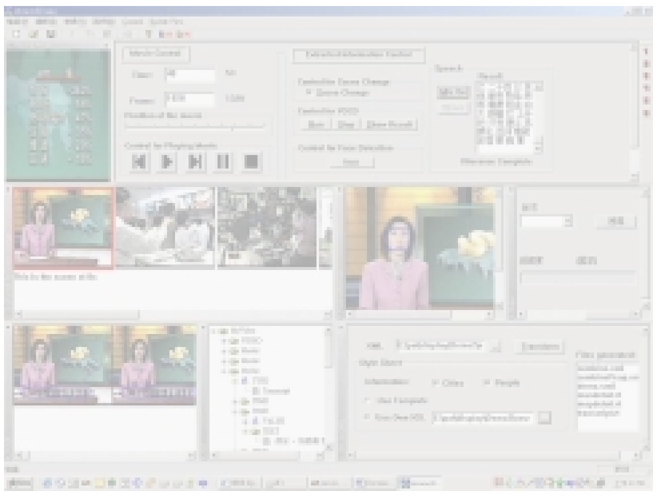
Department of Computer Science & Engineering

The Chinese University of Hong Kong

2001-2002 Final Year Project

LYU0102

XML for Interoperable Digital Video Library



Supervised by Prof. LYU, Rung Tsong Michael

Group Members: Chan Pik Wah
Ngai Cheuk Han

Prepared by
Chan Pik Wah 99581663

Abstraction

This report is subject to the final year project LYU0102 which title is XML for Interoperable Digital Video Library.

This report is divided into 10 parts, which includes: introduction of this final year project, architecture of XVIP (the system we built), current digital video libraries, extraction techniques that we implemented (scene change, VOCD, speech recognition and face detection) in the project, XML-based video information storage, the knowledge enrichment on primary video data, XML transformation and multimedia presentation, our implementation on XVIP, problems encountered and our solutions. Finally, it is our working progress and conclusion.

Our final year project describes an XML-based video information processing system, which extracts information from video and stores the information in a multimedia digital video library. We have named our system as XVIP, short for XML-based video information processing system. XVIP encapsulates a number of extraction techniques, including scene change detection, video optical character detection, speech recognition, and face detection. Apart from the primary extraction techniques, it provides geographical and biographical information by doing knowledge enrichment. It also provides a seamless approach to scale up the contents created in and delivered by the target multimedia digital video library. Furthermore, XVIP is based on a multi-modal concept, which treats each content extraction component as a mode, and users can easily add new modes into XVIP. The information extracted from the video is then stored in a flexible, scalable and reusable way based on a generic XML structure, providing a convenient mechanism for data representation on web browsers. Also, the content in the XML file can be used to perform knowledge enrichment on top of the primary information extracted from the video. This enriched data representation helps users search for multimedia video content more efficiently. Moreover, the XML-enriched video data can then be presented in players or browsers using XSLT or emerging presentation format like SMIL at user’s discretion. In XVIP, we provide an XML to SMIL transformer. It is able to generate different presentation templates of SMIL based on the XML file generated from the video. The SMIL presentation includes the video, information extracted and additional information for enriching the video content.

Table of Content

Introduction

1.1 Motivations

There has been a rapid increase in the usage of multimedia information in recent years. Of all media types (text, image, graphic, audio and video), video is the most challenging one, as it combines all the other media information into a single data stream and it becomes the most popular source of multimedia information.

As videos represent rich media in multimedia systems, many digital video libraries are developed and a lot of attentions are paid on how to present multimedia information. However, a little study has been performed on how to extract information from video and to store this information with flexibility for indexing and searching purpose. Our final year project will focus on the extraction of the pertinent, multi-modal information from videos, the integration of the extracted information into a flexible, XML-based, the knowledge enrichment on top of the primary information extracted from video, and the transformation and presentation of the XML documents in other structure, e.g. SMIL.

Not only the accuracy of these techniques are keys to the success of a digital library, but also the increasing number of different techniques affect the design of an “open” digital video library system to a great extend. How to scale the digital video library in terms of adding new extraction component also presents challenges. Whenever a new extraction method is developed, it imposes a series of new indexing and presentation functions to be included. Therefore, a generic framework for presentation and visualization of video information is curial to the deployment of the digital library. We attempt to implement a system that can meet this challenge.

1.2 Project Objective

Our project aims at illustrating how different information in videos can be extracted and edited, how information can be stored into an XML format, how secondary information can be included and used, and how useful the XML documents can be presented in other structure, e.g. SMIL. Consequently, we designed and implemented an XML-based video information processing system to address these issues.

XVIP achieves the following targets.

- To provide an open architecture that can ease the overhead of integrating different video processing, searching, indexing and presentation of various digital video library functions.
- To increase the reusability of the information extracted form the videos, includes information interchange between distributed video libraries and different publishing media.
- Video information is extracted once and delivered and presented multiple times to different computing platforms.

The above objectives are achieved via the following methods encapsulated in XVIP,

- Modal concept of the digital video library functions throughout the video information life-cycle.
- Collaboration of the video information processing modules.
- Generic framework for presentation and visualization of video information.

XVIP is complete system that can do video information processing, from information extraction and enrichment, XML-based data storage, to presentation. It is a multimodal system that contains different modules such as scene change detection, video optical character detection, speech recognition, geographic information, biological information etc. XVIP collaborates different modules in a flexible and scalable way. Moreover, it provides a generic framework for presentation and visualization of video information. XVIP demonstrated that XML-based data could be transformed to different format for presentation, such as SMIL.

2. Architecture of XVIP

2.1 Introduction

We describe XVIP, an XML-based video information processing system, which extracts information from video and stores the information in a multimedia digital video library. XVIP encapsulates a number of extraction techniques, including scene change detection, video optical character detection and recognition, face detection, speech recognition, and geometric coding. It also provides a seamless approach to scale up the contents created in and delivered by the target multimedia digital video library. Furthermore, XVIP can handle multilingual contents. XVIP is based on a multi-modal concept, which treats each content extraction component as a mode, and users can easily add new modes into XVIP. The information extracted from the video is then stored in a flexible, scalable and reusable way based on a generic XML structure, providing a convenient mechanism for data representation on web browsers. Also, the content in the XML file can be used to perform knowledge enrichment on top of the primary information extracted from the video. This enriched data representation helps users search for multimedia video content more efficiently. Then, the XML is transformed to SMIL, which is Synchronization Multimedia Integrated Language, for presentation purpose. SMIL can synchronize the playback of all multimedia elements.

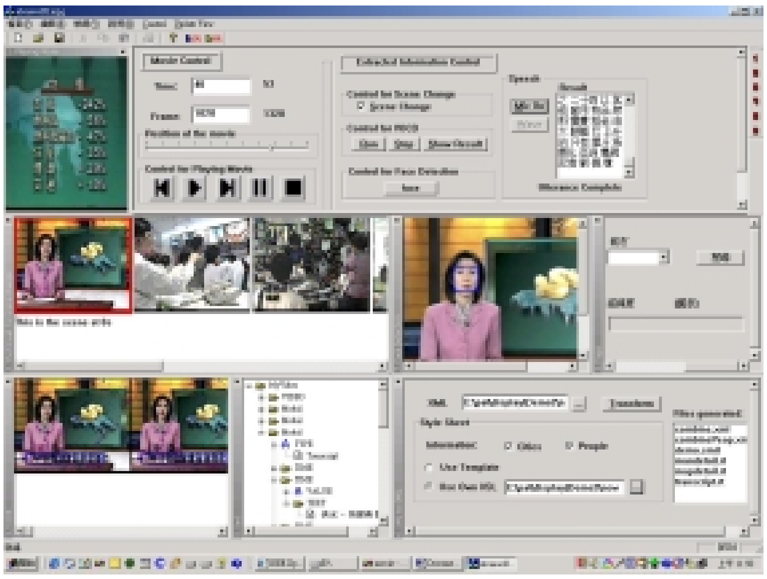


Figure 2.1 Interface of XVIP

2.2 XVIP Overview

XVIP can be divided into four sections, extraction techniques, storage, knowledge enrichment and presentation. The overview of XVIP is shown in Figure 2.2. Video is the input of the system, then, XVIP will apply three extraction techniques (shot break, VOCR, face detection) to the video channel and speech recognition to audio channel. The information extracted (scene change, text, name and transcript) is integrated as different modality in XML. Then knowledge enrichment (geocode and names of people) is done base on the extracted information. Lastly, the XML is transformed to SMIL with XSL for presentation the extracted information.

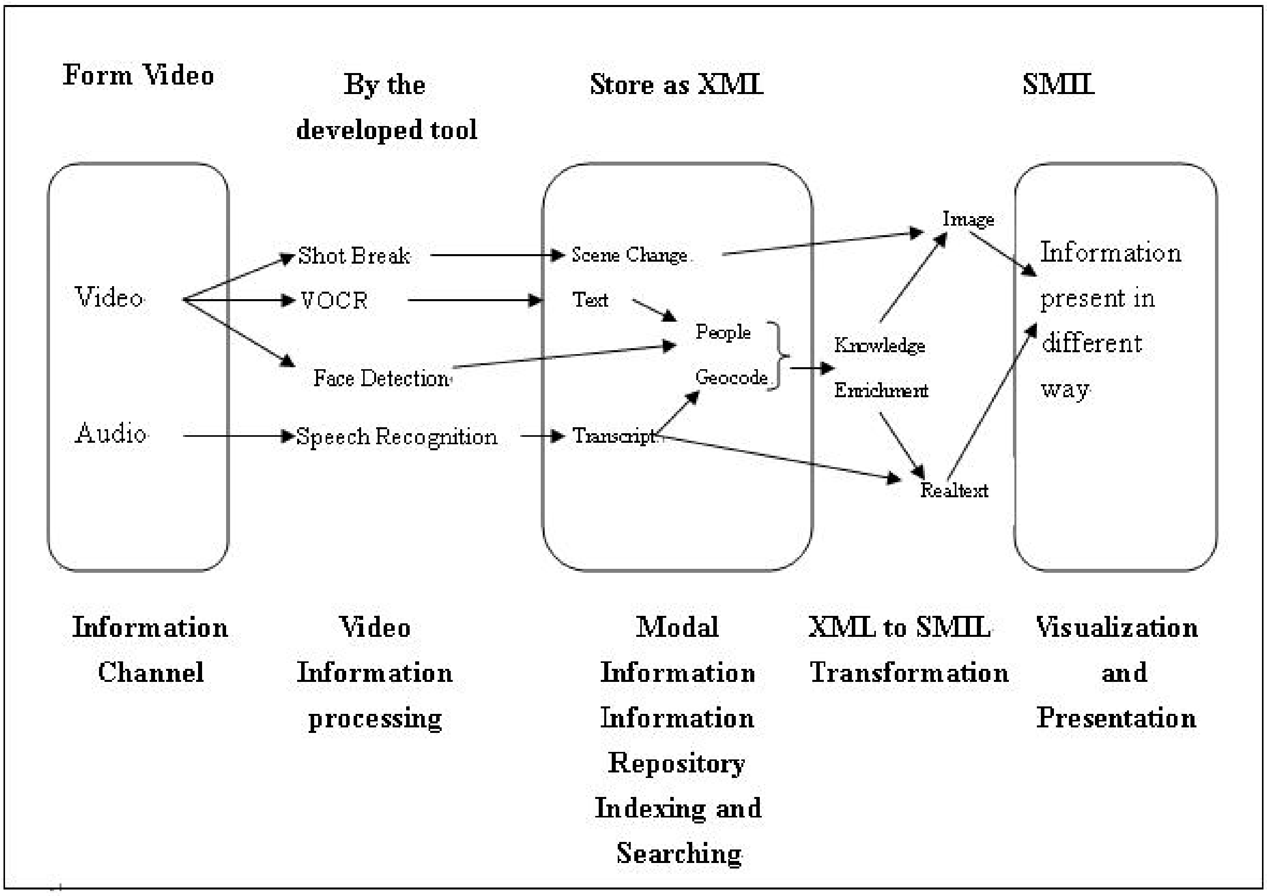


Figure 2.2 overview of XVIP

2.2.1 Extraction techniques

There are many information can be extracted from the video sequence: generating multimedia abstractions, segmenting video into stories, text clustering and topic classification, speech recognitions, face detection, key frame selection, skim, and video OCR.

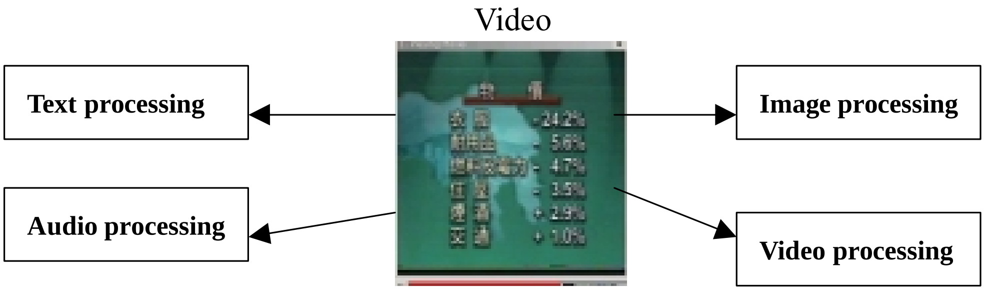
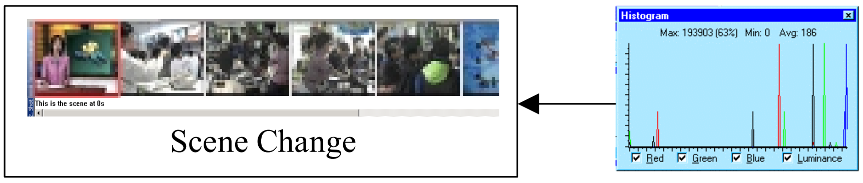


Figure 2.3 Data can be extracted from video

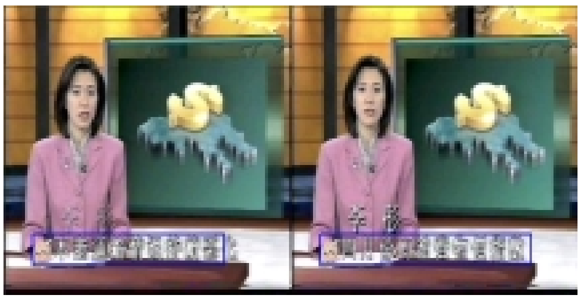
Digital Video Library combines all these techniques, shown in Figure 2.3. It is a new approach for automated video and audio indexing, navigation, visualization, searching and retrieval and embedded them in a system for use in education, information and entertainment environments.

In XVIP, four extraction techniques are applied, including shot break, VOCR, face detection and Speech Recognition. The corresponding information are extracted, including scene change, text, name of people and transcript.

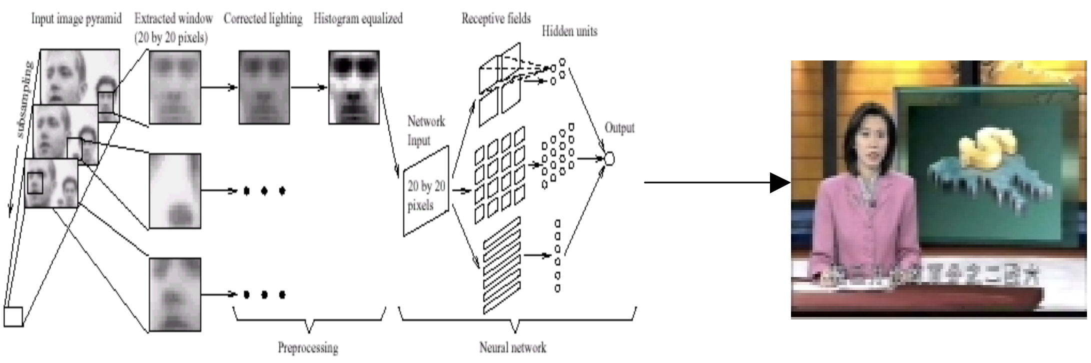
1. Shot break use Histogram Difference Method to find the scene change



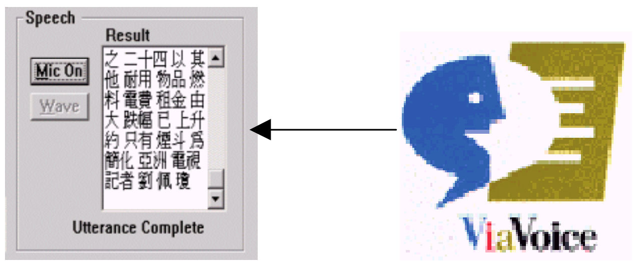
2. Four main step is applied to VOCR include: Detection of Text Region, Image Enhancement, Character Recognition and Segmentation, Post-Processing.



3. Neural network based face detection is use in XVIP



4. A commercial product Via Voice developed by IBM is used for Speech recognition



之 二十四 以 其
他 耐用 物品 燃
料 電費 租金 由
大 跌幅 已 上升
約 只有 煙斗 爲
簡化 亞洲 電視
記者 劉 佩 瓊

Utterance Complete

2.2.2 Storage

Storage of information extracted from video in a flexible, scalable and reusable way becomes very important. We found that XML satisfy all the above requirements, and it is a good choice.

XML stands for EXtensible Markup Language. XML is a markup language much like HTML. XML was designed to describe data. XML is receiving a great deal of attention from the computing and Internet communication since it was developed by the XML working group (known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. XML is a public format and not a proprietary format of any company. The v 1.0 specification was accepted by the W3C as Recommendation on February 10, 1998.

An XML file contains not only data but also metadata – structural and semantic information about that data. In this sense, an XML document is very similar to a database, and it in general will look like one big database. XML is platform and system independent and universal format. So, user can exchange data and communicate through network with XML. It is powerful as everyone could speak the same language and communicate together. XML is flexible, scalable and can let us create any tag to describe the information. So, the information extracted for the video by the tool is stored as XML and an XML editor is implemented for the user to edit the XML without changing the XML format.

XML is a meta-language that permits a set of users to create its own mark-up language for describing the contents of Web documents. An XML file contains not only data but also metadata – structural and semantic information about that data. In this sense, an XML document is very similar to a database, and it in general will look like one big database. In resent year, many existing database system try to adopt XML for exchanging data, such as IBM DB2, Oracle 8i and Microsoft SQL Server.

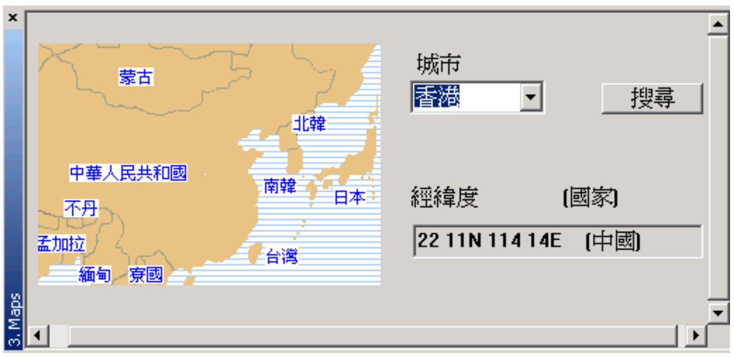
XML is platform and system independent and universal format. So, user can exchange data and communicate through network with XML. It is powerful as everyone could speak the same language and communicate together.

2.2.3 Knowledge Enrichment

Video information processing is the content creation step of the digital video library. The collaboration of different video information extraction techniques is on the information exchange level, mainly

- Knowledge Cross-referencing
- Knowledge Enrichment

For some video processing techniques, the accuracy of the recognition process can be increased by cross-referencing information generated by other modalities. For example, to identify a human face in the video, face recognition technique can be the primary modality information extracted. If available, the on screen title of the person's name can be recognized and served as the cross-reference knowledge for the person identification. Example of knowledge enrichment is the geographical naming process. The geographical naming database is a kind of knowledge repository of geographical names of countries and cities. By applying this knowledge to the text recognized from the speech recognition, the knowledge encapsulated in the text is enriched. In XVIP, geographical names and name of people is chosen for enrichment. A sample result is shown in Figure 2.3.



城市 香港

經緯度 (國家)

22 11N 114 14E (中國)

Figure 2.3 The knowledge enrichment component in XVIP.

2.2.4 Presentation

In our project, SMIL is selected to be the format for video information presentation. SMIL stands for Synchronized Multimedia Integration Language is currently a W3C Recommendation. It is a markup language that can synchronize and integrates multimedia. It enables authors to specify when and what should be presented, enabling them to control the precise time that a sentence is spoken and make it coincide with the display of a given image appearing on the screen. In this example, our SMIL file integrates the video, images from scene change detection and the text obtained from speech recognition.

The basic idea of SMIL is to name media components for text, images, audio and video with URLs and to schedule their presentation either in parallel or in sequence. The SMIL presentation is composed from several components that are accessible via URLs. The components have different media types, such as audio, video, image or text. The "begin" and "end" times of different components are specified relative to events in other media components. For example, in a slide show, a particular slide is displayed when the narrator in the audio starts talking about it.

Presenting multimedia with SMIL has a lot of advantages. SMIL is text-based. This makes it easy for any designer or developer to work with. A text editor can already let them start without any investment, though powerful SMIL authoring tools are also available. They can generate SMIL automatically for each visitor. As SMIL is text-based, it is allowed to create different presentation parts, assembling a customized SMIL file for each visitor based on preferences recorded in the visitor's browser. Because SMIL can stream many media formats, there is no need to merge clips into a single streaming file. This makes it easy to alter the presentation. Moreover, SMIL effort is led by W3C. W3C tries to shape a specification that is beneficial to all parties involved. Unlike a proprietary technology owned by just one vendor, SMIL can lean toward general industry and consumer interests.

After the information is extracted and stored as XML schema designed, we move on to the transformation and presentation of the XML-formatted video data. The work-flow of our project is shown in Figure 2.4.

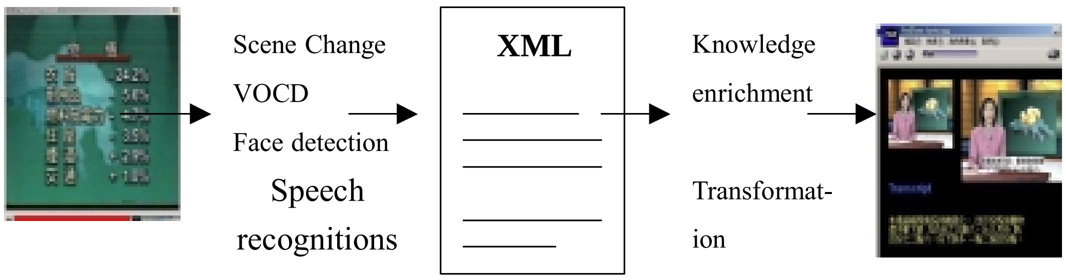
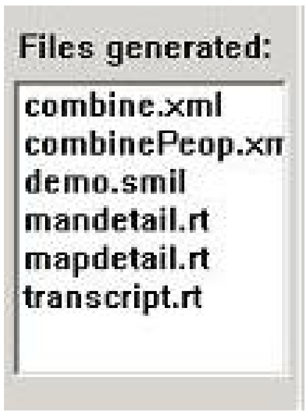


Figure 2.4 Work flow from video information extraction to presentation in XVIP.

- We built a XML to SMIL transformer. It gets the XML video data file as input and generated files for the SMIL presentation. The transformer also allows user to select templates and the additional information that they preferred.



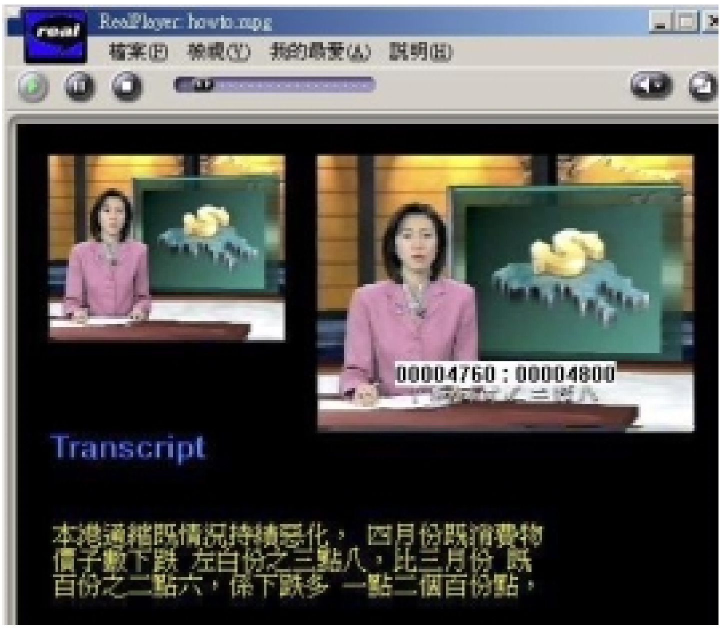
Files generated:

combine.xml
combinePeop.xm
demo.smil
mandetail.rt
mapdetail.rt
transcript.rt

Figure 2.5 File list generated in XVIP for playing back the SMIL or Realtext files.

- By clicking the file name of SMIL on the file list generated in Figure 2.5, the SMIL presentation will start. Two samples of presentation are shown in Figure 2.6. In the left figure below, it shows the SMIL presentation with geographic information added. In the right figure below, it shows the SMIL presentation with biographic information added.

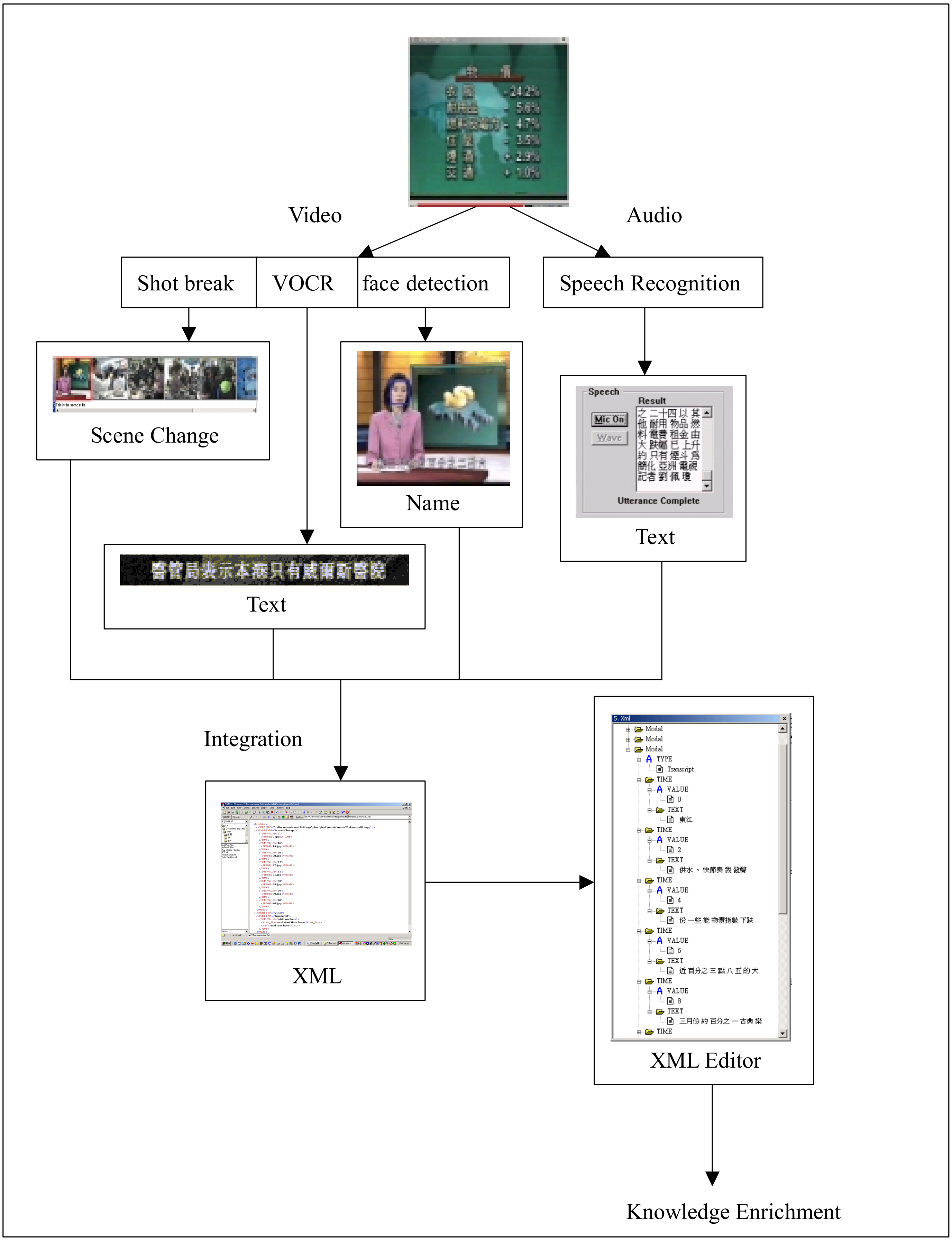
Apart from the sample shown in Figure 2.6, other templates of SMIL presentation can also be generated using our XML to SMIL transformer. For example, we can produce a SMIL presentation that can display bother geographical and biographical information. Moreover, if more additional information or different templates are preferred, our system is flexible and extensible enough to produce them too.



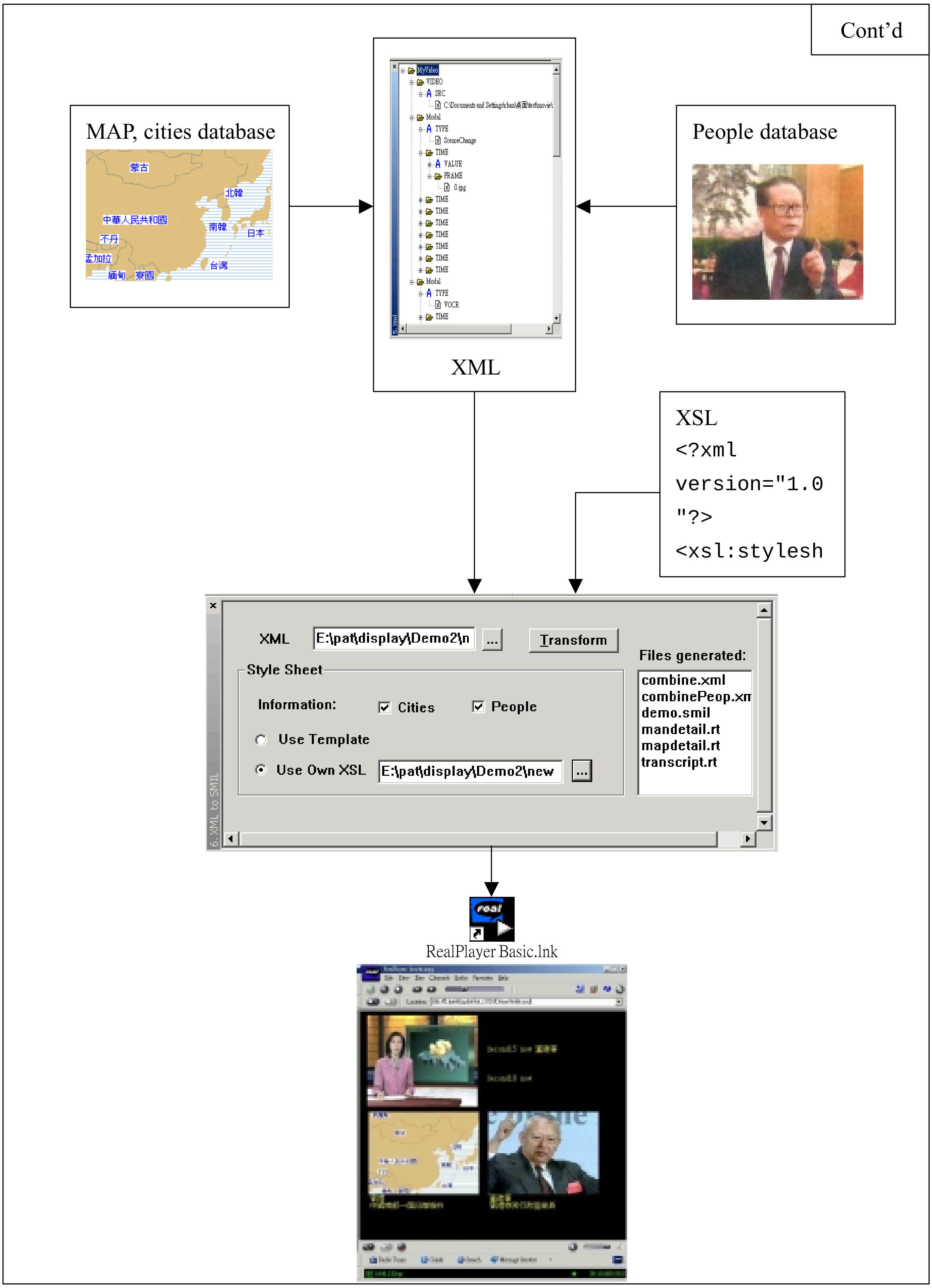
本港通縮既情況持續惡化， 四月份既消費物
價指數下跌 左百份之三點八，比三月份 既
百份之二點六，係下跌多 一點二個百份點，

Figure 2.6 SMIL presentations generated from our transformer..

2.3 Structure of XVIP



Cont'd



```
MyVideo
  VIDEO
    SRC
      C:\Documents and Settings\chan\桌面\test\movie\
  Modal
    TYPE
      ScenceChange
    TIME
      VALUE
      FRAME
        0.jpg
    TIME
    TIME
    TIME
    TIME
    TIME
    TIME
    TIME
  Modal
    TYPE
      VOCR
    TIME
```

```
<?xml
version="1.0
"?>
<xsl:stylesh
```

XML: E:\pat\display\Demo2\n

Transform

Style Sheet

Information: [✓ Cities] [✓ People]

Use Template [✓ Use Own XSL] E:\pat\display\Demo2\new

Files generated:

```
combine.xml
combinePeop.xm
demo.smil
mandetail.rt
mapdetail.rt
transcript.rt
```

Video is the input of XVIP and it is processed in two ways, video and audio. Three extraction techniques are applied to the system, scene change (color histogram difference method), face detection (neural network-based method) and video optical character detection (dynamic library from CUM). For audio, speech recognition is applied to get the transcript. Commercial product IBM ViaVoice is used for speech recognition.

Then all extracted data is integrated into XML with DOM parser. The XML can be edited through the editor that XVIP provided.

With the cities and people database, knowledge can be enriching to the XML. Lastly, with suitable XSL, XML can be transform to SMIL with the transformer in XVIP for presentation.

2.4 Achievement in the first semester

2.4.1 Knowledge gained

In order to enrich our background, we have read papers in different area.

- Existing Video Digital Library

  We read information about current video digital libraries and extraction techniques being used. This enriched our background knowledge in the field and enabled us to implement XVIP.

- Video information

  There is a lot of useful information in videos. By reading papers, we know what information can be extracted from videos and how to classify them. The understanding of different types of video information helps the design of our XML scheme.

- Techniques of extracting scene change

  Different scene change detection techniques have been developed. By studying different techniques, we get familiar with different methods and algorithms can be applied. Then, we can improve our algorithm and get a better result.

- Techniques of VOCR

  In the same sense, several papers are read before implementing the VOCR. This increase our knowledge on the techniques applied in VOCR.

- Current XML issue and techniques

  XML is a format that newly introduced in these few years and it is receiving a great deal of attention from the computing and Internet communication. We study this new technology and try to apply it to XVIP. It is used now for storing the extracted video data in XVIP.

- Knowledge Enrichment

  By reading papers, we know that secondary information can be added to video based on the primary information extracted from videos. Then, we try to enrich video information by extracting names of major cities mentioned in videos and providing further information of the corresponding cities.

- DirectShow

  DirectShow provides high-quality capture and playback functions in multimedia streams. It is very useful for implementing the video player in XVIP.

2.4.2 Programming work

To achieve the goals mention in the previous section, the following work has been done:

- After studying DirectShow, we implemented a video play.
- We built a single document MFC program in Microsoft Visual C++.
- We implemented docking window by adding new class in our Visual C++ project.
- We implemented a dialog box for controlling the video player.
- We implemented scene change detection with dynamic threshold. A sample result is shown in Figure 2.7.

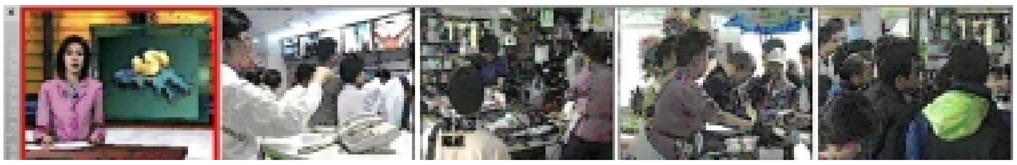


Figure 2.7 the result of the implemented scene change detection

- Video optical character detection is improved from the techniques from the existing Digital Video Library of CMU. A sample result is shown in Figure 2.8.

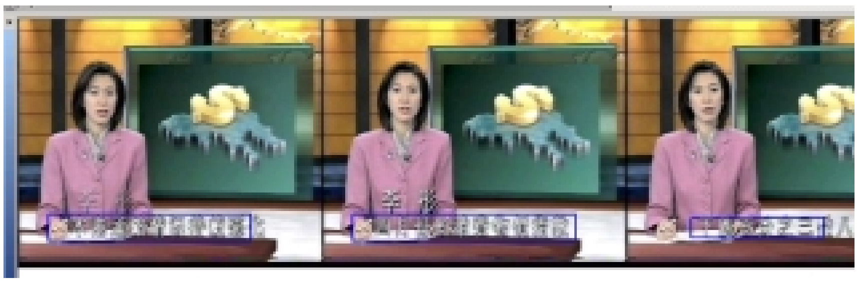
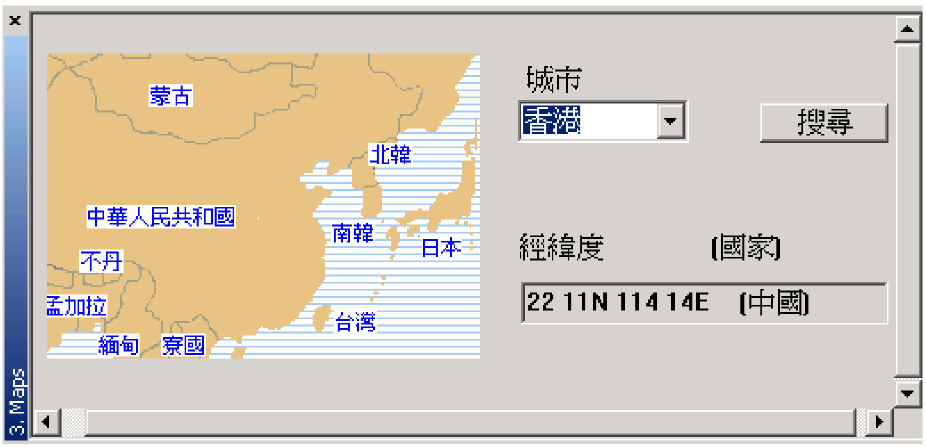


Figure 2.8 the result of the implemented VOCD

- After the information is extracted, it is stored in the format of Extensible Markup Language (XML). We designed a XML scheme for XVIP.

- XML parser is found for reading the XML files.
- Tree model is chosen for representing the XML in the tool. Users can view and edit the XML files created in XVIP.
- A secondary information extractor is implemented. It extracts names of major cities in the XML file and indicates it by adding a new tag in the file. Also, information of the extracted cities is provided with external data files. A sample result is shown in Figure 2.9.



城市
香港
經緯度 (國家)
22 11N 114 14E (中國)

Figure 2.9 View of the secondary information extractor

- Different modalities are included in XVIP. An “extractor & editor” is then developed. The integrated system is shown as Figure 2.10.

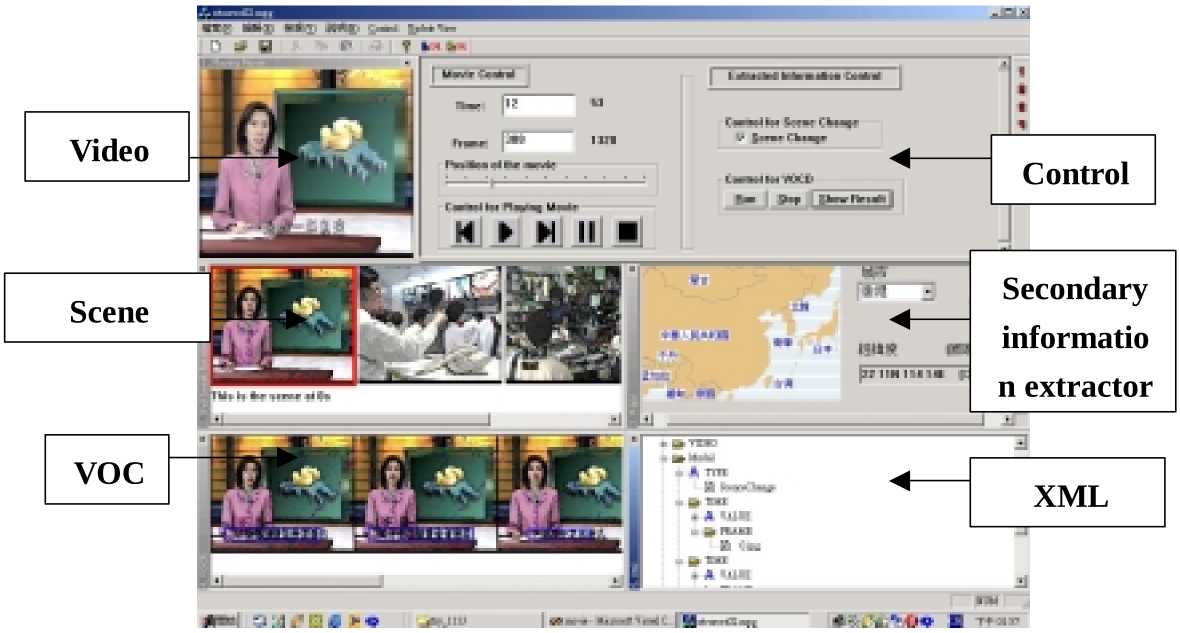


Figure 2.10 View of XVIP in first semester

2.5 Achievement in the second semester

2.5.1 Knowledge gained

- Video information

  We keep on studying what information, other than those implemented in first semester, can be extracted from videos and how to classify them. This knowledge helps us to extend the original design of our XML scheme.

- Techniques of speech recognition

  Different speech recognition techniques and software have been developed. By studying a number of them, we get familiar about different algorithms and the performance of different software. Then, we selected the speech recognizer that is suitable for XVIP.

- Techniques of face detection

  In the same sense, several papers are read before implementing face detection. This increase our knowledge on the techniques applied in face detection.

- SMIL

  By reading online tutorials, books and the documentations, we know that SMIL (Synchronized Multimedia Integration Language) is a powerful language for doing multimedia presentation and is suitable for XVIP. Then, we learn more deeply on its syntax and used it for presenting videos with the corresponding information extracted or added by XVIP.

- XSL

  By reading online tutorials, books and the documentations, we know that XSLT (Extensible Stylesheet Language Transformation) defines a common language for transforming one XML document into another. It can assist to transform XML documents into SMIL presentation. Then, we learn more deeply on its syntax, usage and limitations. This helps a lot in design our XML to SMIL transformer.

- XML to SMIL transformation

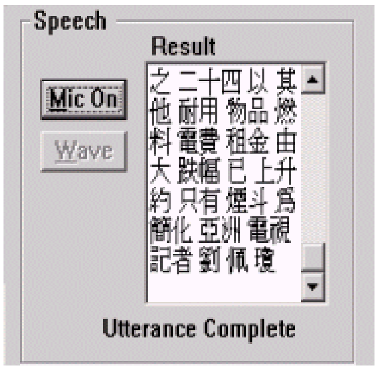
  After studying SMIL, we decided to use this format for presenting video information obtained. In the first semester, we have generated an XML file that contains video information. Therefore, we plan to build a transformer that can transform this file to SMIL presentation automatically. We spent a lot of time to investigate possible ways to build a good transformer before implementing our final design. In our final design, we make use of XSL in some part, but due to the limitation of XSL, we have to implement the remaining part of it with our code.

2.5.2 Programming work

In the second semester, we further improved XVIP by adding two more information extraction components into XVIP. They are speech recognition and face detection. Together with the scene change detection and video optical character detection, there are totally four extraction modules in XVIP.

Apart from video information extraction, we demonstrated how the extracted video information could be presented using SMIL. In order to make XVIP complete, we built a XML to SMIL transformer for converting our XML-based video data to SMIL presentation automatically. XVIP also provides additional geographic and biographic information to be displayed in the SMIL presentation.

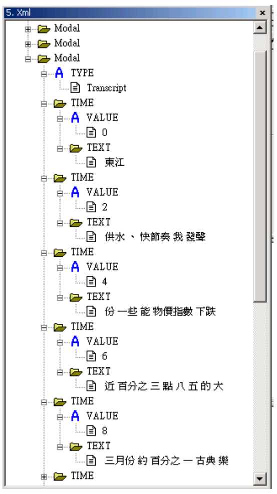
- We implemented speech recognition with IBM ViaVoice and stored the transcript obtained into XML file. A sample result and the produced XML file are shown in Figure 2.11.



Result

之 二十四 以 其 他 耐用 物品 燃 料 電費 租金 由 大 跌幅 已 上升 約 只有 煙斗 爲 簡化 亞洲 電視 記者 劉 佩 瓊

Utterance Complete



```
Modal
Modal
Modal
  TYPE
    Transcript
  TIME
    VALUE
      0
    TEXT
      東江
  TIME
    VALUE
      2
    TEXT
      供水 、 快節奏 我 發聲
  TIME
    VALUE
      4
    TEXT
      份 一些 能 物價指數 下跌
  TIME
    VALUE
      6
    TEXT
      近 百分之 三 點 八 五 的 大
  TIME
    VALUE
      8
    TEXT
      三月份 約 百分之 一 古典 樂
  TIME
```

Figure 2.11 Speech recognition results in XVIP.

- We implemented face detection by calling an external process. The jpeg source comes from the result of scene change detection. Whenever there is face appear in the scene, it will be highlight by a green rectangle. A sample result is shown in Figure 2.12.

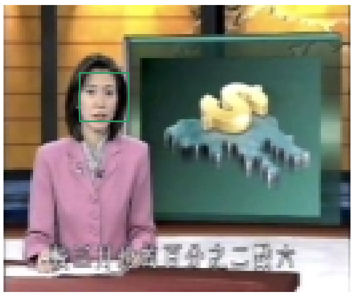
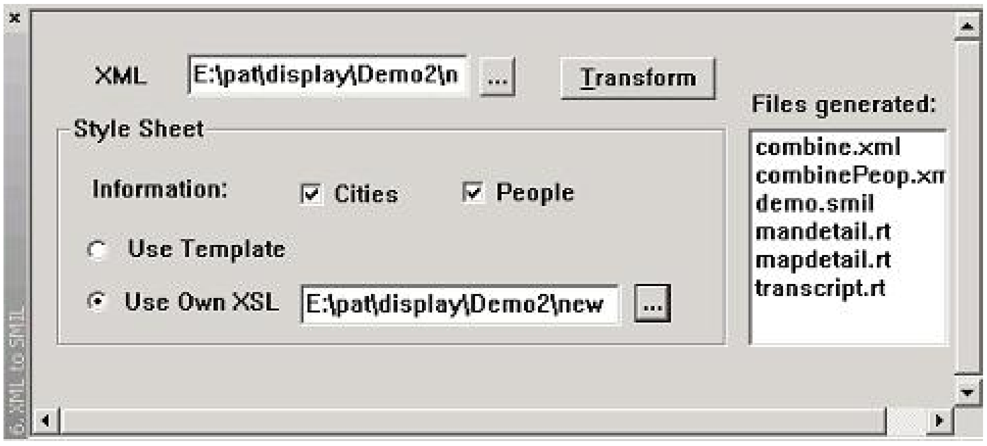


Figure 2.12 Face detection result in XVIP.

After the information is extracted and stored as XML schema designed, we move on to the transformation and presentation of the XML-formatted video data.

- We built a XML to SMIL transformer. It gets the XML video data file as input and generated files for the SMIL presentation. The transformer also allows user to select templates and the additional information that they preferred. The user interface is shown in Figure 2.13.



XML E:\pat\display\Demo2\n ... Transform

Style Sheet

Information: [✓ Cities] [✓ People]

Use Template [✓ Use Own XSL] E:\pat\display\Demo2\new ...

Files generated:
combine.xml
combinePeop.xn
demo.smil
mandetail.rt
mapdetail.rt
transcript.rt

Figure 2.13 User interface of the XML to SMIL transformer in XVIP.

- By clicking SMIL file’s name in the file list generated, the SMIL presentation will start. Two samples of presentation are shown in Figure 2.14.

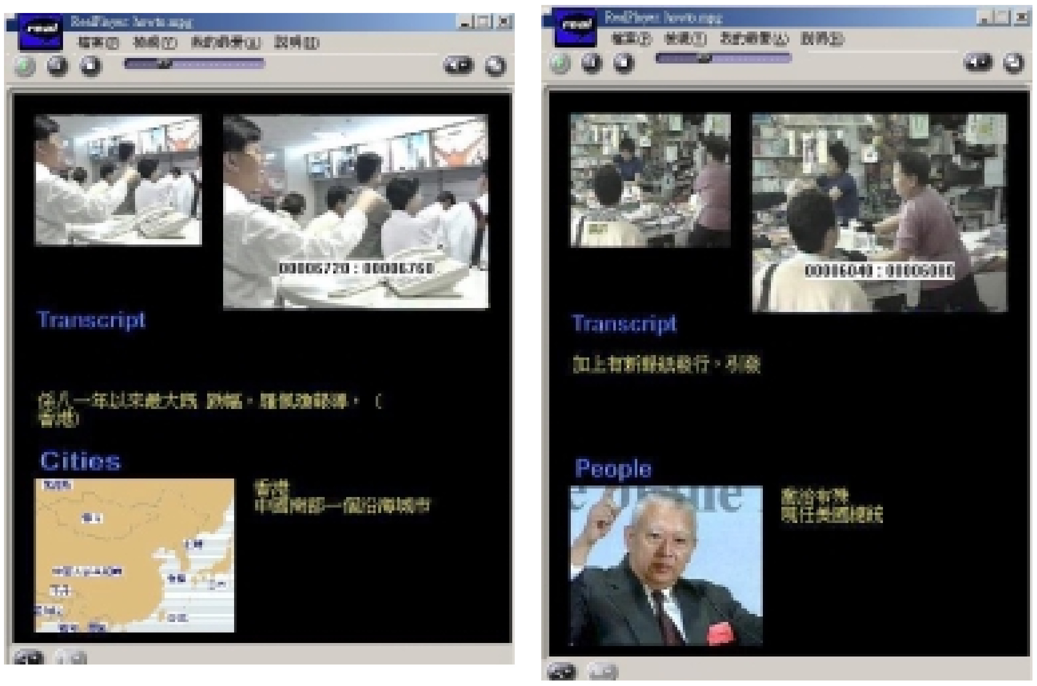


Figure 2.14 SMIL presentations generated from our transformer

- Different modalities are included in XVIP. This includes the video player, the four extraction modules, XML editor, knowledge enrichment component, and the XML to SMIL transformer. All of them are integrated into XVIP using the docking window class in GUI. XVIP can do video information extraction, XML-formatted storage, knowledge enrichment and SMIL transformation and presentation. The integrated system is shown in Figure 2.15.

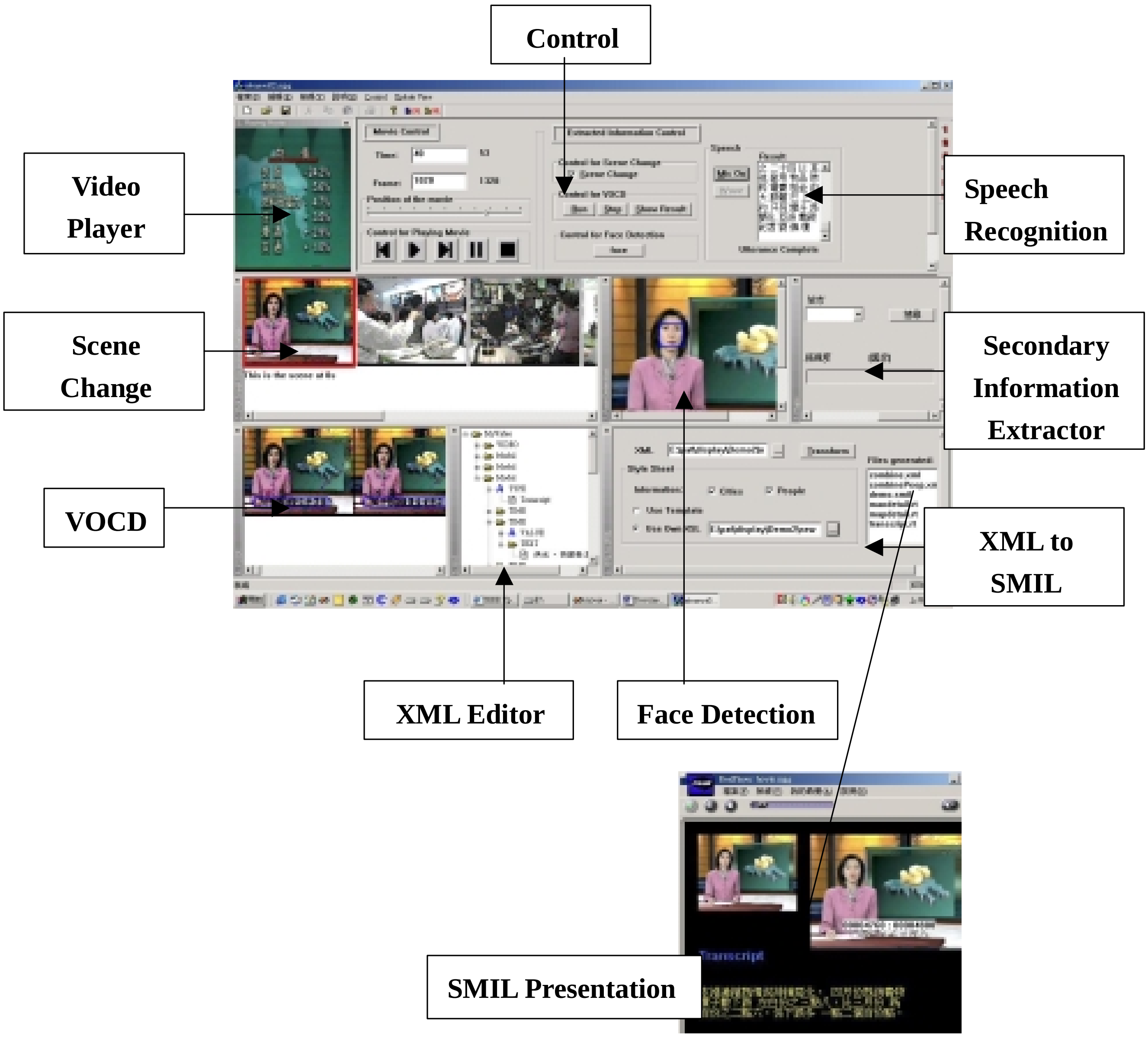


Figure 2.15 View of XVIP in second semester

3. Informedia

3.1 Review

Recent years have been a rapid increase in the usage of multimedia information. Of all media types (text, image, graphic, audio and video), video is the most challenging one, as it combines all the other media information into a single data stream and it becomes the most popular source of multimedia information. Informedia is studied as we try to find out the current techniques used in these Informedia. This enables us to get experience to implement the integrated-tool.

There is a lot of attention paying on how to present multimedia information. However, a little concern is on how to do information extraction from video and storage of this information. These techniques are useful for indexing and searching on video. Our final year project will focus on how to do information extraction from video and storage of this information.

Digital Video Library is a new approach for automated video and audio indexing, navigation, visualization, search and retrieval and embedded them in a system for use in education, information and entertainment environments. It includes a lot of techniques on video information extraction, storage, indexing, etc. Therefore, understanding the techniques in Digital Video Library can gives a lot of ideas on how to do our final year project.

3.2 Digital Video Library System

A digital library of the future will provide electronic access to information in many different forms. Recent technological advances make the storage and transmission of digital video information possible. Digital Video Library System (DVLS) is suitable for storing, indexing, searching, and retrieving video and audio information and providing that information across the Internet.

To be an effective library, users need to be able to find the video segments they want. Realizing this goal requires automatic content-based indexing of videos that will significantly improve the users' ability to access specific segments of interest with videos. Videos, soundtracks and transcripts will be digitized, and information from the soundtrack and transcripts will be used to automatically index videos in a frame-by-frame manner. This will allow users to quickly search indices for multiple videos to locate segments of interest, and to view and manipulate these segments on their remote computer.

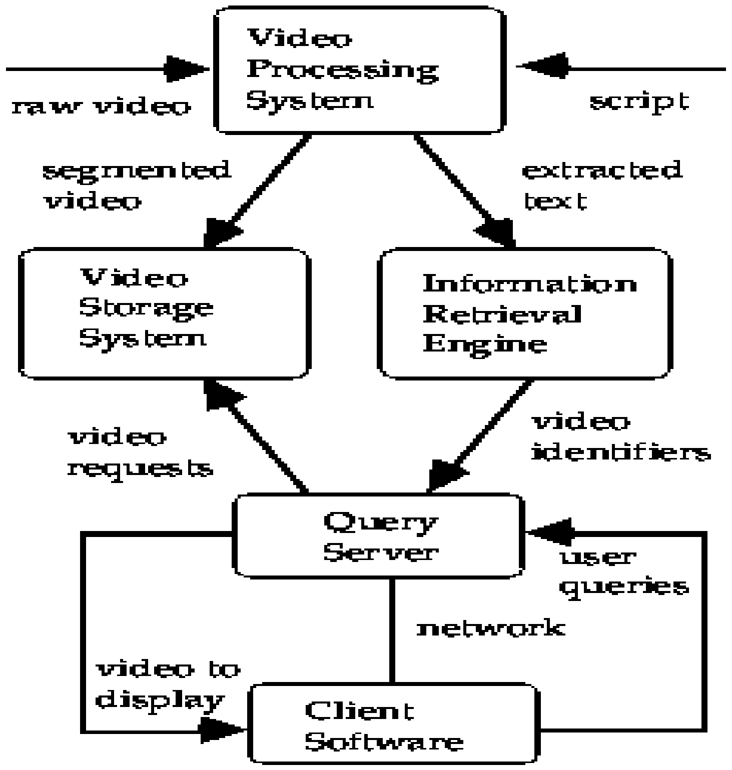


Figure 3.1 An Overview of the Digital Video Library Software components of the University of Kansas.

The Digital Video Library Software is a complex system composed of the following five primary components.

1. **Video Storage System (VSS):** The VSS stores video segments for processing and retrieval purposes. It is to provide intelligent access to portions of a video rather than entire videos; the VSS must be capable of delivering numerous short video segments simultaneously.
2. **Video Processing System (VPS):** The VPS consists of video processing programs to manipulate, compress, compact, and analyze the video and audio components of a video segment. In particular, the VPS contains a component to recognize keywords from the sound track of video segments.
3. **Information Retrieval Engine (IRE):** The IRE is used to store indices extracted from video segments and other information about the video segments, such as source, copyright, and authorization. The IRE will be capable of supporting both free-text and Boolean queries.
4. **Client:** The Client is a graphical user interface, which resides on the user's computer. It includes interfaces for conducting structured and free text searching, hypertext browsing and a simple video editor.

5. **Query Server (QS):** The QS processes video queries from the remote Client and communicates with the IRE and VSS to enable users of the digital library to extract video data and create multimedia representations of the information of interest.

3.3 Digital Video Library in CMU

One representative project working on Digital Video Library is the Informedia Project at Carnegie Mellon University. The Informedia Digital Video Library project will establish a large, online digital video library featuring full-content and knowledge-based search and retrieval. Intelligent, automatic mechanisms will be developed to populate the library.

Search and retrieval from digital video, audio, and text libraries will take place via desktop computer over local, metropolitan, and wide area networks. Initially, the library will be populated with 1,000 hours of raw and edited documentary and education videos.

In our Final Year Project, the digital video library structure that we are using is basically referenced to the Informedia Digital Video Library project at Carnegie Mellon University.

Digital video presented a number of interesting challenges for library creation and deployment. For example, the way it embeds information, its voluminous file size, and its temporal characteristics. These challenges are basically addressed by:

1. Automatically extracting information form digitized video
2. Creating interfaces that allowed users to search for and retrieve videos based on extracted information.
3. Validating the system through user test beds.

The following is the Informedia Digital Video Library System Overview:

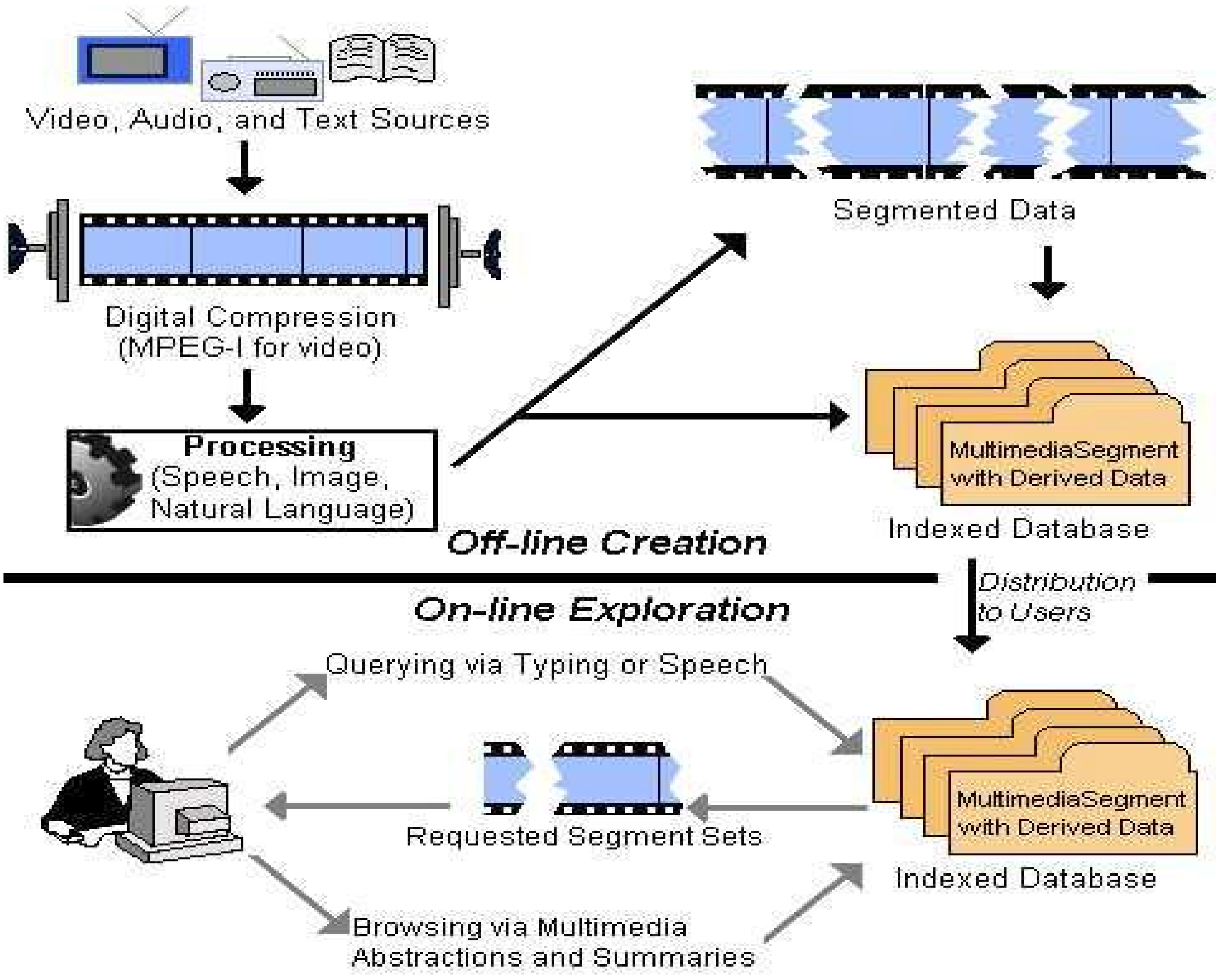

Video, Audio, and Text Sources
Digital Compression (MPEG-I for video)
Processing (Speech, Image, Natural Language)
Segmented Data
MultimediaSegment with Derived Data
Indexed Database
Off-line Creation
On-line Exploration
Distribution to Users
Querying via Typing or Speech
Requested Segment Sets
MultimediaSegment with Derived Data
Indexed Database
Browsing via Multimedia Abstractions and Summaries

Figure 3.2 Informedia Digital Video Library System Overview

3.4 Video Information

There are many information can be extracted from the video sequence:

1. **Integration of speech, language, and image processing:** generating multimedia abstractions, segmenting video into stories, and tailoring presentations based on context.
2. **Text processing:** headline generation, text clustering and topic classification, and information retrieval from spoken documents.
3. **Audio processing:** speech recognitions, segmentation and alignment of spoken dialogue to existing transcripts, and silence detection for better "skim" abstractions
4. **Image processing:** face detection and matching based on regions, textures, and colors.
5. **Video processing:** key frame selection, skim, video OCR, and Video trails.

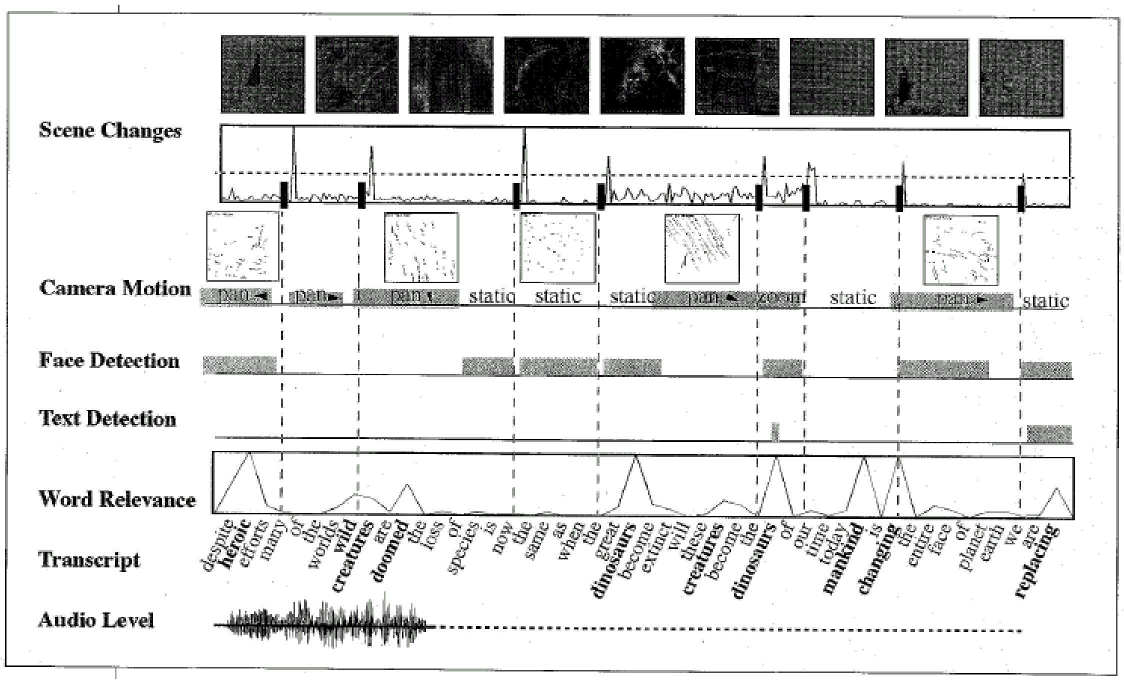

Scene Changes
Camera Motion
pan
pan
pan
static
static
static
pan
zoom
static
pan
static
Face Detection
Text Detection
Word Relevance
Transcript
despite heroic efforts many of the worlds wild creatures are doomed the loss of species is now the same as when the great dinosaurs become extinct will these creatures become the dinosaurs of our time today mankind is changing the entire face of planet earth we are replacing
Audio Level

Figure 3.3 The above diagram shows the Component technologies applied to segment video data.

3.5 Video Information Extraction

Videos contain rich information that will be useful in indexing and searching. There are a lot of techniques for extracting and manipulating this kind of information. Content of a video is conveyed both the narrative (speech and language) and the image. Only by the collaborative interaction of image, speech, and natural language understanding technology can successfully populate, segment, index, and search diverse video collections with satisfactory recall and precision. Now we can go to briefly describe image, speech and natural language one by one.

3.5.1 Image

Image understanding plays a critical role in Informedia for organizing, searching, and reusing digital video. When the digital video library is formed, the first requisite capability is video segmentation. This process can segment video into group of frames that represent different stories. Also, scene transition effects such as fade, dissolves, and cuts can be automatically detected by comparing color histograms, discrete cosine transform coefficients, shape, and texture measures.

The following diagram is the Informedia image-understand video processing overview:

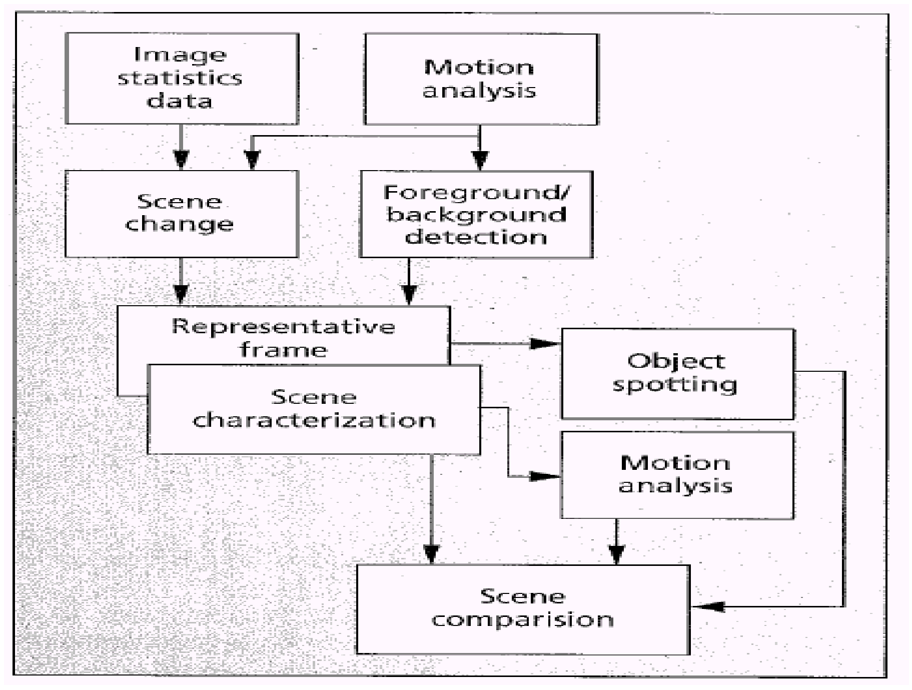


Figure 3.4 Informedia image-understand video processing overview:

Currently image processing techniques are used to partition each video segment into shots, choose a representative frame (key frame) for each shot -- usually the middle frame in one shot, but also can be the last one frame if the shot contains a camera motion.

Definitions:

A **shot:** a video clip recorded with one continuous camera operation.

A **segment:** several shots describing a topic.

Other than segmentation and detection of scene changes, object-presence detections are also an important technique. Identify and index features to support image similarity matching. Face recognition is a good example. It can show the name of people appearing in the video or how they are interacting with the environment.

Another essential detection technique is that of textual information appearing in the video but not repeated in the audio. We can first find the caption in video, then use OCR software to extract the text in it By detecting the clustered and often high- contrast structure of printed characters, we can extract regions form videos that contain text. The video optical character recognition always contains important information, especially for news video.

The following diagram shows an example of face and text detections:

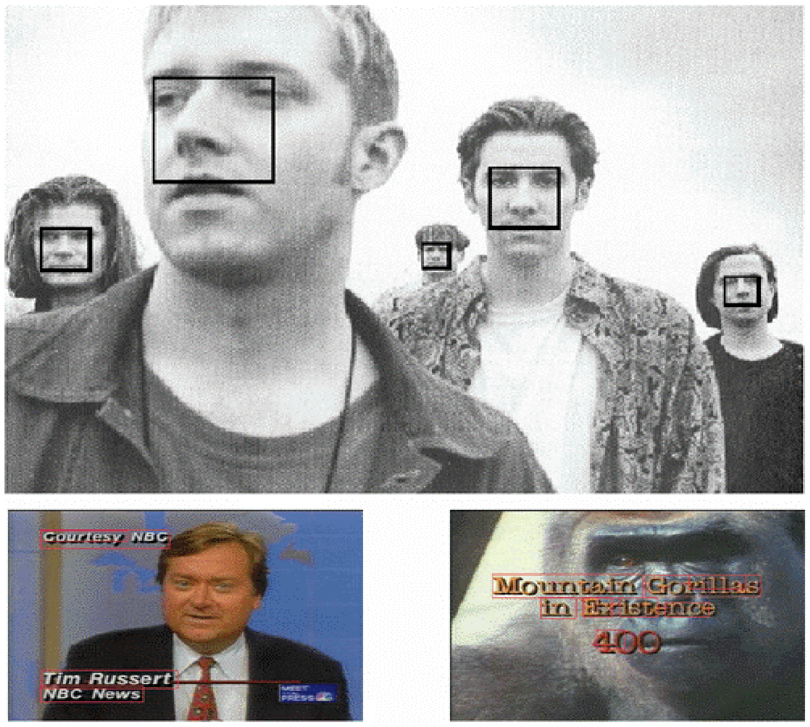


Figure 2.5 example of face and text detections

3.5.2 Speech

Even though much of broadcast television is closed-captioned, most of the nation's video and film assets are not. More importantly, typical video production generates 50 to 100 times more content than what is broadcast and is thus not captioned. We therefore combine automatically generated transcripts, containing tolerable errors, with captioning (where available) for the analysis, indexing, and retrieval of multimedia data.

Unlimited-vocabulary, speaker-independent, connected-speech recognition is an incompletely solved problem. However, recent results in domain-specific applications demonstrate the promise and potential of being able to automatically transcribe spoken language with an unlimited vocabulary. The keyword extracted from the transcript will be useful in indexing the video

3.5.3 Natural language

Library search and retrieval, precision, and recall can be improved through natural- language processing to understand and expand the user's query and to associate it with correct but inexact matches from the library's content. This lets us go beyond limited keyword matching in our library search. Natural-language processing in Informedia is applied to both query processing and library creation. It serves four principal functions--spoken and typed free-form query processing, ranked retrieval, automated transcript correction, and summarization for use in title generation and video abstract creation.

3.6 Integration on extracted information

Through combined techniques from language and image understanding, video skims of the original video at varying compression ratios are obtained. This compact video is created with significant image and audio regions to produce a synopsis of the original, which can also be used to select a single representative frame for each scene.

By examining the audio level for additional clues to detect transitions between speakers and topics, which often correspond to low energy or silence in the signal. Having segmented the video, we statistically compute the relative importance of each scene's image content. Objects such as human faces and text can be identified in video and used as a basis for significance during skim creation.

The unsynchronized audio and video are now integrated into an effective skim of the original content. The keywords and significant images are selected for skim creation as follow:

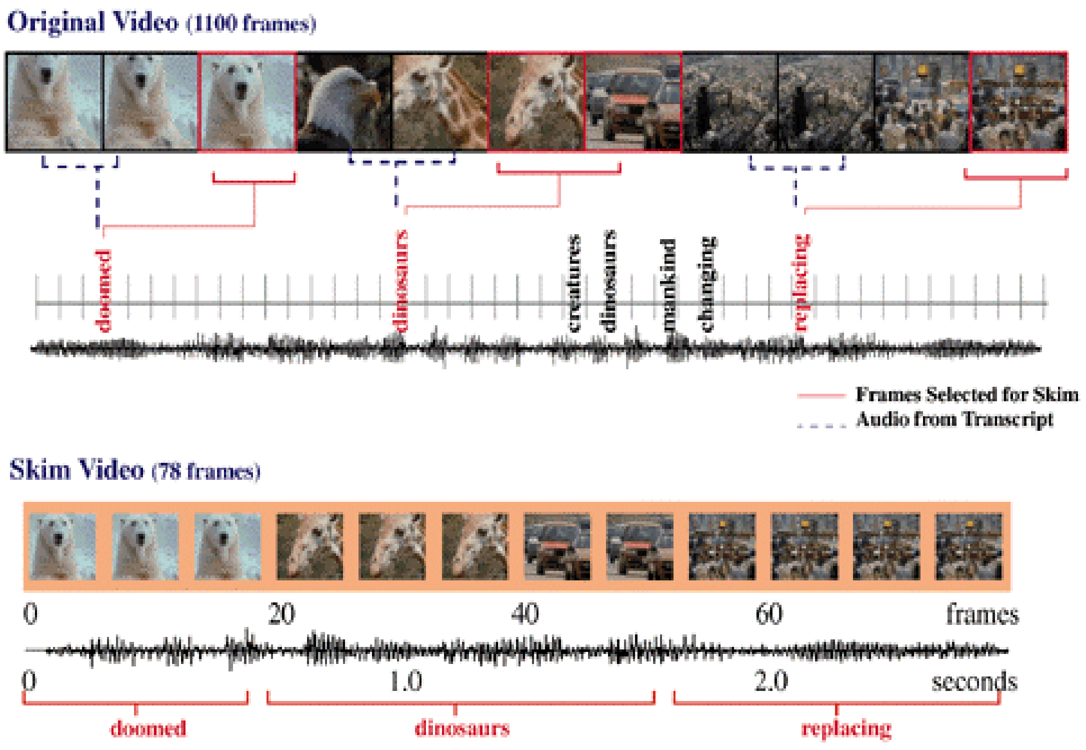
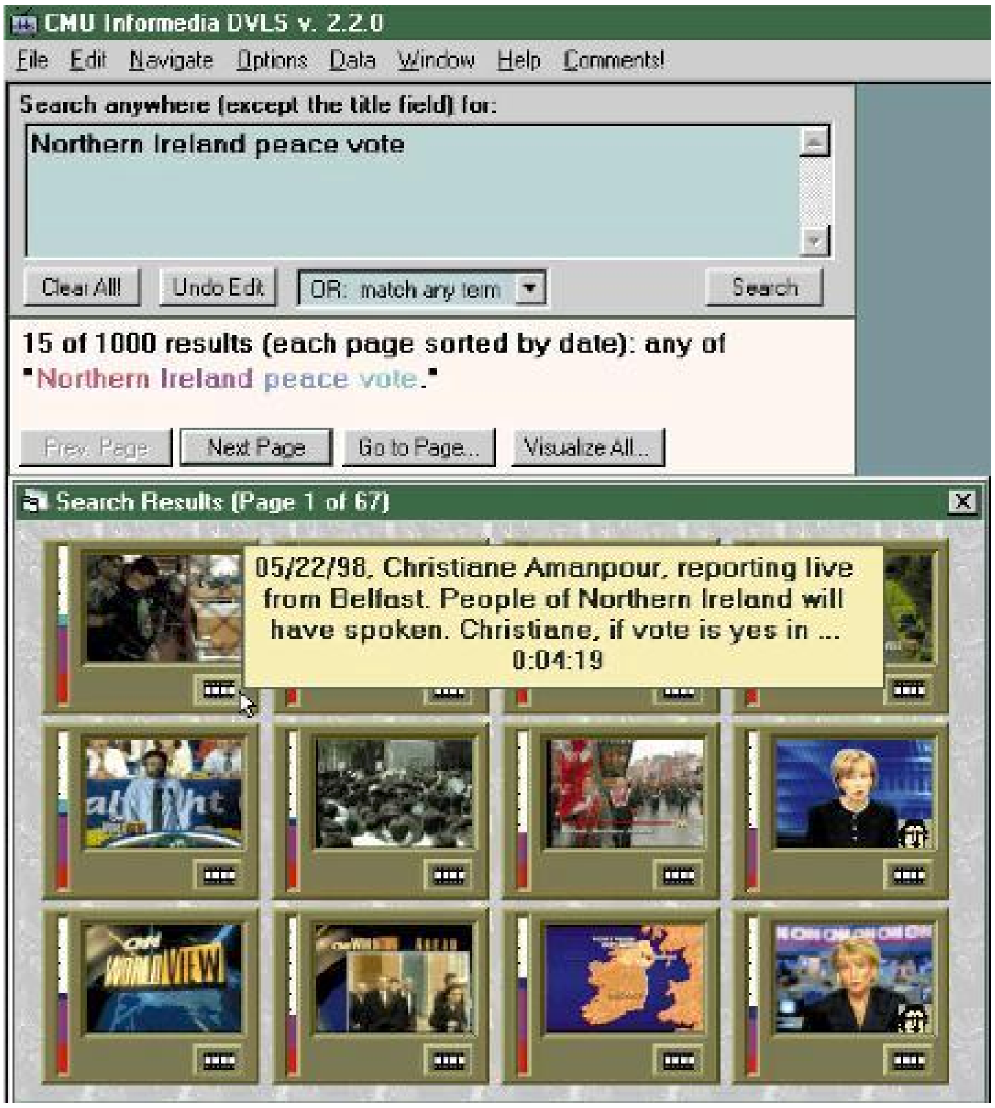


Figure 3.6 keywords and significant images are selected for skim creation

3.7 User interface

(From Lessons learned from building a terabyte digital video library)

The information interface was designed to provide users with quick access to relevant information in the digital video library. The user interface includes query input and result set interface for the Informedia digital video library. Generally, abstractions include headlines, thumbnails, filmstrips, and skims.


CMU Informedia DVLS v. 2.2.0
File Edit Navigate Options Data Window Help Comments!
Clear All! Undo Edit OR: match any term Search
Prev. Page Next Page Go to Page... Visualize All...
Search Results (Page 1 of 67)

Search anywhere (except the title field) for:

Northern Ireland peace vote

15 of 1000 results (each page sorted by date): any of "Northern Ireland peace vote."

05/22/98, Christiane Amanpour, reporting live from Belfast. People of Northern Ireland will have spoken. Christiane, if vote is yes in ... 0:04:19

Figure 3.7 interface for the Informedia digital video library

Headlines: Early Informedia implementations created headlines for segments based on the "most significant" words in their text (for example, the text in the transcript or VOCR). If the word occurred often in the txt associated with the video segment and infrequently in the whole library's text, then that word became part of the headline.

Thumbnails: The thumbnail is some important frame. The thumbnail was chosen based on the query, by using the key frame for the shot producing the most matches for the query, then pictorial menus produced clear advantages over text-only menus.

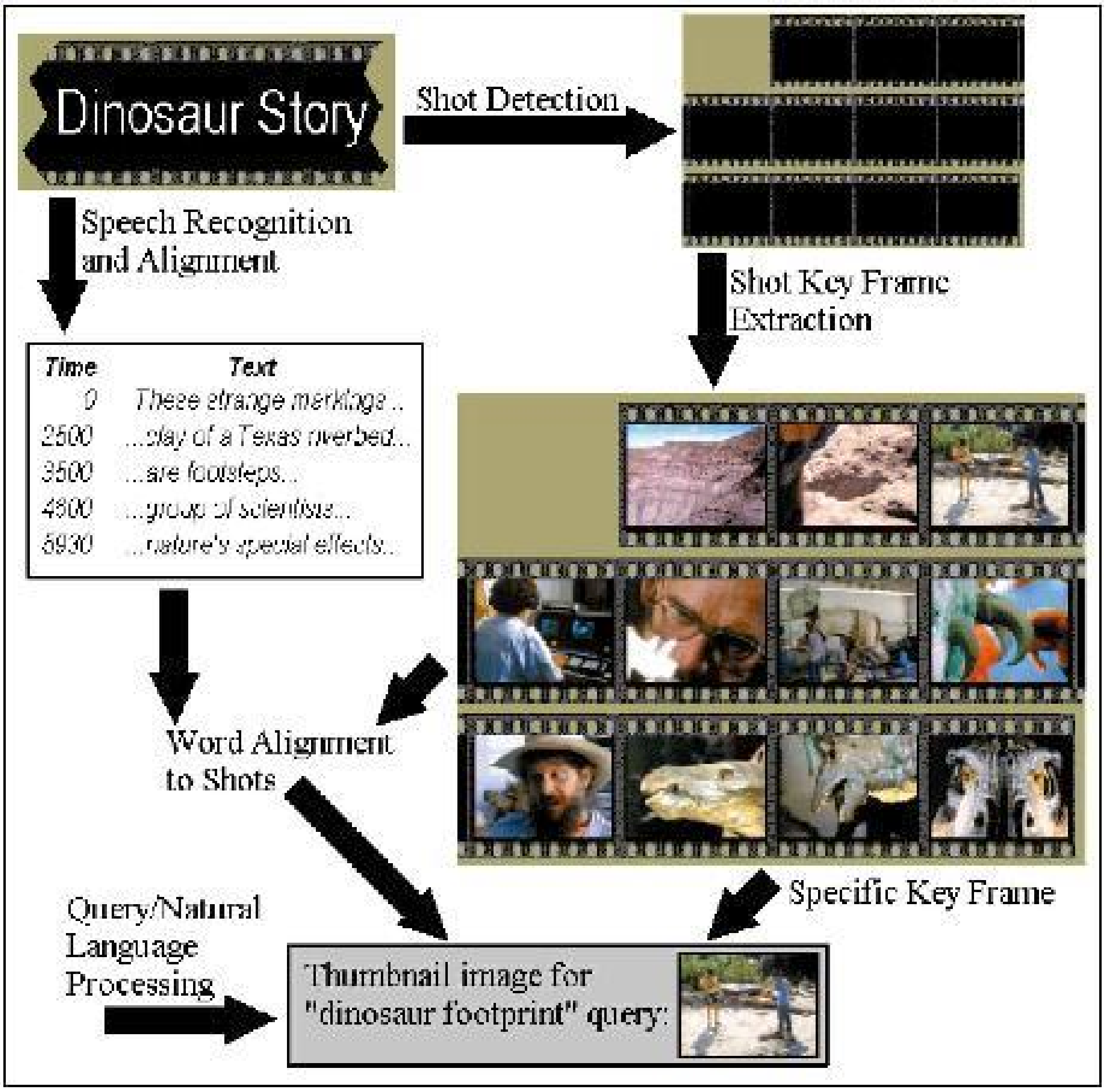

Dinosaur Story
Shot Detection
Speech Recognition and Alignment
Shot Key Frame Extraction
Time Text
0 These strange markings..
2500 ...clay of a Texas riverbed...
3500 ...are footsteps...
4800 ...group of scientists...
5930 ...nature's special effects...
Word Alignment to Shots
Specific Key Frame
Query/Natural Language Processing
Thumbnail image for "dinosaur footprint" query:

Figure3.8 Overview

Filmstrips: Key frames from a segment's shots can be presented in sequential order as filmstrips. The segment's filmstrip can show a story matching the query.

Skims: Video skim is a temporal multimedia abstraction, which is played rather than viewed statically. A skim incorporates both video and audio information from a longer source so that a 2-minute skim, for example, may represent a 20-minute original video.

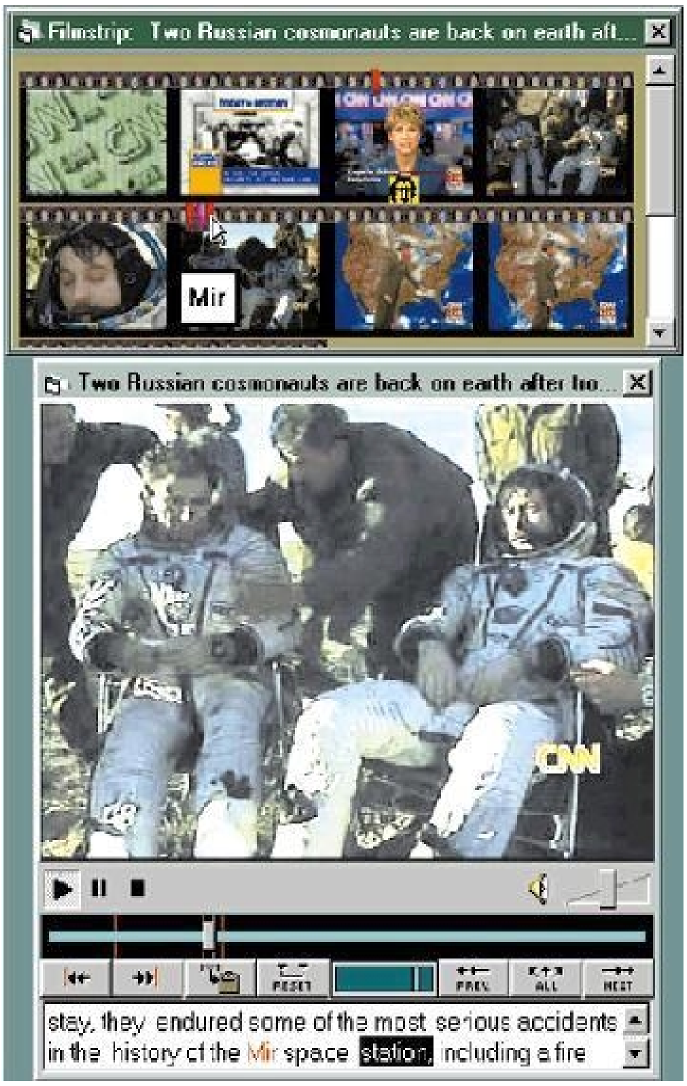

Filmstrip: Two Russian cosmonauts are back on earth aft...
Two Russian cosmonauts are back on earth after lo...
Mir
stay, they endured some of the most serious accidents in the history of the Mir space station, including a fire

Figure3.9 Skim

For our tools, we focus on the video processing and we have chosen 2 extraction techniques of them, scene change detection and VOCR to implement to demonstrate how the data can be extract from the video as the fundamental data and store as XML Then we can extract the second data from the XML.

4. Extraction Techniques

4.1 Introduction

As state in the pervious chapter, there are rich and useful information in videos, it is necessary to have efficient and effective extraction techniques to extract all these useful information from them.

In XVIP, there are four extraction techniques are implemented; they are scene changes, video optical character detection, face detection and speech recognition. Scene change technique is chosen as it is the most effective method for segmenting a video sequence into significant components, generally called shots (scene change also called shot break). For face detection, it is useful for indexing and searching the data. After the location of the face is detected, the face recognition engine can be applied and more data of the video can be retrieved and make the content of the video be more searchable. The input frame for face detection is the output of scene change, as we believe that the most representative frames are included by scene change. For VOCR, it is a technique that can greatly help to understand the details of the video and locate topics of interest in a video as it can extract the captions on frames. To make the system to be more complete, speech recognition is included, so that both video and audio data are processed. Speech recognition engine extracts the transcript from the audio channel and it can help for indexing the data. The following section is going to introduce the extraction techniques applied in XVIP.

4.2 Scene Change Detection

4.2.1 Review

As mention in pervious section, scene change detection technique is need for extracting video information in digital video library. Scene change detection is an effective method for segmenting a video sequence into significant components, generally called shots (scene change also called shot break). So, automatically extract key information from image and videos for the purpose of indexing, fast and easy retrievals, and scene analysis is necessary.

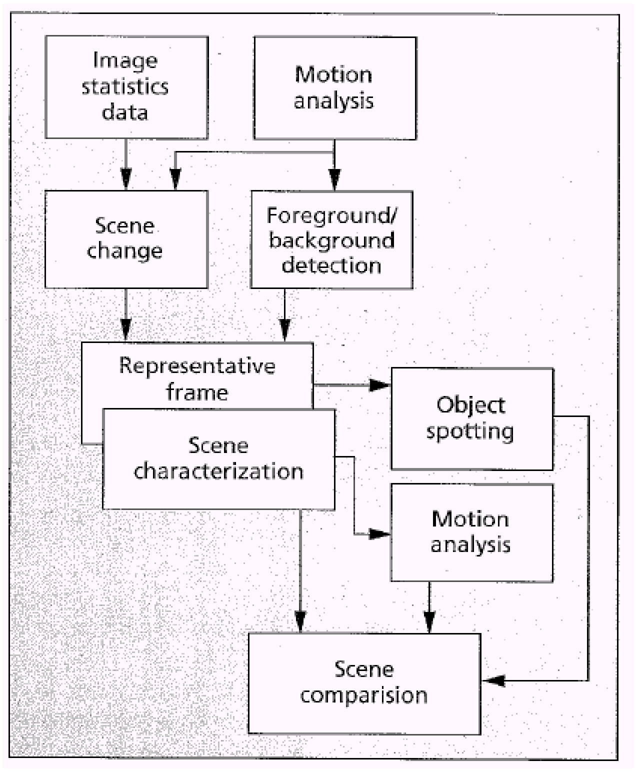


Figure 4.1 Informedia image-understanding video processing overview

We try to implement the scene change detection technique because for video, a common first step is to segment the video into temporal shots. Each represents an event or continuous sequence of action. A shot in video refers to a continuous recording of one or more video frames depicting a continuous action in time and space.

From the result of some research, statistical and structural properties of images are used to identify scene change. These features are used in three steps to identify these scene changes sequentially, such as abrupt scene change detection, dissolve scene change detection, and wiping scene change detection. Since abrupt scene change is the most common in video sequence, we also try to use this approach, abrupt scene change detection, to find the scene change in the video.

The accuracy and execution speed of the abrupt scene change detection algorithm is critical if large amounts of video data are to be processed. In resent years, there is already have effective scene change detection techniques exist. We are going to discuss and compare the performance of the abrupt scene changes (or called shot boundaries) detection algorithm, describe the one that we have chosen to implement in our tools, and a brief description of dissolve and wiping scene change detection method and the technique for key-frame extraction.

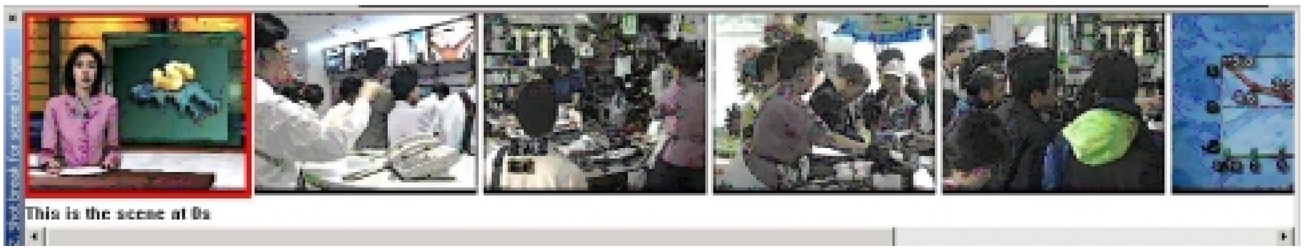


Figure 4.2 Implement result of the Scene change

4.2.2 Review Existing Scene Changes Detection Technique

A shot is defined as an unbroken sequence of frames recorded from a single camera, which forms the building block of a video. The purpose of shot boundary detection is to segment the video stream into multiple shots.

Many techniques to automatically determine shot break in a video sequence have been proposed. Such techniques can be classified into two categories: cut detection techniques for compressed video sequence and uncompressed video sequence.

4.2.2.1 Cut Detection Methods

(a) **Image difference method**

Image difference method is base on the similarity of consecutive frames. If the scene is continuous and no scene change occurs, the frame difference is kept small. At the scene cut point, its difference becomes relatively large compared with that of continuous scenes. Image difference method id defines as:

$$\boxed{D_{i,j}(x,y) = (1/MN)(\Sigma\Sigma\triangle P_{i,j}(x,y)}$$

Where M and N are the number of pixels in the image in the horizontal direction and in the vertical direction respectively.

$\triangle P_{i,j,}(x,y) = | P_{i,}(x,y) - \triangle P_{j}(x,y)|$ is the inter-frame difference between the i th frame and the j th frame. $P_{i}(x,y)$ is the intensity of pixel position (x, y) at the i th frame.

This is a simplest way of evaluating an image difference. However, there are some disadvantages.

- Every pixel in a frame is investigated for every frame in order to achieve the detection. Therefore the detection speed is slower.
- Over-detection caused by various motions.

(b) Histogram Difference Method

Video data is classified into K classes. In each frame, the number of the pixels categorized into each class is calculated. Then the summation if the two consecutive frames is calculated and a threshold can be used to determine scene cut detection. This is defined as:

$$\boxed{HD_{i,j}(x,y) = \Sigma\ |H_{i,}(k) - H_{j}(k)|}$$

$H_{i,}(k) = \Sigma\Sigma_{m,n}$ 1 if $P_{i,}(x,y)$ belongs to k and $k = \{ P_{i,}(x,y) \mid k <= P_{i,}(x,y) < k+1$

$H_{i,}(k)$ represent the relative frequency of occurrence of the k bin in the ith image. For this method

- Advantage is over-detection caused by motions is reduced as long as they move inside of the frame since the histogram is left unchanged.
- Disadvantage is detection speed is also slow since every pixel in a frame is examined.

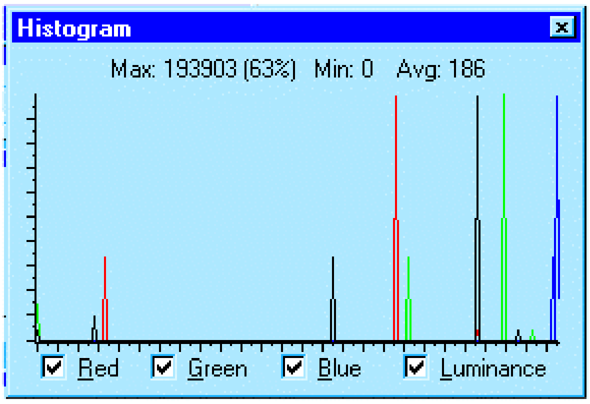


Figure 4.3 Color Histogram

(c) Histogram Difference Method using DC Coefficient Image

Digital video sequences are usually stored in a compressed format such as MPEG. The most computationally expensive part of the decoding algorithm is the inverse DCT (IDCT) operation. DC images are spatially reduced versions of the original images. DC images can be efficiently extracted from compressed videos without IDCT operations. A fast scene cut algorithms may be achieved using DC coefficient images.

In an intra-coded picture (1 picture) the images is divided into 8 X 8 blocks and the DC terms, is 8 times the average intensity of the block given by

$$\boxed{DC = (1/8) \Sigma\Sigma P_{i,}(x, y)}$$

For this method,

- Advantage is that it is a faster cut detection algorithm for the reduction of the processing time using 1/64 scaled images.
- Disadvantage is that it is difficult develop the more accurate detection of the scene changes as approximated images are used.

4.2.2.2 Dissolve and wiping scene change detection

Apart from the abrupt scene change detection, there exist others scene change detection method, dissolve and wiping scene change detection. Dissolve scene change detection is the next common scene transition. This can be achieved by considering the ratio between the second derivative of the variance and the first derivative of the mean and spikes of the second derivative of the variance. If a dissolve region is not detected, then wipe transitions are identified using statistical features together with structural properties.

4.2.3 Our Algorithm & Implementation

After reading several algorithm and the experiment results from the papers, we find that the accuracy of the histogram difference method is acceptable. If we use it with a dynamic threshold, the accuracy of the scene change can be improved and reduce lots of over-detection and un-detection. Let us describe our approach.

4.2.3.1 Histogram difference

We grasp one scene from the video for every 0.05 seconds and it is compared with the pervious scene. The grasped scenes are 24-bit image, 8 bits for each color (red R, green G, blue B). So, we can check each pixel and classified them into different class.

For our algorithm, we only consider the most significant 2 bits of each color. Therefore, we can classify the pixel into 64 different colors. After checking each pixel and add one for the corresponding class, we can build a color histogram for each image.

```
Array[64]
For each pixel
    Get the first 2 bit of R (r1, r2)
    Get the first 2 bit of G (g1, g2)
    Get the first 2 bit of B (b1, b2)
    Array [r1r2g1g2b1b2] ++
```

After building the color histogram for the two successive scene, we compare them. For each column of the histogram, we calculate the difference of them and sum all the square of the difference. If the total difference of the whole histogram is greater than a threshold, we consider it as a scene change.

$$H = \Sigma_{(0, 63)}(P_{a,}(i) - P_{b}(j))^2$$

If H > threshold (T), we consider it as a scene change. The threshold can be adjusted to reduce over-detection and un-detection

Using this approach, the result is acceptable, and the advantages are:

- Fast for detecting scene change
- Over-detection caused by motions is reduced as long as they move inside of the frame since the histogram is left unchanged.
- Un-detection can be reduced by using appropriate threshold

- Easy to implement

It can be improved by checking more bits of each color in the pixel. However, it will be more sensitive to the threshold. If the threshold chosen is not appropriate, the number of over-detection and un-detection will be increase.

4.2.3.2 Dynamic threshold

Fixed threshold cannot perform equally well for all videos, which must be assigned to tolerate variations in individual frames, while still ensuring a desired level of performance. A high threshold value can prevent over-detection, but increase the un-detection. Conversely, a low threshold value enables consistent cuts to be accepted, but increase the number of the over-detection. Therefore, obtaining a adaptive threshold becomes a important problem. With a correct threshold, we can get a accuracy result.

A fixed appropriate threshold value is selected by experiment. Thus, dynamic threshold can be used to improve the accuracy of scene change detection.

This dynamic threshold can be determined by the minimum of the difference of the color histogram. We apply one-dimensional entropic thresholding to histogram difference to find the optimal *(T)*.

Let the largest value of these histogram distance *(SD_i)* features be *W*, that is

$$\boxed{W = \max \{ SD_i \} \quad ;\text{for } i=1, 2, \dots L-1}$$

where L is the sequence length. We divide the range of these values into *W+1* bins, then, there are *W+1* elements of histogram distances. Each element of the histogram disrtance features has a frequency f_i which indicates the number of frames that differ from the succeeding frame by a histogram distance equals to *i.*

$$\boxed{f_i = \Sigma_{(j=1,\, L-1)}\, \delta\, (SD_i - i) \quad ;\text{for} \quad 0 <= i <= W}$$

here δ $(i - j)$ = 1 when i = j otherwise 0. As the distributions are assumed to be independent, the probability for the frames with the scene cut relationships with their successive frames

$$\boxed{P_s\,(i) = f_i / \Sigma_{(h=0,\, T)}\, f_h \quad ;\ 0 <= i <= T}$$

The entropy for can be defined as

$$\boxed{H_s(T) = \Sigma_{(i=0,\,T)} P_s(i) \log P_s(i)}$$

The global threshold T, which is used to classify a frame as being a scene change, is chosen to satisfy the following criterion function (Kapur et al., 1985)

$$\boxed{H(T) = \max \{ H_s(T) \} ;T=0,1,2\ldots.W}$$

4.2.4 Key Frame Extraction

After shots are segmented, key frames can be extracted from each shot. Depending on the content complexity of the shot, one or more key frames can be extracted for a single shot.

There are several difference approaches for extraction key frame:

1. Shot boundary based approach (the approach used in the tool)
   After the video stream is segmented into shots, a natural and easy way of key frame extraction is to use the first frame of each shot as the shot's key frame. Although simple, the number of key frame for each shot is limited to one, regardless of the shot's visual complexity.

2. Visual context based approach
   There are multiple visual criteria to extract key frame:
   - Shot based criteria: the first frame is always selected as the first key frame; more than one key frame need to be selected depends on other criteria
   - Color feature based criteria: by comparing the successive frame, if there is significant content change, select the front one as the key frame
   - Motion based criteria: For zooming-like and panning-like shot, the first and the least frame are selected as the key frame as they represent a global and more focused view respectively.

3. Motion analysis based approach
   The optical flow for each frame is computed, and then computes a simple motion metric based on the optical flow. Finally the metric is analyzed as a function of time to select the key frame at the local minima of motion, this key frame is identified by stillness.

4. Shot activity based approach
   The intra and reference histograms are computer and then compute an activity indicator. Based on the activity curve, the local minima are selected as the key frames.

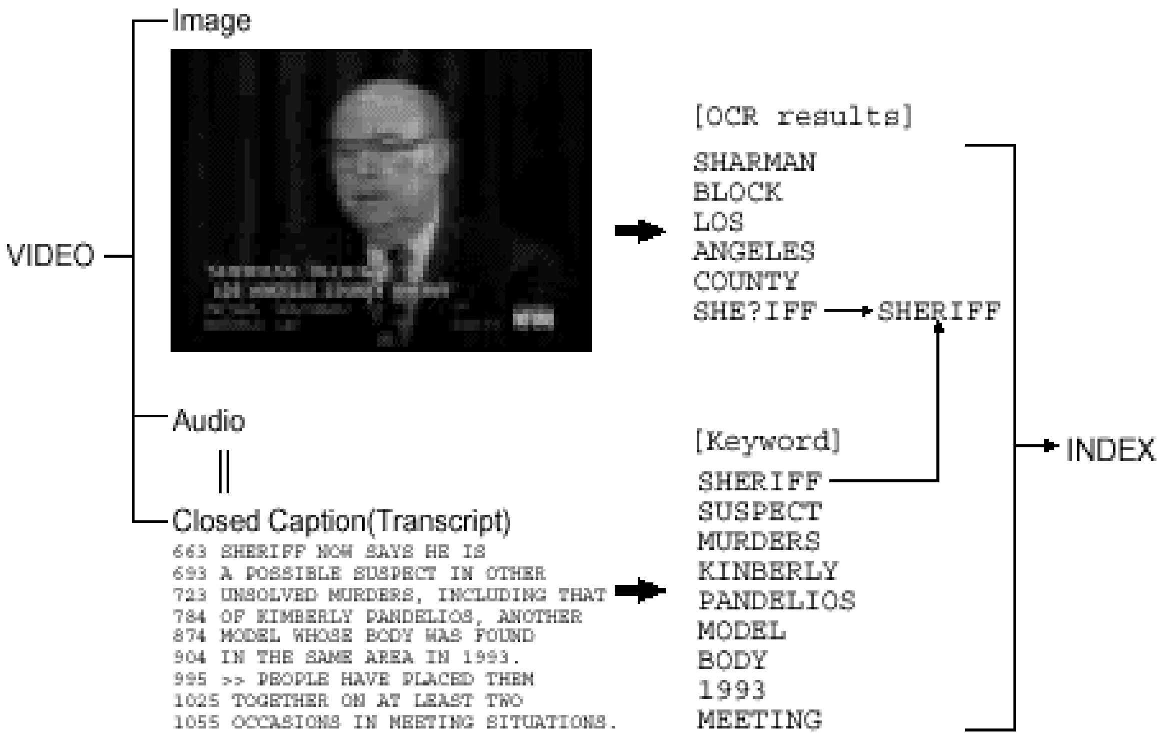
As we implement the scene change detection and get the key frame aims to demonstrate how to extract and store the data as XML, we try to choose the one is less complexity and relatively less difficult to implement. Thus, we choose Shot boundary based approach to get the key frame of each shot.

4.3 Video Optical Character Recognition (VOCR)

4.3.1 Review

Understanding the content of videos requires the intelligent combination of many technologies, such as speech recognition, natural language processing, search stragies, image understanding, etc.

Video OCR is a technique that can greatly help to locate topics of interest in a large digital news video archive via the automatic extraction and reading of captions and annotations. News captions generally provide vital search information about the video being presented -- the names of people and places or descriptions of objects. Performing Video OCR on video and combining its results with other video understanding techniques will improve the overall understanding of the video content. For example, caption graphically superimposed to news video can provide an important supplemental source of indexing information.



```
663 SHERIFF NOW SAYS HE IS
693 A POSSIBLE SUSPECT IN OTHER
723 UNSOLVED MURDERS, INCLUDING THAT
784 OF KIMBERLY PANDELIOS, ANOTHER
874 MODEL WHOSE BODY WAS FOUND
904 IN THE SAME AREA IN 1993.
995 >> PEOPLE HAVE PLACED THEM
1025 TOGETHER ON AT LEAST TWO
1055 OCCASIONS IN MEETING SITUATIONS.
```

Figure 4.4 Video OCR results

The above picture shows the Video OCR results. The OCR results extracted the keywords on this frame. These keywords can be used together with the words extracted from the transcript for indexing the video.

Compared with OCR from document images, caption extraction and recognition in news video presents new challenges:

- Complex background
- Low resolution of characters

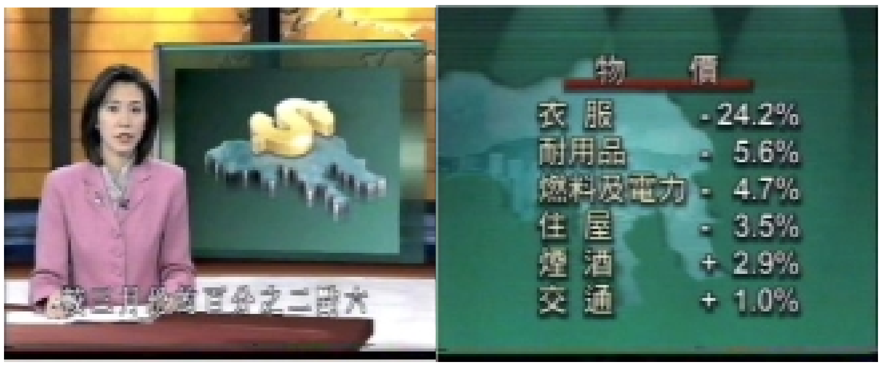


Figure 4.5 Here are two frames extracted from the Digital news with characters on them:

It can be observed that the two frames are with complex background and low resolution of characters.

Since Video OCR is especially important in news video, we will focus on how to do Video OCR for Digital News in the following parts.

The main steps of Video OCR for Digital News are as follow:

1. **Detection of Text Region**

   -- For locating the caption from the video frame.

2. **Image Enhancement**

   -- For increasing the resolution of each caption and reducing the variability in the background.

3. **Character Recognition and Segmentation**

   -- For recognizing characters and doing segmentation between characters.

4. **Post-Processing**

   -- For improving the recognition rate.

The following paragraphs will explain the techniques in each step one by one.

4.3.2 Detection of Text Region

Since a video news program comprises huge numbers of frames, it is computationally prohibitive to detect each character in every frame. Therefore, we first roughly detect text regions in groups of frames to increase processing speed. Some known constraints of text regions can reduce the processing costs.

A typical text region can be characterized as a horizontal rectangular structure of clustered sharp edges, because characters usually form regions of high contrast against the background.

We select frames and extract regions that contain textual information from the selected frame. For extraction of vertical edge features, we apply a horizontal differential filter to the entire image with appropriate binary threshold. If a bounding region, which is detected by the horizontal differential filtering technique, satisfies size, fill factor and horizontal-vertical aspect ratio constraints, it is selected for recognition as a text region. Detection results are selected by their location to extract specific captions, which appear at lower positions in frames.

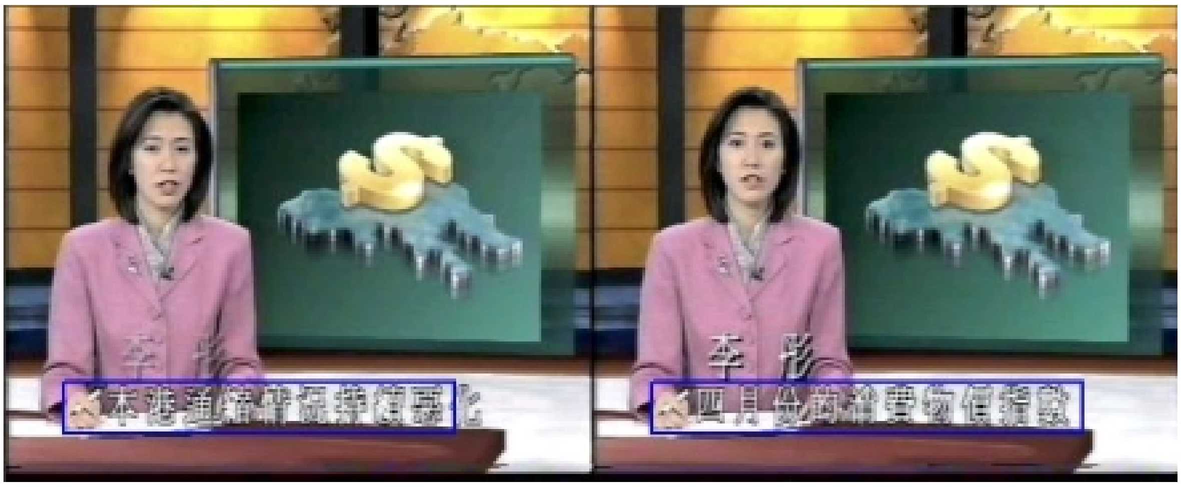


Figure 4.6 It can be seen that the captions are the horizontal rectangular boxes and is outlined with blue color.

4.3.3 Image Enhancement

In television news videos, the predominant difficulties in performing Video OCR on captions are due to low-resolution characters and widely varying complex backgrounds. To address both of these problems, we have developed a technique, which sequentially filters the caption during frames where it is present. This technique initially increases the resolution of each caption through a magnifying sub-pixel interpolation method. The second part of this technique reduces the variability in the background by minimizing (or maximizing) pixel values across all frames containing the caption. The resulting text areas have good resolution and greatly reduced background variability.

4.3.3.1 Sub-pixel interpolation technique

To obtain higher resolution images, we expand the low-resolution text regions by applying a sub-pixel interpolation technique. To magnify the text area in each frame by four times in both directions, each pixel of the original image I (x; y) is placed at every fourth pixel in both x and y directions to obtain the four times resolution image L (x; y): L (4x; 4y) =I (x; y). Other pixels are interpolated by a linear function using neighbor pixel values of the original image weighted by distances the video motion of non-caption areas. This technique results in text areas with less complex backgrounds while maintaining the existing character resolution. The following picture shows the example on how to do Sub-pixel interpolation:

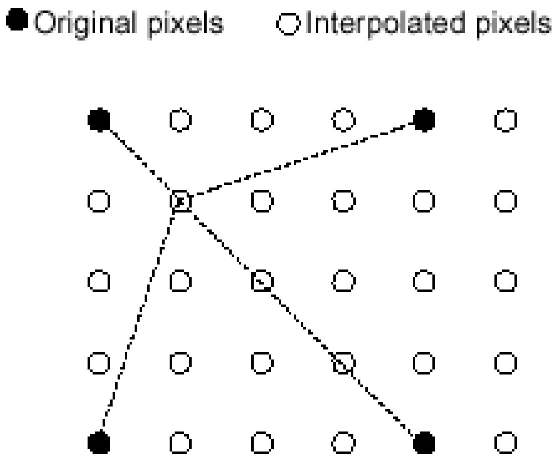


Figure 4.7 Example on how to do Sub-pixel interpolation.

4.3.3.2 Multi-frame Integration

For the problem of complex backgrounds, an image enhancement method by multi-frame integration is employed using the enhanced resolution interpolation frames. This technique reduces the variability in the background by minimizing (or maximizing) pixel values across all frames containing the caption. By taking advantage of the video motion of non-caption areas, this technique results in text areas with less complex backgrounds while maintaining the existing character resolution.

The sub-pixel interpolated frames, Li (x, y), Li+1 (x, y), . . . , Li+n (x, y) are enhanced as Lm (x, y) via
Lm (x, y) =min (Li (x, y), Li+1(x, y), . . . , Li+n (x, y))

where (x, y) indicates the position of a pixel and i and i + n are the beginning frame number and the end frame number respectively. These frame numbers are determined by text region detection. Here is an example of effects of both the multi-frame integration for the original image:

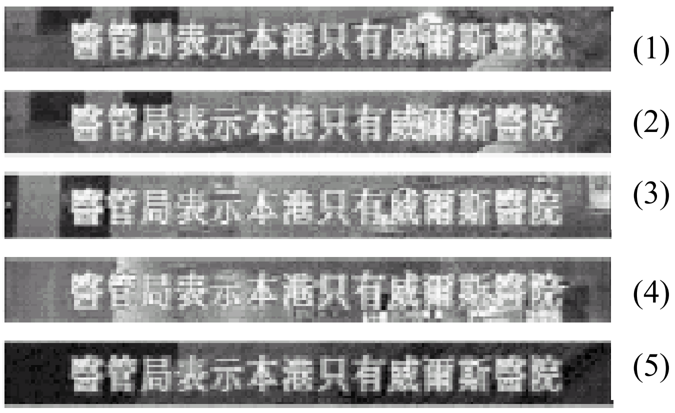


Figure 4.8 Effects of doing the multi-frame integration.

Line 1 to line 4 of the above picture shows the frame No.1, 10, 20, 50 with the same caption. Line 5 shows the enhanced image after doing multi-frame integration.

4.3.4 Character Recognition and Segmentation

A conventional pattern matching technique to used to recognize characters. An extracted character segment image is normalized in size and converted into a blurred gray scale image by counting the number of neighbor pixels.

Thresholding at a fixed value for the output of the character extraction filter Lfilter produces a binary image which is used to determine positions of characters and recognize characters.

The left character of the following picture is the character before doing binarization. The right character shows the binary image after binarization with thresholding.

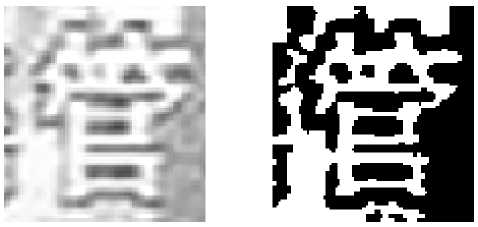


Figure 4.9 Image before (left) and after (right) binarization.

Then, we can filter the binary image with the morphological filter and filter the character image with connected component filter. The result is as follow:

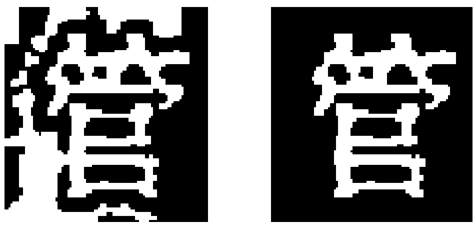


Figure 4.10 Image before (left) and after (right) morphological filtering.

Horizontal and vertical projection profiles of the binary image are used to determine candidates for character segmentation.
First, we have to do edge detection and smoothing. The top image below is the image before processing. The image at the bottom shows the processed image.

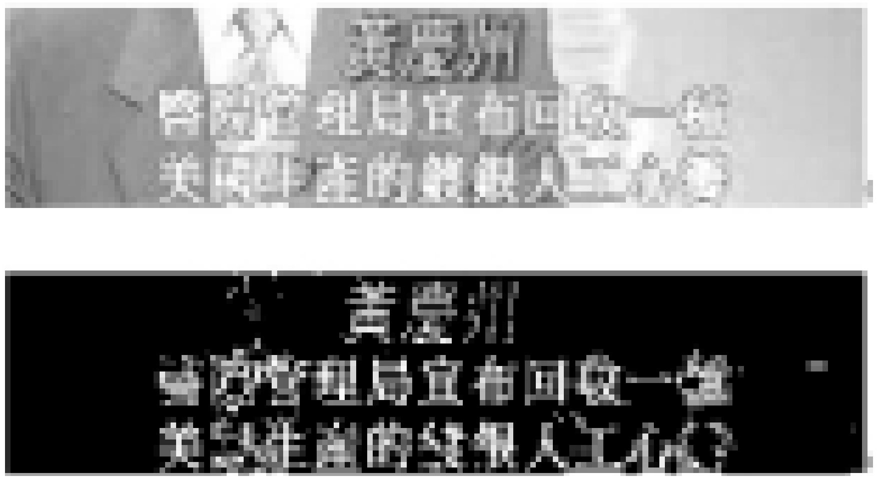


Figure 4.11 Image after doing edge detection and smoothing.

After that, we can do horizontal projection. After getting the result, we can do caption line separation easily. The result of horizontal projection is as follow:

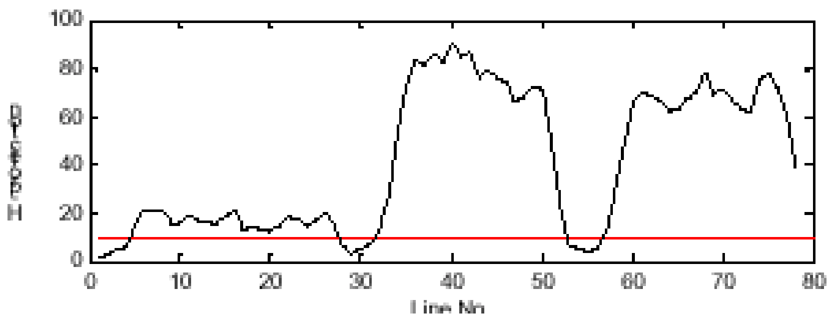


Figure 4.12 Result of horizontal projection.

After analyzing the result of horizontal projection, we can get the caption with line separated. The result caption is as follow:

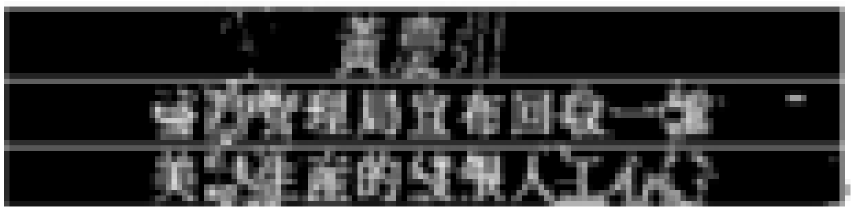


Figure 4.13 Result caption after line separation.

The white horizontal lines indicate the separation between different lines.

Then, we can do a vertical profile the caption on every horizontal line in the caption.
The result of vertical profile will be as follow:

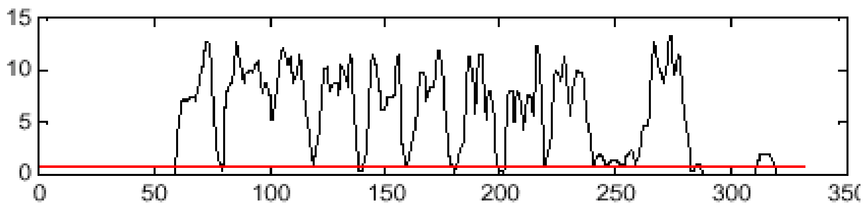


Figure 4.14 Result of vertical profile.

After getting the result, we process 2 consecutive segments at a time; we consider the first segment of the two to be correct and fixed if both the first and the second segments have high similarities. The left and right edges of peak of each character are shown by the arrows as the following picture:

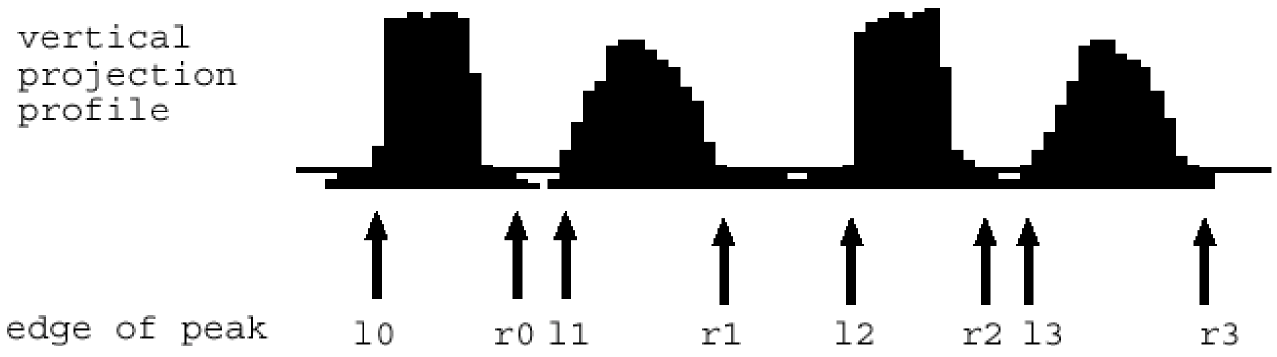


Figure 4.15 The vertical projection profile.

For example, the above picture indicates that there are four characters.

4.3.5 Post-Processing

To acquire text information for content-based access of video databases, high word recognition rates for Video OCR are required. We apply post-processing, which evaluates differences between recognition results with words in the dictionary, and selects a word having the least differences. This can improve the recognition rate

To acquire text information for content-based access of video databases, high word recognition rates for Video OCR are required. The Video OCR recognizes only 48.3% (393 out of 814 words). We apply post-processing, which evaluates differences between recognition results with words in the dictionary, and select a word having the least differences.

4.4 Face Detection

4.4.1 Review

Face detection is locating the face(s) inside a given image. It is a necessary pre-processing process for face recognition. After the location of the face is detected, the face recognition engine can be applied and more data of the video can be retrieved and make the content of the video be more searchable.

4.4.2 Different approaches

There are many ways to detect a face in a scene - easier and harder ones. Many face detection researchers have used the idea that facial images can be characterized directly in terms of pixel intensities. These images can be characterized by probabilistic models of the set of face images, or implicitly by neural networks or other mechanisms. The parameters for these models are adjusted either automatically from example images or by hand. A few authors have taken the approach of extracting features and applying either manually or automatically generated rules for evaluating these features. Here is a list of the most common approaches in face detection:

A. Finding faces in images with controlled background:

This is the easy way out. Use images with a plain monocolour background, or use them with a predefined static background - removing the background will always give you the face boundaries.

B. Finding faces by color

For accessing color images, it may use the typical skin color to find face segments. The disadvantage is that it does not work with all kind of skin colors, and is not very robust under varying lighting conditions.

- Basic colour extraction for face detection
- Face detection in color images
- Face detection in color images using PCA
- Skin colour detection under changing lighting conditions

C. Finding faces by motion

For real-time video, it may use the fact that a face is almost always moving in reality. Just calculate the moving area, and the face can be detected. The disadvantage is what if there are other objects moving in the background?

- Basic motion detection for face finding
- Blink detection: human eyes are simultaneously blinking; this can be used to find and normalize faces

D. Using a mixture of the above

Combining several good approaches normally yields an even better result. Here are some works on that:

- A mixture of colour and 3D
- A mixture of colour and background removal

E. Finding faces in unconstrained scenes:

Amount top of them all, the most complicated thing may be in whole object recognition: Given a black and white still image, where is the face? Humans can do it, so where's the perfect algorithm that can do it, too? Here are some works on it:

- Neural Network-Based Face Detection
- Neural Nets using statistical cluster information
- Model-based Face Tracking

4.4.3 Neural Network-Based Detection

In XVIP, it has followed the project in CMU to use the Neural Network-Based Detection algorithm for face detection.
We have prepared the preprocessed image for the neural network where Informedia of CMU implements the neural network. The XVIP is connected with the Dynamic Library with this neural network to do the face detection.

A retinally connected neural network examines small windows of an image, and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. It uses a bootstrap algorithm for training the networks, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting non-face training examples, which must be chosen to span the entire space of non-face images. Comparisons with other state-of-the-art face detection systems are presented; this system has better performance in terms of detection and false-positive rates

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "non-face" images. Unlike face *recognition*, in which the classes to be discriminated are different faces, the two classes to be discriminated in face *detection* are "images containing faces" and "images not containing faces". It is easy to get a representative sample of images which contain faces, but much harder to get a representative sample of those which do not.

4.4.4 Description of the System

4.4.4.1 Input Image

The process of face detection is very time consuming; it is not efficient to perform the face detection through out the whole video. We decide to the face detection with the frame in the scene change.

After the scene change is detected and the corresponding frame is save as a bitmap and jpeg, the scene change can be used to perform the face detection. As most of the representative frame is included by the scene change, we decided to do the face detection with the scene change.

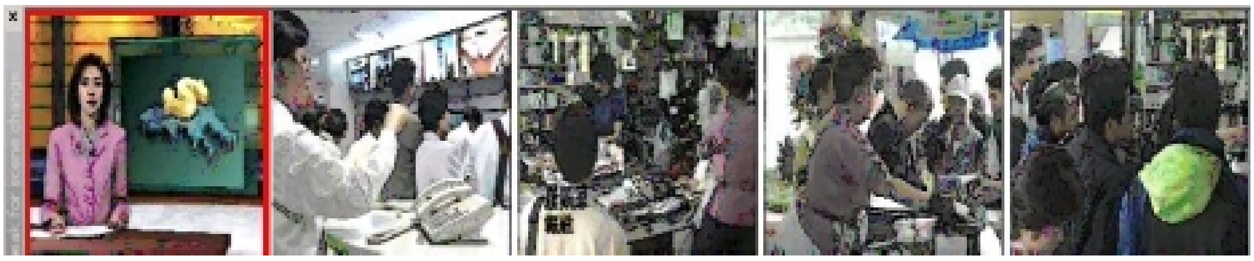


Figure4.16 Scene change result

4.4.4.2 Preprocess

The face detection system operates in two stages: it first applies filter to get the image to do the preprocessing, and then applies a set of neural network-based filters to a preprocessed image to get the outputs. The filters examine each location in the image at several scales, looking for locations that might contain a face.

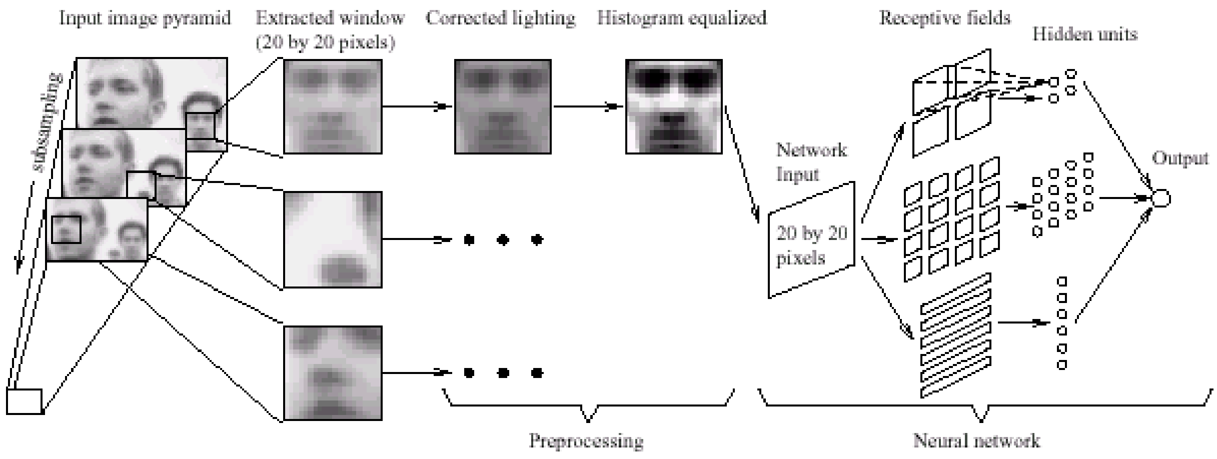


Figure 4.17 The basic algorithm used for face detection

The first component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To detect faces anywhere in the input, the filter is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly reduced in size (by subsampling), and the filter is applied at each size. This filter must have some invariance to position and scale. The amount of invariance determines the number of scales and positions at which it must be applied.

For the work presented here, we apply the filter at every pixel position in the image, and scale the image down by a factor of 1.2 for each step in the pyramid. The filtering algorithm is shown in Fig. 4.17. First, a preprocessing step is applied to a window of the image. The window is then passed through a neural network, which decides whether the window contains a face.

The preprocessing first attempts to equalize the intensity values across the window. We fit a function that varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions.

Then histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases. The preprocessing steps are shown in Fig. 4.18.

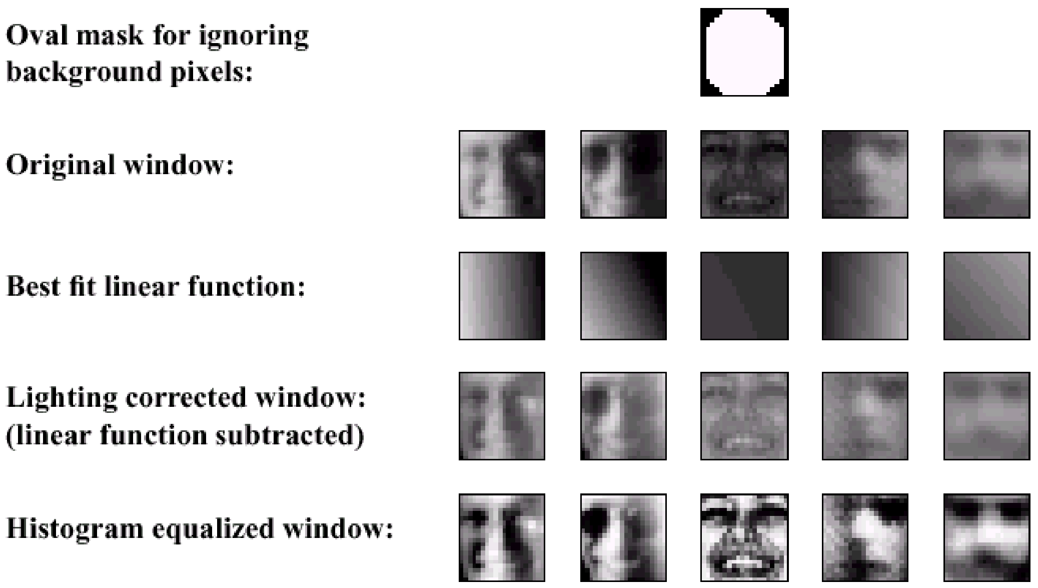

Oval mask for ignoring background pixels:
Original window:
Best fit linear function:
Lighting corrected window: (linear function subtracted)
Histogram equalized window:

Figure 4.18: The steps in preprocessing a window.

4.4.4.3 Neural Network

The preprocessed window is then passed through a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Fig. 1. There are three types of hidden units: 4 which look at 10x10 pixel subregions, 16 which look at 5x5 pixel subregions, and 6 which look at overlapping 20x5 pixel horizontal stripes of pixels. Each of these types was chosen to allow the hidden units to detect local features that might be important for face detection. In particular, the horizontal stripes allow the hidden units to detect such features as mouths or pairs of eyes; while the hidden units with square receptive fields might detect features such as individual eyes, the nose, or corners of the mouth. Although the figure shows a single hidden unit for each subregion of the input, these units can be replicated. For the experiments, which are described later, we use networks with two and three sets of these hidden units. Similar input connection patterns are commonly used in speech and character recognition tasks.

The network has a single, real-valued output, which indicates whether or not the window contains a face. Examples of output from single network are shown in Fig. 4.19. In the figure, each box represents the position and size of a window to which the neural network gave a positive response. The network has some invariance to position and scale, which results in multiple boxes around some faces.

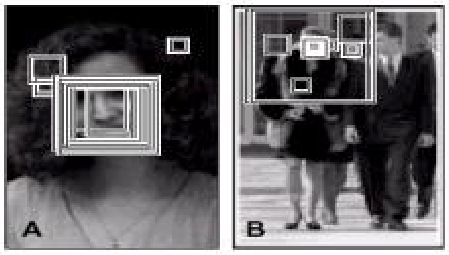


Figure 4.19 Images with all the above threshold detections indicated by boxes.

To train the neural network used in stage one to serve as an accurate filter, a large number of face and non-face images are needed. Nearly 1050 face examples were gathered from face databases at CMU, Harvard, and from the World Wide Web. The images contained faces of various sizes, orientations, positions, and intensities. The eyes, tip of nose, and corners and center of the mouth of each face were labeled manually. These points were used to normalize each face to the same scale, orientation, and position, as follows:

1. Initialize F, a vector that will be the average positions of each labeled feature over all the faces, with the feature locations in the first face .
2. The feature coordinates in are rotated, translated, and scaled, so that the average locations of the eyes will appear at predetermined locations in a 20x20 pixel window.
3. For each face i, compute the best rotation, translation, and scaling to align the faces feature the average feature locations F. Such transformations can be written as a linear function of their parameters. Thus, we can write a system of linear equations mapping the features from to F. The least squares solution to this over-constrained system yields the parameters for the best alignment transformation. Call the aligned feature locations
4. Update by averaging the aligned feature locations for each face i.
5. Go to step 2.

The alignment algorithm converges within five iterations, yielding for each face a function which maps that face to a 20x20 pixel window. Fifteen face examples are generated for the training set from each original image, by randomly rotating the images (about their center points) up to scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20x20 window in the set is then preprocessed (by applying lighting correction and histogram equalization). A few example images are shown in Fig. 4.20.

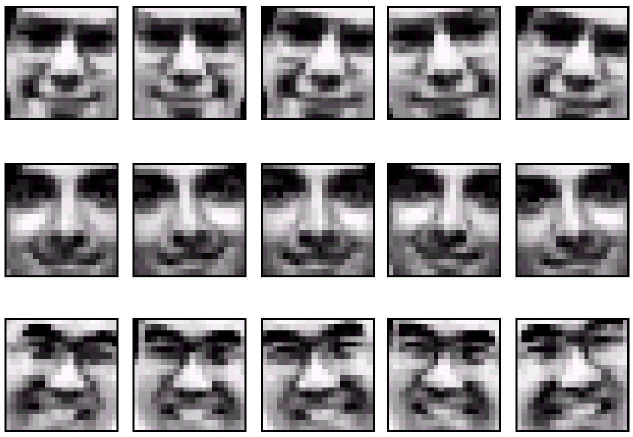


Figure 4.20 Example face images (the authors), randomly mirrored, rotated, translated, and scaled by small amounts.

The randomization gives the filter invariance to translations of less than a pixel and scalings of 20%. Larger changes in translation and scale are dealt with by applying the filter at every pixel position in an image pyramid, in which the images are scaled by factors of 1.2.

Practically any image can serve as a nonface example because the space of nonface images is much larger than the space of face images. However, collecting a “representative” set of nonfaces is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner, adapted from:

1. Create an initial set of nonface images by generating 1000 random images. Apply the pre-processing steps to each of these images.
2. Train a neural network to produce an output of 1 for the face examples, and -1 for the nonface examples. The training algorithm is standard error backpropogation with momentum [8]. On the first iteration of this loop, the network’s weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.

3. Run the system on an image of scenery *which contains no face*s. Collect subimages in whichthe network incorrectly identifies a face
4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

Some examples of nonfaces are collected during training. Note that some of the examples resemble faces, although they are not very close to the positive examples shown in Fig. 4.3.5. The presence of these examples forces the neural network to learn the precise boundary between face and nonface images. It used 120 images of scenery for collecting negative examples in the bootstrap manner described above. A typical training run selects approximately 8000 nonface images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images. A similar training algorithm was described in [25], where at each iteration an entirely new network was trained with the examples on which the previous networks. Informedia of CMU implements this neural network and provides the dll of the network as open.This dll is connected to XVIP for face detection.

4.4.5 Face detection in XVIP

With scene change as the input image, XVIP preprocess the image and pass to the neural network described above.

The face detection modality is added to the XVIP, the following picture shown the result of the detection. The face is enclosed by a blue square.

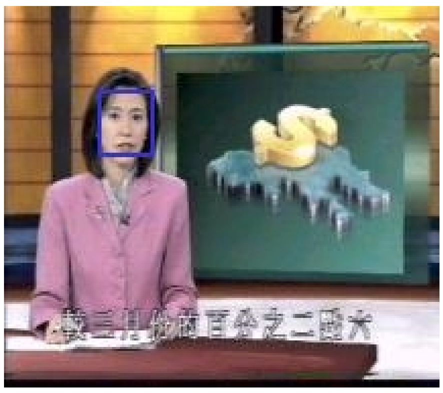


Figure 4.21 Result of the Face detection

4.4.6 Further Process

After the location of the face(s) is detected, other process can be done to enrich the data, such:

- Facial Expression Analysis
- Face Recognition

It makes XML to become more searchable and more information can be found. However these techniques are not to be discussed in the project.

4.5 Speech Recognition

4.5.1 Review

Speech recognition technology can make any spoken data useful for library indexing and retrieval by extracting keyword from the transcript. To make our XVIP more complete, including both video and audio information processing, speech recognition is add to our XVIP as a new modality. The information extracted from speech is also used to enrich the XML. This new modality is named as "Transcript" in the XML.

Speech recognition, or speech-to-text, involves capturing and digitizing the sound waves, converting them to basic language units or phonemes, constructing words from phonemes, and contextually analyzing the words to ensure correct spelling for words that sound alike (such as write and right). The figure 4.22 below illustrates this high-level description of the process

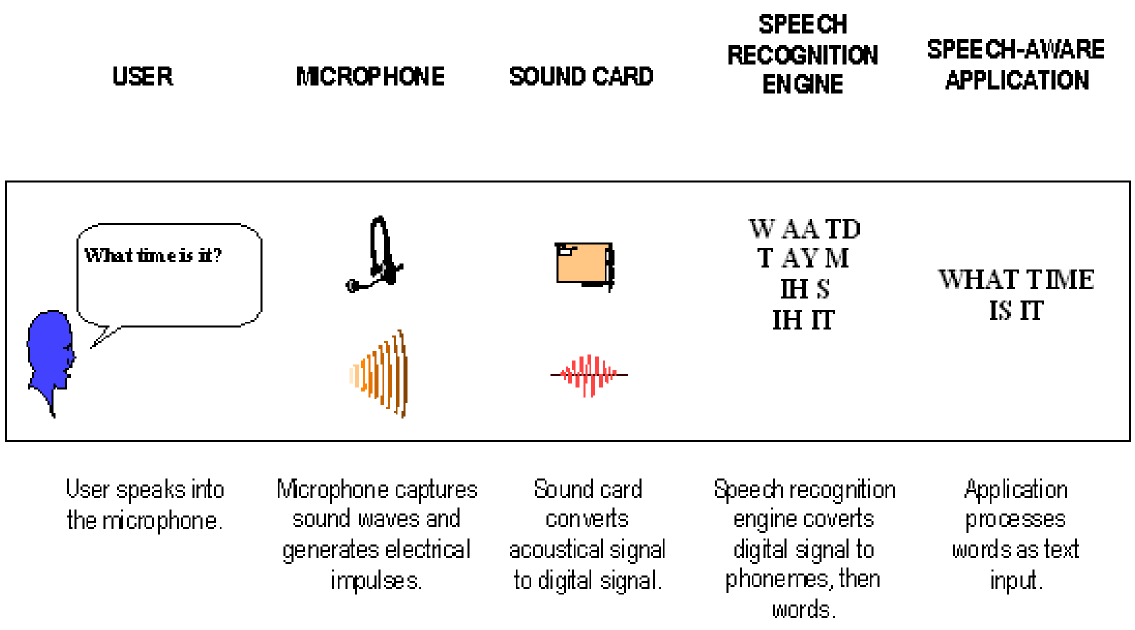

USER
MICROPHONE
SOUND CARD
SPEECH RECOGNITION ENGINE
SPEECH-AWARE APPLICATION
What time is it?
W AA TD
T AY M
IH S
IH IT
WHAT TIME IS IT
User speaks into the microphone.
Microphone captures sound waves and generates electrical impulses.
Sound card converts acoustical signal to digital signal.
Speech recognition engine coverts digital signal to phonemes, then words.
Application processes words as text input.

Figure 4.22 Overview of speech recognition

By applying speech recognition together with natural language processing, information retrieval and image analysis, an interface has been produced that helps users locate the information they want and navigate or browse the digital video library more effectively.

Speech recognition generated transcripts can make multimedia material searchable. The XVIP emphasizes the integration of speech recognition, image processing and information retrieval to compensate for deficiencies in these individual technologies.

4.5.2 Speech Recognition Process

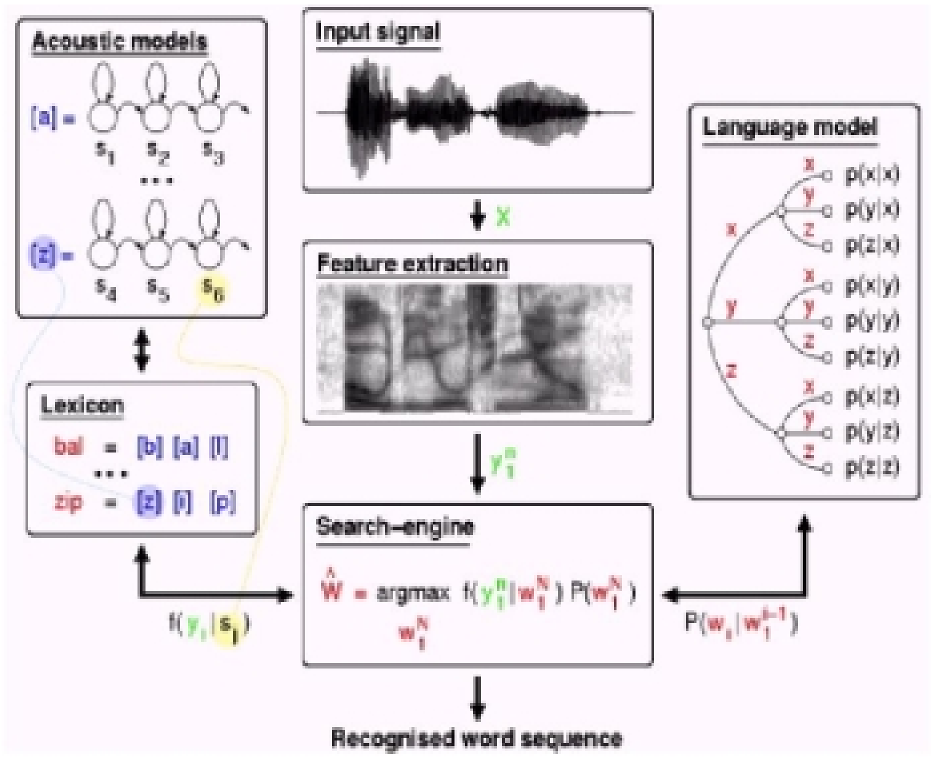


Figure 4.23 Speech recognition Process

4.5.2.1 Process Diagram

The speech recognition process can be divided into different components. Which is illustrated in figure 4.23 above.

4.5.2.2 Process Explanation

- **Input Signal**

  First, we get the speaker's voice from an input device and save it as speech signals. The commonly used input device is a microphone. Note that the quality of the input device cans inference the accuracy of the SR system very much. The came applies to acoustic environment. For instance, additive noise, room reverberation, microphone position and the type of microphone can all relate to this part of process.

  For XVIP, the input signal is a wav file, which is extracted for the original mpeg file.

- **Feature Extraction**

  The next block, which shows the feature extraction subsystem, is tried to deal with the problem created in the first part, as well as deriving acoustic representations. The two aims are separate classes of speech sounds, such as music and speech, and effectively suppress irrelevant sources of variation.

- **Search Engine**

  The search engine block is the core part of speech recognition process. In a typical ASR system, a representation of speech, such as spectral representation, is computed over successive intervals, e.g., 100 times per second. These representations or speech frames are then compared with the spectra frames, which were used for training. Using some measurement of distance of similarity does it.

  Each of these comparisons can be regarded as a local match. The global match is a search for the best sequence of words, in the sense that it is the best match to the data. And it is determined by integrating many local matches. The local match does not usually produce a single hard choice of the closet speech class, but rather a group of distance or probabilities corresponding to possible sounds. These are then used as part of a global search or decoding to find an approximation to the closest sequence of speech classes, or ideally to the most likely sequence of words.

- **Acoustic Model**

  The acoustic model is the recognition system's model for the pronunciation of words, crucial to translating the sounds of speech to text. In reality, the type of speaker model used by the recognition engine greatly affects the type of acoustic model by the recognition system to convert the vocalized words to data for the language model to be applied.

  There are a wide variety of methods to build the pattern models, the three major types are:

  a. Vector Quantization
  b. Hidden Markov Models (HMM)
  c. Neutral Networks

  As XVIP not aims to focus on the recognition system, it is not going to present an in-depth discussion about the above methods here.

- **Language Model & Lexicon**

  Language model is another major component of speech recognition process. It provides the knowledge source for the engine and helps to predict the next word.

  The first component of the language model is the lexicon, which consists of the vocabulary. The vocabulary, as explained before, contains all of the possible words in the voice system that may be encountered.

The second component of the language model is the grammar. It defined the structure and format of the text allowed at any point in the utterance. Without grammar, every word in the lexicon would have an equal likelihood of occurrence at any point within the utterance, requiring algorithm that are too complex, making continuous speech impossible. These systems use the grammar to constrain the word choice at any point, which is more efficient.

4.5.3 IBM ViaVoice

4.5.3.1 Introduction to ViaVoice

For speech recognition, we have chosen to use commercial product ViaVoice by IBM to complete this part of XVIP.

Figure 4.24 ViaVoice for Windows Release 8 Rebate Program

In 1994, IBM was the first company to commercialize a dictation system based on speech recognition. Science then IBM devoted to develop the most advanced technology in this field.

ViaVoice is the speech recognition system that developed by IBM and it supports 13 different languages. In September 1997, Via Voice Chinese version came into being and it draw much attention to the people who concerned. It successfully solves the problems like there are many characters with the same pronunciation and different meaning in Chinese, Chinese language has different tones, etc.

4.5.3.2 System Characteristics

Science the first Chinese version of ViaVoice speech recognition system was born, IBM devoted to increase the accuracy of the system. In 1999, they enhanced the system to be a user independent, no vocabulary constraint, and continuous speech recognition system with high recognition accuracy. And now they have three different versions including Mandarin, Taiwanese and Cantonese.

4.5.3.3 ViaVoice Native Architecture Overview

The heart of a speech recognition system is known as the speech recognition engine. The speech recognition engine recognizes speech input and translates it into test that an application can understand. Application can access the engine through a speech recognition API. For ViaVoice, this API is known as Speech Manager API (SMAPI). SMAPI is a conversational API, meaning that the API is defined as apart of resource; in ViaVoice, SMAPI is defined as part of the speech engine. With an API, speech becomes a resource to all applications

4.5.3.4 ViaVoice Speech Engine Architecture

The speech engine has a rather complex tank to handle, that is, taking the raw audio input and translating it to recognizes text that an application understands.

The Audio input source module encapsulates the methods used by the engine to retrieve the audio input stream. By default, the engine retrieves its audio input from the standard microphone input device in the system, a developer can write a custom audio library so that the input of the engine would be a custom piece of hardware.

The acoustic processor takes raw audio data and converts it to the appropriate format for use. The acoustic processor consists of two components: the signal processor and the labeler.

In the ViaVoice engine, audio input picked up by the microphone in analyzed by the signal processor, this raw audio data is captured at 22 kHz by default, but 11kHz and 8kHz sampling is also supported. It contains both speech data and background noise.

4.5.3.5 Application Programming Interfaces

ViaVoice SDK provides several programming interfaces that developers can use to incorporate speech into their applications. This guide describes the IBM SMAPI and Grammar Compiler API.

- Speech Manager APIs (SMAPI)
  IBM speech recognition engine APIs.
- SMAPI Grammar Compiler APIs
  APIs used to compile grammars used by the speech recognition engine.

SMAPI

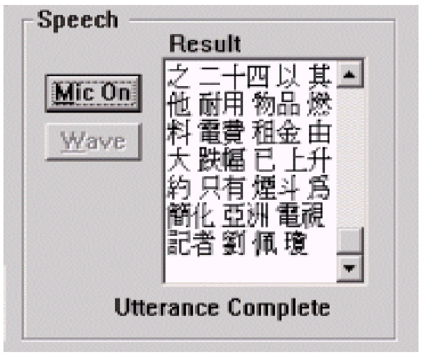
There is significantly more function in the ViaVoice engine beyond raw recognition of spoken words, including dynamic vocabulary handling, database functions to query and select installed users, languages, and domains, and the ability to add new words to the user's vocabulary. SMAPI supports:

- Verifying the API version
- Establishing a database session to query system parameters
- Establishing a recognition session
- Setting up vocabularies
- Setting speech engine parameters
- Processing speech input
- Adding new words to the user's vocabulary
- Handling errors
- Disconnecting from the speech engine
- Closing a speech session

The SMAPI is provided as a library, which is linked into an application. The engine is a separate executable. The ViaVoice architecture supports many speech applications through a single engine, connected to one microphone. SMAPI was derived from earlier IBM speech products so that all functions used by IBM applications are available to the developer.

4.5.4 Speech Recognition in XVIP

The error rate of the speech recognition is quite high, greater then 50%. The performance can be improved by training the ViaVoice.

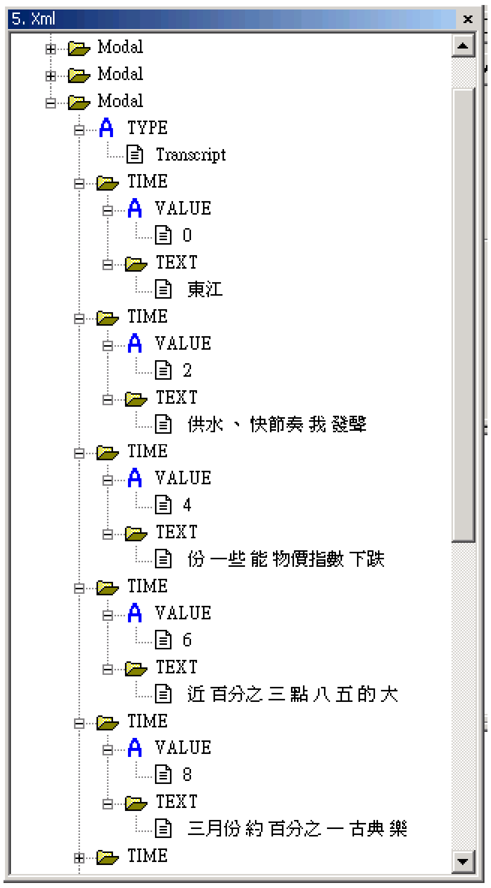


之 二十四 以 其 他 耐用 物品 燃 料 電費 租金 由 大 跌幅 已 上升 約 只有 煙斗 爲 簡化 亞洲 電視 記者 劉 佩 瓊

Utterance Complete

Figure 4.25 interface of the speech recognition engine in XVIP

After text is recognized, words are added to the XML. One more modality is added to the XML, named as "Transcript". The text is added to the XML according to the time. As the error rate of the speech recognition is quite large. The text can be insert or deleted inside the editor, shown in figure 4.26. However, we believe the accuracy of speech recognition engine can be improved by training.

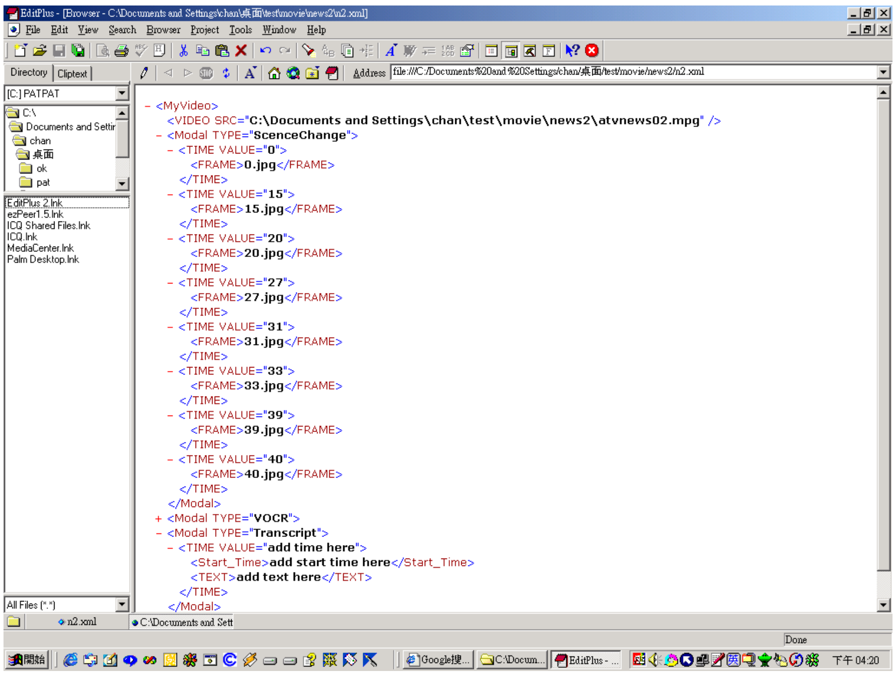


```
5. Xml
+ Modal
+ Modal
- Modal
    - TYPE
        Transcript
    - TIME
        - VALUE
            0
        - TEXT
            東江
    - TIME
        - VALUE
            2
        - TEXT
            供水 、 快節奏 我 發擘
    - TIME
        - VALUE
            4
        - TEXT
            份 一些 能 物價指數 下跌
    - TIME
        - VALUE
            6
        - TEXT
            近 百分之 三 點 八 五 的 大
    - TIME
        - VALUE
            8
        - TEXT
            三月份 約 百分之 一 古典 樂
    + TIME
```

Figure 4.26: interface of the editor

5. XML

For existing digital video library, some of them use relational database system to store the data extracted. For our extracting tool, we choose to use XML for storing the extracted data instead of following them. You may wonder why XML is chosen instead of others, you may find the answer in the following section.



```
- <MyVideo>
    <VIDEO SRC="C:\Documents and Settings\chan\test\movie\news2\atvnews02.mpg" />
  - <Modal TYPE="ScenceChange">
    - <TIME VALUE="0">
        <FRAME>0.jpg</FRAME>
      </TIME>
    - <TIME VALUE="15">
        <FRAME>15.jpg</FRAME>
      </TIME>
    - <TIME VALUE="20">
        <FRAME>20.jpg</FRAME>
      </TIME>
    - <TIME VALUE="27">
        <FRAME>27.jpg</FRAME>
      </TIME>
    - <TIME VALUE="31">
        <FRAME>31.jpg</FRAME>
      </TIME>
    - <TIME VALUE="33">
        <FRAME>33.jpg</FRAME>
      </TIME>
    - <TIME VALUE="39">
        <FRAME>39.jpg</FRAME>
      </TIME>
    - <TIME VALUE="40">
        <FRAME>40.jpg</FRAME>
      </TIME>
    </Modal>
  + <Modal TYPE="VOCR">
  - <Modal TYPE="Transcript">
    - <TIME VALUE="add time here">
        <Start_Time>add start time here</Start_Time>
        <TEXT>add text here</TEXT>
      </TIME>
    </Modal>
```

Figure 5.1 XML view in a browser

5.1 Introduction

The Extensible Markup Language (XML), HTML's likely successor for capturing much Web content, is easier-to-use subset of SGML (Standard Generalized Markup Language) and a text-based markup language. XML is receiving a great deal of attention from the computing and Internet communication. XML is a meta-language that permits a set of users to create its own mark-up language for describing the contents of Web documents. The use of XML has many advantages, since an XML file contain not only data (like HTML) but also metadata – structural and semantic information about that data. In this sense, an XML document is very similar to a database, and it in general will look like one big database. Life of an XML Document is shown as follow:

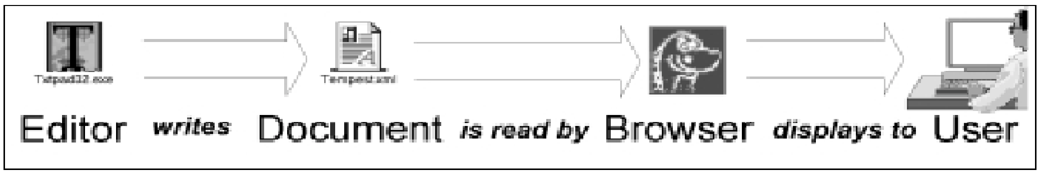


Figure 5.2 Life of an XML Document

5.2 How its work

XML is a markup meta-language that can be used to define a set of languages that represent structured data in text-based documents. Any set of users, such as a group of companies within an industry, can develop its own XML-based language with its own set of markup tags. XML lets information publishers invent their own tags for particular applications or work with other organizations to define shared sets of tags that promote interoperability and that clearly separate content and presentation. This gives users considerable flexibility and functionality.

The XML tags provide metadata, which is information about the data a document contains. For example, in online invoice, users could tag pieces of data as “customer”, “product”, or others. System on different platforms could thus understand what the data means

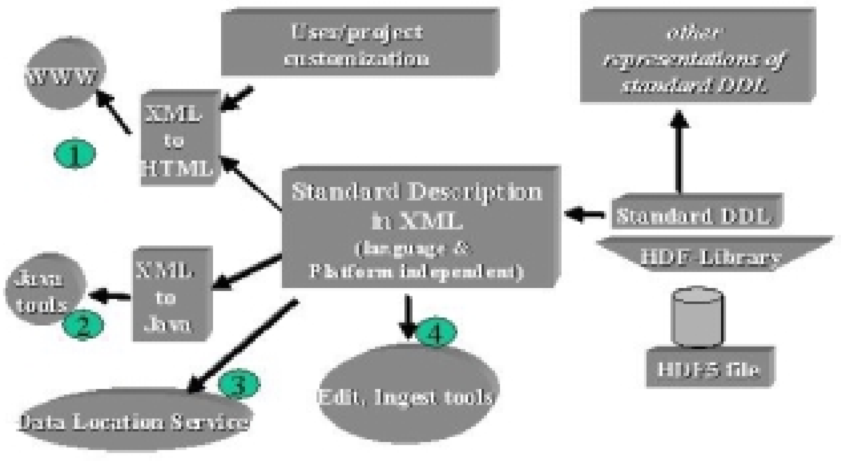

User/project customization
other representations of standard DDL
WWW
XML to HTML
1
Standard Description in XML (language & Platform independent)
Standard DDL
HDF Library
Java tools
2
XML to Java
4
HDF5 file
3
Data Location Service
Edit, Ingest tools

Figure 5.3 Roles for XML

Above is some of the roles for XML: transformed to HTML for standard Web browsers (1), ingested into Java and other tools (2), as input to Data Location Services (Catalogs) (3), and as input for ingest, editing, and validation tools (4). We aim to support and enable as many of these uses as possible.

5.3 Advantages of Using XML

For this markup language, there are many advantages:

- *Platform and system independent*

  The first advantage is that XML is platform and system independent. XML let users, devices, and applications on different platforms communicate across the Internet, and permit organizations to integrate different data types within their system. It works as well on one computer as it does on another.

- *Essentially clear*

  The people responsible for designing XML wanted it to be clear as possible so there are no optional tag or minimization.

- *Create your own tag*

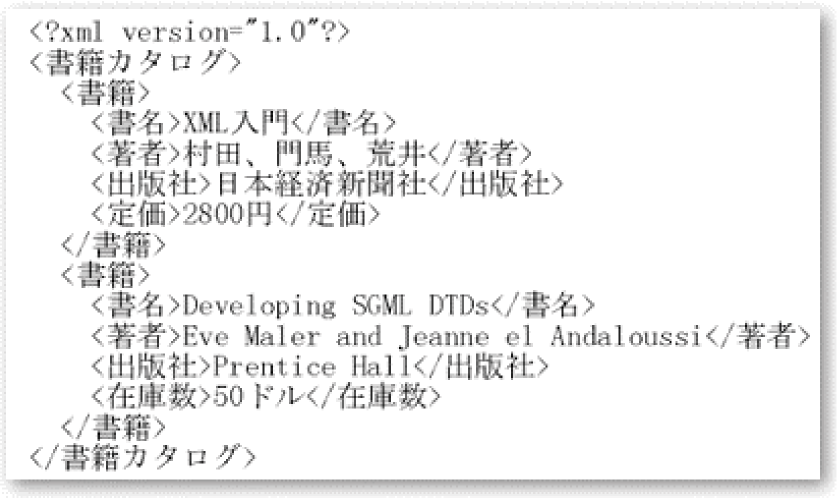
  XML lets information publishers invent their own tags for particular applications or work with other organizations to define shared sets of tags that promote interoperability and that clearly separate content and presentation. Any XML-aware software will be able to work with other custom application.

- *Easy to search*

  Searching on the web at the moment can be the most frustrating task. A search can result in hundreds of matches, but not pinpoint the content for which is searching. This is because today's search engines have no way of differentiating content. When XML is widely adopted, search engines will be able to search based on the content of particular tags.

- *Adopt Unicode*

  XML has adopted ISO 10646, well known as Unicode. This standard is used as a framework to encode characters and it will support most languages. Since XML follows this standard, it is possible to use any language, include Chinese, for the markup. It can contain Chinese names for tags and any XML-aware software will still be able to understand it. XML dose not force people to use English for their coding. Internationalization is built right in.



```
<?xml version="1.0"?>
<書籍カタログ>
  <書籍>
    <書名>XML入門</書名>
    <著者>村田、門馬、荒井</著者>
    <出版社>日本経済新聞社</出版社>
    <定価>2800円</定価>
  </書籍>
  <書籍>
    <書名>Developing SGML DTDs</書名>
    <著者>Eve Maler and Jeanne el Andaloussi</著者>
    <出版社>Prentice Hall</出版社>
    <在庫数>50ドル</在庫数>
  </書籍>
</書籍カタログ>
```

Figure 5.4 XML offers global language support based on the broad compatibility of Unicode

- *Universal format*
  XML is universal format for exchanging information between software components that is legible to both computers and human beings. It is powerful as everyone could speak the same language and communicate together. XML creates a lingua franca for the computer world that has been missing for the longest time. XML can treat as a tool for engendering utopia among computer users.

- *Display in different ways*
  XML allows users to download a document and then display it in different way. Because the style sheets are separate, not written right into the document, the document can be display in a number of different. Extensible style-sheet Language (XSL) expresses rules that indicate how to transform an XML document to a presentation format such as HTML or PDF, or to an alternate representation of the content such as an XML document with different DTD.

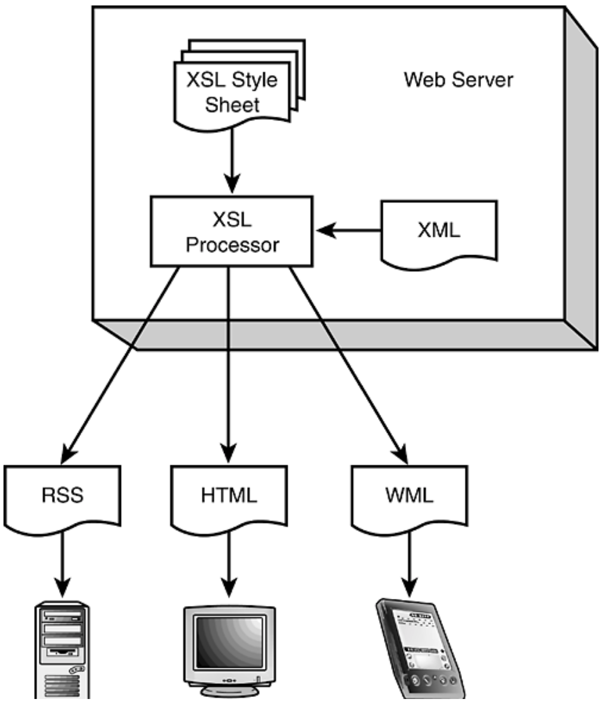
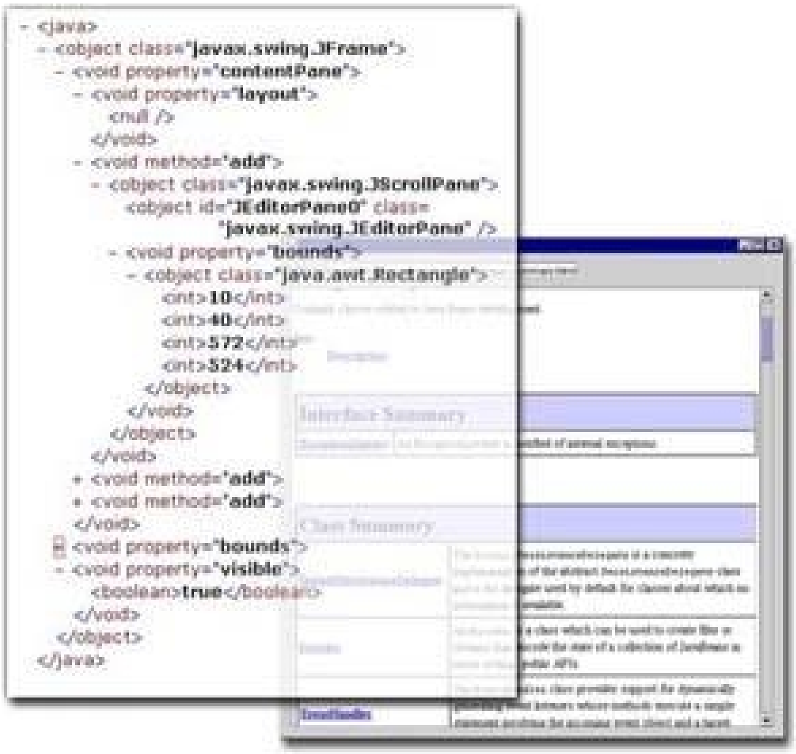


Figure 5.5 (a)



```
- <java>
 - <object class="javax.swing.JFrame">
  - <void property="contentPane">
   - <void property="layout">
      <null />
     </void>
   - <void method="add">
    - <object class="javax.swing.JScrollPane">
       <object id="JEditorPane0" class=
              "javax.swing.JEditorPane" />
     - <void property="bounds">
      - <object class="java.awt.Rectangle">
          <int>10</int>
          <int>40</int>
          <int>572</int>
          <int>524</int>
        </object>
       </void>
      </object>
     </void>
   + <void method="add">
   + <void method="add">
    </void>
  + <void property="bounds">
  - <void property="visible">
     <boolean>true</boolean>
    </void>
   </object>
 </java>
```

Figure 5.5 (b) XML can be displayed in different format

➢ *Other markup language*

As XML let user to create its own tag, there are dialects of XML. They used by different industries and disciplines and define commonly used tags to be specific and useful to that community. Of particular interest to technical communicators is Synchronized Multimedia Integration Language (SMIL). This is designed to optimize web-casts of media is a flexible manner and includes streaming audio, streaming video and static images. The goal is to revolutionize educational and entertainment on the web (which is our goal in the second phase of this final year project). There are other dialects, for example, MathML that is useful to mathematician because it can capture the meaning of equations, rather than how it will look on a screen. Chemical Markup Language (CML) help chemists to render molecular structure of compounds.

➢ *Advantage of XML over HTML*

Feature	HTML	XML
Extensibility	Fixed set of tags	Extensible set of tags
Presentation/ content	Tags for presentation only	Tag describe data content
View	Single presentation of each document	Multiple view of the same document
Orientation	Document orientation only	Support for document plus extensive intra-structure for exchange and validation of structured data
Search / query	Search only	Search plus filed-sensitive queries and later update

5.4 Why we choose XML

As XML has so many advantages, we choose XML for storing the data that extracted for videos. The major factors that we think XML is suitable for our tool instead of other database system are:

- XML 's flexibility lets it serve as a meta-language for defining other markup language specialized for specific contexts.

  For each modal dimension, it manages its own XML DTD or XML schema. The DTD would be extended, if knowledge enrichment or cross-referencing were applied. In the above example, the geographical name process will extend the XML as follow,

  ```
  <modal name="VOCD">
        <text start="0s" end="5s">
                  Hong Kong is a beautiful place.
                  <geoname>Hong Kong</geoname>
       </text>
        <text start="5s" end="7.5s">You can find</text>
        <text start="7.5s" end="9s">the best clothing
        here.</text>
  </modal>
  ```

  By extending the XML DTD, the new tag that represents new modality information is added into the original XML file. The new modality, in this example the geoname, can be treated as an extended-modality of the parent modal text

- XML can let us create any tag to describe the information the extracted from videos. It may the read easy the read and understand what information that can be extracted from videos.

- XML can describe the information, so information in the XML can be searched easily and accurately, only the related result is show for the searcher. So, the information extracted can be search easily so that the information becomes useful and meaningful.

- XML is scalable; we can add more information without affecting others. As our tool is a multi-modal tool, more components will be added to the tool if other extracting information techniques is developed. For two independent modalities, the XML part will be combined as follow to form the final XML document.

```
<modal name="VOCR">
      <text start="0s" end="5s">
                Hong Kong is a beautiful place
                <geoname>Hong Kong</geoname>
     </text>
      <text start="5s" end="7.5s">You can find</text>
      <text start="7.5s" end="9s">the best clothing
      here.</text>
</modal>
<modal name="scene change">
      <scene start="0s" end="20s" src="/101.jpg"/>
      <scene start="20s" end="33s" src="/102.jpg"/>
</modal>
```

After choosing XML as a format for storing the data extracted, it is time to design how the schema is and how to read and present is our tool.

5.5 How to design schema and our format

The process of designing an XML document for the videos' extracted information necessarily includes several element tags as schema representations of the document content and structure. This process starts with choosing a vocabulary, i.e. words and phrases that are able to describe all the required aspects of the extracted video information content and therefore can be used as tag name. For example, video is chosen for representing the video information, time for representing the time that the information is extracted from the video, frame for the scene change that is detected, and text is the context for describing the information or the transcript.

The next step is to show relationship between vocabulary entries, or in other words, to create ontology of the video information domain. A common way of doing this is using UML (unfilled markup language), its diagram and notations.

The UML-based ontology of extracted video information shown as follow:

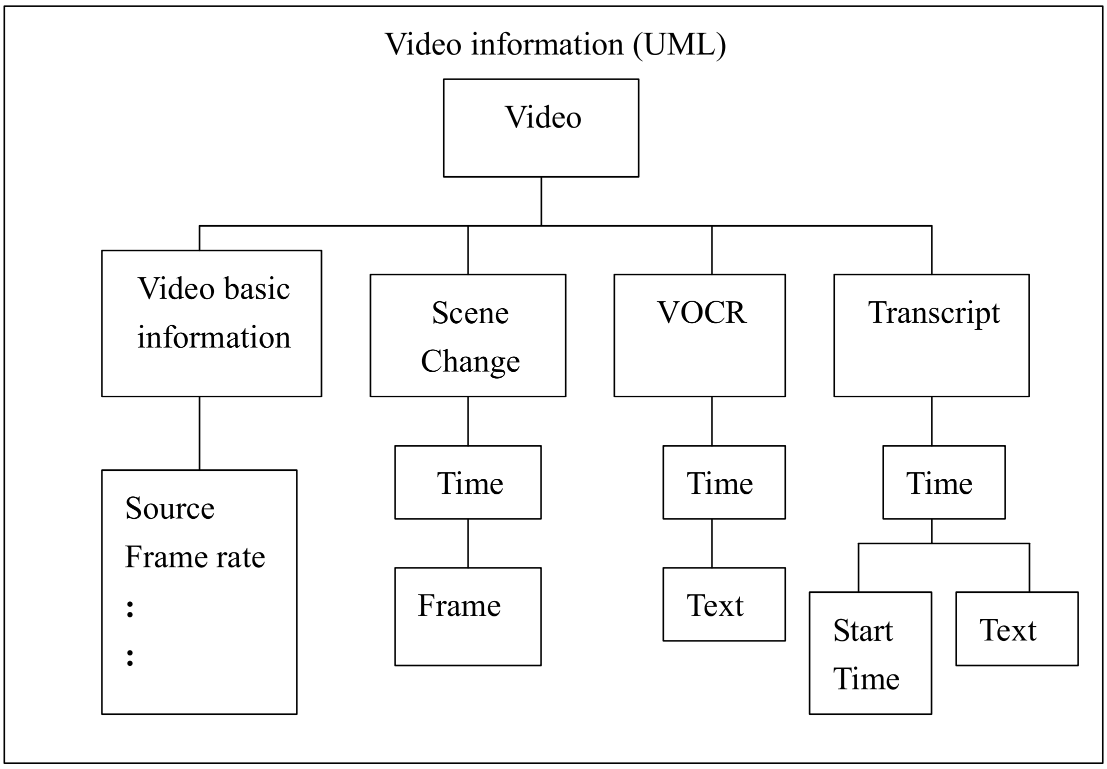


Figure 5.6 UML-based ontology

The UML-based ontology is the initial native visualization of the content of the XML documents to be created. These diagrams are much more human-oriented than any other native visualization. It is a thorough conceptual model that captures a human view of the domain, its object (elements), their properties (attributes) and relations. Therefore, this visualization may have a very high heuristic potential for user interfaces that enable visual interaction with XML ontology.

The XML schema of video information present in diagrammatic form:

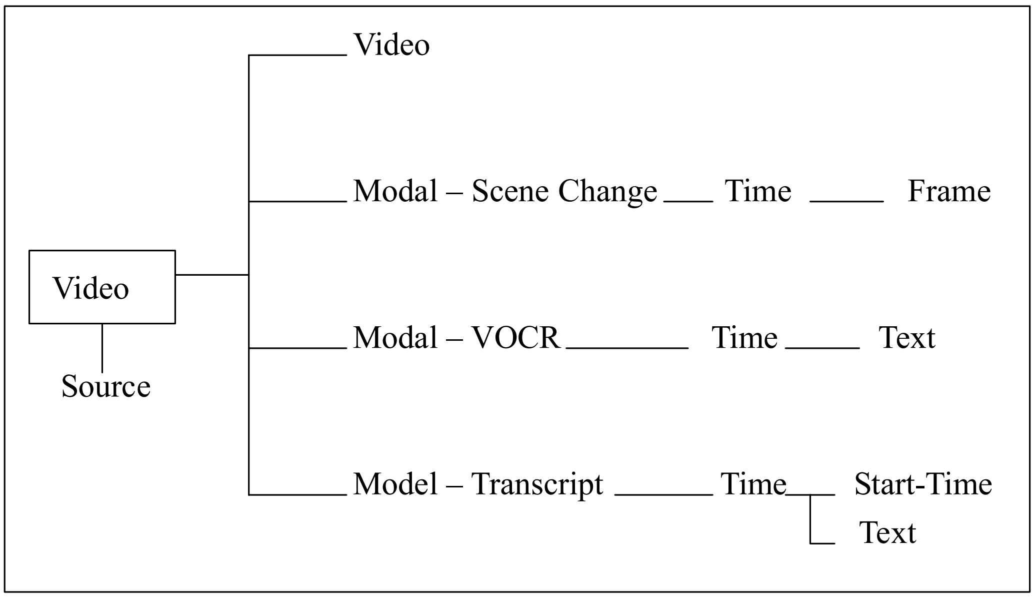


Figure 5.7 XML scheme

It looks very similar to the above ontological model. In principle, a UML model can be converted into an XML schema automatically, although inn the case of complex XML documents it may be difficult to achieve without human intellectual help. In comparison to a UML ontology, an components that are necessary for computers but are not so important for human understanding of the domain structure.

Once an XML schema is constructed, it can be used as a template for creating an unlimited number of XML document. The design process is finished, and the next stage of reading and displaying the created documents lies ahead.

5.6 Parser

An XML document dose not do anything by itself, it must be combined with an application program that dose something useful with it. A parser is an interface between an XML document and the application program that uses it. XML parser is a special program for reading XML file. The parser reads the XML documents and provides application programs with access to the documents' internal structure and content. It can be a standalone parser or an integral part of a Web browser.

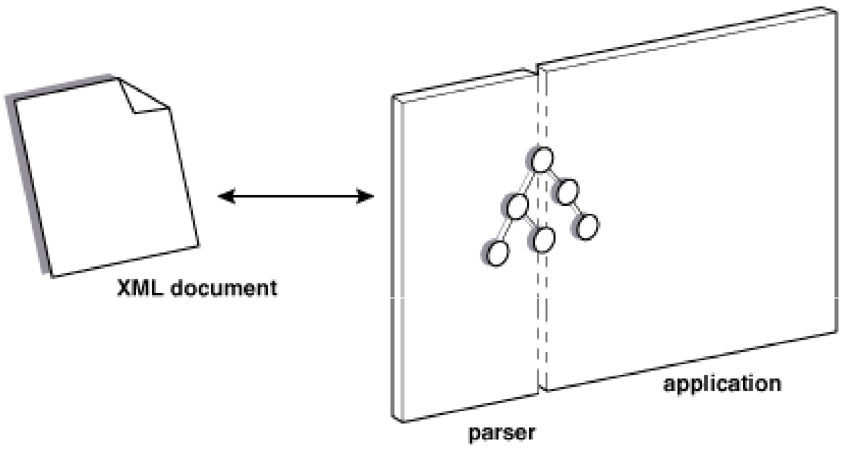


Figure 5.7 Relationship between the XML and its parser

Two types of XML parser application programming interfaces (APIs) are available:

- Document Object Model (DOM), which is a tree-structure-based API issued as a W3C recommendation
- Simple API for XML (SAX), which is an event-driven API developed by the members of the XML-DEV mailing list.

5.6.1 DOM

For our tool, we choose DOM. DOM is a platform and language-neutral interface that lets programs and scripts dynamically access and update the content, structure, and style of the documents. The DOM model represents an XML document as a tree whose nodes are elements, text, and so on. As the following figure,

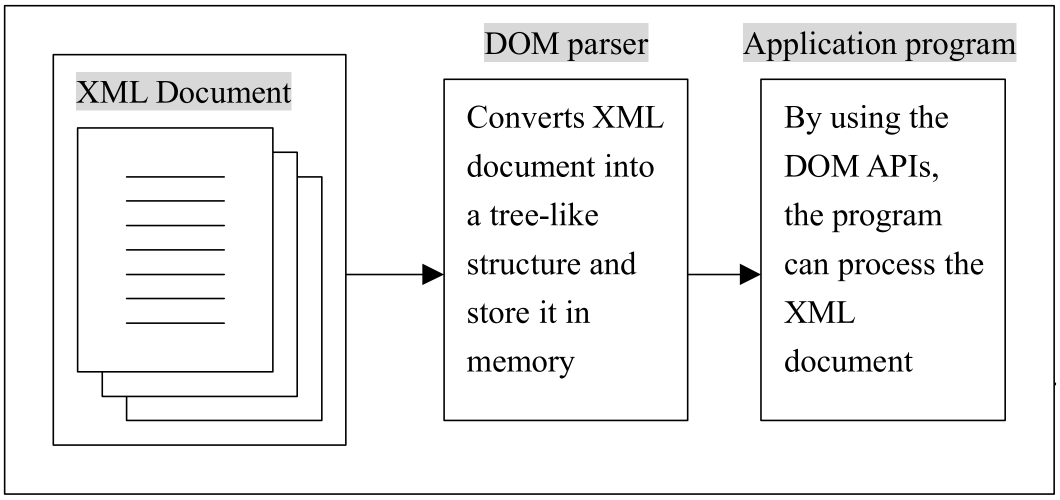

XML Document
DOM parser
Converts XML document into a tree-like structure and store it in memory
Application program
By using the DOM APIs, the program can process the XML document

Figure 5.8 DOM model

DOM provides a set of APIs to access and manipulate these nodes in the DOM tree. A DOM-based XML processor creates the entries structure of an XML document in memory. DOM is best suited for

- Structurally modifying or dynamically creating an XMLL document, just as our tool need to create the XML and then modify it dynamically.
- Sharing the document in memory with other application.

Sample XML

```
<chapter id="cmds">
   <chaptitle>FileCab</chaptitle>
   <para>This chapter describes the commands that manage
   the <tm>FileCab</tm>inet application.</para>
 </chapter>
```

Source document viewed as a tree This is how the source XML document may be represented after it has been parsed

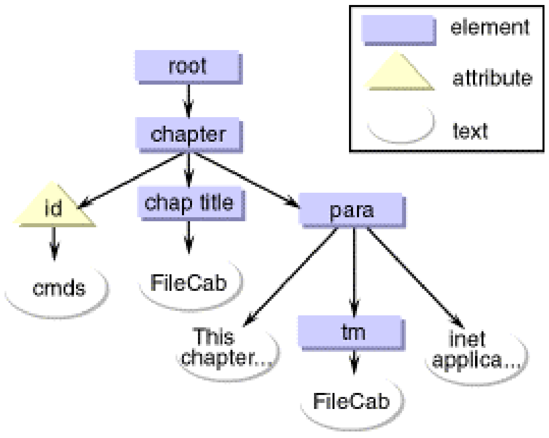


Figure 5.9 XML represent as a tree

5.6.2 SAX

An XML processor with SAX does not create a data structure. Instead, it scans an input XML document and generates events, such as an element start or end. The application programs implement handlers that receive these events and process them appropriately. SAX is best suit for

- Handling large documents that do not fit in memory
- Extracting the contents of the specific element

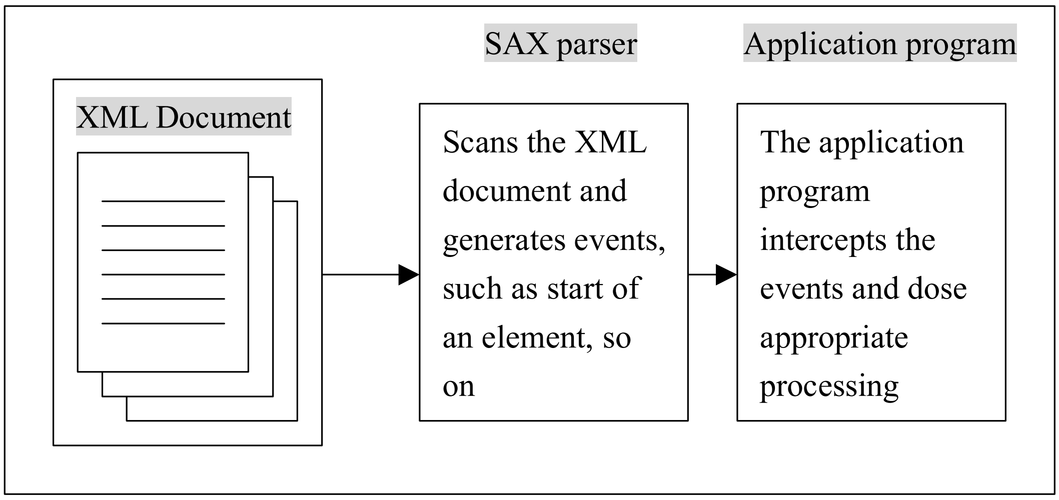

SAX parser
Application program
XML Document
Scans the XML document and generates events, such as start of an element, so on
The application program intercepts the events and dose appropriate processing

Figure 5.10 SAX model

As we choose DOM as the type of parser for the tool, we found some example from the web site. We found a DOM parser called “CXML” from the site – “Code-Project” which is written in Visual C++ and easy to use. It does not need to link to any library when using it in the Visual C++. Thus, we choose it as our parser.

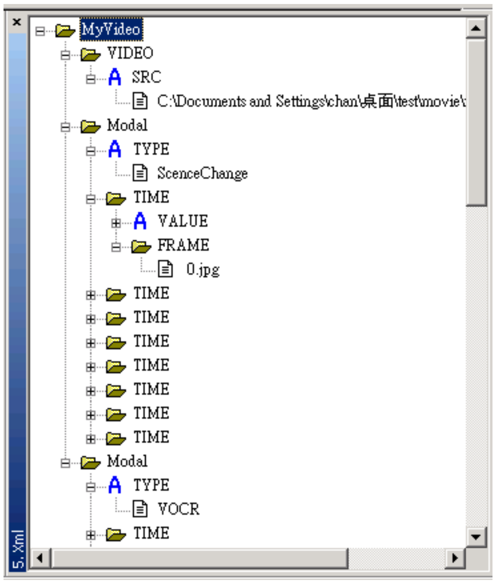
5.7 How to present XML in XVIP

Various native visualizations, a novel and efficient representation of the XML document ontology have been developed – the Generalized Document Object Model Tree Interface. The interface represents XML metadata and their structure in a comprehensive, intelligible and compact way.

This interface is based on the XML DOM visualization, generalized and enhanced to make it move suitable foe effective visual interaction with meta-data. Basically, it is a part of a DOM tree where all logical and structural components of an XML document are represented as nodes. The main difference from a DOM tree is that our model borrows from a standard DOM only its recurring metadata structure.

The Tree model becomes very similar to an XML schema. But, at the same time, the Tree modality inherits the structural organization of a DOM rather than of a schema.

All types of metadata (elements, attributes and text) are represented as nodes that show element/attribute names or the text content and their relative places within the XML document structure. Each attribute has the status of a special child element of its parent element. The bitmap "🗀" in front of a node name is to indicate an element. The bitmap "A" in front of a node name is to indicate difference between an attribute and an element. The bitmap "🗎" in front of a node name is to indicate a text.



```
⊟ MyVideo
  ⊟ VIDEO
    ⊟ A SRC
        C:\Documents and Settings\chan\桌面\test\movie\
  ⊟ Modal
    ⊟ A TYPE
        ScenceChange
    ⊟ TIME
      ⊞ A VALUE
      ⊟ FRAME
          0.jpg
    ⊞ TIME
    ⊞ TIME
    ⊞ TIME
    ⊞ TIME
    ⊞ TIME
    ⊞ TIME
    ⊞ TIME
  ⊟ Modal
    ⊟ A TYPE
        VOCR
    ⊞ TIME
```

Figure 5.11 XML represent as a tree in our tool

The feature of the Tree view that make it an efficient and promising visualization of XML meta-data:

- The ontology of an XML document is represented in a comprehensive form is much more compact than an XML schema.
- It is easy to read and understand because of its indentation-based form that match the recurring structural pattern of an XML document and also because of the use of the document tag names.
- User cannot modify the structure of the XML as the structure of the tree is fixed and the user cannot only modify the content of the text. So, this prevents the users change the XML structure.

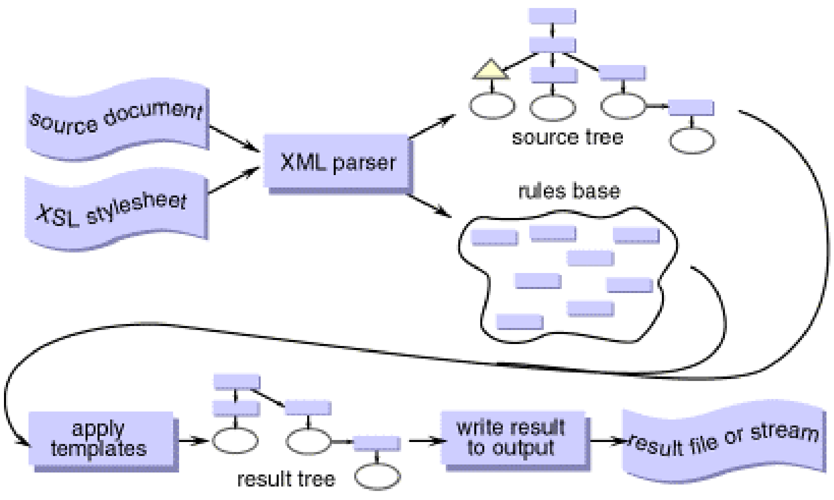


Figure 5.12 The overview of how the XML can be used

6. Knowledge Enrichment

After the XML is build from our tool, secondary information is being extracted for knowledge. This section will briefly describe what is secondary information and why it is useful to add in our tool

6.1 Introduction

Video information processing is the content creation step of the digital video library. The collaboration of different video information extraction techniques is on the information exchange level, mainly

- Knowledge Cross-referencing
- Knowledge Enrichment

For some video processing techniques, the accuracy of the recognition process can be increased by cross-referencing information generated by other modalities. For example, to identify a human face in the video, face recognition technique can be the primary modality information extracted. If available, the on screen title of the person’s name can be recognized and served as the cross-reference knowledge for the person identification. Example of knowledge enrichment is the geographical naming process. The geographical naming database is a kind of knowledge repository of geographical names of countries and cities. By applying this knowledge to the text recognized from the speech recognition, the knowledge encapsulated in the text is enriched.

The Informedia processing provided state of the art access to video by Content. This new research direction will communicate information trends across time, space, and sources by emphasizing analysis and understanding of context as well as content.

The informdia Digital Video Library contains large number of hours of video. Through automatic processing, descriptors are derived for the video to improve library access. The sheer volume of video data reveals new issue for interfaces to digital video libraries in the future. A good query engine is not sufficient because often the candidate result sets grow in number as the library grows. Interfaces for browsing both the library and defined library subsets such as the results from a query become increasingly important. The ability to extract names of organizations, people, locations, dates and times is essential for correlating occurrences of important facts, events, and other metadata in the video library, and is central to production of information collages.

Users are interested in quickly finding the set of video stories or segments relevant to their needs. When the library was on the order of a hundred hours, a statistical word query engine adequately provided this focus. Users entered text queries, and a small set of segments was returned sorted by the query engine's relevance score. When the library grew to a thousand hours, queries returned hundreds of segments, overwhelming users much like Web search engines can return lists whose length and default ordering no longer meet the needs of the user. An information visualization interface was developed to let the user browse the whole result space without having to resort to the time-consuming and frustrating traversal of a list of results. The employed visualization techniques allowed the user to browse and retrieve video from the Informedia library based on date (i.e., "when") and word occurrences (i.e., "what"). We realized that a potentially rich vein of information was the location information (i.e., “where”). This information dimension could be used in presenting overviews of the video content, summarizing multiple video segments, and as a query mechanism to find segments dealing with a particular region of interest.

Geographical references (georeferences) will be associated with each video segment and represented as a single value, a set of distinct values, or range of values corresponding to the locations where the video was situated as well as the locations referred to in the video. The user will be able to specify a named location or location coordinates in order to query or browse for events at that location or within some "distance" of that location.

Since geographic information is especially important among secondary information, our project will focus on implementing this technique. In the following, we are going to discuss this technique in details.

6.2 Extracting Video Geographic Information

The transcript of the narrative is the greatest source of geographic reference information for the videos in the Informedia library. If closed-captioned text exists for a video, it is integrated with the output of the speech recognizer.

While the transcript provides the primary source of geographic references, it is not the sole source. Often a location name and perhaps a person's name are overlaid on the video, especially for news. The Informedia Video OCR (VOCR) process identifies video frames containing probable text regions, in part through horizontal differential filters with binary thresholding. VOCR then filters the probable text region across the multiple video frames to improve the quality of the image used as input for OCR processing. Commercial OCR software converts the final filtered image of alphanumeric symbols into text. The VOCR-produced text is another potential source of geographic references. For example, a video segment discussing volcanic activity included shots of lava with the overlaid text stating "Mount Etna, Italy." While the transcript text was associated to video times through Sphinx speech alignment, the VOCR text is associated to video times through image processing which identifies the frames containing the probable text regions.

Extracting geographic information from video begins by using the text metadata as the source material to be processed. A known set of places along with their spatial coordinates, i.e. longitude, latitude are collected to form a geographic database. The text metadata, which associates text with video times, is then matched to terms in the geographic database which maps geographic text terms to longitude and latitude. The end result is the tagging of video sequences with latitude and longitude.

6.3 Our Implementation

In our project, we try to extract the names of major cities mentioned in the video. The names of major cities and their details are stored in an XML file. Our tools read the XML file as the database. Then, we compare the text extracted from the video with the names of major cities in the XML file.

The following is the XML input file we used:

```
<MAJORCITY>
    <CITY ID=0>
            <NAME>上海</NAME>
            <COUNTRY>中國</COUNTRY>
            <LONGITUDE>31 15N</LONGITUDE>
            <LATITUDE>121 26E</LATITUDE>
    </CITY>
    <CITY ID=1>
            <NAME>北京</NAME>
            <COUNTRY>中國</COUNTRY>
            <LONGITUDE>39 55N</LONGITUDE>
            <LATITUDE>116 20E</LATITUDE>
    </CITY>
    <CITY ID=2>
            <NAME>香港</NAME>
            <COUNTRY>中國</COUNTRY>
            <LONGITUDE>22 11N</LONGITUDE>
            <LATITUDE>114 14E</LATITUDE>
    </CITY>
</MAJORCITY>
```

The unit of information retrieval in the Informedia library is the video segment, which (when segmentation strategies work to perfection) contains a single story. On average, each hour of broadcast news consists of 20 segments. For each segment, a list is constructed during the geocoding process consisting of those places that are mentioned in a segment. A place may be named more than once in a segment, but it is represented only once in the segment's list. The number of references is included in the entry for each place to enable subsequent interfaces to emphasize locations visually based on how frequently the places are mentioned.

The geocoding process establishes a relationship between the video and place names. For a given video time interval, the place names referenced in that interval can be identified.

For places identified in transcript metadata, the sentence or sentences where each place is mentioned is tracked. If a place is only mentioned once, the beginning and ending times, in milliseconds from the beginning of the video, of the sentence in which it is mentioned are used as the time interval for that place. If a place is mentioned more than once, the time span from the start of the first sentence containing the place reference to the end of the last sentence containing a mention of the place is used.

For VOCR, the time span approximates the duration when the overlaid text appears in the video. As with transcripts, if a place occurs multiple times then its time span extends from the start time for its first mention to the end time for its final mention in the video segment.

In our project, the XML file is used as the outputs, which can shows the geographic information extracted from the text obtained by VOCR or speech recognition. In our project, we did not implement the part of speech recognition. So we use the text resulting from VOCR to do geographic information extraction. We try to extract the names of major cities from this text. The following XML is extracted from the XML output file in our tools:

```
<MyVideo>
<VIDEO SRC ="C:\Documents and Settings\movie\news2\atvnews02.mpg"/>
<Modal TYPE="VOCR">
      <TIME VALUE="1">
            <TEXT>
            The reporter is speaking   ...
            </TEXT>
      </TIME>
      <TIME VALUE="34">
            <TEXT>
            A flight flying from 西雅圖  to  華盛頓  ….
                  <CITY>
                  西雅圖
                  </CITY>
```

```
                <CITY>
                華盛頓
                </CITY>
            </TEXT>
        </TIME>
        <TIME VALUE="38">
            <TEXT>
            There is an accident   ...
            </TEXT>
        </TIME>
    </Modal>
    </MyVideo>
```

The words in red above "A flight flying from 西雅圖 to 華盛頓 ...." is the text in VOCR that contains names of city names. In this example, it contains "西雅圖" and "華盛頓" (the words in blue), which are two major cities in The United States of America. They are stored under the tag "CITY" inside the tag "TEXT". At the same time, each text extracted from VOCR is referenced to a time value (in second), which indicate when this information appears in the video. The words in green shows that the names of city "西雅圖" and "華盛頓" appear in the video at 34 seconds after the video being played.

In our XML file, the whole content of VOCR, including the text, the time referencing and the city names extracted from this text are included inside the tag "MODEL". This indicates that VOCR is only one of the models that are used in our tools. There can be a number of different models. For example, we can have a model called Speech Recognition in the future. The result of Speech Recognition can be represented similar to that of the VOCR. We can treat every sentence extracted from the recognizer as a text and we can store the time that the reporter is speaking this sentence. Then we can do similar text extraction as we do in VOCR, to extract the geographic information that contains in this text.

The following figure shows the flow of geographic information extraction:

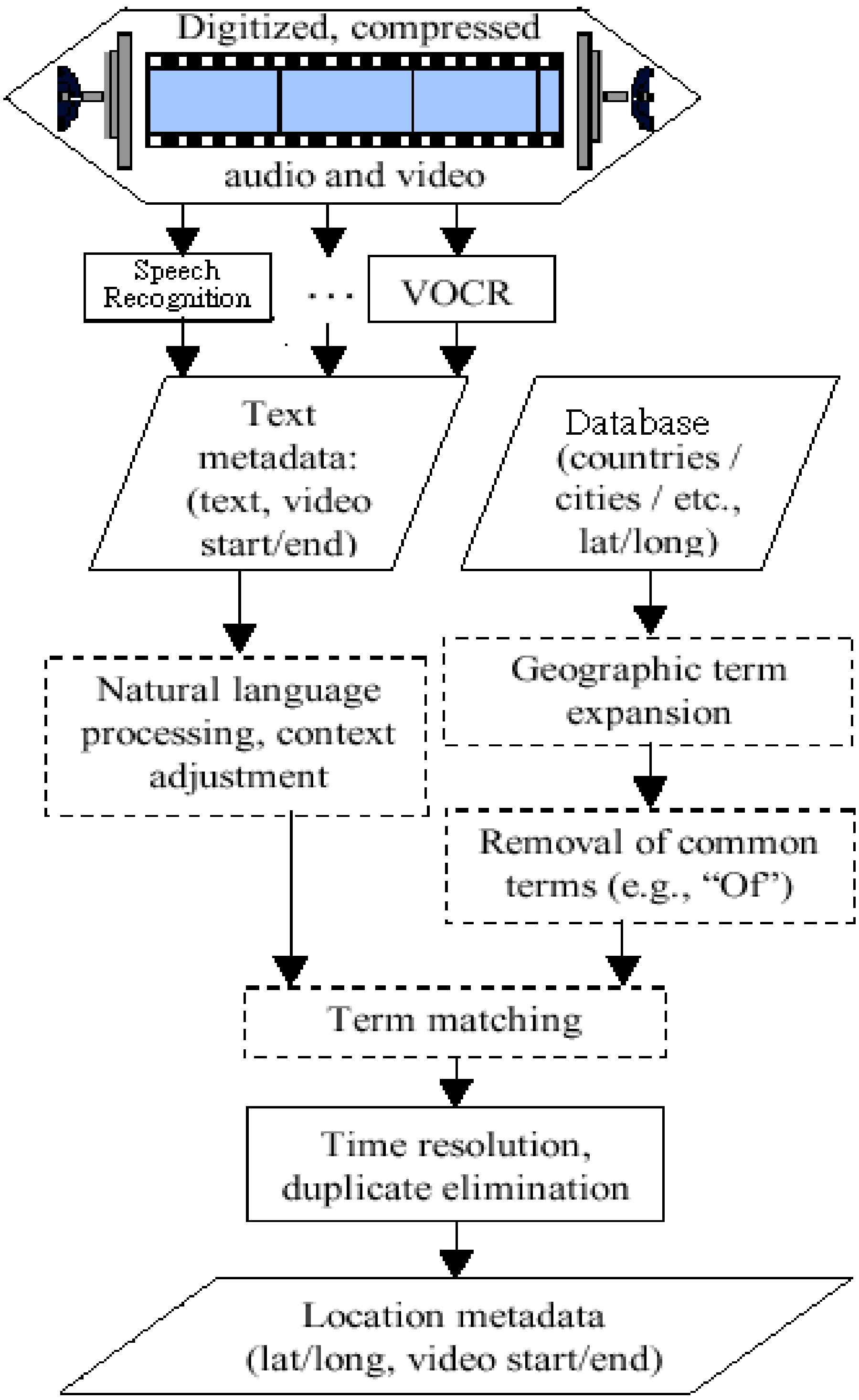
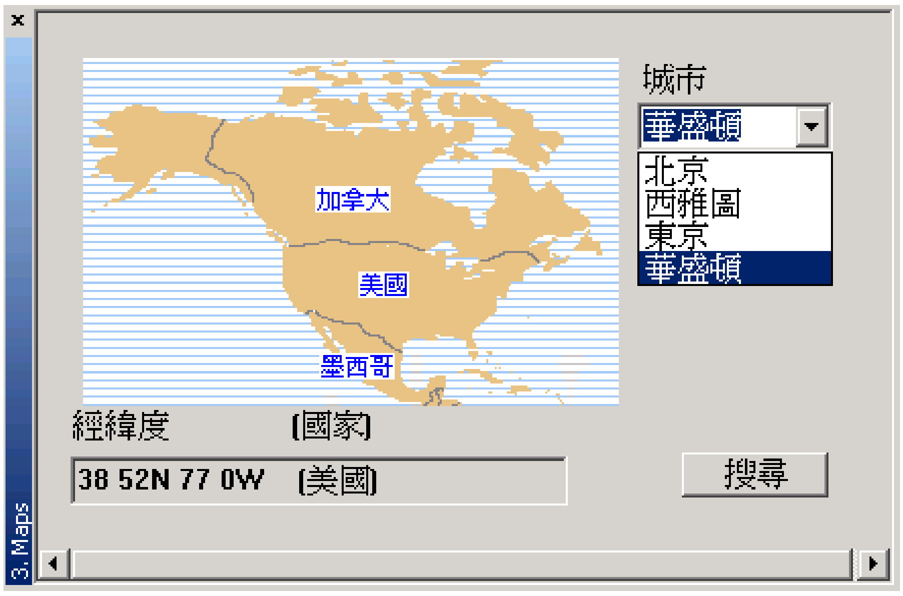

Digitized, compressed
audio and video
Speech Recognition
. . .
VOCR
Text metadata: (text, video start/end)
Database (countries / cities / etc., lat/long)
Natural language processing, context adjustment
Geographic term expansion
Removal of common terms (e.g., "Of")
Term matching
Time resolution, duplicate elimination
Location metadata (lat/long, video start/end)

Figure 6.1 flow of geographic information extraction

It shows that the system use the text extracted from VOCR and Speech Recognition, then compare them with the names of major cities in our database. Then we can do term matching to find out the names contained in our video. Using the information provided by the database, we can get further information about these cities. For example, we can get the name of country that the cities belong to and the latitudes and longitudes of them.

In our project, the names of major cities of The United States of America, China and Japan are extracted from the text resulting from VOCR. The list of names of major cities is shown and user can choose anyone of them. When a name of city is selected, the details of that city will be displayed. The longitude and latitude of that city will be shown and the name of country that the city belongs to will be shown too. Also, a map of that country and its surrounding countries will be displayed.

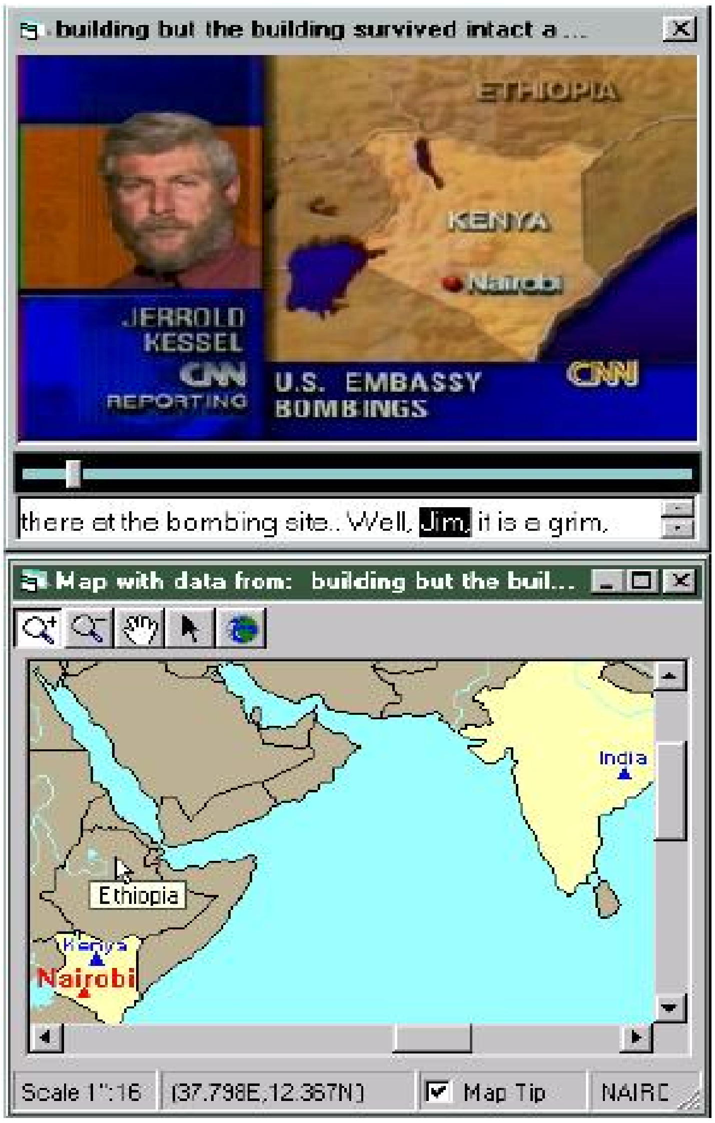


38 52N 77 0W (美國)

Figure 6.2 Secondary information extractor in the tool

6.4 Examples on Map Representation of a Video Segment

The following figure shows how the geocoded information is incorporated into the Informedia interface. When a video segment is played back, an optional window pops up that contains a map displaying all of the places discussed in that segment.



there at the bombing site.. Well, Jim, it is a grim,

Figure 6.3 how the geocoded information is incorporated into the Informedia interface

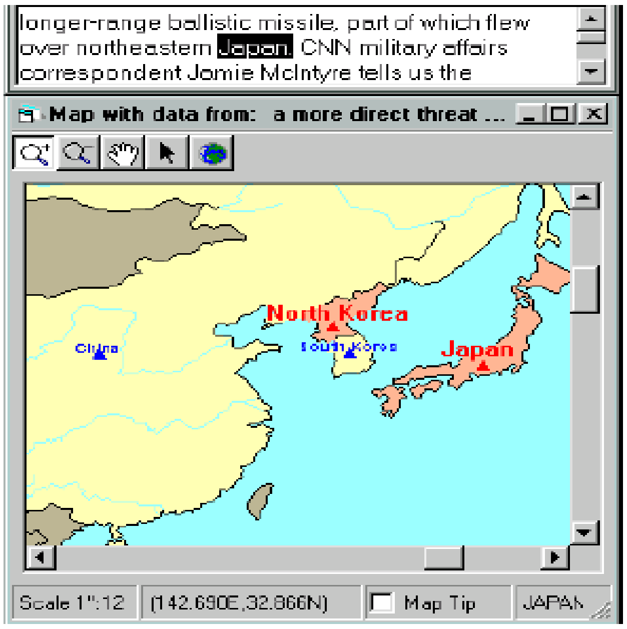
The use can interact with the map through the use of the toolbar icons:

- Zooming in to reveal more detail and less area
- Zooming out to show more area and perhaps less detail
- Panning to a different area at the same resolution
- Selecting an area to search (discussed in the next section)
- Returning to the full map extent which shows the countries of the world

It should be note that the real value to the geocoding and map interface is in displaying location information for video segments for which the producer has not previously added a map within the video data. Every news story does not have an embedded map that becomes part of the broadcast, but through our geocoding maps can be automatically produced to reflect the areas discussed within each story. Another benefit is that the user can interact with the interface map using the toolbar icons to get additional detail, whereas no such interaction is possible with an image of a map encoded as part of the video stream.

The maps accompanying videos are not static displays. They are animated in synchronization with video playback. As places are discussed, they are highlighted on the map. For countries and administrative areas such as states or provinces, the areas contained within their respective polygon boundaries are highlighted by changing the color with which they are shaded. Areas covered at some time during a video segment are colored yellow; when the video frames during which an area is discussed are played, that area is then colored orange. For cities and other places, the marker color is changed from blue to red and the label is shown. A glance at the map can then show the areas of current focus.

The following figure shows a story where the current focus is on North Korea and Japan, with China, South Korea and Russia mentioned elsewhere in the story. The highlighting changes over time to show the story flow within the video segment.


longer-range ballistic missile, part of which flew over northeastern Japan. CNN military affairs correspondent Jamie McIntyre tells us the

Map with data from: a more direct threat ...

Scale 1":12 | (142.690E,32.866N) | Map Tip | JAPAN

Figure 6.4 story where the current focus is on North Korea and Japan, with China, South Korea and Russia mentioned

6.5 Accessing Video through Spatial Queries

The maps in the Informedia interface are not merely for presentation but also can be used to specify a location query. The arrow icon in the toolbar for the map window is used to drag a rectangular region on the map that serves to identify the user's region of interest

For example, the user can select the region that he is interested in. Then, searching against the corpus, within a few seconds the results are displayed with headlines and representative thumbnail images, about the news on the region selected. Feedback is provided on the map to indicate the locations within the specified query that actually produced results.

The Informedia digital video library interface supports word query, image query, and now spatial query through the use of maps. One interesting area of work will be to enable mixed modality query, such as finding all the video segments mentioning "famine" for a specified area on the map, or finding all the faces like a given face for a specified country. The presentation of information can be improved by utilizing not only the information dimensions of date/time and topic, but also the dimension of location. Results from word queries could be visualized on a map, revealing interesting patterns for discovery.

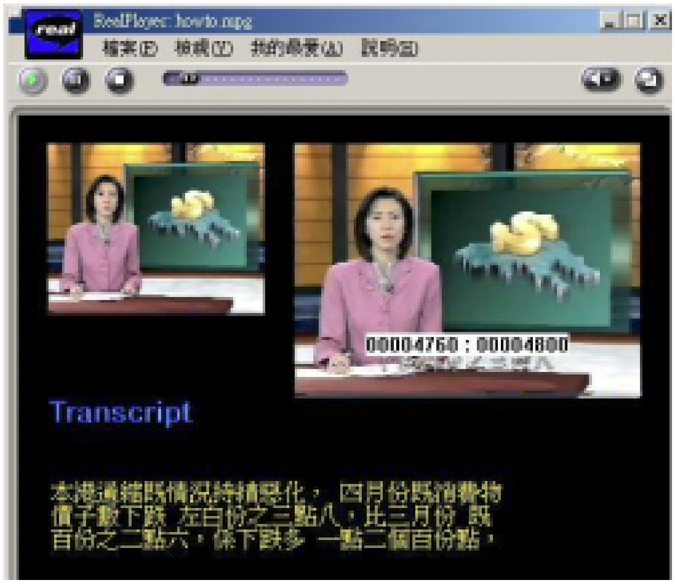
7. Multimedia Transformation and Presentation

After generating the enriched XML-based video data, we plan to present the data properly. Different types of media objects can be presented in this multimedia, like transcript from speech recognition, scenes from scene change detection, maps and geographic information from knowledge enrichment, the video itself, etc. Therefore, finding a good language that can integrate and synchronize all media objects in a single presentation is important. Fortunately, SMIL was found to be a good language that can complete the task. In our project, we are building a XML to SMIL transformer for converting the enriched XML-based video data from XVIP to different templates of SMIL presentation. Also, we are integrating the transformer into XVIP and allowing users to trigger the SMIL presentation in XVIP.

7.1 SMIL

7.1.1 Introduction

Rich and interactive multimedia applications, where audio, video, graphics and text are precisely synchronized under timing constraints are becoming ubiquitous. Like our multimodal system, which combine different modalities, like speech recognition, scene change detection, face detection and VOCD. A mulitmodal, multimedia services become very important. Fortunately, the W3C has sponsored the development of SMIL, an elegant notation for multimedia applications, which has been embraced by both Microsoft and RealNetworks [30]. SMIL, stands for Synchronized Multimedia Integration Language, is currently is a markup language that can synchronize and integrates multimedia. It enables authors to specify when and what should be presented, enabling them to control the precise time that a sentence is spoken and make it coincide with the display of a given image appearing on the screen [31]. Figure 7.1.1 is example showing our SMIL file, which integrates the video, images from scene change detection and the text obtained from speech recognition.



本港通縮既情況持續惡化， 四月份既消費物價指數下跌 左百份之三點八，比三月份 既百份之二點六，係下跌多 一點二個百份點，

Figure 7.1.1 A snapshot of SMIL presentation in our FYP.

7.1.1.1 Basic idea

The basic idea of SMIL is to name media components for text, images, audio and video with URLs and to schedule their presentation either in parallel or in sequence. A typical SMIL presentation has the following characteristics:

- The presentation is composed from several components that are accessible via URLs, e.g. files stored on a Web server.
- The components have different media types, such as audio, video, image or text. The "begin" and "end" times of different components are specified relative to events in other media components. For example, in a slide show, a particular slide is displayed when the narrator in the audio starts talking about it.
- Familiar looking control buttons such as "stop", "fast-forward" and "rewind" allow the user to interrupt the presentation and to move forwards or backwards to another point in the presentation.
- Additional functions are "random access", i.e. the presentation can be started anywhere, and "slow motion", i.e. the presentation is played slower than at its original speed.
- The user can follow hyperlinks embedded in the presentation.

The SMIL language has been designed so that it is easy to author simple presentations with a text editor. The key to success for HTML was that attractive hypertext content could be created without requiring a sophisticated authoring tool. The SMIL language achieves the same goal for synchronized hypermedia.

SMIL allows user to put text, still images, audio, video, and animation into interactive presentation. Before the development of SMIL, people had great difficulties sending and moving still images and sound to a Web user, because each element is separate from the others and can't be coordinated with other elements without elaborate programming. SMIL enables web developers to deliver multiple movies, still images, and sound separately but coordinate their timing and control spatial layout and hyperlinks. As a new and powerful web format, SMIL will definitely change the scene of web-based information delivery and human communication [32].

7.1.1.2 Advantages

- SMIL is text-based
  It is easy for any designer or developer to work with it. A text editor can already let you start without any investment, though powerful SMIL authoring tools are also available.
- Put together customized presentations.
  Because a SMIL file is a simple text file, you can generate it automatically for each visitor. You can therefore create different presentation parts, assembling a customized SMIL file for each visitor based on preferences recorded in the visitor's browser.
- SMIL effort is led by the W3C.
  The W3C is focused on ironing out standards that everyone can follow. As a standards body, the W3C tries to shape a specification that is beneficial to all parties involved. Unlike a proprietary technology owned by just one vendor, SMIL can lean toward general industry and consumer interests.
- Avoid using container formats.
  Because SMIL can stream many media formats, you don't need to merge clips into a single streaming file. To alter your presentation, for example, you simply edit the SMIL file rather than merge the clips again into a different container file.
- Use clips from different locations.
  Because a SMIL file lists a separate URL for each clip, you can put together

presentations using clips stored on any server. You can use a video clip on a RealServer, for example, and a text clip on a Web server.

- Time and control a presentation.
  The SMIL file lets you easily control the presentation timeline. You can start playing an audio clip at 2.5 seconds into its internal timeline, for example, without changing the encoded clip.
- Lay out a presentation.
  When your presentation includes multiple clips, such as a RealVideo clip playing simultaneously with a RealPix slide show, you use SMIL to define the layout.
- Stream clips in multiple languages.
  A SMIL file can list different language options for clips. To create a video with soundtracks in different languages, for example, you produce one video clip with no soundtrack, and then create separate audio clips for each language. Your Web page needs just one link to the SMIL file. When a visitor clicks that link, the visitor's RealPlayer chooses a soundtrack based on its language preference.
- Reach viewers at multiple bandwidths.
  A SMIL file can also list presentation choices for different bandwidths. RealPlayer then chooses which clips to receive, based on its available bandwidth. This lets you support multiple connection speeds through a single hypertext link, rather than separate links for modem users, ISDN users, T1 users, and so on.
- Create interactive multimedia experiences.
  Using SMIL, you can easily create interactive media presentations; such as an audio or video jukebox that plays a different clip each time the viewer clicks a button [33].

7.1.2 Players and browsers

The following listed some of the most famous players and browsers for showing SMIL presentations:

RealPlayer

RealNetworks' RealPlayer is one of the best-equipped Web multimedia players around. The G2 version of RealPlayer started supporting SMIL. SMIL for RealPlayer is the most dominant use of SMIL nowadays. RealPlayer supports multiple platforms and is available as a free download directly from RealNetworks.

Its link is as follow:

http://www.real.com/

QuickTime

QuickTime has a long history in multimedia. Its implementation of SMIL in Apple Computer's QuickTime multimedia player started out modestly. It is a cross-platform player that can be played both on Windows and Macintosh.QuickTime forms a very effective Web multimedia environment with SMIL.

Its link is as follow:

http://www.apple.com/quicktime/

Internet Explorer

Although most other companies are approaching SMIL from the player side, Microsoft is approaching SMIL thoroughly from the browser side. SMIL has been gradually started to appear since version 5 of the Microsoft Internet Explorer. Internet Explorer was one of the first of the major browsers and players to support functions of SMIL2.0.

Its link is as follow:

http://www.microsoft.com/windows/ie/default.asp

GRiNS

The GRiNS player is short for Graphical Interface for SMIL. It is one the oldest SMIL players. It was originally a research project of the National Research Insttitute of Mathematics and Computer Science in the Netherlands. Later, it spun off as a product of a company called Oratrix, it developed the GriNS player with a SMIL authoring tool.

Its link is as follow:

http://www.oratrix.com/GRiNS/

7.1.3 Document structure

The structure of SMIL document is quite similar to the structure of HTML document. SMIL has the <smil>, <head>, and <body> elements. Both SMIL 1.0 and SMIL 2.0 are compatible with the same structure. Though SMIL and HTML are very similar in their structures, the differences are as follow:

- SMIL is case-sensitive, so all the tags should be written in lower case.
- SMIL is XML-based, so all the tags should be ended properly.

The SMIL document is divided into two parts, the head and the body. They are under the tags <head> and <body> individually. However, both of them are under the parent tag <smil>. <smil> element is the outer container of the whole SMIL document. Figure 7.1.2 is an example showing a simple SMIL document.

```
<smil>
  <head>
      … Here is the head of the document …
  </head>
  <body>
      … Here is the body of the document …
  </body>
</smil>
```

Figure 7.1.2 A simple SMIL document.

7.1.3.1 Head

In the head of the SMIL document, it normally includes the <meta>, <layout>, and <switch> element as the children of <head>. The details of the above three elements are as follow:

1. The <meta> element can be used to define properties of a document (e.g., author, expiration date, a list of key words, etc.) and assign values to those properties. Each <meta> element specifies a single property/value pair.
2. The <layout> element determines how the elements in the document's body are positioned on an abstract rendering surface (either visual or acoustic).
3. The switch element allows an author to specify a set of alternative elements from which only one acceptable element should be chosen [34].

Figure 7.1.3 shows the head of the SMIL document used in our final year project. It contains the meta name and layout tags.

```
<smil>
    <head>
        <meta name="news" content="roundup" />
        <meta name="author" content="LYU0102" />
        <meta name="copyright" content="(c) 2002" />
        <layout>
        <root-layout width="568" height="540" background-color="black" />
        <region id="scene" left="16" top="30" width="200" height="150" fit="fill" />
        <region    id="video"    left="272"    top="30"    width="240"    height="180"
        background-color="black"/>
        …
        </layout>
    </head>
    <body>
        ……
    <body>
</smil>
```

Figure 7.1.3 A SMIL document showing head in details.

7.1.3.2 Body

For the body of the document, it allows user to design when and what to be displayed in the regions defined in the head of the document. The media object elements allow the inclusion of different media objects into a SMIL presentation. Media objects are included by reference (using a URI).

There are two types of media objects. media objects with an intrinsic duration (e.g. video, audio). It also called "continuous media"). Media objects without intrinsic duration (e.g. text, image). It also called "discrete media".

Anchors and links can be attached to visual media objects, i.e. media objects rendered on a visual abstract rendering surface. When playing back a media object, the player must not derive the exact type of the media object from the name of the media object element. Instead, it must rely solely on other sources about the type, such as type information contained in the "type" attribute, or the type information communicated by a server or the operating system. Authors, however, should make sure that the group into which of the media object falls (animation, audio, img, video, text or textstream) is reflected in the element name. This is in order to increase the readability of the SMIL document. When in doubt about the group of a media object, authors should use the generic "ref" element [34].

7.1.4 Timing and synchronization

The heart of the SMIL lies on the timing and synchronization different modules in the multimedia presentation. All this information should to be defined in the body of the SMIL document. In the following section, we will discuss more details on it.

Basically, SMIL can define media playing in sequence or in parallel to each other. In sequence means that the media objects are playing one after the others, while in parallel means the media objects are playing at the same time.

7.1.4.1 Element: <seq>

It contains a child set of elements and specifies how they will relate to each other along a timeline. It plays each child element sequentially, as it is listed.
Figure 7.1.4 shows part of the code in our final year project that implemented with the <seq> element:

```
<seq>
      <img src="pix/0.jpg" dur="15" region="scene"/>
      <img src="pix/15.jpg" dur="5" region="scene"/>
      <img src="pix/20.jpg" dur="7" region="scene"/>
      <img src="pix/27.jpg" dur="4" region="scene"/>
      <img src="pix/31.jpg" dur="3" region="scene"/>
      <img src="pix/34.jpg" dur="5" region="scene"/>
      <img src="pix/39.jpg" dur="1" region="scene"/>
      <img src="pix/40.jpg" dur="13" region="scene"/>
</seq>
```

Figure 7.1.4 A SMIL example plays a sequence of jpeg images one after another.

Figure 7.1.5 is the result of a sequence of jpeg images one after another by SMIL. It is similar to a slide show. In our final year project, we use the above code to display the scene change pictures one by one in the region “scece” that we defined in the layout of the SMIL document. The “dur” attribute indicates the duration of display for each scene.

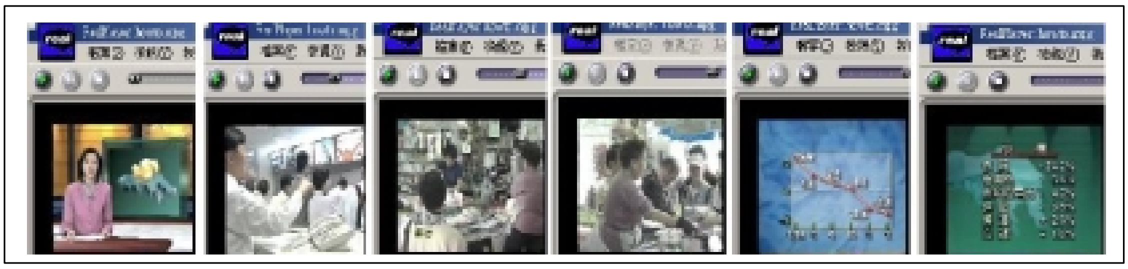


Figure 7.1.5 Result of a series of snapshot in SMIL presentation.

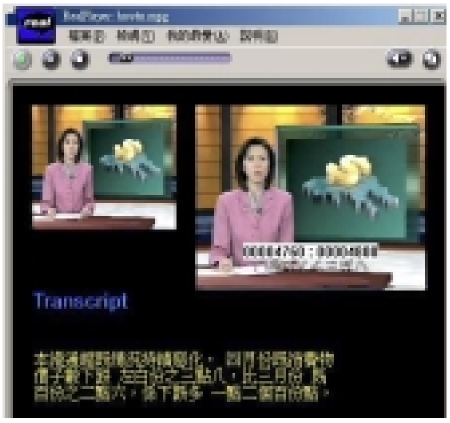
7.1.4.2 Element: <par>

The children of a par element can overlap in time. The textual order of appearance of children in a par has no significance for the timing in their presentation. The following shows part of the code in our final year project that implemented with the <par> element. Figure 7.1.6 shows that the transcript and the video playing at the same time.

```
<par>
      <text src="text/transcript.rt" region="transcript" />
      <video src="news.mpg" region="video" fill="freeze"/>
      …
</par>
```

Figure 7.1.6 A SMIL example makes transcript and video playing at the same time.

The above example plays a few media channels at the same time. In our final year project, we use the above code to display the transcript, the video, and some other channels together in different pre-defined regions in the SMIL document layout. The result is shown in Figure 7.1.7.



[illegible]

Figure 7.1.7 A snapshot of the SMIL presentation in our FYP.

7.1.5 RealPix and RealText

One of the strength of SMIL 1.0 is its simplicity, but this also brings some limitation on it. SMIL is good at working with pre-build pieces of animation, video, or audio. However, for smaller independent building blocks, such as individual images, it offered few options. It lacks the hooks to do much with those images though they can be put into the presentation easily. Also, text supported by SMIL 1.0 is very simple that it does not offer a lot of power beyond its simple role as another media type.

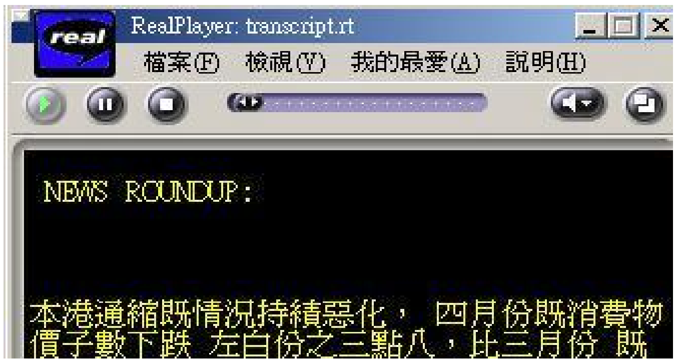
Due to the above limitation on SMIL 1.0, RealNetworks defined two proprietary SMIL-based streaming picture and text formats that reinforce SMIL presentations in RealPlayer. They are RealPix and RealText.

7.1.5.1 RealPix

RealPix allows us to create the illusion of a high-bandwidth user experience with low-bandwidth streaming images. Rendered locally by the user’s machine, images could artfully fade, dissolve, or wipe to transition from one to the next. The reference file for RealPix should use the file extension of .rp. After the RealPix files are loaded, RealPlayer renders those instructions in its display area.

7.1.5.2 RealText

RealText allows us to add tricks to the basic text in the presentations. In addition to style changes, text can be made to roll, scroll, and hyperlink. The reference file for RealText should use the file extension of .rt. After the RealText files are loaded, RealPlayer renders those instructions in its display area. Figure 7.1.8 shows a realtext example in our FYP.



NEWS ROUNDUP:

本港通縮既情況持續惡化， 四月份既消費物
價子數下跌 左自份之三點八，比三月份 既

Figure 7.1.8 A realtext file on transcript in our FYP

It should be note that RealPix and RealText are not true SMIL. They are rather the proprietary RealNetworks solutions. RealPix or RealText is unable to be run on the player other than RealPlayer G2 or later. However, RealPlayer G2 or later players have been released for a long time and RealNetworks has a huge installed user base. Also, SMIL for RealPlayer is the most dominant use of SMIL nowadays. Therefore, RealPix and RealText are still very popular formats nowadays.

7.2 The Extensible Stylesheet Language (XSL)

7.2.1 Introduction

XSL stands for Extensible Stylesheet Language. It is the language defined by the W3C to add formatting information to XML data. Stylesheets allow data to be formatted based on its structure; so one stylesheet can be used with many similar XML documents. XML and XSL are standards defined in the interest of multi-purpose publishing and content reuse. They have been gaining popularity rapidly both in industry and in academia [35].

XSL is a language for expressing stylesheets. It consists of three parts:

1. **XSL Transformations (XSLT)**: a language for transforming XML documents.
2. **XML Path Language (XPath)**: an expression language used by XSLT to access or refers to parts of an XML document. (XPath is also used by the XML Linking specification).
3. **XSL Formatting Objects**: an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary [36].

7.2.1.1 Brief History

Within a few months of the introduction of XML came the introduction of the Extensible Stylesheet Language (XSL). This early XSL standard tried to incorporate the following two functionally different operations under a single process:

Styling— Controlling the manner in which the content appears. This operation covers visual display characteristics, such as font attributes, boxes, and so on, and also allows for non-visual presentations of content, particularly aural presentation.

Transforming— Parsing an input document into a source tree with a collection of distinct nodes, and then converting that tree into a separate, perhaps quite different, result tree.

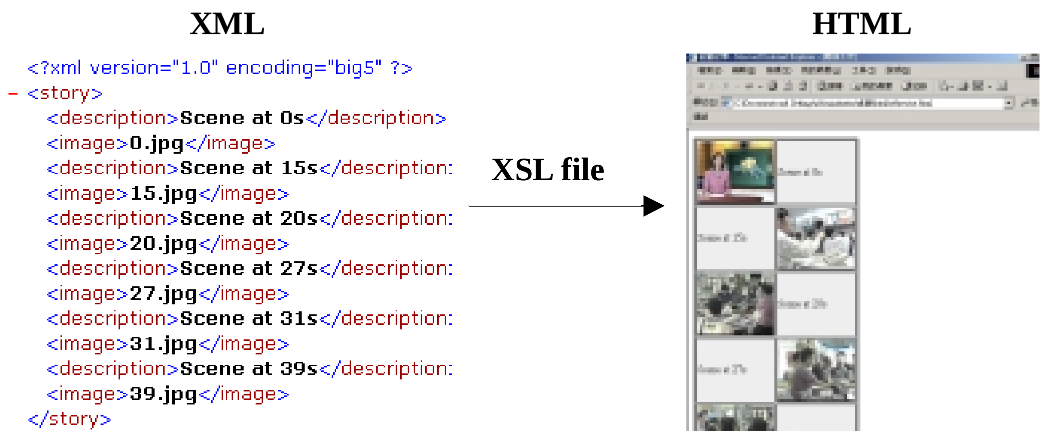
Since that time, XSL has gone through several permutations, the most important of which was a split into **XSL** proper—specifications for how to display document content—and XSL Transformations **(XSLT)**—specifications on how to transform data from one document to another.

7.2.1.2 About XSL

XSL is now generally referred to as XSL Formatting Objects (XSL-FO) to distinguish it from XSLT. The extent to which it will be widely adopted is still uncertain, because much of its functionality overlaps with that already provided by CSS and indeed the HTML tag set. Also, there are currently no XSL-FO processing applications in common use. Thus, for now, XSL-FO standalone style sheets serve largely an academic function. Until the XSL-FO standard moves to full Worldwide Web Consortium (W3C) recommendation status, vendors are unlikely to invest resources in supporting it in more than limited experimental ways. So—again, for now—the XSL-FO portion of the original XSL Working Draft is in limbo.

7.2.1.3 About XSLT

XSLT, on the other hand, in its brief history has already become profoundly successful. Its purpose is to transform marked-up documents into something else. For instance, you can convert an XML document into HTML, as shown in Figure 7.2.1; or into a comma-separated- values (CSV) flat file; or into another XML vocabulary. Originally, XSLT used a pattern-matching syntax of its own. Later, with the availability of the XML Path Language (XPath) specification, XSLT adopted XPath as its pattern-matching tool.



```
  <?xml version="1.0" encoding="big5" ?>
- <story>
    <description>Scene at 0s</description>
    <image>0.jpg</image>
    <description>Scene at 15s</description:
    <image>15.jpg</image>
    <description>Scene at 20s</description:
    <image>20.jpg</image>
    <description>Scene at 27s</description:
    <image>27.jpg</image>
    <description>Scene at 31s</description:
    <image>31.jpg</image>
    <description>Scene at 39s</description:
    <image>39.jpg</image>
  </story>
```

Figure 7.2.1 An example converting an XML document into HTML.

7.2.2 XSLT

As the above description, XSLT defines a common language for transforming one XML document into another. It defines how to create a result tree from a source tree.
It is the major function that we have applied on our final year project.

XSLT is used to transform XML documents into other XML documents. XSLT processors parse the input XML document, as well as the XSLT stylesheet, and then process the instructions found in the XSLT stylesheet, using the elements from the input XML document. During the processing of the XSLT instructions, a structured XML output is created. XSLT instructions are in the form of XML elements, and use XML attributes to access and process the content of the elements in the XML input document [37].

XSLT is not generally used for formatting. There is a separate specification for formatting from the W3C called XSL, which is generally called XSL FO (formatting objects). XSLT can affect formatting if, for instance, the XSLT stylesheet is designed to output GTML tags for display in a browser, but this is only a small fragment of its capabilities [38].

7.2.2.1 Working principles

XSLT works on two principles: pattern matching and templates. The XSLT processor takes two inputs: the XSL stylesheet and the source tree, and produces one output – the result tree. The stylesheet associates patterns with templates. The patterns identify parts of the source tree. When a match is made, the associated template is applied [38].

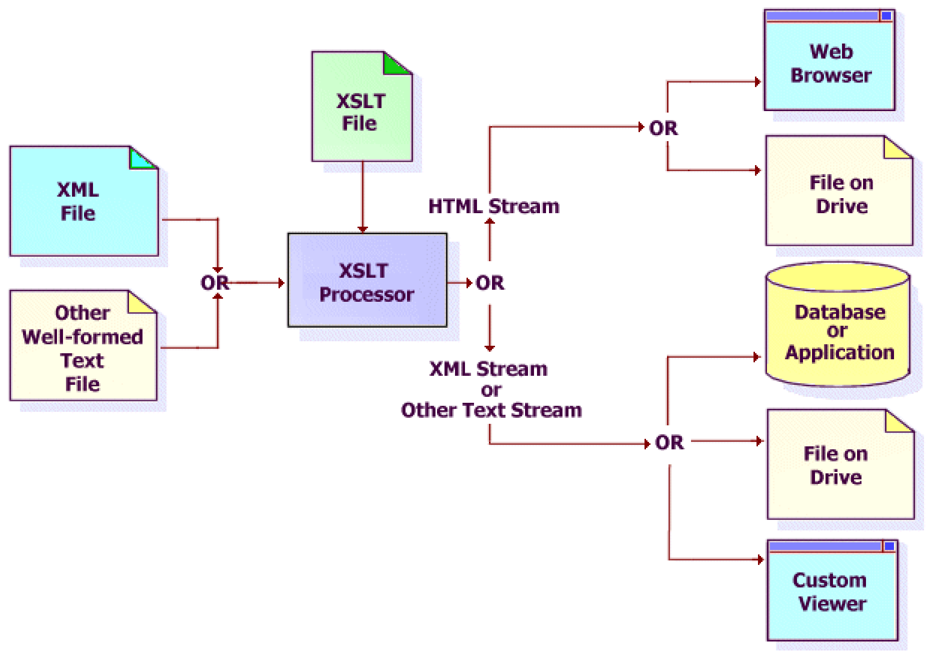


Figure 7.2.2 The XSLT process

Figure 7.2.2 shows the various input files and output destinations for an XSLT transformation. The input and output can be any text format. The most common transformation uses an XML file as input and an HTML document as output.

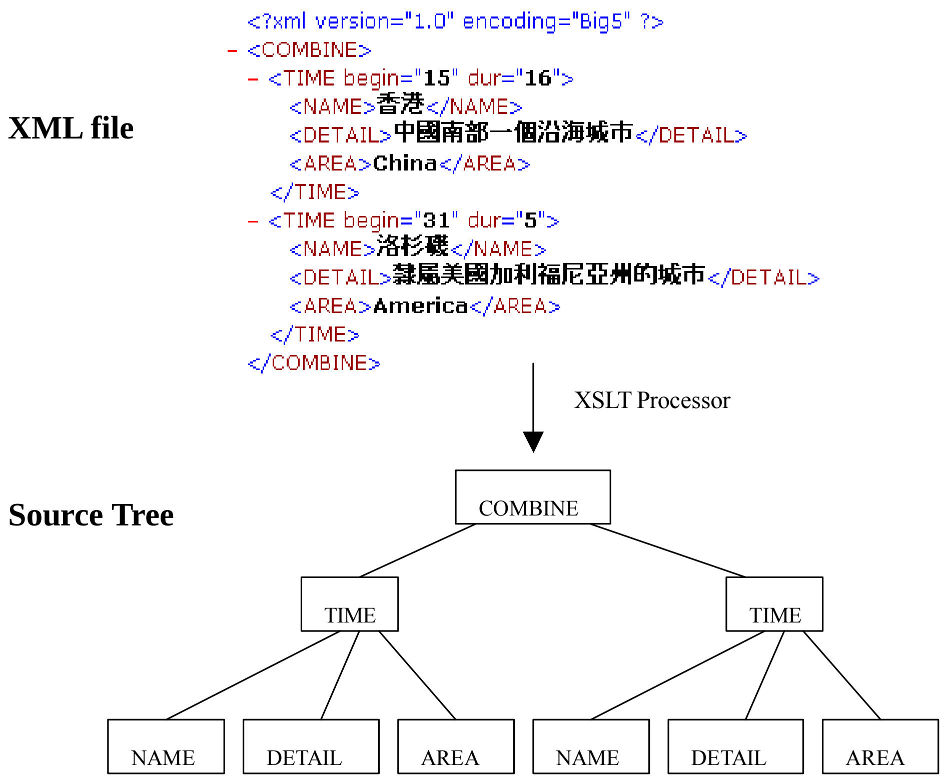
7.2.2.2 Stylesheet

An XSLT stylesheet is an XML document instance that is processed by an XSLT engine to perform a transformation on other XML documents. A stylesheet follows XML well-formedness rules, uses XSLT-specific namespace declarations, and can contain one or more XSLT templates that select and process elements from the source XML document.

In the process of transformation, the XSLT processor will first parse the XML document. A source tree is then created by the XSLT processor and stored in "memory" for reference during the processing of the XSLT stylesheet. The source tree is a representation of a well-formed XML document that is created during the parsing of the XML document. A valid source tree will always have a root node and with all nodes from the input XML document, including element nodes, attribute nodes, etc.

After the processing of the XSLT stylesheet, a result tree is produced as an output. Matching the root node of the input XML document instance and generating and output based on the rules found in the stylesheet create the tree.

The output, or result tree, from an XSLT stylesheet is always a well-formed XML document, unless another output format like HTML or text is specifically selected [37]. Figure 7.2.3 shows how to XSLT parse the XML file to a source tree.

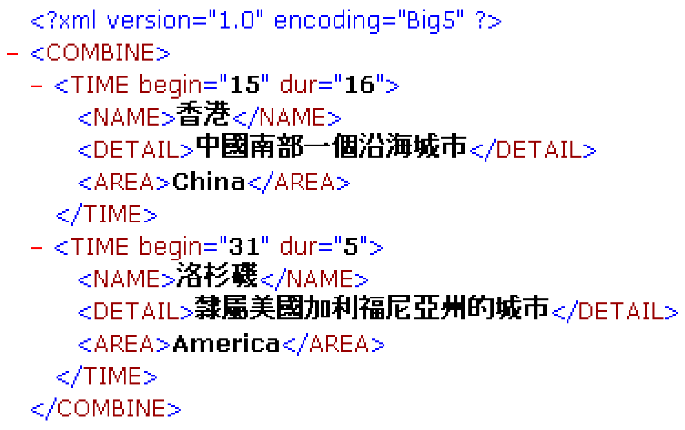


```
  <?xml version="1.0" encoding="Big5" ?>
- <COMBINE>
  - <TIME begin="15" dur="16">
      <NAME>香港</NAME>
      <DETAIL>中國南部一個沿海城市</DETAIL>
      <AREA>China</AREA>
    </TIME>
  - <TIME begin="31" dur="5">
      <NAME>洛杉磯</NAME>
      <DETAIL>隸屬美國加利福尼亞州的城市</DETAIL>
      <AREA>America</AREA>
    </TIME>
  </COMBINE>
```

Figure 7.2.3: Creating a source tree by the XSLT Processor

7.2.2.3 Document order

The concept of document order is important because the process and order by which nodes are evaluated depend on it. Basically, the order would be from left-to-right and top-to-bottom. The document order of nodes is based on the tree hierarchy of the XML instance. The first node, then, would be the root node, or document root. Element nodes are ordered proper to their children, so the first element node would be the document element, followed by its children. Nodes are selected in document order based on their opening tag. Children nodes are processed prior to sibling nodes, and closing tags, are implicitly ignored. Attribute and namespace nodes of a given element are ordered prior to the children of the element [37].



```
  <?xml version="1.0" encoding="Big5" ?>
- <COMBINE>
  - <TIME begin="15" dur="16">
      <NAME>香港</NAME>
      <DETAIL>中國南部一個沿海城市</DETAIL>
      <AREA>China</AREA>
    </TIME>
  - <TIME begin="31" dur="5">
      <NAME>洛杉磯</NAME>
      <DETAIL>隸屬美國加利福尼亞州的城市</DETAIL>
      <AREA>America</AREA>
    </TIME>
  </COMBINE>
```

Figure 7.2.4 The XML file for showing document order (Figure 7.2.5).

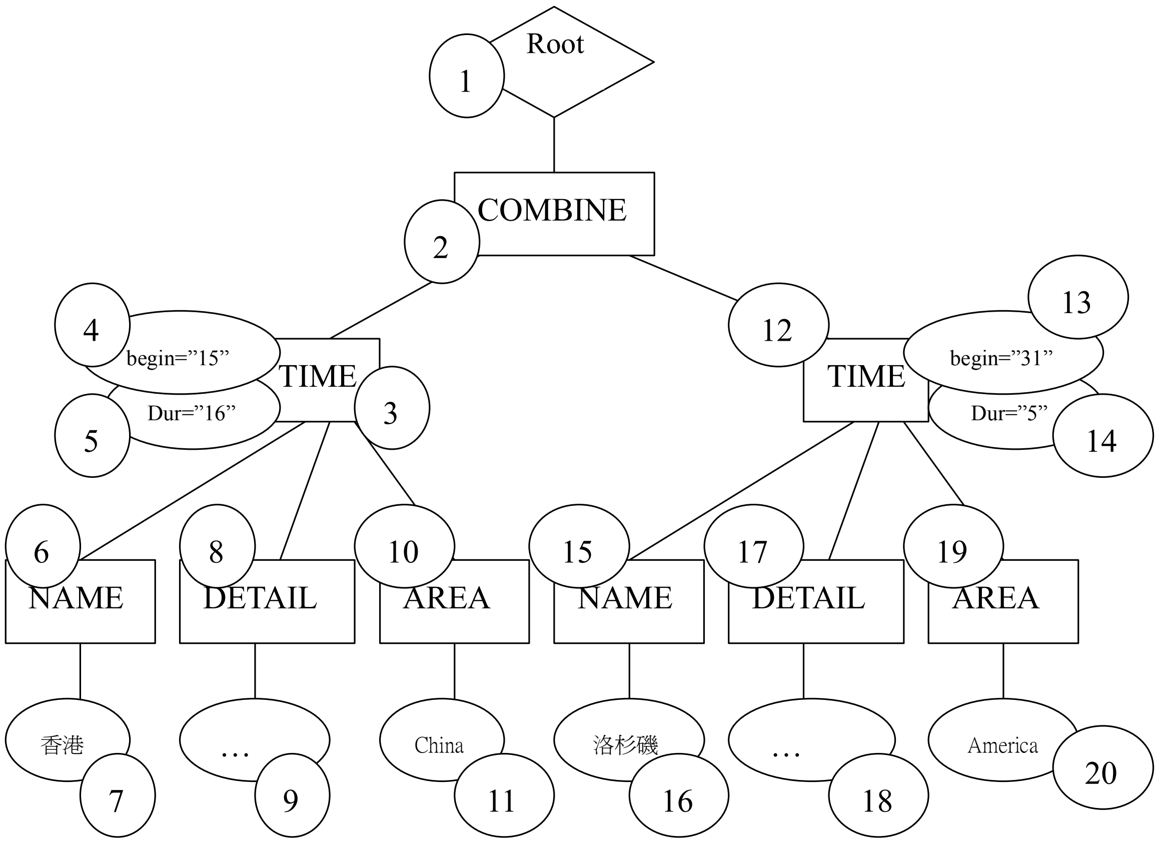


Figure 7.2.5: The document order of the above XML file (Figure 7.2.4).

Figure 7.2.4 shows an example XML file with its document order in Figure 7.2.5. The circles containing numbers indicate the order of elements in the XML file.

7.2.2.4 Templates

The main building block of an XSLT stylesheet is the template, which contains all the instructions used to process XML elements. In the simple sense, a template is a model. The XSLT stylesheet has rules that process a source tree and convert it to a result tree. The basic process associates a pattern with a template. In other words, the expression in the match statement of an <xsl:template> element is used to select nodes from the input tree. The nodes are then processed by the instructions in the template. A template is instantiated for a particular source element to create part of the result tree. The result tree is constructed by finding the template rule for the root node and instantiating its template [37].

Apart from the <xsl:template> element, the following shows some common instruction elements that are used in our final year project:

1. <xsl:apply-templates>

   Function: Directs the XSL Transformations (XSLT) processor to find the appropriate template to apply, based on the type and context of each selected node.

   Calling Method:

   ```
   <xsl:apply-templates
     select = expression
     mode = QName>
   </xsl:apply-templates>
   ```

   Example:

   ```
   <xsl:apply-templates select="AREA"/>
   ```

2. <xsl:for-each>

   Function: Applies a template repeatedly, applying it in turn to each node in a set.

   Calling Method:

   ```
   <xsl:for-each
     select = expression>
   </xsl:for-each>
   ```

   Example:

   ```
   <xsl:for-each select="TIME">
   <seq>………</seq>
   </xsl:for-each>
   ```

3. <xsl:value-of>

   Function: Inserts the value of the selected node as text.

   Calling Method:

   ```
   <xsl:value-of
     select = expression
     disable-output-escaping = "yes" | "no"
   </xsl:value-of>
   ```

   Example:

   ```
   <xsl:value-of select="."/>
   ```

4. <xsl:if>

   Function: Allows simple conditional template fragments.

   Calling Method:

   ```
   <xsl:if
     test = boolean expression>
   </xsl:if>
   ```

   Example:

   ```
   <xsl:if test="position()=1">
       <seq>…</seq>
   </xsl:if>
   ```

5. <xsl:text>

   Function: Generates text in the output.

   Calling Method:

   ```
   <xsl:text
     disable-output-escaping = "yes" | "no">
   </xsl:text>
   ```

   Example:

   ```
   <xsl:text>.jpg</xsl:text>
   ```

6. <xsl:attribute>

Function: Creates an attribute node and attaches it to an output element.

Calling method:

```
<xsl:attribute
        name = "attribute-name"
     namespace = "uri-reference">
</xsl:attribute>
```

Example:

```
<img region="map">
      <xsl:attributename="src"><xsl:apply-templates
      select="AREA"/></xsl:attribute>
      <xsl:attribute name="dur"><xsl:value-of
      select="@dur"/></xsl:attribute>
</img>
```

7.2.3 XPath

XSLT is raraly discussed without a reference to XPath. XPath is a separate recommendation from the W3C that uses a simple path language to address parts of an XML document. Although XPath is used by other W3C recommendations, there is hardly a use for XSLT that does not involve XPath. Generally speaking, XSLT provides a series of operations and manipulators, while XPath provides precision of selection and addressing [37].

XPath and XSLT specifications became recommendations from the W3C on the same day in November 1999. XSLT rely entirely on the expressions provided by XPath to select and extract nodes from the input XML document. XPath provides a common syntax and semantics for addressing nodes in an XML document, and is used by both the XSLT specification and XPointer. XPath was given its name to reflect its primary style of notation syntax, the path. This path reflects the same functionality as that of the URL, for instance. While the URL indicates a path to where a particular file is located, XPath indicates the particular path, based on the logical structure of an XML document, of where a given node is located. Paths are what the XSLT processor traverses in the course of matching, selecting, transforming, manipulating, and counting nodes. The paths are a form of expressions, which are the fundamental construct in which XPath manifest itself [37].

For example,

```
<xsl: template match=”/MyVideo/Model/Time”>
```

The words in double quotes are the XPath Expression. The syntax looks like a directory hierarchy, but it does not mean a file and the various directories containing it. The true meaning is there are MyVideo, Model and Time elements in the tree structure of the XML document.

The above paragraphs described the underlying principles of XPath, but the full value of XPath is limited without functions. Functions provide a “programming” aspect to XPath. In the XPath specification, functions are declared with a function prototype, which contains a function name, a function return type, and an argument set. The XPath function library provides functions that can be divided into groups based on the four object types: node-set, string, Boolean, and number.

The following example shows a function that is used in our FYP,

```
<xsl:for-each select="TIME">
        <xsl:if test="position()=1">
        <seq>…</seq>
        </xsl:if>
        …
</xsl:for-each>
```

The position() Function belongs to the Node-set function group. It returns a number equal to the context position from the expression evaluation context.

7.2.4 XSLT Processors

An XSL processor is something that can read the XML document and follow the instructions in the XSL style sheet, and then it output a new XML document or XML-document fragment. The following listed three of the most common XSLT processors nowadays:

Xalan

Xalan is an XSLT processor developed by Scott Boag at Lotus. The engineers at Lotus developed both Xalan C++ (using C++) and Xalan-J (using Java). Both versions implement XSLT 1.0 and XPath 1.0. XML parser is needed to validate the input XML document instances. Both the Xalan C++ and Xalan-J use the Apache parser, Xerces. The related links of the Xalan XSLT processor are as follow:
http://xml.apache.org/xalan-c/
http://xml.apache.org/xalan-j/

Saxon

Saxon is an XSLT developed by Michael Kay with his Saxon product. It has one of the largest sets of built-in extension top-level elements, instruction elements, and functions. It runs on Java and includes a servlet that allows it to be invoked directly from a URL entered into a browser. There are two forms of Saxon XSLT processor. One is the "complete" Saxon API for Java; another is a simple command-line version of the processor. The related link of the Saxon XSLT processor is as follow:
http://users.iclway.co.uk/mhkay/saxon

MSXML

Microsoft® XML Core Services (MSXML) 4.0, formerly known as the Microsoft XML Parser was released in October 2000. It allows customers to build high-performance XML-based applications that provide a high degree of interoperability with other applications that adhere to the XML 1.0 standard. Since the MSXML is convenient to be used in visual C++ and it provides complete and latest XSL functions, our final year project chose to use MXSML as our XSLT processor. Also, MSXML provides a command utility for testing. It is useful for testing the correctness of the XSL codes in our program. The related link of MSXML is as follow:
http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/594/msdncompositedoc.xml

Among the core services MSXML 4.0 provides is developer support for the following:

- The Document Object Model (DOM), a standard library of application programming interfaces (APIs) for accessing XML documents.
- The XML Schema definition language (XSD), a current W3C standard for using XML to create XML Schemas. XML Schemas can be used to validate other XML documents.
- The Schema Object Model (SOM), an additional set of APIs for accessing XML Schema documents programmatically.
- Extensible Stylesheet Language Transformations (XSLT) 1.0, a current W3C XML style sheet language standard. XSLT is recommended for transforming XML documents.
- The XML Path Language (XPath) 1.0, a current W3C XML standard used by XSLT and other XML programming vocabularies to query and filter data stored in XML documents.
- The Simple API for XML (SAX), a programmatic alternative to DOM-based processing.

The process provides by the XSLT processor, can be divided into to parts:

(1) Parsing the Documents into Trees in the DOM

MSXML parses both the source file and XSLT file into a source tree and XSLT tree, respectively. Parsing the source file into a tree is critical since the XML Path Language (XPath) is not able to navigate a source document directly. XPath treats XML documents as trees with interrelated nodes. So, by parsing a source document into a tree, XSLT is able to use XPath to navigate that tree.

Part of the parsing process is to check the documents to make sure they have well-formed XML. Processing halts with an error message if the document isn't well formed [39].

(2) Transforming and Outputting Data

The XSLT processor begins the transformation process by first looking at the transformation information stored in the XSLT tree. This transformation information is in the form of template rules.

As the XSLT processor processes the template rules, the processor outputs the transformed data into a result tree. This result tree can be either XML or any other structured text content, such as HTML, comma-separated-values (CSV) flat files, or RTF documents [39].

Figure 7.2.6 illustrates the processing relationship among the XSLT processor, XSLT tree, source tree, and result tree [39].

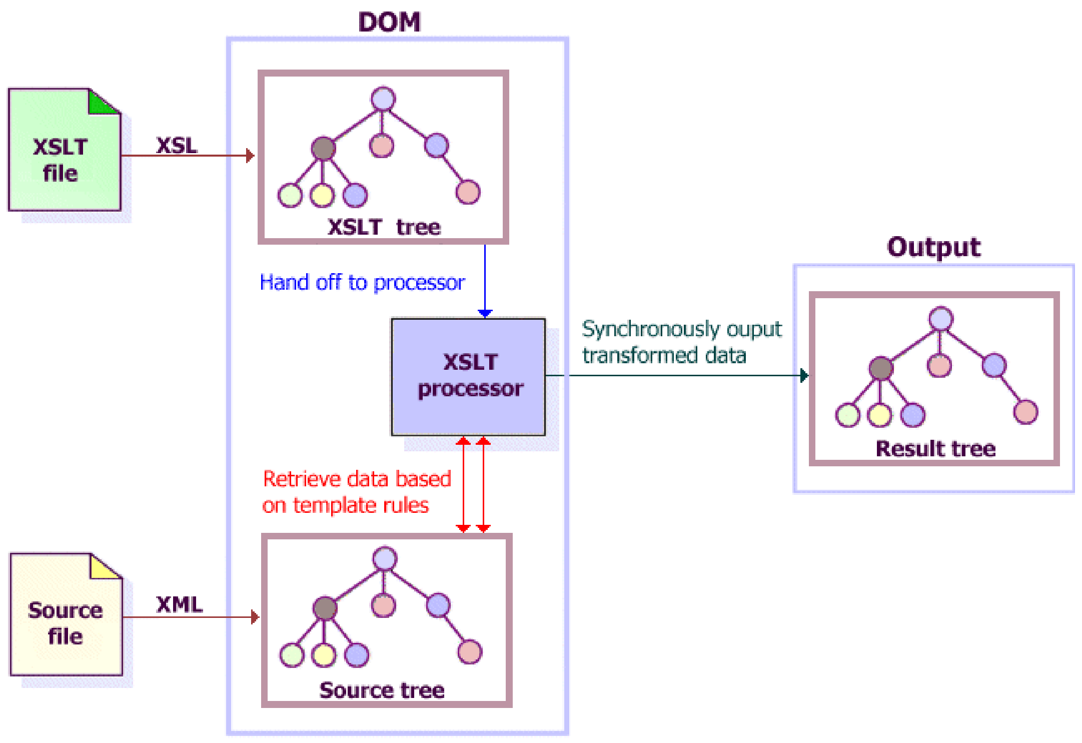


Figure 7.2.6: The processing relationship among the XSLT processor, XSLT tree, source tree, and result tree.

7.3 XML to SMIL Transformation

7.3.1 Introduction

We have extracted a lot of information from the video after doing Scene change detection, Video Optical Character Detection (VOCD), Face Detection and Speech Recognition. Then, we have stored the information extracted in the XML format. Also, we have done knowledge enrichment based on the XML file generated. The new XML file is full of information; however, it is just a text file. Although it can be delivered to different platform and be used for different purpose conveniently, it is not presentable without further implementation. Due to the above reason, we decided to think of some ways to make the XML presentable to users.

In the previous chapters, we have discussed the advantages of presenting video information with SMIL. SMIL is a markup language that can synchronize and integrates media components for text, images, audio and video. It is very suitable for presenting the video with the information stored in our XML file. The reason is that each type of information being extracted and presented with SMIL can be treated as a media component. For example, the transcript can be seen as a media component for text, the geographic information with a map can be seen as a media component for image, etc. Using SMIL, different media objects can be presented together.

In order to present the video with its extracted information efficiently, we have to think of a process that can transform XML to SMIL automatically. The following is our design and implementation.

7.3.2 Basic concept of the transformation process

In our XML to SMIL transformation process, the main purposes is to generate a SMIL presentation based on the XML files that we possess. A set of input files is required and a set of output files should be generated. The details is as follow:

Input files:

1. XML file generated from XVIP

   The most important input file, of course, is the XML file that XVIP generated after applying different video extraction techniques on the video. This file contains all the extracted information, like the transcript from speech recognition, the scenes from scene change detection, etc.

2. Data files for additional information
   A number of data files are required for doing knowledge enrichment. For example, we need a data file with names and details of major cities if we want to present information about major cities mentioned in the video. If we also want to see the map of the cities, we need to have a field for storing the file names of the maps in the data file too. Similarly, if we want to have additional information about famous people in the world, we need another data file for storing this kind of information.

Output files:

1. A SMIL file
   A SMIL file for the layout and synchronization on different media objects in the presentation should be generated. It should include the layout of the presentation in the head, and define the timing and synchronization for different media objects in the body. It should also define which type of media object and the source to be appeared in different regions.
2. Some realtext files
   Since a lot of text need to be displayed in the SMIL presentation, just like the transcript, the additional information for major cities and famous people. All these text are stored in separate realtext files, treated as different media objects. They will be integrated and displayed together with the right timing using SMIL.

7.3.3 Designs on the transformation process

Our transformer is basically an XML to SMIL transformer. Its purpose is to generate a SMIL presentation based on the XML input files. In the above section, we have already discussed the input and output files of our transformer. To build a real transformer, we have to consider a lot of factors in order to get a good design and think of the possibility to implement our design. We have thought of a few designs of our transformer before coming to our final design. The following is the details of each design:

7.3.3.1 Design 1

In our first design, we plan to build the whole transformer with Visual C++. It is the most direct method that we think of. The reason is that all the input and output files are actually text-based. It is convenient to read the input files by our program, get the information we need, then generate the output files we want. Though this design is quite simple and easy to implement, it has great disadvantages.

The layout and design of the SMIL presentation needed to be hard-coded in the Visual C++ program, this makes it program-dependent.

Disadvantages:

1. Whenever the layout and design of the SMIL presentation want to be changed, the code in our program need to be changed accordingly. This brings a lot of inconvenience, as lots of codes have to be modified.
2. Whenever a new media module or some additional information was added to our system, the code in our program need to be changed accordingly. This makes our transformer hard to extend.

7.3.3.2 Design 1 with modification

Since our first design has lot of disadvantages, we try to modify it to eliminate some of the disadvantages. We plan to provide an additional file or a user interface as a template. It is used for defining the layout and basic design of our SMIL presentation. The most important thing is that user can modify this template in order to change the layout of the SMIL presentation, without modifying the transformer program. Though users now have more freedom to design their own presentation, but there are still some disadvantages in this design.

Disadvantages:

1. The flexibility provided for modifying the layout and design of the SMIL presentation is still limited. The reason is that there is too many attributes provided in SMIL for setting the layout and design the presentation; simply an additional file or interface is impossible to define all kind of design.
2. Our program defined the format of the additional file or interface provided for user input. It is not an international standard for defining a template. User may feel unfamiliar in defining the presentation with our template.

7.3.3.3 Design 2

Aims at the disadvantages in our previous designs, we try a new approach. We found that XSL Transformations (XSLT), which a language for transforming XML documents, can assist us in doing transformation between XML and SMIL. As we mention in a chapter before about XSL, XSLT defines a common language for transforming one XML document into another. It defines how to create a result tree from a source tree. It is the major function that we can apply in our transformation process.

XSLT works on two principles: pattern matching and templates. The XSLT processor takes two inputs: the XSL stylesheet and the source tree, and produces one output – the result tree. The stylesheet associates patterns with templates. The patterns identify parts of the source tree. When a match is made, the associated template is applied.

Advantages:

1. Since the stylesheet defining the template in the XSLT is a separate file. This makes the template program-independent. Whenever the layout of the SMIL presentation wanted to be modified, the code in the transformer need not be modified.
2. Whenever a new module is added to our system, we just have to modify or create another stylesheet to assist the transformation. This makes our system more extensible.
3. XSL is the language defined by the W3C to add formatting information to XML data. It defines a universal standard in defining template that many people know about. User may feel more familiar in using this format in defining templates.
4. Stylesheets in XSL allow data to be formatted based on the structure of the XML, so one stylesheet can be used with many similar XML documents. This makes the stylesheet reusable.

Due to the above advantages, it is the design that we are currently adopting. In the following section we will have more detail explanation on our transformation process.

7.3.4 Our transformation process

Our transformation process is divided into two parts, which are knowledge enrichment and outputting files. The details are as follow:

Part 1 – Knowledge enrichment

The purpose of knowledge enrichment is combining the data files of additional information with the original XML file generated after video information extraction. For example, we can have a data file storing the name and information of major cities in the world and our XML file on video information. From the XML file on the video, we find some names of major cities have been mentioned in the video. We can then find the relative information about those cities from the data file and combine them.

Part 2 – Outputting files

The purpose of outputting files is to use the enriched result to generate a set of output files for SMIL presentation with the assist of XSL. The output files include a SMIL file defining the layout and the appearing of images and video, and some realtext files for showing the transcript and additional information.

In the following sections, we will go through each part of the transformation process with more details.

7.3.4.1 Knowledge enrichment

In the part of knowledge enrichment, we will read different data files of additional information and find out the information that is related to our video from these files. For example, we want to do knowledge enrichment on the major cities that have been mentioned in the video. We have to read a data file containing information of major cities, and also the XML file generated after video information extraction. Then, we have to combine them and generate a new XML file with the combined information. The process is shown in Figure 7.3.1.

It should be note that XSLT cannot read 2 input files that are XML files. Also, XSLT is not target for doing comparison and combination between XML files. Therefore, we have to implement this part with our own code.

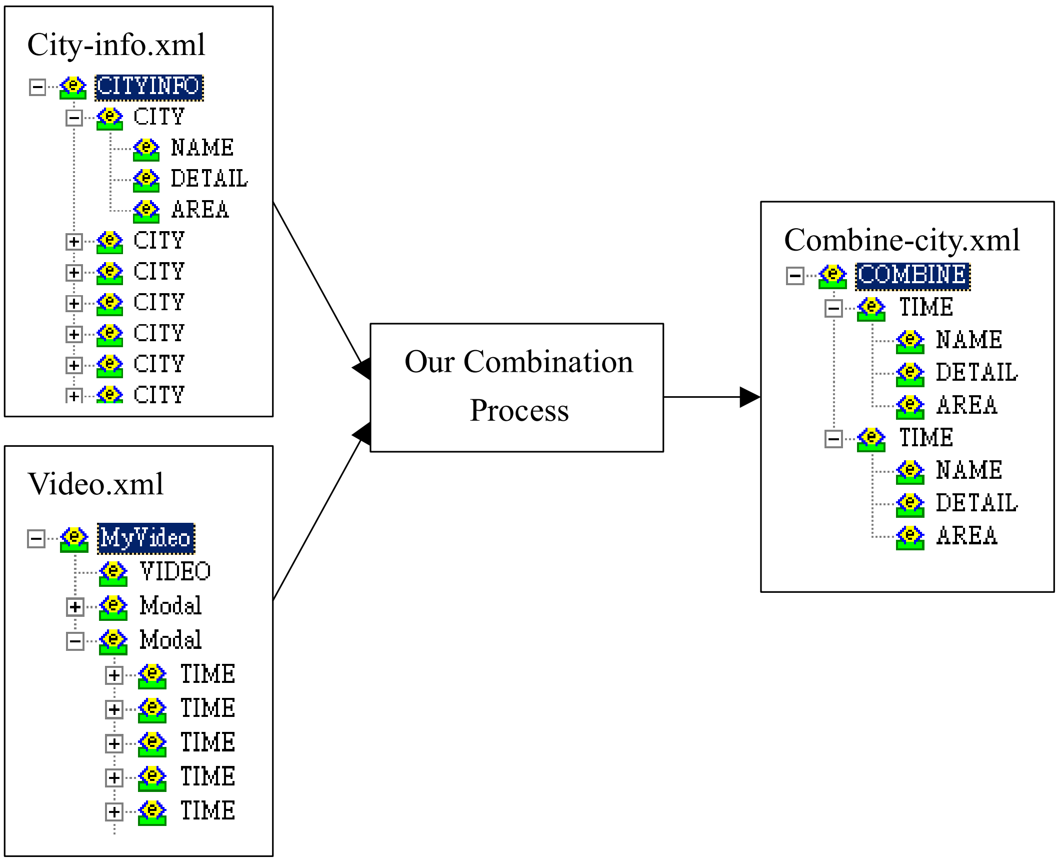


Figure 7.3.1 Combination process in knowledge enrichment.

The combined XML output file shown in Figure 7.3.2. It contains additional information related to the video. In this example, the cities "香港" and "紐約" are mentioned, so the details of these two cities are stored in the output file for further use. It should be noted that under the tag "TIME", there are two attribute "begin" and "dur" indicating the starting time and the duration of the city that is mentioned relative to the video.

```
<?xml version="1.0" encoding="Big5"?>
<COMBINE>
<TIME begin="10" dur="11">
<NAME>香港</NAME>
<DETAIL>中國南部一個沿海城市</DETAIL>
<AREA>China</AREA>
</TIME>
<TIME begin="21" dur="20">
<NAME>紐約</NAME>
<DETAIL>隸屬美國紐約州的城市</DETAIL>
<AREA>America</AREA>
</TIME>
</COMBINE>
```

Figure 7.3.2 XML file created (combine-city.xml) created after combination process.

The combination process can be applied to different kind of additional information. For example, apart from the geographical information, biographical or other information can also be combined. The process is shown in Figure 7.3.3.

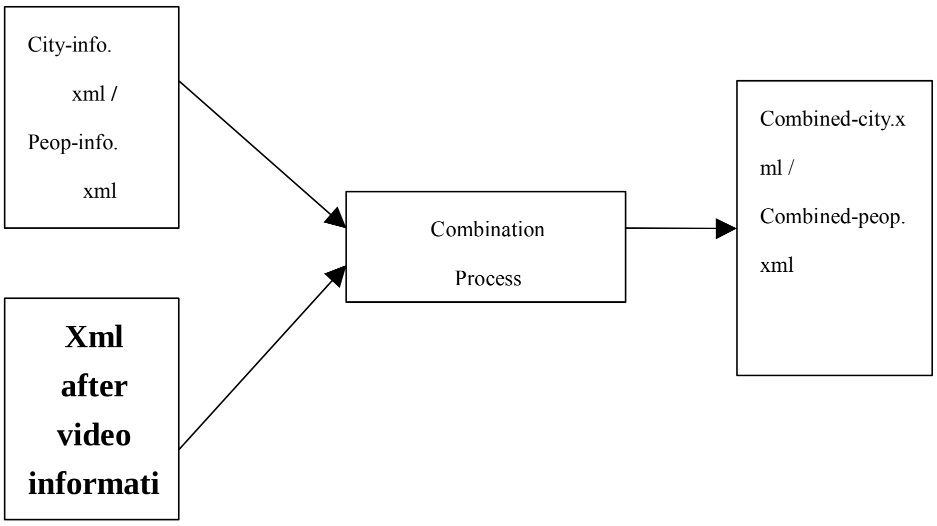

City-info. xml / Peop-info. xml
Xml after video informati
Combination Process
Combined-city.xml / Combined-peop.xml

Figure 7.3.3 Knowledge enrichment applies to different kinds of additional information.

7.3.4.2 Outputting files

As we mentioned before, there are two type of files need to be created in order to produce a SMIL presentation. One is the realtext files for displaying text, another is the SMIL file for the layout of the presentation. In the following part, we will discuss the process of creating these files with details.

Creating realtext files

There are different realtext files needed to be created, such as the realtext files for transcript, details of major cities and famous people mentioned in the video, etc. In our transformer, we define different templates (stylesheets/ XSL files) for each realtext file to be created. These processes are shown in Figure 7.3.4.

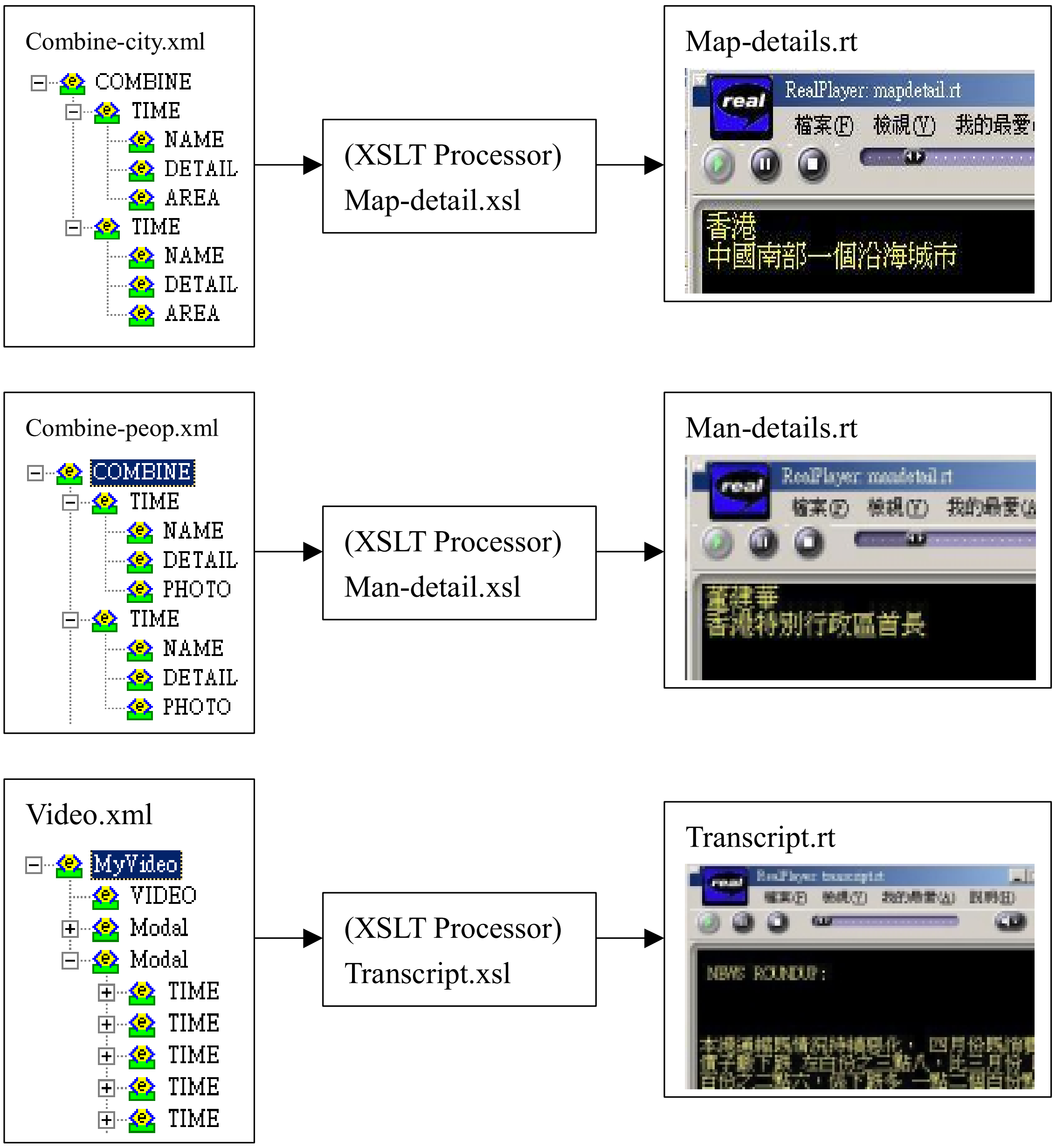


Figure 7.3.4 Creating realtext files in our transformer.

The process of generating realtext files can be extended easily. Whenever a new additional information or a new module is added into our system, we just have to modify or create a new stylesheet. With that new stylesheet and the data file, we can output the realtext file immediately using the XSL processor.

Creating the SMIL file

The SMIL file that we need to create contains the layout and definitions for the content to be displayed in different regions. This SMIL file is actually the heart of the presentation. It integrates different media objects that are going to be displayed, such as the video, images from scene change detection, transcript from speech recognition, images of maps, images of faces, description of major cities and famous people, etc. Also due to this reason, we need to combine all the above information together, with proper layout for the presentation and correct displaying order for the objects in different media. It is a great challenge and not so simple.

After our analysis, we found that the combined XML files of additional information on the video, the XML file after video information extraction, a stylesheet for the layout of the SMIL presentation are needed. Also, different regions may need individual for defining the timing and synchronization order for their child objects.

At the same time, we understand the limitation of XSL. It can only read one input data file and one XSL file, and then generate one output file. It cannot do combination between files and it is not allowed to read more than one data file. This gives difficulty in reading a number of XML files, and generates a complicated file that need to join different results for SMIL presentation. Therefore, we know that we cannot simply write a stylesheet and use XSLT to do the transformation as we did for creating the realtext files.

Finally, we think of a solution that allows us to complete the above task. First, we use a XSL file to generate the layout of the whole presentation and define the timing and synchronization of the video, scenes and different realtext files that we have generated before. It is shown in Figure 7.3.5. Basically, the layout file is for generating the layout and timing and synchronization of those media objects that do not need additional information. Secondly, we use separate XSL files to generate the timing and synchronization result of the media objects that with images, like slide show. This media objects include the images of maps or photos of famous people, etc. After creating all the timing and synchronization result of the media objects, we saved them in temporary files. It is shown in Figure 7.3.6. Thirdly, we try to put the temporary files that we generated into our SMIL file and delete all the temporary files. After the about process, a SMIL file with the layout, timing and synchronization content are created as shown in Figure 7.3.7. It can then be played and presented. The resulting presentation is shown in Figure 7.3.7.

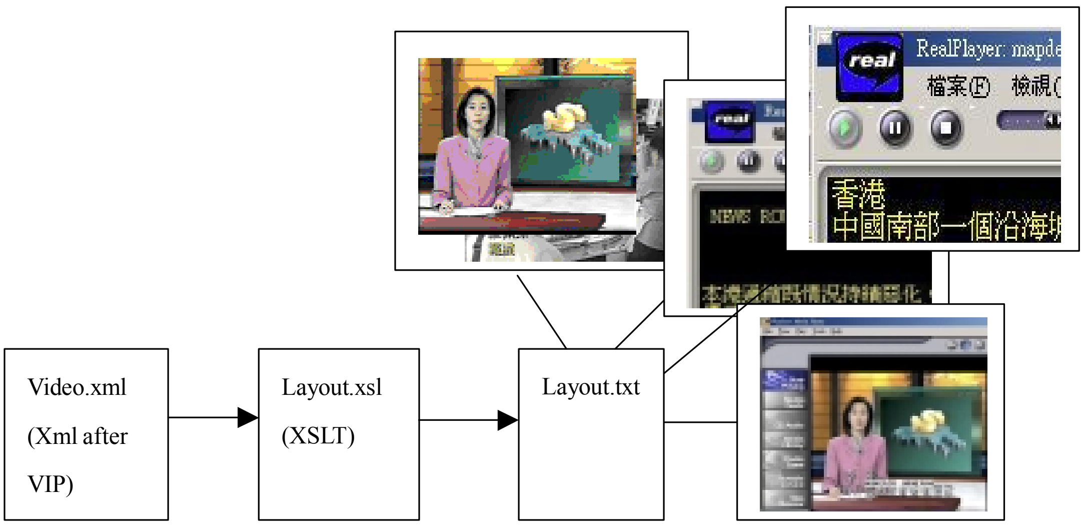


Figure 7.3.5 Process for generating layout of the whole presentation.

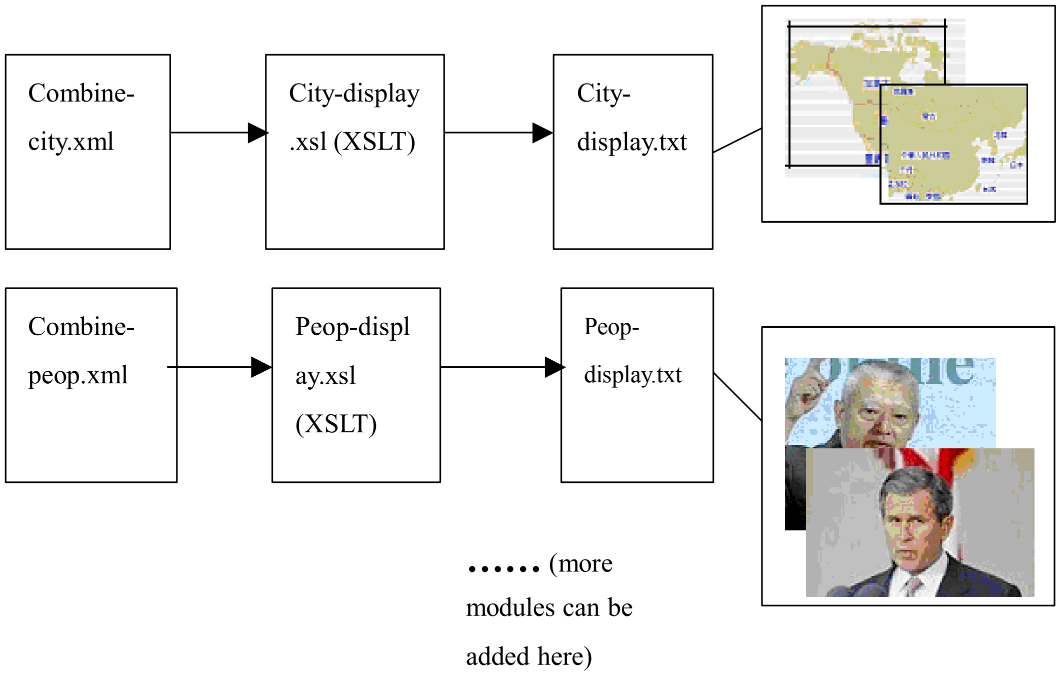


Figure 7.3.6 Process for generating timing and synchronization results for different media objects.

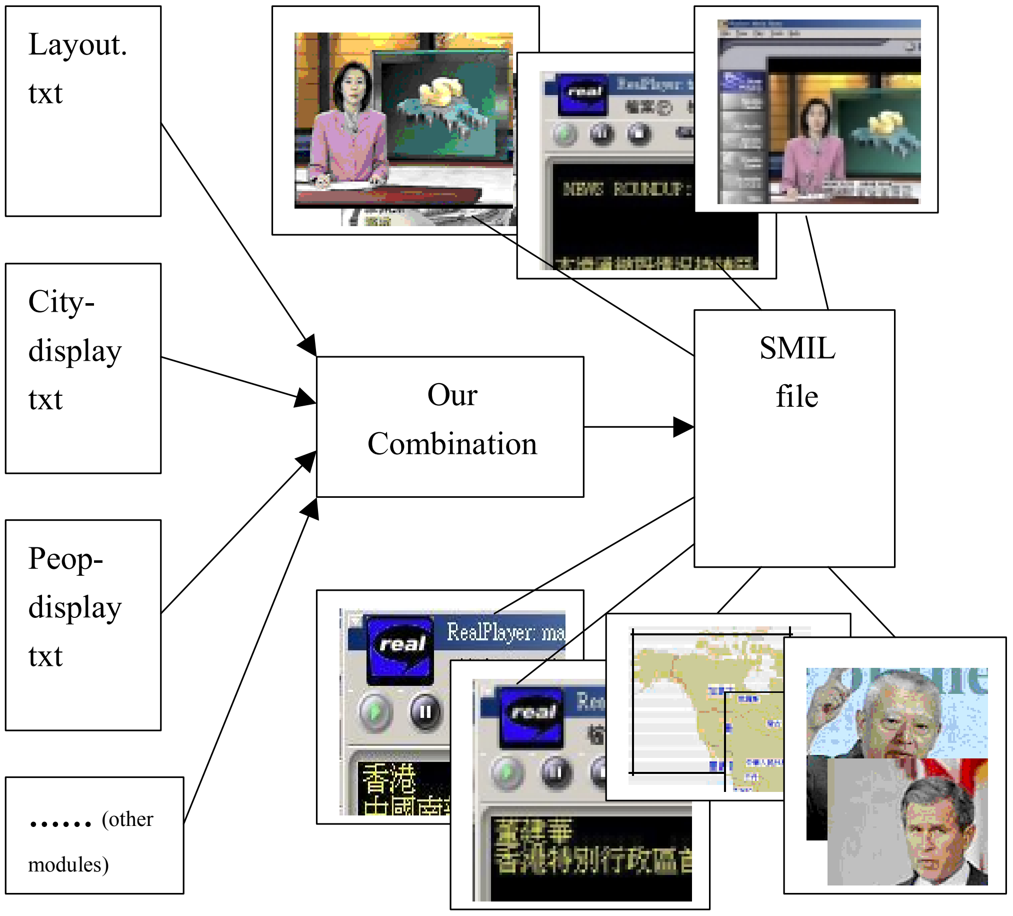


Figure 7.3.7 Putting all the temporary files generated into our SMIL file.

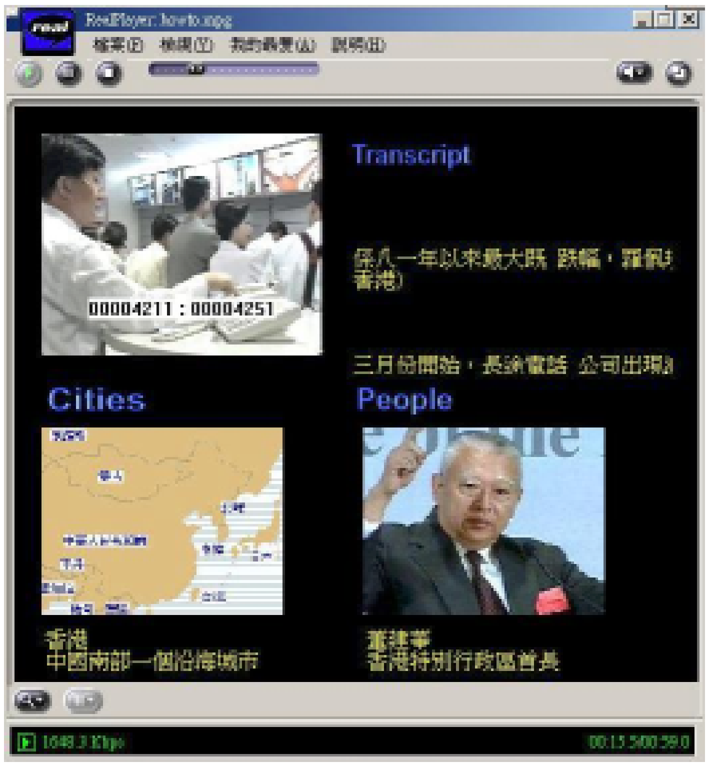
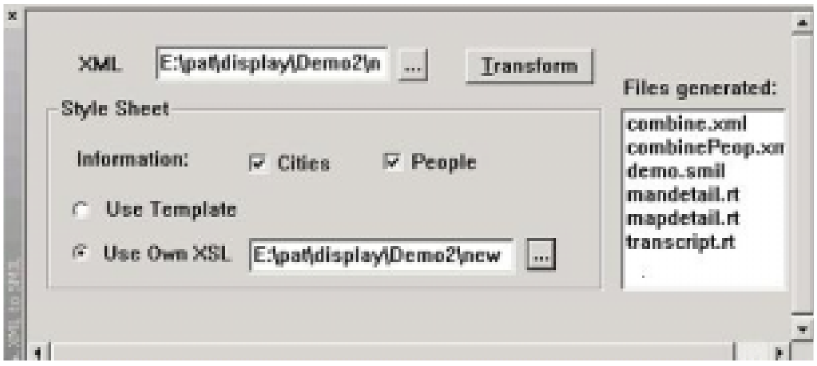


Figure 7.3.8 The SMIL presentation with the files we created.

7.3.5 User interface

In our final year project, we integrated our XML to SMIL transformer into XVIP. This make XVIP can complete all the jobs from video information extraction, XML storage to producing SMIL presentation.

In the window that is providing the transformation function in XVIP, we require user to input the XML file that is generated after video information extraction. Also, user can specify if they want to use our default template or their own template to do the transformation. The user interface of the transformer is shown in Figure 7.3.9.



XML E:\pat\display\Demo2\n

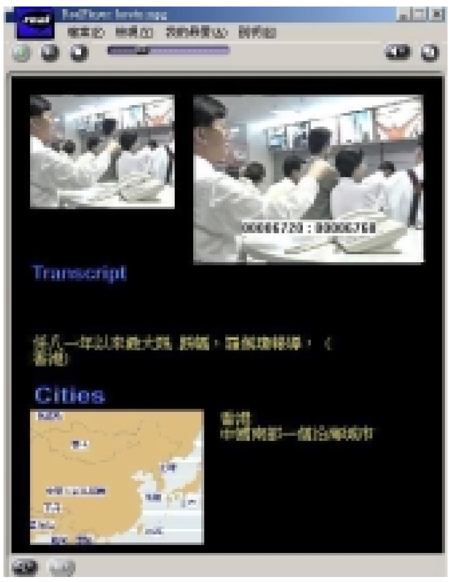
Information: [✓ Cities] [✓ People]

Use Template [✓ Use Own XSL] E:\pat\display\Demo2\new

Files generated:
combine.xml
combinePeop.xn
demo.smil
mandetail.rt
mapdetail.rt
transcript.rt

Figure 7.3.9 User interface of our transformer

Apart from the selection of templates, we also implemented two types of additional information that can be used to enrich the SMIL presentation. They are information of major cities and famous people. User can select the check box of the additional information that they are interested in. For example, they can select only the geographic information, as shown in Figure 7.3.10; or only the biographic information as shown in Figure 7.3.1; or both types of information, as shown in Figure 7.3.12.



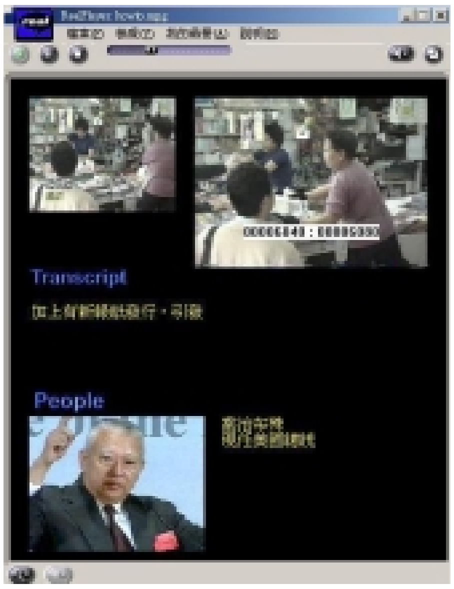


Figure 7.3.10 (Left) Geographical information selected to be displayed.
Figure 7.3.11 (Right) Biographical information selected to be displayed.

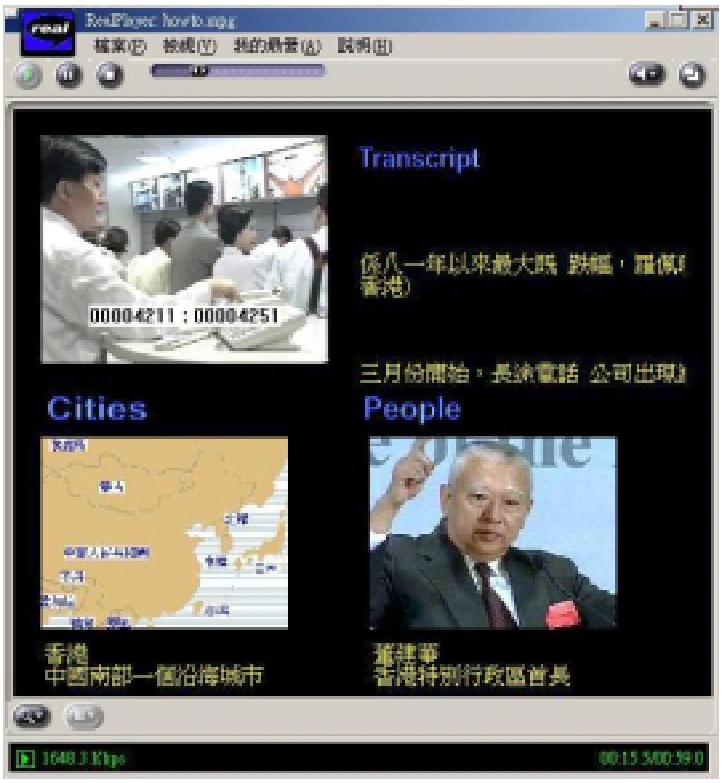


Figure 7.3.12 Both geographical and biological information selected to be displayed.

In the user interface of our transformer, we provided a listbox showing the files generated for the SMIL presentation our user requested. Users can also playing the SMIL file and all the realtext files by double clicking the items on the listbox. After selection a file name and double clicking, a realplayer will start up and play the file selected. This makes XVIP user-friendlier.

8. Implementation

After knowing all the background of the project, the extraction techniques (scene change, VOCR, face detection and speech recognition), the techniques of XML, knowledge enrichment and the way of presentation, it's time to describe the detail implement of the tool.

8.1 Overview

In our final year project, we are trying the built a XML-based Video Information Processing system (XVIP) to extract some of the information from the video and save them as XML XVIP is also acts as an editor. After extracting the data, users can edit the information with XVIP, they can add more or eliminate data of the XML through XVIP. Then knowledge enrichment can be done based on the information provided by the database. Lastly, the XML generated can be transformed of SMIL with XSL for presentation.

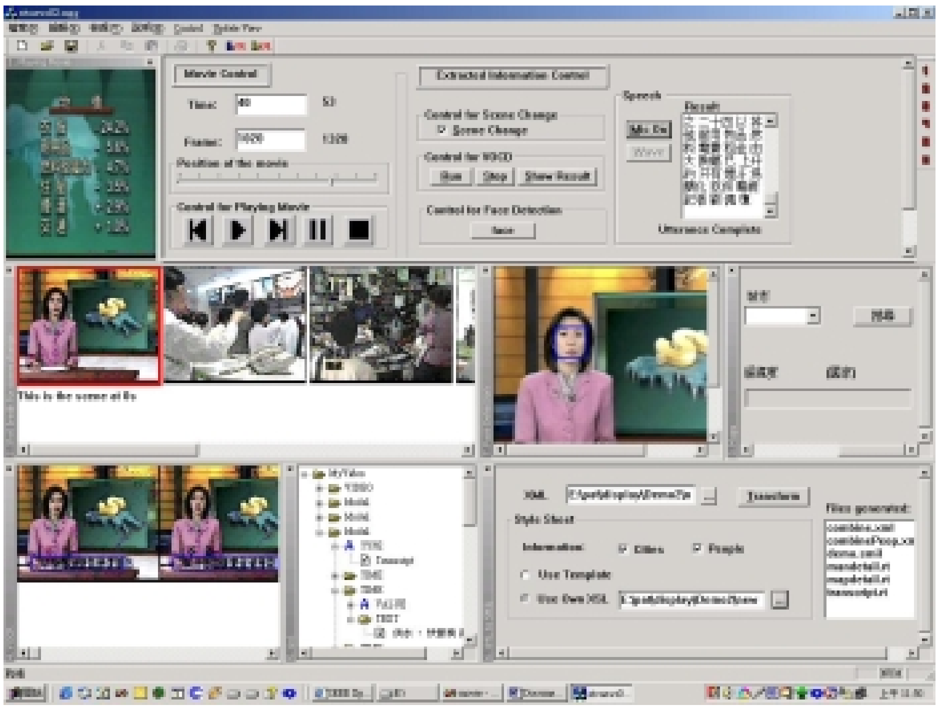


Figure 8.1 interface of our tool

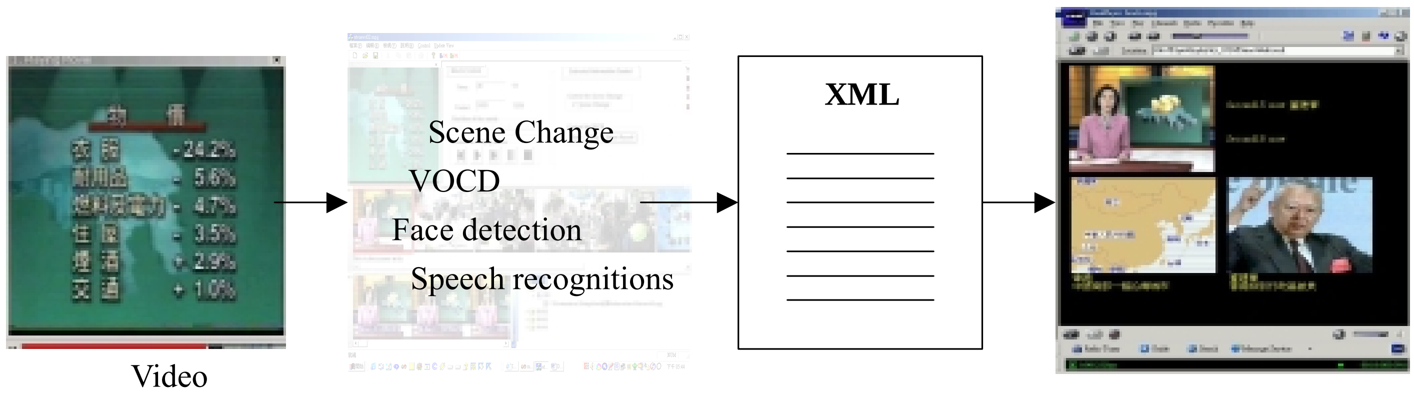


Figure 8.2 Overview of the tool

This tool is implemented with Visual C++ and Direct Show.

8.2 Programming Platform

Before describing the detail of the implementation of the tool, let’s brfely introduce the basic knowledge of the program platform and the software we used All of the programs are written in Microsoft Visual C++ with DirectShow

8.2.1 Microsoft Visual C++

Microsoft Visual C++® is the most productive C++ tool for creating the highest-performance applications for Windows® and the Web. Nearly all world-class software, ranging from the leading Web browsers to mission-critical corporate applications, is built using the Microsoft Visual C++ development system. Visual C++ 6.0 brings a new level of productivity to C++, without compromising flexibility, performance, or control.

Visual C++ is object-oriented it is an event driven programming language, when there is event is track in the interface, the corresponding message is send and the corresponding action is taken or program is run.

8.2.1.1 Benefits

- Enjoy a new level of productivity with new features that significantly reduce development time. Developers will spend less time building applications, less time coding, less time compiling, and less time debugging, while enjoying greater component reuse.

- Get blazing speed. Visual C++, already the standard for speed, now generates faster code than ever before. Visual C++ 6.0 is tuned in a number of places so that developers can build the fastest, smallest components and applications possible.

- Develop for Windows and the Web. Now you can easily build the smallest ActiveX controls, take advantage of the latest user interface enhancements from Microsoft in your applications, and create multimedia-based highly interactive, Dynamic HTML pages.

8.2.1.2 Features

- **IntelliSense Technology**
  Increase productivity and greatly simplify coding with auto list members, parameter info, type info, code comments, and complete word that eliminates the need to memorize complex syntax, parameters, and component properties.

- **Edit & Continue**
  Get faster turnaround with new Edit and Continue in the debugger. Developers can edit code while debugging without having to quit the debugging session, rebuild, restart the debugger, and return the application to the state where the problem occurred.

- **Dynamic ClassView updating**
  Easily navigate your code and save time as changes like adding a variable or method are reflected immediately in ClassView.

- **Active Document Containment**
  Easily and seamlessly integrate the functionality of Active Documents from Microsoft Word, Microsoft Excel, and other applications.

- **Composite Controls**
  Get state-of-the-art ActiveX development with new Composite Controls. Easily reuse any ActiveX control within your own.

- **Faster compiler throughput**
  Compiler throughput on debug projects is as much as 30 percent faster, and on non-debug projects as much as 15 percent faster (without sacrificing any optimizations).

- **Faster MFC applications**
  MFC runs faster with better granularity and less code overhead in dynamic link scenarios.

- **Optimizing compiler keywords**
  Aggressively optimize with new compiler performance keywords.

- **Delay Load Imports**
  Speed applications by deferring DLL loading until necessary for continuing execution.

- **Dynamic HTML support**
  Build multimedia-rich, highly interactive dynamic HTML Web pages that fully exploit Internet Explorer 4.0 and capitalize on your existing skills and code.

- **Multiple monitor support**
  Divide and conquer with the multiple monitor support. Run an application on one screen and debug on another (Requires Windows 98 or Windows NT 5.0).

- **Improved Features Optimizing compiler**
  Take advantage of industry-leading performance and the number-one optimizing compiler for the smallest, fastest executables.

- **Microsoft Foundation Classes (MFC)**
  Build world-class applications with the most robust, productive, and widely used application framework available for Windows.

- **Active Template Library (ATL)**
  Quickly create the smallest, most scalable server-side components and ActiveX controls with the Active Template Library (ATL).

- **Wizards**
  Save time with numerous new and enhanced wizards throughout the product for MFC, ATL, and more.

8.2.1.3 Microsoft Foundation Class

Microsoft Foundation Class Library (MFC) was created to make programming in Windows easier. As an object-oriented wrapper for Win32, it automates many routine programming tasks (mostly passing references around). Paradigms like the document/view architecture were added to automate even more tasks for the programmer, but in the process, some control was taken away. MFC provides many data structure and function library, which can help us in programming.

Using the MFC with the Visual C++ to implement the tool, and we can take advantages from them.

8.2.2 DirectShow

Microsoft® DirectShow® (formerly known, in part, as ActiveMovie) is a multimedia architecture for streaming media on the Microsoft® Window® platform developed by Microsoft. It is part of Windows 98, Windows 2000 and Internet Explorer, and available separately for free download from Microsoft. It is based on the Component object Model (COM). DirectShow provides for high-quality capture and playback of multimedia streams. The streams can contain video and audio data compressed in a wide variety of formats, including MPEG, audio-video interleaved (AVI), MPEG-1 Layer 3 (MP3), and WAV files. Capture can be based on either Windows Driver Model (WDM) or legacy Video for Windows (VFW) devices. DirectShow is integrated with DirectX technologies so that it automatically takes advantage of any video and audio acceleration hardware to deliver the highest possible performance.

8.2.2.1 Overview of DirectShow

The following diagram shows the relation between an application, the DirectShow component, and some of the hardware and software components that DirectShow supports.

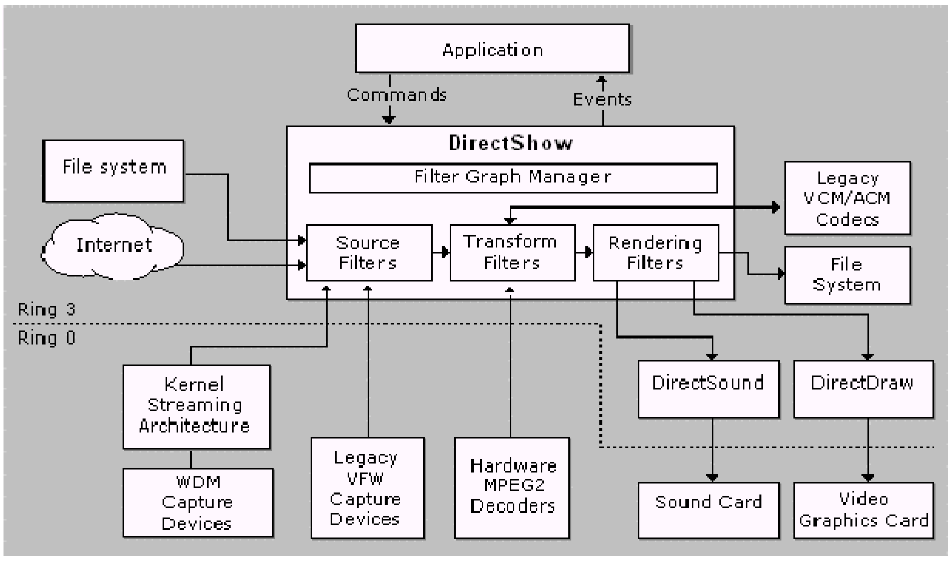


Figure 8.4 relation between an application

DirectShow enables applications to play files and streams from various sources, including local files, local CD and DVD drives and remote files on a network.

DirectShow also has native compressors and decompressors for some file formats, and many third-party hardware and software decoder are compatible with DirectShow. Playback makes full use of DirectShow hardware acceleration and DirectSound capabilities when the hard supports it.

8.2.2.2 DirectShow Application Programming

At the heart of the DirectShow services is a modular system of pluggable components called filters, arranged in a configuration called a filter graph. A component called filter graph manger oversees the connection of these filters and controls the stream’s data flow.

DirectShow is set up with the ideas of a number "filters" joined together to create a "graph". It divides the processing of multimedia tasks such as video playback into a set of steps known as *filters*. Filters have a number of input and output *pins*, which connect them together. The generic design of the connection mechanism means that filters can be connected in many different ways to achieve different tasks, and developers can add their own effects or other filters at any stage in the graph. DirectShow *filter graphs* are widely used in video playback (in which the filters will provide steps such as file parsing, video and audio de-multiplexing, decompressing and rendering) as well as being used for video and audio recording and editing. Interactive tasks such as DVD navigation are also successfully based on DirectShow.

Here's a visual representation of the graph an filters using a utility that comes with the DirectXMedia SDK, called GraphEdit

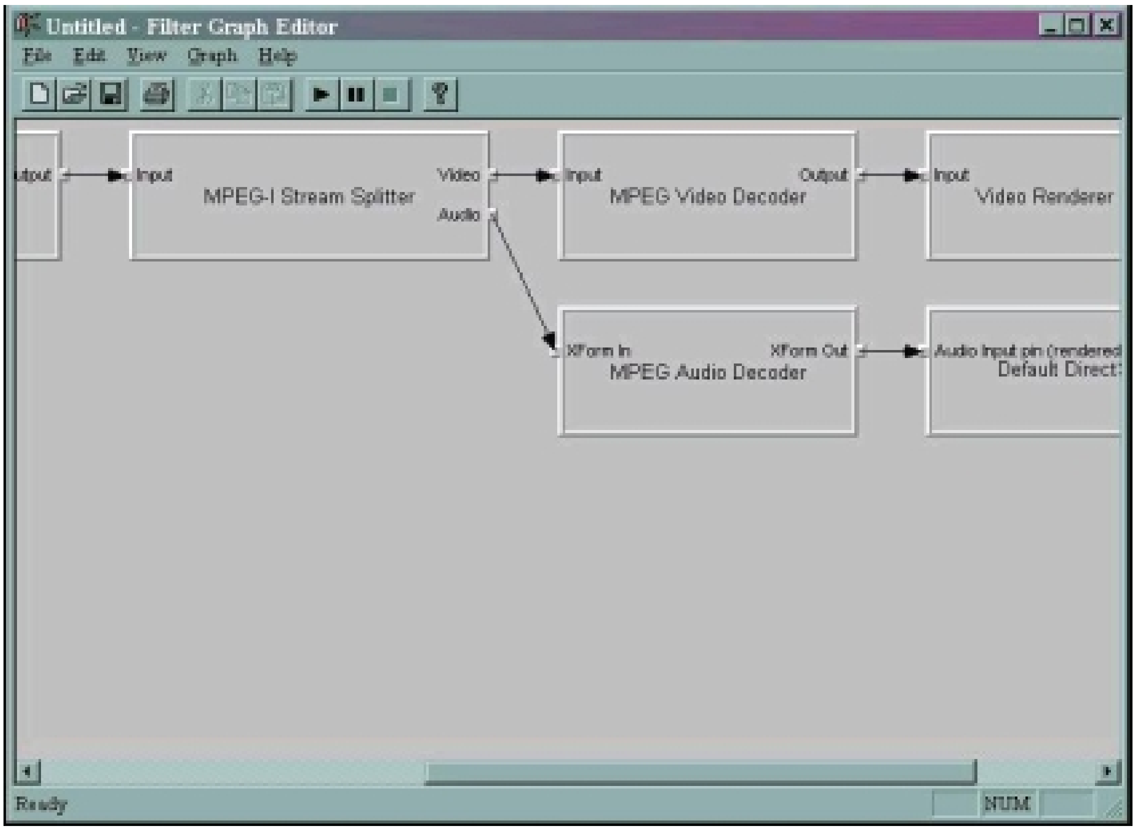


Figure 8.5 Screen of GraphEdit

Each box represents a filter. Arrows connecting boxes represent the output of one filter being passed to the input of another filter. Arrows also show the flow of data in the graph.

8.2.2.2.1 Filter

DirectShow applications use a graph of filters to process multimedia data. Typically, the multimedia data flows through the graph from a source file to a destination device or window with the application controlling the operation, but in some cases the application itself will want to supply or receive the data rather than just providing control.

Below is a typical playback graph for an mpeg movie file. It shows

- a source filter, responsible for reading data from a file (or a URL)
- a parser filter that separates out chunks of audio and video data and supplies them to the appropriate decoder
- decoders for audio and video that decompress the data
- *'rendering filters'* for both audio and video that take the decompressed data and draw or play it.

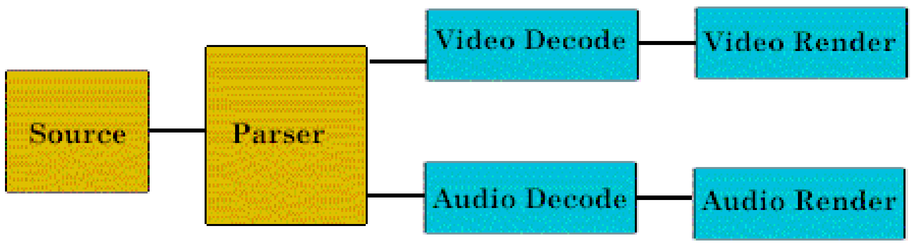


Figure 8.6 DirectShow filter graph

Filters are COM objects that provide a set of COM interfaces, principally IBaseFilter. They can have any number of input and output pins (sub-objects obtained through IBaseFilter::EnumPins) which support a set of pin interfaces including IPin. When two filters connect their pins together the pins agree a *media type* that defines the data to be exchanged, and obtain interfaces on each other which they will use to exchange data.

To play back a file using DirectShow, an application would create a *filter graph manager*, and ask this object to create a graph for the file. The application can then use the IMediaControl interface to start and stop playback of the file.

8.2.2.2.2 Filter Graph Manager

A DirectShow application interacts with a *filter graph manager* . This provides a central point of control for the application and is responsible for building and controlling the graph, and distributing state change information to the filters.

The *filter graph manager* is a COM object created by CoCreateInstance. It will build the graph using IGraphBuilder either by passing in a file or by directly inserting and connecting filters. Filters can be connected indirectly, in which case the filter graph manager will use transform filters as necessary to transform the data type so that a connection can be made.

In addition to basic graph-building services, the filter graph manager also acts as a single point of control. The application queries for control and status interfaces from the filter graph manager, which will implement the interfaces by calls to the graph. For example, IMediaControl::Run is implemented by calling the Run method of every filter (in a particular order: the threading rules of DirectShow which are designed to ensure that filters do not have to have their own thread require state changes such as Run to be done in an upstream order or a deadlock may result: one reason for letting the filter graph manager do it).

This mechanism is extensible via *plug-in distributors.* You can register a COM object as a distributor for any interface not currently supported by the filter graph: when the application queries for that interface on the filter graph manager, it will look in the registry for a Distributor key in the registry and will load that object. The distributor is then responsible for implementing its interfaces by calls to the filters in the graph. This provides an extension mechanism so that any filter can implement custom interfaces that are then exposed straightforwardly to the application, even if that application is Visual Basic. All the standard control interfaces are implemented by plug-in distributors in DirectShow. Note that although they are in theory replaceable, in practice you are unlikely to survive replacing any of them as they are all interwoven.

8.2.2.2.3 DirectShow Application

DirectShow application typically performs three basic steps, as the following diagram:

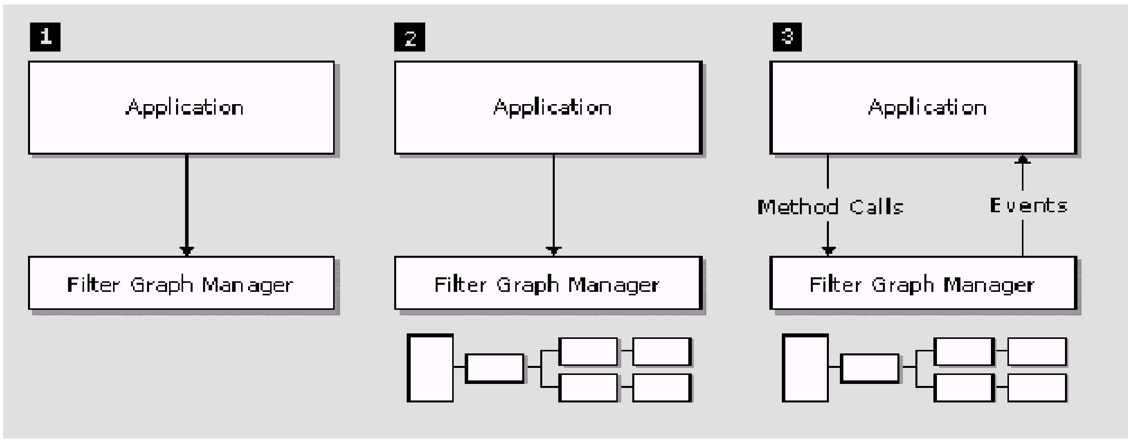


Figure 8.7 DirectShow application typically performs three basic steps

1. Creates an instance of the filter Graph Manager.
2. Uses the filter graph manager to build the filter graph.
3. Control the filter graph and responds to events.

8.3 Implementation of the tool

After have a background of the basic knowledge of the programming environment, lets have a detail look of the implementation.

As it is implemented as a multi-modal tool, any extraction techniques can be easily added to this tool. In this project, scene change, VOCD, face detection, speech recognition are developed and added to XVIP.

The multi-modal tool is consisted of several parts and each of them and implement as a view in a docking window. View is a class in Visual C++ for displaying content, including text, picture, dialog box, ActiveX control and any software that can embedded into the Visual C++. Different docking window need a View for displaying its own content. Different View is chosen for different docking window according to the purpose of the window.

8.3.1 overview

The following is the overview of the tool.

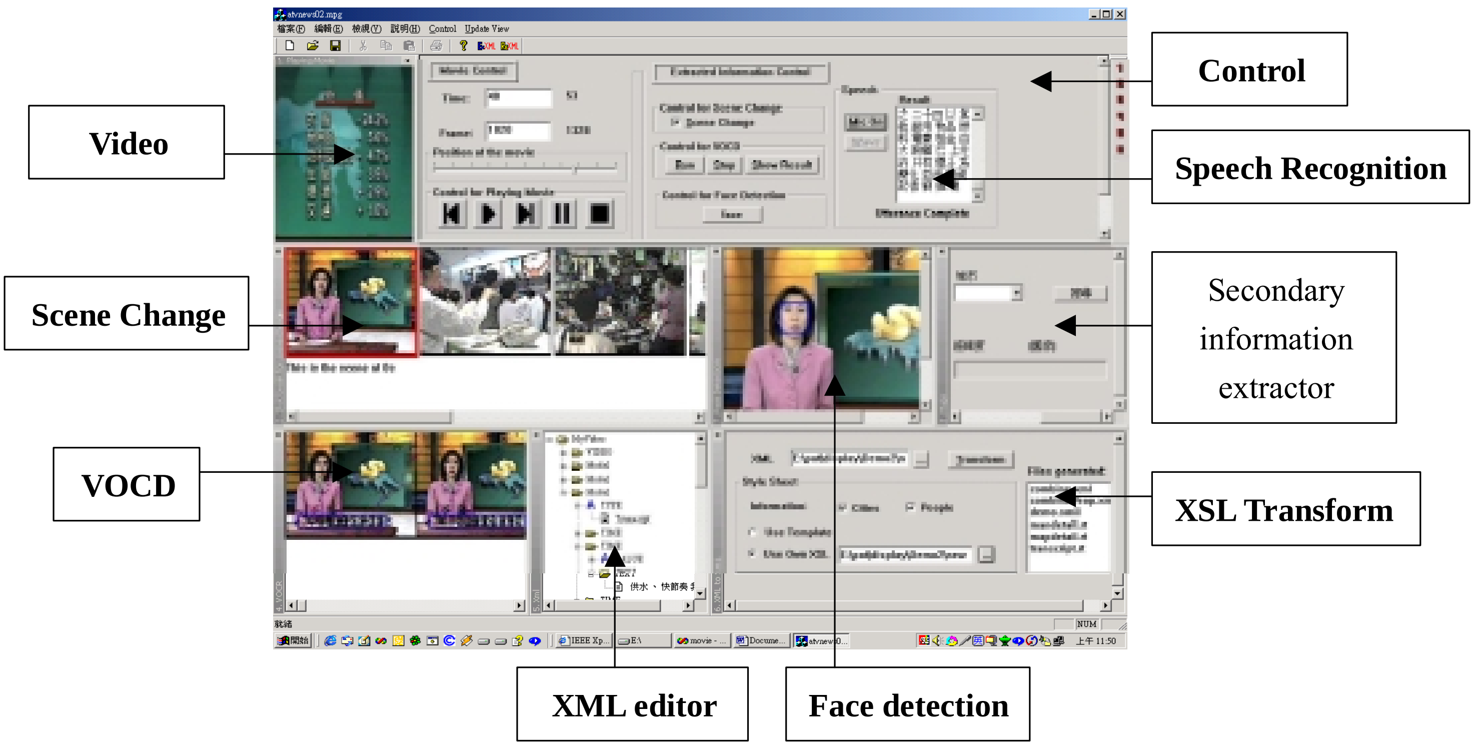


Figure 8.8 View of the interface of the integrated tool

There are nine main components in the tool,

1. Modified video Player from DirectShow
2. Control
3. Scene Change
4. Video Optical Character Detection (VOCD)
5. Speech Recognition
6. Face detection
7. XML editor
8. Secondary information extractor
9. XSL transformer

8.3.2 Implementation

8.3.2.1 Docking Window

The multi-modal tool is consist of several parts and each of them and shown in a docking window. In the tool, it need edit control, tabbed tree and windows to be docked to the frame's sides. Docking window is chosen because it can present our tool clearly and it is clear for the user to distinguish different modalities.

There is no such class in the MFC, and we have used the library from http://www.datamekanix.com/. It provides source code for implementation of docking window. The class CsizingControlBar is provided and it is an easy to use collection of classes, allowing us to focus on our application needs rather than on implementation tweaks.

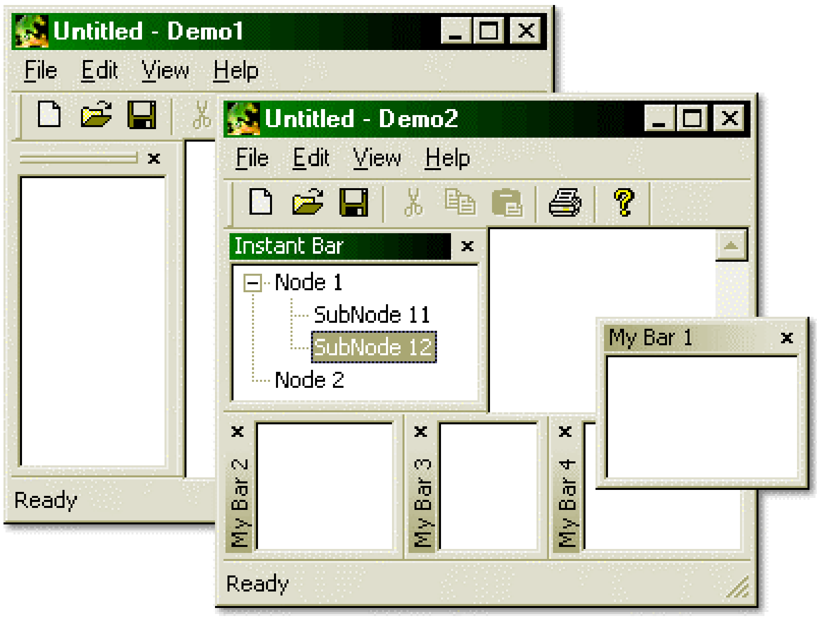


Figure 8.9 This is the original look of the docking window from the website

Using the class of the docking window, the following files are included in the Visual C++ project:

- sizecbar.h
- sizecbar.cpp
- scbarg.h
- scbarg.cpp

Two header files are included for this, "sizecbar.h" and "scbarg.h".

8.3.2.2 Video Player

The video player is implemented with Direct Show. It supports a wide variety of formats including MPEG, audio-video interleaved (AVI), MPEG-1 Layer 3 (MP3), and WAV files. Capture can be based on either Windows Driver Model (WDM) or legacy Video for Windows (VFW) devices.

The video player supports function of pause and stop a video that is playing. IT also supports resize of the window that is playing.

In this player 6 filter of Direct Show is used.

1. A source filter to read the data off the disk
2. An MPEG filter to parse the stream and split the MPEG audio and video data streams
3. A transform filter to decompress the video data
4. A transform filter to decompress the audio data.
5. A video render filter to display the video data on the screen.
6. A audio render filter to send a audio to the sound card.

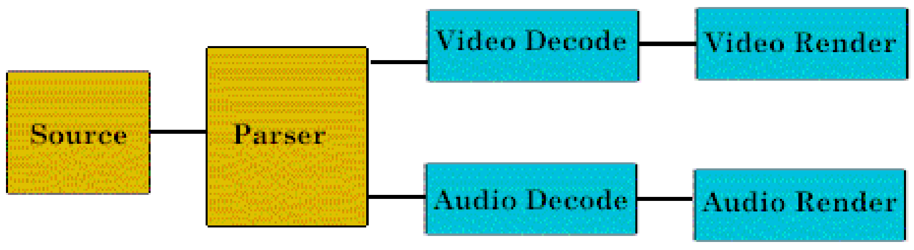


Figure 8.10. filter graph of video player

Figure 8.11 the video player implemented

The video player is embedded in the MFC program and the video player pass the handler to the main frame and display it in the docking window, no specified view need to create for it.

8.3.2.3 Control

Control is divided in four parts

(A) Input and output control

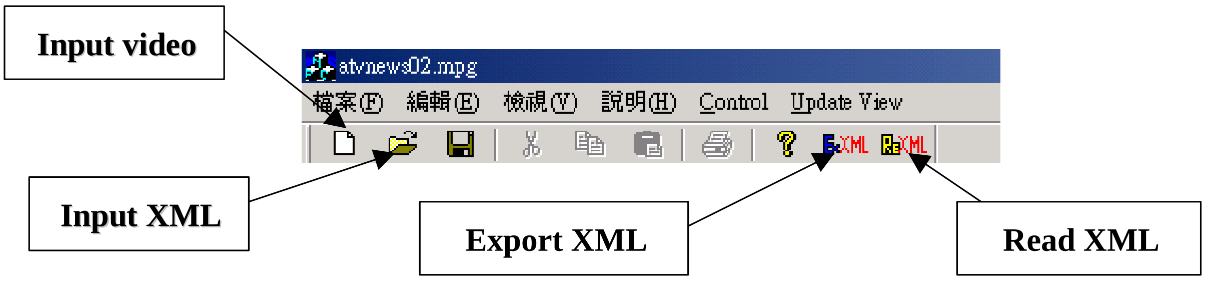


Figure 8.12 Implemented control bar

It is implemented with the class *Ctoolbar* in the visual C++. Toolbar can be created using this class. It allows the user to press the buttons on the tool bar and a corresponding message is send to the mainframe and the corresponding action is taken.

The detail of the function of the buttons is describe below:

☞ If the video is not processed before, button "🗋" should be pressed. Then different type of video file (*.avi; *.qt; *.mov; *.mpg; *.mpeg; *.m1v, *.avi; *.qt; *.mov; *.mpg; *.mpeg; *.m1v) can be chosen for processing.

☞ If the video is being processed by this tool before, there must a XML file is generated and there is the location of the video source in the XML. So, if button "🗁" is pressed, the tool can get the video file directly without asking the path of the video file and play the video automatically.

☞ When button ExXML is pressed, the information extracted from the video by the tool can be store as a XML file.

☞ When button is pressed, a XML file is read and it will display as a tree in the XML editor and it let user edit

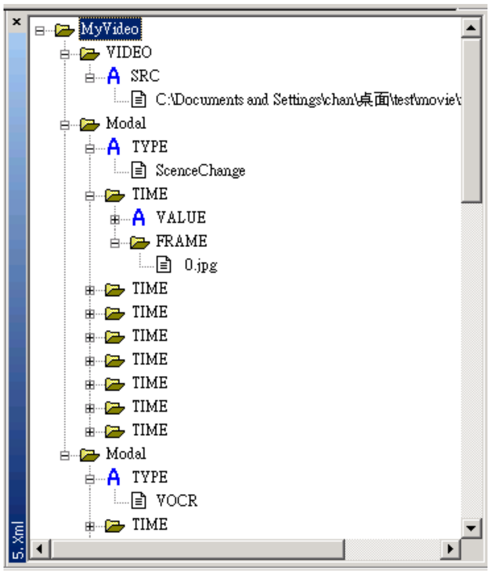


Figure 8.13 Implemented XML editor

(B) Video control

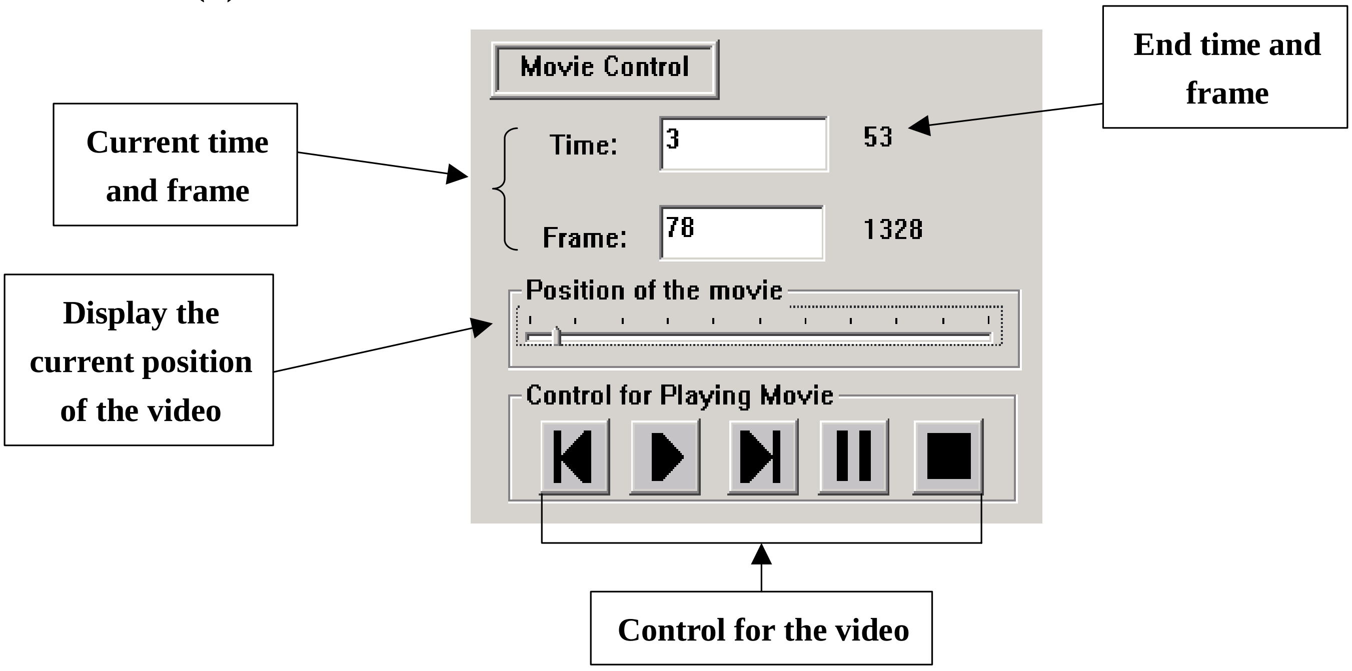


Figure 8.14 Implemented video control

The provide the information of the current position of the playing video and the user can seeking the position wherever they like by moving the slider or enter the destination time or frame in the edit box.

This is a dialog based class and create with the class CFormView. Create with this class, the dialog box will not be floating, it is attach to the docking window.

By creating the CoCreateInstance of *filter graph manager* with COM object, this provides a central point of control for the application and is responsible for building and controlling the graph, and distributing state change information to the filters.

Forward⏮, Play ▶, Backward⏭Parse ⏸, and Stop ■ are provided for the user to control the playing video. In direct Show, there are some functions for controlling the playing video. Thus, if the event of clicking the button is received, the corresponding function is called and the action will be performed.

(C) Extracting information control

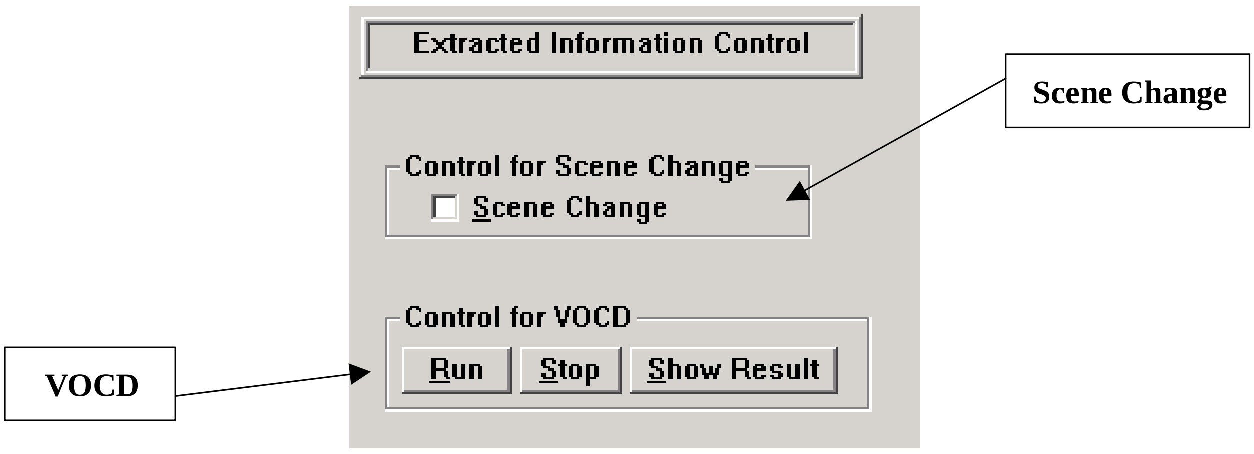


Figure 8.15 Implemented Extracting information control

These are for controlling the information extractor. If the box for scene change is tick, then the video will play the video again and perform the scene change.
It also is a dialog based class and create with the class CformView. Create with this class, the dialog box will not be floating, it is attach to the docking window.

For VOCD, there are three buttons "run", "stop" and "show result" are provided for the user to control the VOCD.

(D) Control for docking window

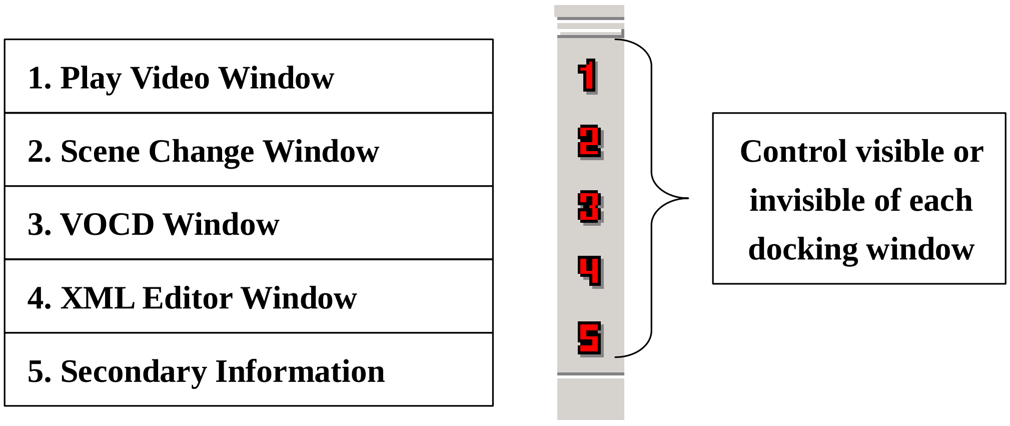


Figure 8.16 Control for docking window

As limited to the size of the monitor, not all the things in the tool can be viewed clearly. Thus, some window can be closed if they are not in used.
It is implemented with the class *Ctoolbar* in the visual C++. Toolbar can be created using this class. It allows the user to press the buttons on the tool bar and a corresponding message is send to the mainframe and the corresponding action is taken.
As the class provide a function *ShowControlBar*, it allow us the make the control bar be visible or invisible.

8.3.2.4 Scene change

A view is created for displaying the scene change. The view is the class *CscrollView*, which provide scroll bar for the user scroll along the view. The bitmap can draw on the screen by creating a memory DC that's compatible with the window's DC. With this DC, the bitmap can be stretched on the view

For detecting scene change, the Histogram difference method with a dynamic threshold is used.
For every 0.05 second, a frame in the video is grabbed by the sample grabber, which is provided by Direct Show. If the frame is grabber, a bitmap handler is given and the color histogram can be built with this bitmap handler.
If the scene change is detected, the corresponding bitmap is draw on the docking window. The information of that frame will be shown if the frame is selected and a red rectangular of enclose the frame. The Class *CPen* let us draw color line on the view. The video will play from that frame if the user double click the frame on the docking window.

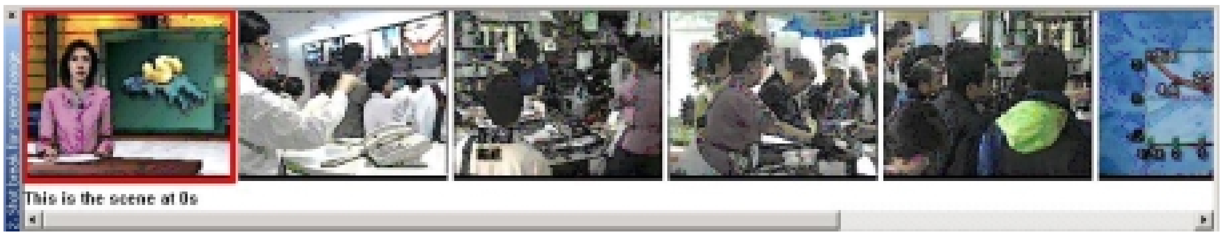


Figure 8.17 View of Docking window of scene change

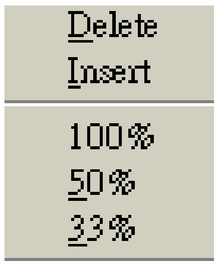


Figure 8.18 Control for Scene change

As this tool also is an editor, in the scene change's docking window, it let user to insert or delete any frame they think which is suitable. This is because the scene change detected may be incorrect and this can let user to improve the result and get more accurate information for the video.

A Menu bar is implemented with the class *Cmenu*. When the user right click the screen, a message is sent to the view and the menu bar will be displayed. When the button on the menu bar is select, the corresponding event is created and sent

8.3.2.5 VOCD

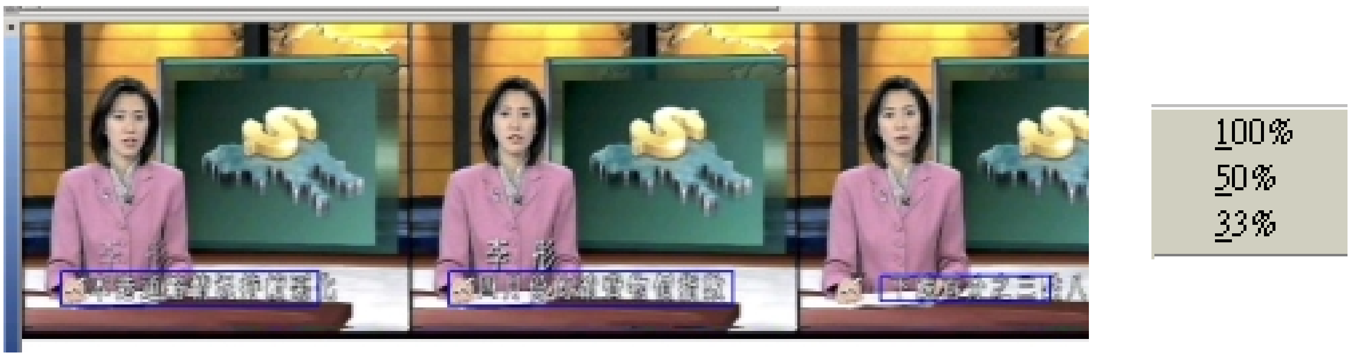


Figure 8.19 View of Docking window of VOCD and its control

The frame display on the view is same as the one in the scene change. When there is character found in a frame, the information of the frame and the place of the character appears is recorded and stored in a specific data structure. After all the frames are processed, some of the frames may be incorrect, the incorrect data is filter out. Then the corresponding frames are grabbed and display the frame on the docking window and the characters are pointed out.

8.3.2.6 Face Detection

Neural network-based method is applied to face detection. After the preprocessing the image, it is passed to the neural network to detect the location of the scene change.

A docking window is added to XVIP and the picture is draw on the scene and the face is enclosed by a blue square.

The face detection modality is added to the XVIP, the following picture shown the result of the detection. The face is enclosed by a blue square.

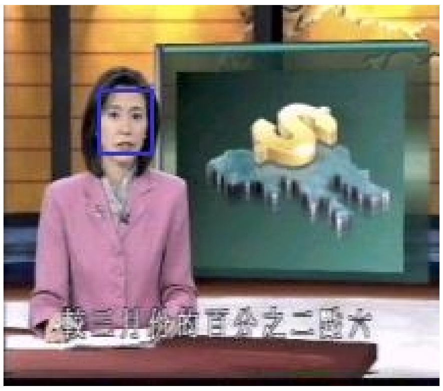
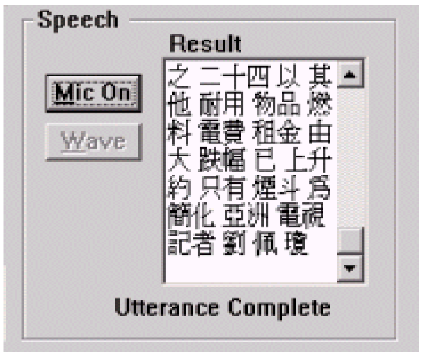


Figure 8.20 Result of the Face detection

8.3.2.7 Speech Recognition

Speech recognition is preformed by the IBM ViaVoice. After the speech engine is connected to XVIP, the text is output and an edit box is added to the interface.

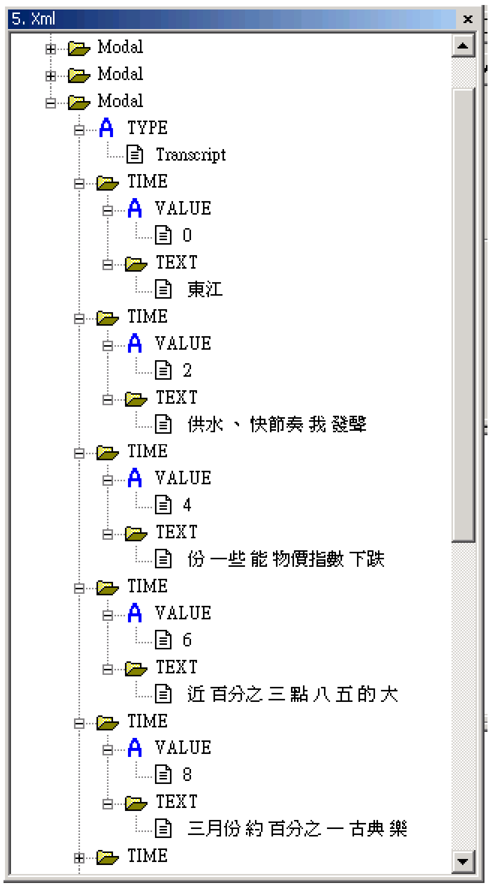


之 二十四 以 其
他 耐用 物品 燃
料 電費 租金 由
大 跌幅 已 上升
約 只有 煙斗 爲
簡化 亞洲 電視
記者 劉 佩 瓊

Utterance Complete

Figure 8.21 interface of the speech recognition engine in XVIP

After text is recognized, words are added to the XML. One more modality is added to the XML, named as "Transcript". The text is added to the XML according to the time. As the error rate of the speech recognition is quite large. The text can be insert or deleted inside the editor, shown in figure 4.4.5. However, we believe the accuracy of speech recognition engine can be improved



```
Modal
Modal
Modal
  TYPE
    Transcript
  TIME
    VALUE
      0
    TEXT
      東江
  TIME
    VALUE
      2
    TEXT
      供水 、 快節奏 我 發聲
  TIME
    VALUE
      4
    TEXT
      份 一些 能 物價指數 下跌
  TIME
    VALUE
      6
    TEXT
      近 百分之 三 點 八 五 的 大
  TIME
    VALUE
      8
    TEXT
      三月份 約 百分之 一 古典 樂
  TIME
```

by training.

Figure 8.22 interface of the editor

8.3.2.8 XML editor

By the XML parser, the XML file is read as an XML tree and hand to the tool. The XML is represented as a tree in the docking window.

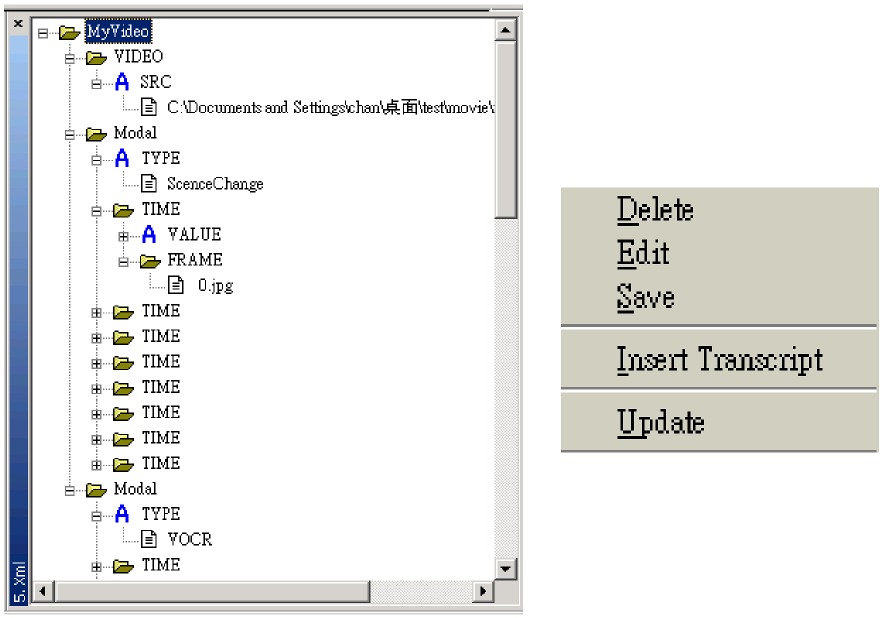
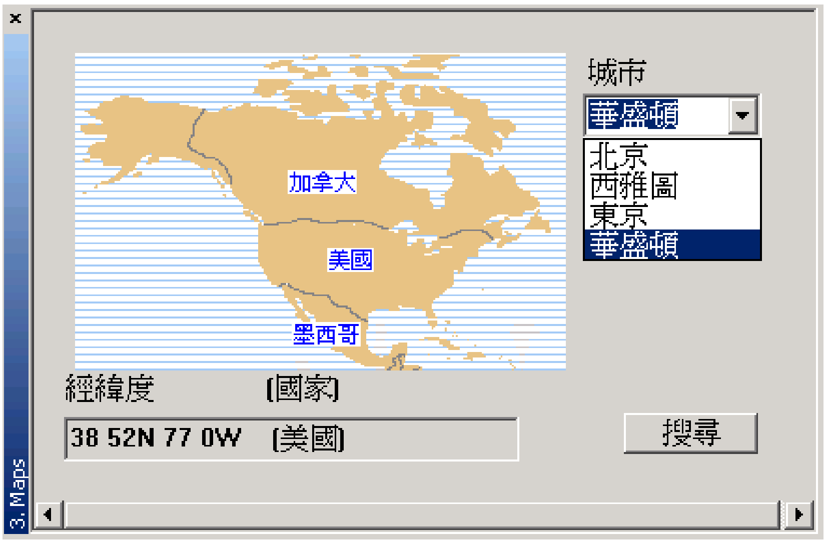


Figure 8.20 View of Docking window of treeview and its control

A *TreeView* is created for displaying a tree in a view. When the view is created, a tree handler is created automatically. Node can be added to the tree and it will be display on the view as in figure 8.20. So, each tag in the XML tree is converted to the node of the tree in the tree view.

The node can be deleted or inserted from the tree to enrich the information of the XML. The text in the node can also be edit. After editing, user can save the XML again. A Menu bar is implemented with the class *Cmenu*. When the user right click the screen, an message is sent to the view and the menu bar will be displayed. When the button on the menu bar is select, the corresponding event is created and sent

8.3.2.9 Knowledge Enrichment

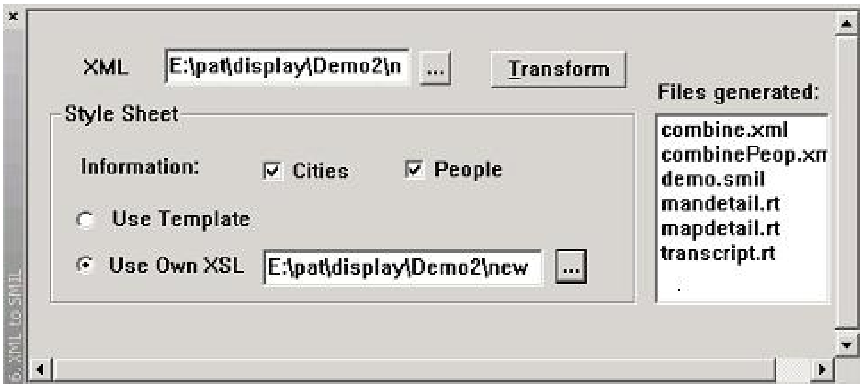


38 52N 77 0W (美國)

Figure 8.21 View secondary information extractor

The extracted information is stored in the XML in text format. This text may be the result from VOCR and speech recognition etc. They contains rich information for understand the video. We are trying to extract geographic information from this text. In our project, we extracted the names of major cities from the video. Then, the longitude, latitude and the corresponding country are listed for these cities. A map of the country a city belongs to will also be shown.

8.3.2.10 XML to SMIL transformer



XML E:\pat\display\Demo2\n

Information: ☑ Cities ☑ People

Use Own XSL E:\pat\display\Demo2\new

Files generated:

combine.xml
combinePeop.xm
demo.smil
mandetail.rt
mapdetail.rt
transcript.rt

Figure 8.22. Picture for the user interface

The XML to SMIL transformer read the XML file generated after video information extraction. Then, it will do knowledge enrichment based on the type of additional information being selected by users. Finally, it will output files for the SMIL presentation as shown in Figure 8.22. In our interface, users can also select the files produced by the list box provided to play individual files or see the SMIL presentation.

For the process of transformation, we divided it into 2 parts, which are knowledge enrichment and outputting files. For the knowledge enrichment part, a combination process that we implemented in Visual C++ did it. For the part of outputting files, it was done with the assistance of XSLT and another combination process that we implemented. The XSLT processor that we are using in our project is the MSXML 4.0, which was developed by Microsoft. After the transformation process, a set of files is output and can be played as a SMIL presentation in Figure 8.23.

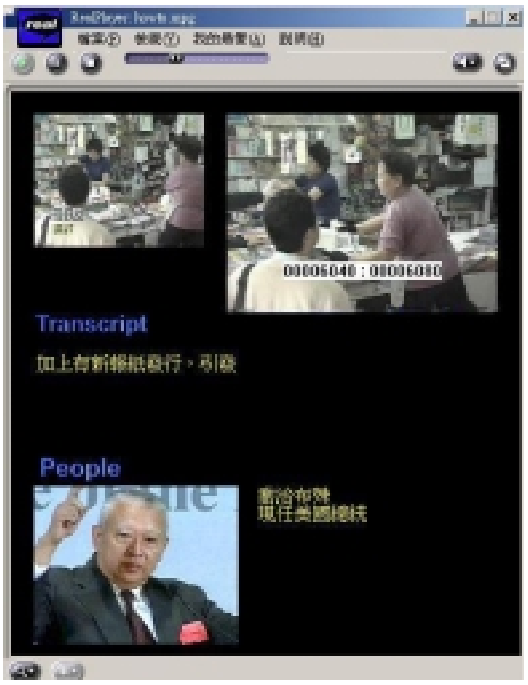


Figure 8.23 Picture for the SMIL presentation

9. Problems & Solutions

Problem 1

We should familiarizes the tool Visual C++ and DirectShow which are we are not familiarize before.

Solution

We have spent 1 month to read some reference book and try to familiarize with them. Also, we followed some online Visual C++ tutorial to implement some basic program to get more experience to use this language and program.

Problem 2

We want to get the control of the video play and it let us to do the information extraction on a video. At first, we try to use the player provided by ActiveX control in Visual C++, but we cannot get the handler.

Solution

We try to study DirectShow and implement a video player by this. By creating the CoCreateInstance of *filter graph manager* with COM object in DirectShow, this provides a central point of control for the application and is responsible for building and controlling the graph, and distributing state change information to the filters. After getting the control, we can start to implement the extraction tool.

Problem 3

The increasing number of different techniques affect the design of an "open" digital video library system in a great extends. How can the digital library being easily scaled up in terms of adding new extraction component is also very challenging. Whatever a new extraction method is developed, at the same moment; it would imply a series of new indexing and presentation functions to be added.

Solution

Modal Concept is introduced to the project. We define the modality is a domain or type of information that can be extracted from the video. Examples are the text by speech recognition and human identity by face recognition. The set of functions that the modality supports the digital library applications is called the modal dimension. Typical processes are video information extraction, indexing and presentation. For different modalities, the requirements for the whole series of library functions are totally different. Compare the query input for a text search and a face search where former is the string but the latter is picture. The text indexing method is certainly different to the face indexing method. Furthermore, at the front end of the presentation client, a text visualization tool is very different from the human face visualization tool in layout, content and presentation styles.

By introducing the modal concept, the integration interface and information exchange of different modality (dimension) can be easily identified. This provides an efficient way to adding different new modalities into the system without losing the flexibilities.

Problem 4

Different video information extraction techniques need to be included in the tool, it is difficult to display all of them in one window and it is not clear for the user.

Solution

Docking Window is introduced to the tool. Each component is implemented in the view class in the Visual C++ and attach to the docking window. This makes the interface more clear and user friendly.

Problem 5

Each component in the tool is controlled by its own view, and other view cannot get the handler of others' view. So different cannot communicate.

Solution

We try to use global variable for different component to communicate. At different time, different can read or write the global variables. However, we believe that it is not a good solution to solve this. We will try to find a better solution.

Problem 6

As mention in problem 3, view's control cannot be get by others, when the information is extracted by the mainframe, it cannot update other view to show the result immediately.

Solution

When we know the information is ready, refresh the view by clicking the corresponding docking window. We also know it is not a good solution but it does not affect the performance of the tool, we try to find a better solution in the future time

Problem 7

In doing scene changes, we have to build histogram for the frames and compare them for consequence frames. Doing the comparison, it takes time for us to tune the threshold in order to get a good result.

Solution

We introduce dynamic threshold to scene change, it improve the performance of scene change detection and do not need to tune the threshold.

Problem 8

It has great freedom on designing the tags and format of our XML file. We need to familiarize with XML

Solution

We have to read many papers, consider our future work and the flexibility in order to design a good format.

Problem 9

We do not have much knowledge on geography.

Solution

It takes time for us to collect the information for the major cities to build the database. We have to search the latitude, longitude and other information for each major city.

Problem 10

There are a number of components integrate together in our tools. Each component includes a number of classes. This makes our program large and complicate. We have to keep in mind what classes belong to which component and it is not easy for us to remember the functions of all the classes.

Solution

We still try to find out a good solution.

Problem 11

The speech recognition engine IBM ViaVoice is collapsed and cannot do the speech recognition.

Solution

It collapsed because some callback function is missed and need to be implemented by us. After the callback function is implemented and the

Speech recognition engine is reinstalled, the problem is solved.

Problem 12

The face detection result is not accurate. The coordinate given out is not the face position

Solution

We miss understand the coordinate output by the neural network. We think that the output point is the left upper corner but after read the detail of the specification of neural network, it was the point of the center of the face.

Problem 13

The XML with the Chinese character cannot read by the MSXML

Solution

The tag <?xml version="1.0" encoding="Big5"?>is added to the XML generated by XVIP and the problem is solved.

Problem 14

We do not know much about SMIL before doing the project. We just know it is a language that can integrate different kinds of media, but we do not know its structure and how to code it. Also, we found that very few books teaching about SMIL in library and bookshops.

Solution

We spent some time to search some useful tutorials about SMIL on web and we read the w3c specification in order to build some simple examples with SMIL. In this process, we gained more and more experience on writing SMIL.

Problem 15

The early design and implementation of our transformer have a lot of disadvantages, such as program-dependent, not extensible and not user friendly.

Solution

We abandoned our first transformer and try to think of other approach. Finally, we found that with the assistance of XSLT can eliminate the disadvantages on our first design.

Problem 16

Though we know that XSLT is a language that can be used to transform XML document, we do not know the details of its functions and how to implement it.

Solution

We spent some time to study XSLT. We borrowed some books from library and read the w3c specifications on web to make us understand more on it. Finally, we are getting more familiar with its syntax and can code it.

Problem 17

We used the Microsoft XSLT command utility in our early testing stage. However, this command utility is not for coding in Visual C++. We had some difficulties in finding and using a XSLT processor in our project.

Solution

We learned from web that MSXML could be used as XSLT processor in Visual C++. Then, we read the MSDN and searched some sample code in using the MSXML on web. Finally, we could use it in our own Visual C++ project.

Problem 18

We found that the XSLT has some limitation. For example, it can only read one XML file and one XSL file, then produce one output file. This makes it impossible to read more than one input files while doing the transformation. However, in producing the realtext files that contains additional information, we have to read more then one input files (both the XML file from video and the data file).

Solution

We try to do knowledge enrichment before using XSL to do the transformation. For example, we can combine the data file with the XML file from video and output a new XML file. Then, we can use these newly created XML file as the input file in the XSL transformation.

Problem 19

A number of files are needed in the SMIL presentation. However, each time the XSLT process can only generate one output file. Also, the input file and stylesheet are different for different media objects.

Solution

We try to carry out separate XSL transformation for each media object and prepare separate stylesheet for each of them.

Problem 20

The SMIL file to be generated requires different input files providing it information. However, XSLT can only read one input files each time doing the transformation

Solution

We try to do transformation a few times to generate the required results for different media objects and the layout of the presentation. Then, we write a combination process to join all these information and create a completed SMIL file as output.

Problem 21

The files created by our transformation process should be run using a SMIL player, like realplayer. This makes it inconvenient to play, as users need to open the folder and realplayer in order to see the presentation created.

Solution

We listed out all the files produced in the user interface. Also, we implemented the functionality that the realplayer process will be called and run the file being selected in the user interface.

10. Project Progress

Month	**Tasks completed**
May 2001	Get familiar with the programming environment in Visual C++ and study the basic idea on informedia and Digital Video Library.
June 2001	Build a media player using Visual C++, which include the functions of play, stop, pause and start playing from any specific moment.
July 2001	Study the techniques on building histogram and doing scene changes. Implement this technique to obtain the shot breaks from the video being played.
Aug 2001	Study the techniques on doing Video Optical Character Detection. Integrate this module into our system, so the captions in the video being played can be detected.
Sep 2001	Study XML and design a suitable format for our tools. Data extracted from the video by the previous steps are stored into a XML file and our tools can export it.
Oct 2001	Study the tree structure presentation in Visual C++ and method for displaying an XML file by tree structure. Implement a tree display for XML file in our tools and allow user to edit that XML file on this display.
Nov 2001	Do knowledge enrichment base on the data stored in the XML file. Names of major cities are extracted from the file and details of the extracted cities are shown with a map displayed. This new information gained is added to the original XML file to enrich it.
Dec 2001	Study XSL and SMIL and design a suitable XSL that transform XML to SMIL for presenting the element in the XML. In this stage XML is transform by MSXML and without knowledge enrichment
Jan 2002	Study the principle of speech recognition and different speech recognition engine and their algorithm. ViaVoice is chosen and added to XVIP to perform speech recognition. Implemented Design 1 of XSLT, which is program dependent. The transformer is added to XVIP
Feb 2002	Study the principle of face recognition and the algorithm used by informedia of CMU. Study the preprocessing part of face recognition. Evaluate the Design 1 of XSLT and start to design a better version of the transformer

Mar 2002	Implemented the preprocessing part of face recognition and connected the neural-network dll of CMU to XVIP to do face recognition. And implement a better interface for it. Finished the design of Design 2 and implemented the new transformer
Apr 2002	Improved the interface of XVIP Improved the interface of SMIL Writing the report

11. Contribution of Work

11.1 Introduction

In this section, it is going to state the contribution of work of my part and the knowledge gained through this final year project.

For our final year project, it can main mainly divided into two parts that is information extraction and post-processing the extracted data. For information extraction, several extraction techniques are included into XVIP with multi-modal concept. They are: shot break, face detection, video optical character detection and speech recognition. Then all extraction information is integrated into XML with our own schema. For post-processing part, it included secondary information extraction and transformation (from XML to SMIL for presentation purpose).

I was focusing on the first part that is information extraction and I also have taken parts of the work in the post-processing part. And my partner (Table) had focused on the post-processing part. And the detail will be stated in the following part.

11.2 Preparation work

Before implementing the system, it is necessary for us to have some background knowledge about the project and some technical skill should be learnt. I have studied the current digital video library to find out what techniques they have applied for extracting information from video and storing the data.

After we have decided to use Microsoft Visual C++® and Microsoft® DirectShow® to implement the system XVIP, I have studied Visual C++ and MFC programming skill and studied the detail of DirectShow.

After I have enriched the background and have better programming skill, we started to implement the system.

11.3 Information Extraction

As me and Table are not familiarize with Visual C++ and DirectShow, we had developed the video player with DirectShow which is for inputting the video to our system together, which is the primary step for building our system. Then we had implemented a control for the video player.

Next, we started to implement the first extraction technique – shot break in our system. We tried to use the color histogram difference method to detect the scene change in the video. After that, we had worked separately; I focus on extraction techniques and Table focus on the post-processing the extracted data. Here I am going to stated the extracted techniques that I have applied to XVIP.

1. **Scene change**

   As stated in before, we had used the color histogram difference method to detect the scene change for the video and then I tried to grape the frame of the scene change for the video. As the frames are saved as bitmap and this it too large for storage, I tried to convert the saved bitmap files to jpeg, which have smaller file size.

   After the scene change is detected, I had studied how to draw the result onto the screen. I had studied the view class in MFC and draw the result onto the view.

   To improve the performance of scene change, I had applied the one-dimensional entropy threshold to the histogram difference method. This improves the result of the scene change. As the result is not prefect, I tried to add some edit function, including insert and delete, to edit the result.

   Then, the extracted information is added to the XML according to the time of the shot break with the modality name "Scene Change".

   To improve the interface, I had added the docking window to the system to make it looks tidier.

2. **VOCD**

   The dll used in VOCD is implemented by Informedia of CMU but their version is for detection the English character only. So, if directly apply the dll to XVIP, the accuracy would be very low.

   To make the detection engine fit to XVIP, Table and me tried to modified the parameters for VOCD and make it to be suit for detecting Chinese character. After that, I connected the dll to XVIP. The dll will output the coordinate of the text area of the frame. With this coordinates, I tried to draw rectangles on the frame to indicate the position of the text and the frame are displayed onto the view in the system.

Then, the extracted information is added to the XML according to the time of the character detected with the modality name “VOCD”.

3. **Face detection**

   After studying the face detection techniques used by Informedia of CMU, I decided to connect the neural network they implemented to the system XVIP. Before that, the preprocessing part of the face detection should be implemented, that is extracting a 20 by 20 pixel image from the frame and do the equalization to reduce the background noise. After the preprocessing part is connected to the neural network. The detected face region will be output and the result is displayed on the screen.

   As more that one face can be detected within a frame, more than one square will be drawn on the frame displayed.

   Then, the extracted information is added to the XML according to the time of the face detected with the modality name “Face”.

4. **Speech recognition**

   After studying several speech recognition engines, including CMU Sphinx and IBM ViaVoice. IBM ViaVoice is chosen to connect to XVIP to perform the speech recognition.

   With the speech recognition SDK, the speech recognition engine can add to our system. As the input file to ViaVoice should be a wave file, a wave file should be prepared before using the speech recognition engine.

   Through the speech recognition engine, the transcript of the video can be recognized and the result is shown in the XVIP. The texts are added to the XML according the time with the modality name “Transcript”.

5. **Storage – XML**

   XML is used to store the extracted information. As to use XML efficiently, I had tried to study the XML and the parser. Then DOM XML parser is chosen to parse the XML to XVIP, which parse the XML as a tree structure.

   With the multi-modal concept, 4 modalities are added to the XML, these are scene change, VOCD, face detection and speech recognition. I tried to designed our own XML schema to store the extracted information

   As the extraction techniques are not prefect, an XML editor is implemented in XVIP for the user to edit the XML after the extracted data is added to the

XML. By studying tree structure in VC++, XML can be displayed in XVIP as a tree.

11.4Post-processing information

As I responsible for extracting and storing the extracted information, I have less focus on post-processing information part. But I still had participated in schema designing, studying different XSLT and designing XSL and SMIL.

1. **Knowledge enrichment**
   After the useful information in extracted from the video, knowledge enrichment is applied to the XML generated by XVIP. I was responsible of designing the schema of the XML for knowledge and try to minimize the affect to the original generated XML.

2. **Connect MSXML to XVIP**
   To transform the XML to SMIL with XSL, an XSL transformer is need. I tried to study different XSLT and designed to use MSXML in XVIP. With MSXML 4.0 SDK, I tried to connect to MSXML to the XVIP to transform the XML to SMIL.

3. **Design simple XSL for XML**
   As I responsible to study and connect the MSXML to XVIP, I have to design a simple XSL that is suitable for the generated XML. After studying XSL, a simple XSL is designed for testing the MSXML. My partner designs the more detail XSL for the project.

4. **Design simple SMIL for testing purpose**
   As I responsible to study and connect the MSXML to XVIP, I have to design a simple SMIL that is suitable for the generated XML. After studying SMIL, a simple SMIL is designed for testing the MSXML. The more detail SMIL is designed by my partner.

11.5 Report

As state before, our project divided into two main pasts and each of us responsible for different part, we write the part of the report that we have more contribution of work. I responsible for the following part:

- Architecture of XVIP
- Extraction technique
- Scene change detection
- Face Detection
- Speech Recognition
- XML
- Implementation
- Problems & Solutions
- Project Progress
- Contribution of work
- Conclusions

11.6 Summary of work contribution

- Study current DVL, VC, MFC, DirectShow, XML, XSL, SMIL
- Implement video player
- Build docking window
- Add all the extraction techniques to XVIP with modal concept
  - Build color histogram and implement scene change
  - Tune VOCR parameter and added to XVIP
  - Implement preprocessing part of face detection, and connect the dll to XVIP
  - Study and connect ViaVoice to XVIP
- Extract wav file from the original video file for speech recognition
- Integrate all the information extracted to a signal XML
- Do most of the interface work
  - Insert / delete frame from scene change
  - Different size display
  - Let user to edit XML
  - View VOCR result
  - View Face detection result
  - View speech recognition result
- Design schema of XML
- Design the schema of XML for knowledge enrichment and add the new information to XML

- Build the XML parser
- Study tree structure in VC++ and display the XML in XVIP
- Build XML editor
- Connect MSXML to XVIP
- Design simple XSL for XML
- Design simple SMIL for testing propose
- Improve the interface of XVIP
- Writing report

11.7 Conclusion

After the final year project, I more understand how the current digital video library work and it have strengthened my programming skill. Also, I have learnt lots of new technologies and knowledge through this FYP, including VC, MFC, DirectShow, XML, XSL, SMIL and different extraction techniques. And let me keep in touch with the state of the art technologies.

12. Conclusions

We develop XML based Video Information Processing system (XVIP) with multiple functions for processing and storing information extracted from video files. It can integrate different video information extraction techniques with a multi-model concept, generate an XML file for storing the extracted information, and allow users to perform editing on the XML file exported from our system. The multi-model concept and XML format give us flexibility to integrate new techniques, do further enrichment or modification on data in the future. Finally, we implement an XML to SMIL transformer to generate different presentation templates of SMIL.

We studied current methods on doing video information extraction and presentation. There are four modalities in the XVIP system. We have implemented four of these techniques on video information extraction. These include scene changes, video optical character detection face detection and speech recognition. Then, we generated an XML file for storing the extracted information. In addition, users are allowed to do editing on the XML file exported using our tools. After that, we are doing information enrichment using the data in our XML file. Some geographic information is extracted from the text in our XML file. The result of information extracted from the current data will be used to enrich the XML file. This newly added information helps us to do indexing and searching. XML is then transform to SMIL with XSL for presentation purpose.

In this semester, we focus on how to use the XML file format we designed in this semester to do multimedia presentation effectively. We notice that the XML format allow us to fill in templates with content retrieved dynamically from a database or transformation of structured documents using style sheets (e.g. XSLT). This method can generate HTML pages on demand. It is much effective than encoding information in handwritten (HTML) Web pages. However, these text-oriented techniques may not be suitable for multimedia document generation. The current stile sheet technology cannot be applied to multimedia. Because of these reasons, new formats on multimedia presentation are developed. And we focus on using SMIL.

After this final year project, we have learnt different programming skill, including visual C++, DirectShow and it gives a chance for us to enrich ourselves by studying different Digital Video Library, extraction techniques, XML, XSL and SMIL and let us to keep in touch with the state of the art technology.

13. Acknowledgement

We would like to thank Prof. Michael Lyu, our project supervisor, for giving us valuable advice. He has provided many suggestions and comments to us throughout this project. We are much appreciated by his patience and kindness in advising us.

Moreover, we would like to thanks Edward, Technical Manager of VIEW project, who gives us ideas in our project.

14. Reference

[1] Howard D. Wactlar, New Directions in Video Information Extraction and Summarization, June 1999.

[2] Howard D. Wactlar, Michael G. Christel, Yihong Gong, Alexander G.Hauptmann, Lessons Learned from Building a Terabyte Digital Video Library, Feb 1999, IEEE.

[3] Howard D. Wactlar, Taceo Kanade, Michael A. Smith, and Scott M. Stevens, Intelligent Access to Digital Video: Informedia Project, May 1996 IEEE.

[4] Michael Christel and David Martin, Information Visualization within a Digital Video Library, June 1998.

[5] Susan Gauch, Ron Aust, Joe Evans, John Gauch, Gary Minden, Doug Niehaus, and James Roberts, The Digital Video Library System: Vision and Design, *Digital Libraries '94,* June 1994.

[6] Jong Whan Jang and I1 Kyun Oh, Performance Evaluation of Scene Change Detection Algorithms, Spring 1997

[7] Yueting Zhuang, Yong Rui, Thomas S. Huang and Sharad Mehrotra, Adaptive Key Frame Extraction Using Unsupervised Clustering, IEEE International Conference on Image Processing, 1998(ICIP98), Oct. 1998

[8] D. Lelescu and D. Schonfeld, Real-time scene change detection on compressed multimedia bitstream based on statistical sequential analysis, Proceedings of the IEEE International Conference on Multimedia and Expo, Compact Disk, pp. 1–4, New York, New York, 2000.

[9] Micha Haas and Michael S. Lew and Dionysius P. Huijsmans, Shot Break Detection and Camera Motion Classification, Oct 1998

[10] Xinying Wang, Zhengke Wong, Scene Abrupt Change Detection, IJCV(24), 1997

[11] W. A. C. Fernando, C. N. Canagaraiah and D. R. Bull, A Unified Approach to A Scene Change Detection In Uncompressed and Compressed Video, Image Communications Group, Oct 2000.

[12] Sato, T., Kanade, T., Hughes, E., Smith, M., Video OCR for Digital News Archives, IEEE Workshop on Content-Based Access of Image and Video Databases (CAIVD'98), Bombay, India, January, 1998.

[13] Sato, T., Kanade, T., Hughes, E., Smith, M., Satoh, S., Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption, ACM Multimedia Systems Special Issue on Video Libraries, 7(5): 385-395, 1999.

[14] Xiaoou Tang, Cheung Yung Chan, and Jianzhuang Liu, Handwriting Chinese m Character Recognition through a Video Camera, Lecture notes of IEG4190 of The Chinese University of Hong Kong, Nov 2001.
[15] Xiaoou Tang, Feng Lin, Shadow-resistance Stroke-tracing for Handwritten Chinese Character Recognition, Lecture notes of IEG4190 of The Chinese University of Hong Kong, Nov 2001.
[16] Micheal G. Christel, Bryan Maher, Andrew Begun, XSLT for Tailored Access to a Digital Vied Library, JCDL 2001, June 2001
[17] Vladimir Geroimenko and Larissa Geroimenko Visual Interaction with XML Medata
[18] XML Schema Language: Takinf XML to Next Level, IT PRO March/April 2001
[19] The Challenges that XML Faces, IEEE Technology News, October 2001
[20] XML's Impact on Databases and Data Sharing, IEEE Research Feature, June 2001
[21] Integrating XML and Databases, IEEE Internet Computing July+August 2001
[22] Michael G. Christel, Andreas M. Olligschlaeger, Carnegie Mellon University, Interactive Maps for a Digital Video Library, June 1999, IEEE.
[23] Howard D. Wachtlar, Carnegie Mellon University, Multi-Document Summarization and Visualization in the Informedia Digital Video Library, New Information Technology 2001 Conference, May, 2001
[24] Jacco van Ossenbruggen, Joost Geurts, Frank Cornelissen, Lynda Hardman and Lloyd Rutledge, Towards Second and Third Generation Web-Based Multimedia, May 2001
[25] Henry A. Rowley, Shumeet baluja, and Takeo Kanade, Neural Network-Based Face detection, PAMI. January 1998
[26] Henry A. Rowley, Shumeet baluja, and Takeo Kanade, Human Face Detection in Visual Scenes. Novrmber 1995 CMU-CS-95-158R
[27] Micheal J. Witbrock and Alexander G, Hauptmann, Speech Recognition for a Digital Video Library, JASIS,1996
[28] Alexander G. Hauptmann and Howard D. Wactlar, Indexing and Search of Multimodal Information.
[29] Howard D. Wactar, Alexander G. Hauptmann ans Michael J. Witbrock INFORMEDIA: NEWS-ON-DEMAND EXPERIMENTS IN SPEECH RECOGNITION.
[30] "Towards SMIL as a Foundation for Multimodal, Multimedia Applications" Jennifer L.Beckham, Giuseppe Di Fabbrizio, Mils Klarlund, W3C Multimodal Interaction Activity, 2002.
[31] "W3C Synchronized Multimedia Activity Statement",

http://www.w3.org/AudioVideo/Activity.html

[32] "Multimedia Tools for Online Tutorials: Using SMIL to Create Web-Based Instruction" Yuwu Song, Arizona State University, Internet Librarian 2000 Conference, November 2000.

[33] SMIL adding multimedia to the web, written by Tim Kennedy, Mary Slowinski Copyright ©2002 by Sams Publishing

[34]"W3C Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", http://www.w3.org/TR/REC-smil/

[35] "Building and Managing XML/XSL-powered Web Sites: an Experience Report", Clemens Kerer, Engin Kirda, Mehdi Jazateri and Roman Kurmanowytsch, Distributed Systems Group, Technical University of Vienna, IEEE 2001

[36] http://www.w3.org/Style/XSL/

[37] Book: XSLT & XPATH, A Guide to XML Transformations by John Robert Gardner, Zarella L.Rendon, 2002 Prentice Hall PTR

[38] Book: XML/XSL Programming Guide

[39] XSLT Reference from the MSXML4.0 SDK documentation