB database is implemented using the POSTGRES DBMS [SK91] and temporal access control schemata which captures the relative timing relationships between components of the videos. The query language, called PQUEL, has the following simplified format:

retrieve < determined by output selections >
from < m in movie, s in scene, a in actor >
where < determined by input selections >

In terms of diversity of supported semantic content, VVB is limited by the domain-specific attributes. A significant problem with VVB is that no quantitative media presentation control (e.g., playing this video clip for 5 minutes) is supported.

SCORE is another visual query interface for image databases [ATY+95]. The user query interface of SCORE can facilitate effective query construction in an intuitive manner for casual users while it provides efficient mechanisms for experienced users. The spatial relationships are based on a set of spatial relationship operators {`Over, Under, Left, Right, Behind, In-Front-Of, Overlap, Inside, Outside`}. A set of sound and complete reasoning rules [SYH94] have been provided to eliminate redundant spatial relationships. SCORE is restricted to image databases and the use of image contents (texture, color, etc.) is not as sophisticated as in QBIC.

A temporal visual query language (TVQL) has been developed by Hibino and Rundensteiner [HR95]. In such a language users are able to identify temporal trends in video data by querying for relationships

between video annotations. They can analyze a video in terms of temporal relationships between events (for example, events of type A always follow events of type B). They can also specify *relative temporal queries* or browse video data in a temporally continuous manner. Relative temporal queries have two components: relative temporal position, which describes starting points and ending points of temporal relationships among events, and relative duration, which describes the long (or short) active time of some events compared to others. The primitive temporal operators have been visualized by some graphical primitives as shown in Figure 5. Although all the thirteen temporal primitives (based on Allen's work [All83]) can be used in user queries, TVQL provides another method to express those primitives.

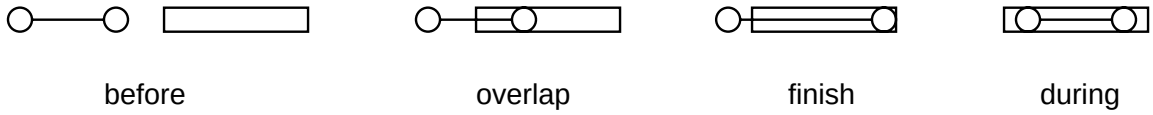

Figure 5: Sample Graphical Temporal Primitives

Consider two events A and B, with starting and ending points A_0, A_1, and B_0, B_1. There are four pairwise endpoint relationships between these two events:

$$A_0 \theta B_0;\ A_0 \theta B_1;\ A_1 \theta B_0;\ A_1 \theta B_1$$

where $\theta \in \{<, >, =\}$. If we suppose the ending time is always greater than the starting time (i.e., $A_1 > A_0$ and $B_1 > B_0$), these relationships can be redefined by starting and ending point differences (e.g. $(A_0 - B_0)\theta\ 0$, $(A_1 - B_0)\theta\ 0$). The benefit of using a difference representation is that it allows users to specify quantitative and continuous ranges of values. Therefore, it is a natural specification to be handled by graphical sliders. All the thirteen temporal relationships can be expressed by one to three of these endpoint relationships. Figure 6 tells us that "The start of A is before the end of B", where the difference of the end time of A and the start time of B is between 0.0 and 5.0 minutes while the difference of the start time of A and the end time of B is between -5.0 and 0.0 minutes. A major problem with this language is in describing complex relationships among multiple events.

Generally, visual query systems are especially good for novice or casual users. Without any knowledge of the underlying system, a user can manipulate the database easily. For experienced users, visual queries can still be beneficial in cases where queries are difficult to express in verbal query forms. Fuzzy queries can be handled much more naturally in visual query models than in verbal query models. However, a major drawback of visual query languages is that specifying well-defined, exact-match queries is not easy in comparison to verbal languages. That is why some multimedia systems, such as OVID, Garlic, and VOD, incorporate both verbal and visual query languages into their systems.

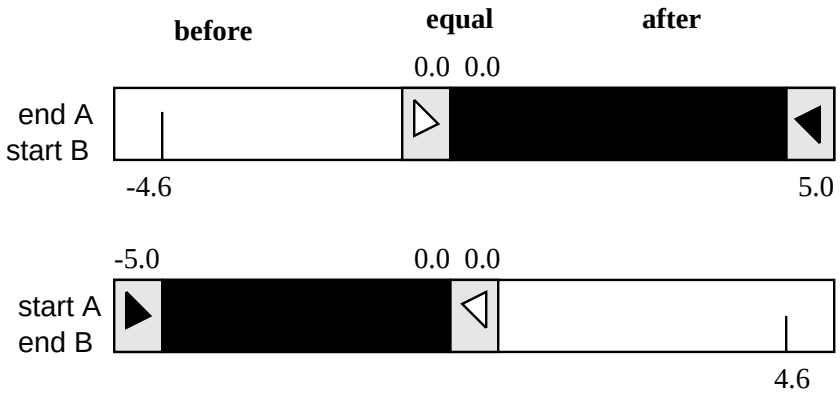

Figure 6: Query Filters for Specifying Relative Temporal Position Queries

2.4 Query Presentations

A *query presentation* refers to the way query results are presented. Presentation of multimedia data is more complex than traditional databases. This is because the modeling of multimedia presentations has to take into account the different aspects important for the presentation and it has to meet the demands of the users of a multimedia presentation. Those different aspects include:

- *Media Composition*: Free composition of different media to a new multimedia product is an essential requirement of any multimedia query model. This kind of composition must describe all temporal and spatial relationships between the media.

- *Interactive Operation*: One of the features of a presentation is the user interaction in the course of a presentation. Some interactive operations (like fastforward, fastbackward, pause etc.) are particularly important in real applications. Furthermore, allowing users to dynamically change presentation parameters (such as the volume of an audio or the speed of a video) is also necessary.

- *Media Synchronization*: Synchronization constraints are needed in order to specify to what extent operations are executed consistent with their temporal constraints, with respect to an ideal execution. In general achieving ideal execution is technically not feasible. The requirements of a multimedia application have to be tied to the functionality of system components. This is why the *quality of service* is introduced to characterize the performance of multimedia system. Some important quality of service parameters are average delay, speed ratio (between the original intended and the actually achieved), data utilization, jitter (the temporal deviation of two simultaneous presentations at a certain point in time), and reliability [KA95].

The query language has to deal with the integration of all retrieved objects of different media types in a synchronized way. For example, consider displaying a sequence of video frames in which someone is speaking, and playing a sequence of speech samples in a news-on-demand video system. The final presentation makes sense only if the speaking person's lip movement is synchronized with the starting time and the playing speed of audio data. This requires fine-grain synchronization (such kind of fine-grain synchronization usually has 25 or 30 synchronization points per second [LG93]). This requires incorporation of synchronization algorithms into query systems.

Because of the importance of delivering the output of query results, supporting query presentation has become one of the most important functions in a query system. Both spatial and temporal information must be used to present query results for multimedia data. The spatial information will tell a query system what the layout of the presentation is on physical output devices, and the temporal information will tell a query system the sequence of the presentation along a time line (either absolute time or relative time). In order to support the spatio-temporal requirement of query presentations, many spatial and temporal operators are provided by different query models. Some typical spatial operators are `left`, `above`, `overlap`, `neighbor`, etc. and some typical temporal operators are `before`, `start`, `meet`, `finish`, etc.

We will see how different query models or languages deal with query presentation by using another example query in this subsection. Consider the query "*Show the videos where Bill Clinton, Jean Chretien, and John Major are chatting at the G7 summit and simultaneously play the audio of their talks; then show the G7 summit logo*".

The representation of the relationship between different objects in EVA is based on the logic of temporal intervals and timed Petri Nets. The temporal relationships are expressed dynamically. The synchronization operators are applied to the objects after the query is evaluated. Following is the query in the EVA system:

{ (content(V) **sim** audio(V) **meets** display(IMG) |
(Video(V) **and exists** Image(IMG) **and**
PERSON **isin** feature(object(V)) **and**
PERSON2 **isin** feature(object(V)) **and**
PERSON3 **isin** feature(object(V)) **and**
name(PERSON) **is** "Bill Clinton" **and**
name(PERSON2) **is** "Jean Chretien" **and**
name(PERSON3) **is** "John Major" **and**
isKeywordIn("G7", keyword(V)) **and**
logo(IMG) **is** "G7") **isin** feature(object(IMG))

Here **sim** means simultaneously and x **meets** y means that object y will be started right after object x terminates. Suppose x and y are time-dependent media objects, then EVA supports the following temporal operators:

- x `before` y: object x is shown before object y
- x `meets` y: object y will be started after x finishes
- x `sim` y: object x and object y are presented simultaneously
- x `starts` y: as soon as object x starts object y start too
- x `finishes` y: as soon as object x ends object y ends too
- x `at` y: object x will be displayed at time y

As for the spatial operators, EVA supports (let x and y be spatial objects):

- $-$ `left` y: object y will be displayed in the left part of the screen (`right`, `bottom`, `up` are similar)
- x `showIn` y: display object x in a window defined by object y which provides the window's upper left corner and lower right corner
- `arrange`(x,y,z): the window of object x covers some portion of the window of object y with degree of visibility z; if $z = 0$, then the portion of the window y that is in the intersection is not visible; if $z = 1$, then all the common points are visible

Although EVA has a rich set of spatio-temporal operators, it lacks some useful operations, such as changing the playing or displaying speed, conditional displaying or playing some objects (e.g., **if** condition **then** present some objects), and playing at a time constraint (e.g., present some objects for 20 minutes).

PICQUERY+ [CIT+93] is a medical image database query language. In PICQUERY+ the same query is shown in Table 2 and the presentation of this query is shown in Table 3. The output of Table 2 is the input of Table 3. *Event* is a general temporal event defined in PICQUERY+. The generality of expressing complex objects and events as an object is a challenging functional feature in multimedia systems. Furthermore, PICQUERY+ undertakes this requirement to integrate a wide range of presentation formats for powerful visualization of query and data processing results.

PICQUERY+ also has a set of spatial operators, like `INTERSECTS`, `CONTAINS`, `IS COLLINEAR WITH`, `INFILTRATES`, `BEHIND` etc., and a set of temporal operators, like `DURING`, `BETWEEN`, `EQUIVALENT`, `ADJACENT`,

Object/Event	RO	Object/Event Value	LO	Group
PERSON	IN	V.feature.object	AND	1
PERSON2	IN	V.feature.object	AND	1
PERSON3	IN	V.feature.object	AND	1
PERSON.name	=	"Bill Clinton"	AND	1
PERSON2.name	=	"Jean Chretien"	AND	1
PERSON3.name	=	"John Major"	AND	1
V.feature.keyword	=	"G7"	AND	1
IMG.feature.logo	=	"G7"		

Table 2: Template Querying in PICQUERY+

Object	Presentation Method
V	MovieLoop
IMG	ShowImage

Table 3: Media Presentation in PICQUERY+

PRECEDES etc. Furthermore, PICQUERY+ supports some *evolutionary operators* which are unique to PICQUERY+.

- EVOLVES_INTO: properties of an object can change generally with respect to time or more specifically with respect to a maturation process
- FUSES_INTO: two or more objects can fuse into a new single object
- SPLITS_INTO: an object can be split into two or more objects

By combining these evolutionary operators with temporal operators, the system is able to answer queries like "*Show in a movie loop all images with delay between image frames 2 seconds from signing a documentation ceremony which demonstrates the fusion of an agreement reached during the G7 Summit'95*". Here, for a particular political or economical issue each participating country's position is modeled as a different original object. After many negotiations the final agreement can be seen as an evolved object obtained from the fusion of those original objects. One of the problems with PICQUERY+ is the lack of facilities for interactive presentation (such as *pause, resume*) or conditional presentation.

The *algebraic video* data model [WDG94] allows users to model nested video structures such as shots, scenes and sequences and to define the output characteristics of video segments. Users can also associate content information with logical video segments, provide multiple coexisting views and annotations of the same data, and provide associative access based on the content, structure and temporal information. A quite comprehensive set of temporal operators has been defined within the algebraic video system. All the operators are classified into four categories:

- Creation: constructing new video objects from existing video objects.
- Composition: defining temporal relationships among video objects.
- Output: defining spatial layout and audio output among video objects.
- Description: associating media content attributes with a video object.

The algebraic video data model consists of hierarchical compositions of *video expressions* with some semantic descriptions. A video expression is constructed by video algebraic operators. Because multiple video frame sequences can be scheduled to play at any specific time within one video expression, the playback may require multiple screen displays and audio outputs. In the algebraic video system, all video expressions are associated with some rectangular screen region in which they are displayed. The layout of all the

components of a query result is decided by the video expression spatial constraints. As expressions can be nested, the spatial layout of any particular video expression is defined relative to the parent rectangle. The parent rectangle is the screen region associated with the encompassing expression.

Besides some basic spatio-temporal operators as presented in EVA and PICQUERY+, the algebraic video system has some extra operators and some of the operators found in other systems have different features. For example, the `window` operator defines a rectangular region within the parent rectangle where the given video expression is displayed. A unique feature about the `window` operator is its *priority* parameter which is used to resolve overlap conflicts of window displays. To repeat a video expression VE for a duration of time t, a user can express it as `loop` VE t where t could be $forever$. Another operator, `stretch` VE *factor*, defines a duration of the presentation which is equal to *factor* times the duration of VE (where VE is a video expression). The `stretch` operator changes the playback speed of the video presentation, but does not alter the playback speed of other presentations. The expression `limit` VE t sets the duration of the presentation to be equal to the minimum of t and the duration of VE, but the playing speed of VE is not changed.

Although the query presentation functionality of the algebraic video system is quite rich, it is not a full-fledged multimedia query model. The data retrieval part of the system is poor. The content-based query access has the format

`search` *query*

which searches a collection of *algebraic nodes* for video expressions that match *query*. An *algebraic node* provides a means of abstraction by which video expressions can be named, stored, and manipulated as units while a *query* is a boolean combination of attributes. The latest work on FBQL [MS95] has shown that all the algebraic composition operators in the algebraic video system can be expressed as constrained queries in the FBQL system.

3 Research Issues

In this section we discuss open problems in multimedia query languages. We will concentrate on the generalization and formalization of query models, spatial queries, temporal queries, and fuzzy queries.

3.1 Generalization of Query Models

Although there are numerous multimedia information systems, not many systems have appropriate query languages to support their applications. Even for those which do have appropriate query languages, they

are designed only for specific applications, and not for general use. For example, OVID and OMEGA are good for video media databases while PICQUERY+ and EVA are designed mostly for image databases. Most systems only address narrow applications, such as medical images or news-on-demand video systems.

Are there any general query languages for multimedia systems? What are multimedia query languages in general, and how can they be formally defined so that they are independent of any specific applications? As far as we know, the only work towards answering the above questions is the FBQL [MS93, MS95] work mentioned earlier. The media-instance model introduced in [MS93] can capture all the media models. As for the query language, some constant symbols, variable symbols, and predicate symbols are defined. Also there is a function symbol *flist*, which stands for feature list, defined as a binary function in the query language. FBQL is based on logic programming, which makes it quite powerful in the sense of expressiveness and correctness because logic programming has a strong mathematical background. However, FBQL does not support enough spatial and temporal presentation properties. It lacks support for content-based information retrieval and heavily depends on user defined features. As we have discussed, metadata features are very important in supporting content-based retrieval, but this vital issue has not been addressed in the FBQL system. The soundness and completeness of FBQL's functions need further investigation. Furthermore, aggregate operations are not allowed in FBQL.

3.2 Spatial Queries

Many applications depend on spatial relationships among the data. *Spatial data* pertains to the space occupied by objects in a database. Typical spatial data includes points, lines, squares, polygons, surfaces, regions, and volumes. The special requirements of multimedia query languages in supporting spatial relationships have been investigated within the context of specific applications such as image database systems and geographic information systems [RFS88, SA95]. The following requirements are necessary for supporting spatial queries from the users point of view:

- Support should be provided for object domains which consist of *complex* (structured) space objects in addition to the simple (unstructured) point and alphanumeric domains. Reference of these spatial objects through their spatial domains must be directed by pointing to or describing the space they occupy and not by referencing their encodings.

- Support should exist for *direct spatial search*, which locates the spatial objects in a given area of images. This can resolve queries of the form "*Find all the faces in a given area within an image or a video frame*".

- It should be possible to perform *hybrid spatial search*, which locates objects based on some alphanumeric attributes and uses the associations between alphanumeric attributes and the spatial objects to facilitate the output. This can resolve queries of the form "*Display the person's name, age, and neck X-ray image if the person's age is greater than 30*" where the neck X-ray image may be extracted directly from the person's X-ray image, whose age is greater than 30.

- Support should exist for *complex spatial search*, which locates spatial objects across the database by using set-theoretic operations over spatial attributes. This can resolve the queries of the form "*Find all the roads which pass through city X*" where one may need to get the location coordinates of city X and then do an intersection with road maps.

- Finally it should be possible to perform *direct spatial computation*, which computes specialized simple and aggregate functions from the images. This can resolve queries of the form "*Tell me the area of this object and find another object which is closest to this one*".

Many query languages we have discussed do not support spatial queries. Visual query languages (or interfaces) usually support spatial queries by the inherited spatial relationships in the icons or user drawings. It should be noted that spatial queries are different from spatial presentation of query results. However, since many query languages have defined a set of spatial operators for data presentation, it is legitimate to investigate whether the semantics of those operators can be extended so they could be used in spatial queries and satisfy the above spatial query requirements.

3.3 Temporal Queries

The inclusion of temporal data modeling in multimedia query languages is an essential requirement. Research in temporal queries has focused more on historical (discrete) databases rather than on databases of temporal media (e.g., [Sno95]). The focus is on the reflections of changes of the representation of real world objects in a database (e.g., President Clinton gave a speech at 2:00pm on July 4, 1995), rather than changes in continuous, dynamic media such as video. A typical temporal query is "*Find a scene where Prime Minister Jean Chretien is shaking hands with President Bill Clinton* after *Bill Clinton steps off an airplane*". While some systems can identify objects that contain shaking hands and stepping off features, the specification of the temporal relationship (*after*) needs special support from query languages.

There are many different approaches for describing temporal relationships (hierarchical, timeline, synchronization points, events etc.) [Bla92]. In the OMEGA system [Mas91], for example, there are no temporal operators, so two temporal relations have been defined. If an OMEGA object has a temporal

property it is called an OMEGA temporal object. The class *TemporalObject* consists of all the temporal objects. Two internal state variables, *birthTime* and *deathTime*, are assigned to each temporal object, where *birthTime* indicates when the temporal object is created while *deathTime* specifies when the temporal object is destroyed. To present temporal precedence and synchronization between objects, those relations are computed first by using the birth time and the death time information, and then their values are assigned to relevant objects with two internal state variables, *tempoPrec* and *tempoSync*, which are defined on temporal objects. That is, the internal state variable *tempoPrec* is used to indicate which object immediately precedes another one, and by how many time units. For example, if a temporal object o is created t seconds before object o' and there are no temporal objects created in this interval, then the *tempoPrec* value of object o' will be object (o, t).

Most query languages discussed do not support temporal queries. A unification of temporal specification between temporal objects and the temporal presentation of query results will definitely ease both multimedia database design and querying tasks. Different query systems have different temporal (and spatial) operators defined. There is no systematic investigation as to how complete those sets of operators are. It is necessary to clarify the basic user requirements of a multimedia system, in order to build a complete system on those basic temporal (and spatial) operators. This completeness is important for evaluating a query language's expressiveness. A sound mathematical definition of those operators is essential.

3.4 Fuzzy Queries

Multimedia query languages should allow fuzzy queries. The nature of queries has to be vague, not due to a lack of user knowledge about a multimedia system, but due to the nature of information and the size of a multimedia database. For example, a personnel information system should be searchable for *a male with short black hair, big eyes, and unsymmetric ears*. Although many existing query languages have some facilities to accept fuzzy queries, they can and should be extended. For example, FBQL provides a framework for relaxing a query by defining a substitute feature set. However, it does not address the feature selection problem in case there are multiple substitutions. A straightforward approach to attack this problem is to assign priorities to those substitute features. Another approach is to build a feature hierarchy, then incorporate this hierarchy into the query model by using an object-oriented inheritance facility. Since extracting features from multimedia objects is an uncertain affair [Gro94], based on current image interpretation and speech recognition, each extracted feature should also have a corresponding certainty factor indicating the accuracy of the given extracted feature. This certainty factor should then influence any feature similarity calculations. Some research results from fuzzy logic should be incorporated

into multimedia query processing.

Support for incremental queries is another functional requirement for multimedia query languages. An incremental query is a query posed over previous ones. For example, if the result of the example personnel information retrieval is too big to check out manually, a second query should be allowed (saying for example that the distance between the mouth and nose should be less) after the user takes a glance at some of the results and provides more restrictions [Jai92]. The important point here is that the second query is only executed against the previous result, not the original multimedia database. This procedure could be repeated until the user is satisfied with the result. None of the known multimedia query languages support incremental queries. As most multimedia queries are fuzzy, this is a very important feature for a query language. Powerful graphic navigation tools (browsers) are necessary for assisting users to complete a sequence of incremental queries.

It is well known that different applications may require different features [Jai92]. To extract all the features is either impractical or time consuming. Since the features must be stored at the time of data entry, we must carefully decide what features will be used in a system. Some features need to be shared by applications and some do not. Furthermore, in order to improve performance some features have to be stored permanently in the system while others may be generated on the fly from existing features. None of the existing query languages at present have probed this issue.

4 Conclusion

In this report we surveyed the query languages for multimedia information systems and justifications are provided for their inclusion in multimedia systems. As far as we know, this is the first survey of its kind. The characteristics of multimedia query languages are classified into four categories: well-defined queries, fuzzy queries, visual queries, and query presentation. The functionalities of different query systems are discussed according to this classification.

Some problems have been identified in this paper. There are important research issues which remain unresolved. First, most multimedia database systems are designed for specific applications. Therefore, the query languages are inherently restricted to a particular domain. This is not acceptable in a dynamically changing research area as new techniques emerge. Second, spatial queries and temporal queries are not supported by most multimedia query systems. This is certainly a serious problem. Furthermore, the uniform processing of spatio-temporal operators in both the data model and the query model is necessary in order to provide a uniform user interface. Third, fuzzy queries should be allowed and supported by

query models. A multimedia DBMS must accept user queries without the requirement of knowing anything about the existing model and structure of the system. Incremental queries can further enhance fuzzy query functionality when a user's initial fuzzy query generates huge results. Furthermore, the completeness and expressiveness of spatial and temporal operators in multimedia query languages need further systematical investigation. The ad-hoc way of adding new operators on top of some other systems results in a poor mathematical base. Last, but not least, is the feature extraction and organization problem. What kind of features are necessary for a particular multimedia database application and what kind of techniques are necessary for using those features in query languages, especially in fuzzy queries?

Most current implementations of multimedia systems are multimedia servers (i.e., no sophisticated query languages, no concurrent control, no data access authorization facility etc). There are more features and functions required than simply the ability to store and deliver multimedia information. These features include and extend typical DBMS functionalities. Multimedia DBMSs should use regular file systems (for efficiently handling traditional data) and multimedia servers (for effectively handling multimedia data) as underlying storage systems and provide additional functions [CC95]. A high level query language is one such additional function and its support is essential in powerful multimedia DBMSs.

References

[ACC+95] S. Adali, K. S. Candan, S. S. Chen, K. Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. accepted for publication by ACM Multimedia Journal, 1995.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832—843, 1983.

[AS94] J. T. Anderson and M. Stonebraker. Sequoia 2000 metadata schema for satellite images. *SIGMOD RECORD*, 23(4):42—48, December 1994.

[ATY+95] Y. A. Aslandogan, C. Thier, C. T. Yu, C. Liu, and K. R. Nai. Design, implementation and evaluation of SCORE (a system for content based retrieval of pictures). In *Proceedings of the 11th International Conference on Data Engineering*, pages 280—287, Taipei, Taiwan, 1995.

[BCN93] A. Bimbo, M. Campanai, and P. Nesi. A three-dimensional iconic environment for image database querying. *IEEE Transactions on Software Engineering*, 19(20):997—1011, October 1993.

[Bla92] G. Blakowski. Tool support for the synchronization and presentation of distributed multimedia. *Computer Communications*, 15(10):611—618, 1992.

[BR94] K. Böhm and T. C. Rakow. Metadata for multimedia documents. *SIGMOD RECORD*, 23(4):21—26, December 1994.

[BW92] R. Basri and D. Weinshall. Distance metric between 3d models and 2d images for recognition classification. A.I. Memo 1373, Artificial Intelligence Laboratory, M.I.T., July 1992.

[Cat94] R. Cattell. *The Object Database Standard: ODMG-93 (Release 1.1)*. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[CC95] S. T. Campbell and S. M. Chung. The role of database systems in the management of multimedia information. In *Proceedings of International Workshop on Multimedia Database Management Systems'95*, Blue Mountain Lake, New York, August 1995.

[CHN+95] W.F. Cody, L. M. Haas, W. Niblack, M. Arya, M. J. Carey, R. Fagin, M. Flickner, D. Lee, D. Petkovic, P. M. Schwarz, J. Thomas, M. T. Roth, J. H. Wiliams, and E. L. Wimmers. Querying multimedia data from multiple repositories by content: the garlic project. In *Third Working Conference on Visual Database Systems (VDB-3)*, Lausanne, Switzerland, March 1995.

[CIT+93] A. F. Cardenas, I. T. Ieong, R. K. Taira, R. Barker, and C. M. Breant. The knowledge-based object-oriented PICQUERY+ language. *IEEE Transactions on knowledge and data engineering*, 5(4):644—657, August 1993.

[CK95] S. Christodoulakis and L. Koveos. Multimedia information systems: Issues and approaches. In W. Kim, editor, *Modern Database Systems*, pages 318—337. Addison-Wesley Publishing Company, 1995.

[DG92] N. Dimitrova and F. Golshani. Eva: A query language for multimedia information systems. In *Proceedings of Multimedia Information Systems*, Tempe, Arizona, February 1992.

[Gro94] W. I. Grosky. Multimedia information systems. *IEEE Multimedia Systems*, 1(1):12—24, Spring 1994.

[HK95] N. Hirzalla and A. Karmouch. The role of database systems in the management of multimedia information. In *Proceedings of International Workshop on Multimedia Database Management Systems*, Blue Mountain Lake, New York, August 1995.

[HR95] S. Hibino and E. A. Rundensteiner. A visual query language for identifying temporal trends in video data. In *Proceedings of International Workshop on Multi-Media Database Management Systems*, pages 74—81, Blue Mountain Lake, New York, August 1995.

[Jai92] R. Jain. Infoscopes: Multimedia information systems. Technical Report VCL-95-107, Visual Computing Laboratory, Department of Electrical and Computer Engineering, University of California at San Diego, 1992.

[JH94] R. Jain and A. Hampapur. Metadata in video database. *SIGMOD RECORD*, 23(4):27—33, December 1994.

[KA95] W. Klas and K. Aberer. Multimedia applications and their implications on database architectures. In P. Apers and H. Blanken, editors, *Advanced Course on Multimedia Databases in Perspective*, The Netherlands, 1995. University of Twente.

[Kim95] W. Kim. Multimedia information systems: Issues and approaches. In W. Kim, editor, *Modern Database Systems*, pages 5—17. Addison-Wesley Publishing Company, 1995.

[KS94] W. Klas and A. Sheth. Metadata for digital media: Introduction to the special issue. *SIGMOD RECORD*, 23(4):19—20, December 1994.

[LAF+93] T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and D. Venkatesh. Digital on-demand video service supporting content-based queries. In *Proceedings of the First ACM International Conference on Multimedia*, Anaheim, CA, 1993.

[LG93] T. C. C. Little and A. Ghafoor. Interval-based conceptual models for time-dependent multimedia data. *IEEE Transactions on knowledge and data engineering*, 5(4):551—563, August 1993.

[LP94] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition. In *Proceedings of the 12th International Conference on Pattern Recognition*, October 1994.

[LPZ93] D. Lucarella, S. Parisotto, and A. Zanzi. MORE: Multimedia object retrieval environment. In *Proceedings of Hypertext'93*, pages 39—50, November 1993.

[Mas91] Y. Masunaga. Design issues of OMEGA: An object-oriented multimedia database management system. *Journal of Information Processing*, 14(1):60—74, 1991.

[MS93] S. Marcus and V. S. Subrahmanian. Multimedia database systems. Submitted for publication, 1993.

[MS95] S. Marcus and V. S. Subrahmanian. Foundations of multimedia information systems. In S. Jajodia and V. S. Subrahmanian, editors, *Multimedia Database Systems: Issues and Research Directions*. Springer Verlag, November 1995.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The QBIC project: Querying images by content using color, texture and shape. In *Proceedings of SPIE*, San Jose, CA, February 1993.

[OT93] E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Transactions on knowledge and data engineering*, 5(4):629—643, August 1993.

[PS94] D. Papadias and T. Sellis. The qualitative representation of spatial knowledge in two-dimensional space. *Very Large Data Bases Journal (Special Issue on Spatial Databases)*, 4:100—138, 1994.

[PS95a] D. Papadias and T. Sellis. A pictorial query-by-example language. *Journal of Visual Languages and Computing (Special Issue on Visual Query Systems)*, March 1995.

[PS95b] P. Pazandak and J. Srivastawa. The language components of DAMSEL: An embedable event-driven declarative multimedia specification language. In *Proceedings of SPIE Photonics*, pages 121—128, October 1995.

[RFS88] N. Roussopoulos, C. Faloutsos, and T. Sellis. Spatial data models and query processing. *IEEE Transactions on on Software Engineering*, 14(5):639—650, May 1988.

[SA95] H. Samet and W. G. Aref. Spatial data models and query processing. In W. Kim, editor, *Modern Database Systems*, pages 338—360. Addison-Wesley Publishing Company, 1995.

[SJ95] S. Santini and R. Jain. Similarity matching. Submitted to: IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995.

[SK91] M. Stonebraker and G. Kemnitz. The POSTGRES next generation database management system. *Communications of the ACM*, 34(10):78—92, 1991.

[Sno95] R. Snodgrass. Temporal object-oriented databases: A critical comparison. In W. Kim, editor, *Modern Database Systems*, pages 386—408. Addison-Wesley Publishing Company, 1995.

[SYH94] P. Sistla, C. Yu, and R. Haddock. Reasoning about spatial relationships in picture retrieval systems. In *Proceedings of VLDB*, 1994.

[WDG94] R. Weiss, A. Duda, and D. K. Gifford. Content-based access to algebraic video. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 140—151, 1994.

[WK87] D. Woelk and W. Kim. Multimedia information management in an object-oriented database system. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 319—329, Brighton, England, September 1987.