Semantic Video Annotation and Vague Query

Qiuying Zhang and Mohan S Kankanhalli, Philippe Mulhem*

School of Computing, National University of Singapore,Singapore 117543
E-mail: mohan@comp.nus.edu.sg

**IPAL-CNRS, School of Computing, National University of Singapore,Singapore 117543*
E-mail: mulhem@comp.nus.edu.sg

Abstract

The Digital Video Album (DVA) system described here integrates various cooperating subsystems to index and query video documents according to their semantic content and other metadata. A simple structured model is proposed to represent the video content. This model is compatible with XML Schemas and supports typed attributes and composition relationships. The architecture of DVA is described, and the workflows related to the semi-automatic indexing of videos are presented. The querying processing on video metadata is based on an extension of Boolean search, in a way to avoid empty answers and to rank query results. The query interface is also described.

1 Introduction

Due to the emergence and proliferation of digital videos in recent years, researchers have been keen on video processing in various fields. Video indexing and retrieval are two important issues in a multimedia system.

We distinguish usually two main approaches to represent and retrieve video data. The first one is the physical feature based video model, which automatically parses the low level features and retrieves the video based on those features. The second one is the semantic content-based model. High-level semantic content-based modeling aims to represent the content of the digital video in a more understandable manner, and may use text annotations and extraction of semantic objects to enrich video metadata. Semantic content-based representation of videos allows retrieval using semantic criteria.

In this paper, we introduce the video data model of Digital Video Album (DVA) system that supports semantic content-based video indexing and vague query. Based on a predefined XML-Schema, this model allows high-level semantic annotation to describe the content of video including objects and events. The annotation process makes use of low-level features processes during tracking of

objects and detection of faces. We also present the vague query processing which supports ranked results generation.

This remaining of this paper is organized into 4 sections. The related works on video indexing and retrieval models are given in the next section. Section 3 describes the formalization of the video model and query processing. The following section 4 provides the system overview and the implementation issues. Finally, section 5 presents the conclusions and directions for future work.

2 Related Works

2.1 Physical Feature Based Model

Physical feature based modeling supports indexing/querying of videos based on low level features [Chang et al. 1998, Flickner et al. 1995, Gupta & Jain 1997]. It provides algorithms that automatically recognize the important audio-visual features in the videos without human intervention.

The representative query-by-example system QBIC [Flickner et al. 1995] allows queries on large image and video databases based on example images, user-constructed sketches and drawings, selected color and texture patterns etc. Gupta and Jain [Gupta & Jain 1997] propose the Virage system based on features allowing query specification based on a collection of nine different tools.

Another representative physical feature based work is VideoQ [Chang et al. 1998], which automatically extracts objects and features from the video. Users are allowed to search for videos based on a set of visual features and spatio-temporal relationships of regions. The novel idea of VideoQ is that users may formulate the query by a temporal sketch with key attributes motion and temporal duration.

Fully automated systems seldom require human input, which is a huge advantage. However the query and retrieval is limited and sometimes ineffective, since low-level features are too concrete for the end users. In addition, due to the limited semantics in the models, it is nearly impossible to retrieve specific semantic objects or stories. Besides, the physical feature based model involves some challenging questions. For example, the survey [Brunelli et al. 1996] of video indexing raises questions like “what is a good measure of visual similarity” and “how to detect a specific person or event”, which are very hard to solve.

2.2 Semantic Content-based Modeling

Semantic content-based modeling is the other main approach to provide content-based access in the multimedia system [Jiang et al. 1997, Kankanhalli & Chua 2000, Omoto & Tanaka 1993]. Data models based on semantic content are capable of supporting more natural queries. Aiming to bridge the gap between low

level media features and high level semantics, M. Naphade, T. Kristjansson, B. Frey and T. Huang in [Naphade et al. 1998] use an integrated HMM (Hidden Markov Model) model (called multiject) taking both audio and visual features as observations to detect events. This analysis cannot however fully interpret the semantics of video, because fully automatic semantic interpretation of video data is not feasible given the state of the art of computer vision and machine intelligence.

Therefore, among all the possible approaches to get video semantics, one efficient method is to use text annotation-based models [Aguierre & Davenport 1992, Chua & Ruan 1995, Hjelsvold & Midtstraum 1994, Jiang et al. 1997, Kankahalli & Chua 2000, Kokkoras et al. 2002, Omoto & Tanaka 199]) which generally provide the concept structures to model the semantic contents of video. [Chua & Ruan 1995] presents a two-layered conceptual model (shot layer and scene layer), in which the shot layer contains a collection of primitive video shots and the scene layer is used to model the domain knowledge. The advantage of such approach is to segment video automatically. But the structures are not flexible. The stratification-based approaches [Aguierre and Davenport 1992, Hjelsvold & Midtstraum 1994, Jian et al. 1997, Kankanhalli & Chua 2000, Kokkoras et al. 2002, Omoto & Tanaka 1993, Weiss and al. 1994] establish the stratification structure on video. In a representative stratification model [Aguierre and Davenport 1992], the video unit is known as a stratum associated with both textual description and a time interval corresponding to a physical segment in the video. Since strata can overlap and encompass each other, the meaning of the video can be flexibly modeled as the combination of all strata present.

The system VideoText [Jiang et al. 1997] proposes a video model that uses free text to annotate the video content. Later, [Kokkoras et al. 2002] integrate the VideoText system with the Conceptual Graph (CG) to model the semantic associations among video annotation. [Kankanhalli & Chua 2000] also proposes a stratification-based video model which focuses more on the entities of the video such as object, dialog, etc. [Omoto & Tanaka 1994] proposed an object oriented video data model, OVID, in which the notion of video objects is introduced to facilitate the identification. This model applies schemaless annotation. [Hjelsvold & Midtstraum 1994] proposed a generic video data model based on annotation. [Weiss et al. 1994] allows users to model nested video structures and define the output characteristics of video segments, and a comprehensive set of temporal operators have been defined within the algebraic video system.

The MPEG-7 [MPEG-7 2001] normative proposal supports descriptions of video content using XML-Schemas for professional use. Our work makes use of such proposal by presenting a simple model specially designed towards home use.

2.3 Retrieval Models

Retrieval model is a major component in an information retrieval system. Among the models proposed in the field of information retrieval, the most well

known models are the Boolean model, the probabilistic model and the vector space model.

The Boolean search model compares Boolean query statements with the terms used to represent document contents. Boolean query statement consists of a variety of terms using the Boolean operators AND, OR, and NOT. The user may narrow or broaden the subject by specifying the relevant amount of information. Due to the exact matching strategy, the result quality may be poor when the query statement is exceedingly simple or complex, either generating too many or too few documents without ranking the output.

The probabilistic retrieval model is based on the computation of relevance probabilities for the documents of a collection. Conventional retrieval model retrieves the document in which the attached keyword set appears similar to the query keywords. In this case the document is assumed to be relevant to the corresponding query [Chowdhury 1999].

Vector processing model establishes a term set, called term vectors, for both the stored records and information requests. Collectively the terms assigned to a given text are used to represent text content. Each document in a set of documents can be represented by a number of properties. The similarity between two documents is computed as a function of the number of properties that are assigned to both documents.

In this paper, we use a Boolean search model with vague match to solve the problem of strict match and generate ranked results. The query processing is explained in section 3.2.

3 Video Document Model and Query Processing

This section presents the design of video annotation model and query processing. The video annotation model is introduced first, which includes definition of Structure Type (ST), Predefined Attributes (PA) and Document Model (D). The second part explains the principles of the query model.

3.1 Video Annotation Model

The DVA (Digital Video Album) system builds an annotation based video model that accomplishes well-defined levels of abstraction of video data. The concepts, *Video*, *Sequence* and *Shot*, are used to represent video, which represents a video file, a sequence of consecutive frames and a video segment made from a one-time camera work respectively.

In addition, our system categorizes two types of objects appearing in video. One is *Person* to represent human being; the other is *Object* that can stand for everything except People. This concise classification not only avoids complex

usage to describe objects in the video for the end users but also brings convenience and efficiency. Each data type has related a predefined attribute set.

3.1.1 Model of a Video Base

The Structured Document Base B is defined as $B \equiv (S, D)$, with:

S the set of document types which are defined in detail in the following section 3.1.2.

D the set of documents in the document base: $D=\{D_j\}$

3.1.2 Model of a Document Type

The model of the document type is defined as below:

$S \equiv (ST, PA, R_{st})$, where

ST defines the structure of documents. Each structured document must fit the framework corresponding to the skeleton (i.e. ST). This skeleton is a syntactic structure (see section 3.1.3).

PA is a set of predefined structural attributes to describe video. Each predefined attribute is defined by attribute name and domain. A function *getAdomain* can obtain the domain according to the attribute name.

R_{st} links the attributes to the types of structural elements. This relation associates each type of structural elements of ST to the structural attributes of the document. The set A_{name} contains the set of the names of the predefined attributes: $R_{st} \subset TYPE \times A_{name}$

3.1.3 Model of a Structure Type

We represent here the structure that defines the types of elements and how such types can be composed:

$ST \equiv (TYPE, \leq_{tcomp})$

TYPE is the set of the types of structural elements of a set of documents. In our system, the TYPE set includes *Video*, *Sequence*, *Shot*, *Object* and *Person*.

③ $\leq_{tcomp}$ is the relation to show the composition of types in TYPE set. It is a binary relation on TYPE: $\leq_{tcomp} \subset TYPE \times TYPE$. The notation $t_j \leq_{tcomp} t_i$ means that the structural elements of type t_i can be directly composed of elements of type t_j.

The figure 1 demonstrates how one type can be composed of other types in our model. *Video* can be composed of varies types *Person*, *Sequence*, *Shot* and *Object*. *Shot* can be composed of *Sequence*, *Person* and *Object* directly (here *Person* and *Object* are omitted in the figure). *Sequence* may consist of *Object* and *Person*. The type *Person* is composed of the type *Object*.

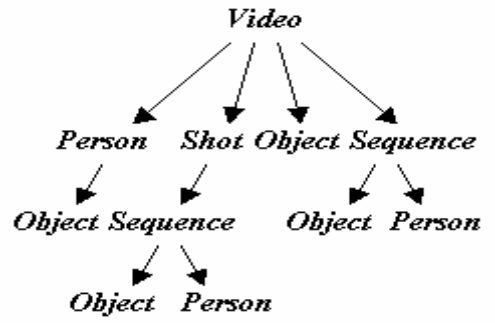


Figure 1. Composing Relationship of Types.

3.1.4 Predefined Attributes (PA)

The predefined attributes (PA) of structured documents are defined by a triplet:

PA ≡ (A_{name}, A_{domain}, *getAdomain*), where

A_{name} is the set of the attribute name, which includes VID, VideoName, Topic, VideoShotDateFrom, VideoShotDateEnd, VideoLocation (Country, State, City and Street), VideoLength, Rating, Description, SID, shot, BeginFrame, EndFrame, SequenceName, Event, SequenceShotDateFrom, SequenceShotDateEnd, SequenceDescription, OID, ObjectName (Name and Alias), PhotoRecord, OnePicture, PicturePath, Location(Country, State, City and Street), PID, PersonName (FirstName, MiddleName, FamileyName, Alias), BirthYYMMDD, BirthLocation (Country, State, City and Street).

A_{domain} is the set of the possible values for the attribute

getAdomain is a partial function that links each attribute name of the set A_{name} to a specific definition of A_{domain}: *getAdomain* : $A_{name} \rightarrow A_{domain}$

According to section 3.1.2, R_{st} of S links the PA to the TYPE set defined in ST. $R_{st} \subset TYPE \times A_{name}$

Here a simple and complete example is shown to illustrate the structure of PA.

- PA ≡ ({Topic, VideoShotDateFrom, Event, Rating}, {string, date, integer}, *getAdomain*)
- *getAdomain* obtains the domain of specific element in the set A_{name}. The domain of Topic is string so that *getAdomain* (Topic) = string; Similarly, this function finds the related domains of the specific attribute names: *getAdomain* (VideoShotDateFrom) = date; *getAdomain* (Event) = string; *getAdomain* (Rating) = integer

- The relation R_{st} links the name of the attribute to the TYPE set. It indicates which type the attribute belongs to. (*Video*, Topic) $\in R_{st}$, (*Video*, VideoShotDateFrom) $\in R_{st}$, (*Video*, Rating) $\in R_{st}$, (*Sequence*, Event) $\in R_{st}$, (*Sequence* , Rating) $\in R_{st}$

3.1.5 Document Model (D)

A structured document content model D is defined as an aggregation of structural elements related to predefined attributes, this is semantic structure:

D ≡ (OS, $<_{comp}$, *type*, *value*), where

OS is the set of structural elements

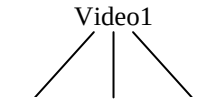
$<_{comp}$ is the containing relationship on OS. It is a binary relation on OS: $<_{comp} \subset OS \times OS$. The notation $o_i <_{comp} o_j$ means that the object o_j is composed of the object o_i.

type is the function that obtains the type of each structural element of OS: *type*: OS → TYPE. OS is defined in D and TYPE is defined in S.

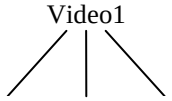
value is the function that obtains the value of attribute α, (α∈ PA) from A_{name} for one structural element of OS:

- *value*: $A_{name}(\alpha) \times OS \rightarrow$ *getAdomain* (α); According to section 3.1.2, R_{st} of S links the A_{name} of PA to the TYPE set defined in ST.

Below there is an example of Document Model.



A document model D1 = {{Video1, S1, S2, P1}, , *type*, *value*); OS is the set of structural elements of D1; In this example,



Video1, S1, S2, P1 ∈ OS and shows the containing relation on OS. Structural element Video1 is composed of structural elements S1, S2 and P1.

Function *type* obtains the type of each structural element in OS. *type*(Video1) = *Video*, *type*(S1) = *Sequence*, *type*(S2) = *Sequence*, *type*(P1) = *Person*

Function *value* obtains the value of the attributes of a structural element. The attributes can be the element of the PA. In the example 3.1.4, Topic, VideoShotDateFrom, Rating, Event ∈ PA. The predefined attributes can be associated with the specific type in TYPE set by the relation R_{st}. (*Video*, Topic) $\in R_{st}$, (*Video*, VideoShotDateFrom) $\in R_{st}$, (*Video*, Rating)

$\in R_{st}$, (*Sequence*, Event) $\in R_{st}$, (*Sequence* , Rating) $\in R_{st}$.
In addition, function *type* connects the structural element of OS to TYPE set. *type*(Video1) = *Video*, *type*(S1) = *Sequence*, *type*(S2) = *Sequence*, *type*(P1) = *Person*
Therefore, the predefined elements can be linked to the structural elements in OS. Based on that, function *value* can obtain the value of the attributes. *value*(Topic, Video1) = "Birthday Party"; *value*(VideoShotDateFrom, Video1) = "01/03/2002", *value*(Rating, Video1) = "good", *value*(Event, S1) = "cheers", *value*(S1,Rating) = "3", *value*(S2, Event) = "singing", *value*(S2, Rating) = "good"

3.2 Query Processing

This section presents the concept of vague query, followed by a formal definition of query expression and the application of the data metric in our system. Finally the calculation of the distances between documents and a query is described.

3.2.1 Generalities on Vague Queries

Section 2.3 shown that Boolean search is unable to rank the results because the query result can only be true or false. Due to its rigid qualification, Boolean search, therefore, often retrieves too narrow or too broad items.

Motro [Motro 1988] presented an approach which employs data metrics to retrieve results. Data metrics are meant to compute the distances between the values of the elements in a given domain. If there is no strictly matching result, the system returns some close alternatives.

Adapting the work of [Motro 1988], we apply the vague query into the formulation of Boolean search with a comparator "*similar-to*". We employ suitable data metrics for specific domains like time interval and character strings.

3.2.2 Query Expression

Obtaining the primary query expression composed of sub-formulas connected with "AND" and "OR" from the user interface, the query composer normalizes it into the normal query expression *expr*. The system analyzes the normal query expression *expr* to retrieve the results in the data storage. Below is the syntax definition for a query expression *expr*.

expr ::= *expr* AND *expr'* | *expr'*
expr' ::= *expr'* OR *cond* | *cond*

cond ::= *keywords* OP_1 *string* | VideoShotDateFrom OP_1 *string* && VideoShotDateEnd OP_1 *string* | SequenceShotDateFrom OP_1 *string* && SequenceShotDateEnd OP_1 *string*

keywords ::= VideoName | Topic | Event | VideoLocation | Country | State | City | Street | VideoShotDateFrom | VideoShotDateEnd | Evaluation | SequenceName | SequenceShotDateFrom | SequenceShotDateEnd | ObjectName | FirstName | MiddleName | FamilyName | Alias | BirthDate | BirthCounty | BirthState | BirthCity | BirthStreet | CurrentLocation |

string ::= $[A\text{-}Za\text{-}z0\text{-}9]^+$

OP_1 ::= *similar-to* | != | =

Expression *expr* is the query condition composed of a chain of sub-formulas *expr'* connected with "OR" operator. Each sub-formula *expr'* is composed of a series of atomic formulas connected with "AND" operator. Generally, an atomic formula is composed of the *keywords*, comparison operators (OP_1) and *string*. OP_1 represents the three optional comparison operators: "=" is the equality operator, "!=" is the inequality operator and "*similar-to*" is the vague comparator. *string* is a non-empty alphanumeric character string. *keywords* contains the set of the names of the predefined attributes.

cond represents the atomic formula which contains one basic query condition. Generally it contains three criteria: *keywords*, comparison operator (OP_1) and *string*, which are necessary to represent a query condition. Since the special atomic formula for time interval should be integrated in the system, there exist "VideoShotDateFrom OP_1 *string* && VideoShotDateEnd OP_1 *string*" and "SequenceShotDateFrom OP_1 *string* && SequenceShotDateEnd OP_1 *string*" in *cond*. The expression of "VideoShotDateFrom OP_1 *string* && VideoShotDateEnd OP_1 *string*" denotes the time interval of *Video* type while the other indicates that of the *Sequence* type. In the expression of "VideoShotDateFrom OP_1 *string* && VideoShotDateEnd OP_1 *string*", "VideoShotDateFrom OP_1 *string*" represents an instant time of the beginning of the video. And the ending time of the video is kept by "VideoShotDateEnd OP_1 *string*". Similarly, the "SequenceShotDateFrom OP_1 *string*" denotes the beginning time of *Sequence* and "SequenceShotDateEnd OP_1 *string*" shows the ending time.

Let α_i (i = 1, …k) represent the sub-formulas of *expr*, which is *expr'*; $\beta_{i,j}$ (j = 1,….n_i) denote the sub-formulas of α_i ; $\beta_{i,j}$ represent an atomic formula.

A query expression *expr* can be represented as:

(α_1) AND (α_2) AND (α_3) AND ····· AND (α_k) i = 1, …k

Each α_i can be represented as:

$(\beta_{i,1})$ OR $(\beta_{i,2})$ OR $(\beta_{i,3})$ OR ····· OR $(\beta_{i,j})$ j = 1,….n_i

Therefore, the query expression *expr* is a composed of atomic formulas via parenthesis and two operators.

$\{(\beta_{1,1})$ OR $(\beta_{1,2})$ OR ··· OR $(\beta_{1,ni})\}$ AND (α_2) AND ··· AND (α_k) i = 1, …k

3.2.3 Data Metrics of two domains

Since video data contains rich temporal information that is often used to filter video data, our system carefully considers handling of temporal information.

In our video model, the vague match is associated with two domains. One is time interval, and the other is string type. The first one is composed of instant beginning time and instant ending time; the latter one contains unlimited terms series. Obviously, traditional precise match is not suitable to the two types of attributes. For example, the query is to find the video clips which start at April 1st to 10th. The video clips shot from March 31st to April 11th can also serve as the related result since it contains the time interval of the query and the time period is quite similar. However, rigid match cannot indicate the differences between the similar results and the results of no relevance.

To solve this problem, vague match mechanism with *data metric* for specific domains are employed to enhance the power of query model. To determine the similarity between results, the notion of distance is introduced here. A *data metric* is used to calculate the distance of two values of same domain. Therefore, the distance between the target qualification and instance values of same domain can judge the similarity. Evidently, short distance indicates high similarity.

Given a set S, A *data metric* for S is a function d: $S \times S \rightarrow R$ such that for all x, y, $z \in S$,

(i) $d(x, x) = 0$;

(ii) $d(x, y) > 0$, $x \neq y$;

(iii) $d(x, y) = d(y, x)$

(iv) $\forall z \in S : d(x, y) \leq d(x, z) + d(z, y)$

Considering the characteristic of the two domains, time interval and string, we utilize the data metric defined as below:

For one set T, $A \in T$, $B \in T$, a *data metric* for T is a function d: $T \times T \rightarrow$ R. For all A, B, $C \in T$, the four properties of *data metric* are satisfied.

$$d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \quad 0 \leq d(A,B) \leq 1 \quad (1)$$

Here domain T can be either time interval or string. The first three features of the definition are obviously satisfied. The fourth property, known as the Triangle Inequality, was proven by Yianilos [Yianilos 1991].

3.2.4 Matching Value Computation

The normalized query formula is composed of the sub-formulas α_i (i = 1, …k) connected by "AND" operator. Each sub-formula is composed of a series of atomic formulas $\beta_{i,j}$ (j = 1,….n_i) connected by "OR" operator.

Assume that R is one instance of structured type which includes all the corresponding domains in the query, and Q is the query condition. Based on that, $R_{i,j}$ represents the structured information which just includes the domain and related value corresponding to $\beta_{i,j}$. The domain of $R_{i,j}$ is matched against the domain of $\beta_{i,j}$. $R_{i,}$ stands for structured information composed of $R_{i,j}$ (j = 1,2…n_j). Thus $R_{i,}$ matches with α_i. All R_i are components of R. Therefore, the R contains all the domains of Q and their values.

Let α_i' denote the distance value between α_i (i = 1, …k) against result R_i, $\beta_{i,j}$' denote the distance value between $\beta_{i,j}$ (j = 1,….n_i) and $R_{i,j}$ and D' denote the distance between the query Q and instance R.

The distance D' is the square root of the sum of the squares of each α_i'. And the distance value of each α_i is the minimum distance value of atomic formulas $\beta_{i,j}$ (j = 1,….n_i). The ideal distance of one complete query expression is 0.

(i) $\beta_{i,j}' = d(\beta_{i,j}, R_{i,j})$ i = 1,2,3…k, j = 1,…n_i .

(ii) $\alpha_i' = getDistance(\alpha_i, R_i) = \min(\beta_{i,j}')$ j = 1,…n_i

(iii) $D' = getDistance(Q, R) = \sqrt{\sum_{i=1}^{k} (\alpha_{i'} \times \alpha_{i'})}$ i = 1,2,3…k

The expression (i) calculates $\beta_{i,j}'$, the distance of atomic formula via data metric, that is, the distance between the ideal result $\beta_{i,j}$ and result $R_{i,j}$. $R_{i,j}$ is sub-structure of R. $\beta_{i,j}$ and $R_{i,j}$ are belong to the same domain. The second formula (ii) computes the distance between α_i and R_i. α_i is a sub-formula in query condition, while R_i is the instance structure corresponding to α_i in result R. The function compares values of the atomic distance, $\beta_{i,j}'$, and sets the minimum value to α_i' which represents the distance of sub-formula. The last formula (iii) uses function *getDistance* to calculate the distance D' between query Q and result R. The value of D' is equal to or greater than 0. Therefore, the system is able to rank the results in order of increasing distances.

4 The DVA (Digital Video Album) System

4.1 DVA Software Architecture

The figure 2 shows the three layers of the DVA (Digital Video Album) system. The layer 1 is the storage layer which contains video metadata DB, video database

and images DB. The layer 2 provides functionalities based on the first layer. Video indexing, query processing, shot detection etc all process the data of layer1 and support the tools of layer 3. The layer 3 is application layer which provides tools to the end users. It calls the function models in layer 2 without handling layer 1 directly. This three-layered architecture makes the system work concise and cooperation efficient.

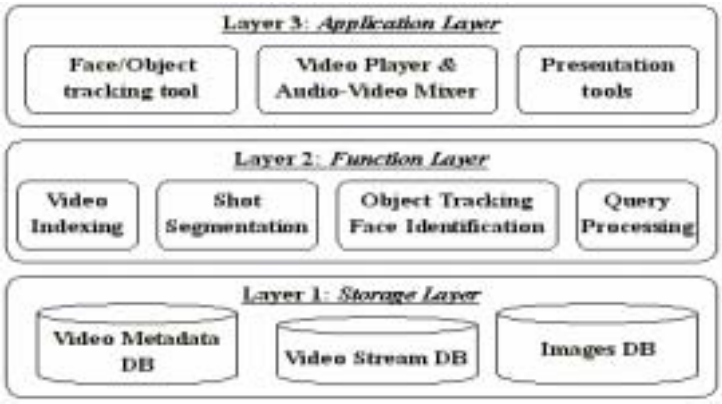


Figure 2. Software Architecture of Digital Video Album (DVA) System.

4.2 Video Indexing/Annotation

4.2.1 Video Indexing/Annotation Architecture

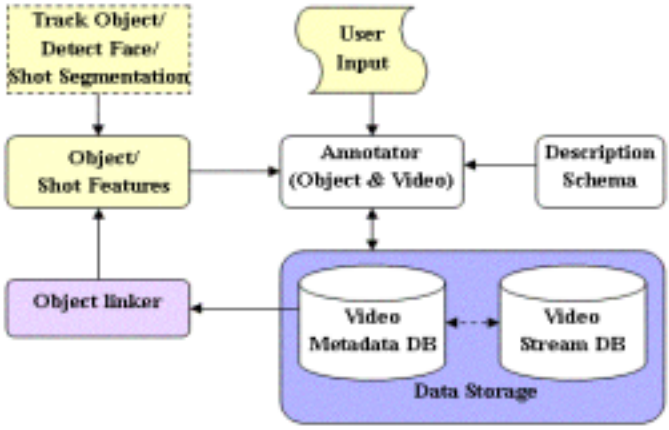


Figure 3 Software Architecture of DVA Index/Annotation.

The architecture of DVA indexing model is shown in Figure 3. The indexing/annotation system consists of two types of metadata sources, user manual input and automatically extracted features. The former is the text input according to our predefined schema to describe video, the latter includes object tracking features, human faces and shot boundaries which are generated by the computer with the launching action of human. In the figure 3:

The **Annotator** enables users to annotate video data and objects appearing in the video. It can communicate with the video metadata DB with the operations save, extract, modify and delete.

The **Description schema** is the predefined XML schema for describing video and object. It will validate the format and content of the video/object description info in Annotator during XML parsing.

The **Object linker** maintains the relationship between low level features and its object, and makes the features associated to the specific object in video metadata DB.

The **Data Storage** is composed of two parts, one is video metadata database which includes the information about video, objects and their low level features, and the other is video stream database. There is a match between each other so that data could be synchronized. For each video clip in video stream DB, there is a corresponding video description file in metadata DB which records the annotation info.

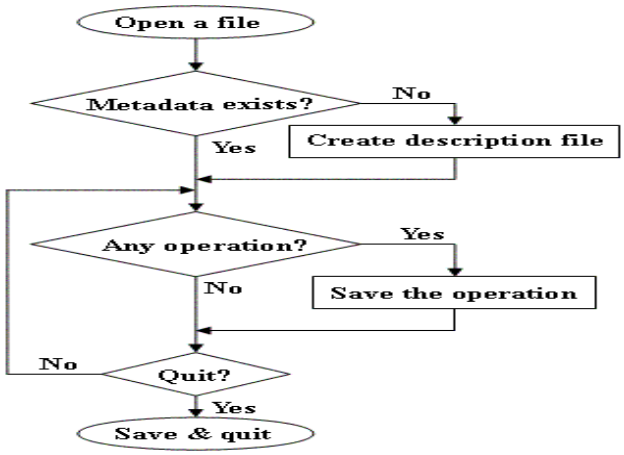


Figure 4 Workflow of Video Annotation.

When the project opens a video file, the annotator extracts the information about this video from metadata DB. If this video is a new file, the annotator creates a corresponding description XML file. When the annotator gets the user manual

input annotation, it just keeps the info in the related XML description file. After parsing and validating the file by means of description schema and XML parser, the annotator calls the function to save this file into metadata DB. If the user launches other operations such as delete, modify, the annotator does the matching process.

4.2.2 Workflow of Video Indexing/Annotation

With the help of semantic annotation, the computer can identify the features with specific objects and integrate that info into a whole information system. The figure 4 shows the workflow of such video annotation.

As the user opens a video file, the system first checks the index file. If the description file exists, the metadata is to be extracted and presented to the user. Otherwise, the system creates a new description file with an ID call VID for this video. Then the user may do any operation such as tracking object, modifying a sequence etc. After the modification, the metadata is still kept in the memory. When the user conforms to close this video, the data is saved into the metadata DB of the hard disk.

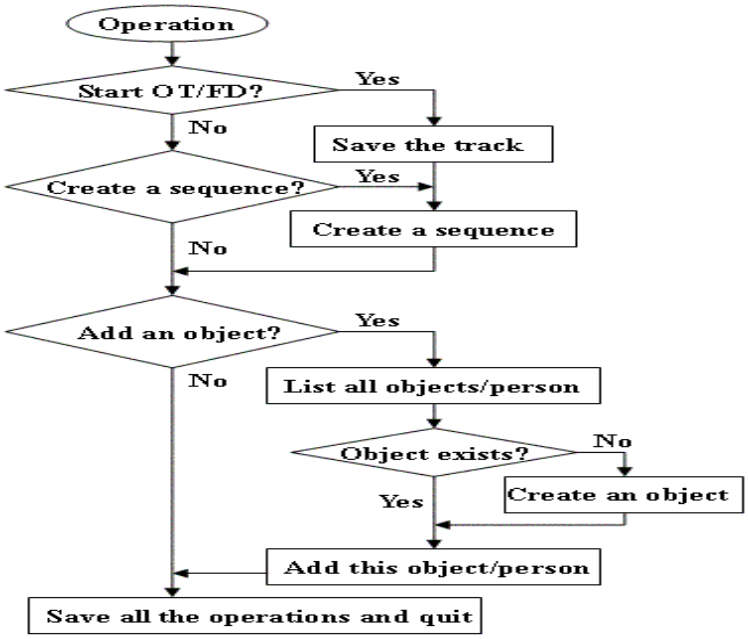


Figure 5 Workflow for annotation creation.

As shown in figure 5, the user may start object tracking/face detection, create a sequence or annotate an object/person. The system first saves the tracked data as a sequence for the related object. The annotation can be attached to the newly created

sequence. And the features of the tracks or human faces must be associated with certain object/person so that they can be indexed and reused. In addition, the new sequence can link the object/person to show that appears in this video.

After the annotation of the new sequence, the system first extracts all the objects and people in metadata DB and presents them to the user so that the user can link the result of the tracking to the semantic object. If the object or person to be associated does not exist, the system creates new object/person to save into metadata DB. If the right object/person is just selected to associate with related faces, track and sequence, the link between low level features and high level features can be built successfully.

4.3 Video Querying/Retrieval

Figure 6 presents the architecture of query processing. The user interface contains several models.

The **Query Composer** obtains user's query expression and translates it into a normalized final expression which satisfies the syntax definition in section 3.2.2. **Annotator** and **Data Storage** are defined in the previous section 4.2. The **Video Player** decodes MPEG stream and displays it for the user from video database. The **Presentation Tool** is a multimedia editor that allows user to combine and organize different multimedia elements such as texts, images, audio and videos into a presentation for sharing the video.

The query composer normalizes the query expression and calls query processing. During the query processing, the query kernel first searches the video metadata database, calculates the distances between the query and videos, and presents the ranked results according to the distance in the interface.

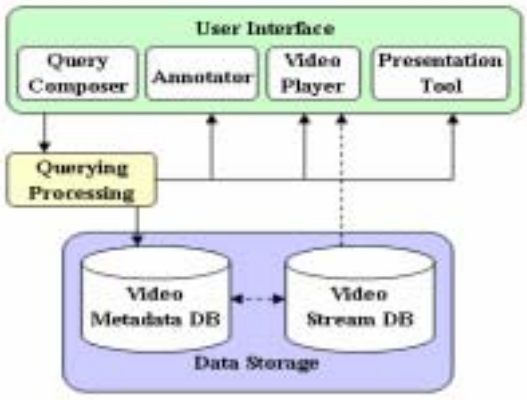


Figure 6 Architecture of the Video Retrieval processing.

4.3.1 User Query Interface

Figure 7 shows the query interface. It is composed of two main components: the upper of the interface composes query expression for retrieval, and the lower of the interface is to present the query result as well as the metadata of the active video clip.

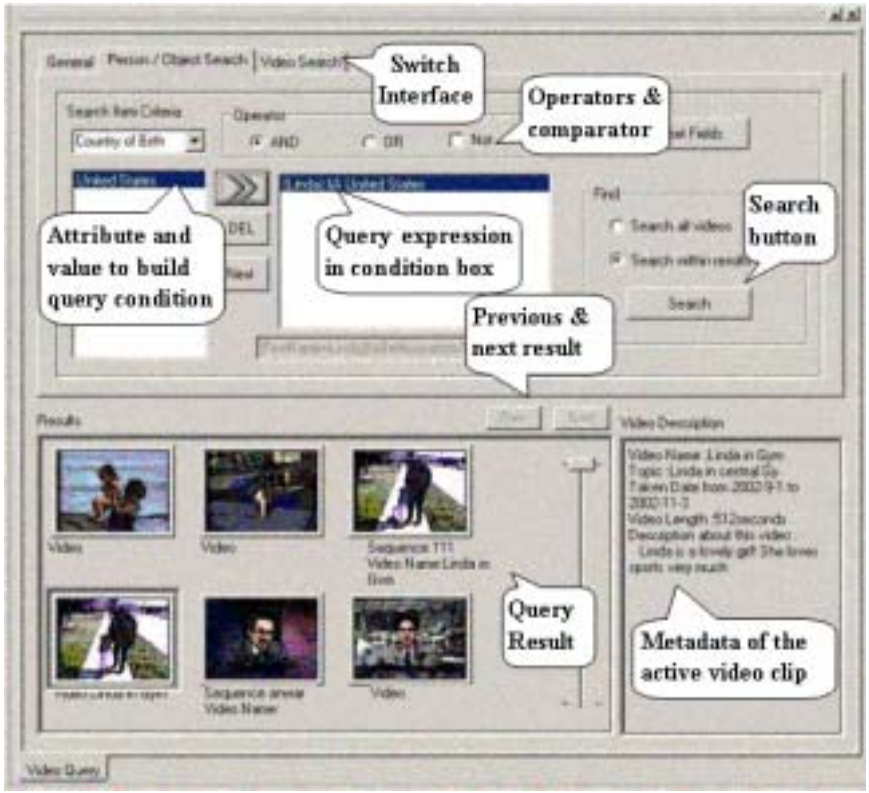


Figure 7 Query Interface Screenshot.

.

4.3.2 Examples

A simple sample query expression is given here to illustrate the distance calculation.

A vague query searches the video which contains events that happened between Jan 1st, 2002 and Jan 15th, 2002. Besides, the topic must contain either “travel” or “trip”. Based on that, the query expression Q1 can be represented as:

(((Topic *similar-to* travel) OR (Topic *similar-to* trip)) AND (VideoShotDateFrom *similar-to* 2002/01/01 && VideoShotDateEnd *similar-to* 2002/01/15)).

Suppose there are three results candidates R1, R2 and R3. Each is a structural element which includes the values and domains of "Topic" and time interval. The values for each element are listed here.

R1.Topic = "last trip to LA"
R1.VideoShotDateFrom = 2002/01/11
R1.VideoShotDateEnd = 2002/01/20

R2.Topic = "wonderful travel in China"
R2.VideoShotDateFrom = 2001/12/27
R2.VideoShotDateEnd = 2002/01/20

R3.Topic = "sea trip"
R3.VideoShotDateFrom = 2002/01/03
R3.VideoShotDateEnd = 2002/01/15

According to the formulas in section 3.2.4, we calculate the distance for each result.

D' = *getDistance* (Q1, R1) = $\sqrt{(\min((1-0\div 5),(1-1\div 4)))^2+(1-5\div 20)^2}$ = 1.061

D' = *getDistance* (Q1, R2) = $\sqrt{(\min((1-0\div 5),(1-1\div 4)))^2+(1-15\div 25)^2}$ = 0.85

D' = *getDistance* (Q1, R3) = $\sqrt{(\min((1-1\div 2),(1-0\div 3)))^2+(1-13\div 15)^2}$ = 0.52

The result is then R3, R2 and R1 in the increasing order.

5 Conclusion and Future Work

The Digital Video Album (DVA) system is an integration of the various subsystems that caters to specific aspects of digital video processing. We focused in this paper on the index/annotation model and processing, dedicated to primarily providing an interface to access the metadata in the XML documents, and links low level features to the high level concepts.

In the query processing, the system retrieves the video data from XML based metadata storage, and presents the ranked video clips from the video database. A noteworthy mechanism, vague query, is adapted from the work of Motro [Motro 1988] in this part. It incorporates a data metric for the domains of time intervals and string to compute the distance between target qualification and candidate qualification which improves the capability of video retrieval.

However, it is still evident that there are limitations to the current system. The quality of both index and query largely depends on the richness of the metadata of the video. And the Boolean queries may be difficult for ordinary people to compose a query in a logically correct way.

Future work is required to extend the work and enhance the system. Firstly, a relevance feedback system may improve the accuracy of the query. Secondly, other intelligent methods like audio segmentation may help to detect meaningful sequences, Lastly, the users may be allowed to express the relative importance of terms in the query condition, which balances the query results according to the users' preferences.

In conclusion, the existing system presents various potential areas for enhancement. Integration of these enhancements would significantly strengthen and extend the power of the current system.

Reference

Aguierre T.G. Smith and Davenport G.. "*The Stratification System: A Design Environment for Random Access Video*". In Proc. of 3rd Int' l Workshop on Network and Operating System Support for Digital Audio and Video, pp 250-261, 1992

Brunelli R., Mich O. & Modena C. M. "*A Survey on Video Indexing*". IRST Technical Report 9612-06, 1996

Chang S.F., Chen W., Meng H., Sundaram H., Zhong D.. "*A Fully Automated Content Based Video Search Engine Supporting Spatio-Temporal Queries*". In IEEE Transactions on Circuits and Systems for Video Technology, 8(5), pp. 602-615, 1998

Chowdhury G.G., "*Introduction to Modern Information Retrieval*". Library Association Publishing, London, pp.158 – 177, 1999

Chua T.S. and Ruan L.Q.. "*A Video Retrieval and Sequencing System*". ACM Transactions on Information Systems. Vol. 3, No. 4, pp. 373-407, October 1995.

Flickner M., Sawhney H., Niblack W., Ashley J., Huang Q., Dom B., Gorkani M., Hafner J., Lee D., Petkovic D., Steele D., Yanker P., "*Query by Image and Video Content: The QBIC System*", IEEE Computer, 1995

Gupta A. and Jain R., "*Visual Information Retrieval*", Communications of the ACM, Vol.40, No. 5, pp. 70-79, May 1997.

Hjelsvold R. and Midtstraum R., "*Modeling and Querying Video Data*". In Proc.of the 20th Int'l Conference on Very Large Databases (VLDB'94), Santiago, Chile, 1994.

Jiang H., Montesi D., and Elmagarmid K.. "*Videotext database systems*". In IEEE Int' l Conference on Multimedia Computing and Systems, Ontario, Canada, June 1997.

Kankanhalli M.S., Chua T.S., "*Video Modeling Using Strata Based Annotation*", IEEE Multimedia, Vol. 7, No. 1, pp. 68-74, 2000

Kokkoras F., Jiang H., Vlahavas I., Elmagarmid A.K., Houstis E.N. and Aref W., "*Smart VideoText: A Conceptual Graph based Video Data Model*", *ACM Multimedia Journal* (accepted for publication), 2002

MPEG-7 Committee, *Overview of the MPRG-7 Standard (version 6.0)*, Report ISO/IEC JTC1/SC29/WG11 N4509, J. Martinez Editor, 2001.

Motro A.. "*VAGUE: A user interface to relational databases that permits vague queries*". ACM Transactions on Office Information Systems (TOIS), Vol.6 No.3, pp.187-214, July 1988

Naphade M.R., Kristjansson T.T., Frey B.J. and Huang T.S., "*Probabilistic multimedia objects (multijects): A novel approach to video indexing and retrieval in multimedia systems*", in Proc. of IEEE Int' l Conference on Image Processing, Vol. 3, pp. 536-540, October. 1998

Omoto E. and Tanaka K., "*OVID: Design and Implementation of a Video-Object Database System*," IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 4, August 1993

Salton G. and McGill M. J.. "*Introduction to Modern Information Retrieval*", McGraw Hill Com- puter Science Series, New York, 1983

Weiss R., Duda A., and Gifford D.. "*Content-based Access to Algebraic Video*". In IEEE Int' l Conference. on Multimedia Computing and Systems, Boston, USA, 1994

Yianilos P. N., "*Normalized Forms for Two Common Metrics*," Tech. Rep., The NEC Research Institute, Princeton, New Jersey, December 1991