An Experimental Video Database Management System Based on Advanced Object-Oriented Techniques

Liusheng Huang

John Chung-Mong Lee

Qing Li

Wei Xiong

Department of Computer Science
Hong Kong University of Science and Technology, Hong Kong

ABSTRACT

Video data management is fast becoming one of the most important topics in multimedia databases. Most of the recent work on video databases has so far focused on video classification, feature extraction, spatial reasoning and image retrieval (video access); little work has been done on supporting advanced video editing and production activities, nor has there been much work done on providing facilities for efficient and versatile video data management. In this paper, we describe the development of an experimental video database system being implemented at HKUST, which employs extended object-oriented features and techniques. By incorporating conceptual object clustering concepts and techniques, it enables users to dynamically form, among other things, video programs (or segments) from existing objects based on semantic features/index terms. A prototype of this system has been constructed, using a persistent object storage manager (viz. EOS), on Sun4 workstations.

Keywords: video database, video editing and classification, Dynamic Object Clustering, extended Object-oriented

1 INTRODUCTION

Human beings can clearly remember text and images as well as whole procedures of events. When people recall something, they often say "I played a movie in my mind." It tells us that image sequences play a very important role in our memory. Similarly, it is very important for computers to "remember" image sequences, i.e., video. The information era has created an enormous number of video sources. Video cassette tapes and laser-discs are used for many purposes such as entertainment, instruction, scientific recording, art storage, etc. To efficiently manage video information has been one of the urgent tasks in computer systems. Video will only become a valuable and effective source in computer systems when we can effectively store, index, retrieve, edit and produce it just like text[1] is currently being used.

Many successful image databases have been developed. They can be divided into two categories, depending on the functionality of the system[8]. The first category consists of those databases which concentrate on retrieving an image according to one or more specific features. The second category of systems work mainly on retrieving requested information from a single image. In some systems, these two functionalities are combined.

More recently, a large amount of research activity has been focused on video databases. Video databases are more challenging to work on than image databases because they are more complex and have more features. Therefore, people in this area are working on different aspects such as video database model, video manipulation and retrieval system, automatic partitioning of video sequences[2][10], and video features analysis.

Relational database systems have been successfully used for text and image management because of their basic advantages such as clear concept and formal basis. However, an entity relationship model is not sufficient for managing video information. Video data involves complex objects which need composition hierarchy and object-specific operations. Therefore, object-oriented databases are considered as a possible alternative because of their power in behavioral modeling and behavior inheritance, complex objects support, type hierarchy, behavior encapsulation, and so on [4]. Despite these powerful features, OODBMS is still not sufficient for video data management especially with respect to editing and production. We believe that meaningful video sequences are identified and associated with their describing data incrementally and dynamically after the video objects are stored in a video database. Video data can be classified in various ways on the basis of its content. Any predefined scheme is hardly flexible enough to permit the later adaptive usage and reorganization of video sequences.

In this paper, we discuss a novel video database model, which is an object-oriented database model using clustering and role techniques. Based on this model, we introduce an experimental video database system being implemented at the Hong Kong University of Science and Technology. By incorporating conceptual object clustering concepts and techniques, our system enables users to dynamically form, among other things, video programs (or segments) from existing objects based on semantic features/index terms. A prototype of this system has been constructed, using a persistent object-storage manager (viz. EOS2.1), on Sun4 workstations. EOS is an object manager being developed at AT&T Bell Labs for providing key facilities for the fast development of high-performance database management systems[15]. EOS supports concurrent access to large objects by programs compiled with C or C++ compilers. Our video database is built on automatic video classification routines[9][10]. We shall describe some of the design and implementation issues of the whole system especially of the conceptual clustering mechanism (CCM). Finally, in our concluding remarks, we shall discuss further research directions.

2 BACKGROUND AND MAIN ISSUES

The work on video databases(VDB) may be classified into two categories: VDB modeling and VDB manipulation. In this section, we introduce the related works in these two aspects and the issues to be resolved.

2.1 VIDEO MODELING

To handle special features in video databases such as user-dependent descriptions of a scene, dynamic property of a video description and sharing among attribute values or methods, people have been searching long and hard for an effective video database model.

E. Oomoto and K. Tanaka[11] proposed a scheme-less object-oriented model. Their model allows users to (1) identify an arbitrary video frame sequence (a meaningful scene) as an independent object, (2) describe its contents in a dynamic and incremental way, (3) share describing data among video objects, and (4) edit author and abstract video objects. As their model is scheme-less, the traditional class hierarchy of the object-oriented

approach is not assumed as a database schema.

R. Weiss, A. Duda and D. K. Gifford[12] proposed an algebraic video data model. This model allows users to (1) model nested video structures such as shot, scene and sequence, (2) express temporal compositions of video segments, (3) define output characteristics of video segments, (4) associate content information with logical video segments, (5) provide multiple coexisting views and annotations of the same data, (6) provide associative access based on the content, structure and temporal information, and (7) specify multi-stream viewing. Their model does not support the exploration of interactive movies and home video editing.

W. I. Grosky et al [5] proposed a content-based hyper-media (CBH) model. It is nothing more than an object-oriented schema over non-media objects which has undergone a transformation. They classified an object-oriented schema into three domains: class hierarchy, nested object hierarchy, and complex object hierarchy. Then they used meta-data classes to build various relations between objects, such as "is-part-of " and "appearing-in".

R. Jain and A. Hampapur[17] proposed a video model (ViMod) based on studies of the applications of video and the nature of video retrieval requests. The features of their model include content dependence, temporal extent and labeling. A feature is said to be content independent if the feature is not directly available from the video data. Certain aspects of video can be specified based on viewing a single frame in temporal intervals, whereas other features like motion can be specified only based on a time interval. The changes that occur in video can be tracked over the extent of a time interval.

Q. Li and C.M. Lee[3] proposed a dynamic object conceptual clustering video database model (CCM) based on Q. Li and J. L. Smith's[14] Conceptual Model for Dynamic Clustering in Object Databases. This new model facilitates dynamic creation, deletion, and manipulation of ad hoc object collections (called "clusters"), with a goal to complementing existing object-class power for accommodating generic application dynamics[14]. In CCM, existing objects can be dynamically grouped with newly introduced roles to form active or passive clusters. So far, this model is the most powerful in supporting video objects especially for dynamic video editing or interactive movies.

2.2 VIDEO MANIPULATION

Manipulation or video sequence retrieval is also a challenging research area. Obviously, video should be retrieved by content instead of just time code or frame number. But the content of a video may be very rich and difficult to represent because video has both temporal and spatial dimensions. Moreover, the volume and unstructured format of digital video data make it difficult to manage, access and compose video segments into video documents. Manipulation and retrieval mechanisms, therefore, should be very different from and more complicated than those of text.

K. Hirata and T. Kato[6] built a query system which uses visual example, i.e., ART MUSEUM. A user has only to draw a rough sketch to retrieve the original image and all similar images in a database. The system evaluates the similarity between the rough sketch, i.e., visual example, and each part of the image data in the database automatically. This method is quite good from the viewpoint of users. However, there is a question about the system's effectiveness.

A. D. Bimbo, E. Vicario and D. Zingoni[7] used Spatio-Temporal Logic to support the retrieval by content of video sequences through visual interaction. Temporal Logic is a language for the qualitative representation of ordering properties in the execution sequences of temporal systems. In their database, video sequences are stored along with a description of their contents in Spatio-Temporal Logic. Retrieval is supported through a 3D iconic interface.

T. D. C. Little et al.[16] implemented a system that supports content-based retrieval of video footage. They

define a specific data schema composed of movie, scene and actor relations with a fixed set of attributes. The system requires manual feature extraction, and then fits these features into the data schema. Their data model and virtual video browser do not support queries related to the temporal ordering of scenes.

S. W. Smoliar and H. J. Zhang[1] used a framed-based knowledge base method to support retrieval. They used frames to represent both classes (the categories) and instances (the elements categorized). In addition to the common techniques in frame-based knowledge base, they translated knowledge of a slot's type into knowledge of how to search it for retrieval purposes.

2.3 MAIN ISSUES

Video DB has more issues to be resolved than text or image databases because of its special features as listed below.

First, complex objects need to be defined and supported in a video DB since video data can be of several different forms, e.g., frames, segments and programs. These objects are not independent, rather they share some inherent relationships in the form of "is-part-of" (or called composition) links.

Second, from different points of view, the same image sequence may be given different descriptions. For example, given a specific video sequence, one user may be interested in its attributes, like color/BW or gun fighting/car racing. Another user may be more interested in whether it is a indoor or outdoor scene. However, it is impossible and inefficient to include all possible attributes in video DB when it is built.

Thirdly, the descriptions may be added or deleted dynamically. Suppose, for example, we have defined a sequence with several attributes like sound/no-sound, sports type, main roles, etc. But later on, we may no longer be concerned with certain attributes, say, sound/no-sound, and need other features for describing the sequence. Unfortunately, the conventional OODBMSs are not good at generating new classes dynamically.

Last but not least, sharing of attributes (and values) or methods is often needed among video data objects, since meaningful scenes, in particular, may be overlapping or included in other meaningful scenes. For instance, we may describe sequence A as being about Clinton's activities. We may also define sequence B as being about Clinton's hobbies. Sequence B is actually a subset of A. Obviously, B will have some attributes that A has, say, they are both color and both outdoor scenes. On one hand, sharing among values and methods is very common. On the other hand, it is tedious to let users repeatedly input the same information for different objects. However, conventional OODBMSs do not support the inheritance of attributes and their values or methods at object instance level.

From the above observations, it is clear that meaningful video sequences are often identified and associated with their describing data incrementally and dynamically (after the video objects are stored in a video database). Therefore, it is important to accommodate video objects to have a flexible data structure which can be changed dynamically. In next section, we will introduce such an approach of developing a flexible video database system based on a novel extension to conventional OODB models.

3 ADVANCED OBJECT-ORIENTED FEATURES AND TECHNIQUES

In a conventional OODB model, the fundamental concepts are object and class. An object represents an encapsulation of attributes and methods; it has a unique object identifier (Oid) in the OODB. A class is described

as a means of gathering all objects which share the same set of attributes and methods. There are two kinds of inter-class relationships captured by a conventional OODB model, one is the subclass ("is-a") relationship and the other is the composition ("is-part-of") relationship. The former facilitates the notion of *inheritance* (i.e., a subclass may inherit the attributes and methods from its superclass, in addition to its local additional ones), and the latter supports the notions of *existence dependency* and *component shareability* between a composite object and its component objects [3][13]. Both types of inter-class relationships form class hierarchies which are useful for modeling video object relationships and data structure. For example, the composition hierarchy can accommodate the description of the structure of video data. In particular, a video program can be viewed as a composite object consisting of a sequence of frames which are component objects of the video program.

The main problems with the conventional OODB modeling approach are that a class has to be predefined statically, and objects of a class must be homogeneous in nature. Hence, such a kind of models does not provide adequate support for those applications involving objects and inter-object relationships which are by nature ad hoc, irregular, tentative and evolving (as exemplified by video data objects). Extensions to the conventional OODB models are therefore needed in order to accommodate such advanced applications effectively.

3.1 DYNAMIC OBJECT CLUSTERING TECHNIQUES

In [3,14], we have defined a basic conceptual clustering mechanism (CCM) which facilitates dynamic creation, deletion, and manipulation of ad hoc object collections (called "clusters") such that this mechanism can effectively accommodate more application dynamics. In CCM, a cluster consists of attributes, methods and a dynamic grouping of existing objects in the database, within which each object is assigned to one or more roles. More precisely, a cluster C_t is a dynamic object which has a tripartite form:

$$C_t =< A, M, X > \tag{1}$$

where A is a set of cluster attributes, M is a set of cluster methods, and X is a set of the role-player associates:

$$X = \{< R_i : S_i > | 1 \leq i \leq n\} \tag{2}$$

where R_i is a role, and S_i is a set of the objects that play that role within the cluster. Hence, the objects in S_i are called the "players" of the role R_i, and they are also the "constituents" of the cluster C_t. A role R_i can be described as follows:

$$R_i =< A_p , M_p > \tag{3}$$

where A_p and M_p are a set of attributes and methods defined by the role R_i; these attributes and methods are applicable to the role players. A role R_i is active if the set of its methods (M_p) is not empty, otherwise it is a passive role. The players of a role may be "homogeneous"/"heterogeneous".

Clusters do not create or delete objects; they only include-in or exclude-out objects of databases. Therefore, they allow the user to form views over existing databases by tailoring at the cluster level without affecting the database itself. This is very useful and important for video data processing, such as video indexing and video production.

Clusters can establish a super-cluster/sub-cluster hierarchy similar to a composite hierarchy of classes. But a cluster may dynamically define its constituent objects and these objects can be of different types (heterogeneous).

Similar to classes that form an "is-a" hierarchy, roles can establish a super-role/sub-role hierarchy. A sub-role may inherit the attributes and methods (if any) of its direct and indirect super-roles. Furthermore, a sub-role can also define new properties and/or overwrite some of the inherited ones. Any objects playing the sub-role should be viewed as players of the super-roles. Examples of clusters and roles in video databases are given in next subsection.

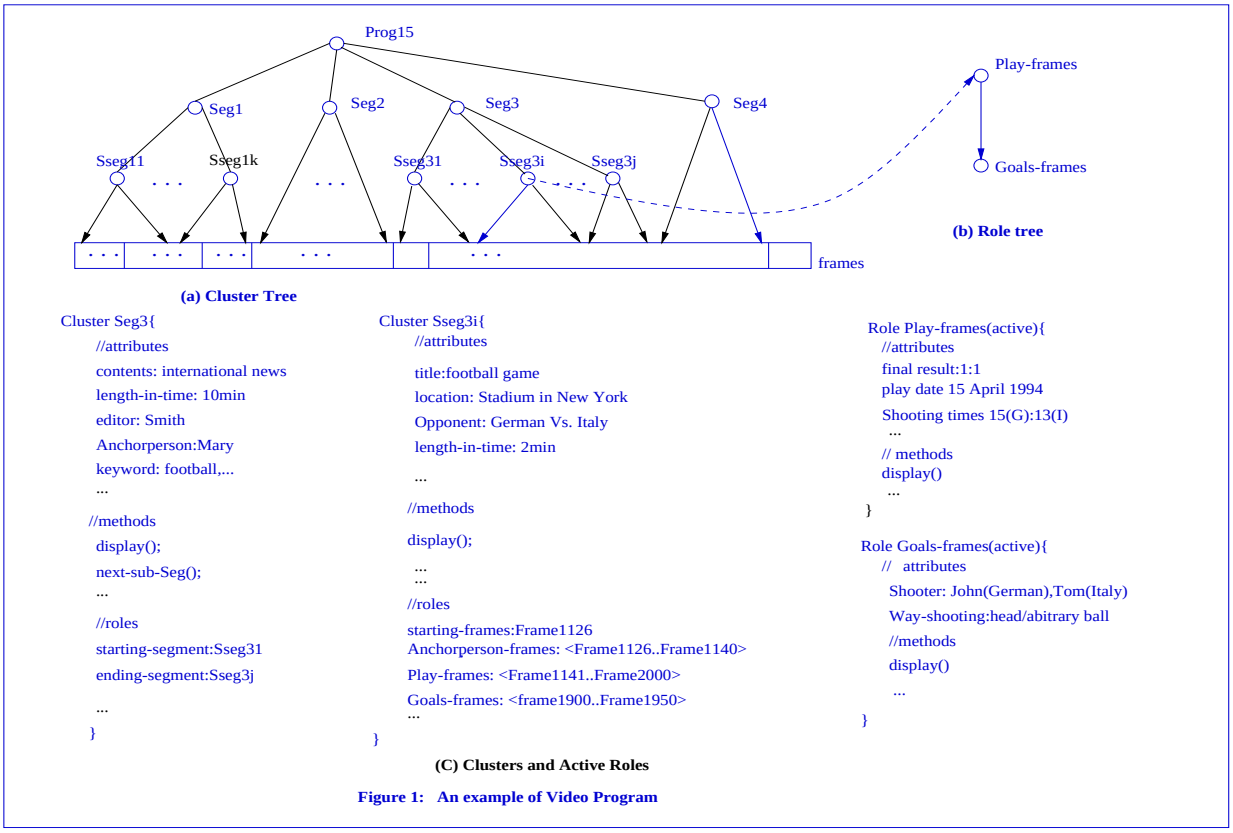

Figure 1: An example of Video Program

3.2 VIDEO DATA PROCESSING BASED ON CONCEPTUAL CLUSTERING

In this subsection we illustrate how CCM is used to support effectively video data processing in a video database context.

Firstly, we consider video indexing. As mentioned before, video classification enables video frames to be decomposed and grouped into semantically meaningful segments. We will show that clusters are an efficient means for describing video structure and semantic indexes derived from the video classification process. Suppose $Prog15$ is a news program comprised of a sequence of news items, some commercials and a weather forecast. Through classification, $Prog15$ is decomposed into segments: $Seg1, Seg2, Seg3$, and $Seg4$, which represent local news, commercials, international news and the weather forecast respectively. Furthermore, $Seg1$ (local news) and $Seg3$ (international news) are broken into some sub-segments, say, $Sseg11$, $Sseg12$,...,$Sseg1k$; $Sseg31$,...,$Sseg3i$,..., $Sseg3j$, each of which is a news item consisting of a sequence of frames. The structure of the news program $Prog15$ is constructed to a cluster tree as shown Figure 1(a), each node of which is a cluster. For nodes $Seg3$ and $Sseg3i$ (suppose it is a news item about a football game), their clusters' definitions are illustrated in Figure 1(b). The cluster $Sseg3i$ is the sub-cluster of $Seg3$. Obviously, most of the attributes and/or methods of $Seg3$ are not inherited by $Sseg3i$. If a sub-cluster inherits some properties from its super-cluster, we must explicitly specify which properties are inherited. For example, we may specify that the attribute "Anchor-person" in $Seg3$

is inherited by its sub-clusters.

```
Cluster Seg5{
        //attributes
        opponent:     German Vs. Italy
        location:     Stadium in New York
        final result:    1:1
        play date:     15 April 1994
        shooter:     John(German),Tom(Italy)
           ...
}
```

Figure 2: An example of Video Production

We define four roles in cluster $Sseg3i$, two of which are active roles as shown in Figure 1(c). These two active roles express play and goal-scoring scenes respectively. The role $Goal\text{-}frames$ is sub-role of the role $Play\text{-}frames$ as shown in Figure 1(b). In fact, super-roles and sub-roles also form a tree. Obviously, the attributes of the super-role $Play\text{-}frames$ may be inherited by its sub-role $Goal\text{-}frames$. The method "display" is defined in cluster $Sseg3i$, role $Play\text{-}frames$ and $Goal\text{-}frames$ respectively. Although the names of the methods are same, the "display" of $Sseg3$ may display all frames in this cluster (i.e., $Frame1126..Frame2000$); whereas the "display" of role $Play\text{-}frames$ only displays the frames of its player (i.e., $Frame1141..Frame2000$); similarly, "display" of role "goals-frames" only displays the frames of its player ($Frame1200..Frame1240$ and $Frame1900..Frame1950$). New attributes and methods may be added dynamically to these two roles in latter use, e.g., we may add a method "slow motion" into role $Goal\text{-}frames$ in order to watching goal-scoring scenes clearly.

Secondly, let us consider video production. Clustering techniques again prove to be very effective means for video data manipulation. Video production is the creation of a new video program out of a collection of video segments which are pre-existing and/or newly produced.

Assume we would like to produce a goal-collecting program out of various pre-existing news programs. We may search news programs from video databases and inspect each news program to see whether or not it contains video sequences of a football game. When a cluster represented football game is retrieved, we extract the frames of goals scored out of this cluster to form a new cluster. A collection of such new clusters forms the program goal-collecting as desired. For example, Figure 2 shows a cluster (suppose it is $Seg5$ in the new program) extracted from cluster $Sseg3i$ in Figure 1(c). Users may add a variety of attributes and methods into the new cluster and their roles (if active) as desired in the latter processing phases (e.g., editing).

Although we only discuss video production that make use of pre-existing video segments and frames, the same principle should also be applicable to producing video programs which involve newly taken segments. The only difference is that these new video segments need to be classified into clusters by a classifier.

4 THE DESIGN AND IMPLEMENTATION OF CCM

In this section, we first introduce the architecture of our VDBMS prototype system, and then give a detailed description of the design and implementation of the CCM.

4.1 ARCHITECTURE AND FACILITIES OF OUR PROTOTYPE SYSTEM

The prototype of VDBMS is built on the EOS2.1, which is an object storage manager being developed at AT&T Bell Labs for high-performance database management systems[15]. EOS supports access to large objects by programs compiled with any C or C++ compiler such as the ones distributed by AT&T, SUN, GNU, and CenterLine.

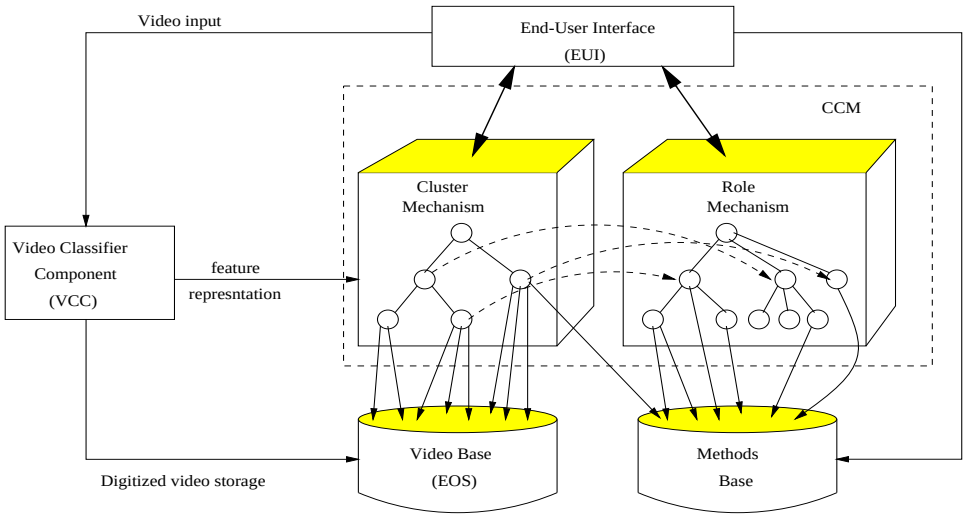

Figure 3: The Architecture of Prototype

Figure 4 illustrates the architecture of our prototype system, in which there exist two main components on top of EOS. One is the Video Classifier Component (VCC), and the other is the Conceptual Clustering Mechanism (CCM). Both are connected to an upper front user-interface. EOS is used to store video materials derived from the classifier. Each user may model his own cluster tree and corresponding role tree according to his/her personal needs. Different cluster trees may share video materials in the EOS database.

Besides the Video Base, another base is the Method Base in which methods may be pre-existing video operation commands/functions provided by the prototype system, as well as those appended by the users. These methods can be invoked by roles/clusters, and any one method can be shared by several different roles/clusters. Note that the objects operated by methods of a role are this role's players who are corresponding clusters component objects. Such objects, in general, are video materials in a video database. Therefore, the role can be viewed as the bridging mechanism between a video base and a method base.

Again, using Figure 1's example, frames are stored in Video Base(EOS), corresponding CCM consists of cluster tree $Prog15$ and role tree $Play\text{-}frames$. the method "display" is stored in Methods Base, it may be shared by cluster $Seg3$, $Sseg3i$, and role $Play\text{-}frames$, $Goal\text{-}frames$.

4.2 PROTOTYPING ISSUES

The prototype has been designed and implemented in an EOS and C++ environment on Sun4. In the remainder of this subsection, we will discuss design and implementation of cluster hierarchy, role hierarchy and dynamic execution of methods respectively.

At present, cluster- and role-hierarchies are only tree structures, namely, sub-clusters (sub-roles) and super-clusters (super-roles) are of a single-inheritance relationship. Generally, however, they are sufficient for video data processing because an existing object in a video database can play many different roles which are not in the same role hierarchy; similarly, an object can also participate in a variety of clusters which are not at all related.

4.2.1 CLUSTER TREE

A cluster tree is expressed as a binary tree with a father pointer in an internal system. For example, the cluster tree in Figure 4(a) is expressed as shown in Figure 4(b). The father pointer is used to fast search all super-clusters of a specific sub-cluster. The structure of each node in a cluster tree is illustrated in Figure 4(c). Its type definition may be viewed as a meta-cluster. All application clusters of any user defined are instances of this meta-cluster. All cluster nodes in a cluster tree may be dynamically created, removed and modified by the user.

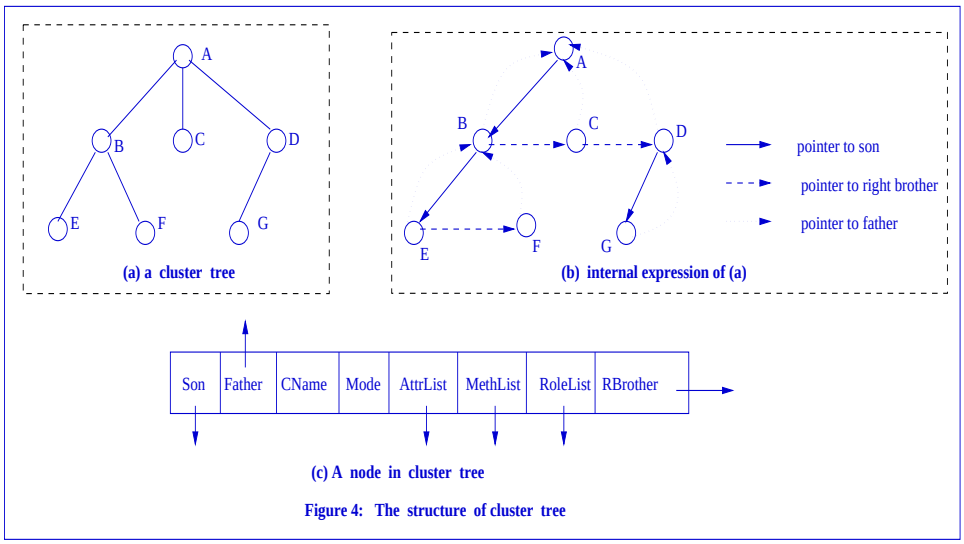

Figure 4: The structure of cluster tree

Since we cannot forecast how many attributes, methods and roles a user will define in a cluster, attribute-, method- and role-lists should be defined as dynamically linked lists. Their structures are illustrated in Figure 5.

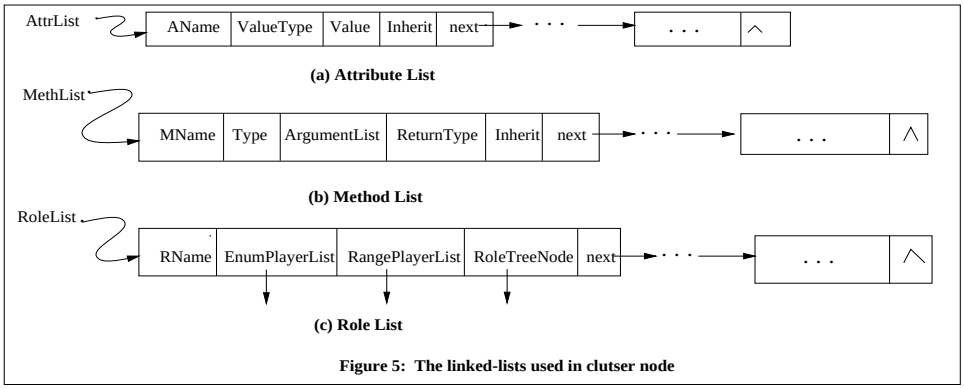

Figure 5: The linked-lists used in clutser node

Figure 6: A node in role tree

In the attribute list (Figure 5(a)), "ValueType" is char, string, int, float etc., and "Value" is a union type in order to save storage. "Inherit" indicates whether the attribute is inheritable by its sub-cluster or not. For a method-list (Figure 5(b)), the "Type" is used to distinguish whether the method is a executable file or a library function. The "ArgumentList" is a linked list which stores the type of these arguments. The "ReturnType" is a type of return value if the method is a library function. "Inherit" determines whether or not a method

may be inherited by its sub-clusters. Each role in role list (Figure5(c)) has a set of players which are also the constituent objects of the cluster. These players may be added and removed dynamically. Hence they should also be dynamically linked lists. The players of a role may be of an enumeration and/or range form, thus we employed both "EnumPlayerList" and "RangePlayerList" to express these two types of players. The attributes and methods (if active) in a role are defined by some node of a role tree. Therefore, each role node in a role list has a "RoleTreeNode" pointing to a corresponding node of role tree.

In the enumeration type of player-linked list, each node includes an object identifier (Oid) of the EOS. In range type of player-linked list, each node includes the upper- and lower-limit of the players which are logically continuous.

4.2.2 ROLE TREE

Similar to the cluster tree, a role tree is represented as a binary tree with a father pointer. The structure of each node in a role tree is shown in Figure 6. Its type definition may also be viewed as meta-role. All role nodes in role tree are also created, deleted and modified dynamically by the user.

In Figure 6, the attribute- and method-lists are the same as those of the cluster node mentioned above (viz. Figures 5(a),(b)). The "LinkType" indicates the link manner (strong/weak) between a role and its super-role. The deletion of a super-role implies the deletion of its sub-roles if the "LinkType" of those sub-roles are strong, otherwise the deletion of a super-role does not imply the deletion of its sub-roles. The "Type" of role node may be active or passive depending on whether or not the role includes methods. The "ReferenceCount" of a role node in a role tree denotes referenced times of the role. When the user adds a new role into a cluster node, the system will search for this role in the role tree and if it finds it then, the "ReferenceCount" is an automated increment otherwise the system adds a new role node whose "ReferenceCount" is initialized by one into the role tree. Similarly, if the user deletes a role in a cluster node, the "ReferenceCount" of the corresponding role node in the role tree is decreased by one.

The role node in a role tree may be created as independent or dependent of the role of the cluster. In other words, we may first establish the role tree and then establish the cluster tree; or first create the cluster tree and, in the meanwhile, create the role tree. A role node in the role tree is deleted if its ReferenceCount value is zero.

4.2.3 THE OPERATIONS ON THE CLUSTER- AND ROLE-TREE

Besides the dynamic creation, deletion and modification of a node in the cluster- and role-trees the current prototype provides many basic operations as follows: It displays the structure of the cluster- and role-trees hierarchically; it displays all methods and attributes of a node in the two trees and inheritable ones of this node's ancestor nodes. A cluster node may display all its roles and those of the super-clusters. In these two trees, the user may find specified role- or cluster-nodes and identify their attributes, methods, member objects. These basic operations (methods) facilitate the creation, deletion and modification of node in cluster- and role-tree, and addition, remove and modification of attributes, methods and member objects. For example, when the user adds attributes of role- and cluster-nodes, the system will invoke the method to display all the attributes of this node and inheritable attributes of its super-nodes. Thus the user knows which attributes have been defined or may be inherited to avoid repeating the definition (Of course, the system can prompt the user if the definition is repeated).

4.2.4 THE EXECUTION OF METHODS

As the prototype allows the user to include a set of methods into cluster and role dynamically, it greatly enhances dynamic characterization of clusters and roles. When including methods into a role or cluster, user is required to provide the method name, the argument type, along with other details. While the user invokes a method from some role or cluster, the prototype will automatically display all methods (including inheritable methods of ancestor nodes) of this role or cluster to allow the user to choose the ones he/she desires.

The prototype supports two kinds of methods: one involves executable files, the other involves library functions. For the former, the prototype will invoke the method selected by the user through the UNIX system. For the latter, the process is more complex because the name of function selected by the user is stored in a string as the value of this string, i.e., it is not an identifier denoted function-name and so the program can not invoke the function through this value. Fortunately, UNIX's C or C++ provides an interface of dynamic link programming. A Method Base (Figure 3) must be created to a shared object library by a compiler system. Thus our prototype can obtain the address of the function by string symbol and then invoke the function by its address. There exist similar problem in passing arguments of function, however, the C++ compiler only provides a means of obtaining the address of global variables by string symbol such that we can not find the address of a real-argument by string value if the real-argument is a local variable. It is sufficient in the context of VDBMS because the executives of the methods are all video data such as frame, segment, etc., so that prototype system may use Oids of EOS as arguments of invoking methods.

C++ allows the same name to denote different functions (i.e., overloading) such that the name of each function has a suffix in the internal symbol table generated by the C++ compiler. Hence our prototype must map the method name provided by the user into its internal expression.

4.2.5 STORAGE AND RESTORAGE OF CLUSTER- AND ROLE-TREES

In the current prototype, cluster- and role-tree are not stored in EOS in order to maintain the efficiency of system. The reason is that the structures of the two trees are very complex such that the EOS database has difficulty in operating efficiently.

In order to save memory, the prototype uses generalized lists as a storage structure of the tree. Namely all information about the tree is converted into a character string by a set of recursive algorithms. The string is then stored into an external file. Restoring cluster- and role-trees are the creation of the tree and all its related linked lists through to read the string from the corresponding file.

4.3 CURRENT STATUS

At present, the first component (VCC) of the whole system is close to being completed, while the second component (CCM) has already been completed. The VCC provides the following functions:

1. Detects camera breaks in an image sequence (i.e., change of scene),

2. Builds a table of contents based on key frames for the entire sequence of images,

3. Allows end-users to index any segment of the image sequence using key frames,

4. Allows end-users to change the order of image sequences,

5. Allows end-users to manually select an object/pattern in a frame and track or search for that object/pattern in image sequences based on color,size or shape,

6. Groups regions in a frame automatically based on color.

Our system is also able to classify the various contents of video segments including scenes with moving objects, camera zooming and panning, complex road scenes, fade-out, noisy scenes, video with degraded quality and very dark video images. These classified features are then fed into CCM. At present, VCC and CCM are being integrated.

5 CONCLUDING REMARKS

In this paper, we have described the development of an experimental video database system, based on extended object-oriented concepts and techniques. After an analysis of the shortcomings of traditional relation database technology and conventional object-oriented database (OODB) approaches, we advocated an advanced approach of developing a video database system based on an important extension of conventional OODB models. The extension is centered around the notion of a conceptual clustering mechanism (CCM). Such extended clustering facilities allow ad hoc, irregular, tentative, and/or evolving video object collections ("clusters") to be dynamically formed, stored, and manipulated, thus various features of video data derived from video classification and/or manipulations can be represented and accommodated in a flexible manner. Further, clusters can impact on and interact with the constituents (i.e., video data objects) through the method defined by the active roles within the clusters. The utility of CCM in various manipulations of video objects has been discussed. Finally, we described some of the design and implementation issues of CCM (which is being developed on top of a persistent object storage manager).

For the current prototype, there are several remaining issues which need to be addressed in subsequent research. For the CCM, a plan is ongoing to extend the current cluster-/role-tree to be lattices, so that a cluster/role can be derived from multiple super-clusters/super-roles. We will also enrich the current method base to provide richer types of functions/operations for video data processing and manipulations. One problem that must be addressed here is the support of passing arbitrary arguments, in order to provide flexible method definitions for the user. The completed CCM is then to be integrated with the video classification component (VCC), and a user-friendly graphics interface needs to be developed with real-time video display support (which may be based on, e.g., MPEG) on Sun. Finally, we plan to test and refine our system by applying it in several real-life environments, including TV news room studios, university educational technology centers, and possibly public libraries.

6 REFERENCES

[1] S. W. Smoliar and H. J. Zhang. "Content-Based Video Indexing and Retrieval" in IEEE Multimedia vol.1, no.2, pp.62-72,1994.

[2] Hongjiang Zhang and Stephen W. Smoliar. "Developing Power Tools for Video Indexing and Retrieval." SPIE, Vol.2185, pp.140-148, 1994.

[3] Qing Li and John C.M. Lee. "Dynamic Object Clustering with Video Database Manipulations." Proc. IFIP 2.6 Working Conf. on Visual Database System(VDB-3), Lausanne, Switzerland, March,1995.

[4] Won Kim. "Object-Oriented Databases: Definition and Research Directions." IEEE Trans. on Knowledge and Engineering, Vol. 2, No. 3, pp.327-341, 1990.

[5] W. I. Grosky, F. Fotouli, I. K. Sethi and B. Capatina, "Using Metadata for the Intelligent Browsing of Structured Media Objects," ACM SIGMOD Record, vol.23, no.4, 1994.

[6] K. Hirata and T. Kato, "Query by Visual Example (Content based Image Retrieval)," in Lecture Notes in

Computer Science, vol.580, pp.56-71,1992.
[7] A. D. Bimbo and E. Vicario and D. Zingoni, "Sequence Retrieval by Contents through Spatio Temporal Indexing," in Proceedings 1993 IEEE Symposium on Visual Languages (Cat. No.93TH0562-9), pp.88-92, 1993.
[8] T.Arndt. "A survey of recent research in image database management," in Proceedings of the 1990 IEEE Workshop on Visual Languages (Cat. No.90TH0330-1), pp.92-97, 1990.
[9] John C.M. Lee, and M.C. Ip. "A Robust Approach for Camera Break Detection in Color Video Sequence." Proc. IAPR Workshop on Machine Vision Application (MVA'94), Kawasaki, Japan, Dec., 1994.
[10] W. Xiong, C.M. Lee and M.C. Ip. "Net Comparison: A Fast and Effective Method for Classifying Image Sequence." IS&T/SPIE Symposium on Storage and Retrieval for Image and Video Databases, San Jose, USA, Feb., 1995.
[11] E. Oomoto and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System," IEEE Transactions on Knowledge and Data Engineering, vol.5, no.4, 1994.
[12] R. Weiss, A. Duda and D. K. Gifford, "Content-Based Access to Algebraic Video," in Proceedings of the International Conference on Multimedia Computing and Systems (Cat. No.94TH0631-2), pp.140-151,1994.
[13] Kim W., E. Bertino and J. F. Garza. "Composite Object Revisited." Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp.337-347, 1989.
[14] Q. Li and J.L. Smith. "A Conceptual Model for Dynamic Clustering in Object Databases." Proc. of 18th Int. Conf. On VLDB, pp.337-347, 1992.
[15] Eos "EOS 2.1 User's Manual", AT&T Bell Lab, Murray Hill, New Jersey 07974, 1994.
[16] T.D.C. Little et al, "A digital on-demand video service supporting content-based queries," in Proc. First ACM International Conference on Multimedia, pp.427-436, 1993.
[17] R. Jain and A. Hampapur, "Metadata in video databases," ACM SIGMOD Record, vol.23, no.4, 1994.