Content-Based Query Processing for Video Databases

Tony C. T. Kuo and Arbee L. P. Chen, *Member, IEEE*

***Abstract*—This paper presents a query processing strategy for the content-based video query language named CVQL. By CVQL, users can flexibly specify query predicates by the spatial and temporal relationships of the content objects. The query processing strategy evaluates the predicates and returns qualified videos or frames as results. Before the evaluation of the predicates, a preprocessing is performed to avoid unnecessary accessing of videos which are impossible to be the answers. The preprocessing checks the existence of the content objects specified in the predicates to eliminate unqualified videos. For the evaluation of the predicates, an M-index is designed based on the analysis of the behaviors of the content objects. The M-index is employed to avoid frame-by-frame evaluation of the predicates. Experimental results are presented to illustrate the performance of this approach.**

***Index Terms*—Query language, query processing, video databases, video indexing.**

I. Introduction

THE retrieval of video data becomes an important issue due to the requirements of the multimedia databases, WWW applications, and video-on-demand systems. Video data contain temporal and spatial relationships between content objects. Based on these characteristics, a new accessing interface can be provided. Traditional database systems only support textual and numerical data. Video data stored in these database systems can only be retrieved through their video identifiers, titles or descriptions.

Many researchers have investigated the enhancement of video query capabilities [3], [5]–[7], [11], [15], [18], [22], [23]. In the past, content-based retrieval was applied in image databases [2], [4], [14], [16]. Similar concepts are extended to enhance query capabilities in video databases. In [18], video data can be queried by image features, such as color, texture and shape. The query capabilities are limited by only using the spatial information and image features of the video frames.

A video query language VideoSQL was proposed in [15]. It used an inheritance mechanism which provides the sharing of common descriptional data among videos and the corresponding operations for specifying queries. In [5], the spatial/temporal semantics of video data were studied. Conceptual spatial objects (CSO), conceptual temporal objects (CTO), physical objects (PO) and a set of predicate logic operations were defined to express queries. Since spatial and temporal semantics are only captured by CSO's and CTO's, semantics that are not defined in CSO's and CTO's cannot be used in queries. A query mechanism based on the spatiotemporal correlation of content objects in the key frames was proposed [22]. A set of *relation description factors* were defined for some special applications. In [7], *hot objects* which are the subject of interest were identified. Hyperlinks were constructed on hot objects for browsing video data.

Various temporal relationships for video query specification were provided in [12]. In [21], a set of temporal operators were designed for specifying video queries. The temporal relationships can only be specified between frame sequences. Spatial relationships of content objects were not considered. In [19], a hierarchical temporal logic for specifying queries was proposed. An extension [13] was presented by providing negation operations in this language.

Sixteen primitive types of motions for specifying the tracks of content objects were considered in [3]. The relationships between content objects were not considered. In [20], we proposed a mechanism to query videos by the motion track of content objects. Query processing is then transformed into the motion track matching problem. In [10], we used the notion of three-dimensional strings (*3-D strings*) to represent the spatial and temporal relationships of content objects. A data structure is also designed for fast query processing.

The above works can only specify the behavior of the content objects to certain degree in a query. More general query mechanisms for users to specify the events in the videos are required. In [8], we proposed a content-based video query language (CVQL) for general query specifications on video databases. This approach was extended for approximate query specifications in [9]. In this paper, based on CVQL, an efficient query processing strategy is presented. The required indices and algorithms are discussed in detail.

The organization of this paper is as follows. In Section II, we present the CVQL. Section III presents the elimination-based preprocessing of CVQL queries. Section IV presents the M-index and the evaluation of the video functions. Section V illustrates various experimental results to show the superiority of our approach. The last section concludes this paper.

II. A Content-Based Video Query Language CVQL

A content object is a symbol in a video, which represents a real world entity. For example, an anchorperson is a content object of a news video. The CVQL allows users to specify predicates by the temporal and spatial relationships of content objects.

A. Video Objects and Content Objects

Various kinds of videos may exist in the database. Videos are organized as a class hierarchy for easy retrieval. For example, in Fig. 1, **Basketball** and **Tennis** are subclasses of video class

Manuscript received August 5, 1999; revised November 3, 1999. The associate editor coordinating the review of this paper and approving it for publication was Dr. Anna Hac.

The authors are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300, R.O.C. (e-mail: alpchen@cs.nthu.edu.tw).

Publisher Item Identifier S 1520-9210(00)01420-6.

1520-9210/00$10.00 © 2000 IEEE

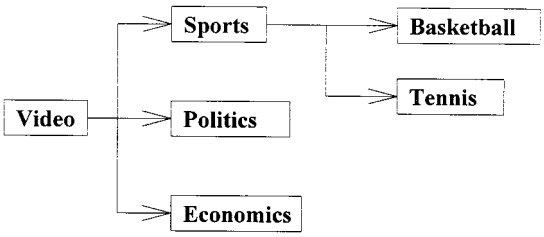


Fig. 1. An example video class hierarchy.

Sports. In this paper, the name of a video will be presented in bold face Roman, and a video class will be presented in bold face Roman beginning with a capital letter. For the convenience of specifying predicates in a query, content objects are also managed in a class hierarchy. When a content object class is used in a query, it represents all content objects belonging to this class. A content object is presented in Arial font and a content object class is presented in Arial font beginning with a capital letter. In Fig. 2(a), we illustrate a class hierarchy of content objects. Fig. 2(b) shows the instances of these classes.

B. Syntax of CVQL

A CVQL query can be expressed by the following structure:

$$\{\text{range; predicate; target}\}$$

- range: The range clause defines the search space of a query. It can be a set of videos or video classes. If the users have no idea about the possible sources where the target may come from, the symbol "*" can be used to represent all videos.
- predicate: The qualification of a query is specified in the predicate clause. Objects in the range clause are evaluated by the specified predicate to get the result.
- target: The target clause specifies the form of the results. The target can be a video, shots, or frames.

C. Predicate Specification

A *state* is a specification of temporal/spatial relationships or existence of content objects, which can be used as a predicate. We introduce various types of state descriptions as follows:

1) *Existence of content objects:* The simplest predicate specification is to search videos which contain the user-specified content objects.
2) *Spatial relationship:* Descriptions of spatial relationships are based on the positions of the content objects in video frames. A frame can be viewed as a two-dimensional (2-D) space. The position of a content object can then be denoted as (X, Y), where $X, Y \geqq 0$. There are two types of spatial relationship descriptions: type 1 regards the position of a content object and type 2 regards the relative positions of two content objects. The spatial relationships among more than two content objects can be represented by concatenating type 2 descriptions with logical operations.
3) *Temporal relationship:* A video stream is a sequence of continuous image frames. The relationships of content objects among video frames are considered as temporal relationships.
4) *Compound relationship:*A more complex state can be specified by combining the descriptions of spatial and temporal relationships of content objects. According to the two types of spatial relationships, we explain their combination with temporal relationships. For type 1, the motion of a content object can be specified by considering the locations of the content object in a continuous frame sequence. For type 2, the variance of the relative locations of two content objects between two continuous frames can be specified. For example, a state describing two dogs running toward each other closer and closer can be specified by the variance of the relative locations of these two dogs in continuous frames.
5) *Compound state:* Combining two or more states in a sequential order in the predicate is named a *compound state*. For example, a ball jumping up and then falling down needs to be described by a compound state: the first state describes the ball jumping up and the second describes the ball falling down.
6) *Semantic description:* It may be inconvenient for users to issue queries by describing a complex relationship, especially for naive users. A semantic description is a way of relieving the difficulty by allowing users to specify a complex state by simple functions. For example, instead of the complex specification of temporal/spatial relationships, a simple operation near(object1, object2) can be used to specify two content objects within a short distance.

In CVQL, a predicate has the following basic form:

$$\text{video-function(parameters)}\,[xy\text{-expression}]$$

It can be parsed into two parts: *video function* and *xy-expression*. The video function part computes a value with an x component and a y component. The xy-expression part specifies the restriction for a predicate. In an xy-expression, the X variable and Y variable represent the x component and y component of the returned value of the video function, respectively. Therefore, if the x component and y component of the returned value of the video function satisfy the xy-expression, the predicate is evaluated as True. Otherwise, it is False. Comparative operators such as "<," ">," "=" and "!=" can be used in the xy-expression. Either X or Y variable can be omitted in an xy-expression when the value of X or Y is irrelevant in the predicate. In the following, video functions for predicate specifications will be introduced.

1) Video functions: A video function returns the information of content objects such as location and motion. There are six functions:

- AP(): It returns absolute position of a content object in a frame:

$$\text{Ex1: AP(dog)}[X < 3 \wedge Y < 3].$$

Ex1 describes whether a dog in a frame stays at the lower-left corner of the frame grid, specified by the xy-expression "$X < 3 \wedge Y < 3$."

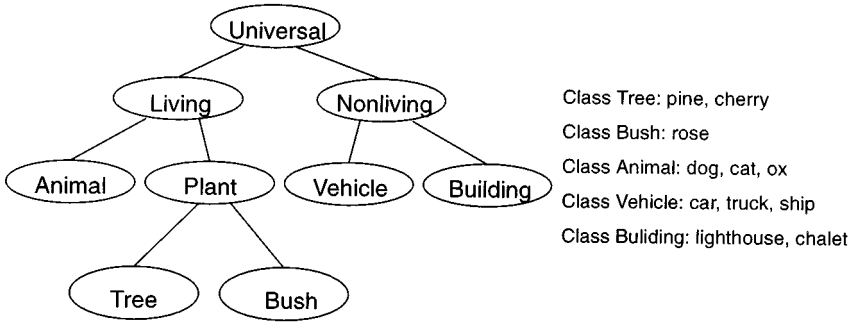


Fig. 2. An example content object class hierarchy.

- RM(): The relative movement of a content object from the previous frame to the current frame will be returned. A sequence of frames showing a bird flying to the right can be specified as

$$\text{Ex2: RM(bird)}\,[X >= 1].$$

In this example, the Y variable is omitted. That is, there is no restriction on the y component of the returned value from RM().

By RM(), users can describe the moving direction and speed of a content object. It can also be used to specify a static content object. For example, Ex3 describes a static bird:

$$\text{Ex3: RM(bird)}\,[X = 0 \wedge Y = 0].$$

- AM(): It calculates the absolute movement of a content object between the frame in which the content object originates and the current frame. Ex4 demonstrates a cat bounded in a 3 × 3 area:

$$\text{Ex4: AM(cat)}\,[|X| < 2 \wedge |Y| < 2].$$

Notice that AP() acts more specifically than AM():

$$\text{Ex5: AP(cat)}\,[3 < X < 7 \wedge 1 < Y < 5].$$

Ex5 tests if the cat is bounded in square (3,1) to (7,5). The difference between Ex4 and Ex5 is that the former bounds the cat in a 3 × 3 area where its position is uncertain, and the latter bounds the cat in an exact area, as shown in Fig. 3.

- RP(): RP() requires two content objects as parameters. By RP(), the relative location of the second content object based on the first content object can be retrieved. Ex6 shows the application of RP():

$$\text{Ex6: RP(tree, car)}\,[X < 0 \wedge Y < 0].$$

In this example, the predicate "the car located at the lower-left of the tree" is specified.

A complex state can be expressed by using these functions. Ex7 is modified from Ex6, which further specifies

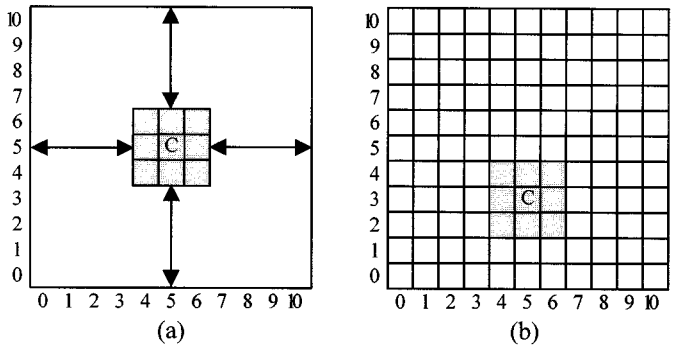


Fig. 3. (a) Cat bounded in a 3 × 3 area. (b) Cat bounded in an area between (3,1) and (7,5).

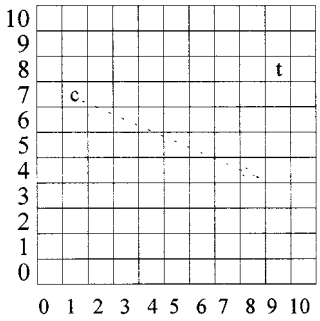


Fig. 4. Illustration of Ex7.

the tree is located statically at (9,8) and the car moves to the right-bottom of the frame, as shown in Fig. 4:

$$\text{Ex7: RP(tree, car)}\,[X < 0 \wedge Y < 0] \text{ and}$$
$$\text{AP(tree)}\,[X = 9 \wedge Y = 8] \text{ and}$$
$$\text{RM(car)}\,[X >= 0 \wedge Y <= 0].$$

- Exist(): Exist() examines whether a content object exists in a frame and returns a Boolean value (True/False). Q1 shows a simple video query which retrieves frames from video **nthu-campus**, in which a content object of content object class Person exists.

$$\text{Q1: } \{\text{video nthu-campus; Exist(Person); frames}\}$$

We show the content object class hierarchy and some frames of video **nthu-campus** in Fig. 5. From Fig. 5(b),

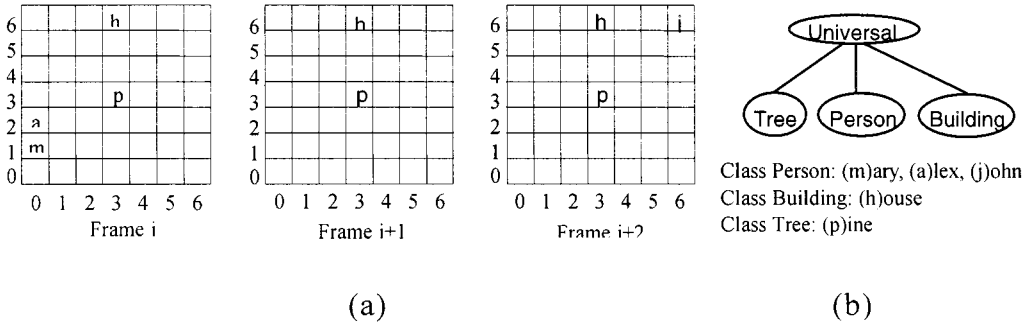


Fig. 5. (a) Some frames of video nthu-campus, and (b) the content object class hierarchy.

we find that content objects mary, john and alex belong to Person. Therefore, the frames of video **nthu-campus** in which content objects mary, john or alex appear will be retrieved as the result of Q1. In Fig. 5(a), frames i and $i+2$ are retrieved since content objects mary and alex exist in frame i and john in frame $i+2$.

- Distance(): Distance() is used to compute the distance for a 2-D value produced by a video function. For example, in frame i of Fig. 5(a), RP(p, a) = (0, 2) – (3, 3) = (–3, –1) and Distance(RP(p, a)) = $\sqrt{(-3)^2+(-1)^2} = \sqrt{10}$. Q2 illustrates an example to find the frames which include two content objects fish and crab in a short distance.

Q2: {video sea; Distance(RP(fish, crab)) $<$ 3; frames}.

Distance() makes different semantics when applied to different video functions, as follows.

1) AP(): It returns the distance of a content object from its location to (0, 0).
2) RM(): It returns the moving distance of a content object from the previous frame to the current frame.
3) AM(): It returns the distance of a content object from the current frame to the frame where the content object first appears.
4) RP(): It returns the distance of two content objects.

By using the video functions, the spatial and temporal relationships among content objects can be specified.

2) Modifiers: In addition to the video functions, a set of modifiers can be used to enhance the power of predicate specifications.

- Increasing, Decreasing and Equal: We have introduced six video functions for retrieving the information of content objects from video frames. However, the basic comparative operators for the xy-expression is not enough for query specification. For example, it cannot specify that two content objects keep the same relative location in two or more continuous frames. The Increasing, Decreasing and Equal modifiers (named XY modifiers) are proposed for supporting the xy-expression for a frame sequence. Q3 illustrates an example which shows two content objects keep the same relative location in two or more continuous frames. In Q4, video objects containing a content object ball which is falling faster and faster will be retrieved.

Q3: {video *; RP(o_1, o_2) [X Equal, Y Equal]; video}

Q4: {video *; RM(ball) [Y Decreasing]; video}.

Therefore, the temporal expression power of the xy-expression is enhanced by these three modifiers, such as the variance of the speed of a moving content object and the variance of the relative position of two content objects. Moreover, a complex xy-expression can be simplified by the XY modifiers. For example, Ex8 and Ex9 are equivalent.

Ex8: RM(o) [$X > 0$] and AP(o) [$X > 3$]
Ex9: AP(o) [$X > 3 \wedge X$ Increasing].

- Continue: In a query, we may have to specify the time interval for a state. The Continue is used to indicate the minimum time interval for a state. The Continue follows an xy-expression or a video function. For instance, Q5 accesses videos which have at least 30 sequential frames containing a content object of Person.

Q5: {video *; Exist(Person) Continue 30 frames; video}.

For convenience, the number of seconds can be used instead of the number of frames as the unit for time interval specification.

- Then: This modifier concatenates two states into a compound state in time sequence. Q6 retrieves frames containing a ball moving left and then moving right. The former state describes a ball moving left and the latter state describes the ball moving right.

Q6: {video sport; RM(ball) [$X < 0$]
Continue 10 frames Then
RM(ball) [$X > 0$] Continue 10 frames; frames}.

- Repeat: The Repeat modifier specifies a repeated compound state. For example,

Q7: {video sport; (RM(ball) [$X < 0$] Continue 10 frames
Then RM(ball) [$X > 0$] Continue 10 frames)
Repeat 1 time; frames}.

In summary, Increasing, Decreasing, and Equal are used to specify the variance of spatial relationships; Continue is used to depict the time duration that a state has to be kept; and Then

and Repeat modifiers are used to arrange the time order of each state.

We have introduced the operations for specifying spatial relationships and temporal relationships between content objects in the query predicate. In the following, we illustrate some more complex query examples.

Q8: {video *; RP(dolphin, ball) $[X = 0 \wedge Y = 1]$
Continue 3 seconds; video}

Q9: {video *; RM(bird) $[|X| <= 1 \wedge |Y| <= 1]$
Continue all frames; video}

Q10: {video *; AP(bird) $[|X| <= 3 \wedge |Y| <= 3]$
Continue all frames; video}

Q11: {video **Disaster**; Distance(RP(train, car))
Decreasing to 0 Then
!Exist(train, car); frames}

Q12: {video **sport**; RM(barrier) $[X = 0 \wedge Y = 0]$ and
RM(horse) $[X >= 0 \wedge Y >= 0]$ and
RP(barrier, horse) $[X <= 0]$ Then
RM(horse) $[X >= 0 \wedge Y <= 0]$ and
RP(barrier, horse) $[X >= 0]$; frames}.

Q8 retrieves video objects containing a frame sequence which has a dolphin crowned with a ball. Such a frame sequence has to be kept for a duration of three seconds.

The targets of Q9 are those video objects containing birds flying slowly. Since the "Continue all frames" is specified, all frames of a result video must have a content object bird satisfying "RM(bird) $[|X| <= 1 \wedge |Y| <= 1]$." Q10 retrieves videos containing birds and these birds are bounded in the area (0, 0) to (3, 3).

In Q11, users want to retrieve video frames of a train crashing with a car from disaster videos. In the predicate of Q11, first we describe the state of a train approaching a car. After the crash, both the car and train disappear from the next frame.

Q12 retrieves the frame sequence containing a horse jumping over a barrier from a sport video. There are two states in the predicate of the query. The first one describes a horse running close to the barrier and jumping up, and the second describes the moving track and the spatial relationships between the horse and barrier after the horse jumps over the barrier.

D. Macros

Macros are defined for the simplification of predicate specifications. It may be difficult for a user to specify a complex state by primitive video functions and modifiers. For example, to describe a crash, instead of the complex predicate specification as shown in Q11, it will be more convenient if a crash function is predefined.

A macro can be defined by primitive video functions and modifiers or the other macros with a set of parameters. For example, Ex10 defines the Near function.

Ex10: $\text{Near}(o_1, o_2) = \text{Distance}(\text{RP}(o_1, o_2)) < 3.$

The 13 temporal relationships proposed in [1] can be defined as macros using our video functions and modifiers. We illustrate the definition of Meet function in the following.

$$\text{Meet}(o_1, o_2) = \text{Exist}(o_1) \text{ and } !\text{Exist}(o \text{ Then } !\text{Exist}(o_1) \text{ and } \text{Exist}(o_2).$$

Ex11 shows the simplification of Q11 by defining a crash macro function, as follows.

Ex11: $\text{Approach}(o_1, o_2) = \text{Distance}(\text{RP}(o_1, o_2))$
Decreasing; $\text{Crash}(o_1, o_2) = (\text{Approach}(o_1, o_2)$
to 0 Then $!\text{Exist}(o_1, o_2))$ and
$\text{Class}(o_1) = \text{Class}(o_2) = \text{Vehicle}.$

The Class function is used to limit the parameters since any content object may be applied as a parameter.

Therefore, Q11 can be simplified to

$$\{\text{video } \textbf{Disaster}; \text{Crash}(\text{car}, \text{train}); \text{frames}\}.$$

For the use of macro, a set of macros are predefined. Users can define new macros by themselves. The definition of a macro can be changed in the user profiles. For example, Ex10 defines two near content objects by their distance "3." Users may change it to "2" for a stricter criterion.

III. Elimination-Based Preprocessing

Query processing evaluates the query predicate for the videos specified in the range clause and returns the qualified targets as query results. The processing is performed in two phases: the preprocessing phase and the evaluation phase. The former performs an elimination-based approach to eliminate the videos which are impossible to be the answers of the query. The candidate videos are retrieved. The latter evaluates the query predicate in detail on these candidate videos and integrates the results to form the query answers.

A. Indices for the Preprocessing

A set of indices are constructed for the preprocessing.

- Class hierarchy of videos: As we discussed in Section II, the videos in the database are classified into a class hierarchy for the convenience of query range specification. Each video class contains the information of
  1) members (videos) of this class;
  2) super-class of this class;
  3) subclasses of this class;
  4) mincov and maxcov of this class (to be described later).
- Class hierarchy of content objects: The content objects are also classified into a class hierarchy for the convenience of predicate specification. Each content object class contains the information of
  1) members (content objects) of this class;
  2) super-class of this class;
  3) subclasses of this class.

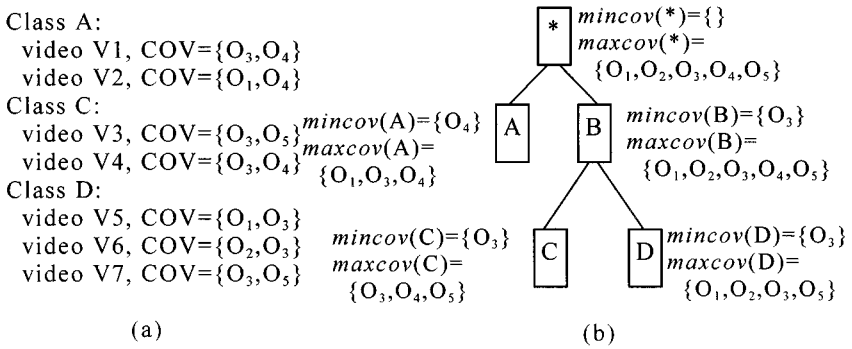


Fig. 6. COV's of a video class hierarchy.

This hierarchy can be used to find the corresponding content objects if the parameter of a predicate contains a content object class.

- Content objects in a video (COV): The COV records the set of content objects in a video. A function *co*() is defined for retrieving the COV of a video. We define two relevant functions to retrieve the COV's for the videos in a video class: (1) the *maxcov*(VC) returns the set of content objects that exist in a video of video class VC and (2) the *mincov*(VC) returns the set of content objects that exist in all the videos of video class VC. The *maxcov*() and *mincov*() of a node N can be computed by the following formula.

if (N is not a leaf node)

$mincov(N) = \cap mincov(n_i)$, where n_i are children nodes of N;

$maxcov(N) = \cup maxcov(n_i)$, where n_i are children nodes of N;

else

$mincov(N) = \cap co(v_i)$, where v_i are videos in video class N;

$maxcov(N) = \cup co(v_i)$, where v_i are videos in video class N.

Fig. 6(a) illustrates the COV's for some videos and Fig. 6(b) shows the *mincov*() and *maxcov*() for the video classes of the video class hierarchy. For example, *mincov*(B) = *mincov*(C) $\cup$ *nmincov*(D) = $\{O_3\} \cap \{O_3\} = \{O_3\}$. The COV is used for the elimination of videos which do not contain the content objects of the query predicate.

B. Steps for the Elimination-Based Preprocessing

For the preprocessing, a query is processed in two steps.

1) Query predicate graph construction: The first step transforms the query predicate into a query predicate graph. The content objects of the query predicate are also computed.
2) Video elimination: According to the content objects of the query predicate, check the COV to eliminate videos impossible to be the answers from the query range. The qualified videos are extracted as *candidate videos*.

The details of each step are elaborated as follows.

1) Query Predicate Graph Construction: We use a top-down approach to construct the query predicate graph. First, construct a state-flow diagram for the query predicate. Second, represent each state by the associated predicates of the query. The state-flow diagram contains two components: (1) the nodes representing the states, and (2) the directed links between the nodes representing the order of the states. The time duration of a state can be specified below the node. There are three types of the state-flow diagram according to the query predicate.

1) Single state: There are no Then or Repeat modifiers in the query predicate. In such a simple case, the query predicate specifies a single condition for the video contents. An example state-flow diagram is shown in Fig. 7(a).
2) Consecutive flow: There are Then modifier(s) in the query predicate. A Then modifier connects two states in the time order. Fig. 7(b) shows state S1 followed by state S2.
3) Repeating flow: There are Repeat modifier(s) in the query predicate. In this case, the query predicate must also contain the Then modifier since a simple query predicate is unnecessary to be repeated. Fig. 7(c) illustrates a repeating flow type state-flow diagram, the n indicates the number of the repetition.

For example, Q6 can be transformed to the state-flow diagram as in Fig. 8. The query predicate contains two states S1 and S2. The duration of each state is ten frames. The flow from S1 to S2 repeats once.

A state can contain a single predicate or a conjunction of more than one predicate. In a query predicate graph, a square node represents a predicate.

When the subpredicates in different states contain the same content object, the content objects are named *relevant content objects*. The relevant content objects will be used for the result integration. A dotted link is used to connect two predicates containing the relevant content objects. Fig. 9(a) and (b) illustrate the query predicate graph construction of Q12. Fig. 9(c) lists the predicates P1–P5.

Furthermore, the content objects of the query predicate are computed from the query predicate graph QG by a function

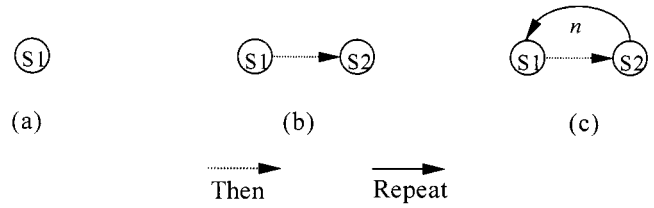


Fig. 7. (a) Single state, (b)consecutive flow, and (c) repeating flow types of the state-flow diagram.

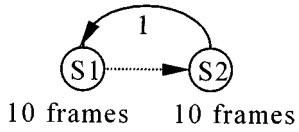


Fig. 8. State-flow diagram of Q6.

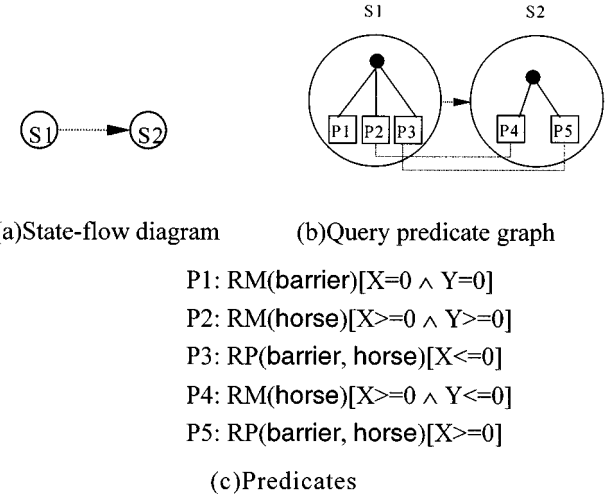


Fig. 9. Query predicate graph construction for Q12. (a) State-flow diagram. (b) Query predicate graph. (c) Predicates.

qco(). For example, the *qco*(QG of Q4) contains a single content object ball.

2) Video Elimination: The elimination process prunes videos which do not have all the content objects specified in the query predicate. It is performed by traversing the video class hierarchy. By evaluating the *maxcov*() and *mincov*() to the nodes in the video class hierarchy from the root downward, the number of nodes visited can be minimized.

Algorithm:

$VE\_processing$(Input: query predicate graph QG,
visiting node N; Output: candidate video set V)
if $(qco(QG) \subseteq mincov(N))\, V = member(N)$;
/* all the descendants of N contain qco(QG) */
else if $(qco(QG) \not\subset maxcov(N))\, V = \varnothing$;
/* none of the descendants of N contain qco(QG) */
else $V = \cup VE\_processing(\mathrm{QG}, n_i)$,
where n_i are the child nodes of N;
/* some descendants of N may contain qco(QG) */
return V.

In summary, the query is first parsed for constructing the query graph QG by its predicates. Second, the content objects of the query can be computed from the QG for video elimination. The resultant candidate videos are then generated for the video function evaluation which will be presented in the next section.

IV. Video Function Evaluation

After the video elimination preprocessing, a set of candidate videos are extracted. The candidate videos are further examined by evaluating the video functions of the query predicate for query results. In this section, we propose an efficient approach of function evaluation based on the examination of the behaviors of the content objects. We define various types of behaviors and analyze the video contents for the construction of the index of behaviors. When evaluating a video function, the video function can be classified into one of the behavior types. By applying the corresponding index, the videos containing qualified behaviors are extracted.

A. Behaviors of Content Objects

According to the CVQL, we discuss the types of behaviors in two ways: 1) the behaviors of a single content object and 2) the behaviors for two content objects.

There are three types of behaviors for a single content object.

- Static: A content object is static if it stays in a position for certain time duration, which can be described by {RM(O)[0, 0]}.
- Regular_moving: A content object is regular_moving if it is moving regularly, which can be classified into the following types.
  1) *Uniform motion:* A content object is moving with a constant speed, which can be described by {RM(O) $[X, Y]$, where X, Y, are constants}. For example, $X = 2$ represents that content object O is moving with speed = 2 along the X axis.
  2) *Directional motion:* A content object is moving in the same direction, which can be described by {RM(O)$[X, Y]$, where $X, Y \in$ {"+," "–"}}. The "+" represents content object O moving in the positive direction and the "–" represents content object O moving in the negative direction. The positive direction along the $X(Y)$ axis is from left(top) to right(bottom) and the negative direction from right(bottom) to left(top).
  3) *Accelerated motion:* A content object is moving with an accelerated speed, which can be described by {RM(O)$[X, Y]$, where $X, Y \in$ {"++," "—"}}. The "++" represents content object O moving faster and faster and the "—" represents content object O moving slower and slower.
- Random_moving: A content object is random_moving if it is moving but not regular_moving. It can be described by {AP(O)$[X, Y]$, where X, Y are in certain intervals}.

When extracting the behaviors for a content object from a frame sequence, it has the following steps.

1) Static examination: Check the content object if it satisfies the predicate of the static behavior.

2) Regular_moving examination: If the content object is not static, it is moving. Evaluate the content object if it satisfies the predicates of uniform motion, accelerated motion, or directional motion.
3) Random_moving examination: Content objects not classified into the static or regular_moving are classified into the random_moving behavior. In this case, the region the content object stays is computed.

The behaviors between two content objects are similar to the behaviors of a single content object. They represent the variation of the spatial relationships of two content objects. They can also be classified into three types of behaviors: static, regular_moving, and random_moving, as follows.

- Static: The relative positions of two content objects are fixed, which can be described by $\{\text{RP}(O_1, O_2)\,[X, Y]$, where X, Y are constants$\}$.
- Regular_moving: Two content objects are moving closer and closer or farther and farther, which can be described by $\{\text{RP}(O_1, O_2)\,[X, Y]$, where $X, Y \in \{$"+," "–"$\}\}$. The "+" represents content objects O_1 and O_2 are moving farther and farther, and the "–" for closer and closer.
- Random_moving: The relative position of two content objects are not constant and cannot be classified into regular_moving (i.e., these two content object are not moving farther and farther or closer and closer). It can be described by $\{\text{RP}(O_1, O_2)\,[X, Y]$, where X, Y are in certain intervals$\}$.

B. Properties of Behaviors

After the video is parsed, the behaviors of the content objects in the video are extracted. The attributes for representing these types of behaviors are shown as follows.

- Static: (OID, Duration, Position). There are three attributes. The OID denotes the ID of the content object which is static. The Duration denotes the starting and ending frames of this behavior. The position of the static content object is denoted in attribute Position. The value of the Position attribute is in the format of (X, Y), where $X, Y \geqq 0$.
- Regular_moving: (OID, Duration, Motion_type, Area, Speed_area). Since there are three types of motions, the Motion_type attribute is required for denoting the type of motion of the content object. The form of this attribute is (X, Y), where X, Y represent the type of the motion along the X and Y axes, respectively, as shown in Fig. 10. The Area attribute denotes the moving area. The Speed_area attribute denotes the range of the speed value of the moving. The notation is $(X_{\min} \sim X_{\max}, Y_{\min} \sim Y_{\max})$.

  The special case is when the Motion_type = 0 (it denotes a Uniform motion), the Speed_area is in the format of (X, Y) since the speed is constant. By examining the values of Motion_type and Speed_area, the details of the motion can be derived. For example, a regular_moving behavior is shown as: (Cat, (123 ~ 130), (++, 0), (0, 0)–(9, 5), (1–3, 1–1)). It denotes a cat moves to right faster and faster with the minimum speed 1 and the maximum speed 3 along the X axis, and keeps a constant speed 1 along the Y axis. The Area attribute denotes an area (0,0)–(9,5) as the position range of the content object in this duration. The duration of this behavior is from frame 123 to frame 130.

Possible value	Motion_type	Denotation
0	Uniform	Moving in a constant speed
+	Directional	Moving in the positive direction
−		Moving in the negative direction
++	Accelerated	Moving faster and faster
−−		Moving slower and slower

Fig. 10. Denotation of the Motion_type attribute.

- Random_moving: (OID, Duration, Area). The attributes of random_moving behavior are similar to the attributes of the static behavior. The only difference is the Area attribute. It is in the format of $(X_1, Y_1) \sim (X_2, Y_2)$ which represents the moving area of the content object.
- Behaviors of two content objects: (OID_1, OID_2, Duration, Type, Area, Distance). The Type attribute denotes the behavior type of the two content objects. The format of this attribute is (X, Y), where X, Y represent the type of the behavior along the X and Y axes, respectively, as shown in Fig. 11.

  The Area attribute denotes the range of the relative position of these two content objects by specifying the minimum and maximum differences along the X axis and Y axis. The Distance attribute denotes the corresponding minimum and maximum Euclidean geometry distances. If the behavior type is static, the value of the Distance attribute will be a constant.

C. The M-Index of Behaviors

The behaviors are used for the evaluation of video functions. In this section, we discuss various types of the predicate specifications by the video functions and present an index structure named *M-index* on the behaviors for efficient processing of the video functions.

- AP() function: the AP() is used for specifying the location of the content object. According to the specification of xy-expression, there are three cases.
  1) Xy-expression specifies constants: It queries for videos with a static content object. The static behavior is applied for the evaluation. The instances of the static behavior are hashed by the x and y values of the Position attribute. When the X, Y values are given, the qualified behaviors can be extracted.
  2) Xy-expression specifies an interval: It queries for videos with the content object bounded in an area. The frame sequences of the static, random_moving, and regular_moving behaviors may qualify the predicates. An R^+-tree [17] is constructed by the Position attribute of the static behavior and the

Possible value	Behavior type	Denotation
*	Random_moving	The relative positions are not constant
0	Static	The relative positions are constant
+	Regular_moving	The distance is increasing
−		The distance is decreasing

Fig. 11. Denotation of the Type attribute.

Area attribute of the random_moving and regular_moving behaviors. The query can then be evaluated by accessing the R^+-tree. The R^+-tree provides an index structure for efficient searching of objects in a on multidimensional space. Since the value of the Area attribute specifies a minimum boundary region of the content object, it can be indexed by the R^+-tree for efficient search.

3) Xy-expression contains Increasing or Decreasing modifier: The AP() function with Increasing or Decreasing modifier can be transformed to the RM() function. For example, AP(O)[X Increasing ^ Y Decreasing] is equivalent to RM(O)[$X > 0$ ^ $Y < 0$]. The evaluation is the same as the evaluation of RM() (as follows).

- RM() function: The RM() function is used to specify the motion of the content objects. According to the xy-expression, there are three cases.
  1) Xy-expression specifies constants: It queries for videos with a content object having a constant speed. The Uniform motion type behaviors can be applied for the evaluation. The behaviors with Motion_type = 0 are hashed by their Speed_area attribute. When the X, Y values are given, the qualified frame sequences can be extracted.
  2) Xy-expression specifies an interval: In this case, the qualified results may come from the behavior type of regular_moving or random_moving behavior. An R^+-tree is constructed by the Speed_area attribute of these behaviors. The query can then be evaluated by accessing the R^+-tree.
  3) Xy-expression contains Increasing or Decreasing modifier: In this case, the qualified results are from the behavior type of accelerated regular_moving. The query can be evaluated by the Motion_type attribute.
- AM() function: The AM() function is usually used for a range query. The xy-expression specifies the area the content object is located. Therefore, the R^+-tree constructed by the Area attribute of all the behaviors is applied for the evaluation.
- RP() function: According to the xy-expression, there are three cases.
  1) Xy-expression specifies constants: It queries for videos with two content objects having constant relative positions. The static type behavior of two content objects is applied for the evaluation. The instances of the static type behavior are hashed by the Area attribute. When the X, Y values are given, the qualified frame sequences can be extracted.
  2) Xy-expression specifies an interval: An R^+-tree is constructed by the Area attribute of all the behaviors of two content objects. The evaluation can then be processed by accessing the R^+-tree.
  3) Xy-expression contains Increasing or Decreasing modifier: In this case, the qualified results are from the regular_moving behavior. The query can be evaluated by the Type attribute.
- Distance() function: The evaluation of Distance() references the Distance attribute of the behavior of two content objects.

An M-index is constructed for a video to store the behaviors of its content objects. Fig. 12 shows the structure of the M-index, (a) for single content object, and (b) for two content objects, respectively.

D. Result Integration

For a query predicate, it may consist of some states and a state may consist of some subpredicates. A subpredicate is relevant to a video function. A frame sequence (or video) is a query result if it satisfies all the subpredicates in the query predicate. We have presented the evaluation of the video functions. In this section, we discuss the flows for the query processing, the evaluation of a state, and the integration of each state to form the query results.

The flow of the query processing is as follows.

1) Evaluating the video function for each subpredicate: As presented in Section IV-C, the qualified behaviors are stored in the form (OID(s), duration) for further processing. For example, a state consists of AP(tree)[$x = 2$^$y = 3$] and RM(car)[x Increasing ^ Y Increasing] with a duration of 30 frames. Fig. 13(a) shows the qualified behaviors of the AP() function, and Fig. 13(b) shows the qualified behaviors of RM().
2) Integrating the results of the subpredicates of each state: Since a state may consist of several subpredicates, the results of each subpredicate have to be integrated. The integration is performed by a join processing. It is based on the intersection of the durations of the qualified behaviors. For example, Fig. 13(c) shows the result of the join processing. After the join processing, the duration of 30 frames is checked, and the result shown in Fig. 13(d).
3) Integrating the results of all the states to generate the query result: For the Then modifier, it represents a state transiting into another state; for the Repeat modifier, the state is self-repeating. Therefore, the integration of the results can be performed by unioning the durations with the same OID when there exists a nonempty intersection of the durations. For example, we use Fig. 13(d) as the results of state 1 and the results of state 2 are shown in Fig. 14. The query result is shown in Fig. 15.

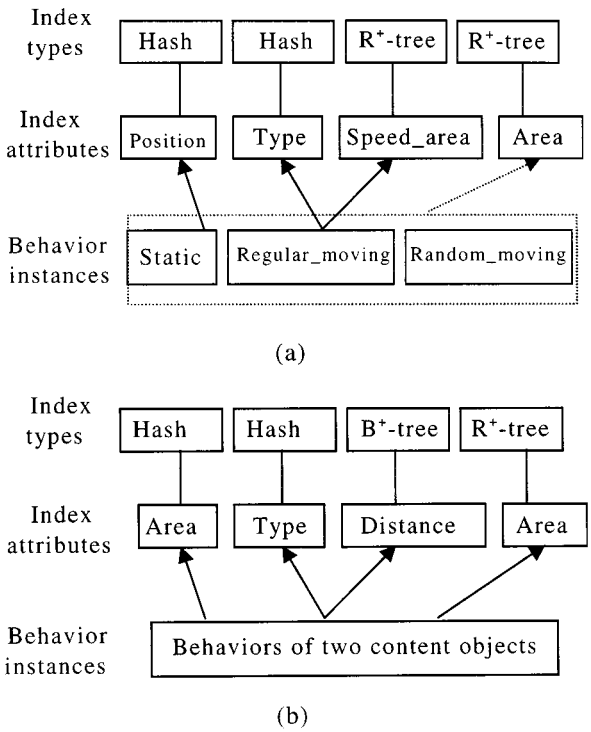


Fig. 12. The M-index.

OID	Duration
tree	(13-56)
tree	(67-99)
tree	(333-418)
tree	(525-571)
tree	(776-841)

(a)

OID	Duration
Car	(85-201)
Car	(254-412)
Car	(563-720)
Car	(863-931)

(b)

OID	OID	Duration
Tree	car	(85-99)
Tree	car	(333-412)
Tree	car	(563-571)

(c)

OID	OID	Duration
Tree	car	(333-412)

(d)

Fig. 13. Joining subpredicates of a state.

V. Performance Analysis

The processing of video query may cause high computation cost since there are thousands of videos in the databases and a video is composed of thousands of frames. In our approach, we consider two ways to accelerate the query processing: (1) reduce the number of videos which need for the video function evaluation, and (2) avoid frame-by-frame matching when evaluating the predicates. Therefore, the elimination-based preprocessing is performed for the former, and the behavior-based index and processing for the later. In this section, we discuss the performances of these two approaches.

A. Video Elimination

For analyzing the effect of the elimination, a set of parameters are defined.

OID	Duration
car	(1-52)
car	(97-232)
car	(395-580)

Fig. 14. Results of state 2.

OID	OID	Duration
tree	car	(333-580)

Fig. 15. Query result.

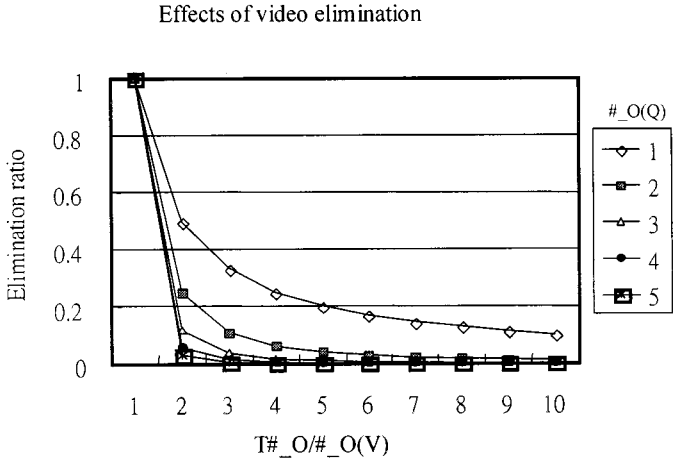


Fig. 16. Elimination ratios for the video elimination preprocessing.

1) $\#\_V$: The number of videos in the database.
2) $T\#\_O$: The total number of content objects in all videos.
3) S_O: The set of content objects in all videos.
4) $\#\_O(V)$: The number of content objects of a video V.
5) $\#\_O(Q)$: The number of content objects of a video query Q.

Assume there is one content object in the query predicate, the probability of video V containing this content object is equal to $\#\_O(V)/T\#\_O$. More generally, the probability is $C^{\#\_O(V)}_{\#\_O(Q)}/C^{T\#\_O}_{\#\_O(Q)}$. In this paper, the videos for the experiments are generated as follows: (1) the number of frames of a video is given between 9000 to 18000, (2) the video is divided into several clips each containing 30 to 300 frames, (3) according to the values of $\#\_O(V)$ and $T\#\_O$, the content objects are uniformly selected from S_O and assigned to each clip (when a content object is assigned to a clip, it appears in every frame in the clip). For video query Q, the content objects specified in the query predicates are also uniformly selected from S_O by the value of $\#\_O(Q)$. For the set of generated queries, the elimination-based query preprocessing is performed. The elimination ratio (the number of candidate videos/$\#\_V$) can then be computed. Fig. 16 shows the elimination ratios of the experiment where $\#\_V$ = 1000, $\#\_O(V)$ = 30, $T\#\_O$ = 30 ~ 300 (i.e., $(T\#\_O/\#\_O(V))$ = 1 ~ 10), $\#\_O(Q)$ = 1 ~ 5 (as shown in the legend).

From this experiment, we observe the elimination effect is good in two cases: (1) larger $\#\_O(Q)$, and (2) larger $T\#\_O/\#\_O(V)$. Since a query predicate usually contains two

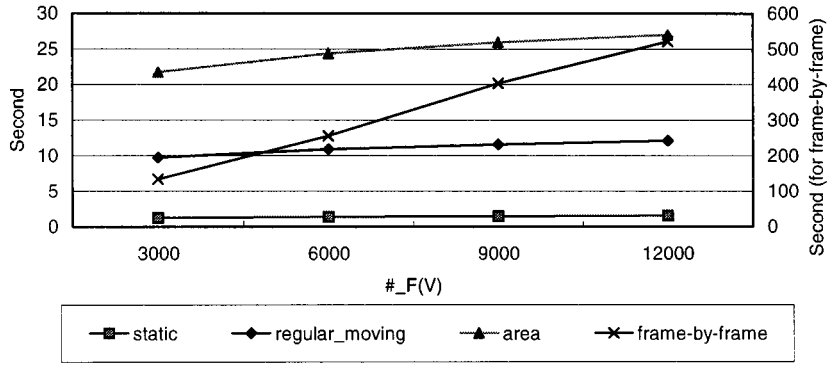


Fig. 17. Access time for processing a subpredicate on various values of $\#\_F(V)$.

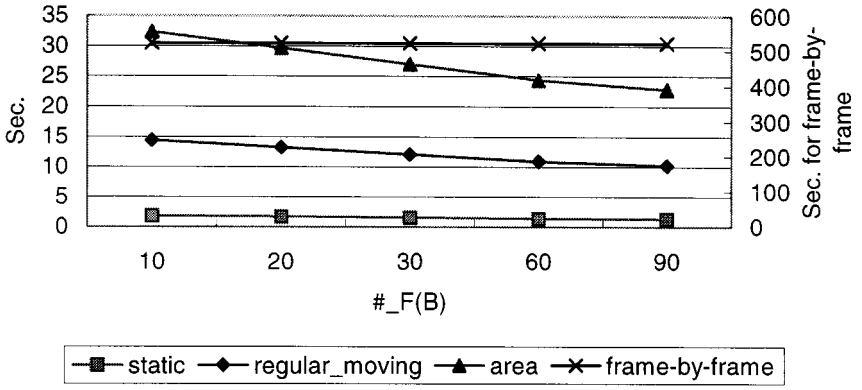


Fig. 18. Access time for processing a subpredicate on various values of $\#\_F(B)$.

or more content objects and the $T\#\_O$ is much larger than the $\#\_O(V)$ if the $\#\_V$ is large, the elimination effect will be good in real situations.

B. Video Function Evaluation

For frame-by-frame matching, the evaluation of video functions is performed by checking all the frames of the video. It causes a high query processing cost. By evaluating video functions on content object behaviors, the frame-by-frame processing can be avoided.

In this section, we discuss the improvement of query processing from the frame-by-frame evaluation approach to the behavior-based evaluation approach. Since the frame-by-frame evaluation approach and the behavior-based evaluation approach can both employ the video elimination preprocessing, we compare their performance on the time of processing a video.

As presented in the previous section, the query processing of a video consists of three phases: (1) find the qualified frame sequences (qualified behaviors) for each subpredicate, (2) integrate the qualified frame sequences of the subpredicates of each state, and (3) integrate the qualified frame sequences of each state. There are no differences for the frame-by-frame evaluation approach and the behavior-based evaluation approach in phases 2 and 3. Therefore, we analyze the processing cost of the phase 1 for these two approaches.

The frame-by-frame evaluation approach evaluates the temporal and spatial information of content objects of video frames in sequence. All the subpredicates of a query have to be examined for the query result. In this approach, different video functions spend the same amounts of time for the evaluation since the same temporal and spatial information of content objects of video frames is accessed. For the behavior-based evaluation approach, a subpredicate needs an access to the M-index for retrieving the qualified behaviors. Different types of behaviors will have different amounts of access time. We perform several experiments for the comparisons. The parameters are defined as follows.

1) $\#\_F(V)$: The number of frames of a video V.
2) $\#\_F(B)$: The number of frames of a behavior B.
3) For the R^+-tree index, the maximum entry of a node M is set to 6. The minimum entry of a node n is set to $M/2$ = 3. The distribution of the behavior types is assumed in uniform.

In this experiment, a set of 1000 videos are randomly generated. Three types of queries: static, regular_moving, and area types are performed. Fig. 17 shows the results of processing time of a subpredicate for all 1000 videos, with various values of $\#\_F(V)$. The value of $\#\_F(B)$ is assumed to be 30 frames. The access time of the frame-by-frame evaluation approach is much larger than the behavior-based evaluation approach. The former needs linear searching on the temporal and spatial information of content objects of video frames and the latter takes advantages of the M-index structure. The access time of the frame-by-frame approach refers to the right Y axis.

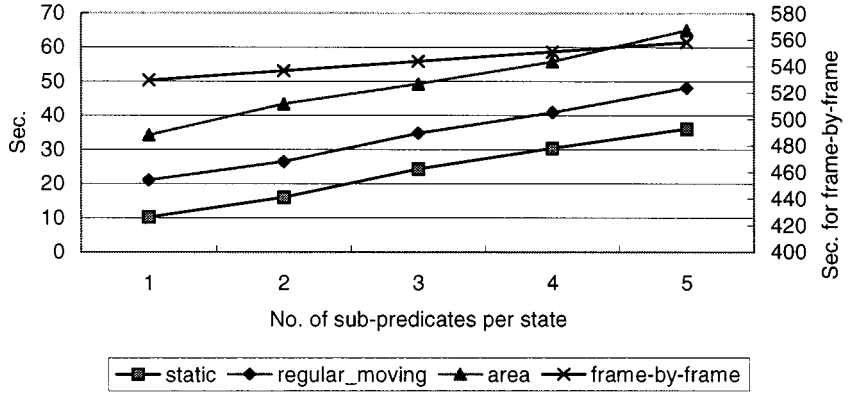


Fig. 19. The processing time for various numbers of subpredicates in a state.

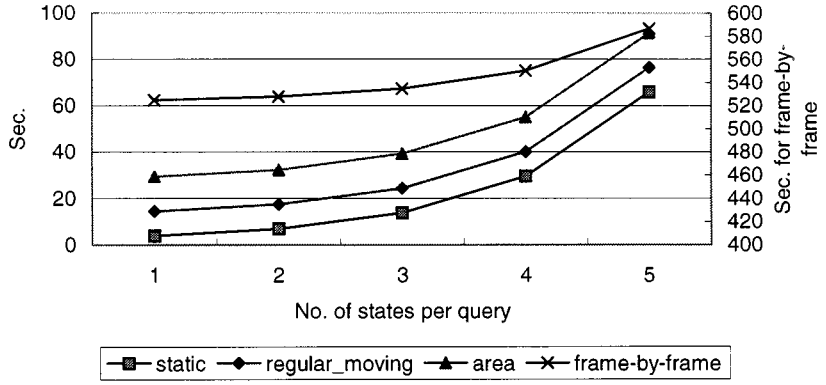


Fig. 20. The proccessing time for various numbers of states in a query.

The processing time for the frame-by-frame evaluation approach grows linearly with the values of $\#\_F(V)$. However, the processing time for the behavior-based evaluation approach is not affected much with the values of $\#\_F(V)$. For behavior-based evaluation approach, static type of queries perform the best due to the use of the hash index. The area type of queries perform the worst due to the need to access the R^+-tree.

Fig. 18 shows the results of processing time of a subpredicate of 1000 videos with various values of $\#\_F(B)$. The value of $\#\_F(V)$ is set to 12 000 frames. The access time of the frame-by-frame approach also refers to the right Y axis. For the behavior-based evaluation approach, the less the number of frames a behavior has, the more the number of behaviors a video has, which also implies the larger the size of the M-index is. Therefore, the access time decreases with the value of $\#\_F(B)$. The access time of the frame-by-frame evaluation approach is a constant since the value of $\#\_F(V)$ does not change. Moreover, the performance of the behavior-based evaluation approach is much better than the frame-by-frame evaluation approach.

In addition to the processing time for the integration of the qualified frame sequences of the subpredicates of each state and the processing time for the integration of the qualified frame sequences of all the states, the total query processing time for a video is computed. Figs. 19 and 20 show the experimental results on various numbers of subpredicates in a state and various numbers of states in a query, respectively. The value of $\#\_F(V)$ is set to 12 000 frames and the value of $\#\_F(B)$ is set to 30 frames. The probability of a subpredicate being qualified is set to 0.2. In the frame-by-frame evaluation approach, the number of tuples for the join processing is much smaller than the required accessing on the temporal and spatial information of content objects of video frames. It does not make much difference in the processing time. For the behavior-based evaluation approach, the more the number of the states or the subpredicates is, the larger the processing time is. Fig. 19 shows the growth of the processing time of the behavior-based evaluation approach is linear since the join processing of the durations of qualified frame sequences is in time complexity $O(N)$, where N is the number of the qualified frame sequences.

Fig. 20 shows the growth of the processing time of the behavior-based evaluation approach is $N\log(N)$ due to the integration processing of the durations of all the states.

VI. Conclusion

In this paper, we present a content-based video query language CVQL. Users retrieve video data by specifying the spatial and temporal relationships of the content objects. For the processing of CVQL, an elimination-based preprocessing for filtering unqualified videos and a behavior-based approach for video function evaluation are proposed. The content objects specified in the query predicate are used to eliminate the processing of the videos which do not contain these objects. The number of videos which need to be further evaluated can

then be reduced. For video function evaluation, the M-index provides a fast accessing of various types of behaviors of the content objects. The frame sequences qualifying the predicate can be efficiently retrieved. The frame-by-frame evaluation on video contents can be avoided.

The experiments for the elimination-based preprocessing and behavior-based video function evaluation are performed. The preprocessing can eliminate more than 90% of videos. The larger the number of the content objects in a query predicate is, the better the effect of the preprocessing does. Furthermore, the experimental results show the M-index can avoid the large number of accesses on the temporal and spatial information of the content objects. The processing is much more efficient than the frame-by-frame evaluation approach.

In this approach, the content object is represented by a point in the 2-D space. It is convenient for specifying the spatial relationships of the content objects. However, the topological relationships between content objects such as overlapping and the shapes of content objects are not supported. The extension for providing these capabilities is currently under our consideration.

REFERENCES

[1] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, Nov. 1983.

[2] N. S. Chang and K. S. Fu, "Query by pictorial example," *IEEE Transactions Softw. Eng.*, pp. 519–524, 1980.

[3] C.-W. Chang and S.-Y. Lee, "Video content representation, indexing, and matching in video information systems," *J. Vis. Commun. Image Represent.*, vol. 8, no. 2, pp. 107–120, 1997.

[4] S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic indexing by 2-D strings," *IEEE Trans. Pattern Anal. Machine Intell.*, pp. 413–428, 1987.

[5] Y. F. Day, S. Pagtas, M. Iino, A. Khokhar, and A. Ghafoor, "Object-oriented conceptural modeling of video data," in *Proc. IEEE Data Engineering*, 1995, pp. 401–408.

[6] R. Hjelsvold and R. Midsraum, "Modeling and querying video data," in *Int. Conf. Very Large Data Bases*, 1994, pp. 686–694.

[7] H. Jiang and A. K. Elmagarmid, "Spatial and temporal content-based access in hypervideo databases," *VLDB J.*, vol. 7, pp. 226–238, 1998.

[8] T. C. T. Kuo and A. L. P. Chen, "A content-based query language for video databases," in *Proc. IEEE Multimedia Computing and Systems*, June 1996.

[9] ——, "Content-based query processing and approximation for video databases," in *Proc. IEEE Knowledge and Data Engineering Exchange Workshop*, 1998.

[10] C. C. Liu and A. L. P. Chen, "3D-List: A data structure for efficient video query processing," *IEEE Trans. Knowl. Data Eng.*, to be published.

[11] J. Z. Li *et al.*, "Modeling of moving objects in a video database," in *Proc. IEEE Multimedia Computing and Systems*, 1997, pp. 336–343.

[12] T. D. C. Little and A. Ghafoor, "Interval-based conceptual models for time-dependent multimedia data," *IEEE Trans. Knowl. Data Eng.*, vol. 5, pp. 551–563, Aug. 1993.

[13] K. L. Liu, P. Sistla, C. Yu, and N. Rishe, "Query processing in a video retrieval system," *IEEE Data Eng.*, pp. 276–283, 1998.

[14] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, and D. Petkovic, "The QBIC project: Querying images by content using color, tecture, and shape," *Proc. SPIE*, vol. 1908, pp. 173–187.

[15] E. Oomoto and K. Tanaka, "OVID: Design and implementation of a video-object database system," *IEEE Trans. Knowl. Data Eng.*, vol. 5, pp. 629–643, Aug. 1993.

[16] E. G. M. Petrakis and S. C. Orphanoudakis, "Methodology for the representation, indexing and retrieval of image by content," *Image Vis. Comput.*, 1993.

[17] T. Sellis, N. Roussopoulus, and C. Faloutsos, "The R^+-tree: A dynamic index for multidimensional objects," in *Proc. 13th Int. Conf. Very Large Data Bases*, 1987, pp. 507–518.

[18] S. W. Smoliar and H. Zhang, "Content-based video indexing and retrieval," *IEEE Multimedia Magazine*, pp. 62–72, 1994.

[19] A. P. Sistla, C. Yu, and R. Venkatasubrahmanian, "Similarity based retrieval of videos," in *Proc. Data Engineering*, 1997, pp. 181–190.

[20] T. T. Y. Wai and A. L. P. Chen, "Retrieving video data via motion tracks of content symbols," in *Proc. ACM 6th Int. Conf. Information and Knowledge Management*, 1997.

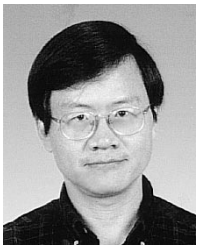
[21] R. Weiss, A. Duda, and D. K. Gifford, "Content-based access to algebraic video," in *Proc. Int. Conf. Multimedia Computing and Systems*, May 1994, pp. 140–151.

[22] A. Yoshitaka *et al.*, "Content-based retrieval of video data based on spatiotemporal correlation of objects," in *Proc. IEEE Multimedia Computing and Systems*, 1998, pp. 208–213.

[23] A. Yoshitaka and T. Ichikawa, "A survey on content-based retrieval for multimedia databases," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 1, pp. 81–93, 1999.

Tony C. T. Kuo received the B.S. degree from Tamkang University, Taiwan, R.O.C., in 1990, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1992 and 1998, respectively.

He is currently serving in the army as a Software Development Engineer. His research interests include content-based retrieval of multimedia databases, image and video indexing, and query processing for image and video databases.



Arbee L. P. Chen (S'80–M'84) received the B.S. degree in computer science from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1977, and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, in 1984.

He joined National Tsing Hua University, Hsinchu, as a National Science Council (NSC) sponsored Visiting Specialist in August 1990, and became a Professor in the Department of Computer Science in 1991. He was a Member of Technical Staff at Bell Communications Research from 1987 to 1990, an Adjunct Associate Professor in the Department of Electrical Engineering and Computer Science, Polytechnic University, Brooklyn, NY, and a Research Scientist at Unisys from 1985 to 1986. His current research interests include multimedia databases, data mining and mobile computing.

Dr. Chen has organized (and served as a Program Co-Chair) 1995 IEEE Data Engineering Conference and 1999 International Conference on Database Systems for Advanced Applications (DASFAA) in Taiwan. He is a recipient of the NSC Distinguished Research Award.