9

Dynamic Object Clustering for Video Database Manipulations

Q. LI and C.M. LEE

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

Abstract

Extensions to the object-oriented data model are described which address the dynamic nature of video database manipulations. These extensions support the dynamic grouping of objects to form a new object or cluster, and within each cluster a set of roles may be employed and/or introduced to define the behavior and interactions of the objects. In the context of a video database system, we examine the types of video data objects that require these extensions, and consider the utility of these extended features in supporting several generic types of video database manipulations, including video classification, video editing and video production.

Keywords

Conceptual object clustering, dynamic roles, object video databases, video data manipulations

1 INTRODUCTION AND MOTIVATION

"Video data" management is emerging as one of the most important topics in multimedia database systems. Within a *video database management system (VDMS),* flexible and advanced facilities are needed to provide support in video data processing and management. Most work on video data processing has concentrated on image analysis and recognition (for video classification), image structuring and indexing (for feature representation), and spatial reasoning and image retrieval (for video access) [(Chang 1992), (Hirata, 1992), (Kato 1992), (Nagasaka 1992)]. Little has been done to provide facilities for other types of video data processing such as *video editing* and *video production*, nor has there been much effort to develop efficient video data management facilities, which we regard as integral part of the VDMS's functionality.

Object modeling and object-oriented technology is having a major impact on the development of multi-media database systems [(Masunaga 1989),(Woelk 1987)]. While many

S. Spaccapietra et al. (eds.), *Visual Database Systems 3*

object-oriented concepts and techniques coincide with what are required by multi-media database systems, extensions are needed to adequately accommodate the new requirements of a VDMS, many of which involve domain dependent information that can be used to facilitate, e.g., more effective extraction of features from images stored in the database or used in queries (Jain 1993). In this paper, we describe an extended "object-oriented" approach of supporting effective video database manipulations based on the concept of *conceptual clustering*. In particular, we demonstrate that conventional object-oriented database facilities supplemented with dynamic conceptual clustering techniques allow a wide range of video data processing and management to be accommodated elegantly. In next section we briefly summarize the key concepts of object-oriented databases, followed in subsequent section by a review of the conceptual clustering model we have developed. Section 4 applies the various concepts and techniques which were introduced to video data processing and manipulation. Section 5 briefly describes our prototype environment and status, and finally Section 6 concludes the paper.

2 OBJECT-ORIENTED DATA MODELS

In this section, we first summarize a collection of concepts that are fundamental to object-oriented data modeling in general, and then introduce some generic types useful to describe video data objects in particular. We also illustrate some important elements in a type/class hierarchy relevant to the subject matter of the paper, and which in some cases require extensions to the current object-oriented data model facilities.

2.1 Some Basic Concepts

The most fundamental concepts of object-orientation include *object* and *class*. In an object-oriented database (OODB), the term object represents an encapsulation of instance variables constituting a state, and methods for manipulating it. Classes are used to describe object types*, as defined by their *properties* (connoting instance variables and methods); they are also used to create, destroy, or store objects through such run-time facilities as object-factory and object-warehouse (Atkinson, 1989). Objects communicate through *messages*, and the handling of messages (the translation of method names to the actual program addresses) is done at run-time and is called "late binding". For each instance variable, the set of values it may have is confined by its class type (which is called the *domain class)*; both atomic (e.g., integer, string of characters) and abstract (i.e., object) domain classes are possible in an object. Objects are uniquely distinguished with their (system-generated) object identifiers (Oids), hence the existence of an object is independent of its values.

Inheritance is a powerful mechanism used for defining specialization ("is-a") relationships between classes in an OODB. In particular, a class X may be defined as a specialization of another class Y; class X (called the subclass of Y) inherits all the properties of Y (called the superclass of X), and the user may specify additional properties for the subclass. A class may have any number of subclasses, but a subclass may have only one direct

*Here, we follow the convention of unifying classes with types as found in most OO languages (such as C++).

superclass (in the case of *single inheritance*) or any number of direct superclasses (in the case of *multiple inheritance*).When only single inheritance is supported, the classes form a strict class hierarchy (a tree). If multiple inheritance is allowed, the classes form a rooted directed acyclic graph (DAG). For simplicity, we will also call the DAG a class hierarchy.

In a class hierarchy, subclasses are allowed to refine/restrict the domain of an inherited instance variable. The domain of an instance variable inherited from multiple superclasses is hence restricted to the intersection of the instance variable domains. Further, subclasses can also have different implementations of any method defined at the superclass level (which is called "overriding"), resulting in a single method name denoting different programs (i.e., *overloading*). In the case of multiple inheritance, it is possible to have naming conflict among the inherited properties (methods and/or instance variables). One way of resolving such property naming conflicts is by means of prefixing the name of the class in which the property is locally defined. In this paper, we shall follow this convention in dealing with naming conflict resulting from multiple inheritance.

Besides the class hierarchy, another type of hierarchy typical in an OODB is the composition hierarchy. This is fundamental in capturing the "is-part-of" relationships between a parent class and its component classes. A *composite object* O can be defined as an object with a set of *abstract* instance variables (whose domain classes are non-atomic), each of which refers to one or more component objects of O. Composite objects add to the integrity features of an OODB model through the notions of existence *dependency* and component *shareability.* Following the treatment of (Kim 1989), the following four cases are pertinent to composite objects and their component objects (also called "sub-objects"):

1. **dependent and shareable**: the existence of the sub-objects is dependent on the existence of a composite object (also called "parent object"). As such, a component object can not be created if its parent object does not already exist. But the sub-objects may be shared, i.e., they can be a component of more than one parent object at the same time.
2. **dependent and exclusive:** the existence of the sub-objects is dependent on the existence of a composite/parent object, and the component objects are not sharable, i.e., they are exclusive sub-objects *owned* by one parent object.
3. **independent and shareable**: the existence of the sub-objects is not dependent on the existence of a composite/parent object; as such, the deletion of the parent object does not imply the deletion of (all) the sub-objects. Further, the sub-objects may be shared by more than one parent object at the same time.
4. **independent and exclusive:** the existence of the sub-objects is not dependent on the existence of a composite/parent object, and the component objects are not shareable, i.e., they are exclusively referenced by only one object (i.e., the parent object).

2.2 Video Data Objects

Video data objects (VDOs) are the basic type of objects involved in video database manipulations. We now introduce two important generic object types which feature in a video composition hierarchy. The first is called a Video-Frame (V-Frame). It is the fundamental type of video data object that embraces the common imagery properties (even with colors) of video data objects; it is also the elementary unit of storage of video data objects (e.g., as BLOBs - Binary Large OBjects). The second generic object type is called a Video-Program (V-Prog). It is a composite object consisting of a finite number of V-Frame objects. A V-Prog object

typically defines bibliographic properties such as the title, subject, producer, and year. Aside from these two generic object types, it is convenient to think there is also an intermediate "virtual object type" called Video-Segment (V-Seg). In particular, a V-Seg object can be viewed as an object derived dynamically by grouping a sequence of V-Frame objects (the derivation of V-Seg objects using the clustering technique is discussed in Section 4). Typical properties of a V-Seg object may include "structural metadata" (e.g., the sequence of scenes of a segment represented by a sequence of frames) (Davenport, 1991), and even "content metadata" (e.g, sets of key frames depicting the major scenes in the video segment) (Rowe 1994). Figure 1 intuitively illustrates V-Prog object and its V-Frame sub-objects together with a number of V-Seg objects each of which groups several V-Frame objects. Note that the composition links from a V-Prog object to the V-Seg sub-objects are in general **dependent** and **exclusive**, whereas the links from a V-Seg object to its V-Frame sub-objects may also be **independent** and/or **shareable** (e.g., a V-Seg object may be edited/formed by including some V-Frame objects that are desirable-to-have, though not necessary-to-have).

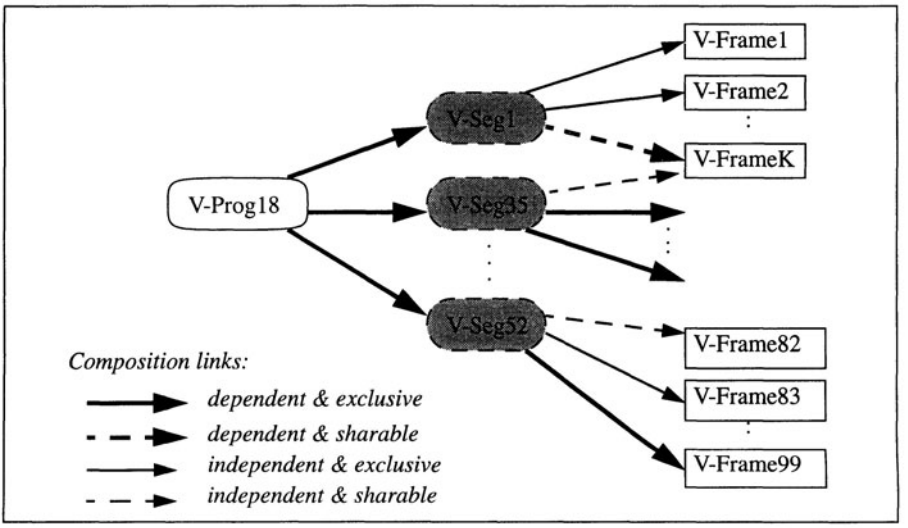

Figure 1 Example of a V-Prog Object and its Component Video Sub-objects

2.3 Exclusive/Inclusive Subclasses and Dynamic Objects

We articulate that in a video object database, an *is-a* hierarchy subclass may be exclusive or inclusive. By exclusive subclasses we mean that if there is more than one specialization stemming from a class at any level in the hierarchy, then an object at the subclass level may be of one and only one subclass. If subclasses are non-exclusive (i.e., inclusive), then an object at that level may be of more than one subclass. Figure 2 illustrates some examples of exclusive and inclusive subclasses in the *is-a* hierarchy of an object-oriented video database. Note that

the most general classes are system abstract data types, which are generalized data types from which there is no instantiation, only inheritance. At this level we add the abstract type of Dynamic Object because, in future applications, objects will have dynamic behavior as well as static behavior. With the Dynamic Object we associate the functionality to add and drop attributes and methods dynamically. Cluster and role are introduced in Section 3.

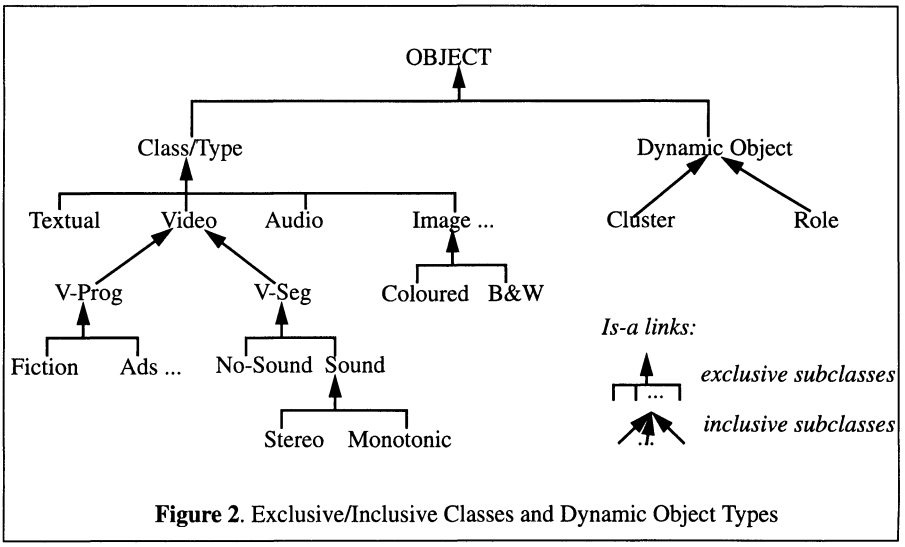

Figure 2. Exclusive/Inclusive Classes and Dynamic Object Types

The Video sub-type hierarchy inherits from the static object Class (or Type). On the other hand, an obvious specialization of the Dynamic Object type is the cluster. As already mentioned this is a type which is added on to an existing OODB model, which is a collection of existing database objects playing particular roles defined within the cluster. In next section, we introduce this "extended" facility in detail. The utility of this facility in video database manipulation is describe in Section 4.

3 DYNAMIC OBJECT CLUSTERING

The conceptual clustering model (CCM) facilitates dynamic creation, deletion, and manipulation of *ad hoc* object collections (called "clusters"), with a goal to complement the existing object class power for accommodating generic application dynamics (Li 1992). In CCM, existing objects can be dynamically grouped with newly introduced *roles* to form active or passive clusters. In this section, we review some of the basic aspects of CCM, including taxonomy and role mechanism, with examples drawn from the video database domain to show the relevance and potential utility of this new concept in supporting video database applications.

3.1 Cluster Taxonomy

From the semantic modeling point of view, a cluster acts like a "context" through which database objects take up new or relinquish existing *roles* dynamically. Roles typically imply extra properties (attributes and/or operations) for their member objects, known as "players" (which are also referred to as the constituents of the cluster). Depending on the manner of clustering and the potential interactions among the constituents, clusters are further distinguished into several kinds. In (Li 1992) a taxonomy of 12 kinds of clusters has been set up, which is based on the perspectives of derivation, uncertainty, and behavior (see Table 1).

Table 1 A Taxonomy of Dynamic Object Cluster Types

Derivation	*Adaptive*	*Infixed*	**Certainty**	*Obscure*		*Explicit*	
Deep	DAC	DIC	- - - →	ODAC*	ODIC*	EDAC*	EDIC*
Shallow	SAC	SIC	- - - →	OSAC	OSIC	ESAC	ESIC
Remarks: (1) Deep clusters are always loosely-coupled.			**behavior**	⋮	⋮	⋮	⋮
(2) Asterisked combinations constitutes the taxonomy.			*Loosely-coupled*	LOSAC*	LOSIC*	LESAC*	LESIC*
			Tightly-coupled	TOSAC*	TOSIC*	TESAC*	TESIC*

First, from the perspective of derivation, clusters can be formed by either "clustering-by-copy" or "clustering-by-reference". The former results in aggregate-like clusters (called **Deep Clusters**) in which *deep copies* of the source constituent objects are made, and subsequently become *owned* members of the clusters; the latter yields complex clusters (called **Shallow Clusters**) whose constituents are simply references/pointers (i.e., *shallow copies*) to existing source objects (of some classes), as opposed to being owned members of the clusters. Orthogonally, clusters can also be formed by applying some local changes or without any adaptations, which leads to the distinction of **adaptive** clusters and **infixed** clusters. For example, a new video segment (internally defined as a cluster) may be formed by composing several of the existing video segments and/or frames with some "special-effect" exercised on some of the segments and/or frames. We note that such special-effect adaptations are of *local scope* only, i.e., they do not carry global effect outside the cluster.

Complementary to the above perspective, a cluster can have constituent objects that are fully determined, are only partially identified, or are identified disjunctively. If a cluster's constituents are fully determined, then it is called an **explicit** cluster, else it is an **obscure** cluster. An obscure cluster may or may not become explicit at a later stage. This depends on the nature and degree of uncertainty involved.

The last perspective of cluster classification regards the behavioral interactions of the constituents. In particular, the constituent objects may (or may not), as a consequence of the clustering, exhibit new behaviors which may interact with their pre-existing behaviors (defined in their classes). If such behavioral interactions exist, the resultant cluster is called **tightly-coupled**, otherwise it is called **loosely-coupled**. For example, a previously silent video

(sub)segment may be upgraded to sound video after it has been incorporated into a new video. On the other hand, a black-and-white video frame may be desired to remain in black-and-white even after it has been added into a colored video segment with other color video frames.

Combining all three perspectives, we obtain 12 possible kinds of clusters based upon *applicable** combinations of the terms identified above (see Table 1). A set of associated cluster operators has been devised on a prototype, which supports the basic definition and manipulation of the clusters (Li 1992). Together, the cluster primitives and operators provide a rich set of facilities that are both semantically expressive and behaviorally powerful in modeling real-world application dynamics.

3.2 Role Facilities

As mentioned earlier, a cluster can be viewed as a context through which its constituent objects are able to play various *roles*. In a sense, therefore roles are like "threads" that link constituent objects together in forming the cluster. Examples of roles in video databases include any features relevant to the description of VDOs, which can be *descriptive* (e.g., theme-frame, break-points, etc.) as well as *active* (i.e., with operations defined, as exemplified by color, background, foreground, etc.). From the functional point of view, roles can be regarded as *virtual classes* which do not create/delete objects, but only include-in or exclude-out objects of the database. Like a class, a role has in its definition a set of attributes and methods (if active) which are applicable to any objects playing this role. But unlike a class, the objects that play a role can be heterogeneous (i.e., of different types, as opposed to homogeneous ones). Furthermore, roles can be created, deleted, or (structurally and/or behaviorally) modified dynamically.

Similar to the classes that form an *is-a* hierarchy, roles can establish a super-role/sub-role hierarchy, in which a sub-role may inherit the attributes and methods defined by the super-role. In addition, a sub-role can also define new properties and/or overwrite some of the inherited ones. For example, in a video documentary to be produced (represented as a cluster), the role called "music" defines the various music pieces used in the documentary, and its objects (i.e., the role players) are the actual audio files stored in the database. Suppose another role called "theme-music" is defined as a sub-role of the role "music", then all the properties defined by "music" become relevant and applicable to the role "theme-music", and any objects (music files) playing the sub-role (viz., theme-music) should therefore be counted as players of the super-role (viz., music) as well. Note that it is possible for the sub-role to define new properties (attributes and methods) in addition to the inherited ones, and also it may modify the definition of an inherited property from the super-role.

4 VIDEO MANIPULATIONS BASED ON CONCEPTUAL CLUSTERING

A frequent requirement in video database applications is to cluster/group existing video data objects to form dynamic collections, as exhibited by such manipulative activities as video classifications, feature extractions, and video editing and productions, the primary function of

*Note that some of the combinations such as **Deep** with **Loosely-coupled** are regarded as inapplicable, since it is trivially true in any case (cf. (Li 1992)).

the latter two is to tell some story (Davenport, 1991). In this section, we examine some of these activities (particularly video classification and video production) in detail, and illustrate how object-oriented modeling facilities, in general, and the conceptual clustering mechanism introduced above, in particular, can be applied to support general-purpose video manipulation operations in a video database context.

4.1 On Video Feature Representation

A fundamental task in video database manipulations is *video classification* [(Lee 1993), (Lee 1994), (Xiong 1995)]. In particular, an "intelligent" video classifier is a software component that can, ideally, classify not only the *structural* but also *semantic* contents (metadata) of a given video program in a (semi-)automated manner. We note that such a view has also been taken and supported in the Berkeley Distributed VOD project (Rowe 1994), which uses Postgres DBMS to store the index/metadata. While developing such a video classifier calls for a set of "intelligent" algorithms (see [(Lee 1993), (Lee 1994), (Xiong 1995)]), an appropriate data structure for the representation of the various features and/or semantic-indices created by the classification process must also be devised to account for the semantic contents of the classified program and its decomposed (semantically meaningful) segments. Clearly, the structure should also facilitate the search of the classified segments and frames of the video program.

We believe clusters as introduced in previous section are a good means for representing semantic features derived from the video classification process. In particular, the role part of a cluster can facilitate the representation of various semantic features, while role players refer to actual video frames. For example, given a video program (e.g., V-Prog17), a pre-processing video classification (based on a set of algorithms) may yield several segments (i.e., V-Seg objects), each consisting of a collection of video frames (i.e., V-Frame objects). Figure 3a shows three resultant V-Seg objects, plus an additional index (e.g., Prog17-Index), all represented as clusters. A sample "zoomed-in" V-Seg object (e.g., V-Seg3) with its cluster definition is shown in Figure 3b. Note that V-Seg3 is a TESAC (tightly-coupled, explicit, shallow and adaptive cluster), therefore it is possible to apply the operations defined in the cluster and/or roles (if active) to the component objects (i.e., V-Frames) directly.

While all three V-Seg objects in this particular example are at the same abstraction level, it is possible that finer-grained segments may be further decomposed and/or derived, with necessary inter-cluster operations provided [(Li 1993), (Li 1995)]. Such inter-cluster operators allow the inter-segment relationships (such as *subset*, *similarity*, *continuity, inverse,* and *dependency*) to be established, which may be utilized later in facilitating efficient access to video segments (and video frames) in the database. Further investigation on this issue is currently being carried out in our subsequent research.

4.2 On Video Editing and Production

Using clusters as primary means for describing video segments (V-Seg objects) and semantic indices is advantageous not only from the video classification (feature representation) point of view, but also in video editing and video producing applications. Below we examine how clusters can be utilized for these kinds of activities.

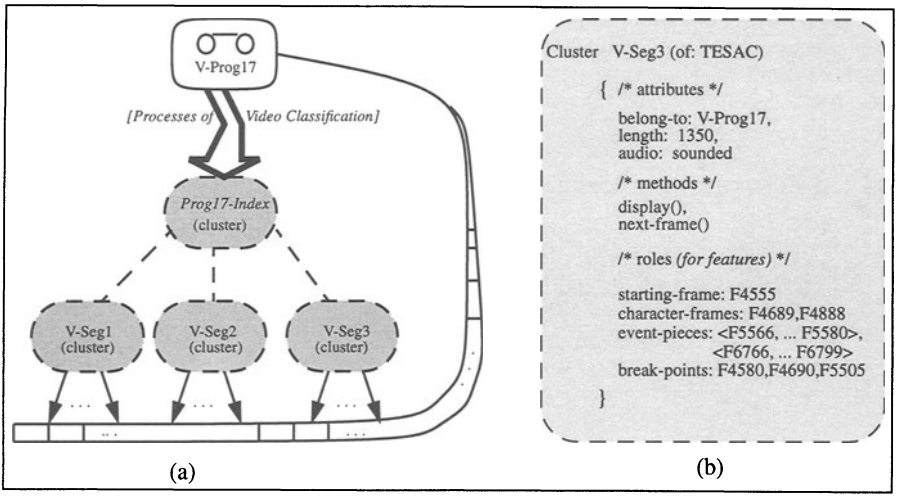

Figure 3 A Classified Video Program (a) and One of Its Segments (b)

4.2.1 Editing

With video segments being represented as clusters, an immediate advantage is that various editing operations that are defined in the clusters can be applied to the segments, and to the video frames of each of the segments. First, let us consider video editing at the video-frame level. As mentioned before, the video classification process enables video frames to be partitioned and grouped into semantically meaningful segments. Such segments can be implemented as tightly-coupled clusters which may define appropriate editing operations (through active roles) to be applied to the member frames. Suppose, for example, that in segment V-Seg3 described in Figure 3 there is a role called "monochrome", which records the frames that are non-colored (i.e., black-and-white) in the segment. Further, assume this role is an *active* one (i.e., it defines operations that are applicable to its role players, including coloring(), darkening(), lightening(), etc.). Thus, individual V-Frame objects that are players of this role may be edited to achieve the desired effect, using some (or all) of the operations defined in the role. Note that additional editing operations may also be dynamically introduced during the editing if necessary, since roles and clusters are dynamic constructs that support dynamic addition (and deletion) of attributes as well as methods (operations) (Li 1992).

At the video segment level, various editing can also be done if a higher-level cluster is created out of the classification process. This higher-level cluster would have all the resultant V-Seg objects to be its constituents. An example would be a global *video index* that describes a classified video program (e.g., *Prog17-Index* for V-Prog17 in Figure 3a). The global video index may record, among other features, such global properties as the *structure* (in terms of the component V-Seg objects), the *semantic* descriptors (e.g., title, key-words, primary-segments, action-segments, music-segments, etc.) of the video program. By defining some of the semantic descriptors as *active* roles (with appropriate operations defined or attached),

individual segments that belong to such roles may be edited as needed according to the operations. For example, a pre-processing (classification) of a video documentary (Doc2) may generate an index as shown in Figure 4a, which is internally defined as a TESIC (tightly-coupled, explicit, shallow and infixed cluster). An active role (viz., music-segments) of this index cluster is shown in Figure 4b. By applying some of the operations defined for the role (e.g., Analog-On(), Stereo-On(), etc.), any segment that plays this role (within the video documentary) may be edited and modified as desired.

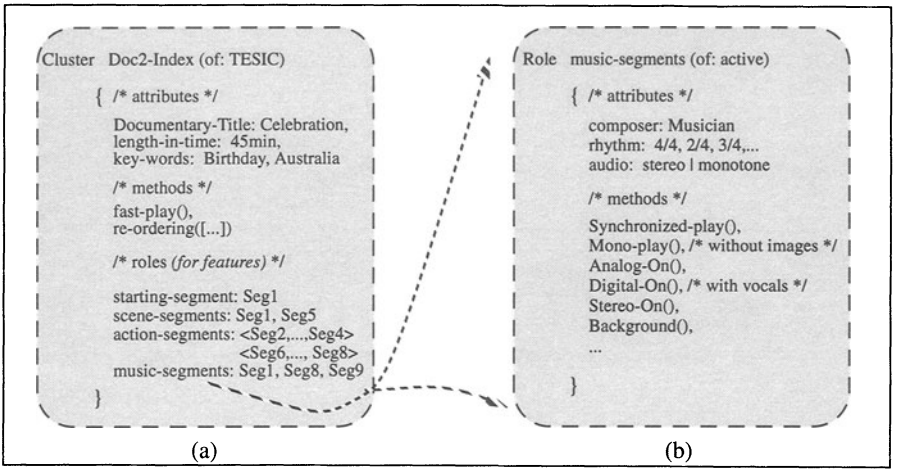

Figure 4 A Global Video Index (a) and One of Its Active Roles (b)

4.2.2 Production

To a large extent, video production can be viewed as an "inverse process" of video classification: the latter generates a set of semantic segments and indices out of existing video programs, whereas the former creates new video programs out of a collection of video segments which are pre-existing and/or newly produced. Here, our discussion focuses on those kinds of video production that make use of pre-existing video segments (and frames) in producing new video programs. Clustering techniques, again, can provide very effective means for accommodating activities in this domain.

Consider, for instance, a short video advertisement of a classic black-and-white film (e.g., "Rome Holiday") is to be produced. Suppose for the sake of entertaining, the producer would like to include tinged pictures extracted from selected segments. By defining the new advertisement (called V-Ads.RomeHoliday) as an EDAC (explicit, deep and adaptive cluster) as shown in Figure 5, it allows deep copies of selected segments to be taken out of the film (i.e., V-Prog objects) when the cluster (i.e., V-Ads.RomeHoliday) is formed. Further, local adaptation operations (e.g., *tinging()*, *fading-out()*, etc.) can be applied to the selected segments in producing the advertisement as desired. "Special effects" techniques such as

quick-motion, background, and foreground may also be used, by defining roles corresponding with role players (i.e., the video segments) and role operations (which may call up existing library functions or players' methods). Note that in the cluster V-Ads.RomeHoliday, there is a special role called "sequencing", which is an active role that sets (and can reset) the playing order of the selected segment copies.

When the new video program to be produced is a long one, making actual copies of the video segments may prove to be neither feasible nor desirable. Hence, shallow-clustering techniques are usually more appropriate in such cases. For example, suppose one needs to produce an hour-long video documentary on Ronald Regan which reuses many of the existing video segments from various documentaries and perhaps even from movies in which he has starred in. Instead of creating deep copies of all the segments that constitute the final documentary, a more cost-effective procedure is to simply use shallow copies of the segments by defining the final documentary as a LESIC (loosely-coupled, explicit, shallow and infixed cluster). This avoid any impact on the video segments (due to loose-coupling), while facilitating the sharing of segments from various programs.

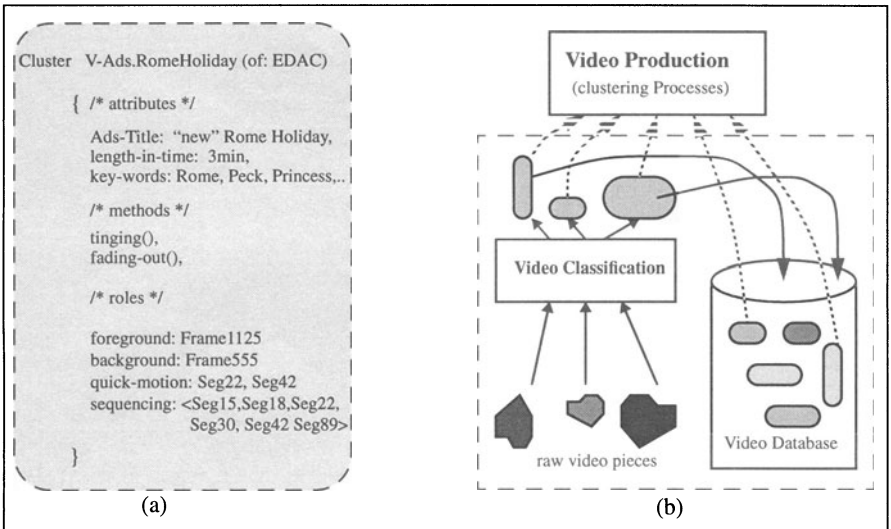

Figure 5 Video Production with Existing Segments (a) and New Ones (b)

Although the examples discussed so far have only considered video production which involves reusing only existing video segments, the same principles should also be applicable to producing video programs that involve newly taken segments, as long as they are represented and stored as clusters. The only difference is that these new video segments (raw video pieces) need to undergo pre-processing or video classification, which generates a corresponding set of classified (possibly edited) video segments (i.e., clusters). Figure 5b intuitively illustrates this situation. Note that processed segments may also be added into the database for later use.

5 SOME DESIGN AND IMPLEMENTATION ISSUES

As part of the research, an experimental prototype of VDMS is currently being developed at Hong Kong University of Science and Technology. The prototype system is being built on the platform of Macintosh Quadra series. Figure 6 illustrates the conceptual architecture of the prototype system, in which it is shown that there are two main development components on top of an object-oriented database. One is the Video Classifier Component (VCC), and the other is the Conceptual Clustering Mechanism (CCM), both are connected to the underlying database and an upper front user-interface. Note that the relationship between VCC and CCM is not always uni-directional, since an edited video segment and/or package may lead to an enhanced classification or trigger a re-classification process to be activated. On the other hand, a newly produced segment/package (through CCM) is automatically a classified video segment/package.

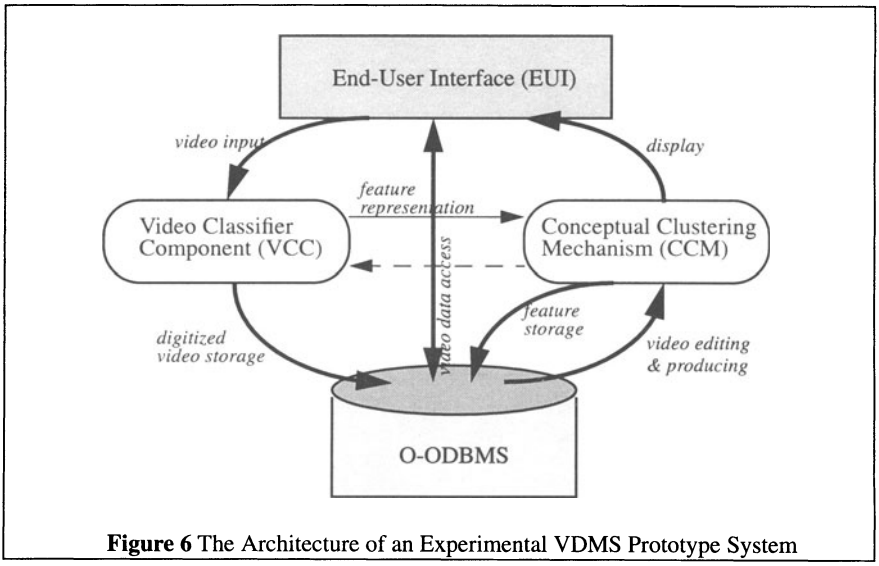

Figure 6 The Architecture of an Experimental VDMS Prototype System

5.1 Current Status

To date, the first component (i.e., VCC) of the whole system is close to be completed, and the second component (CCM) is being implemented. The system consists of a Macintosh Quadra 700, a Pioneer LD-V8000 video laserdisc player and a 24-bit color RasterOps MediaTime video board. The laserdisc player is connected to the Macintosh Quadra through a RasterOps MediaTime video board, which digitizes images from the laserdisc player and displays them at a rate of 30 frames per second (fps). For the first component, a number of sophisticated video classifiers has been devised [(Lee 1993), (Lee 1994), (Xiong 1995)] for:

(1) classifying video images into segments with similar contents, (2) assigning index terms to individual segments, and (3) combining the index terms to form a table of contents or an inverted index file.The aim here is to come up with an "intelligent" VCC that can efficiently organize, through indexing, a sequence of video images in order to enable end-users to access particular video segment and/or frame quickly. At present, our Macintosh-based system is able to read image sequences directly from a laser disc in either CLV or CAV format and 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 present system is able to classify various contents of video segments that include scenes with moving objects, camera zooming and panning, complex road scenes, fade-out (special effect), noisy scenes (e.g., raining, gun flare), video with degraded quality (fluctuating in intensity) and very dark video images. Figure 7 shows some sample indices obtained from a video sequence together with the computer interface of the control panel for the laser disc player. These classified features are then to be fed into the second component (CCM) for later storage into, and future reuse from, the underlying video database (as depicted in Figure 6).

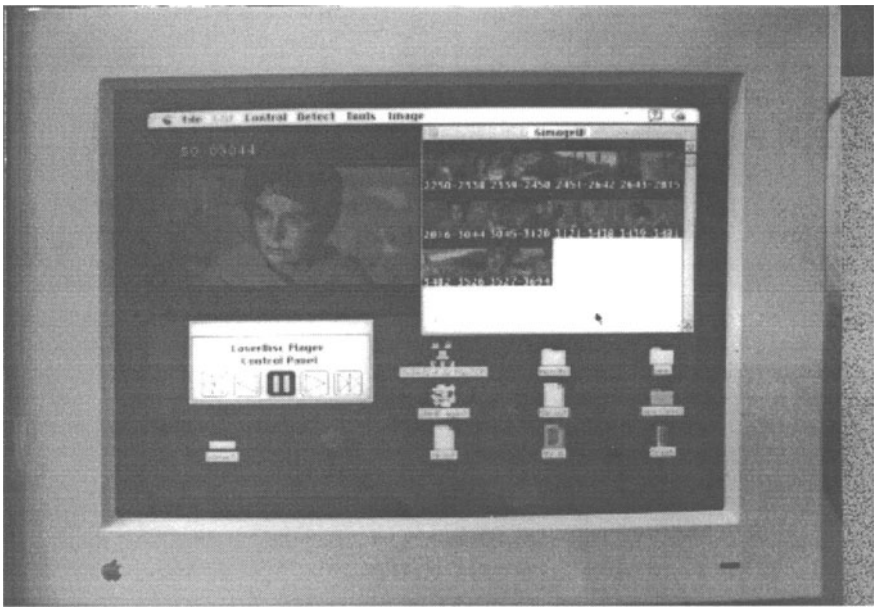

Figure 7 Screen Layout of the Intelligent Video Manipulator

6 CONCLUDING REMARKS

In this paper we have described an approach towards supporting video data management and manipulations using *extended* object-oriented technologies. In particular, conventional object-oriented concepts and features (such as composite objects, is-a hierarchy, method encapsulation, etc.) have provided suitable facilities for storing and manipulating video data objects (VDOs) to a great extent. In order to accommodate and represent dynamic VDO features that arise from video manipulation activities, necessary extensions centred around the notion of *conceptual clustering* have been introduced. Such extended facilities allow tentative, irregular, and/or evolving object collections ("clusters") to be dynamically formed, stored, and manipulated. Such clusters can further impact on, and interact with, the constituent objects (i.e., VDOs) through the functions defined by the active *roles* within the clusters. Example video manipulation activities in the domains of video classification, editing, and production have been considered, and the utility of the clustering facilities in each of the domains is discussed. Finally, we briefly described our experimental prototype environment and system, which is currently being constructed at our institute.

The research described in this paper is being continued and further improved at present. As mentioned, we are close to complete the kernel VDO object-oriented database system, and are adding the conceptual clustering mechanism on top of it. This enhanced VDO database system is then to be interfaced with the devised video classification component (VCC), the latter is also being further extended to support classifying schemes for identifying indoor and outdoor scenes, kung-fu scenes, car racing scenes, sports scenes, etc. In addition to this Mac-based prototype, we are also developing a larger scale prototype based on a persistent object storage management system (namely, EOS from AT&T Bell Lab (Biliris, 1994)) with MPEG on Sun Sparc20 workstations. Other issues of interest include the investigation of the feasibility to incorporate auditory signal processing (ASP) component into the whole system, which may provide us with an additional means for more effective (and meaningful) video data classification and manipulations (e.g., to build keyword indexes from the sound track). Finally, we plan to test and refine our system (upon completion) by applying it into several real-life environments, including TV News Room studios, university Educational Technology Centres, and possibly Telecommunication Companies.

ACKNOWLEDGEMENT

The authors thank the anonymous referees for their constructive comments and helpful suggestions that improved the presentation of this paper. This research is supported, in part, by UPGC Research Grant Council of Hong Kong under grant HKUST 611/94E, and Sino-Software Research Centre of HKUST under grant SSRC 93/94.EG16.

7 REFERENCES

Atkinson, M., F. Bancilhon, D. Dewitt, K. Dittrich, D. Maier and S. Zdonik (1989) Object-Oriented Database System Manifesto, in *Proc. of 1st Int'l Conference on Deductive, Object-Oriented Databases*, Kyoto, Japan, 40-57.

Biliris, A. and E. Panagos (1994) EOS User's Guide (Release 2.1), 600 Mountain Ave, AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.

Chang, S.K. (1992)Image Information Systems: Where Do We Go From Here? in *IEEE Transactions on Knowledge and Data Engineering,* Vol. 4 No. 5, 431 - 442.

Davenport, G., T.G.A. Smith and N. Pincever (1991) Cinematic Primitives for Multimedia, in *IEEE Computer Graphics & Applications,* 67 - 74.

Hirata, K. and Kato, T. (1992) Query by Visual Example - Content based Image Retrieval, in *Lecture Notes in Computer Science,* Vol.580, Springer-Verlag.

Jain, R. (1993) NSF Workshop on Visual Information Management Systems, in *ACM SIGMOD RECORD,* Vol.22, No.3, 57-75.

Kato, T. (1992) Database Architecture for Content-based Image Retrieval, in *SPIE,* Vol.1662, Image Storage and Retrieval Systems.

Kim, W., E. Bertino and J.F. Garza. (1989) Composite Objects Revisited, in *Proc. of ACM SIGMOD Int'l Conference on Management of Data,* 337-347.

Lee, C. M., S.W. Cheng, and M.C. Ip (1993) Camera Break Detection Algorithms and Their Evaluation, *Technical Report HKUST-CS93-10,* Dept. of Computer Science, HK Univ. of Science & Technology (HKUST).

Lee, C. M. and M. C. Ip (1994) A Robust Approach for Camera Break Detection in color Video Sequence, in *Proc. IAPR Workshop on Machine Vision Application (MVA'94),* Kawasaki, Japan.

Li, Q. and J.L. Smith (1992) A Conceptual Model for Dynamic Clustering in Object Databases, in *Proc. of 18th Int'l Conference on Very Large Data Bases,* 337-347.

Li, Q. and M.S. Yuen (1993) Developing a Dynamic Mechanism for Conceptual Clustering in an Object-Oriented DBMS, *Technical Report HKUST-CS93-15,* Dept of Computer Science, HKUST.

Li, Q. (1995) Advanced Functions for Conceptual Clustering in Object Databases, *Technical Report HKUST-CS95-21,* Dept of Computer Science, HKUST.

Masunaga, Y (1989) An Object-Oriented Approach to Multimedia Database Organization and Management, in *Proc. of Int'l Symp. on DASFAA,* 190-200, Seoul, Korea.

Nagasaka, A. and Tanaka, Y (1992) Automatic Video Indexing and Full-Video Search for Object Appearances, in *Transactions of IPSJ,* Vol.33 No. 4.

Rowe, L.A., J.S. Boreczky and C.A. Eads (1994) Indexes for User Access to Large Video Databases, in *Proceedings of IS&T/SPIE Symposium on Storage and Retrieval for Image and Video Databases*, San Jose, USA.

Woelk, D. and W. Kim (1987) Multimedia Information Management in an Object-Oriented Database System, in *Proc. of 13th Int'l Conference on Very Large Data Bases, Brighton*, 319-329.

Xiong, W., C. M. Lee, and M. C. Ip (1995) Net Comparison: A Fast and Effective Method for Classifying Image Sequence, in *IS&T/SPIE Symposium on Storage and Retrieval for Image and Video Databases*, San Jose, USA.

8 BIOGRAPHY

Q. LI received the B. Eng degree from Hunan University, China in July 1982, and the M.Sc and Ph.D degrees (both in computer science) from the University of Southern California in May, 1985 and December, 1988, respectively. From 1989 - 1991, he worked

at the Australian National University as a Lecturer of the Department of Computer Science. Since 1992, he has joinned the newly founded Hong Kong University of Science and Technology (HKUST), where he is now an Assistant Professor at the Computer Science Department. His research interests include object data modeling, schema evolution and object migration, object-oriented video databases, intelligent data/knowledge base systems, information sharing and integration, and interoperable database architecture. Dr. Li is a member of the IEEE Computer Society, and the Association of Computer Machinery (ACM).

John C.M. Lee received his Ph.D from University of Minnesota in1989. From 1989 to 1992 he was a research staff member of the Institute of Systems Science at the National University of Singapore. He is currently an Assistant Professor of Computer Science Department at the Hong Kong University of Science and Technology (HKUST). He received the Digital Equipment Corporation's Alpha Innovator's Award in 1993. His research interests include computer vision, image processing, pattern recognition, artificial intelligent, and robotics. Dr. Lee is a member of the IEEE Computer Society, and the Association of Computer Machinery (ACM).