An Extended Video Database Model for Supporting Finer-Grained Multi-Policy and Multi-Level Access Controls

Nguyen Anh Thy Tran and Tran Khanh Dang

***Abstract*—The growing amount of multimedia data available to the average user has reached a critical phase, where methods for indexing, searching, and efficient retrieval are needed to manage the information overload. Many research works related to this field have been conducted within the last few decades and consequently, some video database models have been proposed. Most of the modern video database models make use of hierarchical structures to organize huge amount of videos to support video retrieval efficiently. Even now, among open research issues, video database access control is still an interesting research area with many proposed models. In this paper, we present a hybrid video database model which is a combination of the hierarchical video database model and annotations. In particular, we extend the original hierarchical indexing mechanism to add frames and salient objects at the lowest granularity level in the video tree with the aim to support multi-level access control. Also, we give users more solutions to query for videos based on the video contents using annotations. In addition, we also suggest the original database access control model to fit the characteristics of video data. Our modified model supports both multiple access control policies, meaning that a user may be affected by multiple polices, and multi-level access control, meaning that an authorization may be specified at any video level. Theoretical analyses and experimental results with real datasets are presented that confirm the correctness and efficiency of our approach.**

***Index Terms*—Video database security, video database model, content-based video retrieval, access control, multimedia database.**

I. Introduction

THE field of multimedia systems has experienced an extraordinary growth during the last decade. Among many visible aspects of the increasing interest in this area is the creation of huge digital libraries accessible to users worldwide. These large and complex multimedia databases must store all types of multimedia data, e.g., text, images, animations, graphs, drawings, audio, and video clips. Video information plays a central role in such systems, and consequently, the design and implementation of video database systems have become a major topic of interest.

With the huge amount of video information stored in archives worldwide, video databases have been researched for many years to introduce efficient ways to manage this kind of data. Below are some criteria that a video database should satisfy:

- The first thing that should be satisfied is how to organize efficiently raw video data. Videos are gathered from various sources with different formats so they need to be normalized to a standard form before being stored. In addition, these videos should also be compressed to reduce storage space because of their inherently huge sizes. Furthermore, video database also extracts key features such as key frames, salient objects, etc. to achieve high performance video content-based retrieval.
- Secondly, the video database access control scheme should be integrated with the database indexing structure in order that video database access control can be achieved more effectively. Since video database access control schemes should exploit semantic visual concepts and not low-level visual features, these database indexing units should correspond to the relevant semantic visual concepts.
- Thirdly, the flexibility and efficiency of transmitting video data through networks are an important consideration because most video databases are deployed over network environments.
- Finally, control over the security of a video database system is important. Videos can be illegally accessed while being transferred over the network, or accessed directly into the database. This is vital for the important video databases such as national video data stores.

To achieve the above requirements, this paper proposes a video database system that supports content-based retrieval and multi-level access control with different policies. This video database system is illustrated in Fig. 1. It contains two main components, *video analyzing* module and *query processing* module.

As can be seen in Fig. 1, the videos come from various sources with different formats so they firstly need to be analyzed. This step consists of three major tasks:

- Partitioning video into several video shot (shot boundary detection), extracting key frames and salient

Manuscript received May 11, 2008. Manuscript accepted for publication October 22, 2008.

This work was supported in part by Advances in Security & Information Systems (ASIS) Lab, Faculty of Computer Science & Engineering, HCMUT, Vietnam.

N. A. T. Tran is with the KMS Company, Ho Chi Minh City, Vietnam (e-mail: thytran@kms.com.vn).

T. K. Dang is with the Faculty of Computer Science & Engineering, HCMC University of Technology, VNUHCM, Ho Chi Minh City, Vietnam (phone:+84-98-3173334, e-mail: khanh@cse.hcmut.edu.vn).

objects of each shot (key-frame and salient object extraction).

– Classifying and clustering video shots.
– Indexing video database using semantic clusters.

These tasks are handled by the *video analyzing* component while the *query processing* component is responsible for controlling access to the database. The access control model should be flexible and reliable. This means that users should have multiple methods to retrieve the desired videos but they should only be able to access those to which they have been authorized.

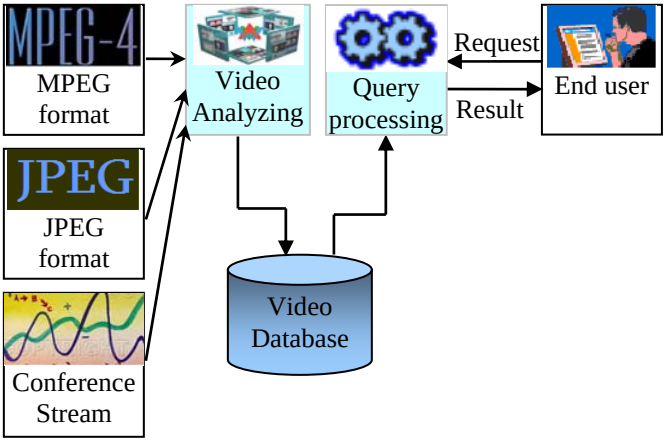


Fig. 1. A video database system structure.

The rest of this paper is organized as follows: In Section II, we briefly introduce the most crucial related work. In section III, after basics of video data processing is presented, we introduce a new hierarchical video database schema with more granularity levels. Section IV introduces the proposed algorithms to manage finer-grained access controls to the proposed video database. In section V, we present and discuss the implementation of a system prototype and experimental results of our proposed video database and access control model. Finally, section VI gives concluding remarks and introduces future work.

II. Related Work

Several efforts have been made to construct video database models to achieve flexible query and reliable access control. Basically, such efforts include the common data model for video information and hierarchical video data models [11], [2]. The first model did support content-based query using annotations, but was not suitable for large video databases since its structure was “flat”, meaning every video was at the same level. This restriction led to a problem in that users could not browse and navigate to find desired videos. In contrast, the hierarchical model proposed by Bernito *et al*. organized videos into semantic clusters within a tree structure. This helped to resolve the semantic gap between the low-level visual features and the high-level semantic visual concepts. However, this model lacked annotations and so although browsing requirements can be satisfied, retrieval options were not flexible.

With regards to video database access control, although there are some proposed models that support multi-level video access controls, on the whole, they do not allow users to specify authorizations at frame and object levels [1], [2]. In [2] and [3], Bernito *et al*. suggested a mechanism to manage access control to hierarchical video objects and another method to support multiple access control policies. However, combining them into a unique model is not a simple task because of authorization conflicts. Consequently, in this paper, we propose a full-fledged model that supports both a multi-policy and multi-level access control mechanism.

III. Video Database Models

In this section, after major video processing steps are presented, we introduce an extended storage model for video data that supports both semantic visual concepts clustering and flexible content-based retrieval. This newly introduced video database model can serve as a solid basis for materializing flexible access control mechanisms in a single video database system, which will be presented in section IV.

A. Video Processing

This paper does not intend to provide detailed information about video processing, but still we will provide some basic background information to offer a context for the proposal of a new video database model. Firstly, video formats are discussed in relation to compressed and uncompressed videos. Secondly, video shot detection methods to split a whole video into a sequence of meaningful video shots are presented. Thirdly, methods to extract video key features which will be utilized by users when searching through the database are introduced and finally, some video shot classification methods used to classify video shots into clusters are presented.

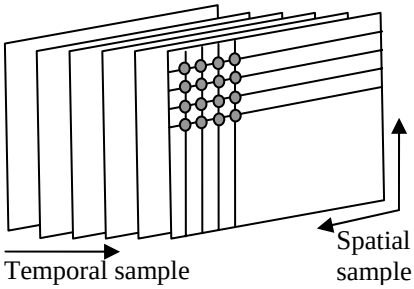


Fig. 2. Spatial and temporal sampling of a video sequence.

B. Digital Video Formats

Digital video is a representation of a natural (real-world) visual scene, sampled spatially and temporally (cf. Fig. 2). A scene is sampled at a point in time to produce a frame. Sampling is repeated at intervals, called spatial sampling, (e.g. 1/24 second intervals) to produce a moving video signal. In each scene, a frame is sampled at certain points, called pixels, positioned on a square or rectangular grid. At each pixel, stored information often includes its color like RGB (Red–Green–Blue) method or its color and luminance like YCbCr method [14], [15].

Videos are often compressed prior to being stored and decompressed before being displayed on the user screen. There are many video formats all using the CODEC model to compress and decompress videos [15]. A video CODEC (cf. Fig. 3) encodes a source image or video sequence into a compressed form and decodes this to produce a copy or approximation of the source sequence. If the decoded video sequence is identical to the original, then the coding process is said to be 'lossless'; if the decoded sequence differs from the original, the process is said to be 'lossy'. A video encoder consists of three main functional units: a temporal model, a spatial model and an entropy encoder.

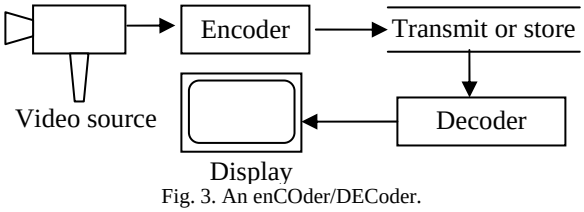


Fig. 3. An enCOder/DECoder.

The goal of the **temporal model** is to reduce redundancy between transmitted frames by forming a predicted frame and subtracting this from the current frame. The output of this process is a residual (difference) frame and the more accurate the prediction process, the less energy is contained in the residual frame. Fig. 4 illustrates the residual form of two adjacent frames. The obvious problem with this simple prediction is that a lot of energy remains in the residual frame (indicated by the light and dark areas) and this means that there is still a significant amount of information to compress after temporal prediction. Much of this residual energy is due to object movements between the two frames and a better prediction may be formed by compensating for motion between the two frames. To reduce this energy, we divide a frame into multiple *NxN* blocks and search for their movement directions called motion vectors. With this approach, the outputs of temporal model are the motion vectors and the residual forms of appropriate blocks belong to two frames. Consider the following example, there is a block (j, k) in the i^{th} frame which moves to the position (m, n) in the $(i+1)^{th}$ frame. If we subtract the whole frame *(i+1)* from frame *i*, the residual form at block (j, k) has high remaining energy because this block already moved to another location. In contrast, this energy is really small if we subtract block (m, n) of the frame *(i+1)* by block (j, k) of the frame *i* because they store the same object.

The function of the **spatial model** is to decorrelate further image or residual data and to convert it into a form that can be efficiently compressed using an entropy coder. The purpose of the transform stage in an image or video CODEC is to convert image or motion-compensated residual data into another domain (the transform domain). The choice of a transform depends on a number of criteria:

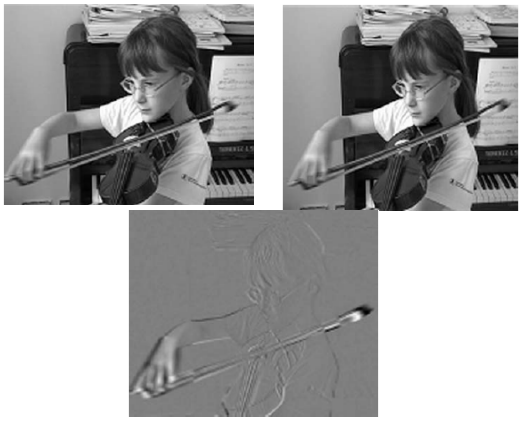


Fig. 4. Residual frame (the third one) of the first two frames.

- Data in the transform domain should be decorrelated (separated into components with minimal inter-dependence) and compacted (most of the energy in the transformed data should be concentrated into a small number of values).
- The transform should be reversible.
- The transform should be computationally tractable (low memory requirement, achievable using limited-precision arithmetic, low number of arithmetic operations, etc.).

The most transform ever-popular is Discrete Cosine Transform (DCT) [15]. The Discrete Cosine Transform (DCT) operates on **X**, a block of *N*×*N* samples (typically image samples or residual values after prediction) and creates **Y**, an *N*×*N* block of coefficients. The action of the DCT (and its inverse, the IDCT) can be described in terms of a transform matrix **A**. The forward DCT (FDCT) of an *N*×*N* sample block is given by:

$$Y = AXA^T \quad (1)$$

and the inverse DCT (IDCT) by:

$$X = A^T YA \quad (2)$$

where **X** is a matrix of samples, **Y** is a matrix of coefficients and **A** is a *N*×*N* transform matrix. The elements of **A** are:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N}, C_i = \sqrt{\frac{1}{N}}(i=0), C_i = \sqrt{\frac{2}{N}}(i!=0) \quad (3)$$

The output of DCT transform will be compressed using the **entropy encoder** which converts a series of symbols representing elements of the video sequence into a compressed bit stream suitable for transmission or storage.

C. Video Shot Boundary Detection (SBD)

The first step in indexing video databases (to facilitate efficient access) is to analyze the stored video streams. Video analysis can be classified into two stages [9]: *shot boundary detection* and *key features extraction*. The purpose of the first stage is to partition a video stream into a set of meaningful and manageable segments, whereas the second stage aims to abstract each shot using representative objects such as frames, salient objects, etc. The problem of shot boundary detection

will be addressed at this point while the problem of selecting key features from segmented shots will be addressed within the next section.

Shot boundary detection methods can be categorized into two main groups. The first one works on the uncompressed domain and the second works on compressed videos. Methods in the uncompressed domain can be broadly classified into five categories: *template-matching*, *histogram-based*, *twin-comparison*, *block-based*, and *model-based techniques*.

Within *template-matching* techniques, each pixel at the spatial location (i, j) in frame f_m is compared with the pixel at the same location in frame f_n , and a scene change is declared whenever the difference function exceeds a pre-specified threshold. Using *histogram-based* techniques, the histogram of a video frame and a difference function (S) between f_n and f_m are calculated using equation 4. If S is greater than a threshold, a cut is detected.

$$S(f_{m+1}, f_m) = \sum_{i=1}^{N} H(f_{m+1}, i) - H(f_m, i) \qquad (4)$$

The third method, *twin comparison*, uses two thresholds, one to detect cuts and the other to detect potential starting frames for gradual transitions. A different trend to detect shot boundary is called a *block-based* technique that uses local attributes to reduce the effect of noise and camera flashes. In this trend, each frame f_m is partitioned into a set of r blocks and rather than comparing a pair of frames, every sub-frame in f_m is compared with the corresponding sub-frame in f_n. The similarity between f_n and f_m is then measured. The last shot boundary-detection technique is termed *model based segmentation* where different edit types, such as cuts, translates, wipes, fades, and dissolves are modeled by mathematical functions. The essence here is not only to identify the transition but also the transition type.

On the other hand, methods for detecting shot boundaries that work in the compressed domain can broadly be divided into three categories. The first category uses DCT coefficients of video-compression techniques in the frequency domain. These coefficients relate to the spatial domain, and as such they can be used for scene change detection. The second category makes use of motion vectors. The concept here is that motion vectors exhibit relatively continuous changes within a single camera shot, while this continuity is disrupted between frames across different shots. The final category merges the above two trends and can be termed hybrid Motion/DCT. In these methods, motion information and the DCT coefficients of the luminance component are used to segment the video.

In summary, techniques that work upon uncompressed video data lack the necessary efficiency required for interactive processing. While other techniques that deal directly with compressed data may be more efficient, but they often lack reliability.

D. Key Features Extraction

Content based video indexing and retrieval requires that key features be extracted in the processing phase to improve query performance. These features include frame, salient object, audio, text, etc.

Key-frames are the represented images of a video in order that users can roughly understand the video without reviewing through its content. Key-frame extraction is closely related to shot boundary detection because to find out a shot bound, SBD algorithms usually search for the frame that has the largest differences compared to the previous one, while key-frame extraction methods detect the most unchanged frame inside each shot. It is cost saving to extract key frames at the same time as shot boundary detection. The other important feature, the salient object, is the key object displayed on the screen as long as the video shot. These are extracted from video shots and used for many purposes such as video shot classification, video retrieval and video access control.

Audio is one other important aspect of video data along with visual information. Audio can be kept as raw data and will be used to search for the stored video using its tune. However, analyzing audio is a poor performing process and so most systems, especially news video database systems, convert audio into text to reduce processing time when querying the videos. In conjunction with audio, text or captions often appear in videos so they too can be used for video classification and video retrieval efficiently because text processing is relatively faster than audio or video. When words have been collected, they need to be ‘cleaned up’ by some natural language processing algorithms to extract keywords and calculate their weights.

E. Video Shot Classifying

After the principal video shots and their visual features are obtained, we focus on generating higher-level visual concepts such as semantic clusters, so that more effective database indexing and access control scheme can be supported. Classification methods have been researched for a long age and there are many methods had been developed such as decision tree, k-nearest neighbor (kNN), Naive Bayes (NB), neural networks (NNet), support vector machines (SVM), etc.

The videos can be classified using its raw visual information such as visual data (color, brightness etc), audio data (tune, frequency etc) or higher level information likes texts or salient objects. To deal with visual and audio data that have a tremendous number of features, we should use methods like neural network or support vector machine who can work smoothly with large number of inputs. For example, we can use the pixels of labeled videos as inputs of a neural network to produce its weight vectors used to classify new unlabeled videos. The most important advantage of these methods is they can work on high dimensional input with acceptable time and quality. However, they are too difficult to understand for human because the result of training step is only a list of numbers.

On the other hand, decision tree or Naive Bayes are suitable for higher level classification because the number of inputs is relatively low in this case. These methods are quite simple and their training results are visual and understandable by human. More details of the above techniques are described in [9], [15], [17], [18], [16], [19], [5], [10], [6], [12].

F. Video Database Model

When very large video data sets are regarded, video database models and indexing can no longer be ignored if we want to support effective video retrieval and access control. In this section, we introduce a hierarchical indexing technique and its improvement to support multi-level access control.

1) Hierarchical Video Database Model

In order to control access efficiently, most video databases are designed as hierarchical structures such as the *semantic cluster tree* [2]. Within this structure, video contents are first partitioned into a set of semantic clusters; each semantic cluster is then partitioned into a set of sub-clusters and each sub-cluster may consist of a set of sub-regions. Using this indexing method, the system can handle multi-level access control efficiently. The indexing structure includes: a root hash table for keeping track of the information about all the clusters in the database; a leaf hash table for each cluster in order to record the information about all its sub-clusters; a second-leaf hash table for each sub-cluster in order to record the information about all its sub-regions and a hash table for each sub-region for mapping all its data points to the disk pages where the videos reside. To improve input/output efficiency, all semantic clusters are stored into a set of independent disks as shown in Fig. 5.

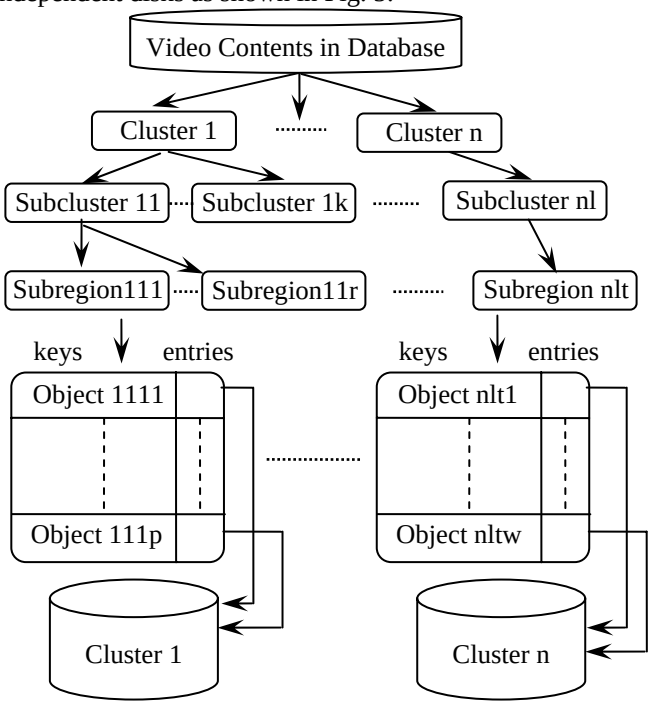


Fig. 5. The hierarchical video database partition and cluster-based indexing structure.

2) New Finer-Granular Hierarchy Video Database Model

The model described above has many advantages but it also has some limitations. Firstly, the only video unit supported is video shot while users are often interested in the whole video contains a certain shots. Secondly, the hierarchy tree is inflexible because in the case of extremely large databases, the tree level cannot be increased. Thirdly, this model cannot support access control at a frame and salient object granularity level. Finally, it looses most of the information needed for flexible content-based retrieval. Even though clusters are high semantic level extracted from other information, we still need to remain that information such as captions, audios, images etc. Given the above reasons, this article suggests a new model as illustrated in Fig. 6 to tackle these issues.

To address the first restriction, two new video levels are introduced; *video* and *scene*, meaning that a complete video may contain some scenes and a scene contain some shots. With this enhancement, a video database administrator can specify authorizations at video (most often), scene and shot levels. This article also proposes to modify the original hierarchy of the video tree to use video groups which consist of videos or other groups instead of clusters, sub-clusters and sub-regions. With this amendment, the path from root to the leaves can be controlled with greater flexibility where new groups can be introduced or existing groups removed.

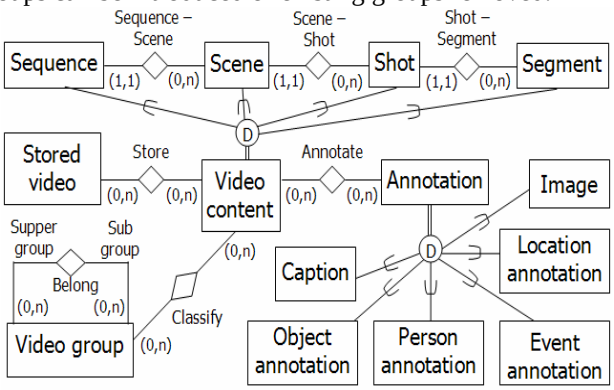


Fig. 6. The extended video database diagram.

Along with the two above amendments, it is suggested that a new video element at the lowest granularity level called *video segment* be introduced. This element would prove very useful when applying access control to a frame or a salient object. Consider the example in Fig. 7, where there are two users *A* and *B* within the system. A policy applies to user *A* that specifies that this user cannot view two frames $(j+1)^{th}$ and $(j+2)^{th}$ of video shot *V*. In addition, there is another policy that restricts user *B* from seeing object named *XXX* of video shot *V*. The easiest way to handle these two policies is to copy the whole video shot *V* to two appropriate versions. However, this solution will impinge on memory space since video data is often huge. Using segments is another solution which splits the video shot to certain segments and then copies the segments only when needed. In this example, the video shot *V* is split into 5 separate parts: (1) from the beginning to frame

j^{th}, (2) frames *j+1* and *j+2*, (3) from frame $(j+3)^{th}$ to frame i^{th}, (4) frames *(i+1)* and *(i+2)*, (5) from frame $(i+3)^{th}$ to the end. With this solution, we only need to copy the segment 4^{th}, which contains only two frames, into two versions: version *#1* with original *XXX* objects, and version *#2* with blurred *XXX* objects. Then, when user A requires this video shot, the system will display the 1^{st}, 3^{rd}, 4^{th}- version #1 and 5^{th} segments while user B sees 1^{st}, 2^{nd}, 3^{rd}, 4^{th} –version #2 and 5^{th} segments.

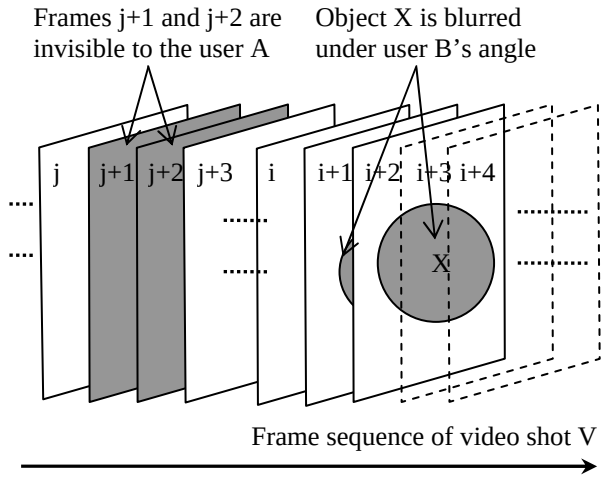


Fig. 7. An example concerning the segments.

The final adjustment is related to annotations. Since information in videos is quite "raw" and dispersed, it is almost impossible to achieve *semantic* content-based access to videos unless some additional information is available. In order to enable flexible and intelligent access to videos, we somehow need to extract "keywords" which describe semantic contents of videos. Typically "keywords" are useful for semantic content-based access to videos include information on:

- what/who appears in the video,
- when the video was broadcast/recorded,
- where the video was recorded,
- what the video is about, etc.

In order to achieve this goal, more annotation is required such as object annotation, person annotation, location annotation, event annotation, caption and image. The first four annotations lend themselves naturally as annotations since they answer four key questions *who*, *what*, *when* and *where* about a video. Caption annotation is broadly used in news video databases where this kind of information exists on almost video news. A video database application rarely uses image annotation because of poor image processing performance. However, it is utilized in some special video databases such as airport and gas station security systems to scan for unusual baggage and terrorist activity.

IV. Flexible Access Control Model

In this paper, a content-based access control model is suggested which is reliant upon high level features extracted during the video processing stage. The goal of the system is to provide a flexible framework that can support different security levels against the video database.

The architecture of the system is illustrated in Fig. 8. There are three main components within this architecture: *authorization*, *query engine* and *authorization management*. The *authorization* component is responsible for filtering authorized videos while the *query engine* searches for interesting videos that a user has requested. The final component, *authorization management*, handles granting permissions and ensures the consistency and integrity of the video database system.

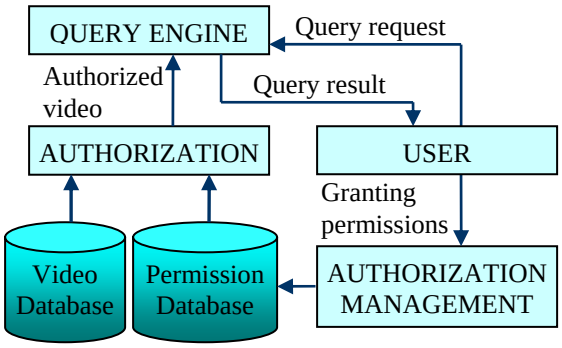


Fig. 8. Video database flexible access control architecture.

A. Authorization Model

In this section, an authorization model based on a flexible authorization model suggested by Bertino *et al.* in [2], [3] is introduced. The proposed model provides both multiple access control polices and a multi-level access control mechanism. This model also allows the administrator to specify multiple authorizations over any users or user groups (named *subject of the authorization*) against any video level such as videos, scenes or video shots.

1) Notation and Definitions

This new model manages access control via authorizations. The subject of each authorization is a user or a user group. A group can contain some users and/or other user groups. The relationship between a subject *s* and a user group G_k can be either *direct* or *indirect* [2]. If *s* is a member of G_k, we count this relationship as direct, written $s \in_1 G_k$. In contrast, the relationship is indirect, written $s \in_n G_k$, $n > 1$, if there exists a sequence $<s_1, s_2, \ldots, s_{n+1}>$, such that $s_1 = s$, $s_{n+1} = G_k$ and $s_i \in_1 s_{i+1}$, $1 \le i \le n$. The sequence $<s_1, s_2, \ldots, s_{n+1}>$ is called a membership path of *s* to G_k, written $mp(s,G_k)$. Let $MP(s, G_k)$ represent a set of memberships of *s* to G_k, either direct or indirect.

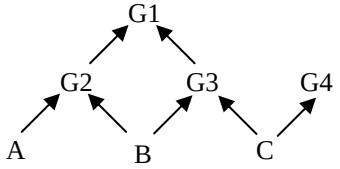


Fig. 9. An extended tree of users and user groups.

In a similar manner for users, the video content of our system is also organized into an extended tree structure. Let *V*, *VG* and *VO* represent the videos, video groups and video

objects (frames or salient objects) respectively. A video group can contain videos and other video groups. We use $v \in_k vg$ to denote the relationship where v belongs to vg with a relationship type that is direct ($k = 1$) or indirect ($k > 1$). We also use $mp(v,vg)$ to represent the membership path of v to vg and $MP(v,vg)$ stands for the set of all paths of v to vg.

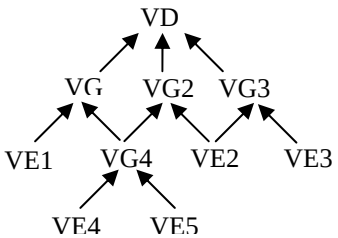


Fig. 10. A sample hierarchical video database.

Authorization handles whether a user or user group can access (*positive authorization*) or cannot access (*negative authorization*) a video element. The authorization model within this article is based upon the multi-level video access control model described in [3]. However, within this new system, the target of each authorization can be *a node on the video content tree* instead of a single table. Also supported are two kinds of authorization called *hard* (authorization that cannot be overridden) and *soft* (authorization that can be overridden). For example, the authorization that an under 18 years of age user not be able to view some specific shots of the videos without any exception should be a hard one.

Let U denote all users, G the set of user groups, $S = U \cup G$ the set of all subjects, V the set of video contents, VG the set of video groups, $VD = V \cup VG \cup VO$ the set of all video elements, AZ the set of all authorizations in our system. Authorizations can be defined as follows.

Definition 1 (Authorizations): An authorization is a 5-tuple of the form (s, v, pt, g, at) where $s \in S, v \in VD, pt \in (+, -), g \in U, at \in \{soft, hard\}$.

The authorization states that s has been granted (if pt = "+") or denied (if pt = "-") access permission on video element v by user g with authorization type is *at (soft* or *hard)*. For example, the tub *(A, VG4, +, B, hard)* means the user *B* has granted access permission on video group *VG4* to user *A* with authorization type is *hard*. Given an authorization a, let $s(a), v(a), pt(a), g(a), at(a)$ denote the subject, target, access type, grantor and authorization type, respectively.

Since a user can belong to a number of different user groups, he or she can be affected by multiple authorizations and some of them have opposite access types over a video element. It is the reason why we need to define the rules to decide which authorization has more priority than the others in case the conflict happens. Our overriding authorization model is a user-driven one means it prioritizes the authorizations based on the relationship between their subjects. The authorization has a more detail subject will have higher priority.

Definition 2 (Overriding authorization): Consider p_i and p_j are two different authorizations, p_i overrides p_j over user s against video element ve, written $p_i >_{s,ve} p_j$, $s \in_m s(p_i)$, $s \in_n s(p_j)$, $m,n \geq 0$, $ve \in_l v(p_i)$, $ve \in_k v(p_j)$, $l, k \geq 0$, iff any of the following conditions is satisfied:

- $at(p_i) > at(p_j)$, means $at(p_i) = hard$ and $at(p_j) = soft$
- $at(p_i) = at(p_j)$ and $s = s(p_i)$, $s \mathrel{!}= s(p_j)$
- $at(p_i) = at(p_j)$ and ($\forall mp \in MP(s, s(p_j))$: $s(p_i) \in mp$ or $\exists s' \in mp, \exists p' \in AZ, s' \neq s(p_j), s' \notin_k s(p_i), p' >_{s',ve} p_j$)

The above definition can be explained as the followings:

- p_i *override* p_j if the authorization type of p_i is *hard* while p_j's authorization type is *soft*.
- p_i override p_j if p_i and p_j have the same authorization type and p_i is applied over s directly while p_j is not.
- p_i override p_j if p_i and p_j have the same authorization type and for all membership path mp of s to $s(p_j)$, either $s(p_i) \in_k mp$ or exists $s' \in mp$, $p' \in AZ$ and p' override p_j over user s against video element ve.

Example 1: Consider a video database that contain the below set of authorizations:

p1: (G1, VG1, -, C, 1, soft)
p2: (G1, VG4, -, C, 1, hard)
p3: (G2, VG1, +, C, 1, soft)

where *G1*, *G2*, *G3*, *C* are users and user groups in Fig. 9 and *VG1*, *VG4* are video elements in Fig. 10. In this example, user *A* is affected simultaneously by *p1*, *p2* and *p3* authorizations. From *p1*, *A* can access *VG1* whereas *p1* does not allow *A* to access *VG1*. Because *G2*, subject of *p3*, has more detail than *G1*, subject of *G1*, so *p3* overrides *p1* over user *A* and video element *VG1*. In addition, *p2* authorization is a hard one so it will override all other authorization, including *p3*. In this example, user A can only access *VE1*.

Definition 3 (Conflict): Two authorizations p_i and p_j are conflict with respect subject s and video element v, written $p_i <>_{s,v} p_j$, with $s \in_m s(p_i)$, $s \in_n s(p_j)$; $v \in_l v(p_i)$, $v \in_k v(p_j)$; $i, j, l, k \geq 0$, iff $pt(p_i) \neq pt(p_j)$ and neither $p_i >_{s,v} p_j$ *nor* $p_j >_{s,v} p_i$.

In our system, we avoid any conflict by checking any actions that may cause the conflict. The detail of this task will be described in section C.

2) Authorization Algorithm

To make sure the users can only view video contents they allowed to access, we suggest an algorithm to retrieve the appropriate videos based on the actor and the set of authorizations in the system. Firstly, we define some more definitions will be used in this section.

Definition 4 (Video database projection): The projection of a video database VD with respect to a video element ve, written $\prod_{ve}(VD)$, is a set of video ve_i such that $ve_i \in_k ve$, $k \geq 0$. It means $\prod_{ve}(VD)$ only contains the child nodes of ve in the hierarchical video tree.

$$\prod_{ve}(VD) = \{v: v \in VD, v \in_k ve, k \geq 0\} \qquad (5)$$

Definition 5 (Video database prune): Consider a set of videos $VS = \{ve_1, ve_2, \ldots, ve_n\}$, the result after VS is pruned,

written $\angle(VS)$, is a set contains the elements of *VS* and each one is not a child of any other elements inside VS. It can be described formally as follow.

$$\angle(VS) = VS - \{v_i: v_i \in VS, \exists v_j \in VS, v_i \in_k v_j, k > 0\} \quad (6)$$

The purpose of the prune operator is to filter the nodes and to keep only nodes at highest levels in the tree. We define this operator because if a user can access a video element, he or she will be able to access all its children in the video tree. For instance, $\angle\{VE2, VE4, VE5, VG2, VG3\} = \{VG2, VG3\}$.

Next is the algorithm to filter the list of video contents that a user can access. First of all, it will get all video elements granted to the user by positive authorizations. Then, it collects the video elements that are inaccessible to that user. This list contains all video elements that were granted by negative authorizations except the video contents that the negative authorization is overridden by a positive one.

ALGORITHM 1. FILTER VIDEO CONTENTS THAT A USER CAN ACCESS

```
METHOD authorizeVideo(u)
  initialize AV to be empty
  let pos_permission and neg_permission are lists of
  positive and negative permissions respectively
  let UV is a list of videos that the user cannot access
  let TV is a temporary list of videos
  for each permission p ∈ P do
    if (u ∈k s(p)) then
      if (pt(p) = +) then
        add p to pos_permission
      else
        add p to neg_permission
      endif
    endif
  endfor
  for each permission p+ ∈ pos_permission do
    AV = AV ∪ v(p+)
  endfor
  for each permission p- ∈ neg_permission do
    uv = v(p-)
    for each permission p+ ∈ pos_permission do
      TV = ∠(∏v(p+) ∩ ∏v(p-))
        for each video element ve ∈ TV do
          if (p+ >u,ve p-) then
            uv = uv – ve
        endif
      endfor
    endfor
    UV = UV ∪ uv
  endfor
  AV = ∠(AV – UV)
  return AV
END AUTHORIZEVIDEO
```

Fig. 11 illustrates the meaning of this algorithm. In this figure, *AV* represents all video elements the user was granted access permission. *UV* represents the video elements the user was denied access permissions. $TV = AU \cap UV$ represents the video elements belong to both accessible and inaccessible ones. $v_i \in TV$ is a video element that was granted by the positive permission *p+* and negative permission *p-* and *p+* override *p-*. Finally, that user can access all videos belonging to orange parts (left).

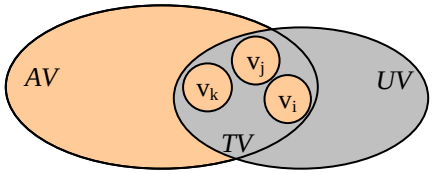


Fig. 11. An illustration of algorithm 1.

B. Query Engine (Video Retrieval)

This component collects requests from end users and searches through the authorized videos to retrieve those relevant and returns them to the users.

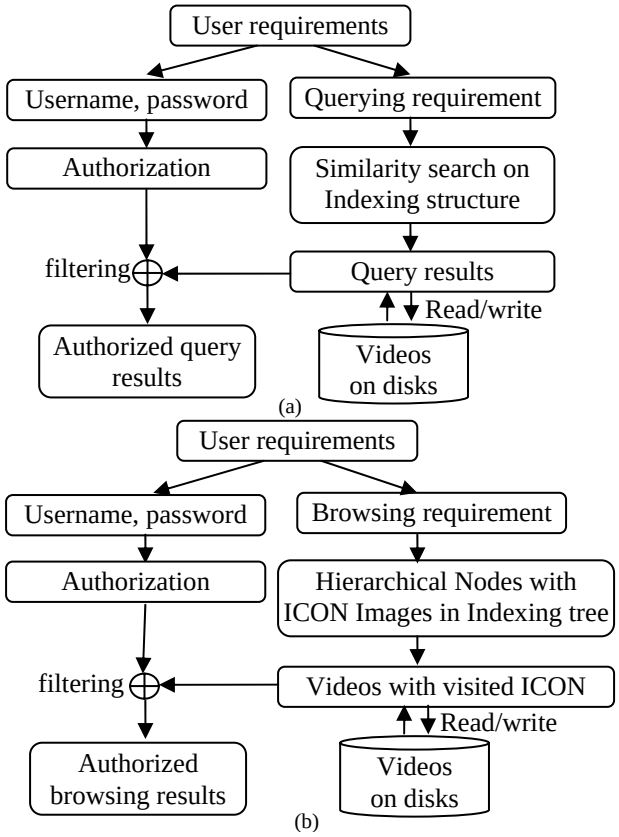


Fig. 12. The access control model under:
(a) querying mode and (b) browsing mode.

The query engine must be reliable, meaning that users can only access those videos to which they have been granted permission. The component must also be flexible, in order to support various means for the users to reach their interesting videos. Bertino *et al*. [2] suggested a system to support two access methods named *querying* and *browsing*. Under the querying method, users request specific video shots based on some criteria. By contrast, under browsing mode, users browse through and navigate the video database through its semantic categories. Based on the previous result, we

introduce two adapted algorithms for the same problem with respect to our newly proposed video database schema. Fig. 12a and 12b illustrate the phrases in querying and browsing mode respectively. As can be seen from the diagrams, both algorithms include an authorization phase as described previously in section A. Next, we will present the two access methods described above in more detail.

1) Querying Mode

Under the querying mode access control, a user submits a query to require the access to a video element. A query is a n-dimensional tuple $(x_1, x_2, \ldots, x_n)$ of which x_i, $i = 1..$ n is a value of the i^{th} feature. Below is the algorithm to retrieve video elements based on the features input.

ALGORITHM 2. QUERYING ACCESS CONTROL MODE

INPUT:
User ID
A query with $(x_1, \ldots, x_n)$ format where x_i is a feature's value
OUTPUT:
AIV (authorized interesting video) – set of authorized filter video elements or
ACCESS_DENIED if there is no video matches the query
METHOD *queryVideo(u, $(x_1, \ldots, x_n)$)*
AV = authorizeVideo(u)
if (*AV* is empty)
return *ACCESS_DENIED*
else
AIV = solve_query(request, AV)
if (*AIV* is empty)
return *ACCESS_DENIED*
else
return *AIV*
endif
endif
END queryVideo

This access control procedure consists of two main steps:

(1) Firstly, it narrows the search space by filter a list of videos the user can access;
(2) Secondly, it calculates all matching ranks between each video element in the *AV* list and the input query. Then, the system returns the result which will be videos ordered by their similarities with the input query.

This is a change compare to the original algorithm introduced in [2]. The original algorithm calculates matching ranks first then filters the list based on authorization rules. Here we have reversed this sequence in order to reduce the search space as soon as possible.

'Solve_query' takes $x = (x_1, x_2, \ldots, x_n)$ as an input and calculates the similarity between the x vector and each authorized video. Then, it only keeps N videos which have the highest similarity measures which exceed a predefined threshold. The most popular features used for searching would be the video group, location (the 'where'), produced date (the 'when'), related persons (the 'who'), texts, pictures and audios (the 'what'). These features were extracted from the contents of videos while they were being processed. This is the reason why the access model presented is called a *content-based* video retrieval model. Each feature may be defined a weight representative of its importance level compared to others. Similarity between a video v and a feature vector x is defined as below.

$$rank(v, x) = \sum_{i=1}^{N} match(v_i, x_i) * w_i \quad (7)$$

where v_i represents for the i^{th} feature of video v and w_i is x_i's weight.

Matching video group, location, date, person features are quite simple since they are obviously matched (*match = 1*) or unmatched (*match = 0*). For example, if video v belongs to sport group then *match(v, 'sport')* = 1. In contrast, matching text, image and audio features is difficult since they require text processing, image processing and audio processing knowledge, respectively.

When extracting text, it is split them into words and only the keywords s are stored. That is, words appearing more frequently than a given threshold. To calculate the similarity between an input string containing n keywords $\{k_1, k_2, \ldots, k_n\}$ and a video v, we use the formula below.

$$match(s, v) = \sum_{i=1}^{n} count(k_i, v) * w(k_i) \quad (8)$$

where $count(k_i,v)$ returns number of appearance of k_i word inside video v and $w(k_i)$ is weight value of k_i.

For the produced date, the matching value is bigger when the video is newer and vice versa. Give d is the interesting date that a user wants to query the videos, the distance between a video v and d is calculated as below.

$$match(d, v) = \frac{| d - d_v |}{\max date - \min date} \quad (9)$$

where v_d id the date when v is produced, *max_date* is the produced date of the newest video and *min_date* is the produced date of the oldest video in the search space.

There are some variants of audio retrieval methods such as *query by keywords* and *query by examples*. ***Query by keywords*** applies to audio content and basically follows the same approach used in traditional text based information retrieval. The user provides several keywords as search terms, and the query engine compares them with the textual information attached with audio files in the database to determine the list of returns. ***Query by example*** is a more natural way for retrieving audio content. For example, suppose we are looking for a music masterpiece. We have no clue of the title, but we have a short portion of it, for example, a 10 second clip. We can use this piece of audio sample, normally in the format of a file, as a query object. The search engine analyzes the content of query example, computes acoustic features, compares the audio content with audio files

in the database, and then generates search results accordingly. The major issue of this query method is how to speed up the search procedure.

Some modern video databases support image searching, especially faces tracking. For example, camera systems in airports are responsible for detecting unusual objects such as unattended bags or terrorist activity. These systems must discover the above items in quick time in order that security guards will have the ability to react. To improve searching performance, the data needs to be prepared offline using machine learning methods like neural network and support vector machines, etc.

2) Browsing Mode

Under the browsing access control mode, a user browses and navigates through video groups without specify searching criteria. Browsing refers to a technique or a process where users skip through information rapidly and decide whether the content is relevant to their needs. Browsing video databases should be like scanning the table of contents and indices of a book, or flipping through the pages, to quickly get a rough idea of the content and gradually focus on particular chapters or sections of interest. We believe the proposed semantic clustering technique and cluster-based hierarchical indexing structure would be very suitable for such fast browsing.

C. Authorization Management

The main purpose of authorization management component is to maintain the consistency and integrity of the system. It is responsible for validating all actions that may cause unsolvable conflicts to occur. Consider the example used in definition 3, where two authorizations p1 and p2 are conflict if exist a video element *ve* and a user *u* affected by them and neither $p1 >_{u,ve} p2$ nor $p2 >_{u,ve} p1$.

With two types of authorization–*soft* and *hard*, we may have three kinds of relationship between the authorizations: *hard–hard*, *hard–soft* and *soft–soft*. The second relationship (*hard-soft*) cannot be a conflict because a hard authorization always overrides a soft one. In addition, to prevent the conflicts between hard authorizations, this newly proposed system would only accept negative hard authorization. This means all positive authorizations have a soft property.

$$\forall(p) \in P,\ pt(p) = + \Rightarrow at(p) = soft \qquad (10)$$

This restriction is quite natural because we might often prohibit a user from accessing to some kinds of videos and rarely do we force a user to always access some particular videos.

Finally, there is only the last relationship, *soft – soft*, needed to be verified for conflicts. Below are four actions of an administrator that may cause a conflict to occur:

- Adding a new authorization.
- Adding an existing user subject to a group.
- Adding an existing video element to a group.
- Deleting an existing authorization.

To support checking the consistency of the system, we define a new term named *general conflict* as follows.

Definition 6 (General conflict): Two authorization *p1* and *p2* are generally conflict, written *p1 <> p2* if exists at least one video *v* and one user *u* such that $p1 <>_{s,v} p2$.

For each kind of action, we suggest a different algorithm to check conflict individually.

1) Check Conflict when Adding a New Authorization

When a new authorization *p(s, v, pt, g, at)* is added to the system, a conflict may occur over children nodes of *s* in the user tree. Consequently, the system must verify the conflict between *p* and each authorization *p'* affects any children of *s*.

ALGORITHM 3. CHECK CONFLICT WHEN ADDING A NEW AUTHORIZATION

```
INPUT:
  Authorization p: (s, v, pt, g, soft)
OUTPUT:
  True: if the system still is consistent means there is no
   conflict happens
  False: otherwise
METHOD checkNewPermission(p)
  PP = empty
  for each s' ∈ ∏(s)
      for each p' ∈ P
          if (pt(p')≠pt(p)) and (∏v(p')∩∏v(p)≠ϕ) and
           (s'∈ks(p'))
              PP = PP ∪ p'
          endif
      endfor
  endfor
  for each p' ∈ PP
      if p' <> p
          return False
      endif
  endfor
  return True
END checkNewPermission
```

The first step in the above algorithm is to collect a list of authorizations needed to be verified for conflict against *p*. This list includes all authorizations *p'* which i) has opposite access type compared with *p*, ii) *p* and *p'* affect to at least one video element, iii) subject of *p'* is an ancestor of *s'* or its children. The second step verifies conflict of each authorization in the above list against *p*. This algorithm will return *False* whenever a conflict occurs. Otherwise, it returns *True* meaning the new added authorization is valid.

We will use Fig. 12 to explain the algorithms in this section and the next two sections. The video database in this figure contains five existing authorizations listed {*p1, p2, p3, p4, p5*}. When adding a new authorization *p6* which restricts the permissions of *G5* over *V1*. Based on the algorithm, *PP* contains three authorizations {*p1, p2, p4*} needed to be verified conflict with *p6*. Authorization *p3* does not belong to this list because it has the same access type as p's. We also don't need to verify *p5* because it and *p6* affect to two disjoin

video set. On completion, the algorithm returns *False* because there is one conflict between *p6* and *p4* over user *D* and video group *V1*.

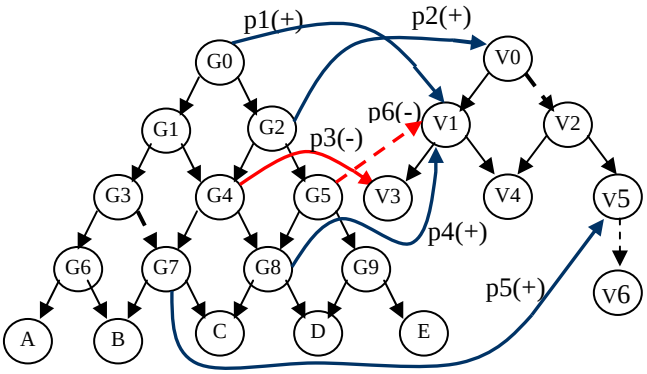


Fig. 13: A video database with some authorizations.

We are now proving that our algorithm is correct and sufficient. Assume that the algorithm is wrong, meaning there exists an authorization named *p'*: $p' \notin GP$ and $p' <> p$. Another assumption is that there is an authorization *p'* conflicts with *p* and the system still is consistent.

With the first assumption, because $p' \notin GP$ we infer that $\prod s(p') \cap \prod s(p) = \varnothing$ or $\prod v(p') \cap \prod v(p) = \varnothing$. This means *p* and *p'* have two separated affected spaces. Therefore, they cannot conflict with each other and hence, this assumption is incorrect.

With the second assumption, let *(u,v)* be a pair of user and video where the conflict happens between *p* and *p'*. The system still is consistent means there is at least one authorization p_1 that satisfies $p_1 >_{u,v} p$ or $p_1 >_{u,v} p'$. If p_1 overrides *p* over *(u,v)*, we can infer that *p'* also overrides *p* over *(u,v)* based on the last item in the authorization definition: $\forall mp \in MP(u,s), \exists u \in mp, p_1 >_{u,v} p$. Similarly, if p_1 override *p'*, we can also infer that *p* overrides *p'*, too. Anyway, *p'* and *p* are not conflict so this assumption is not correct.

2) Check Conflict when Adding an Existing User Subject to a Group

When adding an existing user or user group *s* to other user group *g*, *s* will inherit all authorizations affect to *g*. Thus, we need to check conflict between a set contains the authorizations affect to *g*, named *GP*, and another set *SP* contains the authorizations affect to *s* and its children. Naturally, this algorithm collects the authorizations of *GP* and *SP* first and then checks conflict between every each pair in those two sets.

ALGORITHM 4. CHECK CONFLICT WHEN ADDING AN EXISTING USER SUBJECT TO A USER GROUP

INPUT:
s: user or user group
g: user group where *s* will be added to
OUTPUT:
True: if the system still is consistent means there is no conflict happens
False: otherwise
METHOD *checkMoveMember(s, g)*
SP = *empty*
GP = *empty*
for each $p \in P$ and g $\in_k s(p)$
GP = GP $\cup p$
endfor
for each $p \in P$ and $s \in_k s(p)$
SP = SP $\cup p$
endfor
for each $p \in SP$
for each $p' \in GP$
if $pt(p') \neq pt(p)$ *and* $\prod v(p) \cap \prod v(p') \neq \varnothing$
return *False*
endif
endfor
endfor
return *True*
END *checkMoveMember*

In Fig. 13, if we add the user group *G7* to *G3*, two possible conflict authorization sets are *GP = {p1}* and *SP = {p2, p5}*. Since neither *p1* conflict with *p2* nor *p5*, *G7* will be added to *G3* successfully.

3) Check Conflict when Adding an Existing Video Element to a Group

Assuming that we are adding an existing video element *v* to a video group *vg*. The fact that two authorizations *p1* and *p2* can only conflict if they affect at least one common video, means we only verify conflict between the authorizations affecting *v* and all its child nodes in the video tree. Below the algorithm is presented in detail.

ALGORITHM 5. CHECK CONFLICT WHEN ADDING AN EXISTING VIDEO ELEMENT TO A GROUP

INPUT:
v: a video element
vg: video group where *v* will be added to
OUTPUT:
True: if the system still is consistent means there is no conflict happens
False: otherwise
METHOD *checkAssignVideo(v, vg)*
PP = *empty*
for each $v_i \in \prod v$
for each $p \in P$ and $v_i \in_k v(p)$
PP = PP $\cup p$
endfor
endfor
for each $p_i \in PP$
for each $(p_j \in PP)$ and $(pt(p_i) \neq pt(p_j))$ and $(\prod v(p_i) \cap \prod v(p_j) \neq \varnothing)$.
if $p_i <> p_j$
return *False*
endif
endfor
endfor

return *True*

END *checkAssignVideo*

In Fig. 13, if we add the video group *V2* to *V0*, the possible conflict authorization set is *PP = {p1, p2, p4, p5}* and *SP = {p2, p5}*. Since there is no conflict that occurs between any pair of authorizations of *PP* list, *V2* is added to *V0* successfully.

4) Check Conflict when Deleting an Authorization

When an authorization *p(s, v, pt, g, at)* is deleted, *s* and its children will be affected again by the authorizations *p'* which was overridden by *p*. Consequently, we must check the conflict between a set containing the authorizations affecting s, named *SP*, and another set *CP* containing the authorizations affecting *s* and its children.

ALGORITHM 6. CHECK CONFLICT WHEN DELETING AN AUTHORIZATION

INPUT: Authorization *p: (s, v, pt, g, soft)*
OUTPUT:
True: if the system still is consistent means there is no conflict happens
False: otherwise
METHOD *checkDeletePermission(p)*
SP = *empty*
CP = *empty*
for each $p' \in P$ and $s \in_k s(p')$
$SP = SP \cup p'$
endfor
for each $s' \in \prod(s)$
for each $p' \in P$ and $s' \in_k s(p')$
$CP = CP \cup p'$
endfor
endfor
for each $p_1 \in SP$
for each $p_2 \in CP$
if $pt(p_1) \neq pt(p_2)$ and $\prod v(p_1) \cap \prod v(p_2) \neq \varnothing$
return *False*
endif
endfor
endfor
return *True*

END *checkDeletePermission*

V. SYSTEM PROTOTYPE AND EVALUATION

In order to establish the practical importance of our extended video database model and novel access control mechanism, we implemented a system prototype and carried out empirical evaluations with real-world datasets. The prototype and experimental results are presented below.

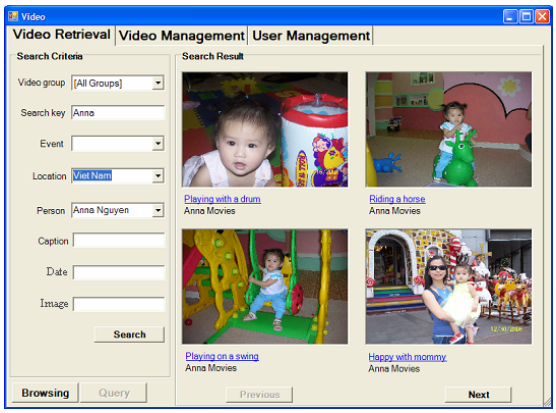
A. Choosing the Database Management System

For supporting digital video, the chosen Database Management System (DBMS) has to provide a multimedia data types such as image and video. In our framework, the video data type will be used to store the *StoredVideo* entities (cf. Fig. 6). Neither the BLOB (Binary Large Object) nor the file solutions are satisfactory because they could not provide a mechanism to identify, retrieve and use a small piece of a stored video segment. The file-based solution brings along the additional solution of managing data that is not fully under control of the DBMS. It will usually be more difficult to maintain the consistency of the system and in some cases impossible to provide necessary access restriction.

Due to these reasons, after considering popular commercial DBMSs, we decided to choose the Oracle interMedia to implement our video database by using its new multimedia data types such as ORDImage and ORDVideo. The first one, ORDImage data type, supports storing and image matching that is ideal for content-based retrieval. While the second one, ORDVideo data type, allows us to retrieve a part of the whole video stream and also to define the video stream's quality via the bit-rate parameter.

B. The Access Control Model

Implementing the *browsing mode* is quite simple because we only need to implement algorithm 1, *authorizeVideo*. In contrast, in addition to authorization problem, the *query mode* require us more efforts to refine the *solve query* algorithm. Fig. 14a and 14b are the screenshots of our system with the query and browsing modes.



(a) Querying mode.

(b) Browsing mode.
Fig. 14. Video retrieval pages.

The videos shown in Fig. 14a are the ones that match the criteria entered by a user and sorted by the distances between their contents and the input query which contains keyword,

video group, produced date, event, location, object, person, caption and image. Since the expressions for matching video group, location, produced date, person, object and texts (caption and keyword) had been presented in section B, hence, in this section, we will suggest a formula for the last query element, the image.

$$match(i,v) = 1 - \frac{\max(\text{ORDSYS .ORDImageS ingnature.}\\ \text{evaluateSc ore(i, I}_j\text{, weight))}}{100} \quad (11)$$

where *ORDSYS.ORDImageSingnature.evaluateScore* is a built-in function of Oracle interMedia. It returns the distance between two images, 0 if they are identity and 100 if they are totally different. In this formula, I_j stands for the j^{th} key frame of the video *v* and *weight* has a value of *"color=1, texture=0, shape=1, location=0"*, meaning we only focus on the color and the shape while searching. In this case, we compare the input image and every key frame of the video *v* to find out the maximum similar frame. There are a number of previous works that deal with estimating the similarity between two data objects. Interested readers are directed to [8], [13], [7].

C. The Permission Management Model

In our system, we allow a user to specify a permission at any user level (user or user group) against multiple video levels (video group, video, scene, shot, segment and object). In addition, we implemented a strict permission management system, meaning there is no conflict accepted. It always checks the conflict occurrence when an actor adds/edits permissions, a user/user group, or a video/video group. When a conflict happens, a message is shown that indicates exactly which user and video generated the conflict. Fig. 15 shows a screenshot of the permission management page.

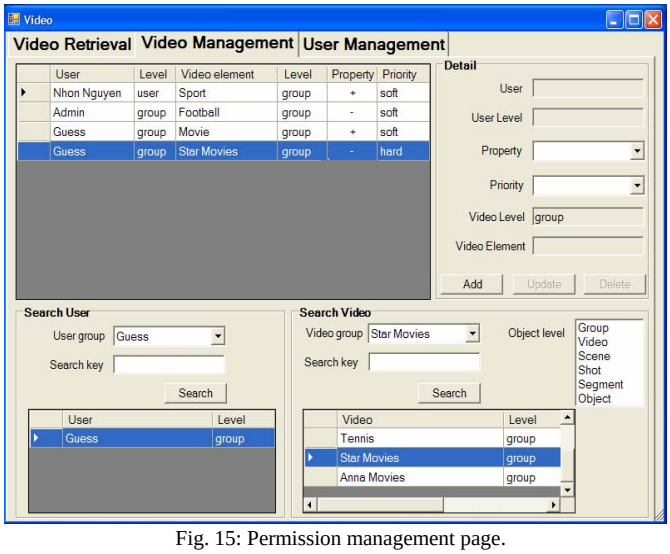


Fig. 15: Permission management page.

D. Preliminary Experimental Results

The data set has been used to validate the accuracy and performance of the system includes 142 video clips extracted from movies, news and sport clips that fill up 2.8GBs of memory. The Movies are divided into three groups named Action, Children, and Music movies while Sport category contains Football, Tennis and Others groups. Below we present the video list in detail.

TABLE I.
EXPERIMENTAL DATASETS

Video group	Subgroup	Number of Video
Movies	Action movies	35
	Music movies	17
	Children movies	20
News		25
Sport	Football clips	25
	Tennis clips	10
	Other clips	10

There are three user groups access to this video database including: Adult, Children and Disabled groups. To fully control the access of the above groups over the scenes, 11 Action movies have been separated into multiple shots (about 5 shots for each one) and 19 shots are duplicated in order to hide some inappropriate objects. Totally, to efficiently control the access, there are 100 MB of memory added to store the extra shots.

We implemented the prototype using Visual Studio 2005 with Visual Basic/.NET. All the tests were tackled on a laptop with an Intel Pentium M processor 1.73 GHz running Windows XP/SP4, 512 Mbytes of shared memory and some Gigabytes of hard disk capacity. The disk page size was 8Kb for the datasets. With respect to the system performance, we tested and collected the intervals to query the videos, to add new permissions and to retrieve a video. Table 2 shows experimental results for these operations over our datasets.

TABLE II.
EXPERIMENTAL RESULTS

Action	Condition	Time	Description
Add a new permission	No permission in our system	5 ms	
	There are 10 existing permissions	40 ms	
	There are 40 existing permissions	120s	
Query video database	Query using text criteria (title, actor's name, etc.)	43 ms	With 12 videos returned (averagely)
	Query using image field	94 ms	With 3 rows returned (averagely)
Retrieve a video	There are 15 concurrent users are viewing videos	12 ms	

It is obvious from the above results that there are two items that have poor performance and need to be improved. Firstly, time to check conflict when adding a new permission is huge, especially when there are many existing permissions in the system. Secondly, querying using image also consumes too much time. To solve these problems, we need to study an

efficient way to check conflict between two permissions and to compare two images. The correctness of our system's access control mechanism is proved by the fact that the system is robust at controlling access to the database since every user can only query the authorized videos and no "false hits" occurred in the tests.

VI. Conclusion and Future Work

In this paper, our main contribution is twofold: (1) Proposing an extended storage model for video data to support semantic visual concepts clustering and flexible content-based retrieval, and (2) Introducing a novel and flexible access control mechanism to support both multi-policy and multi-level access control in the newly proposed video databases. Our access control approach combines video indexing mechanisms with a hierarchical organization of video contents, so that different classes of users can access different video elements or even the same video element with different versions on the basis of their permissions. Besides, robust conflict checking algorithms have also been presented, ensuring conflict-free authorizations in the whole system. Preliminary experimental results with real-world datasets have confirmed the effectiveness of our proposed solutions.

In the future, we plan to investigate the efficiency of the proposed solutions with respect to the large video databases. Also, we will apply results of this research to real-world application domains such as surveillance and satellite video databases.

References

[1] N. Adam, V. Atluri, E. Bertino, E. Ferrari. A Content-based Authorization Model for Digital Libraries. IEEE TKDE, 14(2), 2002, 296-315.

[2] E. Bernito, J. Fan, E. Ferrari, M-S. Hacid, A.K. Elmagarmid, X. Zhu. A Hierarchical Access Control Model for Video Database Systems. ACM TOIS, 21(2), 2003, 157-186.

[3] E. Bernito, S. Jajodia, P. Samarati. Supporting Multiple Access Control Policies in Database Systems. IEEE Symp on Security & Privacy, 1996, pp. 94-107.

[4] A. Baraani-Dastjerdi, J. Pieprzyk, R. Safavi-Naini. A Multi-level View Model for Secure Object-oriented Databases. Data & Knowledge Engineering, 23(2), 1997, 97-117.

[5] J. Calic, E. Izuierdo. Efficient Key-Frame Extraction & Video Analysis. In: Proc. Int. Conf. on Information Technology: Coding & Computing, 2002.

[6] Chang S. F., Chen W., Zhong, D. A Fully Automatic Content-based Video Search Engine Supporting Spatiotemporal Queries. IEEE Trans. Circ. Syst. Video Tech, 1998, 1-4.

[7] Chen J., Taskiran C., Albiol A., Delp E., Bouman C. A Video Indexing and Browsing Environment. In: Proceedings of SPIE/IS&T Conf. Multimedia Storage and Archiving Systems IV, 1999, pp. 1-11.

[8] T. K. Dang. Semantic Based Similarity Searches in Database Systems (Multidimensional Access Methods, Similarity Search Algorithms). PhD thesis, FAW-Institute, Johannes Kepler University of Linz, Austria, 2003.

[9] S. Deb. Video Data Management and Information Retrieval. IRM Press, 2005.

[10] B. Furht, O. Marques. Handbook of Video Databases: Design and Applications. Taylor & Francis Group, 2005.

[11] R. Hjelsvold, R. Midtstraum. Modelling and Querying Video Data. VLDB 1994, pp. 686-694.

[12] K. Hoashi, M. Sugano, M. Naito, K. Matsumoto, F. Sugaya, and Y. Nakajima. Shot Boundary Determination on MPEG Compressed Domain and Story Segmentation Experiments for TRECVID 2004. KDDI R&D Laboratories, 2004, pp. 7-12.

[13] H.-P. Kriegel, P. Kunath, A. Pryakhin, M. Schubert. MUSE: Multi-Represented Similarity Estimation. In: Proc. 24th Int. Conf. on Data Engineering (ICDE'08), Mexico, 2008.

[14] H. Kosch. Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21. CRC Press, 2003.

[15] I.E.G. Richardson. H.264 and MPEG-4 Video Compression. John Wiley & Sons, 2003.

[16] B. L. Yeo, B. Liu. Rapid Scene Analysis on Compressed Video. IEEE Trans Circuits & Systems for Video Technology, 5(6), 1995, 533-544.

[17] J. Y. Zhang. Advances in Image and Video Segmentation. IRM Press, 2006.

[18] H. J. Zhang. Content-based Video Browsing and Retrieval. CRC Press, 1999.

[19] H. J. Zhang, A. Kankanhalli, S. Smoliar, S. Tan. Automatically Partitioning of Full-Motion Video. Multimedia Systems, 1(1), 1993, 10-28.