

# MINDEX: An Efficient Index Structure for Salient Object-based Queries in Video Databases

Lei Chen<sup>1</sup>, M. Tamer Özsu<sup>1</sup>, Vincent Oria<sup>2</sup>

<sup>1</sup> School of Computer Science, University of Waterloo, {l6chen, tozsu}@uwaterloo.ca

<sup>2</sup> Department of Computer Science, New Jersey Institute of Technology, oria@cis.njit.edu

The date of receipt and acceptance will be inserted by the editor

**Abstract** Several salient object-based data models have been proposed to model video data. However, none of them address the development of an index structure to efficiently handle salient object-based queries. There are several indexing schemes that have been proposed for spatio-temporal relationships among objects and they are used to optimize *timestamp* and *interval* queries, which are rarely used in video databases. Moreover, these index structures are designed without consideration of the granularity levels of constraints on salient objects and the characteristics of video data. In this paper, we propose a multi-level index structure (MINDEX) to efficiently handle the salient object-based queries with different levels of constraints. We present experimental results showing the performance of different methods of MINDEX construction.

## 1 Introduction

has been used in many application fields, such as sports video analysis, surveillance video monitoring systems, digital news library, etc. However, current computer vision and image processing techniques can only offer limited query ability on primitive audio-visual features. The query techniques which have been used in image databases, such as query-by-example [22], can not be easily applied to video retrieval because of the limited number of video examples. Based on the characteristics of video data, the content-based video retrieval approaches can be classified into the following three categories:

1. *Visual feature-based retrieval* [33, 19]: In this approach, a video is recursively broken down into *scenes*, *shots* and *frames*. Key frames are extracted from the shots and the scenes to summarize them, and visual features from the key frames are used to index them. With indexed key frames, this approach converts video retrieval problem into the retrieval of images from image databases.

2. *Keywords or free text-based retrieval* [26, 14]: In this approach, a content description (annotation) layer is put on top of the video stream. Each descriptor can be associated with a logical video sequence or physically segmented shots or scenes. Content-based video retrieval is converted to a search for the specified text in annotation data.
3. *Salient object-based retrieval* [18, 20, 5, 15, 9]: In this approach, salient objects are extracted from the videos and the spatio-temporal relationships among them are described to express events or concepts. The salient objects are the physical objects that appear in video data, (e.g. houses, cars and people), that are of interest in one or more applications.

Visual feature-based retrieval has the advantage that visual feature extraction and comparison can be automatically performed, with very little interpretation required on visual features. However, it is not realistic to expect the users to be knowledgeable about low level visual features. Most importantly, high level semantic similarity may not correspond to the similarity of low level features. For example, sky and sea have a similar visual component, “blue color”, however, they express totally different concepts. Keyword or free text-based retrieval is directly related to the semantics of video content and is easier for users to understand and use. It remains the most popular approach in current video database systems such as news video and documentary video databases. However this approach requires too much human effort to annotate video data and annotations are subjective. Furthermore, text annotations can not cover all the aspects of video data content. For example, it is very difficult to textually describe the moving trajectory of a salient object. Compared to these, salient object-based search is more intuitive and more suitable for human understanding, especially for naive users. Users can directly manipulate salient objects, their properties, and the spatio-temporal relationships among them. They can also construct queries to retrieve videos which contain events that the users are

interested in. These events can be expressed through the spatial or temporal relationships among the salient objects. For example, an interleaving pattern of the temporal relationship “before” between two cars can be used to express a car chase event. Queries related to the spatio-temporal relationships of salient objects can be classified into four types:

1. *Salient object existence.* In this type of query, users are only interested in the appearance of an object. For example, given a movie database, a director may submit the query “give me all the video shots in which actor  $a$  appears” in order to observe the acting skills of the actor.
2. *Temporal relationships.* These queries involve temporal relationships among objects in videos. One possible application for this type of query is to extract interesting shots from movies and construct a trailer. For example, in order to use the shot/reverse shot patterns [1] to construct a car chase scene in a movie trailer, a video editor may first submit two queries, “Give me all the video shots in which car  $a$  appears before car  $b$ ” and “Give me all the video shots in which car  $b$  appears before car  $a$ ”. After that, he/she can choose the shots from two results and concatenate them in an interleaving pattern to build a chase scene between cars  $a$  and  $b$ .
3. *Spatial relationships.* In these queries, users express simple directional or topological relationships among salient objects. These queries may be useful, for example, in sport video analysis. Consider a coach who may want to analyze Michael Jordan’s movements when he is under the backboard in order to train his defense team. A query that he may submit the query over a NBA video database: “Give me all the shots in which Michael has an *under* relationship with the backboard”.
4. *Spatio-temporal relationships.* Users are concerned with the spatio-temporal relationships among salient objects in these queries. This type of query is useful, for example, in surveillance video systems. Consider the case where one may want to retrieve all the shots in which suspect  $a$  enters bank  $b$  by submitting a query “Give me all the shots in which  $a$  enters  $b$ ”.

A major problem in video databases is to find an effective index that can optimize video query processing. In the four types of queries listed above, the query constraints are set on the spatial or temporal relationships among salient objects. Therefore, it may appear that well developed spatio-temporal index structures, such as 3DR-tree [28], HR-tree [21], RT-trees [31], and MVR-tree [27], may be used to improve the query efficiency of salient object-based queries. However, these index structures are mainly designed to optimize *timestamp* and *interval* queries [31], which are common in spatio-temporal databases, but not in video databases. Timestamp queries retrieve all objects that intersect with

a value range window at a specific time. Interval queries consider sequences of timestamps.

These types of queries are rarely, if ever, used in video databases, because they require users to have a comprehensive knowledge of the story line of a video. It is very difficult for users to accurately specify the timestamp or time interval in which the events that they are interested in occur, even though they may be interested in finding the timestamps or intervals of interesting events. In this paper, we focus on index structures to improve efficiency of salient-object based queries.

Two aspects need to be considered when we design an index structure for video databases:

1. Characteristics of queries: Creating index structures without knowing the characteristics of queries may result in the maintenance of redundant information. For example, if we create an independent index for each type of salient object-based queries, the index on spatial relationships also contains the information about object existence. Salient object-based queries allow users to set constraints on salient objects at four different granularity levels corresponding to the four types of queries described above. Different amount of information is required for different constraints.
2. Characteristics of video data: Characteristics of video data may affect effectiveness and efficiency of an index structure. For example, in movies, it rarely happens that more than four actors appear in the same frame. Therefore, an index structure on spatial relationships which relate to more than three objects will be less useful compared to an index structure on pairwise spatial relationship, since most of the time, video frames contain only two actors. Another interesting characteristics of video data is due to the shot/reverse shot techniques, which are often used by video editors to construct dialog and action scenes [1]. Even in sport videos, shot/reverse shot are often used to give the audience different points of view. These techniques cause similar spatial layouts of salient objects to appear in an interleaving pattern.

In this paper, based on a video data model [6], we propose a multi-level index structure, called MINDEX, to improve the efficiency of salient object-based queries that takes into account above two points. At the first level, an extendable hash is created to find the ID of a salient object from its name. A B<sup>+</sup> tree is set up at the second level to index pairwise temporal relationships between two salient objects. Finally, at the third level, a perfect hash is developed to index the spatial relationships among salient objects that appear in each shot. In order to find the optimal index methods for MINDEX, we also propose alternative approaches: signature files and inverted files as the second and third level of MINDEX.

The rest of the paper is organized as follows: Section 2 presents some related works on index structures.

The video data model that forms the base of our index structure is introduced in Section 3. Section 4 presents the MINDEX. We give experimental results on the performance comparison on different methods of MINDEX construction in Section 5. We conclude in Section 6.

## 2 Related Work

As mentioned earlier, there has not been much work on indexing salient objects and their spatio-temporal relationships in video data. Several related index structures are proposed for spatio-temporal databases, image databases and video databases. We briefly review some of the index proposals in this section.

The 3DR-tree [30] is originally proposed to speed up the operations of multimedia object composition and synchronization. It requires that indexed objects do not change their locations over time. 3D Minimum Bounding Boxes (MBB) are used to encapsulate objects over which a 3D R-tree is constructed. With the 3DR-tree, an interval query can be efficiently answered by finding the intersection between the 3D MBB of the query and the MBBs in the 3DR-tree. However, MBBs of moving objects that cover a large portion of the data space may lead to high overlap and low discrimination ability of the 3DR-tree.

RT-trees [31], HR-trees [21], and MVR-trees [27] have been proposed to index spatio-temporal relationships. The RT-tree is a spatio-temporal version of the R-tree; it indexes all the spatio-temporal information in one R-tree, which makes it unmanageable when the number of changing objects is large. The MVR-tree can be considered as a variation of the 3DR-tree in that it combines the concepts of multi-version B tree [2] and the 3DR-tree. An HR-tree can efficiently handle timestamp queries since such a query can be converted into a search over a static R-tree. However, for an interval query, all the trees whose timestamps are located inside the interval have to be searched. The aims of these index structures are to improve the efficiency of the system in dealing with the *timestamp* and *interval* queries. In order to answer queries that involve salient objects, all the timestamped R-trees have to be searched in the HR-tree or a large number of timestamp queries have to be executed against the 3DR-tree.

Several approaches have been proposed to improve the efficiency of spatial relationship-based queries on image databases [12, 17, 3, 10]. Basically, three index structures or their variations exist: inverted file, hash table, and signature file. Inverted files [12] index the appearance of objects in images by creating an index on the name of each object. Querying multiple objects requires taking the intersection of the results of multiple queries over each of the objects. Building a perfect hash over pairwise spatial relationships have been proposed [3]. Again, querying over multiple pairs of spatial relationships requires multiple queries. Furthermore, the perfect

hash structure requires a priori knowledge of all the images. Finally, various signature file structures have been proposed to represent 2D-strings [25] in image databases. Two-level signature file [17] creates an image signature based on spatial relationships among the objects in an image and forms a block signature from all the objects of the images in the block. This is improved upon by a multi-level signature file structure [16] that creates higher-level signatures with bigger size blocks. A two signature-based multi-level signature file structure has also been proposed handel a wider set of queries over 2D-strings.

In video databases, in addition to spatial relationships, temporal relationships are important to describe the characteristics of salient objects. A content-based video query language (CVQL) is proposed in [15], that supports video retrieval by specifying spatial and temporal relationships of content objects. The queries are processed in two phases: the elimination-based processing phase and the query predicate evaluation phase. The elimination phase is proposed to eliminate the unqualified video without accessing the video data and the *behavior-based* function evaluation phase is introduced to examine video functions that are specified in query predicates for retrieving query results. The behavior of salient objects are classified into static, regular moving, and random moving. In order to improve the efficiency of evaluating video functions, an index structure named M-index is proposed to store the behaviors of its content objects. For each type of behavior in a video, an independent index structure is created (e.g. Hash, B<sup>+</sup> tree, or R<sup>+</sup> tree). However, M-index only indexes spatial position information of salient objects, while the temporal relationships among salient objects are not considered.

Döndeler et al. [9] propose a rule-based video database system which supports salient object-based spatio-temporal and semantic queries. In their system, video clips are first segmented into shots whenever the current set of relationships among the salient objects are changed. The frame at which the change occurs is selected as a key frame. The directional, topological and 3D relations of salient objects in a shot in are stored as Prolog facts of a knowledge-base. The comprehensive set of inference rules of the system helps reduce the number of facts to be stored. However, the system does not provide explicit index structures to support salient object appearance or spatio-temporal queries. It relies on the implicit indexes provided by the implementation language SWI-prolog. Therefore, the indexes in their system is implementation dependent.

In an earlier work [4], we proposed a two level index structure for salient object-based queries. A *Salient Object Inverted List* acts as the first level of the index structure to index key frames in which salient objects appear. A variation of HR-tree (called VHR-tree) is used to index the spatial relationships among the salient objects in key frames, which constructs the second level of the

index structure. VHR-trees are designed with the consideration of shot/reverse shot patterns. Figure 1 shows how a VHR-tree handles the spatial patterns brought by shot/reverse shots. At timestamp T1, all the salient objects that appear at T0 disappear, and new set of nine salient objects appear, thus setting up a new R-tree at T1. At timestamp T2, instead of only searching its immediate precedent R2 at T1 as the HR-trees does, the VHR-tree checks the R-trees at T1 and T0, and uses the unchanged part of the R-tree at T0. The same construction procedure applies for timestamp T3.

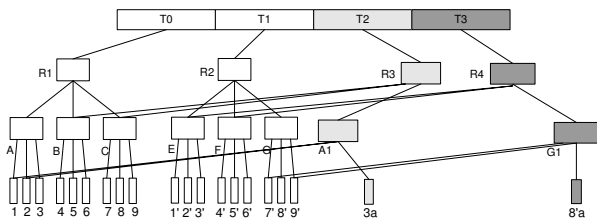


Fig. 1 An example of VHR-tree

However, the two level index structure does not consider the temporal relationships among the salient objects, and sequential scan has to be used to answer temporal and spatio-temporal queries.

### 3 Modeling Video Data

#### 3.1 Overview of the Video Data Model

We use a video data model [6] that captures the structural characteristics of video data and the spatio-temporal relationships among salient objects that appear in the video. The model extends the DISIMA model [23] by adding a video data layer. Figure 2 shows an overview of the improved video data model and its links to the DISIMA image data model.

The DISIMA model captures the semantics of image data through salient objects, their shapes and the spatial relationships among them. It is composed of two main blocks (a block is defined as a group of semantically related entities), as shown on the right hand side of Figure 2: the image block and the salient object block. The image block consists of two layers: the *image* layer and the *image representation* layer. The DISIMA model captures both specific information on every appearance of a salient object in an image (by means of *physical salient objects* and properties) and the semantics or meaning of the salient object (by means of *logical salient objects*). The DISIMA model supports a wide range of queries, from semantic-based to feature-based queries.

A video is often recursively decomposed into scenes and shots. Shots are summarized by key frames. A frame is an image with some specifics (time information, shot it is extracted from, etc.) which can be represented as a

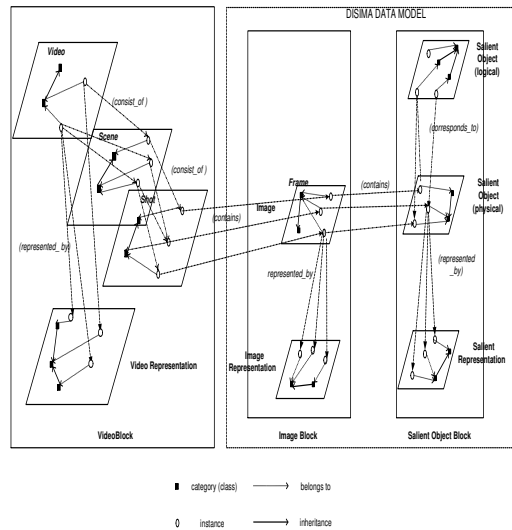


Fig. 2 Overview of the video data model and its links to the DISIMA data model

special kind of image (subclass of image). A new video block, as shown on the left hand side of Figure 2, is introduced to capture the representation and the recursive composition of videos. The lowest level in the video block is the shot, and a shot is linked to its key frames stored as images in the image block. The video block has four layers: *video*, *scene*, *shot* and *video representation*. Because a video *frame* is treated as a special type of image, it inherits all the attributes defined for image entities in addition to a time-related attribute that models the temporal characteristics. The relationship between key frames and shots sets up the connection between a video block and a DISIMA image block.

#### 3.2 Components of the Video Data Model

The definitions of components of the video data model are given below.

**Definition 1.** A *key frame* is a video frame that is selected from a shot to represent the salient contents of the shot. A key frame  $KF_i$  is defined as a six-tuple

$$\langle i, R_i, C_i, D_i, SH_i, LS_i \rangle$$

where

- $i$  is the unique frame identifier;
- $R_i$  is a set of representations of the raw frame (e.g. JPEG, GIF);
- $C_i$  is the content of a key frame  $KF_i$  (Definition 3);
- $D_i$  is a set of descriptive alpha-numeric data associated with  $KF_i$ ;
- $SH_i$  is the shot (Definition 5) to which  $KF_i$  belongs;
- $LS_i$  is the *life-span* of the frame represented as a closed time interval  $[T_s, T_e]$ , which specifies the portion of the shot that  $KF_i$  represents. Since  $LS_i$  is

within the shot, it must satisfy  $LS_i \preceq SH_i.I_i$  where  $\preceq$  is a “sub-interval” operation, defined as follows. Given two time intervals  $I_A$  and  $I_B$ ,  $I_A \preceq I_B$  if and only if  $I_B.T_s \leq I_A.T_s$  and  $I_A.T_e \leq I_B.T_e$ , where  $T_s$  and  $T_e$  are the starting and end times of an interval.

In this data model, key frames are first selected through the automatic processes (using any of the existing key frame selection algorithms (e.g. [32])) and manual interpretation processes are used to mark out the changes of salient objects. With these two steps, a key frame is selected to represent a duration within a shot in which the spatial relationships among salient objects contained in that video frame hold.

We identify, as in DISIMA, two kinds of salient objects: physical and logical.

**Definition 2.** A *physical salient object* is a part of a key frame and is characterized by a position (i.e. a set of coordinates) in the key frame space. A *logical salient object* is an abstraction of a set of physical salient objects and is used to give semantics to that set.

Based on the definitions of physical and logical salient objects, we define the content of a key frame as follows:

**Definition 3.**  $C_i$ , the content of key frame  $KF_i$ , is defined by a triple

$$\langle \mathcal{P}_i, s, \text{Triplelist} \rangle$$

where

- $\mathcal{P}_i$  is the set of physical salient objects which appear in  $KF_i$  and  $\mathcal{P}$  is the set of all physical salient objects ( $\mathcal{P} = \cup_i \mathcal{P}_i$ );
- $s : \mathcal{P}_i \rightarrow \mathcal{L}$  maps each physical salient object to a logical salient object, where  $\mathcal{L}$  is the set of all logical salient objects.
- *Triplelist* is a list of *spatial triples* (Definition 4), which is used to represent the spatial relationships among objects that appear in  $KF_i$ .

**Definition 4.** A *spatial triple* is used to represent the spatial relationship between two salient objects, denoted as:

$$\langle O_i, O_j, SR_{ij} \rangle$$

where

- $O_i$  and  $O_j$  are the physical objects that appear in the key frame.
- $SR_{ij}$  is the spatial relation between  $O_i$  and  $O_j$  with  $O_i$  as the reference object. The spatial relations consist of 8 directional relations (*north, northwest, west, southwest, south, southeast, east, northeast*) and 6 topological relations (*equal, inside, cover, overlap, touch, disjoint*).

Given  $n$  salient objects in a key frame, we need to store  $n \times (n - 1)/2$  pairwise spatial relations in order to capture all the spatial relationships among the salient objects.

**Definition 5.** A *shot* is an unbroken sequence of frames recorded from a single camera operation. A shot  $SH_j$  is defined as a five-tuple

$$\langle j, I_j, KFS_j, SC_j, D_j \rangle$$

where

- $j$  is the unique shot identifier;
- $I_j$  is a time interval which shows the starting and end time of  $SH_j$ ;
- $SC_j$  is the scene (Definition 6) to which  $SH_j$  belongs. Since  $SH_j$  is within  $SC_j$ , it satisfies:  $I_j \preceq SC_j.I_j$ ;
- $KFS_j$  is a sequence of key frames  $[KF_{j,1}, \dots, KF_{j,m}]$ , where  $m$  is the number of key frames in  $SH_j$ .  $KFS_j$  is used to represent the content of a shot.
- $D_j$  is as given in Definition 1.

**Definition 6.** A *scene* is a sequence of shots which are grouped together to convey the concept or story. A scene  $SC_k$  is defined by a five-tuple

$$\langle k, I_k, SHS_k, V_k, D_k \rangle$$

where

- $k$  is the unique scene identifier;
- $I_k$  is a time interval which shows the starting and end times of the  $SC_k$ ;
- $V_k$  is the video (Definition 7) to which  $SC_k$  belongs.  $SC_k$  is a part of  $V_k$ , therefore,  $SC_k$  satisfies  $I_k \preceq V_k.I_k$ ;
- $SHS_k$  is a sequence of shots  $[SH_{k,1}, \dots, SH_{k,m}]$ , where  $m$  is the number of shots in  $SC_k$ .  $SHS_k$  is used to construct  $SC_k$ ;
- $D_k$  is as given in Definition 1.

**Definition 7.** A video consists of a sequence of scenes. A *video*  $V_n$  is defined by a five-tuple

$$\langle n, I_n, R_n, SCS_n, D_n \rangle$$

where

- $n$  is the unique video identifier;
- $I_n$  is a time interval which describes the starting and end times of the video  $V_n$ .  $I_n.T_s = 0$ , since all the video start at time 0;
- $SCS_n$  is a sequence of scenes  $[SC_{n,1}, \dots, SC_{n,m}]$  that is contained by  $V_n$ , where  $m$  is the number of scenes in  $V_n$ ;
- $R_n$  is a set of representations of  $V_n$ . We consider two main representation models for videos: *raster* and *CAI*. Raster representations are used for video presentation, browsing and navigation, while CAI (Common Appearance Interval) representations are used to express spatio-temporal relationships among salient objects and moving trajectories of moving objects. The raster presentation may be one of MPEG-1, MPEG-2, AVI, NTSC, etc. Shots and scenes are not directly represented in the representation layer. Through time intervals that record durations of shots or scenes and video identifiers that indicate the video to which shots

or scenes belong, portions of video representations can be quickly located and used as the representation for shots or scenes;

- $D_n$  is as given in Definition 1.

### 3.3 Modeling Temporal Relationships Within a Shot

In our proposed video data model, the video shot is the smallest querying unit. Therefore, efficient capture of the appearance of salient objects and the temporal relationships among them directly affects the performance of salient object-based queries.

The CAI model [5] captures the appearance and disappearance of the salient objects. A video shot can be represented as a sequence of CAIs, each representing an interval in which salient objects appear together. Figure 3 shows an example shot extracted from the movie “Gone in 60 seconds”. In this video, object  $O_1$  is Randall and object  $O_2$  is Sara,  $CAI(O_1) = I_1$ ,  $CAI(O_2) = I_2$  and  $CAI(O_1, O_2) = I_3$ .



**Fig. 3** CAIs of an example shot

For any two salient objects that appear in a video shot, we define two types of temporal relationships between them: *appear\_together* and *appear\_before*:

**Definition 8.** Given two salient objects  $O_i$  and  $O_j$  that appear in shot  $SH_k$ , if there exists a time interval  $[T_s, T_e] \preceq SH_k.I_k$  such that both  $O_i$  and  $O_j$  appear in  $[T_s, T_e]$ , we say  $O_i$  and  $O_j$  *appear\_together* in  $SH_k$ , denoted as  $O_i \simeq O_j$ .

**Definition 9.** Given two salient objects  $O_i$  and  $O_j$  that appear in shot  $SH_k$ . If  $O_i$  and  $O_j$  appear in two time intervals  $[T_s^i, T_e^i]$  and  $[T_s^j, T_e^j]$ , respectively, and  $T_e^i \leq T_s^j$ , then  $O_i$  is said to *appear\_before*  $O_j$ , denoted as  $O_i \lesssim O_j$ .

The temporal relationships *appear\_together* and *appear\_before* can be used to construct other temporal relationships [13]. For example,  $O_1 \lesssim O_1 \simeq O_2 \lesssim O_2$  represents that  $O_1$  *overlaps*  $O_2$ .

**Definition 10.** Given a shot  $SH_i$  with  $n$  salient objects  $O_1, O_2, \dots, O_n$ , the *temporal string* of  $SH_i$  is  $O_1 \theta O_2 \theta \dots \theta O_n$  ( $\theta \in \{\simeq, \lesssim\}$ ). A temporal string represents the temporal relationships among salient objects that appear in a shot.

For the sample shot in Figure 3, a temporal string is:  $O_1 \lesssim O_2 \lesssim O_1 \simeq O_2$ . Since  $\simeq$  is symmetric, another valid temporal string of this shot is:  $O_1 \lesssim O_2 \lesssim O_2 \simeq$

$O_1$ . Note that unintuitive relationships such as  $O_2 \lesssim O_2$  are acceptable since they represent occurrences in different intervals (e.g. frames) and the relationship is temporal.

## 4 MINDEX: A Multi-Level Index Structure

The analysis of the four types of salient object-based queries discussed earlier reveals the following:

1. For salient object existence queries, it is only necessary to find all the shots in which the specified salient objects appear without any regard to their temporal appearance orders.
2. For queries related to temporal relationships among salient objects, in addition to checking the existence of the salient objects, it is necessary to investigate the temporal relationships among the salient objects.
3. For spatial queries, all the shots should be retrieved in which the specified salient objects appear, followed by a filtering of the shots in which the specified salient objects have *appear\_together* temporal relationships. These are kept as candidate shots, over which the spatial relationships among the salient objects are checked.
4. For spatio-temporal queries, besides following the same steps as (3), temporal relationships among the salient objects in the candidate shots are also checked.

Among these four different granularity levels of constraints on salient objects, object existence queries help remove the shots that do not contain the specified salient objects; these shots also do not satisfy the other three types of queries. Similarly, the temporal queries with *appear\_together* constraints can be considered as filters to avoid unnecessary searches for spatial and spatio-temporal queries, because the shots in which the salient objects do not appear together can not satisfy any spatial relationships. Therefore, we create an index that considers different granularity levels of constraints on salient objects. Figure 4 shows an overview of MINDEX. The first level is a hash table on names of salient objects, a B<sup>+</sup> tree is used to index pairwise temporal relationships and acts as the second level of the index structure. At the third level, a perfect hash table is created to index all the spatial triples that are contained in each shot. Figure 4 only shows one possible construction of MINDEX; we also propose two other alternative approaches: signature files and inverted files [4]. The construction using signature files is presented in this paper and details for inverted files is described in [4].

### 4.1 The First Level Index Structure: Hash on Names

It is more natural for users to query video databases using names of salient objects instead of their IDs (e.g. “give me all the shots in which Tom Cruise appears”

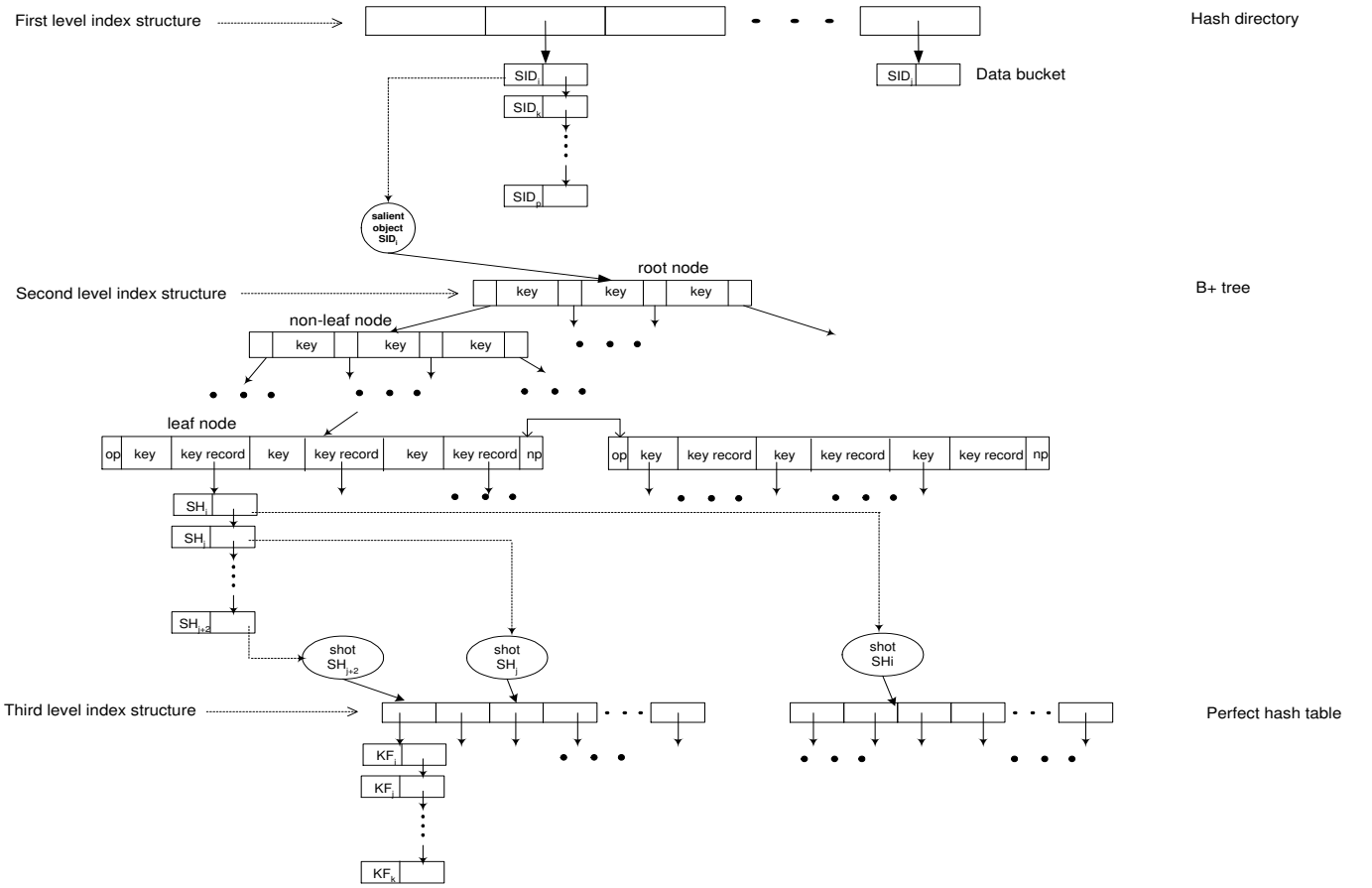


Fig. 4 An overview of MINDEX

is more intuitive than “give me all the shots in which salient object 001 appears”). Furthermore, generally, users do not know salient object IDs. Therefore, we create an extendable hash on the names of salient objects as the first level index structure. Hash is selected because it offers  $O(1)$  access time on data files. We assume that a name is assigned to a salient object when the video is added to the video database. Among the many possibilities, we select the *shift-add-xor* hash function [24] due to its low collision rate. We use  $L = 5$  and  $R = 2$  in our index structure as suggested in [24].

Since there exists the possibility that different name strings have the same hash value, chained lists are used to handle collisions (as shown in Figure 4). Each data bucket of the hash table stores the ID of a salient object and a pointer which points to next data bucket in the chain. Each salient object is stored in a object record structure which is defined as:

```

struct object record{
    int ID;
    string name;
    pointer rootnode;
}

```

where “rootnode” refers to a root node of  $B^+$  tree which is created as the second level of the index structure.

#### 4.2 The Second Level Index

The second level of MINDEX is proposed to quickly filter out the false alarms in answering salient object existence queries and temporal relationship queries. We propose three approaches, a  $B^+$  tree, a multi-level signature file filter, and an inverted file. In the experiment section, we compare the performance of these three access methods.

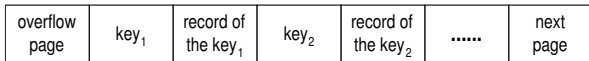
##### 4.2.1 A $B^+$ tree on Pairwise Temporal Relationships

Spatial, temporal, and spatio-temporal relationship queries set constraints on at least a pair of salient objects, since there is no spatial or temporal relationship for a single salient object. Queries that involve more than two salient objects can be handled by taking the intersection of a set of query results on pairs of objects. Therefore, for each salient object  $O_i$ , we create a  $B^+$  tree to index all pairwise temporal relationships between  $O_i$  and other salient objects. IDs of salient objects are used as keys. Each non-leaf node contains  $q + 1$  pointers and  $q$  entries for keys. The structure of leaf node is shown in Figure 5. The value of “key” is the ID of a salient object,

which is an integer, “overflow page” is a pointer to the overflow pages and “next page” points to next leaf node. The internal structure of “record of the key” is defined as following:

```
struct record of the key{
    int tempRel;
    string linkedList;
}
```

where *tempRel* is used to store the temporal relationship, it is a mapped integer value from a temporal relationship. The IDs of shots in which the *tempRel* relationship hold are stored in the *linkedList*.



**Fig. 5** Layout of the leaf node of the B<sup>+</sup> tree

#### 4.2.2 A two signature-based multi-level signature filter

Signature files have been widely used in information retrieval [11, 8, 16]. Recently, they have been used in spatial similarity-based image retrieval [17, 29, 10]. The steps to construct signatures for images are described as follows:

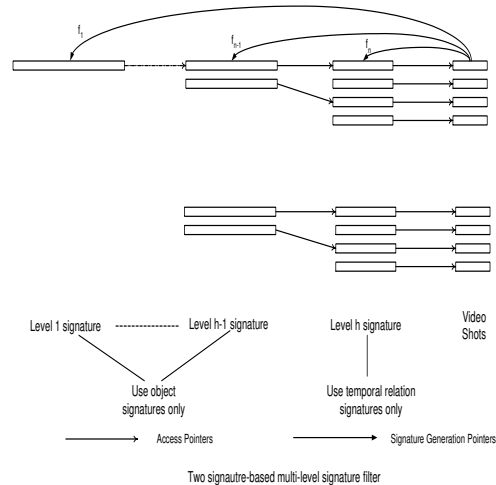
1. For each salient object in an image, we transform it into a binary code word using a hash function. The binary code is  $m$  (signature width) bits wide and contains exactly  $w$  (signature weight) 1s.
2. Each image signature is formed by superimposing (inclusive *OR*) of all the binary codes of salient objects that are contained in the image.

When querying the appearance of salient objects, the objects that are specified in the query are also transformed into binary codes and all the binary codes are superimposed together to form a query signature. The query signature is checked (*ANDed*) with each image signature. However, due to the false drop probability  $P_f$  of signature files (the probability that the signature is identified as containing the query signature, but the real record does not contain the query terms [11]), the images that are pointed at by matched signatures need to be further verified to remove false drops. With  $P_f$ , the number of image  $n$  and the number of distinct salient objects  $s$ , we can compute the optimal values of signature width  $m$  and signature weight  $w$  [16].

The image signatures that are generated using only salient objects are called salient object-based signatures. We can also use the same steps as above to generate spatial relation-based image signatures by coding the spatial relationship pair of salient objects that appear in the images. Elkwaie and Kabuka [10] integrate two signatures into a single index structure, named two signature-based

multi-level signature filter (2SMLSF), to index appearance of salient objects and spatial relationships among objects in images.

In this paper, we use 2SMLSF to index the appearances of salient objects and pairwise temporal relationships of salient objects in video shots.



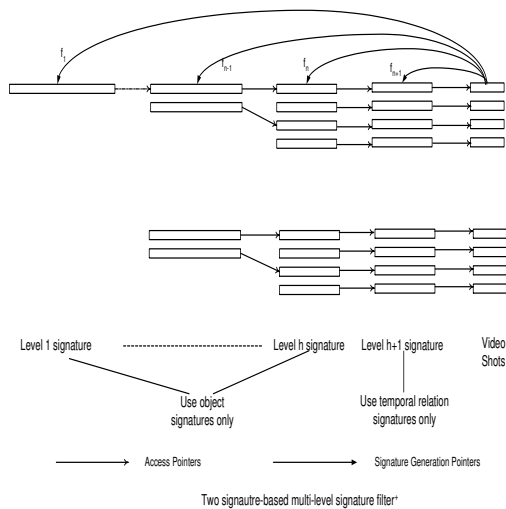
**Fig. 6** Two signature-based multi-level signature filter

As shown in Figure 6, a 2SMLSF file is a forest of  $b$ -ary trees. Every non-leaf node has  $b$  child nodes. There are  $h$  levels in the 2SMLSF. Assume all the trees are complete  $b$ -ary trees, the number of nodes in the structure is:  $n = b^h$ . We generate salient object-based shot signatures by superimposing binary codes of salient objects that appear in the shots and temporal relation-based shot signatures from the temporal relation pairs in the shots. The temporal relation-based shot signatures are used as leaf nodes of the 2SMLSF, the rest of non-leaf (block) signatures are only based on salient objects. We also propose a modified version of 2SMLSF, named 2SMLSF<sup>+</sup>. Compared to 2SMLSF, we add one more level signatures into the 2SMLSF that are generated from salient objects. As shown in Figure 7, in 2SMLSF<sup>+</sup>, the level 1 to  $h$  signatures are generated from signatures of salient objects and level  $h + 1$  signatures (same as  $h$  level in 2SMLSF) are generated from temporal relationships among the salient objects. Due to one more level filtering, 2SMLSF<sup>+</sup> can remove more false alarms when it is used to answer salient object appearance and temporal relationships queries. Our experimental results in Section 5 verify this claim.

#### 4.3 The Third Level Index: Perfect Hash for Spatial Relationships

Both spatial and spatio-temporal queries relate to spatial relationships among salient objects; therefore, an index structure on spatial relationships will be helpful





**Fig. 7** Two signature-based multi-level signature filter +

to answer these queries. In our video data model, a sequence of key frames are chosen to represent a shot. The spatial relationships among the salient objects in each key frame are described by a list of spatial triples. Although the number of key frames that are selected to represent a shot may not be large, scanning each key frame to find the specified spatial relationship is still time consuming, especially when there exists a large number of candidate shots. We use a hash table as the third level index structure to index pairwise spatial relationships in each shot and adapt the technique described in [3] for this purpose. As depicted in Figure 4, key frames that contain the same spatial triple  $(O_i, O_j, SR_{ij})$  will be mapped to the same hash entry and IDs of these key frames are linked together. For each shot, since the number of key frames and the spatial relationships among salient objects in each key frame are known, a minimum perfect hash algorithm can be employed to reduce the storage and avoid the conflicts. 14 integers are used to denote spatial relationships (8 directional and 6 topological). The hash address of a spatial triple  $(O_i, O_j, SR_{ij})$  is:

$$h(O_i, O_j, SR_{i,j}) = SR_{i,j} + \text{associated value of } O_i \\ + \text{associated value of } O_j$$

To assign the associated values to symbols of spatial tuples, we use Cook and Oldehoft’s algorithm [7], which is also used in [3].

The third level index structure is not created for each shot. We define a shot as *spatial indexable* if there are at least two salient objects that appear together in the shot. Shots in which only one salient object appears or no salient objects appear together do not need indexes on spatial relationships, since there are no spatial relationships that can be derived from these shots. However, creating a hash table for each spatial indexable shot induces

redundant information because shots/reverse shots cause similar spatial layouts to appear in an interleaving pattern. Figure 8 shows an example of this interleaving pattern which appears in a dialog scene between “Maximus” and “Princess” in movie “Gladiator”. In the four example shots,  $SH_1$  has a spatial layout similar to that of  $SH_3$  as well as to  $SH_2$  and  $SH_4$ . Therefore, when we create



**Fig. 8** An example of shot/reverse shot pattern in a dialog scene

a hash table for an indexable shot  $SH_i$  and  $(i > 1)$ , two precedent shots  $SH_{i-2}$  and  $SH_{i-1}$  are checked first. If  $SH_{i-2}$  or  $SH_{i-1}$  has exactly the same spatial triples with those of  $SH_i$ , we define the corresponding hash table as *sharable* to  $SH_i$ . Thus, instead of creating a new hash table for  $SH_i$ , the hash table pointer of  $SH_i$  is pointed to the *sharable* hash table, which removes the possible redundant information from the index structure. Figure 4 shows an example that  $SH_{j+2}$  and  $SH_j$  share a hash table.

We also implement two other alternatives approaches to third level indexes: spatial relation-based image signature files and inverted files. The pairwise spatial relationships between two salient objects that exist in a key frame are hashed into binary code words and all the binary code words of the key frame are superimposed to get the image signature of that key frame. Since there are a limited number of key frames for each shot, we store the image signatures sequentially. The inverted files are used to index each distinct pairwise spatial relationship of each shot.

#### 4.4 Creation of the Multi-level Indexing Structure

In general, given a set of  $n$  shots, three steps are needed to create a MINDEX when we use a  $B^+$  tree as the second level of MINDEX:

1. For all salient objects of each shot, use their names to find their IDs in the hash (the first level index) and update the hash directory if these objects are new ones.
2. For all the temporal relationships in the shot, use IDs of involved salient objects to update the corresponding  $B^+$  tree (the second level index structure).
3. For all the spatial relationships in the shot, create a new perfect hash table (the third level index structure) for all the spatial triples contained in the shot if

there is no sharable hash table in the two precedent shots.

In the first step, besides finding the IDs for salient objects, we also need to create an ID for the new salient object and insert it into the hash directory. We consider a salient object as a new one if *NULL* is returned after searching the data bucket which is pointed by the hash directory entry of the salient object. Algorithm 1 presents the steps that are followed to update the second level index structure. The standard update operation of B<sup>+</sup> tree is used in the algorithm. When we use signature files at the second and third levels of MINDEX, we can use the first step as we use for creating MINDEX with B<sup>+</sup> tree as the second level, the second and third steps are described as follows: step 2 is similar to the creation of 2SMLSF for image databases [10]:

---

**Algorithm 1** The algorithm for updating second level index structure

---

ht

**Require:** /\*input: the IDs of all salient objects in a given shot  $SH_i$ \*/

**Ensure:** /\*output: updated B<sup>+</sup> tree of each object record\*/

- 1: Compute all distinct pairwise temporal relationships between two salient objects from temporal strings of  $SH_i$
  - 2: For each temporal relationship  $TR_{ij}$  (*appear\_together* or *appear\_before*) between two salient objects whose IDs are  $ID_i$  and  $ID_j$ , respectively
  - 3: Insert  $ID_j$  and  $TR_{ij}$  into the B<sup>+</sup> tree of the object record which is identified by  $ID_i$
  - 4: Insert  $ID_i$  and  $TR_{ij}$  into the B<sup>+</sup> tree of the object record which is identified by  $ID_j$
- 

---

**Algorithm 2** The searching algorithm for salient object existence queries using B<sup>+</sup> trees

---

**Require:** /\*input: the names of salient objects \*/

**Ensure:** /\*output: the set of IDs of shots that specified salient objects appear\*/

- 1: find IDs of all the salient objects that specified in the queries through the first level of MINDEX
  - 2: the object records that stored specified salient objects are identified through IDs
  - 3: select one B<sup>+</sup> tree that pointed one of the identified object records
  - 4: **if** only one object in the query **then**
  - 5:   insert all the IDs of shots that are stored in the leaf node of the B<sup>+</sup> tree into result set  $R_{set}$
  - 6: **end if**
  - 7: search the B<sup>+</sup> tree with the rest of object IDs and get the intersection of all the searching result sets as the result set  $R_{set}$
  - 8: return the result set  $R_{set}$
- 

1. Determine the height  $h$  of 2SMLSF and 2SMLSF<sup>+</sup> according to the number of shots  $n$ , maximum num-

ber of distinct salient objects of each shot  $s$ , and global false drop probability  $p^f$ . For each level, compute the signature width and weight.

- For each shot, compute all distinct pairwise temporal relationships between two salient objects from the temporal string of that shot. For each pairwise temporal relationship, generate a signature to represent it and superimpose all the signatures of pairwise temporal relationships to construct the temporal relation-based shot signature as leaf node of 2SMLSF or 2SMLSF<sup>+</sup>. A pointer is used to link the leaf node and the logical shot.
  - For each shot, generate  $h - 1$  salient object-based signatures ( $h$  signatures for 2SMLSF<sup>+</sup>) that are based on salient objects of the shot. For each level  $i$ , superimpose  $b^{h-i}$  salient object-based signatures to get block signature at that level.
2. For each key frame in a shot, generate a signature for each distinct pairwise spatial relationship and superimpose all the signatures of spatial pairs to construct a spatial relation-based image signature for the key frame. Store all the image signatures of the shot sequentially in a file.

#### 4.5 Query Processing using Multi-level Indexing Structure

In this section, we discuss how the four types of queries are executed using MINDEX. To answer object existence and pure temporal relationship queries, only the first and second level indexes are needed, while for queries involving spatial and spatio-temporal relationships, all three levels are used. We first present the algorithms in answering different types of queries when a B<sup>+</sup> tree is used as the second level of MINDEX.

1. *Salient object existence queries:* Algorithm 2 presents the steps in answering salient object existence queries.
2. *Temporal relationships queries:* Algorithm 2 can be used to search results for temporal relationship queries; the only additional work is to check the temporal relationship stored at leaf node of the B<sup>+</sup> tree.
3. *Spatial relationship queries:* Algorithm 3 presents the steps in answering spatial relationship queries.
4. *Spatio-temporal relationships queries:* Steps similar to Algorithm 3 are followed to find the candidate shots that contain the specified salient objects and spatial relationships. The candidate shots are further checked to verify whether the specified the temporal relationships are satisfied among the spatial relationships in the candidate shots.

When we use signature files as second and third levels of MINDEX, the following algorithms are used to answer different type of queries:

1. *Salient object existence queries:* The steps in answering salient object existence using 2SMLSF is described in Algorithm 4.

---

**Algorithm 3** The searching algorithm for spatial relationship queries using  $B^+$  tree

---

**Require:** /\*input: the names of salient objects and spatial relationships among them\*/

**Ensure:** /\*output: the set of IDs of shots that contain salient objects and specified spatial relationships\*/

- 1: find IDs of all the salient objects that specified in the queries through the first level of MINDEX
  - 2: the object records that stored specified salient objects are identified through IDs
  - 3: select one  $B^+$  tree that pointed one of the identified object records
  - 4: search the  $B^+$  tree with the rest of object IDs and get the intersection of all the searching result sets as candidate set  $CanSet$
  - 5: **for all** each candidate shot in  $CanSet$  **do**
  - 6:   compute the hash address of the third level index structure for the spatial triple which is constructed from the object IDs and the specified spatial relationships
  - 7:   load the corresponding hash table that is referred by the candidate shot
  - 8:   **if** the hash entry of computed hash address is not empty **then**
  - 9:     insert ID of the candidate shot into  $Rset$
  - 10:   **end if**
  - 11: **end for**
  - 12: return the result set  $Rset$
- 

2. *Temporal relationship queries:* Algorithm 5 presents the steps in answering temporal relationship queries using 2SMLSF. Algorithms 4 and 5 can be applied to 2SMLSF<sup>+</sup> by slightly modifying the searching level and generating query signatures.
3. *Spatial and spatio-temporal queries:* Due to space limits, the detailed algorithms are not given here, we briefly describe the steps as follows:
  - (a) Algorithm 5 is used to find the candidates shots that contained specified query objects and temporal relations.
  - (b) Check the spatial relation-based query signature with each image signature of the candidate shots.
  - (c) If the result is equal to the spatial query signature, the corresponding key frame will be checked to see if it indeed contains the specified spatial relations. If this is the case, insert the candidate shot into the result set.
  - (d) Return the result set.

## 5 Experiment Results and Discussion

We have run experiments to compare the performance of different methods of MINDEX construction in answering salient object-based queries. Due to the lack of sufficient amount of annotated video data, we generated synthetic data to test the performance of MINDEX.

---

**Algorithm 4** The searching algorithm for salient object existence queries using 2SMLSF

---

**Require:** /\*input: the names of salient objects \*/

**Ensure:** /\*output: the set of IDs of shots that specified salient objects appear\*/

- 1: find IDs of all the salient objects that specified in the queries through the first level of MINDEX
  - 2: generate  $h - 1$  salient object-based shot signatures as query signatures:  $S_q^1, S_q^2, \dots, S_q^{h-1}$
  - 3: check (AND operation)  $S_q^1$  with root signatures of 2SMLSF
  - 4: **if** the result equals to  $S_q^1$  **then**
  - 5:   put the access pointer of the root into the candidate block set  $CanBlkSet_1$
  - 6: **end if**
  - 7: level  $i \leftarrow 2$
  - 8: **while** level  $i \neq h$  **do**
  - 9:   **if**  $CanBlkSet_{i-1}$  is empty **then**
  - 10:     return NULL
  - 11:   **end if**
  - 12:   check (AND operation)  $S_q^i$  with the block signature at level  $i$  which is pointed by the access pointer in  $CanBlkSet_{i-1}$
  - 13:   **if** the result equals to  $S_q^i$  **then**
  - 14:     put the access pointer of the block signature into the candidate block set  $CanBlkSet_i$
  - 15:   **end if**
  - 16:   level  $i \leftarrow i + 1$
  - 17: **end while**
  - 18: **if**  $CanBlkSet_{h-1}$  is empty **then**
  - 19:   return NULL
  - 20: **else**
  - 21:   check each shot pointed by the access pointers in  $CanBlkSet_{h-1}$
  - 22:   **if** the shot contains the IDs of salient objects specified in the query **then**
  - 23:     insert the ID of the shot into the result set  $Rset$
  - 24:   **end if**
  - 25: **end if**
  - 26: return  $Rset$
- 

### 5.1 Experiment Setup

In order to generate the synthetic data similar to real movie data, we investigated the appearance frequencies of salient objects in each key frame and the number of salient objects in each shot in three movies<sup>1</sup>. As expected, the appearance frequencies of main actors are higher than those of supporting actors and the number of actors that appear frequently is much less than that of actors who appear only once or twice in the whole movie. Figure 9 shows the number of appearances of actors in three movies. The horizontal axis denotes the actor IDs, the lower IDs are given to the main actors. The vertical axis indicates the number of shots in which an actor appears. This data distribution is very similar to the ZIPF distribution [34]:  $P_i \sim 1/i^a$ , where  $P_i$  is the frequency

<sup>1</sup> 1. "Gladiator", 2000; 2. "Crouching Tiger Hidden Dragon", 2000; 3. "Patch Adams", 1998.

---

**Algorithm 5** The searching algorithm for temporal relationship queries using 2SMLSF

---

**Require:** /\*input: the names of salient objects and temporal relationship among them \*/

**Ensure:** /\*output: the set of IDs of shots that contain specified salient object and temporal relations \*/

```

1: find IDs of all the salient objects that specified in the
   queries through the first level of MINDEX
2: generate  $h-1$  salient object-based shot signatures  $S_q^1, S_q^2, \dots, S_q^{h-1}$ 
   and one temporal relation-based shot signatures  $S_q^h$  as query signatures
3: check (AND operation)  $S_q^1$  with root signatures of 2SMLSF
4: if the result equal to  $S_q^1$  then
5:   put the access pointer of the root into the candidate
   block set  $CanBlkSet_1$ 
6: end if
7: level  $i \leftarrow 2$ 
8: while level  $i \neq h+1$  do
9:   if  $CanBlkSet_{i-1}$  is empty then
10:    return NULL
11:   end if
12:   check (AND operation)  $S_q^i$  with the block signature
   at level  $i$  which is pointed by the access pointers in
    $CanBlkSet_{i-1}$ 
13:   if the result equals to  $S_q^i$  then
14:     put the access pointer of the block signature into the
     candidate block set  $CanBlkList_i$ 
15:   end if
16:   level  $i \leftarrow i+1$ 
17: end while
18: if  $CanBlkSet_h$  is empty then
19:   return NULL
20: else
21:   check each shot pointed by the access pointers in
    $CanBlkSet_h$ 
22:   if the shot contains the IDs of salient objects and tem-
   poral relationships specified in the query then
23:     insert the ID of the shot into the result set  $Reset$ 
24:   end if
25: end if
26: return  $Reset$ 

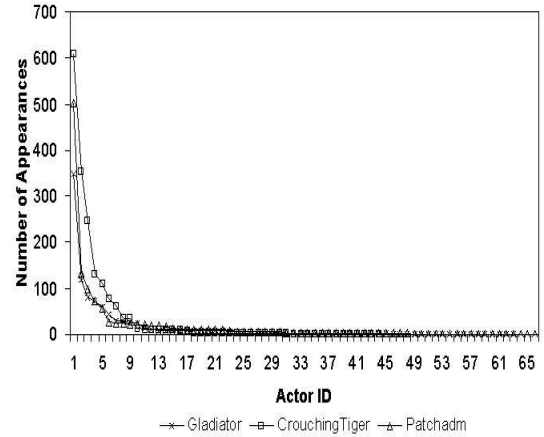
```

---

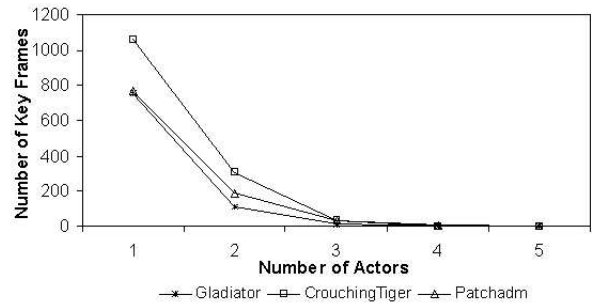
of occurrence of the  $i^{th}$  ranked event,  $i$  is the rank of the event which is determined by the above frequency of occurrence and  $a$  is close to 1. Thus, a few events occur very often while many others occur rarely.

We also found that there are at most five salient objects that appear in one frame (not counting the crowd) and the number of salient objects that appear in key frames also follows ZIPF-like distribution as shown in Figure 10.

According to the changes of spatial relationships among the salient objects that appear in a shot, we manually selected the key frames for each segmented shot of these movies. We found that for each shot, the number of selected key frames is around 1 to 5. In these experiments, we assumed that all salient objects are people and created synthesized names from a list of top 1000 given



**Fig. 9** The appearances distribution of actors



**Fig. 10** The distribution of number of actors

names with appearance probability of each name<sup>2</sup>. We do not use randomly generated strings as names for the salient objects simply because it is not realistic. Furthermore, since the hash value is computed based on each character of a string, randomly generated strings do not reflect the real data distribution of each character as it appears in person names. A random number generator with ZIPF distribution was used to select object IDs that may appear in each key frame. We used another random number generator with ZIPF data distribution to simulate the number of salient objects that may appear in each key frame. Five data sets were created with different numbers of shots and different numbers of salient objects. For each shot, 1 to 5 key frames are randomly generated.

1. 4,096 shots with 41 salient objects;
2. 8,192 shots with 82 salient objects;
3. 16,384 shots with 164 salient objects;
4. 32,768 shots with 328 salient objects;
5. 131,072 shots with 1311 salient objects.

We generated the number of shots as the power of 2 in order to satisfy the assumption of complete  $b$ -ary ( $b = 2$ ) trees of 2SMLSF and 2SMLSF<sup>+</sup>. We set the false drop

<sup>2</sup> Obtained from <http://www.ssa.gov/OACT/babynames/index.html>.

probability of signature files as  $1/n$ , where  $n$  is the number of shots in the data set. We randomly generated three types of queries: salient object existence, temporal relationship, and spatial relationship queries, with a uniform distribution and a ZIPF distribution, 100 queries for each type. Spatio-temporal relationship queries are not tested. In order to answer spatio-temporal relationship queries, MINDEX is used to find the candidate shots that satisfy the specified spatial relationships, and within those shots, we need to check whether the temporal relationships among the spatial relationships satisfy the query constraints. Therefore, we only need to test spatial relationship queries. For object existence queries, the randomly generated queries specify the appearances of 1-5 salient objects. For temporal and spatial queries, the number of objects that are specified is 2 to 5. All the results are averages that are obtained from 100 query results.

The experiments were run on a Sun-Blade-1000 workstation with 512M RAM under Solaris 2.8.

## 5.2 Query Performance

No. obj.	4096		8192		16384		32768		131072	
	U	Z	U	Z	U	Z	U	Z	U	Z
1	0.087	0.797	0.047	0.093	0.031	0.416	0.018	0.243	0.011	0.067
2	0.011	0.602	0	0.031	0.001	0.159	0	0.069	0	0.005
3	0	0.095	0	0	0	0.092	0	0.037	0	0.002
4	0	0.016	0	0	0	0.055	0	0.015	0	0.001
5	0	0.006	0	0	0	0.028	0	0.009	0	0

**Table 1** The selectivity ratios of two types of salient object-based existence queries on five data sets

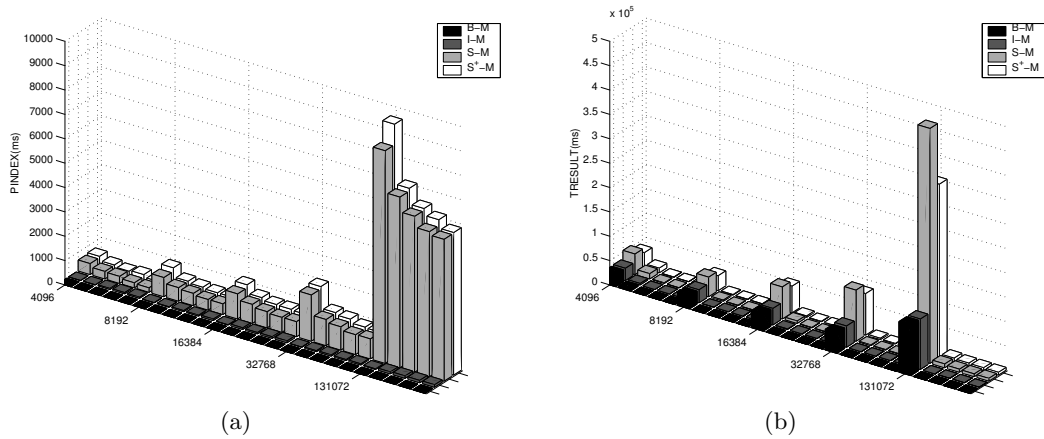
The first experiment was designed to test the performance of MINDEX with different construction on answering salient object existence queries. We use the following abbreviations for four types of MINDEX: B-M for using  $B^+$  tree in MINDEX, S-M for using 2SMLSF,  $S^+$ -M for using 2SMLSF<sup>+</sup> and I-M for using inverted file. We use two performance measures: one is the time that is spent on index retrieval and processing time, called *PINDEX*. For queries that are related to more than two salient objects, *PINDEX* includes the time that is spent on finding the intersection of candidate sets for  $B^+$  tree and inverted files. The other one is the total time that is spent on index retrieval, processing and retrieving the results, named *TRESULT*. For signature files, *TRESULT* includes the time to remove the false drops. Table 1 presents the selectivity ratios of queries and shows that object existence queries that are generated from ZIPF (Z) distribution have much higher selectivity ratios compared to those of queries from uniform (U) distribution. We tested both types of queries. Figure 11 shows the *PINDEX* and *TRESULT* values of four types of MINDEX in answering salient object existence queries that are generated from a uniform distribution and Figure 12 shows the same for ZIPF distribution.

The horizontal axis denotes the different sizes of data sets. For each data set (e.g. data set 4096), the results of queries on 1 to 5 salient objects are shown sequentially starting from the position of the label. The vertical axis indicates the time that is spent on *PINDEX* or *TRESULT* in milliseconds.

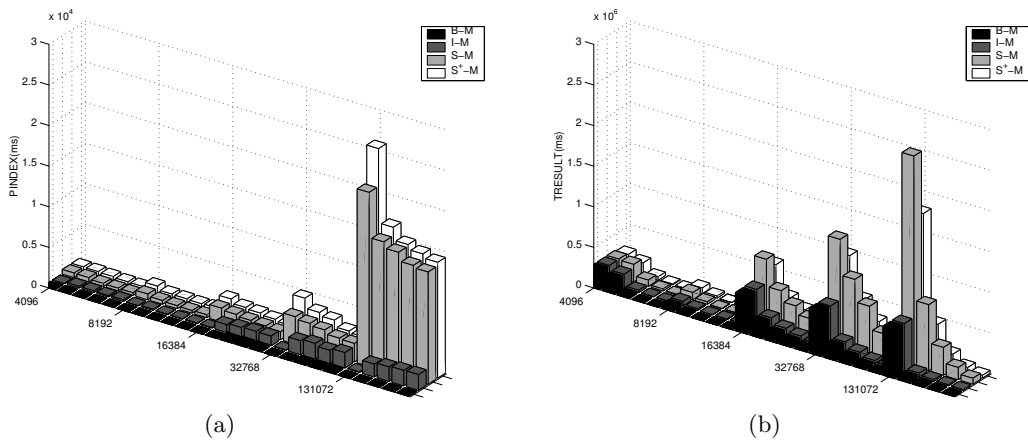
As shown in Figure 11(a), considering index processing time, B-M outperforms other three index structures. When the selectivity ratio is low, B-M and I-M are always better than S-M or  $S^+$ -M, because of the false drops that are introduced by signature files. Since B-M uses  $B^+$  trees to index pairs of salient objects, it produces fewer candidate shots compared to that of I-M which creates an inverted list for each individual object. As a consequence, B-M needs less time to find the intersection from candidate sets. This has been shown in both figures, especially in Figure 12(a) in which I-M spent much more time in finding the intersection of candidate sets. We also find that the index processing time for  $S^+$ -M is nearly the same as that of S-M, because signature files are binary words and the time for loading and comparing one more level of signature files is minimal especially when we use multi-level filtering. Considering the total time that is used to retrieve the answers (Figures 11(b) and 12(b)), B-M is also the best. However, the difference between B-M and I-M is very little when the selectivity ratio is higher, since the time that is spent on index processing only counts for a very small portion of total retrieval time. There is another interesting fact that is shown in Figures 11(b) and 12(b); the difference between  $S^+$ -M and S-M on *TRESULT* becomes larger with the increasing of the size of the data set. This confirms that  $S^+$ -M can remove more false drops than S-M; the bigger the size of data set, the more false drops can be removed by  $S^+$ -M.

The second experiment was designed to test the query performance of four different types of MINDEX on answering temporal relationship-based queries. We present results of queries from uniform and ZIPF distributions in Figures 13 and 14, respectively. Similar to Figures 11 and 12, the horizontal axis denotes the different sizes of data sets. For each data set, the results of queries on 2 to 5 salient objects are shown sequentially.

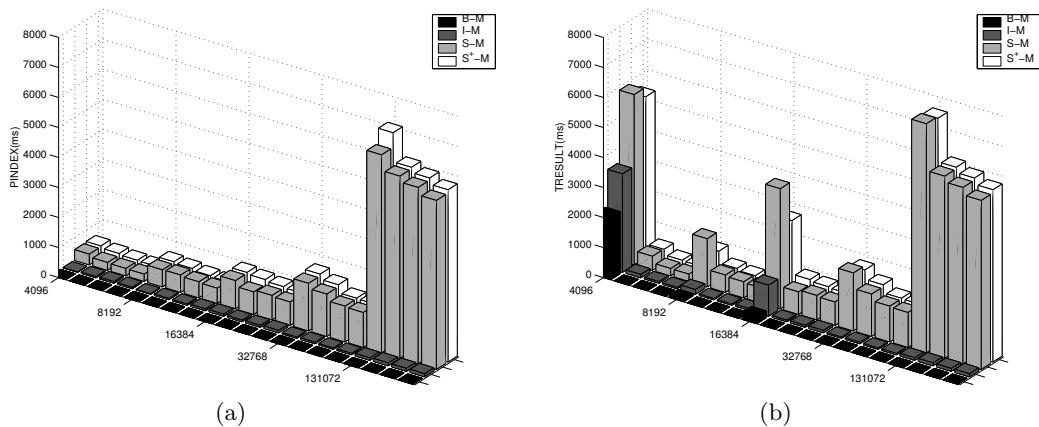
Figures 13 and 14 show that B-M saves significant amount of time in *PINDEX* and *TRESULT*. Since B-M encodes the pairwise temporal relationship into the key record of  $B^+$  tree, the size of candidate sets that are obtained from B-M is much smaller than those from I-M. Therefore, B-M saves time on both index processing (finding the intersection set) and retrieving results (removing false alarms). I-M only creates an index on the appearance of salient objects, it requires further examination on the candidate shots to confirm the existence of the query temporal relationships. The existence of false drops for signature files again leads to the inefficiency in answering temporal relationship queries. As shown in



**Fig. 11** A comparison of four types of MINDEX on salient object existence queries (Uniform)



**Fig. 12** A comparison of four types of MINDEX on salient object existence queries (ZIPF)



**Fig. 13** A comparison on four types of MINDEX on temporal relationship queries (Uniform)

Figures 13(b) and 14(b), S<sup>+</sup>-M again performs better than S-M in terms of total retrieval time.

From above two experiments, we conclude that B<sup>+</sup> tree on pairwise salient objects is the best index struc-

ture that acts as second level of MINDEX. As discussed in Section 4.5, spatial relationship queries help to identify the candidate shots for spatio-temporal queries. Therefore, the last experiment was designed to check the per-

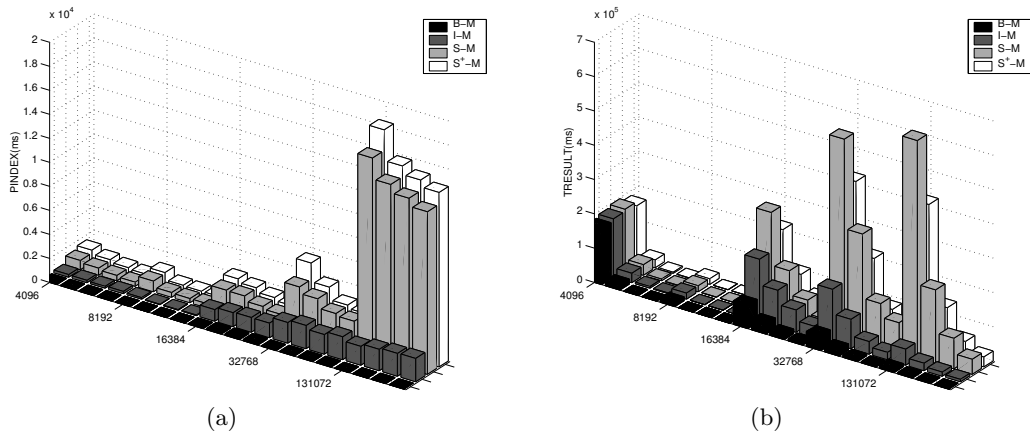


Fig. 14 A comparison on four types of MINDEX on temporal relationship queries (ZIPF)

formance of processing spatial queries on different index structures for third level of MINDEX. In this experiment, we use  $B^+$  tree as second level of MINDEX, and three index structures are tested: perfect hash table (HT), sequentially stored signature files (SSF), and inverted files (IF). Compared to salient object existence and pure temporal queries, spatial queries incur extra cost on reading in key frames to obtain spatial information. Figure 15 shows the  $TRESULT$  of retrieval key frames to answer spatial queries on two salient objects that generated from ZIPF distribution. The reason that we select queries on two salient objects is because they have higher selectivity ratio. The results show that HT is the best candidate for the third level of MINDEX.

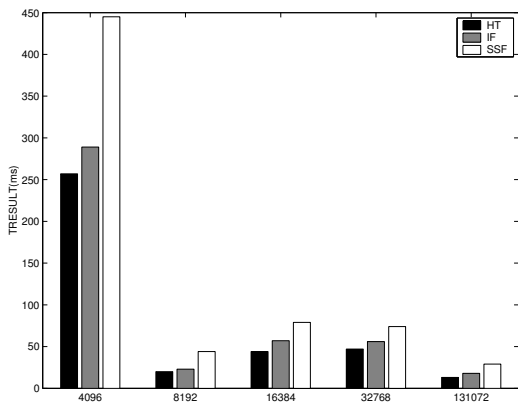


Fig. 15 Performance of spatial relationship queries on two salient objects

## 6 Conclusions

Several approaches have been proposed in literature for salient object-based queries on video databases. However, most of them focus on modeling video data using

salient objects and sequential search is used to answer these queries. However, when the size of a video database grows, it is quite time consuming to answer queries using sequential scan. Very few indexes have been proposed to quickly answer salient object-based queries, and these either create indexes only for spatial relationships or rely on implicit indexing mechanism provided by the implementation languages. Querying video databases on salient objects is quite different from querying spatio-temporal databases. The two fundamental types of queries in spatio-temporal databases, *timestamp* and *interval* queries, are rarely used in salient object-based video databases, since users normally do not have any knowledge about the timestamps or intervals in which some specified event happens. What they are interested in is exactly to retrieve those timestamps or intervals! Therefore, the index structures on spatio-temporal databases cannot be directly applied to salient object-based video databases.

In this paper, we present a multi-level index structure (MINDEX) for salient object-based queries. The index structure considers the different levels of constraints on salient objects that users may have when they pose queries to the video database. An extendable hash table is created for quickly locating IDs of salient objects through their names, which act as the first level of the index structure. Four candidate index structures,  $B^+$  trees, two types of multi-level signature files and inverted files, are proposed for the second level of MINDEX. Perfect hash table, sequential stored signature files and inverted files are selected as candidates for the third level. All the index structures have been tested with various sizes of synthetic data which are generated according to the data distribution of real movies. Based on the experimental results, we conclude that a  $B^+$  tree used to index pairwise temporal relationships between two salient objects is the best one for the second level index structure. The ideal index structure for the third level of MINDEX is a perfect hash table which indexes all the pairwise spatial

relationships within a shot. The characteristic of video data brought by shot/reverse shots is utilized to share hash tables of the third level index, which avoids saving redundant information.

## References

1. D. Arijon. *Grammar of the Film Language*. Focal Press, 1976.
2. B. Becker, S. Gschwind, T. Ohler, and B. Seeger. An asymptotically optimal multi-version B tree. *The VLDB Journal*, 5(4):264–275, 1996.
3. C.C.Chang and S.Y. Lee. Retrieval of similar pictures on pictorial databases. *Pattern Recognition*, 24(7):675–680, 1991.
4. L. Chen, V. Oria, and M. T. Özsu. A multi-level index structure for video databases. In *Proc. 8th Int. Workshop on Multimedia Information System*, pages 28–37, 2002.
5. L. Chen and M. T. Özsu. Modeling video objects in video databases. In *Proc. 2002 IEEE Int. Conf. on Multimedia and Expo*, pages 171–175, August 2002.
6. L. Chen, M. T. Özsu, and Vincent Oria. Modeling video data for content based queries: Extending the DISIMA image data model. In *Proc. of 9th Int. Conf. on Multimedia Modeling*, pages 169–189, 2003.
7. C.R. Cook and R. Oldehoeft. A letter-oriented minimal perfect hashing function. *ACM SIGPlan Notices*, 17(9):18–27, 1982.
8. R.S. Davis and K. Kamamohanarao. A two-level superimposed coding scheme for partial match retrieval. *Information Systems*, 8(4):273–280, 1983.
9. M.E. Dönderler, Ö. Ulusoy, and U. Güdükbay. A rule-based video database system architecture. *Information Sciences*, 143(1):13–45, 2002.
10. E.A. El-Kwae and M.R. Kabuka. Efficient content-based indexing of large image databases. *ACM Trans. Information Sys.*, 18(2):171–210, 2000.
11. C. Faloutsos and S. Christodoulakis. Signature files: an access method for documents and its analytical performance evaluation. *ACM Trans. Information Sys.*, 2(4):267–288, 1984.
12. V.N. Gudivada and G.S. Jung. An algorithm for content-based retrieval in multimedia databases. In *Proc. 2nd Int. Conf. on Multimedia and Computing System*, pages 90–97, June 1995.
13. J.F.Allen. Maintaining knowledge about temporal intervals. *ACM Communications*, 26(11):832–843, 1983.
14. H.T. Jiang, D. Montesi, and A. K. Elmagarmid. Video-Text database systems. In *Proc. 4th Int. Conf. on Multimedia and Computing System*, pages 344–351, June 1997.
15. T.C.T. Kuo and A.L. Chen. Content-based query processing for video databases. *IEEE Trans. Multimedia*, 2(1):1–13, 2000.
16. D. L. Lee, Y. M. Kim, and G. Patel. Efficient signature file methods for text retrieval. *IEEE Trans. Knowledge and Data Eng.*, 7(3), 1995.
17. S-Y. Lee and M-K. SHAN. Access methods of image database. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 4(1):27–42, 1990.
18. J.Z. Li, M.T. Özsu, and D. Szafron. Modeling of moving objects in a video databas. In *Proc. 4th Int. Conf. on Multimedia and Computing System*, pages 336–343, 1997.
19. W. Mahdi, M. Ardebilian, and L.M.Chen. Automatic video scene segmentation based on spatial-temporal clues and rhythm. *Networking and Information Systems Journal*, 2(5):1–25, 2000.
20. M. Nabil, A. H. H. Ngu, and J. Shepherd. Modeling moving objects in multimedia database. In *Proc. of the 5th Int. Conf. on Database Systems for Advanced Applications*, pages 67–76, Melbourne, Australia, 1997.
21. M.A. Nascimento and J. R. O. Silva. Towards historical R-trees. In *Proc. 1998 ACM Symp. on Applied Computing*, pages 235–240, 1998.
22. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. In *Proc. 5th Int. Symp. on Storage and Retrieval for Image and Video Databases (SPIE)*, pages 173–185, 1993.
23. V. Oria, M. T. Özsu, L. Liu, X. Li, J.Z. Li, Y. Niu, and P. Iglinski. Modeling images for content-based queries: The DISIMA approach. In *Proc. of 2nd Int. Conf. on Visual Information Systems*, pages 339–346, Dec 1997.
24. M. V. Ramakrishna and Justin Zobel. Performance in practice of string hashing functions. In *Database Systems for Advanced Applications*, pages 215–224, 1997.
25. S.K.Chang, Q.Y. Shi, and C.W.Yan. Iconic indexing by 2d strings. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(3):413–428, 1987.
26. T.G.A. Smith and G.Davenport. The stratification system: A design environment for random access video. In *Proc. of Int. Workshop on Networking and Operating Syste Support for Digital Audio and Video*, 1992.
27. Y.F. Tao and D. Papadias. MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 431–440, Rome, Italy, 2001.
28. Y. Theodoridis, T. Sellis, A. N. Papadopoulos, and Y. Manolopoulos. Specifications for efficient indexing in spatiotemporal databases. In *Proc. of 1998 IEEE Int. Conf. on SSDDBM*, pages 242–245, 1998.
29. G. Tseng, T. Hwang, and W. Yang. Efficient image retrieval algorithms for large spatial databases. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 8(4):919–944, 1994.
30. M. Vazirgiannis, Y. Theodoridis, and T. Sellis. Spatio-temporal composition and indexing for large multimedia applications. *Proc. 6th ACM Int. Conf. on Multimedia*, 6:284–298, 1998.
31. X. Xu, J. Han, and W. Lu. RT-tree: An improved R-tree index structure for spatiotemporal databases. In *Proc. of 4th Int. Symp. on Spatial Data Handling*, pages 1040–1049, 1990.
32. H. J. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu. Video parsing, retrieval and browsing: An integrated and content based solution. In *Proc. 3rd ACM Int. Conf. on Multimedia*, pages 15–24, 1995.
33. H.J. Zhang, A. Kankanhalli, and S.W. Smoliar. Automatic partitioning of full-motion video. *ACM Multimedia Systems*, 1:10–28, 1993.
34. G.K. Zipf. *Psycho-Biology of Languages*. MIT Press, 1965.