AN MPEG-7 CONTENT-BASED ANALYSIS/RETRIEVAL SYSTEM AND ITS APPLICATIONS

以 MPEG-7 媒體內涵為基礎之分析及擷取系統設計及其應用

Jin-Hau Kuo* Ja-Ling Wu** Shao-Ying Lin† Song-Wen Wang‡
郭晉豪* 吳家麟** 林劭穎† 王頌文‡

*Post-doctorate **Professor †M.S student ‡ Ph.D. student

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10617, R.O.C.

*博士後研究 **教授 †碩士班學生 ‡ 博士班學生
國立台灣大學資訊工程研究所通訊與多媒體實驗室

Abstract

In this paper, we proposed a content-based multimedia analysis/retrieval system basing mainly on the MPEG-7 defined features. Some new and specific features are also included for supporting various requirements of different applications. The proposed system is capable of extracting a variety of features from different kinds of media data, such as videos, audios, and images. The system design is well modularized, so that the tasks of system development and feature construction can be done independently and separately. In other words, adding a new feature into the system can be done without modifying the structure of the system. In addition, a user can also input some hints about what he or she is looking for into the system, and the system will search through the database and return the best matched candidates. On the basis of this well-modularized system, a lot of interesting applications, such as scene change detector, video news program analyzer, Logo verification system, fish shape image analysis and retrieval system, and home media center (a home media management system), have been built in the Communications and Multimedia Lab., Department of Computer Science and Information Engineering, National Taiwan University.

Keywords: content-based retrieval, scene change detection, MPEG-7.

摘 要

在本篇論文中，我們提出了一個以 MPEG-7 媒體內涵為基礎的分析及擷取系統。此外，為了提供多元應用的不同需求，一些特殊的特徵值也包含在這個系統當中。此系統擁有從不同的多媒體資料（如影像、聲音及圖像）中萃取多樣化特徵值的能力。由於此系統採用充分模組化的設計，因此，系統發展與特徵值萃取這兩部分的工作，可以各自獨立地發展，換句話說，當新的特徵值要加入系統中作測試時，並不會影響原本系統的設計。此外，此系統提供使用者輸入一些簡單的線索，讓使用者可以找到他們想要的多媒體資料。以此充分模組化的系統為基礎，本實驗室發展了許多相關的應用，如場景切割器，電視新聞節目分析器，商標驗證系統，魚類外形搜尋系統及家庭媒體中心（家庭媒體資料管理系統）。

關鍵詞：以媒體內涵為基礎之擷取、場景變換偵測、MPEG-7 國際標準。

1. INTRODUCTION

A great amount of multimedia data is created today, and is available everywhere in our life-in digital archives on the web, in broadcast data stream, or in personal and professional databases. In addition, with the digital capturers become more and more popular and

inexpensive, producing multimedia data becomes an easy task. People can scan pictures into a computer with a scanner, take pictures with a digital camera, or make a home video with a digital camcorder, and these daily activities make the amount of digital data growing immeasurably. Today, we can type in some keywords in an Internet search engine, and the search engine will

list vast amount of documents related to the keywords. Having experienced the efficiency and convenience of these search engines, now we not only wish to find text documents but also multimedia data such as images, audios and videos. And we want to enjoy the same accessibility as that provided by text-based search engines. This turns out to be a difficult problem, because extracting information from other types of data is much more complicated and requires higher technical supports than parsing text documents. Therefore, how to manage, find, retrieve, access and filter multimedia data efficiently and effectively becomes a critical and serious problem in digital world, and this is not only a problem to professional users, but also to family users. Exactly as expected, this area of research is very open and active in the multimedia and computer-vision communities [1~3].

MPEG-7 [4~7], proposed by the Moving Picture Experts Group (MPEG), gives an essential solution to the challenges of content management. It provides a great set of tools for describing multimedia data, and gives a standard way to follow. MPEG-7 focuses on applications that we can store (online or offline) or stream, and can operate in both real-time and non-real-time environments. Examples of applications of MPEG-7 include digital multimedia libraries, broadcast media selection, multimedia editing on home entertainment devices, searching multimedia content, managing content archives, opening up archives to the public, and semiautomatic multimedia presentation and editing. The standard is not designed for any specific application, but is designed so that it can support as broad range of applications as possible, and this means MPEG-7 providing a generic framework that can support various applications. And therefore, it facilitates exchange and reuse of multimedia contents across different application domains.

The major elements defined in MPEG-7 include Descriptors, Description Schemes, a Description Definition Language and System Tools. In MPEG-7, "data" means multimedia information that will be described basing on MPEG-7 regardless of storage, coding, display, and transmission. And a "feature" is a distinctive characteristic of the data that signifies something to somebody. Generally, there are 4 major elements in MPEG-7 [4~7]:

1. Descriptor: A representation of a feature. A descriptor defines the syntax and the semantics of the Feature representation.
2. Description Scheme: A Description Scheme is used to specify the structure and semantics of the relationships between its components, which may be

both Descriptors and Description Schemes.

3. Description Definition Language (DDL): A language that allows the creation of new Description Schemes and, possibly, Descriptors. The extension and modification of existing Description Schemes are also allowed.
4. System Tool: Tools to support multiplexing of descriptions, synchronization of descriptions with content, media delivery mechanisms, and coded representations for efficiently storage, transmission and management, and intellectual property protection of MPEG-7 Descriptions.

With these tools, we can describe a data in a standard way by building an MPEG-7 description. For getting more detailed information about MPEG-7, references [4~7] are recommended.

In this paper, we proposed a content-based multimedia analysis/retrieval system basing mainly on the MPEG-7 defined features. Some specific features are also included for supporting different applications. The proposed system is capable of extracting a variety of features from different kinds of multimedia data, retrieved in daily life. A user can also input some hints about what he or she is looking for into the system, and the system will search through the database and return the best matched candidates. The rest of this paper is organized as follows. The object and scope of the proposed system are addressed in Section 2. In Section 3, a user interface independent MPEG-7 kernel, which is constructed based on standard MPEG-7 description tools, is presented. All the other system functions such as user interface, database management, query process and database construction sequences are addressed in Section 4. In Section 5, the interface between user interface (UI) and the MPEG-7 kernel is depicted. In Section 6, some applications on the basis of the proposed system are investigated. Surely, specific domain knowledge is needed to be taken into account when exploring different applications. Some proprietary features are also included into the system to improve the system performance. This system is one of the results of the 3-year term industrial-academic collaborative project—Content Engineering: Research on MPEG-4/7 Multimedia Technologies, sponsored by the National Science Council Taiwan, the Cyberlink Corp., and the Digimax Corp., under the grant number NSC90-2622-E-002-008. The project aims at developing a multimedia content framework which covers most of the phases of the content manipulation lifecycle, including search, manipulation, management, presentation and interaction with users. The system architecture of the proposed framework and the corresponding realized modules are illustrated in Fig. 1.

Fig. 1 The framework architecture of the proposed system and the corresponding realized function modules

2. SYSTEM OVERVIEW

2.1 Scope and Object of the System

Creating a content-based multimedia analysis/retrieval system that has the ability to analyze, index, and retrieve the multimedia data from a database efficiently is our main object. The system allows a user to input some hints for what he or she is looking for, and then the system automatically searches through the multimedia databases and presents the best matches. Besides, we also treat the system as a platform for testing the efficiency and effectiveness of features used in a multimedia query process. For this reason, the system design must be well modularized and provides a user-friendly interface for adding new features.

At the beginning of the design, we address the requirements of the system as follows.

1. The programming interface mentioned above must be suitable for all multimedia types. We would prefer that the interface for adding an image feature is the same as the interface for adding a video feature.
2. The programming interface mentioned above must be application independent.
3. The system should be easy to use, and allow users to perform a query in a nature and straight way. Moreover, the system should have the ability to meet all requirements in all stages of a query. This means the system should support as many formats of multimedia data as possible, and must include the decoders for image, video and audio as well.
4. The system should provide functions of logical manipulations among the results queried by using different features. This gives the possibility of making a more precise query and enables the user to

make a query such as “find a picture with dominant color being blue and containing a shape of a fish.”

5. The system must process the query fast enough to provide real-time and interactive services for user. We are certainly not happy to wait several minutes just for finding a movie that contains intense motion. Besides, the system should be able to interact with users and provide as much information as possible to help them to complete the processes of queries.

We will make use of both standard MPEG-7 features and user-defined features, in the proposed system. They can be both added into the system through the same interface. A user can query for image, video and audio in the same system, thus the system should contain three kinds of databases, one for each type of multimedia data. As mentioned above, the system is designed to support as many formats as possible. For images, we support all formats that existed in the Magick image library, which has also been used in the Reference Software of MPEG-7. As for video, MPEG-1 and MPEG-2 decoders are included into the system. Combining the decoders into the system will provide the ability to give some real time video demos.

2.2 System Configuration

Figure 2 depicts the functional blocks of the proposed system. Roughly, the system can be divided into four parts: system manager, user interface, MPEG-7 kernel, and multimedia libraries.

2.2.1 System manager

The system manager is the heart of the system. It defines the application of the system and the rules for how to use the MPEG-7 kernel. In other words, it manages the flow of queries made to the system and controls all other parts of the system.

2.2.2 MPEG-7 kernel

The MPEG7 kernel is a set of tools needed for processing a query. Tools for feature construction, matching, encoding and decoding bit-streams the descriptors, managing database, and decoding video, image and audio, are all included in this component. It has been carefully designed so that this part is independent of applications and the user interface of the system. The tools included in the MPEG-7 kernel are like the actors of a movie, while the system manager is the director. The director tells the actors what to do and decides how the movie goes.

For making the system to be an extensible and generic platform, we chose the dynamic linking library (Dll) as the basis of developing features for the system.

Fig. 2 The functional blocks of the proposed system

All feature specific codes are put together into the Dll, so that if a programmer wants to add a new feature into system, he needs only to create a Dll with a pre-defined common interface for the new feature. In this way, the programmer needs no extra efforts to understand the flow of the system (which may be very complicated) and needs not to worry about modifying wrong portion of codes.

2.2.3 User interface

UI is the appearance of the system. It is where the system accepts questions from end users and responds with appropriate answers. Graphic interface is preferred, because multimedia data will not be presented correctly and naturally through a text or console-based interface. And the design of the user interface must satisfy the needs of every stage of a query, including navigating, searching and browsing.

2.2.4 Media library

Since the system is for querying multimedia, a set of multimedia data libraries must be included. They are responsible for decoding compressed images into RGB formats, extracting motion vectors from a video clip, or decoding a midi clip. Both system manager and MPEG-7 kernel interact with these libraries. System manager needs them to decode a compressed JPEG image file for displaying in a browsing window, while MPEG-7 kernel needs them to decompress an MPEG video bit-stream for performing the feature extraction.

3. THE MPEG-7 KERNEL

In this section, we will describe the MPEG-7 kernel in detail. The components of the MPEG-7 kernel can be categorized into two groups, feature tools and system tools. For every feature included in the MPEG-7

kernel, there is a set of feature tools associated with it. These tools are responsible for dealing with things related to a certain feature, such as extracting feature related characteristics from a multimedia data, or calculating the distance between two multimedia data measured by the same feature. And these feature tools are then packed into the DLL for that specific feature (will be detailed in Section 5). As for system tools, they are responsible for building an efficient content-based retrieval system. A multimedia data management system must include some important components, such as databases to store multimedia data, libraries for processing multimedia data, and matching lists to record the matched multimedia data. These functions are provided by the system tools of the MPEG-7 kernel.

The above-mentioned tools are not designed for any specific application; on the other hand, they provide an effective interface for using MPEG-7 standard features. In other words, any system that is designed for using MPEG-7 standard features can cooperate with the provided tools, easily. And this is the most important characteristic of the proposed system.

3.1 Feature Tools

The MPEG-7 standard defined many descriptors of features, and in general, a descriptor is a representation of a feature. Descriptors store the feature values extracted from multimedia data, and can then be expressed in bit-stream format or DDL (XML-based) format. So, for every feature, we have defined a structure for storing content of the descriptor and an interface for accessing the content of the descriptor. Since our system is written in C++ programming language, writing a class for each feature would be the first step to add a feature into the system. An “instance of Descriptor class” is the place to store the description of the feature temporarily, and therefore, it is the input and output of the ExtractionTool, the MatchingTool, or the CodingTool.

Functions related to a feature can be categorized into four types: functions for extracting feature values from multimedia data, functions for calculating the distance between the content of two descriptors, functions for decoding a descriptor bit-stream file and storing the result into a Descriptor, and functions for encoding the content of a Descriptor into a bit-stream file. Therefore, classes of ExtractionTools, MatchingTools and CodingTools are respectively defined to fit the prescribed purposes, as shown in the following:

3.1.1 Extraction tool

An ExtractionTool is the functional block

responsible for extracting a feature value from multimedia data. Before performing the real extraction task, a FeatureInput and an empty instance of class Descriptor must be set. A FeatureInput is a representation of multimedia data being extracted, and it can be in many types. An ExtractionTool will check the type of a FeatureInput first, and decide whether to accept it or not. If an ExtractionTool accepts a FeatureInput, then the ExtractionTool will perform the feature value extraction according to the type of that FeatureInput. On the other hand, if an ExtractionTool finds that the type of a FeatureInput is not what he wants, the ExtractionTool will reject it and return with error.

For example, if the ExtractionTool for Color Layout feature receives a FeatureInput, which has the type of TYPE_FILE, then it will load the image file from memory with the path and name specified in the FeatureInput and start the extraction process. Conversely, if the ExtractionTool receives a FeatureInput having the type of TYPE_VECTOR_FILE, which is dedicated for video, it will do nothing and return with error.

After extracting the feature value from multimedia data, the extracted value is filled into the empty Descriptor, which has been set in the beginning. At this time, the extraction process for the feature is complete, and the result is a Descriptor filled with the value just extracted.

Figure 3(a) shows the input and output of an ExtractionTool, it takes a FeatureInput and an empty Descriptor as input and the Filled Descriptor (filled with the extracted data) as output.

3.1.2 Matching tool

A MatchingTool is the functional block responsible for comparing two Descriptors of the same type, and calculating the distance between the two Descriptors. One query Descriptor and one reference Descriptor should be set before performing the matching process. A MatchingTool will check what features the Descriptor are related to, and reject the input if the proposed features for the Descriptor are irrelevant. For example, the MatchingTool for Color Layout should only take the Descriptors related to Color Layout as input. After running the algorithm to compare two input Descriptors, a distance is calculated. The distance is used to decide whether a multimedia data is suitable to be a matching candidate of the input. The smaller the distance, the higher the score of match.

Figure 3(b) shows the input and output of a MatchingTool, it takes two Descriptors as inputs and a floating distance value as output.

Fig. 3 The inputs and outputs of (a) an ExtractionTool, (b) a MatchingTool, and (c) a CodingTool

3.1.3 Coding tool

A CodingTool is the functional block responsible for encoding a Descriptor into a bit-stream file, and decoding a bit-stream file to a Descriptor. Before performing the encoding process, a Descriptor is set as the source. The CodingTool then encodes the Descriptor into a bit-stream. As for decoding process, a file containing the bit-stream of the description of a multimedia data is set as the source, and an empty Descriptor is set as the target. After the content of the bit-stream file is decoded, the decoded result is filled into the empty Descriptor.

Figure 3(c) shows the input and output of a CodingTool. In the encoding phase, it takes a Descriptor as input and outputs a bit-stream file. And in the decoding phase, a CodingTool takes a bit-stream file and an empty Descriptor as inputs and a filled Descriptor as output.

3.2 Media Library Interface

As prescribed in Section 2.2(b), the Media Library provides functions for decoding lots of formats for images and audios and the MPEG-1/MPEG-2 encoded videos. The Media Library Interface, included in the MPEG-7 kernel, is a more high level interface that provides a common interface for the same type of multimedia data. For example, the Media Library

Fig. 4 The inputs and outputs of the ImageLoader and the VideoLoader

Interface for images, which is called ImageLoader, providing a common interface among all formats of image files. When a user wants to load an image file into memory, he just needs to input the file name into the ImageLoader no matter the image is a bitmap file or a compressed JPEG file. It is the ImageLoader's responsibility to check what format the input image file is, to call the correct function from the Media Library, and to output the image data. In this way, a user needs not to worry about which library to call, and therefore, speeding up the development of an application.

As for the VideoLoader, which is a Media Library Interface for video, it can decode an input video using the right decoder according to the syntax of the video. For example, after the decoding process is completed, a file containing motion vectors of the input video is generated, and the file can be used as an input for feature value extraction, such as the Motion Activity and the Camera Motion features.

Figure 4 presents what functions the ImageLoader and VideoLoader provided.

3.3 Input and Output of the Kernel

3.3.1 FeatureInput

A feature input is a hint for describing what a user wants to search in the database; therefore, it can be in many forms and depends on the selected feature type, as well. Input forms that are suitable for one type of feature may not be suitable for others. For example, a user may form a query by using a sample image for querying an image database, which is the so-called "query by sample" form, but he or she will not use a sample video to query a video database. Even for the same type of features, the forms of inputs may also be different from one another. Furthermore, one feature may be queried in more than one form. For example, a user can use a sample image or draw an image as an input for forming a query of Color Layout.

Since we wish to use the same interface when adopting tools of the MPEG-7 kernel, we would not like to see the ExtractionTool of Color Layout taking a file name as input while the ExtractionTool of Motion Activity taking a quantized direction, which is in a very different form from a file name, as input. So the class of FeatureInput is defined to package all forms of inputs into the same representation. In this way, all ExtractionTools, regardless of what features they are related to, will receive instances of FeatureInput as inputs for processing queries. The types of inputs included in the FeatureInput class are as follows.

- TYPE_FILE
- TYPE_DESCRIPTOR
- TYPE_BUFFER
- TYPE_MEDIA_DATA
- TYPE_VECTOR_FILE

For querying an image, the “Query by sample” form mentioned above would be one way. So, the TYPE_FILE, which specifies the name of the input sample file, is included. The TYPE_MEDIA_DATA is the input form for loading images. The RGB values of the input image are temporarily stored in it with certain structure. And the structure is called MediaData.

Sometimes, inputs of some features are best formed as descriptions, which are the attributes of TYPE_DESCRIPTORs. For example, we would like to make a query as: “Find a video with very high intensity”. For this query, the ‘very high intensity’ might already be one of the attributes of a Motion Activity Descriptor. Users can directly specify the value of a Descriptor for processing a query. For some features of video, motion vectors are the main information to conduct a query. So, a file, TYPE_VECTOR_FILE, containing motion vectors is also a reasonable type of input for such features. As for the TYPE_BUFFER, it is preserved as a general input type.

(b) MatchList

After the system find the best matches for inputs of a user, the system presents a MatchList as output, which is a list of candidates. The MatchList records the information of candidates and also the distances between the input and the candidates.

In our design, each element of the MatchList stores the candidate’s entry number in the database, and the corresponding distance calculated by using the MatchingTool. The elements in a MatchList are ordered in an ascendant way according to their distances. Thus, we will have the best match candidate at the top of the MatchList.

4. SYSTEM ARCHITECTURE

4.1 Stages of a Query

The multimedia data query is more complicated than a traditional text-based query. The most common way of making a query to text database is to use key words, but can we use the same approach to query a multimedia database? The first difficulty we met is that we have to give every multimedia content a text-based description, and this description must be as general as possible such that every one who wishes to make a query will have no difficulty to decide what keyword to use. This is certainly an unreasonable way, because how one feels about a sound, a picture or a video is very subjective. So how one can formulate a multimedia query naturally and easily is a big challenge. After a user made a query, the system must perform database search very quickly and show the results of query in an organized and sensible manner, so that the user can realize whether she or he is asking the right question.

A user may query a multimedia data in many situations. There are three types of queries expected from end users as discussed in [8]:

- (1) A user may have once seen a particular piece of data and wants to retrieve it for viewing or reuse.
- (2) A user may look for a specific data but has never actually seen it before.
- (3) A user may have only some vague idea of what it is that she or he is actually looking for.

Ideally, the query-formulation process should accommodate these types of queries, so we model the formulation of a multimedia query as a sequence of stages, each stage behaves like a filter of information, to reduce the number of candidates in the matching list.

The sequence of query stages is as follows: querying on categories of multimedia data (Navigating), querying on feature descriptions of multimedia data (Searching), and querying on full representatives of multimedia data (Browsing).

(a) Navigating

In this stage, we have to, first, decide what type of media is searching for. Depending upon the goals of queries, *e.g.*, a video clip, a sound clip or a piece of picture, the corresponding query stages will be proceeded in different ways. After deciding which type of media we wish to query, the choice of specific type of data is the next step. Some databases categorize multimedia data according to their semantic meanings. For example, one may categorize videos into news, sports and home videos, and categorize pictures into landscapes, scientific pictures, living things, etc.

In the stage of navigating, it is reasonable, and probably more sensible to formulate the query in text form, just as we did in the proposed system. An effective information retrieval system should prompt different categories of navigation approaches for users to choose.

(b) Searching

Searching is the most important stage of information retrieval for most database systems, the result of searching is a list of candidates that satisfy the constraints of a query. In this stage, we have to choose which feature of specific type we wish to use. Different features are defined for different types of media data. After selecting the feature, we have to input the thresholds or the acceptable ranges of the feature as constraints of the query process.

In order to quickly narrow down the search range and make the query more precious, one can also use logical manipulations among different features. Using AND, OR, and NOT logical operations can number down the query times we have to make before data of interest are found. So, these logical operations are included in our system, too.

(c) Browsing

The result of searching stage is a list of candidates, but these candidates may or may not contain what we really want to find. So, in this stage, the main goal is to enable user to do fast browsing through the matching list, and let user realize whether she or he asked the right question, as soon as possible. If the answer is yes, the user may start to enjoy watching a video or listening to a music clip, but if the answer is no, the user should be able to adjust the constraints of the query efficiently and re-start a new query. To do this, some specific design must be done to provide effective system-user interactions, as will be described in the next section.

4.2 The Design of User Interface

In the navigation stage, we need to show a list of supported types of media data. In fact, in our system, three types of media data are supported, that is Audio, Video, and Image. Moreover, there is also a list of features for each type of data. In addition to MPEG-7 standard features, one can also include user-defined features into the proposed system. The relations between different types of media and different features can be represented as a tree structure.

In the searching stage, the input of chosen feature is the most relevant factor to be considered in the UI design. Input methods for different types of features are very different from one another, for example, query by sample would be a nature way for querying an image

database, but it may not be so straight for querying a video. While for querying an audio database, singing would be the most nature input method. Moreover, input methods for each feature of the same type of media data are different from feature to feature. For example, the input method for dominant direction motion of the Motion Activity feature can only be quantized direction values, but for the Motion Trajectory feature, a direct drawing of the route of motion regions may be a more appropriate one. So, the system should have one specific design for setting up each feature.

We chose dialogic style as the basis of UI design for feature setup. When a feature is utilized to make a query, the system popping up the setup dialog for that feature, and after user pressing "OK" button, all information obtainable from that feature is retrieved and stored as input data for conducting the query.

After finishing the searching procedure, a matching list will be formed. The system should display the matching list and give each candidate a score to represent its distance to the query input. User can check whether the query is successful or not by examining the corresponding scores or by browsing the candidates subjectively.

In the browsing stage, we need a place to show the image, to play the video, or to listen to the audio. For video and audio, random access functions are desirable; therefore, in our system, play, stop, and pause are basic browsing functions while fast forward and fast backward are advanced options.

As prescribed, effective system-user interaction is also a desirable function for users to re-start a new query, the system should provide a place to display the current query status.

To support this requirement, in our system, the following query status related information is provided:

- (1) Current state of query: In which stage the user is.
- (2) Current status of query: Whether the query is completed, or whether the query is successful or not.
- (3) Hint: What user can do next.
- (4) Error and warning: When user did not follow the normal usage procedure of the system, a warning message will appear for guiding the user to the correct direction.

In conclusion, to construct a generic and flexible content-based information retrieval system, some human-centric and system-user interaction issues should be taken into consideration in the UI design. For example, in the proposed system, the following specific UIs have been built:

- A Setup UI to display the tree structure of the

relations between types of features and features themselves (c.f. Browse Window in Fig. 5).

- A Feature Setup Dialog Window for setting up each feature (c.f. Setup Dialog Window in Fig. 6).
- A Result UI to display the search results (c.f. Result Window in Fig. 5).
- A Browse UI, for end users, to browse the matching list (c.f. Browse Window in Fig. 5).
- A status UI to report to users the system information about the current query (c.f. Status Window in Fig. 5).

In addition, to provide more visual aids for content analyzing, a specific Demo UI (c.f. Demo Window in Fig. 5) is included to display some analysis-related special effects. For example, we can display the 8×8 color matrix for the Color Layout feature to see its spatial color distribution, or we can show the histogram of temporal distribution parameters of the Motion Activity feature of a video to see its motion behavior along the time axis. Figure 7 shows the operational appearance of each window respectively.

In summary, we need a Setup UI to display the tree structure of the relations between types of features and features themselves, many Feature Setup Dialogs for setting up features, a Result UI to display the search results, a Browse UI for end user to browse the matching list, and a Status UI to give user the information about the system. In addition, we added another window called Demo UI to display special effects.

5. THE DYNAMIC LINKING LIBRARY

In Section 2, we had pointed out that the proposed system is not only a content-based retrieval system, but also a platform for testing efficiency and effectiveness of adopted features. One can add a new feature into the system and test its corresponding retrieving performance, on the same platform. Methods of adding a feature into the system must be very easy and common for all types of features.

In addition, basing on the low-level features supported in the system, and with the aid of logical operations, one can retrieve high-level and possibly semantic features. We all wish that the ideal way of retrieving semantics of multimedia data is to input a piece of media data into the system, and the system can automatically generate the semantic high-level features presented in some general forms. Unfortunately, this domain is still an open research area and needs further investigations. Directly retrieving the semantics of a multimedia data seems not a practical way, because it involves media understanding and machine learning

heavily, may be the AI technology, in the future, can help to resolve this problem. At this time, one possible way is to use automatically extracted low-level features to compose high-level features. In this approach, the most challenging problem would be how to find the relations between low-level and high-level features, and this requires a good of domain knowledge. By observing relative changes of low-level features when a high-level semantic event occurred, some semantic feature of the media data could possibly be learned. For example, for a basketball video, we found that when a shoot is made, the sound of crowd would be very loud and the mid-court camera would be halted for a while. Thus, by combining the Camera Motion feature and some audio features, the shot of basketball shooting could possibly be retrieved and indexed.

In order to test these concepts, the system provides logical manipulations among different features. It makes the system very flexibility and can be used to test lots of cases of feature combinations.

In the design of the system, it is our wish that system developments and features constructions can be separated very clearly. So, the interface between the system and features must be defined. In addition, after the system is developed, we also wish that adding a new feature into the system needs as little modification of the system as possible, and new features can be added into the system at run-time. So, we chose to use Dynamic Linking Library (Dll) as our basis for developing new features; in other words, adding a new feature is equivalent to write a Dll. And the system can load the Dll into memory at run-time, and use the Dll to do the feature's specific work. In this way, to include a new feature, there is no need to re-build the system and the Dll of the new feature can be developed independently.

The Dll of a feature exposes a function called "CreateIFeature," which returns an interface with functions for using the new feature. A user loads the Dll into memory and makes a call for the CreateIFeature function to get the interface. The functions included in the interface of a feature are listed below.

- HRESULT Release ();
- HRESULT SetDataBase (MediaDB* db, CHAR* path);
- HRESULT GetUserInput (FeatureInput* input);
- HRESULT StartQuery (FeatureInput* input, MatchList* result);
- HRESULT StartDatabaseConstruction (FeatureInput* input);
- HRESULT GetFeatureType (INT* type);
- HRESULT GetFeatureID (INT* id);
- HRESULT GetFeatureName (CHAR* buf, INT buflen);

- HRESULT SetDemoDC (HDC memDC, HDC realDC);
 - HRESULT CreateFeatureInput (FeatureInput** input);
 - HRESULT DestroyFeatureInput (FeatureInput* input);
 - HRESULT CreateMatchList (MatchList** output);
 - HRESULT DestroyMatchList (MatchList* output);
- All the details of the above DLLs can be found in

Appendix A.

After retrieving the interface of a feature, a user can then call the functions to do real tasks, such as retrieving input for a user, querying a multimedia database, or constructing a multimedia database. We can briefly categorize these functions into 5 different categories (*i.e.*, Feature Identification, Input/Output Management, Query Process, Database Construction, and Others), as illustrated in Fig. 8.

Fig. 5 The layout of the proposed multimedia retrieve system

Fig. 6 The setup dialog window for the Motion Activity feature.
Each feature has its own setup dialog window.

Fig. 7 The operational appearance of each window: Setup Window shows the list of features for each type of multimedia data; Result Window lists the matching list; Browse Window plays the candidate movie; Status Window presents the information related to the current query; and, Demo Window demos the special effect of the retrieved data

Fig. 8 The relationship between Dll and interface of a feature, and function categories provided by the interface of the feature

6. APPLICATIONS

6.1 The Scene Change Detector

As shown in Fig. 1, in order to realize the whole content-based retrieval system, some modules such as the *Parser* and the *Transcoder* have to be executed in advance. More specifically, the purposes of using these modules are to pre-process the huge volume of

multimedia data into some elementary indexed units for the convenience of succeeding modules. Giving video content as an example, it is difficult to search and index the video database, directly. Generally agreed, the first step towards video indexing is to detect boundaries between consecutive camera shots so as to organize videos into elementary indexed units. Prior to this work, we have proposed an interactive scene analysis system, including scene and dissolve change detections [9,10]. The algorithm is developed based on macroblock type statistics, and thus possesses the advantages of accuracy and efficiency. A snapshot of the realized scene analysis system is presented in Fig. 9. It allows for inputting MPEG-1, -2 video files and operates with two analyzing-modes: automatic and semi-automatic, selected by the user. In the automatic mode, the system adopts the default parameters of each frame and uses the distance measure of the shape and color information, defined in MPEG-7, to detect the shot. Alternately, if semi-automatic mode is chosen, the corresponding parameters of the adopted features changed interactively by the user (c.f. Fig. 9(b)). On the left side of the system's snapshot, there is a column window displaying the detection results (*i.e.*, key-frames). The frame number and the frame type (*i.e.*, I-, P-, or B-) of the detected shot are also marked in the window. By clicking the small shot icon on the window, the video clips from the clicked shot point to

the next detected point will be played for subjective previewing. If the user is unsatisfied with the results (e.g., missing some important shot points), he can set the parameters interactively and re-do the analysis, of course, only for the unsatisfied clips.

Besides the scene change and dissolve detections, the *Parser* module further needs the functions like image segmentation and color filtering (just as done by the Color Layout and Dominant Color descriptors defined in MPEG-7). By previewing the visual effect

of MPEG-7 descriptors, user can decide which descriptors they need to encode the image or video, in favor of the subsequent querying and browsing stages. Figures 10 and 11 show the resulting images described by using the color layout and dominant color descriptors, respectively. As for the image segmentation, its purpose is to support the video trajectory descriptor defined in MPEG-7. The algorithm we adopt to implement this descriptor is mathematical morphology based, and the details of it can be found in [11].

(a)

(b)

Fig. 9 The snapshots of the content analysis system operated in (a) automatic mode and (b) semi-automatic mode, with the user feedback interface (i.e., the Setting up Dialog in (b))

Fig. 10 The original (*i.e.*, the right-hand-side) images and the color layout (*i.e.*, the left-hand-side) images

Fig. 11 The original (*i.e.*, the right-hand-side) images and the dominant color layout (*i.e.*, the left-hand-side) images with 12 dominant colors

6.2 The Video News Program Analyzer

By observing the structure and domain knowledge of news programs, we developed a hierarchical video news program analyzer. Obviously, the basic unit of a news program is not the low level shot change anymore. From the aspects of end users, the basic unit of a news program should be a *news item*. The anchorperson roughly describes the summary of an item of news, and then the corresponding video content of the news is displayed. Figure 12 shows the semantic structure of a news program. According to this property, we can regard the first frame with an anchorperson as the beginning of a news item, and a semi-automatic mode news video program analyzer is developed. The first step of the analysis is still the prescribed shot change detection process. After the low level shot change detection is done, all boundaries between consecutive camera shots are detected and represented as key-frames. An interactive interface is designed and user can choose which frame is with the anchorperson. The chosen frame is described by two color descriptors: color layout and color histogram. By comparing bin-to-bin color layout or color histogram distances between the anchorperson-frame and detected shot change frames (key-frames), the whole news program video can be divided into some news items of semantic meaning. By our experiments, using color layout descriptor provides more accurate result than that of the color histogram. The reason is that the color layout descriptor preserves one more color information, *i.e.*, the color's spatial distribution, of an image than that of the color histogram descriptor. Figure 13 shows the

snapshot and results of the news program analyzer. Notice that, in Fig. 13(b), the head-and-shoulder image of the anchorwoman has been extracted by using the prescribed image segmentation descriptor. And this extracted image is then treated as the input for querying news items from the key-frames.

6.3 The Home Media Center

Nowadays, people can scan pictures into a PC by a scanner and take pictures by a digital camera easily. And this makes the amount of digitized media data at home grows immeasurably. Therefore, a home image analyzer or a home media center (in a more general scenario) is of great interest. In this application, in addition to the color layout and dominant color descriptors, we use some other specific features, such as face finder [12] and sky detector [13], to enhance its applicability. These features are designed based on the domain knowledge of images taken from the digital camera. For example (c.f. Fig. 14), a user can query digital pictures by using a sample image or a user drawn image (the colors in the upper and the lower portions of the right-bottom window of Fig. 14 are blue and green, respectively, and the dominant color of this image is blue) as input for forming a query based on color layout and dominant color descriptors. With the aid of appropriate features, the system can further tell the pictures being indoor or outdoor, and with one person or two persons appeared in the picture. We have collected more than 2,000 digital pictures captured by digital camera to test the applicability of this application.

Fig. 12 The semantic structure of a news program. (Each small image represents a key-frame of a shot, and the number attached under the image is the frame number. There are totally two items of news in the sequence.)

(a)

(b)

Fig. 13 The operational snapshots of the news program analyzer. (a) Frames with anchorwoman queried by color layout descriptor, and (b) the extracted anchorwoman image (left) and the corresponding retrieved news item (right)

Fig. 14 The operational snapshot of the home media center

Fig. 15 Examples of various shapes which are accepted for the region-based shape descriptor

6.4 The Logo Verification System

As discussed in [6], the MPEG-7 defined region-based shape descriptor can be applied to retrieve any types of shape information in various applications of image and video databases retrieval. In addition, it is robust to input noise, such as the salt-and-pepper noise that occurs quite often in the process of image segmentation.

More specifically, the region-based shape descriptor has the following features:

- (1) Minor non-rigid deformations are allowed for the retrieval, as illustrated in Fig. 15(a).
- (2) An isolated object generated during the process of

segmentation, as shown in Fig. 15(b), is also allowed.

- (3) It can also describe a complex object in which holes carry as much information as that of the outer boundaries, as shown in Fig. 15(c).
- (4) An object that consists of separated regions, as shown in Fig. 15(d) can also be described.

By above features, the region-based shape descriptor is very useful for Logo or trademark verification. The system can verify whether the querying logo is used or registered by other groups or companies. The database we used has about 2,800 logo images provided by the MPEG-7 content set [14,15]. Figure 16 shows the operational snapshots of

the Logo verification system, which is believed to be useful for digital right management.

6.5 A Contour-Based Fish Images Analysis and Retrieval System

To show the effectiveness of the proposed system, a contour-based fish images analysis and retrieval system is realized. The query interface of the system is shown in Fig. 17(a). The system provides two kinds of queries: Query by Sample and Query by Depiction, to retrieve fish images from the database. In the mode of Query by Sample, one can choose any example image existed in the database as a query input. The parameter “Query-Num” is used to denote the number of similar resultant images. In the mode of Query by Depiction, one can draw the contour of the wanted fish for querying. The drawn is then represented by two global shape parameters and the maximum of curvature zero-crossing contours in its Curvature Scale Space (CSS) image (which are all defined in the MPEG-7).

We also developed a wireless fish images retrieval system. The query interface is shown in Fig. 17(b). By a handheld device like PDA, we can connect to the server and retrieval the fish images through wireless connections. This system also supports the prescribed two kinds of queries. The parameters “Server-IP” and “Server-Port” are used to set the IP address and the port

number of image server, respectively.

For the purpose of realizing a practical fish retrieval system, we collaborate with the Laboratory of Fish Ecology and Evolution (LFEE), Institute of Zoology, Academia Sinica. Figure 18 shows some of the fish-outline images provided by LFEE’s fish database (available at <http://fishdb.sinica.edu.tw>). In order to represent these images by MPEG-7 contour-based shape descriptors, we have to preprocess the fish-outline images to obtain the corresponding contour images. One of the original fish-outline image and the corresponding contour image are shown in Figs. 19(a) and Fig. 19(b), respectively.

Since all the fish-outline pictures in the database are gray-level images, we intuitively making use of the brute-force (full scan) method. We scan the image vertically from top to bottom and repeatedly until the pixel value is larger than a pre-defined threshold. The same work is done horizontally from right to left, too. Figure 19(c) shows the conceptual interpretation of the adopted brute-force approach. This approach works very well for most of the simple fish-outline images.

However, for complex fish-outline images, as shown in Fig. 19(d), the brute-force method did not work well. As shown in Fig. 19(e), the region marked by green line is missed, and it is addressed as a missed region. We will further investigate it with the aid of some graph theorems so as to solve the problem, in the future.

Fig 16. The operational snapshot of the Logo/Trademark Verification System

Fig. 17 The query interface of (a) the storage-based fish images database and (b) the wireless fish image retrieval system

Fig. 18 The fish-outline images

Fig. 19 (a) The original fish outline image, (b) the contour image corresponding to (a), (c) the conceptual interpretation of the scanning process, (d) a complex fish outline image, and (e) the missed region corresponding to image (d)

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a content-based multimedia analysis/retrieval system basing mainly on the MPEG-7 defined features. The system can be used as an image, video and audio query system, or a platform for testing the efficiency and effectiveness of features used in a multimedia query process. The architecture of this system is designed rather flexible to meet the above-mentioned requirements. It is not only designed for certain type of applications, but providing a common interface among all features used among different application scenarios. Furthermore, some new and specific features are also included for supporting various requirements of different applications. On the basis of this well-modularized system, a lot of interesting applications, such as scene change detector, video news program analyzer, Logo verification system, fish shape image analysis and retrieval system, and home media center (a home media management system) have been built and investigated.

Besides, the Multimedia Description Schemes (MDS) in MPEG-7 are defined as description tools (Ds and DSs) that are generic, *i.e.*, neither purely visual nor purely audio. They can be considered as high-level meta-data information. For multimedia data with timing information, like video sequences, the sub-part structural aspect defined in the content description MDS is very useful. At present, our MPEG-7 content-based analysis/retrieval system did not contain the prescribed structural aspect MDS. Therefore, we will investigate this part and the corresponding applications, like semantic error resilience, the content-aware encoding and content utility tools in the near future.

APPENDIX A

The functions related to the interface, of the proposed system, for a given feature are:

- HRESULT Release ();
- HRESULT SetDataBase (MediaDB* db, CHAR* path);
- HRESULT GetUserInput (FeatureInput* input);
- HRESULT StartQuery (FeatureInput* input, MatchList* result);
- HRESULT StartDatabaseConstruction (FeatureInput* input);
- HRESULT GetFeatureType (INT* type);
- HRESULT GetFeatureID (INT* id);
- HRESULT GetFeatureName (CHAR* buf, INT buflen);
- HRESULT SetDemoDC (HDC memDC, HDC realDC);
- HRESULT CreateFeatureInput (FeatureInput** input);
- HRESULT DestroyFeatureInput (FeatureInput* input);
- HRESULT CreateMatchList (MatchList** output);
- HRESULT DestroyMatchList (MatchList* output);

We can briefly categorize these functions into 5 different categories, as illustrated in Fig. 8. They are:

- Feature Identification function,
- Input/Output management function,
- Query Process function,
- Database Construction function,
- Others.

In the following section, we will depict these categories and discuss why we need to provide these functions.

A.1 Feature Identification Functions

As can be expected, every feature has a unique ID and a unique name. For example, the name and the ID

for the Color Layout feature are “ColorLayout” and “0” respectively. Therefore, it is easy to identify an interface by examining the feature ID and the feature Name of the interface related feature. Functions GetFeatureID() and GetFeatureName() are defined for this purpose. Function GetFeatureID() returns the ID of a feature, and function GetFeatureName() returns the name of a feature.

It is important for a Dll-based system to identify the type of a feature, because different databases, different query queues, and different tools are required for different types of features. It will cause some serious problem if we use an audio related feature to retrieve a video data. Therefore, in addition to feature ID and feature name, the feature related media type must be able to be identified. The function GetFeatureType() does this job.

A.2 Query Process Functions

It is clear that the interface for each feature must provide sufficient methods for acquiring user input and performing the query. The functions belonging to this category are:

- HRESULT SetDataBase (MediaDB* db, CHAR* path);
- HRESULT GetUserInput (FeatureInput* input);
- HRESULT StartQuery (FeatureInput* input, MatchList* result);

Before doing the real query task, database, which is the host for searching candidates, must be specified. An instance of MediaDB which contains a list of files included in the database and paths where these files can be reached are passed to the Dll for setting up database information of a query. After setting up the database, it is ready to acquire inputs from the users. An initialized FeatureInput instance is allocated to store the input information after communicating with the user. And then, according to the input information, the query for that feature is started, and the resultant candidates are stored in the MatchList.

The above functions provided by the interface are generic for all kinds of features, regardless of their types.

A.3 Input/Output Management Functions

In the proposed system, the functions for managing the input and output of a query include:

- HRESULT CreateFeatureInput (FeatureInput** input);
- HRESULT DestroyFeatureInput (FeatureInput* input);
- HRESULT CreateMatchList (MatchList** output);

- HRESULT DestroyMatchList (MatchList* output);

These functions are designed for creating and removing objects of the input and output. CreateFeatureInput() and CreateMatchList() allocate and initialize instances of FeatureInput and MatchList, respectively. While DestroyFeatureInput() and DestroyMatchList() free the memory previously allocated to CreateFeatureInput() and CreateMatchList(), respectively.

Why should we manage these input/output objects in the Dll? Why can't we just allocate an instance for FeatureInput and another instance for MatchList and pass them to the Dll of a query? The main idea for managing the memory of input and output objects, of a query, is that we should allocate and free the memory of the same object at the same place. It means that if we allocate an object in a Dll, then we have to free the object in the same Dll, and if an object is allocated to a system which uses the Dll, then it must be freed in the same system, too. This is because the surroundings and parameters for allocating memory to a Dll and a system may be varied. If we free the memory in a place which has different surroundings and parameters from where the memory is allocated, some unpredictable problems may arise.

As for the second question, it is because that the forms of an input are feature dependent and are decided by the function GetUserInput(). The function finds the type of an input and allocates memory for storing data accordingly. As prescribed, these allocated data for a function which is performed in a Dll must be freed at the same Dll, too. For example, if a user input some hints by sketching, the instance of FeatureInput created by calling CreateFeatureInput() will have a buffer allocated to GetUserInput() for storing the content of the sketch. After using the instance of FeatureInput for a query, the buffer and the memory for the instance itself are freed by calling DestroyFeatureInput(). Therefore, these functions are provided for system designers to manage input and output objects of a query. The designer won't have to worry about the allocation and freeing of the memory and make the use of Dlls more easily.

A.4 Database Construction Functions

The functions provided by an interface for database construction are:

- HRESULT SetDataBase (MediaDB* db, CHAR* path);
- HRESULT StartDatabaseConstruction (FeatureInput* input);

The process of database construction extracts the description of a feature, from a multimedia data, which will be added to the database, and stores the description

of the multimedia data for supporting later queries. In our design, the descriptions are stored in form of bit-stream files. So, the input of database construction is the same as the query process, while the output of database construction is a bit-stream file containing the description of the multimedia data. Still, before performing database construction, we need to specify the database for adding the new content to. By calling SetDataBase(), we setup a path of the database and find where to save the description of the multimedia data, accordingly.

A.5 Others

The rest of the functions are listed below.

- HRESULT Release();
- HRESULT SetDemoDC (HDC memDC, HDC realDC);

Release() is the function to delete the constructed interface.

For the purpose of providing some demos, the function SetDemoDC() is designed to set up the Device Context (DC) for showing demos. A Dll can then draw some special demonstrations in the DC. For example, we can draw the 8×8 color blocks of an image to the DC for showing the Color Layout feature, visually. The content of a demo is, of course, feature dependent.

REFERENCES

- [1] IS&T/SPIE, *Storage and Retrieval for Still Image and Video Databases*, I-V, 1993~2002.
- [2] IEEE, *Proceedings of the International Conferences on Multimedia Computing and Systems*, 1994~1999.
- [3] IEEE, *Proceedings of the International Conferences on Multimedia and Expo*, 2000~2002.
- [4] ISO/IEC 15938-1, *Information Technology — Multimedia Content Description Interface—Part 1: Systems*, 2002.
- [5] ISO/IEC 15938-2, *Information Technology — Multimedia Content Description Interface—Part 2: Description Definition Language*, 2002.
- [6] ISO/IEC 15938-3, *Information Technology — Multimedia Content Description Interface—Part 3: Visual*, 2002.
- [7] ISO/IEC 15938-4, *Information Technology — Multimedia Content Description Interface—Part 4: Audio*, 2002.
- [8] R. M. Bolle, B.-L. Yeo and M. M. Yeung, "Video query: research directions," *IBM Journal of Research & Development*, Vol. 42, No. 2-

- Multimedia Systems, 1998.
- [9] S.-C. Pei and Y.-Z. Chou, "Efficient MPEG compressed video analysis using macroblock type information," *IEEE Trans. Multimedia.*, Vol. 1, Dec. 1999, pp. 321–333.
 - [10] J.-H. Kuo and J.-L. Wu, "An efficient algorithm for scene change detection and camera motion characterization using the approach of heterogeneous video transcoding on MPEG compressed videos," *IS&T/SPIE, Storage and Retrieval for Still Image and Video Databases*, I-V, 2002.
 - [11] W.-C. Lin, C.-H. Huang and J.-L. Wu, "A novel supervised image/video segmentation system based on mathematical morphology," *Proceedings of 2001 IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Baltimore, U.S.A., June 2001.
 - [12] R.-L. Hsu, M. Abdel-Mottaleb and A. K. Jain, "Face detection in color images," *Pattern Analysis and Machine Intelligence*, IEEE Transactions on, Vol. 24, Issue 5, May 2002.
 - [13] J. Luo and S. P. Etz, "A physical model-based approach to detecting sky in photographic images," *Image Processing*, IEEE Transactions on, Vol. 11, Issue 3, Mar. 2002.
 - [14] ISO/IEC JTC1/SC29/WG11, "Coding of moving pictures and audio, MPEG 98/N2466, licensing agreement for the MPEG-7 content set."
 - [15] ISO/IEC JTC1/SC29/WG11, "Coding of moving pictures and audio, MPEG 98/N2467, description of MPEG-7 content set."

Ja-Ling Wu (吳家麟) received the B.S. degree in electronic engineering from TamKang University, Tamshoei, Taiwan, R.O.C. in 1979, and the M.S. and Ph.D. degrees in electrical engineering from Tatung Institute of Technology, Taipei, Taiwan, in 1981 and 1986, respectively. From 1986 to 1987, he was an Associate Professor of the Electrical Engineering Department at Tatung Institute of Technology, Taipei, Taiwan. Since 1987, he has been with the Department of Computer Science and Information Engineering, National Taiwan University, where he is presently a Professor. He was also the Head of the Department of Information Engineering, National Chi Nan University, Puli, Taiwan, from Aug. 1996 to July 1998. Prof. Wu was the recipient of the 1989 Outstanding Youth Medal of the Republic of China, the Outstanding Research Award sponsored by the National Science Council, from 1987 to 1992, and the Excellent Research Award from NSC, 1999. Prof. Wu has published more than 200 technique and conference papers. His research interests include algorithm design for DSP, data compression, digital watermarking techniques and multimedia systems.

Jin-Hau Kuo (郭晉豪) received the B.S. degree in information and computer engineering from Tatung University, Taipei, Taiwan, in 1997, and the Ph.D. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2003. His current research interests include data compression, digital image processing, content-based retrieval, code optimization, and multimedia systems.

Shao-Ying Lin (林 劭 穎) received the B.S. degree in computer science and information engineering from National Taiwan University, in 2000, and the M.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2002. His current research interests include content-based retrieval, multimedia systems, and embedded system.

Song-Wen Wang (王 頌 文) received the B.S. degree in computer science and information engineering from TamKang University, Tamshoei, in 2001, and the M.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2003. He is currently a Ph.D. student in the Dept. of Computer Science and Information Engineering, National Taiwan University. His current research interests include image and video coding, multimedia systems and optimization techniques for video compression algorithms.

收稿日期 92 年 12 月 17 日、修訂日期 93 年 2 月 9 日、接受日期 93 年 2 月 12 日
Manuscript received December 17, 2003, revised February 9, 2004, accepted February 12, 2004