

# The Development of A Video Metadata Authoring and Browsing System in XML

Andrew Yao & Jesse Jin

School of Computer Science & Engineering  
The University of New South Wales  
Sydney 2052, Australia  
{andrewy, jesse}@cse.unsw.edu.au

## Abstract

Facing a large amount of rich visual video information, conventional video search techniques such as fast forward/rewind are no longer sufficient. Users want to be able to browse, to be selective at what they see just like how they have accessed textual information. This creates a problem because raw video bits do not possess the same user-level information as text and thus are not directly search-able in the same way. Consequently, video needs to be retrieved and indexed through its semantic content represented in a well structured manner. Thus the challenge is to provide ways of creating this well structured information effectively.

Today's technology in multimedia computing however, is a long way from providing solutions for fully automatic video content extraction. Thus we are exploring solutions of video annotation, providing tools to facilitate a human (the annotator), whose role is to annotate and extract video semantic content. The added semantic information allow more effective video information retrieval and management.

In this paper, we propose a hierarchical metadata model to represent video information. This model consists of two separate hierarchies of metadata. The first hierarchy is a directed acyclic graph, captures the relationship between video segments at the semantic level. The second hierarchy is a object composition graph, holds objects that represent meaningful content appearing in the video. We have developed a Video Metadata Authoring and Browsing System that uses video segmentation result and generates the above mentioned hierarchical metadata output in XML. The metadata output uses some terminologies from the current MPEG-7

## 1 Introduction

Interest in digital video is accelerated by advancements in CPU power, storage space, signal processing techniques, and availability of broader bandwidth in digital communication networks. The increase availability of digital video has led to needs of effective organization and efficient retrieval of video information. Although video contains richer information than text, the retrieval process is substantially harder since raw video pixel bits unlike text, do not convey meaningful information by themselves individually. Consequently video needs to be retrieved through information represented by its semantic meaning. For example, a particular object or an event occurred in

Copyright ©2001, Australian Computer Society, Inc. This paper appeared at Visualisation 2000, Pan-Sydney Workshop on Visual Information Processing, December, 2000. Conferences in Research and Practice in Information Technology, Vol. 2. P. Eades and J. Jin, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

a video can be understood by the viewer because the pixel correlation in both spatial and temporal domain (see section 2).

Textual information is usually organized in a hierarchical structure, i.e. volume  $\Rightarrow$  book  $\Rightarrow$  chapter  $\Rightarrow$  section  $\Rightarrow$  sub-section  $\Rightarrow$  paragraph  $\Rightarrow$  word. There is usually also a Table of Contents or indexes to facilitate search of a particular topic or a key word in a book. (Corona System, (Yap, Jin & Feng 1999, Page 1)). A similar technique can be applied to organize video information to overcome limitations in traditional video search methods: add additional metadata information to the video, and this metadata can enable intelligent, efficient access and management of video data. According to IEEE Mass Storage Systems and Technology Committee's Definition quoted in (Xia, (Xia & Jin 1999)),

Metadata is information about:

1. The stored information entities, which include semantics/information content, structural mapping to storage, type and encoding of elements, relationships among entities, format/structure/type, related data and inferential/derived information;
2. Storage management and administration, which includes location/name, access time and method of access;
3. Use of storage and entities, which includes permissions, usage and history.

We aim to provide a system of tools that will be useful and will ease the process of building the video metadata information. we use the video segmentation result produced by the **Corona system** (Yap et al. 1999) and provides the annotator an environment to build metadata information of the first type mentioned above: organize video segments in a hierarchical manner, extract static frame objects via image segmentation techniques, input textual annotation and establish m-n relationships between meaningful objects or concepts with the corresponding video segment. The system is called **Vimix (V**ideo **M**etadata **I**n **X**ML). Useful applications can be found in areas such as media broadcasting, distance learning, video-on-demand, video conferencing and entertainment.

### 1.1 Existing Systems

Several tools have been implemented to facilitate video metadata creation in various ways. Some of these tools are discussed here.

The **Corona System** (Yap et al. 1999) provides ways of segmenting a video into multiple shots. This is an important preprocessing step as initial video data is organized as a linear sequence of frames that do not support video annotation directly.

The **Vane System** (Carrer, Ligestri, Ahanger & Little 1996) constitutes a semi-automated tool that facilitates the creation of large video database. It provides time-based, sequences-scenes-shots model to segment and annotate a video and the resulting metadata is stored in **SGML** (Standard Generalized Mark-up Language, ISO 8879) format.

The **VMGS and ViMetaVU Systems** (Isa, Tharmapalan, Jin, Phillips & Lambert 1998) provide metadata annotation and browsing to metadata information such as Cast, Location, or Events. The VMGS system has been extended (Xia, (Xia & Jin 1999) ) to allow creation of video metadata types and output to **XML**(W3C 1999).

Because these systems were developed at different time with different purpose in mind, the resulting metadata has a wide variety of structures and contents, thus making information sharing and collating difficult. The **Vane system** is clearly the most flexible system as it does not restrict the format of the metadata at compile time, it allows user to load the format of the metadata via loading a **DTD** (Document Type Definition) file.

## 1.2 XML

**XML** (W3C 1999) (eXtensible Markup Language) is a new standard based on **SGML**. **XML** is an excellent framework for creating metadata because it standardizes the file structure format and the parsing and the storage process, yet retains the flexibility to have arbitrary structured content of metadata just like **SGML**. **XML** will also very likely to be the format of online data and next generation database systems in the near future, following the footsteps of **HTML**. Thus we will benefit from share multimedia information without having to define new file formats.

## 1.3 MPEG-7

**MPEG-7** (MPEG n.d.) is a new multimedia specification and its purpose is to provide and standardize **XML** schema for building video metadata information. To date, the **MPEG-7** standard is still an ongoing research topic and a working draft is expected to be released in October 2000. Full **MPEG-7** support is currently not feasible. **IBM** has released a **MPEG-7** Schema Visualization Tool in July 2000, showing a snapshot of the current **MPEG-7** schema.

## 2 Video Structure & Organization

Raw video is organized as a linear sequence of frames. It is cumbersome to index video information based on this model alone, because video information is usually delivered visually as a continuous sequence of frames accompanied by audio effects. Visual information is manifested through the spatial and the temporal domain.

- **Spatial Domain** expresses information within a single frame and can be derived from the presence, location, features of objects and even the setting of the background.
- **Temporal Domain** expresses information through a sequence of frames. This information can be derived from the activity of objects, depicted event and action of objects.

Majority of visual information is expressed through the temporal domain. A collection of continuous frame sequence form a **shot**, a collection of shots form a **scene**, and a collection of scenes form a **video**.

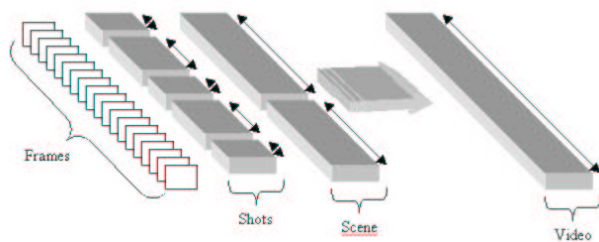


Figure 1: Previous Video Structure

While this model is simple and elegant, there are several constraints limiting its scope for expansion:

1. Hierarchy depth is fixed to four layers.
2. Overlapping is not allowed.
3. Sharing is not allowed.

These constraints restrict the video structure to a fixed depth tree hierarchy. An analogy with textual information may be as the following: frames = words, shots = paragraphs, scenes = chapters, video = book.

The first constraint is undesirable when the video has a more complicated semantic structure. This can be seen in a word processor environment where there are a lot of different style tags such as chapter, section to cover all needs. On the other hand, a totally unrestricted hierarchy depth may also lead to non-uniformities and thus more difficult to understand.

The second two constraints are undesirable and can be illustrated using the following example:

A video scene consists of three characters A, B, C and five shots, where A appears in shot 1 to 3. B appears in shot 2 to 5, and C appears in shot 4. In this model, user must annotate each shot separately to identify the presence of each character.

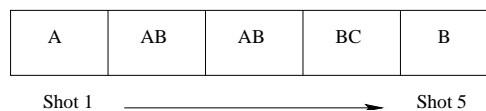


Figure 2: Non-Overlapping Scenes

A better approach is to allow different scenes to overlap and share shots as shown in the following diagram:

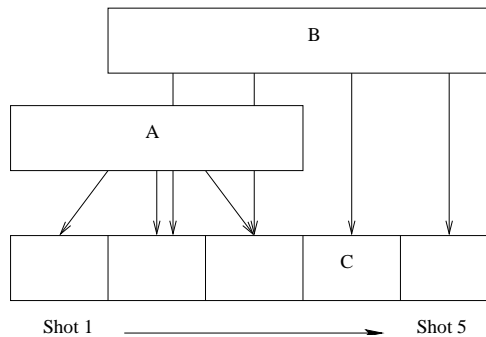


Figure 3: Multiple Overlapping Scenes

In this simple example, two additional overlapping scenes are created, and now the presence of A, B & C can be easily identified. Thus by exploiting the inter-relationship between shots and scenes, a more concise description of the video structure can be obtained that is easier to understand and to annotate.

## 2.1 The Video Hierarchy

To support the inter-relationship between shots/scenes, both shots and scenes will now also be referred to as segments and each node in the diagram below represent a video segment. This structure is essentially a directed acyclic graph (DAG), similar to the class inheritance model found in Object Oriented language that support multiple inheritance such as C++.

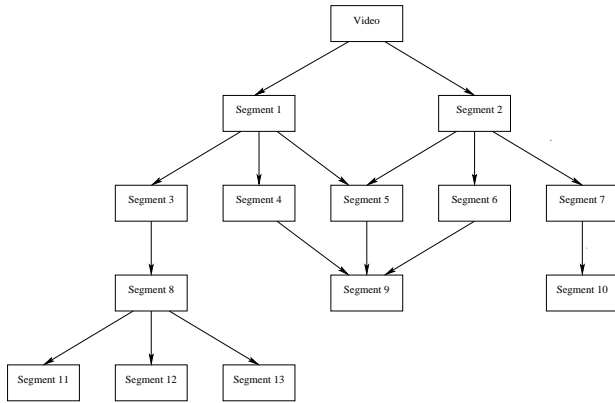


Figure 4: Modified Video Hierarchy Structure

In this diagram, the arrow represents the relationship “is a parent of”, and the arrow side always points to the child. A parent segment may have multiple child segments, each child segment may also belong to several parent segments, and the lowest level segments in the graph may correspond to shots identified by the shot-segmentation algorithm and these nodes will always have out-degree = 0. There will always be a node with in degree 0 and this node will conveniently represent the entire video, or some range of the video of our interest.

The relationship described here does need additional restrictions. This is because raw video is essentially a linear structure, and to say the segment A with range [100..200] is a parent of a segment B with range [150..200] makes sense, but to say segment A is parent of a segment C with range [200..300] does not make sense. In addition, to say segment A is a parent of segment A doesn’t make sense either, which is the acyclic property.

Therefore this relationship requires each video segment to have a range [start frame position..end frame position] to be a super range of all its children and no cycles:

$$Parent_{start} \leq Child_{start} \leq Child_{end} \leq Parent_{end}$$

We can also relax the disjoint constraint where now overlapping is possible :

Each segment will also be represented visually by a key-frame and the key-frame forms the static representation of the video segment and its index must be within the range of the corresponding segment.

$$Segment_{start} \leq Segment_{key} \leq Segment_{end}$$

## 3 Annotation

After having defined a video structure, we need to consider an appropriate annotation method. Annotation should allow meaningful information to be associated with any video segment in the structure. The

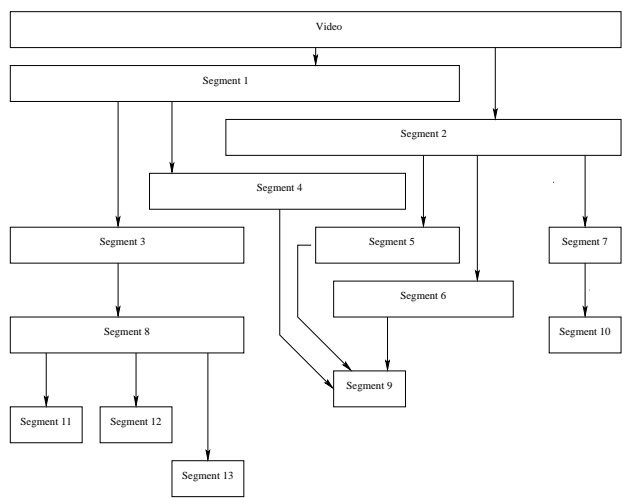


Figure 5: Modified Video Hierarchy (with temporal constraint)

basis of our work is taken from the **ViMetaVU system** (Isa et al. 1998), where shots are annotated with some objects or concepts. From now on, these objects/concepts will be referred to as **keys**.

In the existing model, every **key** is simply a collection of attributes, created under a particular category, and each **key** references all those video shots where the key appears. It is simple a  $m \times n$  relation:

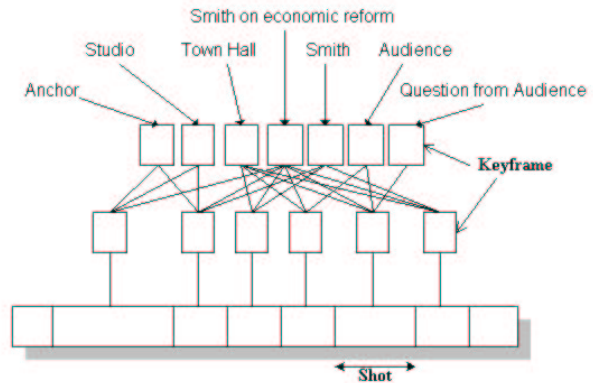


Figure 6: Previous Method of Video Shot-Key Association

## 3.1 Key Annotation

The model described above is no longer adequate for our DAG video structure for two reasons. Firstly, annotation shouldn’t be only limited at the shot level, rather we should allow to annotate any segment in the video structure. This is because it is possible that some shots/scenes/segment form a larger video segment, where certain annotation information only make sense when it is applied to the group as a whole.

This modification introduces a subtle inconvenience and can be illustrated using a simple example. Suppose we have segments A, B & C, and segment A is a parent of segment B and segment C. Segment B & C are associated with a key Z and Y respectively via the existing idea from above. Since the range of A completely covers the range of B and C, it is safe to claim via transitivity that A should also be automatically associated with the same key Z and Y. This is the desirable behaviour because searching for a segment containing both key Z and key Y should return segment A.

The implementation of the solution to this problem will be discussed in section 4.4.

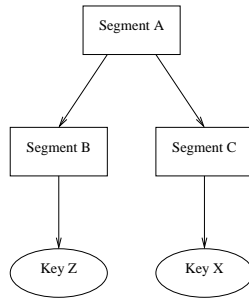


Figure 7: Transitivity of Association

The classification of keys in the existing model is also linear. It is possible to extend from this model to have the following definition for a key in the new model:

- A **Key** is a collection of attributes or a collection of other keys that when put together, identifies an unique concept or an entity that can found in the video, and may have semantic meaning or interest to the user. e.g A particular actor appearing in a movie is a **Key**.

In addition, we need the concept of a **KeyType**:

- A **KeyType** defines the formation or the structure of a **key**. e.g. the particular actor mentioned above may be of type **Cast**, and **Cast** may be defined to have description fields such as Name, Filmography, etc. **Cast** is a **KeyType**.

The new model of classification has considerable number of advantages over the existing linear model:

- Allows a more accurate concept representation.
- Allows related concepts to be grouped together.
- Allows several concepts with the same name and structure, but used under different context to be treated separately.

### 3.2 Textual Annotation

In addition to annotation using the concept of keys, information unique to video segments or individual video frame, that is not shared with other segments/frames should also be allowed. This may include a short description of the segment.

### 3.3 Static Image Segmentation & Annotation

While video segments usually contain far more meaningful information than a single video frame, single frame usually can be analyzed using another powerful technique - **Image Segmentation**. By combining neighbourhood regions in the segmentation result, objects within a single frame may be extracted using a contour descriptor and annotation should be applicable to the extracted object in the same way.

Figure 8 illustrates all types of association between video segments/frames and objects/keys that may be created in the system.

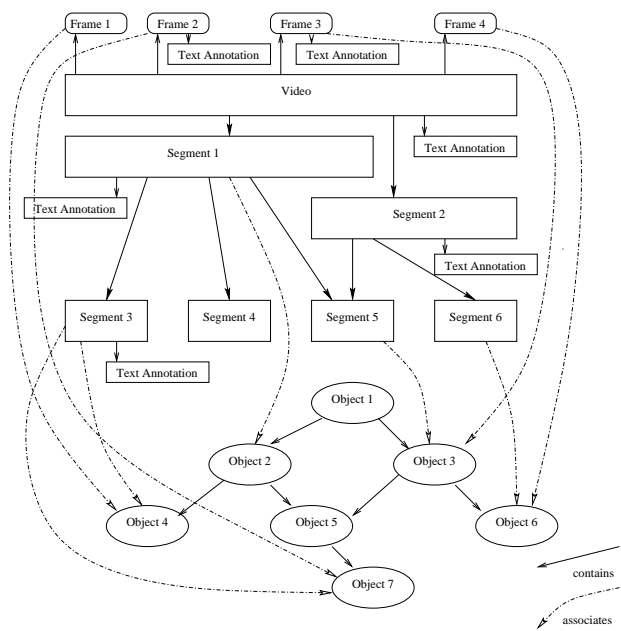


Figure 8: Modified Method of Video Segment/Frame-Key Association

## 4 XML Representation

### 4.1 Schema

The purpose of a schema is to define a class of XML documents. A XML file consists of a main element, and a number of sub-elements. These sub-elements in turn contain other sub-elements, until a sub-element such as **Name** contains a string rather than any sub-element. Element that contain sub-elements or carry attributes are said to have complex types, whereas elements that contain numbers (strings and dates, etc) but do not contain any sub-elements are said to have simple types. Some elements have attributes; attributes always have simple types. Schema document use the tag `<complexType>` and `<simpleType>` to describe complex and simple types respectively.

### 4.2 Video Structure

In the following schema example, an element with tag `<Video>` contains a number sub-elements `<VideoSegmenti>`, and one sub-element `<SegmentRelationshipGraph>`. The `<VideoSegment>` element contains attributes that specify the id, start, key and end positions of the segment, and two sub-elements, one for the name of the segment, the other for a short description of the segment.

```

<complexType name="Video">
  <element ref="MediaURL"
    minOccurs="1"
    maxOccurs="1"/>
  <element ref="MediaLength"
    minOccurs="1"
    maxOccurs="1"/>
  <choice
    minOccurs="0"
    maxOccurs="unbounded">
    <element
      ref="SegmentRelationshipGraph"
      minOccurs="1"
      maxOccurs="1"/>
  </choice>
  <element
    ref="EntityRelationshipGraph"
    minOccurs="1"
    maxOccurs="1"/>
</complexType>
  
```

```

    <element ref="VideoSegment"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </choice>
  <attribute name="id"
    type="ID"
    use="required"/>
</complexType>

<complexType name="VideoSegment">
  <element ref="Name"/>
  <element ref="TextAnnotation"
    minOccurs="0"/>
  <attribute name="id"
    type="ID"
    use="required"/>
  <attribute name="startIndex"
    type="mds:Frame"
    use="required"/>
  <attribute name="endIndex"
    type="mds:Frame"
    use="required"/>
  <attribute name="keyIndex"
    type="mds:Frame"
    use="required"/>
</complexType>

```

The video structure is specified by the element `<SegmentRelationshipGraph>` and it is slightly more complicated. It contains an element `<SegmentNode>` which correspond to the root segment of the video.

Each `<SegmentNode>` may contain sub-elements of the same type `<SegmentNodei`, or a reference, `<SegmentReferencei`. Only nodes of type `<SegmentNodei` may define child nodes, whereas nodes of type `<SegmentReferencei` may not.

```

<complexType name="SegmentRelationshipGraph">
  <element name="SegmentNode"
    type="mds:SegmentNode"/>
</complexType>

<complexType name="SegmentNode">
  <choice minOccurs="0"
    maxOccurs="unbounded">
    <element name="SegmentReference"
      type="mds:SegmentReference"/>
    <element name="SegmentNode"
      type="mds:SegmentNode"/>
  </choice>
  <attribute name="idref"
    type="IDREF"
    use="required"/>
</complexType>

<complexType name="SegmentReference">
  <attribute name="idref"
    type="IDREF"
    use="required"/>
</complexType>

```

The value of the `idref` attribute in `<SegmentNode>` and `<SegmentReference>` elements is used to reference the actual `<VideoSegment>` element, which has the same value in the `id` attribute.

Let us consider an instance XML output that describes a video structure:

```

<VideoSegment id="ID4"
  startIndex="0"
  endIndex="1000"
  keyIndex="0">
  <Name>Segment 4</Name>
  <TextAnnotation/>
</VideoSegment>

```

```

...
<VideoSegment id="ID62"
  startIndex="500"
  endIndex="700"
  keyIndex="500">
  <Name>Segment 62</Name>
  <TextAnnotation/>
</VideoSegment>
...
<VideoSegment id="ID76"
  startIndex="0"
  endIndex="700"
  keyIndex="500">
  <Name>Segment 76</Name>
  <TextAnnotation/>
</VideoSegment>

<VideoSegment id="ID77"
  startIndex="500"
  endIndex="1000"
  keyIndex="700">
  <Name>Segment 77</Name>
  <TextAnnotation/>
</VideoSegment>

<SegmentRelationshipGraph>
  <SegmentNode idref="ID4">
    <SegmentNode idref="ID76">
      <SegmentReference idref="ID62"/>
    </SegmentNode>
    <SegmentNode idref="ID77">
      <SegmentNode idref="ID62"/>
    </SegmentNode>
  </SegmentNode>
</SegmentRelationshipGraph>

```

The above output describes the following DAG graph:

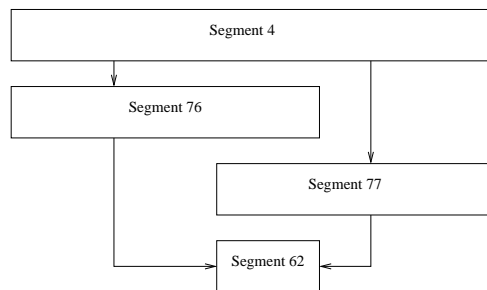


Figure 9: XML-Video Hierarchy Structure Example

### 4.3 Video Annotation

Unlike the schema for the video structure, there is no fixed schema for annotation, because it is always application specific. In the following schema example, `Key`, `Cast`, `Place` are all **KeyTypes**, where each object type must have an `id` attribute, a sub-element `Name`.

```

<complexType name="Key">
  <attribute name="id"
    type="ID"
    use="required"/>
  <element ref="Name"/>
  <element ref="Place"/>
  <element ref="Cast"/>
</complexType>

<complexType name="Place">
  <attribute name="id"

```

```

        type="ID"
        use="required"/>
    <element ref="Name"/>
    <element ref="TextAnnotation"/>
</complexType>

<complexType name="Cast">
    <attribute name="id"
        type="ID"
        use="required"/>
    <element ref="Name"/>
    <element ref="TextAnnotation"/>
</complexType>

```

Using the above schema example, a video annotation file may contain the following annotation key hierarchy:

```

<Key id="0">
  <Place id="ID1">
    <Name>Park</Name>
    <TextAnnotation/>
  </Place>
  <Place id="ID2">
    <Name>Mountain</Name>
    <TextAnnotation/>
  </Place>
  <Cast id="ID3">
    <Name>Sam</Name>
    <TextAnnotation/>
  </Cast>
  <Cast id="ID4">
    <Name>Mark</Name>
    <TextAnnotation/>
  </Cast>
</Key>

```

Video segments or video frames use the `idref` attribute to reference the object:

```

<EntityRelationshipGraph>
  <EntityNode idref="ID14">
    <EntityNode idref="ID1"/>
    <EntityNode idref="ID2"/>
    <EntityNode idref="ID3"/>
  </EntityNode>
  <EntityNode idref="ID15">
    <EntityNode idref="ID3"/>
    <EntityNode idref="ID4"/>
  </EntityNode>
</EntityRelationshipGraph>

```

This way of specifying the video structure is based on the current MPEG-7 draft. There has been several modifications and simplifications being made to accommodate our model. This is mainly because in the draft, range is specified using time instead of frame index. Although our implementation differ with **MPEG-7**, a lot of terminologies used above do from the current MPEG-7 proposal and the system has been designed to allow future transition to MPEG-7 easier.

#### 4.4 Browsing & Navigation

The whole purpose of building annotation data as described above is to facilitate search and browse operations, where the user can choose some criteria (e.g. match certain object or perform textual search) and then only browse through the video hierarchy that has been annotated with those matched objects.

Each criteria is in the form of (*Tag, Value*). A video segment matches a criteria if one of the following occurs:

1. A XML node belong to the video segment matches the *Tag*, and either the node content itself or a child node's content matches the *Value*.
2. A XML node belong to an associated key of the video segment matches the *Tag*, and either the node content itself or a child node's content matches the *Value*.
3. One of its children matches the criteria.

For multiple criteria, boolean operator **OR** and **AND** can be applied in the conventional way.

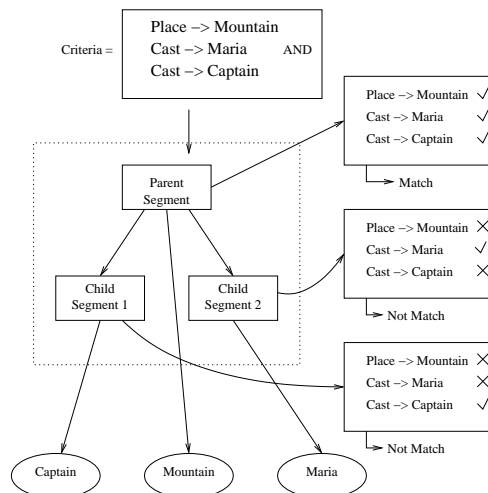


Figure 10: Search Example 1

Because annotated key may be associated with child segment, we need to traverse through the entire video hierarchy to determine which segment satisfy the given criteria. The traversal is an efficient search algorithm as the video hierarchy is searched exactly once and result is saved for later requests on the same criteria.

## 5 User Interface

### 5.1 Clip Tree View

The graph can be modeled as multi-layered graph with a single root, namely the entire video segment of interest. This can be viewed as a tree, where it is possible that multiple tree nodes reference the same video segment node.

This view shows the structure of all the annotated video as trees, highlights the parent-child relationships between video segments, and user can search or play any of the video segments visible in this view. Each segment's name may be modified directly through this view. A screen shot example is illustrated in Figure 11.

### 5.2 Time Line Editing View

Because the underlying video structure is a DAG, a single video segment may appear as multiple Time-LineClip objects via different paths traversed from the root segment. Adding or deleting a video segment actually correspond to adding or deleting one of its connections to a parent.

There are several ways of adding a video segment to the structure. User may choose a start and end range from the Media Player, and click "Add Clip", then a video segment with the specified start and end



Figure 11: Clip Tree Interface

position will be inserted as a child segment of the current root segment if allowed by the range constraint.

Alternatively, selecting on union, intersection options will insert the union, intersection of all the currently selected segments in this video. Traditional operations such as copy/delete may also be used to change the video structure. A screen shot example is illustrated in Figure 16.

### 5.3 Key Hierarchy View

For each video segment, the annotator may wish to annotate it with some information. This view primarily shows the different types of possible annotation keys that we may wish apply to video segments. Elements labeled with a blue C represents a complex KeyType object, in terms of XML schema, it is a `<complexType>` node. A red S represents a simple KeyType object, and it is a `<simpleType>` in XML schema. A red a represents an attribute. A screen shot example is illustrated in Figure 12.

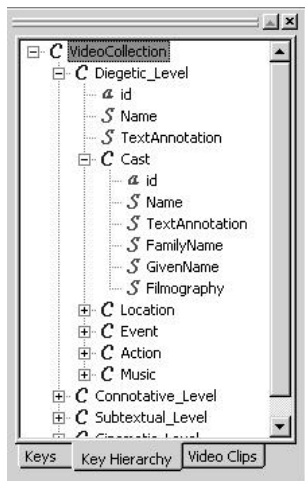


Figure 12: Key Hierarchy Interface

### 5.4 Element Annotator

The Element Annotator allows video segment/frame to be associated with keys. There are three types of controls in this dialog as shown in Figure 13. The left drop down box specifies the KeyType, the middle drop down box lists the available keys of the selected KeyType, and the right hand button brings up the Property Annotator (discussed in the next section) for modifying properties of the currently selected key.

This view can be opened while the user interacts with other parts of the system. When the user selects a video segment/frame, existing associations will be shown in this view. Although the number of keys available in this dialog is currently fixed to 6 items, multiple dialog pages will be generated if all the existing pages are full.

### 5.5 Condition Editor

The system allow user to search for video segments that match the specified criteria. For example, user may wish to search for all the video segments that have a particular Cast being present in a particular Location. The output will be sent to a new Navigation View (discussed in the next section) for browsing.

### 5.6 Navigation View

The **Navigation View** allows sub-graph of video segments to be browsed, or those that match a specified criteria discussed in the previous section. This view consists of two panels, The left hand view is the tree view, and the right hand view display those video segments under the current selection in the left panel if it is expanded. This view synchronizes selection made with the annotation view.

## 6 Results & Discussion

We use the system to create a large set of metadata information based on the movie "**The Sound of Music**". We illustrate how a story line can be structured in an organized fashion, and how key objects are annotated.

The context menu also provide options to copy and paste selected video segments to other parts of the video structure. We used this option to place the video segment "Do Re Mi" under "Music Clips" and "Being a Governess". Note this segment is shared, and any subsequent additions of child segment to "Do Re Mi" will appear synchronized in two places of the time line editing view.

Video play back is extremely efficient as well. Any segment may be selected to play via the right mouse context menu, even if the media player is currently loaded with another video. Improvements is very clear comparing to previous video decoders found in systems such as **VMGS/ViMetaVU**, or the **Corona System**.

We can also perform content annotation by associating each segment with meaningful data. The Figure 13 show how the video segment "Do Re Mi" may be annotated.

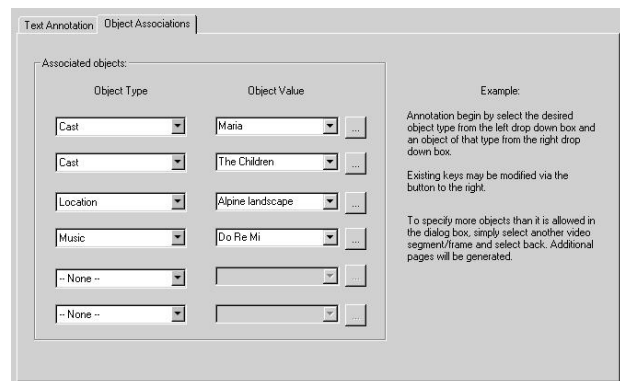


Figure 13: Annotation Interface

After adding relevant annotation information to other video segments, queries can be performed via

the search page. Searching for segments that match with the criteria  $Cast=Maria \ \&\& \ Cast=Captain \ \&\& \ Location=Trap \ Villa$  produces a tree result shown in Figure 14. Any node visible in this tree simply means either it satisfies the criteria, or at least one of its children satisfy the criteria. Take away the third condition, we see more matching results in Figure 15.

Experiments with various criteria search show results very quickly. Although a qualitative time analysis is not performed on the searching process, the algorithm that performs the search traversal is  $O(n)$  where  $n$  is the total number of XML nodes in the file.



Figure 14: Segments match  $Cast=Maria \ \&\& \ Cast=Captain \ \&\& \ Location=Trap \ Villa$



Figure 15: Segments match  $Cast=Maria \ \&\& \ Cast=Captain$

We present a screen shot of our entire system in Figure 16.

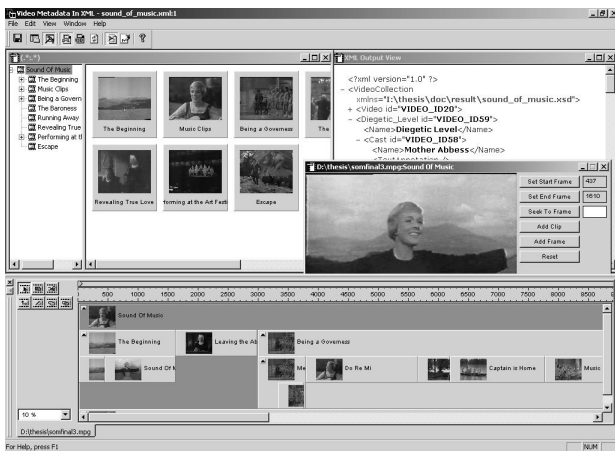


Figure 16: System User Interface

## 7 Conclusion

The system is intended to allow organization and annotation of video segments on a collection of videos so they may be searched using semantic content. The generated metadata is in XML.

The video structure is organized based on a Directed Acyclic Graph model. In this model, the entire video is decomposed to several smaller video segments, and each smaller segment can be further decomposed into sub segments. Video segment may be shared and overlapped. This provides much more

flexibility to the annotator over the existing shot-based, non-overlapped, linear or hierarchical video structure.

The structure of allowable annotation keys can be specified via a separate XML schema file and it is also organized base on a (DAG) model. We allow KeyTypes to be grouped, and allow annotation keys to be composed of other annotation keys. This results a better controlled and more organized annotation output.

A suitable and attractive user interface is designed and implemented for our system. The interface makes video structuring and annotation process extremely easy. In addition, efficient video playback, XML output viewing, browsing and searching operations have been included in the system to aid the annotation process.

The development process and preliminary results have inspired some interesting uses of the system. The system may be used as a post-processing tool for video segmentation result. Remote Learning may be setup this way to allow student to search as well browse materials provided by the lecturer. It may be used for individuals to collate, exchange thoughts and ideas about videos, or used in a movie/entertainment industry for content archival and retrieval purposes.

## References

- Carrer, M., Ligresti, L., Ahanger, G. & Little, T. (1996), 'An annotation engine for supporting video database population', *MCL Technical Report No. 08-15-1996*.
- Isa, W. Y. H. B., Tharmapalan, J., Jin, J. S., Phillips, C. J. E. & Lambert, T. (1998), 'Video cataloging in video archive and information retrieval', *Proceedings of the Workshop on Visual Information Processing* pp. 19–26.
- MPEG (n.d.), 'Mpeg home page', <http://drogo.cse.it/mpeg/>.
- W3C (1999), 'Extensible Markup Language (XML) 1.0', <http://www.w3.org/XML/>.
- Xia, J. & Jin, J. S. (1999), 'Multi-level video metadata generation and management using XML', *Proceedings of the Workshop on Visual Information Processing* pp. 77–83.
- Yap, S., Jin, J. S. & Feng, D. D. (1999), 'Automatic segmentation and semi-automatic organization in compressed video sequence', *Proceedings of the Workshop on Visual Information Processing* pp. 59–68.