Design and Implementation of a Content-based Video Retrieval System

by

Yaping Shi

B. E. Tsinghua University, China, 1993

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

we accept this thesis as conforming
to the required standard

The University of British Columbia

September 1997

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

The University of British Columbia
2366 Main Mall
Vancouver, Canada
V6T 1Z4

Date: Sep. 25. 97

Abstract

As computer application enters a multimedia era, video data is becoming an information source. In this thesis, we investigate various video indexing and retrieval techniques and describe an implementation of a prototype video retrieval system that supports queries based on either visual or semantic content of video data.

In our system the basic unit for video indexing and retrieval is the shot. Several shot boundary detection techniques have been implemented and evaluated.

Each shot is represented by one or more key frames. The image feature vector of each key frame usually has very high dimensionality. So first we reduce the dimensionality of the feature space using the Principal Component Analysis (PCA) technique because most existing multi-dimensional indexing structure grows exponentially in size as the number of dimension increases. We've successfully reduced the dimension of the data sets obtained from our test video sequences from 256 to less than 10 while preserving more than 90 percent of the distance information between data points. Then we choose adaptive bucket k-d tree to index the PCA-transformed data points.

For the semantic annotation of video data, we propose a predicate-based annotation which has several advantage over the existing key word based annotation, including expressiveness and the ability to support inference. We also introduce a knowledge base which contains some common facts and rules that can be shared by different videos. Some of the annotations are derivable from other annotations using the rules in the knowledge base, so they need not to be explicitly added. In this way we can save a human annotator's work as well as storage space.

Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to express my sincere appreciation to my supervisor, Dr. Raymond Ng, for his guidance and support. His helpful suggestions, productive discussions, and constant encouragemant have made this thesis possible.

I would like to thank Dr. Norm Hutchinson, my second reader, for his careful reading of the thesis and valuable comments.

I would also like to thank David Finkelstein, Dwight Makaroff, and Jining Tian from Dsg lab, who spent a lot of time helping me to digitize hours of video programs. Special thanks to Andishe Sedighian, for allowing me to use her code of bucket adaptive k-d tree.

Finally, I would like to thank staff members and students in database Lab: Ross Carton, Dwi Faulus, Ed Knorr, and Carson Leung, for all their help.

YAPING SHI

The University of British Columbia

September 1997

Chapter 1

Introduction

1.1 The need for video indexing and retrieval tools

During the past decade, significant developments have taken place in VLSI, large capacity storage devices, multimedia compression technologies, and high-speed networks. Developments such as fast CPU, CD ROM, Digital Versatile Discs(DVD), JPEG/MPEG compression algorithms, ATM switches, and gigabit Ethernet have generated unprecedented computing power, storage capacity and network bandwidth which have helped greatly in overcoming the key obstacle imposed by the vast amount of data involved in video applications.

As a result of the above advances in the video enabling technologies, digital video can be played on most of the modern computer platforms; Numerous video applications have emerged, such as video-on-demand(VOD), digital video libraries, video conferencing, etc. However, while video applications are becoming ubiquitous, many of them do not take full advantage of digital video data. They treat the video largely as an unstructured linear medium. In these applications, the retrieval of

video data is based on the bibliographic description associated with each video, and the access granularity is the whole video. For example, some Video-on-Demand services allow the user to select movies by titles, actors, etc. Once a title is selected, the whole movie is delivered to the user, and only the traditional VCR functionalities, such as play, pause, fast forward, etc, are provided to the user. If the user wants to search for a certain part of the video, the only tool available is the fast forward function of the video player. In a video library with thousands of hours of video materials, using such sequential search techniques in order to locate a desired set of video sequences is impractical. In general, video data are not used very effectively in these applications.

Video will become an effective part of everyday computing environment only when we can use it just as efficiently and effectively as we use alphanumeric data today. The efficient and effective usage of video data depends on the availability of tools that can support some efficient means of classification, organization and retrieval of video data. However, the lack of such viable video indexing and retrieval tools is a serious limitation to the wide spread usage of video data.

In the following, we will first identify the issues that are related to the design of video indexing and retrieval systems, will then summarize the state of the art in these areas, and will finally present our solutions to some of the issues.

1.2 Design issues

As a new data type, video has imposed several new requirements to a video retrieval system. First, video cannot be used effectively without a way of accessing it in terms of its content. When a user only wants to retrieve a segment of the video, a straight forward method for him to express his interest is by referring to the content of the

video segment that he is looking for. Second, two new properties that distinguish video data from traditional data, namely their unstructured format and their large volume, have introduced additional difficulties to the design of a video retrieval system.

Following are three of the most important issues in the design of an effective and efficient content-based video indexing and retrieval system:

- Video parsing

  There are two inherent units for video indexing and retrieval: the entire video and the individual frames. For most applications the entire video is too coarse a unit since the user may want to access only a part of the video. On the other hand, a single frame is typically too small a unit since there are too many frames in a video. The video has to be partitioned into some meaningful units and then organized into some structures to facilitate its indexing and retrieval.

  Inspired from film theory, most recent research works use the /textitshot as fundamental unit for indexing and retrieval. Shots can be further organized into scenes, sequences, etc. As defined in [Kat94], a shot consists of a group of frames continuously recorded by a single camera; a scene is made up from a collection of shots unified by time and space; and a sequence is formed by a number of scenes linked together by narrative continuity. Shot, scene and sequence form a hierarchical structure of the video.

  Automatic video parsing which includes video partitioning and organization is the common first step in many content-based video indexing and retrieval systems.

- Video content abstraction

Due to the large volume of video data, it is impossible to put a video segment into a record in any practical indexing structure. It is also very difficult to directly process, for example to compare, the contents of video segments without some forms of abstractions. Consequently, some descriptive abstractions for concisely describing the visual and semantic video contents are desirable.

- High dimensional indexing structure for video abstraction

  Indexing structure can be constructed with respect to one or more abstractions of video content. Some video abstractions, though greatly reduced in volume compared to the video itself, are expected to have high dimensionality. An appropriate high dimensional indexing structure has to be selected. Techniques that can reduce the number of dimensions of the video abstraction while at the same time minimize the loss of information are also desirable.

1.3 Overview of existing content-based video indexing and retrieval techniques

Before we introduce our framework for a content-based video retrieval system, we briefly review the current status in the design issues discussed in the above section. A more detailed discussion of related works will be given in Chapter 2.

1.3.1 Video parsing

Video partitioning can be achieved via shot boundary detection, a technique that locates the boundary between two adjacent shots. There has been a great deal of research related to the automatic shot boundary detection on both uncompressed [NT92, ZKS93, ZMM93, ATe92, Sha95, HJW94, BR96, CB95, DLM+96] and com-

pressed [LZ95, lYL95, SP95, MJC95] video data . Fairly good accuracy has been achieved on detecting abrupt shot changes; the error rate is relatively higher in the detection of gradual shot transitions. Motion analysis can be used to reduce false positive identification of gradual transitions, but it also results in higher computational costs. In 2.2.1 and chapter 3, we will discuss this technique in more detail.

The automatic organization of shots in order to form higher level meaningful units, e.g. scene and sequence in the video, is more difficult since it requires some level of understanding of the semantic content of the video. Some methods have been proposed in this area, but most of the existing methods either need an a priori model or rely on certain repetitive patterns in a scene to assist the classification. For example, [Ze94] models a news item as a sequence of shots with anchor person shots at the beginning or the end, and [BYY96] assumes similar shots will be repeated in a scene as a result of montage presentation.

1.3.2 Video content abstraction

There are two basic types of content abstractions: text based abstraction and visual feature based abstraction.

In the text based approach, text descriptions (also called annotations by some researchers) of video content are attached to video segments. Video segments can then be retrieved by referring to their associated text descriptions. Keywords are a commonly used text abstraction in earlier image and video databases. All the traditional textual database techniques can be used in this case. Some enhanced text based abstractions such as attribute-value pairs [DSS96] support synonym, and some such as object-oriented frames [FSZ95] support taxonomy.

Visual feature based abstractions are usually achieved by using static images

such as key frames to represent shots or scenes, each of the frames can be further characterized as some image features, such as color, shape, texture, etc. This approach can capture the visual content of the video more easily, and usually can be generated automatically using image processing techniques.

Text based abstractions have two inherent problems. First, it is especially difficult to describe visual contents such as color and shape accurately using text. Second, automatic generation of text description is still beyond the capability of todays machine vision. Therefore, most text abstraction has to be generated by human operators. Because of the inherent problems with text abstraction, many recent research works have been focusing on the visual feature based abstractions. However, visual feature abstraction by itself is not complete since a lot of semantic contents of the video are lost during this abstraction procedure. Text annotation of video semantic content is indispensable, as text is nonetheless the most suitable form of semantic abstraction. The time-consuming nature of manual annotation makes text based abstraction hard to scale to large video databases. Therefore, machine assistance in the annotation process in order to minimize human involvement is desired. Moreover, how to make the annotations rich in descriptiveness also needs to be explored.

1.3.3 High dimensional indexing methods

Once the video abstractions are available, indexes can be created with respect to one or more abstractions. If the visual content is represented using static images characterized as image feature vectors, the video database is essentially transformed into an image database, and all the existing indexing techniques for image databases can be used.

However, existing multi-dimensional indexing structures such k-d trees and R-trees have two limitations [FSZ95]: the dimensionality of the search space is relatively low (< 20), and the distance in the search space has to be Euclidean. The existing indexing structure in most cases cannot be applied directly to the image feature vectors, since the dimensionality of the vectors is usually high (can be over 100) and the Euclidean distance is not always a good measure.

One way to alleviate the problems is by introducing a filtering step ([SH93], [FEe94]) in which a low dimensional coarse representation together with an Euclidean distance measure is used to approximate the original distance measure on the full image feature vector. This is achieved either by bounding the non-Euclidean distance, i.e. full quadratic distance, with a simple Euclidean distance; or by introducing some orthogonal transformation such as the K-L transformation to reduce the dimensionality if the original distance measure is Euclidean. The Euclidean distance measure on the coarse representation underestimates the original distances so it guarantees no false dismissals but allows false positives. The result of the filtering step will be further evaluated on the full representation without indexing structure using the original distance measure.

When the dimensionality of the original image feature vector is high and the database is large, 2-level filtering may not be sufficient. If the dimensionality of the coarse representation is low, then it may no longer be a close approximation. Thus the result of the filtering step will contain too many false positives, and the non-indexed search after the filtering step may be too costly. On the other hand, if the dimensionality of the coarse representation is high, the efficiency of the indexing structure in the filtering step will degrade. Multi-level filtering [Tam96] has also been proposed to deal with this problem. This approach basically feeds the result

from the first filter to other indexed filters before the final non-indexed search is performed on the original image feature vector.

1.4 Thesis contribution

The aim of this thesis is to provide a framework for content based video indexing and retrieval, and develop a prototype system to demonstrate the feasibility of the framework.

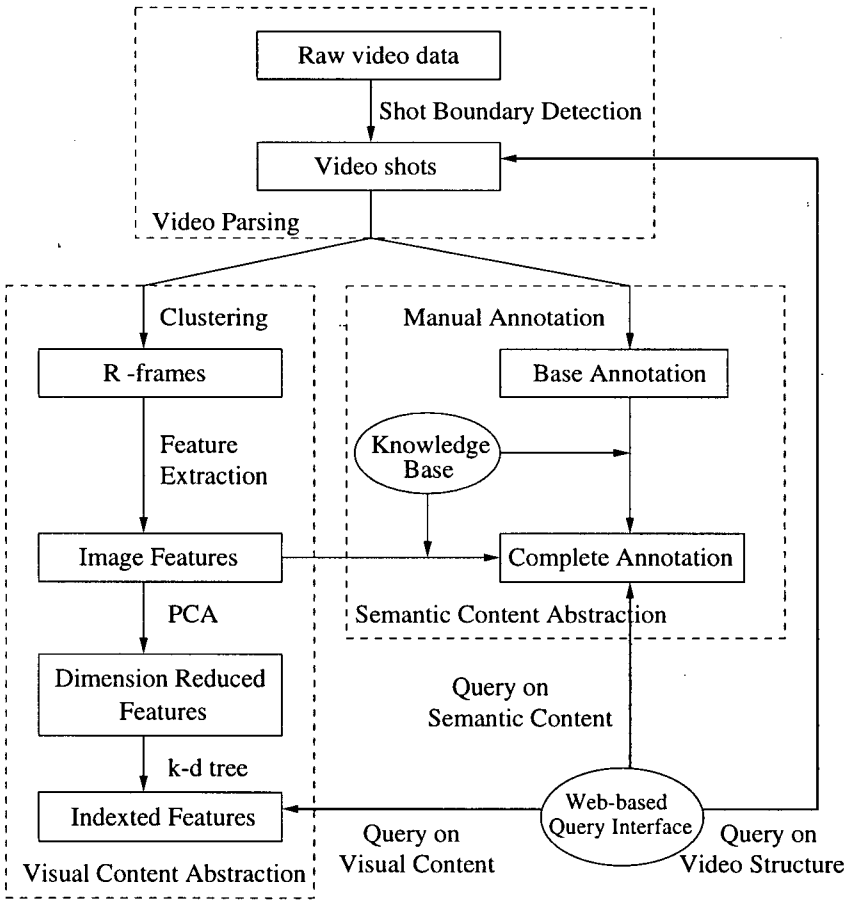

Figure 1.1: Framework of the content-based video retrieval prototype system

Figure 1.1 is the framework upon which this thesis is based. It includes three major parts, video parsing, visual content abstraction and indexing, and semantic content abstraction.

Following is a brief introduction of each part in the framework. Corresponding thesis work is also described.

Part I: Video parsing

For shot boundary detection, we chose and implemented several techniques. Our comparative analysis shows that the twin threshold histogram based approach with motion analysis [ZKS93] can produce satisfactory results. This approach is integrated into our prototype system. Shots can be further organized into scenes and sequences, but these issues are not discussed in this thesis.

Part II: Visual content abstraction and their indexing structure

In this part, first representative frames (R-frames) are selected for each shot. Then image features are extracted from the R-frames. The dimensionality of features may need to be reduced. An appropriate multi-dimensional indexing structure needs to be chosen to index the image features.

R-frame selection: We use a clustering based method to select the representative frames of each shot. We classify the frames in a shot into one or more clusters depending on the amount of motion in the shot. The frame closest to the center of each cluster is selected as a R-frame. Shots with lots of motion usually have more than one R-frame. This approach is more accurate since the frame closest to the center of each cluster is usually more representative compared to the first frame or the last frame as chosen by some other R-frame

selection methods.

Image feature extraction from each R-frame: We use color histograms as the basic representation of image features. Other features such as shape or texture are not discussed in this thesis, but they can be introduced in the future. Color histogram distance measure with a color similarity matrix is integrated in our system, since the measure is closer to human perception.

Dimension reduction: We choose and implemented the Principal Component Analysis(PCA) technique to reduce the dimensionality of the image features (i.e. color histogram). PCA is a dimension reduction technique that transforms the original data into a new data space in which most of the variances are captured in the first few dimensions, while at the same time preserves the Euclidean distances between data points. Our experiments demonstrate that PCA is very effective. Using PCA, a 256-bin color histogram can be reduced to less than 10 dimensions while preserving more than 90% of the distance information.

For the color histogram distance measure with color similarity matrix, PCA cannot be applied directly due to its quadric form. We solved the problem by exploiting a transformation derived from the color similarity matrix, to transform the data into a new space in which the Euclidean distance is equal to the quadric form distance in the original data space.

Indexing structure for image features: We chose the bucket adaptive k-d tree as the indexing structure in our prototype system, since it handles point data very well and color histograms, the image feature vector used in our prototype system, are typical type of point data.

Part III: Semantic content abstraction

Predicate for annotation: We introduce predicates for video annotation. Predicates are rich in descriptiveness and can more precisely express complex relations among objects and events. They can also support inferences between descriptions easily.

Knowledge base for annotation derivation: There are many types of relations among the annotations: synonym, taxonomy and logical implications. Using inference rules, these relations can be supported easily. Exploiting these relations can remove redundancies in the annotations. The human annotator only needs to key in the base annotation and the inference rules, and the machine can derive the complete annotation. In this way, the work of a human annotator can be reduced.

We introduce a knowledge base to store the inference rules and annotations that are reusable. Knowledge specific to a certain domain can be used in a number of videos in that domain, and common knowledge can be used in the annotation of all the videos.

Probability inference rules: We also introduce probability inference rules by which a semantic annotation may be derived from visual features. Probability rules can improve the effectiveness of the query at times when accurate annotations are not available. Such inferences may not be 100% true, but it will only generate false positives but no false dismissals. False positives generated by probability rules are basically harmless since the human user is the final judge of the relevance of the retrieved material.

In summary, the key contributions of this thesis are as follows:

1. We evaluate and integrate/implement several existing techniques including shot boundary detection, bucket adaptive kd-tree and PCA, and confirmed their effectiveness.

2. We improved R-frame selection by using a clustering based method.

3. We propose a novel model for semantic video abstraction using predicate based video annotations. We introduce a knowledge base to allow derivation and reuse of annotations. We also introduced probability based rules by which a semantic annotation may be derived from visual features.

4. We implement a prototype content-based video retrieval system which integrates all the above techniques. A simple query interface is also provided to support queries based on video structure, visual feature, and semantic annotation.

1.5 Thesis outline

The remainder chapters of this thesis are organized as follows: Chapter 2 gives the related works about three important issues in content based indexing and retrieval: video parsing, video content abstraction, and high dimensional indexing methods. Chapter 3 investigates the shot boundary techniques. Chapter 4 describes indexing and retrieval based on visual content of video. Chapter 5 discusses semantic video annotation. Chapter 6 presents some implementation details of the Web-based inferface of the prototype system of our framework. Chapter 7 concludes the thesis and discusses future work.

Chapter 2

Related Works

Three important issues in content-based video indexing and retrieval are: video parsing, video content abstraction, and high dimensional indexing structure for video abstractions. In this chapter, we will discuss some of the related works on these three issues.

2.1 Video parsing

As we have already presented in the first chapter, video needs to be structured in order to ease content-based retrieval. Most research works structure video into shot, scene, and sequence as defined in film theory. Video parsing includes two parts: shot boundary detection and shot organization. There has been significant progress on automatic shot boundary detection, but very few on automatic shot organization.

2.1.1 Shot boundary detection

Shot boundary detection is a common first step for many video retrieval and browsing applications, and much effort [NT92, ZKS93, ZMM93, ATe92, Sha95, HJW94,

BR96, SP95, CB95, LZ95, DLM+96, lYL95, MJC95] has been put into this area.

There are two types of transitions between shots: the abrupt change such as a cut which is the natural boundary of the scene, and the gradual transitions such as fade, dissolve and wipe which are usually caused by video editing.

Two consecutive frames with significant difference between them tend to indicate that an abrupt shot transition has occurred. If we can express those differences by a suitable quantitative measure, then we can declare a shot boundary whenever that measure exceeds a carefully chosen threshold. Therefore, the key issue in locating abrupt shot boundaries are the selections of suitable difference measures and thresholds. The major difference measures that have been used include pixel differences, statistical differences, histogram differences [NT92], edge differences [ZMM93], and motion vectors [AT+92].

Gradual transitions are more difficult to detect since the differences between consecutive frames during a gradual transition are less significant than the differences observed in abrupt shot changes, and are often quantitively comparable to the inter-frame differences caused by the moving objects in the shot or camera movement. Among all the gradual transition detection algorithms, the twin thresholds approach [ZKS93] stands out because it is simple and effective. The basic idea behind the algorithm is that the inter-frame difference between consecutive frames are relatively larger during the gradual transition than before and after the transition (threshold 1), and the accumulated difference between the starting and ending frames of the gradual transition are significant and comparable to a clear cut (threshold 2).

Recently, some researchers are focusing on the shot boundary detection based on compressed video ([LZ95], [lYL95], [SP95], [MJC95]), avoiding the need to decom-

press the frames. Essentially these works are trying to define some new characteristic measures that can utilize the expensive parameters of the images that are readily available in the compressed video, such as DCT coefficient in MPEG/Motion-JPEG and motion vectors in MPEG.

2.1.2 Classification and organization of video shots

Shots are the building blocks, but not the only embedded logical structure in a video. A shot by itself is usually too short and thus insufficient to tell a complete story. In modern cinema, montage, "the editing of the film, the cutting and piecing together of exposed film in a manner that best conveys the intent of the work", is extensively used to combine shots into intellectual contexts and series.

These properly combined shots often can demonstrate continuity of the meaning despite of the discontinuity of the presentation. Some very important semantic information is not embedded in any shots. Instead, it is conveyed through the context, the editing and ordering of shots. So some higher level organization units are needed in the index to reflect the inherent logical structure of the video. One common way of organizing the shots is to structure them into scenes and sequences, just like words can form sentences and paragraphs.

[Ze94] presents a news parsing algorithm in which the partitioning is done at both the syntactic and semantic level. It locates the boundaries between the shots as well as news items. Since the automatic extraction of semantic information is beyond the capability of current machine vision techniques, an a priori model of video structure is needed to assist the classification of shots. This method only works on video programs, such as news, which follow a rather fixed pattern: a sequence of news items possibly interleaved with commercials, each of which may include an

anchor-person shot at the beginning and/or end. The whole classification process contains three steps. First, the spatial and temporal model of a certain type of shot is defined. Second, a similarity measure is developed which will be used to determine whether or not a shot and the model matches. Finally, a temporal structure of the entire video is used to finalize the classification.

[BYY96] introduced the time constrained clustering method to classify shots into more meaningful story units which are close approximations of scenes. The classification is based on both visual characteristics and temporal locality of shots. The basic idea of their approach is to take advantage of the repetition of shots with similar contents which are commonly used in montage to represent parallel or simultaneous events. The constraint on time prevents shots that are far apart in time from being clustered together.

2.2 Video content abstraction

2.2.1 Text-based abstraction

Text description can be attached to video segments in a video database and can be indexed using traditional database techniques. Video segments can be retrieved based on the text descriptions associated with them. In the following, we describe some of the existing text-based abstraction methods.

Keyword

Keywords are the simplest form of text-based annotations. They can be attached to a video segment to describe its content. When a keyword matches the query, the video segment associated with the keyword is returned to the user. Keyword

annotation is inadequate in several aspects. First, it does not support inheritance, similarity, and inference between descriptions. Second, it is not rich in descriptiveness due to its unstructured nature. Third, it is difficult to express complex relations since keywords are independent of each other. Some research work has been done trying to extend the keyword approach by adding new capabilities.

Attribute-value pair

In the paper [DSS96], annotations are represented in terms of attribute-value pairs. Both the attribute and value can be arbitrary strings. This extends the keywords scheme in two ways. First, the attribute-value pair structure in the annotation makes the approach richer in expressiveness when compared to the flat unstructured key-word approach. Second, this approach supports attribute synonym matching to avoid inconsistencies in the attribute names given by different users. When comparing two attribute-value pairs, the similarity between the two attributes are considered as well as the values of the two attributes.

Object-oriented frame

In paper [FSZ95], frames, a form of knowledge representation, are used to annotate the shots. Each frame is a structure that contains several slots. Each slot can store an object of a certain class describing a certain attribute or a set of attributes of the shot. For example, a frame may contain slots that store time objects describing the starting and ending time of the shot. Each object belongs to a class, and classes are organized into a hierarchy that allows inheritance. Here inheritance means a class at a lower position in the hierarchy inherits attributes from its ancestors above it. Though this approach supports inheritance by exploiting the capability of an object

oriented system, it lacks the descriptive richness due to the rigid format of the slot structure.

2.2.2 Visual feature based abstraction

Visual feature based abstraction is used to describe the visual content of the video. Typically, key frames are first selected or constructed from the video. Image features of the key frames are then extracted and used as the visual feature abstraction. In this fashion, the problem of video retrieval is somewhat transformed to a problem of image retrieval.

Key frames

The most common visual representation of the video shot is through a set of key frames. This technique transforms the video database into an image database and most of the temporal information is lost. But this approach is very simple, and can be done automatically [zls95]. The most naive method of selecting key frames is to pick the first, last or middle frame of the shot as the key frame of the shot. [ZLS95] introduced a more elaborate method in which one or more representative frames are selected for each shot depending on the variances of the frames in a shot. The algorithm initially picks the first frame as the key frame, then all the subsequent frames are compared with the key frame. If the difference is greater than the threshold, that frame is selected as a new key frame and all the subsequent frames are compared with the new key frame. The procedure repeats until the end of shot is reached.

Salient still is an augmentation of the pure key frame approach in that it attempts to preserve the camera operation in the representation. An single image

that combines the features of all the frames in a shot is constructed and used as the representation. Camera pan can be represented using a panorama image, and zoom can be represented using the broadest view of all the images.

Image features and their similarity measures

Color Color histogram is by far the most commonly used method to represent the distribution of colors in an image. An N-bin color histogram can be defined by a vector $(h_1, h_2, ..., h_N)$ where each bin i represent a color in a given color space, and each element h_i is the number of the pixels that are most similar to the color i. h_i maybe normalized to represent the percentage of the pixels with color i.

Many distance calculation methods have been proposed to measure the similarity between two color histograms. The simplest solutions are to use L_1-norm [SB90] or L_2-norm(also called Euclidean) distance metrics. The problem with these methods is that all the color bins are independent. In other words, the perceptual similarity between different colors are not taken into account by these distance measures. This will cause two images with perceptually similar colors but none in common to have the maximum distance. For example, the distance between a pure yellow image and a pure orange image, which are perceptually similar, is the same as the distance between a pure yellow and pure black image, which are perceptually very different.

To overcome this problem, Ioka proposed a quadratic form histogram distance [Iok89] which takes into account the "cross talk" between colors. If $\vec{Q}$ and $\vec{I}$ are two N-bin color histograms, according to [Iok89], then the distance between $\vec{Q}$

and $\vec{I}$ is defined as

$$D(I,Q) = (I-Q)^t A(I-Q) = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}(x_i - y_i)(x_j - y_j) \qquad (2.1)$$

where A is a $N \times N$ symmetric color similarity matrix.

The above approach has been demonstrated to have desirable performance. However, the quadratic histogram distance is computationally intensive, and it can not be used together with existing multi-dimensional indexing structures because of its non-Euclidean distance nature and its high dimensionality (Current multi-dimensional indexing structures such as k-d trees, and R-trees can only support low dimensionality (e.g. less than 20)).

In order to solve these problems, H. Sawhney proposed a filtering based approach [SH93]. The basic idea is that the retrieval proceeds through two steps. First, a cheap distance measure based on dimension reduced histograms will be used on the whole database. The reduced vectors are precomputed and can be indexed with a multi-dimensional indexing structure such as R-tree or k-d tree. They provide a way to construct the dimension reduced histogram vectors and a distance formula that will under-estimate the original distance. The under estimation of the distance guarantees the answer will include all the records that are desirable, but will also include some false positives. This is called the filtering step. Then, a more accurate matching using full color histograms will be performed only on those records that have passed through the filter.

This filtering approach leads to considerable savings on processing time, because now the quadratic distance is computed only on a much smaller set of images. The distance measure used in the first step is less expensive, and it can be used in many existing multi-dimensional indexing structures since the dimensionality has been reduced. So ideally this approach should be able to achieve efficiency without

missing any hits.

Based on this approach, Dominic Tam proposed multi-level filtering [Tam96]. As described in section 1.3.3, Dominic's approach basically feeds the images which passes coarest filtering stages to other indexed filters before the final non-indexed search are performed on the original image feature vector. By adding additional intermediate levels of filtering stage, both the coarsest and finest filtering stages can be very efficicient.

2.3 Indexing high dimensional video feature vectors

Indexing is an important technique used to speed up the query answering process. An index is a data structure that can help to quickly and easily find the data of interest. Indexing is especially important for multimedia applications due to the huge volume of image and video data. Most traditional indexing structures are not suitable for indexing video data because of the high dimensionality of video feature vectors. Hence techniques such multi-dimensional indexing structure and dimension reduction need to be explored in order to support video indexing.

2.3.1 Multi-dimensional indexing structure

There are two types of commonly used multi-dimensional data: point data and rectangle data. An image can usually be represented using a high dimensional feature vector, such as vector of color histograms, which are point data. An object with a certain shape can usually be represented using its minimum enclosing rectangle. Such bounding rectangles can provide both the position and the extent of the object. The indexing structure on rectangles usually functions as a filter which is used to minimize the access to real shape information. K-d tree and its variants are

widely used when indexing multi-dimensional points, while R-tree and its variants are commonly used when indexing multi-dimensional rectangles.

K-d tree and its variations

The k-d tree invented by Bentley in 1975 [Ben75], is essentially an extension of the binary search tree for multi-dimensional points where k denotes the dimensionality of the space being represented. The major difference between a k-d tree and binary search tree is that in the binary search tree there is only one discriminatory, the attribute/key used to decide which branch to traverse; while in a k-d tree there are multiple discriminators available. Only one discrimination key is used at each level, and the keys chosen at each level circulate among all the available keys as the tree is descended.

The problem of the k-d tree is that the shape of the tree heavily depends on the order in which the nodes are inserted. In the worst case, a k-d tree can degenerate into a linear linked list. Moreover, the key chosen may not be the optimum in terms of discriminative power, since the discrimination key at each level is fixed. The adaptive k-d tree [Sam90] tries to alleviate some of the problems of the standard k-d tree. In the adaptive k-d tree, data are stored only in leaf nodes, and the key chosen at each level has the highest variance of value range. This approach is static in that it requires all the data to be known a priori in order to build the tree. The bucket adaptive k-d tree is similar to the adaptive k-d tree. The only difference is that the leaf nodes are now buckets with capaticty c ($c > 1$) instead of single data points. The size of bucket is chosen corresponding to storage unit (i.e., page) size of the disk to ensure efficient disk access.

Other dynamic approaches have also been proposed, but they are either not

very efficient due to the frequent rearrangement or not well-suited for range searches.

R-tree and its variations

R-trees are extensions of B-trees for multi-dimensional objects that are either points or rectangles. Like B-trees, they are balanced, i.e. all leaf nodes appear on the same level.

In the R-tree the data and the indexing keys are all represented using hyper-rectangles. The leaf nodes store the actual data and the interior nodes store the minimum rectangle that encloses all the objects in its subtree. R-trees and their variants differ in the strategies of how the data rectangles are grouped and how the interior indexing rectangles are created.

The original R-tree [Gut84] creates the interior nodes in a way that the area of the indexing rectangles are minimized. The problem with this approach is that indexing rectangles in sibling interior nodes may overlap such that queries for certain data rectangles may unnecessary follow multiple paths in the index tree.

In order to solve the rectangle overlapping problem, the R+-tree [SRF87] structure has been proposed. In an R+-tree, the indexing rectangles are no longer required to completely enclose all the data rectangles in its subtree. A data rectangle can be split and covered by different higher level indexing rectangles, and its corresponding data entry is duplicated in each of its overlapping indexing rectangles. R+-tree is shown to have better search performance.

R*-tree [BKSS90] is another R-tree variant which tries to integrate several criteria into the optimization strategy. These criteria include minimizing the area of the indexing rectangles, minimizing the overlap between indexing rectangles, minimizing the sum of lengths of the rectangles and optimizing the storage utilization.

Experiments show that R*-tree outperforms the original R-tree and R+-tree.

2.3.2 Principal Components Analysis for dimension reduction

Though the multi-dimensional indexing structures introduced in the previous section have provided the possibility of indexing high dimensional data, there is still a big gap between the number of dimensions that are needed to describe multimedia data, which is usually over 64, and the maximum number of dimensions that can be supported, which is less than 20. Obviously some kind of dimension reduction technique has to be introduced before these indexing structures can be applied to the multimedia data.

Dimension reduction algrithms operate by identifying and eliminating statistical redundancies in the data. The optimal linear technique for dimension reduction is principal components analysis (PCA), which has been applied in many applications to reduce the dimensionality of the original data set. Following is a description of PCA by Jolliffe [Jol86]: "The central idea of principal component analysis is to reduce the dimensionality of a data set which consists of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables."

2.4 Content-based image and video retrieval systems

2.4.1 QBIC

QBIC ([NBe93], [FEe94]) is an image database system, which allows query by image content such as color, texture , and shape of image objects and regions. The color feature of an object or image is presented as average color and N-bin color histogram. Two-level filtering approach is used when retrieving image based on color features. Texture is represent as modified coarseness, contrast, and directionality features [TMY78]. Similarity between textures of two images is computed using weighted Euclidean distance in the three dimensional texture space. Query by shape is one of the most challenging aspect to QBIC. Currently, they use a total of 20 shape features which includes area, circularity, eccentricity, major axis orientation and a set of algebraic moment invariants [N93+]. The similarity between two shape vectors is based on weighted Euclidean distance where the weights reflect the importance of each features. Query can be asked on any one or combination of the above mentioned image features.

QBIC supports two ways of specifying a query, namely query by example and direct query. In the first way, the user can choose one of displaying images, and ask for all the images with a certain pattern similar to the selected one. Direct query allows user specify desired image features directly using provided tools, e.g., selecting colors from multi-color picker or drawing a sketch.

2.4.2 ViewStation

ViewStation [?] is a sytem, developed at computer science deparment of MIT, that is capable of transferring and displaying multimedia information. It supports au-

tomatic content-based indexing of digitized video programs by capturing and processing the closed-captioned information of video. On the ViewStation, captions are first captured and translated from raw video caption signal into text. Then, a Caption Parser module is used to break captions into seperate components, and the associated video sequence is also divided into corresponding smaller meaningful units that can be replayed and examined seperately.

Chapter 3

Shot Boundary Detection

In this chapter, we present a comparison of several shot boundary detection techniques which includes the histogram approach, region histogram approach, motion compensated approach, and twin thresholds approach. The algorithms are evaluated based on the performance of our implementation of these algorithms on a set of benchmark video clips.

3.1 Review of the shot boundary detection algorithms

3.1.1 Histogram approach

A simple and common the used in shot boundary detection is histogram difference comparison. The principle behind this is that two frames having an unchanging background and objects will show little difference in their respective histograms. Although there are some cases in which two images have similar histograms with completely different contents, the probability of such an event is so low that we can tolerate such errors in practice.

Nagasaka and Tanaka [NT92] use the following formula to measure the frame-to-frame histogram difference:

$$D_i = \sum_{j=1}^{G} |H_i(j) - H_{i+1}(j)| \tag{3.1}$$

where H_i denotes the histogram value for the ith frame and j is one of the G possible gray levels or color bins; D_i refers to histogram difference between frame i and frame $i+1$. If the overall difference D_i is larger than a given threshold T, a shot boundary is declared.

In most cases, this approach can successfully identify camera shots. But there are two possible errors that may occur:

- false positive which is mainly caused by moving objects of either large size or high speed, or high speed camera motion, or a sharp illumination change between two frames within a shot.

- false negative which is mainly caused by video editing. Most video editing procedures impose a gradual transition between two shots, such as fading or dissolving. These gradual transitions cause the histograms of successive frames during transition to be quite similar so that the inter-frame histogram difference is not significant enough to be distinguished from noise caused by object motion and illumination changes.

3.1.2 Region histogram approach

When the illumination does not change over the entire frame or motion is not very large and fast, the number of false positives can be reduced by a robust histogram approach developed by Nagasaka [NT92]. It is based on the assumption that the momentary noise usually influences no more than half of an entire frame. Therefore,

a frame can be divided into a 4X4 grid of 16 rectangular regions; then, corresponding regions instead of entire frames are compared. This yields 16 difference values, and the camera break detection is only based on the sum of the eight lowest difference values. This region histogram approach is more robust in the presence of object motion and illumination changes, but it also increases the probability of missing gradual scene change or even some cuts, since it lessens the difference between two frames.

3.1.3 Motion compensated approach

In the region histogram approach, the region comparison is done by superimposing each block of the first frame on exactly the same location of the second frame. A more robust measure [Sha95] is proposed by utilizing a block matching process in which each of the K blocks in the first frame is compared against the best fitting region of the second frame in a neighborhood of the corresponding block. This method is not only more robust in the presence of local or global motion between two frames, but also has the advantage of extracting motion information for each block.

The following steps are used in our implementation which is very similar to the algorithm described by Shahraray [Sha95]. First, we divide each frame into K blocks. For each block B_i in a frame, we search within a certain window trying to find the best match B_m in the next frame. We use normalized mean-squared difference given by Pascal Fua in the paper [Fua93] to measure correlation between B_i and a trial block B_{it} in the next frame:

$$C = \frac{\sum((I_i - \overline{I}_i) - (I_{it} - \overline{I}_{it}))^2}{\sqrt{\sum(I_i - \overline{I}_i)^2 \sum(I_{it} - \overline{I}_{it})^2}} \tag{3.2}$$

Where C denotes correlation value between B_i and B_{it}; I_i and I_{it} are the intensity values of a pixel in B_i and B_{it} respectively; and $\overline{I}_i$, $\overline{I}_{it}$ are the mean intensity of pixels in B_i, B_{it} respectively. A small correlation value means the two blocks are similar. For all the trial blocks B_{it} within the search window in the second frame, we calculate the correlation between B_i and B_{it}. The best match B_{im} is declared when the smallest correlation value C_{min} is found. The difference between the positions of block B_i and B_{im} is the motion vector which can be utilized to analyze camera motion later. The inter-frame difference is then defined as:

$$D_i = \sum_{j=1}^{k} C_j \tag{3.3}$$

where D_i is the difference between frame i and $i+1$, C_j denotes the correlation value between block j in frame i and its best matching block in frame $i+1$. If this difference value exceeds a certain threshold, a scene change is declared.

3.1.4 Twin thresholds and motion analysis

The region histogram approach and the motion compensated histogram approach are better than the histogram approach in the sense that they are more robust against the noise caused by object motion and illumination change. But none of them can detect gradual shot transition. A naive way to detect gradual shot change would be lower the threshold. However, this cannot be effectively employed because if the cutoff threshold is too low then too much false detection will be caused. In the paper [ZKS93], Hongjian Zhang developed a new approach, twin thresholds approach, to detect gradual shot change. This is a major improvement over the methods based on the comparison of the difference value against a single threshold.

Twin thresholds

The following is Zhang's twin thresholds method [ZKS93]: in twin-comparison, two cutoff thresholds are used. T_b is used for camera break detection in the same manner as was described before. A second, lower threshold T_s is used for gradual shot change detection. The detection process begins by comparing consecutive frames using a difference measure such as region histogram difference. Whenever the difference value exceeds threshold T_b, a camera break is declared. In addition, the twin-comparison also detects differences that are smaller than T_b but larger than T_s. Any frame that exhibits such a difference value is marked as the potential start of a gradual transition. This frame is then compared to subsequent frames. The difference measured is called the accumulated difference. During a gradual transition, this accumulated difference value will normally increase. The end frame of the transition is detected when the frame-to-frame difference decreases to less than T_s, while at the same time, the accumulated difference has increased to a value larger than T_b. The accumulated difference is only computed when the difference between consecutive frames exceeds T_s. If the frame-to-frame difference value drops below T_s before the accumulated difference value exceeds T_b, then the potential start point is dropped and the search continues for other gradual transitions or camera breaks.

Such an approach performs well in detecting gradual transitions, but it also increases the possibility of false detection caused by some camera operations such as panning and zooming, because changes due to camera movements tend to induce successive difference values of the same order as those in gradual transition. In paper [ZKS93], motion detection and analysis techniques are applied to distinguish camera movements from gradual transitions.

Motion analysis

The specific feature that serves to detect camera movement is optical flow which is also called motion vector. The optical flow fields resulting from camera pan and zoom have some special patterns that are not shared with gradual transitions. Therefore, if such motion vector fields can be detected and analyzed, then changes induced by camera movements can be distinguished from those due to gradual transition. During a camera pan, motion vectors will have relatively large length and predominantly the same direction (Certainly, if there is also object movement in the scene, not all vectors need share this property). In the case of zoom, the field of motion vectors has a minimum value at the focus center. If the focus center is located in the center of the frame and there is no object movement, then the mean of all the motion vectors will be the zero vector. For simplicity, we assume the focus center is always located in the center of the frame.

In our implementation, the block matching algorithm, which is the same as described in section 3.1.3, is used to get the field of motion vectors. This algorithm requires partitioning of the frame into blocks, and motion vectors are computed for each block by finding the minimum value of correlation over a set of trial vectors. Then the mean length L, mean phase Φ, the phase deviation σ_Φ of the motion vectors, along with the mean motion vector are calculated. If the length L is large and σ_Φ is small, which means the vectors are almost parallel, then it is a camera panning. If both the length L and σ_Φ are large and the mean vector is close to zero, then a camera zooming is declared.

3.2 Experiment

In our implementation, we use a 256-bin color histogram in the histogram approach, and 4X4=16 regions with a 256-bin color histogram in the region histogram approach. For motion compensated approach, after block matching, color histogram, instead of correlation value, is used to measure inter-frame differences since the latter performs poorly in scene change detection. Here we only give the result of 4x4 blocks with (-10, 10) search range which appears to have the best result in our experiment. In the test, if the multiple consecutive frames are detected as shot boundaries, only one is counted.

For the threshold used in shot change detection, we use following formula:

$$Thresh = m + \alpha d \tag{3.4}$$

Where m is the mean of the frame-to-frame difference and d is the standard deviation of difference values. The selection of α varies according to different detection methods and video sources.

The test is conducted on two sets of video clips: one involves intensive object motion and camera movement, and the other contains a lot of gradual transitions. Using these two sets of data, we can test both the robustness of a method against noise and its ability to detect gradual transitions.

3.2.1 Experiment Result

- test on videos containing motions

  The following tables give our test results, in which N_c stands for the number of shots correctly detected; N_m stands for the number of shots missed; N_f stands for the number of false detections.

tested file: pirates.mpg, num of frames=280, num of shots = 18, type:movie											
histogram approach m=59, d=99, runtime=0.3s				region histogram approach m=39, d=60, runtime=3.9s				motion compensated histogram m=37, d=59, runtime=197.4s			
α	N_c	N_m	N_f	α	N_c	N_m	N_f	α	N_c	N_m	N_f
1.2	18	0	4	2.5	18	0	2	1.2	18	0	1
1.5	17	1	3	3.0	18	0	1	1.4	18	0	0
1.8	15	3	2	3.5	18	0	0	1.6	18	0	0
2.0	12	6	1	4.0	17	1	0	1.8	17	1	0

Table 3.1: Test result for pirate.mpg

tested file: us.mpg, num of frames=123, num of shots = 9, type:movie											
histogram approach m=38, d=66, runtime=0.1s				region histogram approach m=20, d=38, runtime=1.6s				motion compensated histogram m=25, d=35, runtime=90.4s			
α	N_c	N_m	N_f	α	N_c	N_m	N_f	α	N_c	N_m	N_f
0.8	9	0	2	3.0	9	0	0	1.0	9	0	0
1.0	8	1	1	3.5	9	0	0	1.2	9	0	0
1.2	7	2	0	4.0	8	1	0	1.4	8	1	0

Table 3.2: Test result for us.mpg

- test on videos containing gradual transitions The tests of twin threshold approach carried on two 8 minutes long video clips which contain totally 125 gradual transition. The approach successfully detected 105 of them.

3.2.2 Summary of the experimental results

- The region histogram approach is simple and efficient. It is a good choice for the application where video sources do not contain many gradual scene changes, correctness is important, and missing some scene changes is not so crucial.

- If a video program contains many special effects like dissolving and fading, then the twin thresholds approach with motion analysis is a good solution. However, it may need a better filter to reduce the number of false detections,

tested file: perry.mpg, num of frames=548, num of shots = 12, type: animation											
histogram approach m=33, d=67, runtime=0.5s				region histogram approach m=26, d=37, runtime=10s				motion compensated histogram m=22, d=40, runtime=646.9s			
α	N_c	N_m	N_f	α	N_c	N_m	N_f	α	N_c	N_m	N_f
1.8	12	0	1	3.5	12	0	1	1.4	12	0	1
2.0	12	0	1	4.0	12	0	1	1.6	12	0	1
2.5	11	1	1	4.5	11	1	1	1.8	12	0	1
3.0	11	1	0	5.5	10	2	1	2.0	11	1	1

Table 3.3: Test result for perry.mpg

since this method can only handle false positives that result from pans and zooms, but cannot deal with those arise from object movement or from more complex camera motions.

- Motion compensated histogram method is an improvement of the region histogram approach in the sense that it can reduce some noise caused by large object motion. But this slight improvement is obtained at the cost of great increase of computational complexity. Thus it might not be a wise choice. However, if an application is also interested in camera movement analysis, then the computation of motion vectors is inevitable. In that case, we can use these motion vectors to further improve the result of region histograms in shot change detection.

- Threshold selection is very crucial to the result of scene change detection. Setting a high threshold will reduce the number of false positives at the cost of missing more scene changes, while a lower threshold will get the opposite result. So threshold selection requires a trade-off between false negative and false positive according to different application scenarios.

3.3 Conclusion

For the video "American castle" which is used in our video retrieval system, twin threshold approach with region histogram difference is selected as the shot boundary detection algorithm since this approach has good perfomance on detecting gradual transitions which occur frequently in video "American castle".

Motion analysis discussed in the section 3.1.4 is not incorporated in the detection procedure since it is too expensive and can not handle the complicated combinations of camera movement and object motion.

Chapter 4

Video Retrieval Based on Visual Content

In this chapter, we will discuss the second part of our framework: visual content abstractions of video and their indexing structures.

4.1 R-frame selection

In this section, we will present an algorithm that can select one or more frames as representative frames (R-frames) for each shot. Our approach is to first group all the frames in a shot into a number of clusters using a certain clustering method, and then to select the centrally located frame in each cluster as an R-frame. In this way, a shot is well represented by R-frames.

In our implementation, we choose PAM (Partitioning Around Medoids), a k-medoid based clustering method developed by Kaufman and Rousseeuw [KR90], as the basis of our algorithm. PAM works satisfactorily on small data sets, e.g. 100 objects in 5 clusters [NH94]. So it is considered suitable for our application, in which

a shot usually contains dozens to hundreds of frames and less than 5 R-frames.

4.1.1 PAM

To find K clusters, PAM first determines a representative object (called medoid) for each group, which is the most centrally located object within the cluster. Then each non-selected object is grouped with its most similar mediod.

Initially, PAM arbitrarily selects K objects. Then, in each iteration, all possible swaps between a selected object O_i and a non-selected object O_h are evaluated, and the most effective swap in terms of improvement in the quality of the clustering is made. This procedure repeats until no swap can lead to a better clustering.

The quality of the clustering can be measured by the average distance from an object to its corresponding medoid. The cost of a swap can be evaluated by comparing the quality of the clustering before and after the swap. The most effective swap is the one that can lead to the most reduction in the average distance.

The cost C_{jih} of a swap between a selected object O_i and a non-selected object O_h as far as object O_j is concerned is calculated as follows:
If O_j belongs to the cluster represented by O_i, and $O_{j,2}$ is the second most similar medoid to O_j, then

$$C_{jih} = \begin{cases} d(O_j, O_{j,2}) - d(O_j, O_i) & \text{if } d(O_j, O_h) \geq d(O_j, O_{j,2}) \\ d(O_j, O_h) - d(O_j, O_i) & \text{if } d(O_j, O_h) < d(O_j, O_{j,2}) \end{cases}$$

If O_j belongs to the cluster represented by the object other than O_i, say $O_{j,2}$, then

$$C_{jih} = \begin{cases} 0 & \text{if } d(O_j, O_h) \geq d(O_j, O_{j,2}) \\ d(O_j, O_h) - d(O_j, O_{j,2}) & \text{if } d(O_j, O_h) < d(O_j, O_{j,2}) \end{cases}$$

where $d(O_j, O_h)$ denotes the distance between two objects O_j and O_h. The total

cost of replacing O_i with O_h is

$$TC_{ih} = \sum_j C_{jih}$$

Following is the PAM algorithm given in [NH94]. A detailed discussion of the algorithm can also be found there.

Algorithm PAM

1. Select K representative objects arbitrarily.
2. Compute TC_{ih} for all pairs of objects O_i, O_h where O_i is currently selected, and O_h is not.
3. Select the pair O_i, O_h which corresponds to the minimum TC_{ih} for all possible O_i and O_h. If the minimum TC_{ih} is negative, replace O_i with O_h, and go back to Step (2).
4. Otherwise, for each non-selected object, find the most similar representative object. Halt.

4.1.2 Algorithm for selecting R-frames

The number of R-frames in a shot is determined by the amount of motion in a shot, which can be evaluated by means of the variance of frames in the shot. If a shot is relatively static then only one R-frame is need. More R-frames should be selected when there is a lot of object movement or camera motion in the shot.

Following we present the algorithm for selecting R-frames. Here $MaxRep$ is the maximum number of R-frames that are allowed in each shot; TH is the threshold value used to determine the appropriate number of R-frames for each shot.

1. Let K=1.

2. Apply PAM to the set of frames in the shot to determine K medoid-frames, and assign frames in the shot to the groups represented by each medoid-frame. In step 1 of PAM, K trial medoid frames are needed. We pick the $K-1$ medoid-frames from the previous iteration as $K-1$ trial medoid-frames and randomly pick the Kth trial medoid-frame.

3. For each group calculate variance of color histograms of all the frames that belong to that group.

4. Compare all variances with the threshold TH. If any $variance > TH$ and $K < MaxRep$, then increment K and go to step (2).

5. Otherwise let the K medoid-frames be the R-frames for the shot and halt.

4.2 Image feature extraction from R-frames

Once the R-frames are selected, image features can be extracted from these frames as visual content abstraction of the video shots. As we have discussed in chapter 1 and 2, the image features can be color, shape, texture, etc. In our system, we choose color histogram as the visual content abstraction because of its simplicity and popularity. The system can be augmented later to include the support for other image features such as shape and texture.

The distance measure $d_{N,A}(\vec{Q}, \vec{I})$ between two color histograms $\vec{Q}$ and $\vec{I}$ is calculated using the formula 2.1 discussed in chapter 2. The color similarity matrix $A = \{a_{ij}\}$ is a $N \times N$ symmetric matrix, where N is the number of color bins in the histogram. Each element a_{ij} in the matrix is a real number between 0 and 1 denoting the similarity between color bin i and color bin j, with 1 representing the most similar and 0 the least. It is computed according to the formula given in

[SH93]:

$$a_{ij} = 1 - \frac{d(c_i, c_j)}{d_{max}} \tag{4.1}$$

where c_i and c_j are the ith and jth color in the color histogram, $d(c_i, c_j)$ is the Euclidean distance between color i and color j in the CIE LUV color space, and d_{max} is the maximum distance between any two colors. CIE LUV space is used here because the Euclidean distance in this color space is close to human perception.

In our implementation, the number of color bins is 256, and thus the similarity matrix A is a 256×256 symmetric matrix.

4.3 Indexing visual features

In order to facilitate video retrieval, the extracted visual features need to be indexed using a multi-dimensional indexing method. There are two obstacles that prevent us from directly employing some existing multi-dimensional indexing structures: 1) the dimensionality of the feature vector may be too high; 2) the desired distance measure of the visual feature may not be Euclidean. In this section, we will discuss how these two problems are solved in our system using color histogram as an example. The bucket adaptive k-d tree is selected as the underlying indexing structure. Its performance on dimension reduced color histograms is evaluated.

4.3.1 Dimension reduction using PCA

First, let us deal with a simpler case in which the Euclidean distance is used as the color histogram difference measure. Thus we can focus on the dimension reduction of feature vectors, which is the solution to the first obstacle that we mentioned above.

A simple way to reduce the high dimensionality of color histograms, 256 in our experiment, would be the direct merge of color bins. However, this usually will lead to unacceptable loss of information. The method we used in our system is the principal components analysis (PCA), the optimal linear technique for dimension reduction. The principal components analysis of a data set is essentially a certain orthogonal transformation and a ranking of the dimensions by descending variance. This procedure concentrates most of the information in the first few dimensions so that those few dimensions can be used to represent the whole data set without losing much of the information.

Perform PCA on color histograms

In our system, PCA is performed on color histograms by the following procedure:

1. First we select m sample frames from the whole collection of R-frames of the videos (m is between 1000 and 3000 in our experiment). The color histograms of sample frames form a data matrix $D = [H_1, H_2, \cdots, H_m]$ where $H_i (i = 1, 2, \cdots, m)$ is the n-bin color histogram column vector of sample frame i (n=256 in the experiment). So here D is an $n \times m$ matrix.

2. Next, the average color histogram of the m sample images is calculated:

$$H_a = 1/m \sum_{i=1}^{m} H_i$$

3. Then the covariance matrix C of m sample color histograms is computed:

$$C = \frac{1}{m-1} D' D'^T$$

Where $D' = [H_1 - H_a, H_2 - H_a, ..., H_m - H_a]$

4. The PCs of the data set can be obtained by computing the eigen values λ_i and eigen vectors b_i of the covariance matrix C. The eigen vectors are ranked in decreasing order according to their corresponding eigen values.

5. The first few PCs with largest eigen values will be chosen and the rest of PCs be discarded. In our experiment, we choose the first k PCs with over 90% of total variance. Here k is the smallest number that satisfies the following condition:

$$\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{m} \lambda_i >= 0.9 \tag{4.2}$$

The first k PCs then form the transformation matrix $T = [b_1, b_2, ..., b_k]$.

The number of PCs that need to be selected depends on how much of the total variance an application wants to keep. A typical choice is the first K eigenvectors that contain between 60% to 80% of the variance in the original data set. The reason we choose 90% of the variance is that the result of PC calculation shows that the energy of our data set is highly concentrated. Such a choice can improve the accuracy of the approximated distance, and thus may avoid a subsequent refinition step using full dimension data without indexing structure, which is necessary if the approximated distance is not very close to the original one.

6. Finally, the original data can be projected into the new data space using the following formula:

$$Y = T^t(X - H_a)$$

where X is the original n-bin histogram, H_a is the average color histogram in the original data space, and Y is the transformed k-bin histogram. Here k is

a much smaller number than n.

Experiment result

In our experiment, we test PCA on four sets of video clips which contain movies, news, documentary, and a combination of all the above three categories. The results are similar on all of the four different data sets. Below we only present the result from the data set that contains a mixture of different video types, because we believe it is more representative. The data set contains color histograms extracted from 1000 R-frames in around 40 minutes of video material.

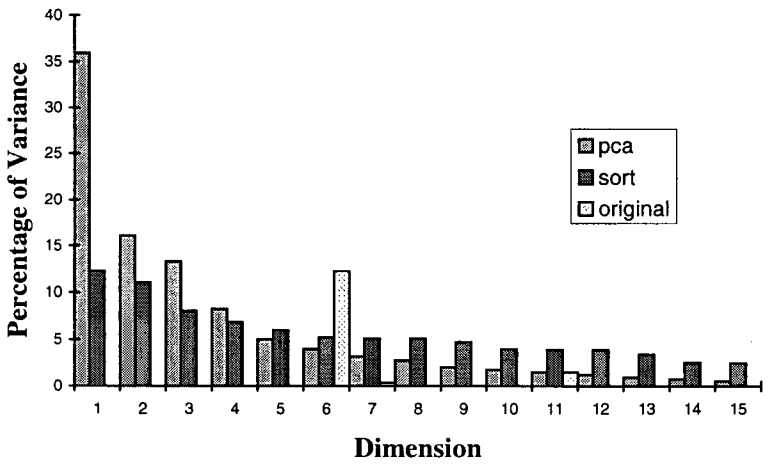

Figure 4.1: The energy distribution

Figure 4.1 is the energy (variance) distribution of 3 sets of color histograms, in which horizontal axis is the dimension, and the vertical axis is the variance in percentage that each dimension carries. Only variances of the first few dimensions are shown in the figure due to limited space. The first set of data is the original color

histogram obtained directly from the R-frames of the test video clips. The second is the color histogram with color bins rearranged by decreasing variance. The third is the PCA-transformed color histogram.

The result shows that the energy of the original data (labeled as "original") is distributed sporadically throughout all the dimensions. After the ranking of the dimensions (labeled as "sort"), the dimensions with the most variances are moved to the front, and the energy is concentrated to some extent. After performing PCA on the color histograms, which essentially rotates and ranks the dimensions, the energy concentration is more significant. The first dimension after PCA actually accounted for over 35% of the total variance.

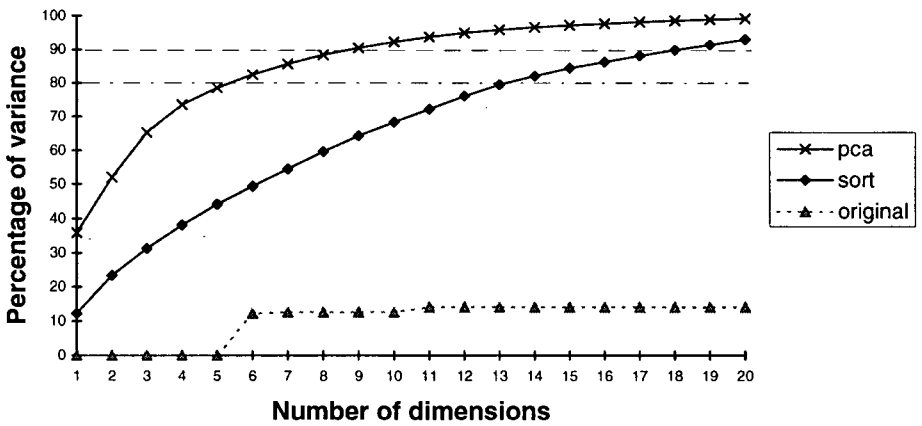

Figure 4.2: The accumulated variance of the first K dimensions

Figure 4.2 shows the accumulated variances in percentage that the first K dimensions contain. After PCA, the first 5 dimensions capture 80% of the total variance, and the first 9 dimensions cover over 90% of the variance of the original

data.

4.3.2 Transforming quadratic form distance to Euclidean distance

For color histograms, Euclidean distance is not a good distance measure in that it does not take into account the similarities among different color bins. So the full quadratic form distance measure involving all cross terms, as given in formula 2.1, is used in our system. However, for most of the existing multi-dimensional indexing structures, the distance in the search space is Euclidean. Moreover, PCA can only preserve Euclidean distance. Therefore, some sort of preprocessing is necessary before we can compress or index the color histogram.

Following we present a solution to the problem which is inspired from Sawhney's filtering approach [SH93]:

For any normalized color histogram x, y, the quadratic distance between x and y is:

$$d = (x-y)^T A(x-y) = z^T A z = [\tilde{z}^T z_N] \begin{bmatrix} A_{N-1} & a_{*N} \\ a_{*N}^T & a_{NN} \end{bmatrix} \begin{bmatrix} \tilde{z} \\ z_N \end{bmatrix} = \tilde{z}^T \tilde{A} \tilde{z}$$

where $z = x - y$, $\tilde{z}^T = [z_1 \cdots z_{N-1}]$, and $\tilde{A} = [A_{N-1} - a_{*N}1^T - 1a_{*N}^T + a_{NN}11^T]$.

It can be proven [SH93] that for the choice of a_{ij} in equation 4.1, $\tilde{A}$ is symmetric and is Positive Semi-definite(PSD). Thus $\tilde{A}$ can be decomposed into a product of three matrixes using Singular Value Decomposition (SVD):

$$\tilde{A} = B^T \Lambda B$$

where $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. The quadratic form distance measure can then be written as :

$$d = \tilde{z}^T \tilde{A} \tilde{z} = \tilde{z}^T B^T \sqrt{\Lambda}^T \sqrt{\Lambda} B \tilde{z} = (\sqrt{\Lambda} B \tilde{z})^T (\sqrt{\Lambda} B \tilde{z}) = (U\tilde{z})^T (U\tilde{z})$$

where $U = \sqrt{\Lambda}B$. By using matrix U, we can transform the original N-dimension color histogram x to a $(N-1)$-dimension histogram x':

$$x' = U\tilde{x} \tag{4.3}$$

where $\tilde{x}$ contains the first $N-1$ components of x. Thus the quadratic form distance on the original color histograms can be transformed to the Euclidean distance on U-transformed data. Now, we can apply the PCA method as described in section 4.3.1 to reduce the dimensionality of U-transformed data.

Our approach differs from Sawhney's approach [SH93] in that we use the transformation obtained from SVD only as a method to convert quadratic form distance to the Euclidean distance, so that we can subsequently apply PCA to reduce the dimensionality of the transformed data. Here we do not care whether the transformation U can concentrate energy or not. In Sawhney's approach, they are relying on U to reduce the dimensionality of the color histogram. First they derive another matrix U_k by selecting the first k rows of U. The original $(N-1)$ dimensional histogram vectors $\tilde{x}$ are then projected into a k dimensional data space:

$$x_k = U_k\tilde{x} \tag{4.4}$$

The Euclidean distance

$$d_k = \tilde{z}^T U_k^T U_k \tilde{z} = (U_k\tilde{z})^T(U_k\tilde{z}) \tag{4.5}$$

is used as an approximation of the original quadratic distance. Sawhney proves that for a given k, d_k can minimize the maximum difference between the true distance d and the approximation d_k for all possible $\tilde{z}$.

The reason that we do not directly adopt Sawhney's approach is that we think his approach is not the optimum in terms of dimensionality reduction since

it doesnot utilize the statistical properties of the data; nor does it use any a-priori model that well understands the data's statistical properties.

Experiment result

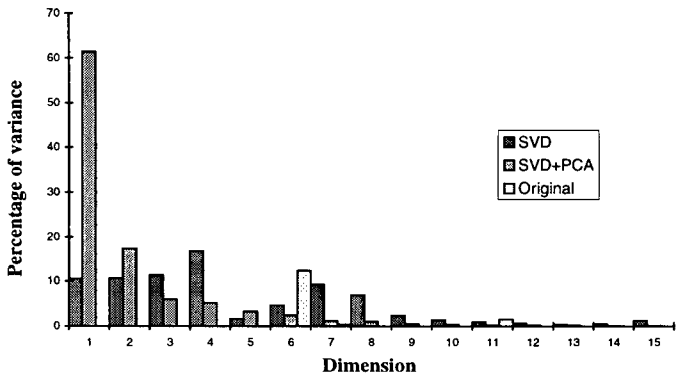

Figure 4.3: The energy distribution

To compare the effectiveness of PCA and SVD in terms of energy concentration, we conducted several experiments on the test data which are the same as those used in figure 4.1 and figure 4.2.

The energy distribution and accumulated variance that first K dimensions can capture are shown in figure 4.3 and figure 4.4. The first set of data are the results from Sawhney's approach (labeled as "SVD"), i.e., the data transformed from the original color histogram using the matrix U according to equation 4.3. The second set of data are the results from our approach (labeled as "SVD+PCA"), i.e., the data transformed by first using matrix U, then further processed by using PCA. The results from the original color histogram (labeled as "Original") are also shown

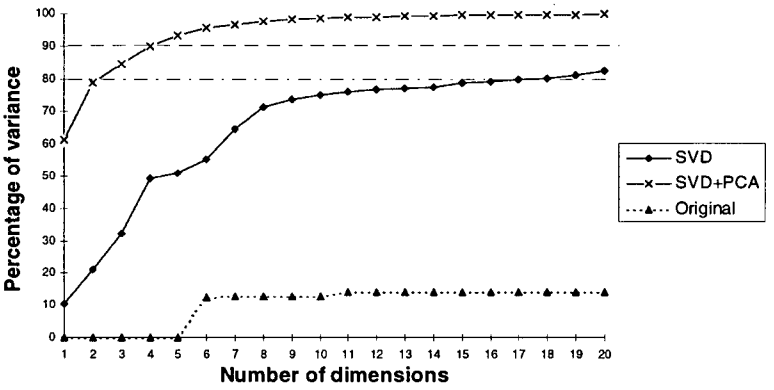

Figure 4.4: The variance of first K dimensions

here as a reference.

From figure 4.3 we can see that SVD does have some ability to concentrate energy, as some of the first few dimensions have relatively high variance. But we can also see that the dimensions are not arranged in decreasing variance. Applying PCA after SVD can significantly enhance energy concentration. The first dimension after PCA contains over 60% of the total variance. In figure 4.4 the difference between PCA and SVD are more evident. The first 5 dimensions after PCA account for over 90% of the total variance, while the first 5 after SVD account for only 50%.

From our experimental results, we can conclude that: 1) For image features with Euclidean distance, applying PCA can result in significant reduction in the dimensionality of the feature space while still keeping most of the information. This agrees with the result of QBIC [FEe94], in which PCA is applied to reduce the dimensionality of shape features, and the result from Andieshi's experiments on eigenimages. This further confirms the applicability of the PCA technique in di-

mensionality reduction of image features. 2) For image features with non-Euclidean distance, if possible, the data should first be transformed to a new space with Euclidean distance; then PCA can be applied to reduce the dimensionality.

4.3.3 Indexing dimension reduced data using bucket adaptive k-d tree

Once the dimensionality of the image features is reduced to an indexable size, the next stage would be to select an appropriate multi-dimensional indexing structure to organize the dimension-reduced image features. According to our application scenario, we need a multi-dimensional indexing structure for point data that supports efficient fixed radius search and/or n-nearest neighbor search which are often used in similarity match. The structure should be able to scale to a large number of data points, since the video database is expected to be large. It should also be able to support a relatively high number of dimensions.

The reason for chosing bucket adaptive k-d tree

Based on these requirements and the analysis of various multi-dimensional indexing structures given in [Sed96], we choose the bucket adaptive k-d tree (BA-kd tree) as our underlying indexing structure for the following reasons:

1. It is designed for point data.
2. It supports several search types including fixed radius search and n-nearest neighbor search.
3. It does not explode in index size for large data sets. And it does not seem to grow as rapidly as some other structures when the number of dimensions

increases.

4. PCA transformed data are usually non-uniformly distributed. But this non-uniform distribution has no significant adverse effect on BA-kd tree.

5. The dimensions at different levels of the indexing structure are prioritized according to their spread of values.

6. According to Andishe Sedighian's [Sed96] comparative analysis of B-PR-Q trees, B-PR-KD trees, BA-kd trees, Gridfile, Multi-paging, R*-trees, and her experiments on Gridfile, BA-kd trees, and R-trees, the BA-kd tree is the most suitable structure for organizing PCA-transformed data. Her experiment was conduced on a set of 400 gray-scale face images whose dimensionality is reduced by the eigen image approach which also utilizes the principal components analysis technique.

The only restriction of the BA-kd tree is that it is a static indexing structure as it requires the data to be known a priori. This does not affect our application very much since we are dealing with archival video data which do not change very often.

Experiment results for bucket adaptive k-d tree

In this section we will test the performance of bucket adaptive k-d tree on the dimension reduced color histograms. The implementation of the bucket adaptive k-d tree used in our system is borrowed from Andishe Sedighian at the University of British Columbia. In her implementation, the fixed radius near neighbor search is approximated using range search with the range along each dimension calculated from the given radius.

Given that I/O time is the dominant factor in the overall search time, we only analyzed the I/O behaviour of the bucket adaptive k-d tree in our experiment. The I/O performance on a certain set of data is evaluated in terms of average number of disk accesses per query. The tests are run on three data sets which contain 500, 4000 and 7000 color histograms repectively. For each data set, we use both Sawhney's approach (labeled "SVD-") and our approach (labeled "SVD+PCA-"), which are discussed in the previous section, to transform the original color histogram to new data spaces in which the distance measure is Euclidean and the energy is more concentrated. In the new spaces, we select the first 1, 2, up to 10 dimensions as the coarse representation of the original color histogram, and use bucket adaptive kd-tree to index the resulted coarse representation. We performed fixed radius search on the created k-d trees. For each test data set, some 100 to 200 queries are generated with query points selected from data set and radius set to 0.03.

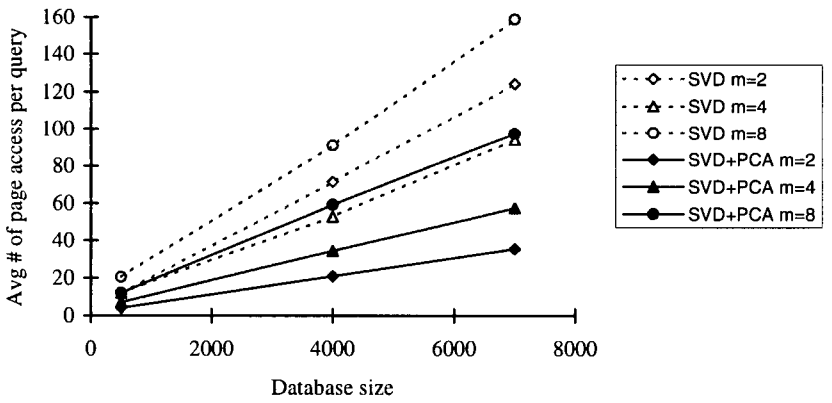

Figure 4.5: Average number of page accesses per query vs. dimensions

The experimental results are shown in figures 4.5 and 4.6. The y axis in

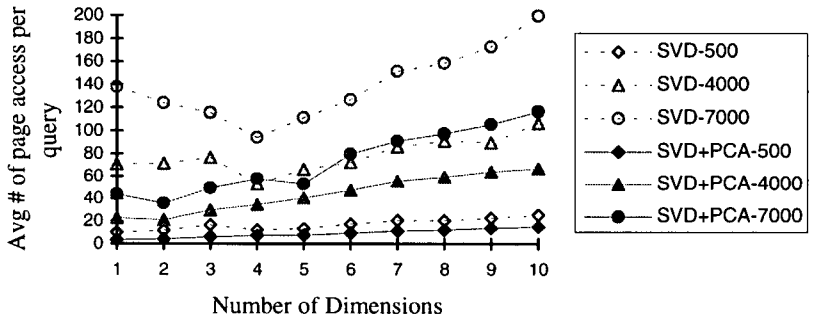

Figure 4.6: Average number of page accesses per query vs. database size

both figures is the avarage number of disk page accesses performed per query, and the x axis is the number of dimensions of the coarse representation in figure 4.5, and database size in figure 4.6. From figure 4.5, we can see that our approach consistantly outperformed Sawhney's approach in terms of savings on I/O time. Figure 4.6 shows how the two methods scale with database size. From the fact that the I/O saving our method gains over Sawhney's approach increases when the dataset size grows, our method is especially suitable for the large size database.

4.4 Conclusion

In this chapter, we have presented a PAM based R-frame selection algorithm which can generate key frames that are more representative. We also showed that PCA can be used to reduce the dimensionality of image features such as color histograms whose desired distance measure is not Euclidean. SVD is used to convert the quadric form distance measure of color histograms to Euclidean distance before PCA can be performed. We also applied bucket adaptive kd-tree to index the dimension reduced

image features.

Chapter 5

Video Retrieval Based on Semantic Annotation

In this chapter, we first present how to use predicates to annotate an object or event that appears in a video clip. Then the knowledge base, which is introduced to reduce the redundancy in the annotation and the amount of human work required, is discussed. Finally, we discuss how to retrieval a video sequence based on video annotations and the knowledge base.

5.1 Video Annotation Using Predicates

As we have discussed in the first chapter, we use predicates to the annotate semantic content of video because of its richness and expressiveness. In the following section we will present how to annotate an object, event and their relations using predicates.

5.1.1 Objects

We use object-attribute-value representation to describe a single attribute of an object. The notation `oav(Obj, Att, Val)` means object `Obj` has value `Val` for attribute `Att`. `Obj` can either be an object name or id. The annotation for a certain object will be a set of `oav()` predicates with the same `Obj`.

The reason we use a set of `oav()` predicates instead of one predicate to describe each object in the video is that the contents of the video are diverse and unpredictable. It is difficult to find a fixed predicate template which anticipates all possible attributes for objects. More importantly, this object-attribute-value representation is modular and flexible. It is easy for an annotator or user to see which values go with which attributes. It is also very easy to add new attributes and to ignore information that is unknown or in which the annotator is not interested.

The predicate `oav()` can also be used to represent the attribute of a certain concept or class in the knowledge base. The notation can only be used to express attribute values that are shared by all the instances of that concept or class. For example, the predicate `oav(Cls, Att, Val)` can be used to express that all the instances of class `Cls` have value `Val` for attribute `Att`.

5.1.2 Events

We can define a similar predicate `eav(Evt, Att, Val)` to represent events. Each event will be assigned with a unique name or id `Evt`. Following are some common attributes of events:

`eav(Evt, actiontype, Val)`, indicates the type of the action;

`eav(Evt, actor, Val)`, specifies the initiator of the action;

`eav(Evt, recipient, Val)`, specifies the recipient of the action;

`eav(Evt, object, Val)`, indicates the object involved in the action;

`eav(Evt, location, Val)`, describes the location where the event happened;

`eav(eid, time, Val)`, describes the time when the event happened.

For example, the statement "John gave Marry a book yesterday in the library" can be expressed using the following `eav` predicates:

```
eav(e01, actiontype, give)
eav(e01, actor, john)
eav(e01, recipient, marry)
eav(e01, object, book)
eav(e01, location, library)
eav(e01, time, yesterday)
```

5.1.3 Relations between Objects and Events

Relations between objects and relations between events can be described by predicate `rel(Id1, Id2, RelType, RelVal)`, where RelType, RelVal are the type and value of a relation respectively, and `Id1`, `Id2` can be either event or object ids depending on the type of the relation. For example, `rel(obj1, obj2, postion, left)` can be used to express the description of "obj1 appears on the left of obj2".

5.1.4 Mapping Annotations to Video Sequence

Predicate `sequence(StartId, EndId, OE)` is used to map a video sequence to the annotations. In the predicate, `StartId` and `EndId` are start and end shot ids, and `OE`

can be an object, an event, or an identifier associated with an object or event. The usage of predicate `sequence()` can be extended to map video sequences to image properties. This feature will be discussed in section 5.2.2.

But the granularity of the mapping is at the object level which may not be enough for some cases. For example, a video clip may use separate shots to present several aspects of an object. A user may be interested only in one of the aspects, but the system will return the shots related to every aspect of that object. The problem can be alleviated by assigning multiple ids to an object, each of which corresponds to certain attributes.

5.2 Knowledge Base

People may view the same video segment differently with different levels of abstraction or from different angles. The annotator and the query users are very likely to use different ways or vocabularies to express the same idea. From this point of view, video annotation can never capture all the details present in the video segments. However, a good video annotation system should consider as many interpretations of video as possible to increase the possibility of matching the query to the annotation. On the other hand, if we include all the possible annotations for every video segment to accommodate a wide range of user query, there will be redundancies in the annotations. For example, in one episode of "American Castle", Jay Goud appears in the video repeatedly. It will be very tedious, if we include the fact that "Jay Goud is the owner of castle Lynhurst, and he was a very rich businessman ..." in the annotations for each segment where Jay Goud appears. By introducing a knowledge base to store the facts that are reusable and inference rules which can derive some annotations from existing ones, we can avoid the redundancy and at

the same time provide a high level of query flexibility. In fact, this can also save a lot of annotator's work since the derivation of new annotations from existing ones based on knowledge base can be done automatically. In our annotation system, we put the facts and rules shared by a group of videos in a domain specific knowledge base, and put the knowledges shared by all the videos in a common knowledge base.

5.2.1 Taxonomy and Synonym Based Knowledge

Synonyms and taxonomy hierarchies are the two major types of knowledge that describe relations among concepts. With this knowledge many of the different views can be linked together. The video annotation system can be much more flexible and powerful by supporting synonyms and taxonomies.

Taxonomy Hierarchy and Taxonomy Based Rules

Taxonomy relations between objects and concepts are represented by predicate `isa(A, B)` which means A is a subclass of B or A is an instance of class B.

The individual inherits all the attribute-value of the class that it belongs to, and the subclass inherits all the attribute-value that its superclass has. This is because according to our definition of predicate `oav()`, the attribute of a class has the same value for all the instances of that class.

Attribute values of objects and events can be generalized. One can substitute a specific attribute value with a more general concept. For example, `Marry has a textbook` logically implies Marry has a book. Here the concept book subsumes the concept textbook. If a predicate says an object has a certain attribute value `AttVal`, then the predicate is also true for the subsumer of that attribute value (i.e., concepts that subsumes that attribute value). General attribute values are represented by

super classes in the taxonomy hierarchy, and specific attribute values are represented by subclasses in the taxonomy hierarchy.

In order to perform taxonomy related reasoning, we need the taxonomy hierarchy and the related rules. In our annotation system, the taxonomy hierarchy is represented using a set of `isa()` predicates. The rules for handling property inheritance and concept generalization, which can be used in conjunction with any particular taxonomy hierarchies, are shown as following:

$$isa(A, C) \longleftarrow isa(A, B) \wedge isa(B, C) \tag{5.1}$$

$$oav(A, Att, Val) \longleftarrow isa(A, B) \wedge oav(B, Att, Val) \tag{5.2}$$

$$oav(Obj, Att, C) \longleftarrow isa(B, C) \wedge oav(Obj, Att, B) \tag{5.3}$$

$$eav(E, Att, Y) \longleftarrow isa(X, Y) \wedge eav(E, Att, X) \tag{5.4}$$

Rule 5.1 defines the transitivity of the `isa` relation. Rule 5.2 allows objects or subclasses to inherit all the attribute-values of the class they belong to. Note that the inheritance rule can not apply to events, since there is no taxonomy definition among events. Rules 5.3 and 5.4 mean the value of an attribute can be generalized, i.e., specific values can be replaced by more general values.

Synonym Based Rules

Similarity between two concepts can be expressed by predicate `issimilar(Concept1, Concept2)`. Following are synonym based rules:

$$issimilar(A, B) \longleftarrow issimilar(B, A) \tag{5.5}$$

$$issimilar(A, B) \longleftarrow issimilar(A, C) \wedge issimilar(C, B) \tag{5.6}$$

$$oav(Obj, Att, A) \longleftarrow issimilar(A, B) \wedge oav(Obj, Att, B) \quad (5.7)$$

$$oav(Obj, Att1, Val) \longleftarrow issimilar(Att1, Att2) \wedge oav(Obj, Att2, Val) \quad (5.8)$$

$$eav(E, Att, A) \longleftarrow issimilar(A, B) \wedge eav(E, Att, B) \quad (5.9)$$

$$eav(E, Att1, Val) \longleftarrow issimilar(Att1, Att2) \wedge eav(E, Att2, Val) \quad (5.10)$$

Rule 5.5 and 5.6 define the symmetry and transitivity of the `issimilar` relation. Rule 5.7 and 5.9 allow an attribute value to be replaced by a similar concept. Rule 5.8 and 5.10 means the name of an attribute can also be replaced by a similar concept.

5.2.2 Image Property Related Rules

We also need rules to establish relations from the syntactic properties of a frame, such as color histogram or color layout, to its semantic properties, such as objects in the frame. This might be the ultimate solution towards automatic video semantic annotation, since image properties can be generated automatically by machine. In reality, we know that there will never be such a rule which can guarantee the correctness of the result. For example, if the dominant color of a frame is blue, then it can be the sky, or the ocean, or even a picture with blue background. So these image property related rules should always have a probability range to indicate the level of uncertainty.

As we discussed in Chapter 1, it is more appropriate to use numerical feature vectors to represent image features rather than to use text description (e.g. predicates). But for simplicity, we only illustrate image property related rules by using predicates to describe image features. In this thesis, we will not investigate the issue of how to generate the predicates from image feature vectors.

We use predicate `iav(Iid, Att, Val)` to represent an image property, where

`Iid` is used to uniquely identify the property, and `Att` is the attribute name which can be any image related properties, such as dominant color or dominant texture. Predicate `sequence(Start, End, Iid)` can be used to map a video sequence to an image property identified by the `Iid`.

Predicate `iav()` can only express properties of the whole image. If we want to express a property of only a part of the image, we need a new predicate. Predicate `pav(Iid, PartitionMethod, PartitionId, Att, Val)` can be used to describe color or texture layout. Here `PartitionMethod` should be a predefined identifier indicating how the frame is partitioned. For example, if we use P03 to define a partition that divides a frame into 3 horizontal parts (top, middle, and bottom), and CL to represent color layout, then a frame with color layout as top blue, middle white, and bottom green can be expressed as follows:

```
pav(Iid, P03, 1, CL, blue)

pav(Iid, P03, 2, CL, white)

pav(Iid, P03, 3, CL, green)
```

For video series "American Castle", through some sampling procedures, we can draw a conclusion that if the color layout of a frame is top blue, middle white, and bottom green then it has a 80% probability that the picture is related to the outer look of a castle. We can express the above rule as follows:

$$[.8, .8]sequence(X, Y, \text{'}outlook\text{'}) \longleftarrow sequence(X, Y, Iid) \wedge pav(Iid, P03, 1, CL, blue)$$
$$\wedge pav(Iid, P03, 2, CL, white) \wedge pav(Iid, P03, 3, CL, green)$$

5.3 Video Retrieval based on Video Annotation

5.3.1 Overview

Video retrieval based on video annotation starts with the query input expressed by a set of predicates. The query should always contain the predicate sequence(X, Y, OE/ID), where X and Y are the start and end shot ids respectively. For example, if a user wants to retrieve all the shots in which someone is talking, then he can submit the query "$sequence(X, Y, ID) \wedge eav(ID, actiontype, talk)$". The query is then submitted to the query processing engine, which will find the appropriate values for all the variables in the query based on the annotations of the video and the knowledge base. Multiple answers may be found, each of which is associated with a probability range indicating its relevence to the query. Then the values of X and Y are used to retrieve the actual video segments.

The query processing procedure for video annotation can be implemented in one of the following two ways:

1. Annotation derivation before query

   The base annotation needs to be pre-compiled using the rules in the knowledge base. All the possible annotations are generated by applying the rules to the existing annotations until the result stops growing. Now we have a complete version of the annotation. Then a simple matching algorithm can be used to find answers for the user queries. The advantage of this method is that the query processing will be very fast. The disadvantage is the problem of annotation explosion. The complete version of the annotation will be too large. Besides, whenever some new facts or new rules are added to the system, the annotation needs to be recompiled.

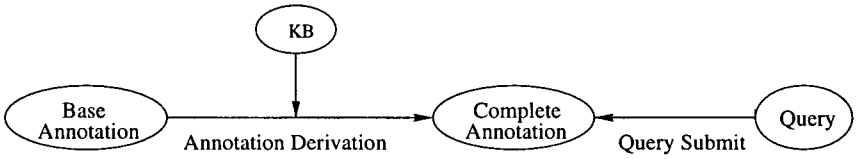

Figure 5.1: Annotation Derivation

2. Query Relaxation

   Here the rules are directly applied to the query. Usually the user query is translated into a set of new queries. The new queries will then be asked on the base annotations to find answers. The advantage of this approach is that there is no need to generate a complete version of the annotation, but the disadvantages is that the relaxation must be done in run time so the query processing will be slower. We choose this strategy because it is easy to implement and it can avoid annotation explosion.

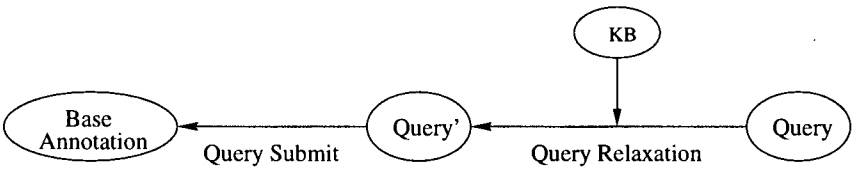

Figure 5.2: Query Relaxation

5.3.2 Query Processing for Video Annotation

The knowledge base in our annotation system contains the probability rules, which will be consulted when no definite answer to the query can be deduced. Special care must be taken in the query processing procedure when probability rules are involved. For the query processing of our annotation system, we adopt some of the procedures

proposed in [Ng94] which can support inductive reasoning based on probability rules and ranking of the answers when more than one are found. We simplified ranking of the answers by sorting the answers in decreasing probability ranges of matched clauses instead of calculating and ranking the maximally preferred classes.

In the paper [Ng94], a database or program containing empirical probability rules is usually called an empirical program or an empirical deductive database. Query processing for empirical programs consists of compilation of the empirical program and query answering. Following we briefly describe some related definitions and algorithms applied in our annotation system. Detailed discussion can be found in [Ng94].

Definition

Definition 5.1 gives the formal definition of an empirical program or empirical deductive database.

Definition 5.1 An **empirical program**, or **empirical deductive database**, $P = \langle C, E \rangle$ consists of a **context** C and a finite set E of **empirical clauses**, where

- a **context** C is a finite set of clauses of the form $L_0 \leftarrow L_1 \wedge ... \wedge L_n$ where
  - for all $0 \leq i \leq n$, L_i is a literal; and
  - any variable appearing in L_0 must appear in any one of $L_1, ..., L_n$.
- an **empirical clause** is of the form $[c_1, c_2]L_0 \leftarrow L_1 \wedge ... \wedge L_n$ where
  - c_1, c_2 are real numbers in [0, 1] such that $c_1 \leq c_2$, $[c_1, c_2] \neq [0, 0]$, and $[c_1, c_2] \neq [1, 1]$;
  - for all $0 \leq i \leq n$, L_i is a literal; and
  - any variable appearing in L_0 must appear in any one of $L_1, ..., L_n$.

Example 5.1 Following is an example of an emprical program. It annotates that the dominant color in the upper half of frames in shot 10 is blue, and the dominant color in the lower half of frames in shot 12 is green. It contains two inference rules saying that shots with meadow or with sky in them are about outdoor views. It also contains two emprical rules saying that 50% of the shots with blue top contain skies, and 30% of the shots with green bottom contain meadows.

C:

$$
\begin{aligned}
iav(iid1, topcolor, blue) &\leftarrow \\
iav(iid2, bottomcolor, green) &\leftarrow \\
sequence(10, 10, iid1) &\leftarrow \\
sequence(12, 12, iid2) &\leftarrow \\
sequence(X, Y, outdoorview) &\leftarrow sequence(X, Y, sky) \\
sequence(X, Y, outdoorview) &\leftarrow sequence(X, Y, meadow)
\end{aligned}
$$

E:

$$
\begin{aligned}
[.5, .5]sequence(X, Y, sky) &\leftarrow sequence(X, Y, Id) \wedge iav(Id, topcolor, blue) \\
[.3, .3]sequence(X, Y, meadow) &\leftarrow sequence(X, Y, Id) \wedge iav(Id, bottomcolor, green)
\end{aligned}
$$

Compilation of Empirical Programs

An empirical program can be compiled to simplify the query processing. Algorithm 5.1 describes how to use empirical clauses in E and clauses in context C to generate other empirical clauses. This algorithm should be carried out at compile-time since the generation process is query-independent.

Algorithm 5.1 Let $P = \langle C, E \rangle$ be an empirical program.

1. Set T_0 to E and i to 1.

2. Construct the set $S_1 = \{\ [1-c_2, 1-c_1]\ \neg L_0 \leftarrow L_1 \wedge ... \wedge L_n\ \mid\ [c_1, c_2]\ L_0 \leftarrow L_1 \wedge ... \wedge L_n$ is a clause in $T_{i-1}\ \}$.

3. Construct the set $S_2 = \{\ [0,0]\ L_1 \leftarrow L_0\ \mid\ [0,0]\ L_0 \leftarrow L_1$ is a clause in $T_{i-1}\ \}$.

4. Construct the set $S_3 = \{\ [c_1, 1]\ L' \leftarrow L_1 \wedge ... \wedge L_n\ \mid\ [c_1, c_2]\ L_0 \leftarrow L_1 \wedge ... \wedge L_n$ is a clause in T_{i-1} and $L' \leftarrow L_0$ is a logical consequence of $C\ \}$.

5. Set $T_i = T_{i-1} \cup S_1 \cup S_2 \cup S_3$. If T_i is the same as T_{i-1}, then set $comp(P) = \langle C, T_i \rangle$ and halt. Otherwise ($T_i \neq T_{i-1}$), increment i and go to Step 2.

Example 5.2 Apply Algorithm 5.1 to compile the emprical program given in example 5.1. In the first iteration, S_1 consists of $[.5, .5] \neg sequence(X, Y, sky) \leftarrow sequence(X, Y, Id) \wedge iav(Id, topcolor, blue)$ and $[.7, .7] \neg sequence(X, Y, meadow) \leftarrow sequence(X, Y, Id) \wedge iav(Id, bottomcolor, blue)$. S_2 is empty. S_3 consists of $[.5, 1] sequence(X, Y, outdoorview) \leftarrow sequence(X, Y, Id) \wedge iav(Id, topcolor, blue)$ and $[.3, 1] sequence(X, Y, outdoorview) \leftarrow sequence(X, Y, Id) \wedge iav(Id, bottomcolor, green)$. In the second iteration, nothing is generated, and the compilation ends.

Query Answering

Finally, we describe the algorithm which can handle non-ground queries as well as ground queries in an empirical program. Basically, this algorithm poses the query against the Context C. If the context can deduce the definite answer, then the process stopped. Otherwise, the process tries to induce the probability by consulting the empirical clauses in E. This procedure first gets all of the applicable empirical clauses, then ranks the matched clauses according to the corresponding probablity

range, and finally returns the unifiers and corresponding probability ranges of the top N matched clauses in order of decreasing rank. The details of the algorithm are as follows:

Algorithm 5.2 Let L be a query, and $comp(P) = \langle C, E \rangle$ be the compiled version of an empirical program.

1. For all most general unifiers θ, if $L\theta$ (or $\neg L\theta$) is a logical consequence of C, return θ and [1,1] (or [0,0] respectively), and halt.
2. Construct the set $S = \{\ Cl \mid Cl \equiv [c_1, c_2]\ L' \leftarrow Body$ is an empirical clause in E, and there exists a unifier θ that unifies L and L' and that $Body\theta$ is a direct logical consequence of $C\ \}$. If S is empty, halt.
3. Otherwise, rank the clauses in E in descending order based on the value of the lower bound of the associated probability range. The unifiers with their probability ranges of the top N ranking clauses are returned.

Example 5.3 Consider posting a query "retrieve all the sequences about outdoor view" ($Q = sequence(X, Y, outdoorview)$) to the compiled version of the empirical program given in example 5.1. According to algorithm 5.2, no definite answer is found in step 1. In step 2, S consists of $[.5, 1]sequence(X, Y, outdoorview) \leftarrow sequence(X, Y, Id) \wedge iav(Id, topcolor, blue)$ and $[.3, 1]sequence(X, Y, outdoorview) \leftarrow sequence(X, Y, Id) \wedge iav(Id, bottomcolor, green)$. In step 3, two answers are found: X=10, Y=10 with probability range [.5, 1], and X=12, Y=12 with probability range [.3,1].

5.4 Implementation

Our implementation of processing for query on video annotation is a mixture of C and Prolog. The base annotation, the knowledge base (complied by using algorithm 5.1), and algorithm 5.2 are written in SICStus Prolog 2.1 #9 which was developed in the Swedish Institute of Computer Science.

The interface between the Prolog engine and the CGI program are written in C. The two parts are linked together using the SICStus Prolog runtime library. When the user submits a query, the prolog engine is invoked via the CGI interface. The query string is passed to the prolog engine from CGI interface, and the corresponding start, end shot ids and probability ranges of the top N matched video sequences will be returned. The following steps are used to interact with the Prolog engine from C:

1. Initiate the prolog engine with SP_initialize().

2. Load Prolog code compiled to Prolog object files into the runtime system with SP_load(). Those files include base annotations, the complied version of the knowledge base, and the Prolog code for query answering.

3. Form query in Prolog format with SP_predicate(). This also involves converting C data to Prolog term. The C functions which can achieve such conversion include SP_put_variable(), SP_put_string(), etc.

4. Issue the query to Prolog engine and retrieve multiple solutions:

   - Issue the query with SP_open_query().
   - Call SP_next_solution() to find a solution and call it again to find more solutions.

- Termination the query with SP_close_query().

5. Getting the results in C data from the Prolog terms: the query results obtained from SP_next_solution() are in Prolog terms, so functions such as SP_get_integer() must be called to convert Prolog terms to C data.

Chapter 6

Implementation Detail of Web-based Query Interface

We have discussed the three central parts of our video retrieval system in the previous chapters. Now we will give some implementation details on the Web-based query interface of the system.

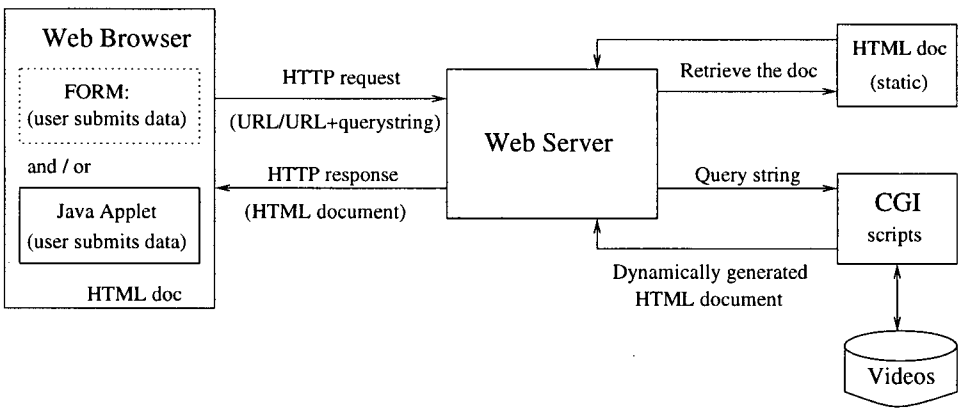

Figure 6.1: Information flow between the client, server, and CGI application

The web-based user interface of our video retrieval system is designed and

implemented using HTML/JAVA/CGI. One part of the interfaces is web pages containing HTML forms or Java applets that let users enter query parameters to search the database. The rest of the interface lies in the CGI scripts, which take the user input, search the database, and return the results in the form of dynamically generated web pages. The information flow between the web brower, the web server and the CGI scripts are show in the Figure 6.1. We will discuss how this is achieved in our implementation in the following sections.

6.1 Entering query on the Web page

1. Use HTML FORM

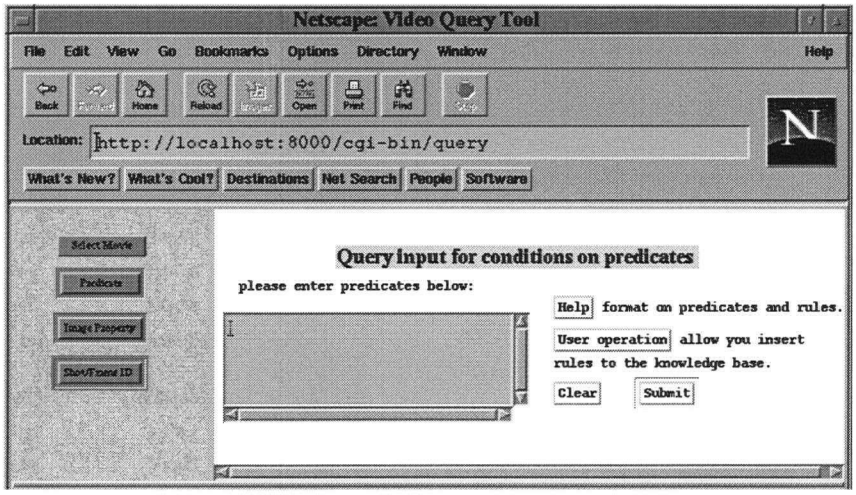

Figure 6.2: User input: HTML Form

A web page which contains the <FORM> tag allows customizable input forms, including check-boxes, text-fields, choice list, radio buttons, regular buttons, and etc. Figure 6.2 shows an example of form used in our interface which allows users to input the queries on predicate based video annotation.

2. Use Java applet

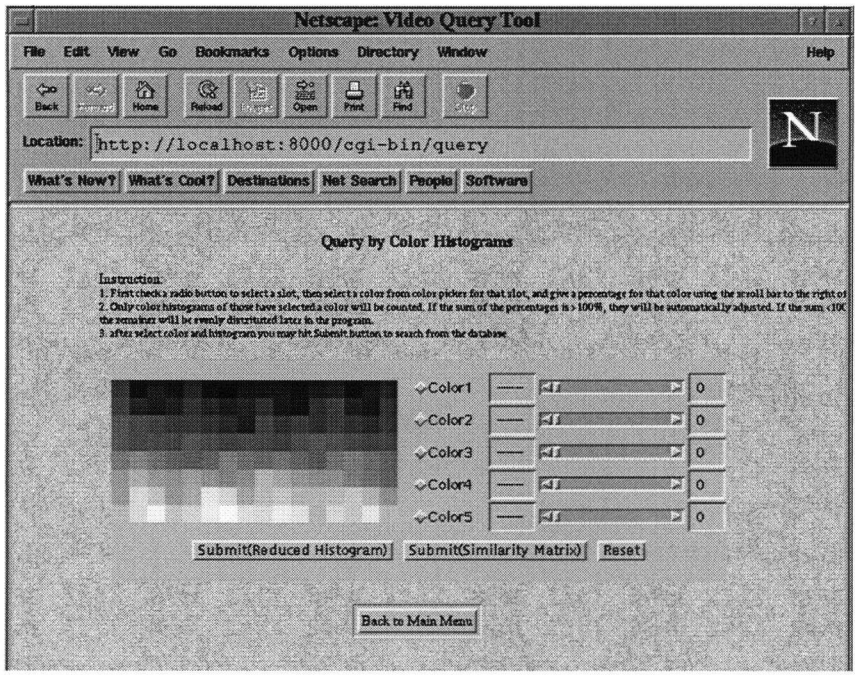

Figure 6.3: User input: Java applet

Java also provides buttons, choice lists, text-fields, etc. The advantage of using a Java applet is that it enables the program to run at the client machine, as opposed to form (CGI), which runs programs on the web server. So Java applet is suited to the query input that needs a lot of user interaction which does not involve accessing database, e.g. detailed manipulation of an image. For our user interface, we use a Java applet to create an interactive color picker (Figure 6.3). It respondes user selection of color and percentage by showing the result at corresponding places, and also calculates the total percentage for the colors picked so far and makes sure this value will be less than or equal to 100%. All this is executed at client side and does not need to talk to the web

server. Only after the user hits the "submit" button, is the entire query sent to the web server and search performed.

The interactive color picker can also be implemented using HTML FORM/CGI. However in this case, whenever a color is chosen or a percentage is adjusted, the web server will be contacted and the whole window in the browser will be refreshed in order to reflect the changes in user selection. This unnecessarily wastes time, server resources and network bandwidth. Besides, the refreshment of window from time to time is often annoying.

6.2 Sending the Query to the Server

1. From an HTML file using FORM

   We use POST method in the <FORM> tag. When the user clicks on the "submit" button, the contents of the form are encoded in the form of "name1=val1&name2=val2&...&nameN=valN". This data are sent to the web server in a data block, and then passed to the CGI program from STDIN by the web server. All the above procedures are done automatically by web browser or server according to HTTP or CGI protocols.

2. From Java Applet

   After the user hits the "submit" button, first we need to encode the query input in exactly the same format as HTTP does for the html FORM. Then we use a `URLConnection` object to generate an HTTP Post request to the CGI program on the machine from which the applet was loaded, with the results being read inside the applet. Following code shows how a Java applet communicates with the CGI program on the web server:

```
try{
    //Connected to server/CGI script
    URL queryURL = new URL("http://hostname:8000/cgi-bin/CGIscript");
    URLConnection connection = queryURL.openConnection();

    // Send user query to server/CGI script
    connection.setDoOutput(true);
    PrintStream os = new PrintStream(connection.getOutputStream());
    os.println(querydata);
    os.close();

    //Read result from server/CGI script
    DataInputStream is=new DataInputStream(connection.getInputStream());
    String line;
    while ((line = is.readLine()) != null) {
        rdata += line ;
    }

    // Showing result on the web brower
    ShowHTML.showPage(this, rdata);
    is.close();
} catch (MalformedURLException me) {
  ...
} catch  (IOException ioe) {
  ...
}
```

6.3 Processing the Query on the Server

After the web server prepares the enviroment, it launches the CGI. Here, the CGI script does not have to know whether the request is initiated by HTML FORM or Java applet, since we chose the request method as POST for both cases and the

input data are encoded in the same fashion.

The basic structure of the CGI scripts we wrote contains three parts: initializing query input, processing query, and ouputing the query result to STDOUT. What the script does in the intialization phase is to read the input stream from STDIN and parse them into the format of query input. Following is the detailed procedure:

```
1) make sure environment variable REQUEST\_METHOD equals to "POST";
2) retrieve value of environment variable CONTENT\_LENGTH;
3) if CONTENT\_LENGTH is greater than zero, read CONTENT\_LENGTH
   bytes from STDIN;
4) parse STDIN data into seperate variables;
5) decode parsed variables;
```

After initializing its environment, the CGI script can process the query by accessing the database. The output of the CGI script, which is a header and an HTML document that contains the query result, looks like:

```
content-type: text/html

<HTML>
<HEAD> ... </HEAD>
<BODY> query result is put in here </BODY>
</HTML>
```

6.4 Showing the Query Result on the Web Browser

For a request initiated from the html file using form, the web browser receives and displays the output from the CGI script.

In the Java applet case, the output of the CGI script was read inside the applet, and there is no portable way in the Java language to tell the browser to display the results. However, if the browser is Netscape (only netscape supports Java script), we can achieve this by exploiting Java script. First, the Java applet encodes the content of the query result page into a special URL starting with "Javascript:". Then it opens this URL in the netscape via showDocument. Netscape will invoke the javascript interpreter to display the entire result page. Following are Java codes to achieve this functionality:

```
public class ShowHTML {
  // Open a URL via showDocument
  public static void showPage(Applet app, String html) {
    URL page = makeJavascriptURL(html);
    app.getAppletContext().showDocument(page);
  }

  //Encode the query result received in the applet into a URL
  public static URL makeJavascriptURL(String html) {
    try {
      URL page = new URL("javascript:'" + html + "'");
      return(page);
    } catch(MalformedURLException mue) {
      System.out.println("Illegal URL: " + mue);
      return(null);
   }
  }
}
```

Chapter 7

Conclusion and Future Work

7.1 Summary

In this thesis, we designed and implemented a content-based video retrieval system. We addressed three important issues in the design of an effective and efficient content-based video indexing and retrieval system: 1) video parsing which segments the video into smaller units for indexing and retrieval, 2) video content abstraction which extracts the most representative features of both the visual and semantic content of video, and 3) indexing structures for the video abstracts with high dimensionality.

In our system, the basic unit for video indexing and retrieval is the shot. Therefore, all the video materials in our system are first partitioned into shots. Several shot boundary detection techniques have been implemented and evaluated. The experiments show that in most cases abrupt shot changes can be detected accurately, but it is still difficult to distinguish shots with intensive object and camera motion from gradual transition. We selected the twin threshold approach with motion analysis proposed by Dr. Hongjian Zhang, which shows better performance

under combined criterion of efficiency and accuracy, and integrated it in our prototype system.

In visual content abstraction, each shot is represented by one or more R-frames selected using a PAM based algorithm. Visual features are then extracted from selected R-frames and the dimensionality of the features are reduced. We demonstrated that PCA can be used to reduce the dimensionality of color features on which the distance measure is not Euclidean, if the color features are transformed properly using SVD. Our experiments show that the combined SVD/PCA transformation can reduce the dimensionality of color histogram image feature from 256 to less than 10, while keeping over 90% of the information.

In semantic content abstraction, we use the predicate-based video annotation. The system also utilizes knowledge bases to derive complete video annotations from the base annotation, and thus reduces the human annotator's work. The annotation system also supports probability inference rules that can derive semantic annotations directly from visual features. We provide basic predicate templates for the annotation of objects and events in the video, and provide rules for synonym and taxonomy based inference. We also adopt the algorithm proposed in [Ng94] to deal with query processing in our video annotation system which contains probability rules.

On the indexing structure for visual features (color histogram), our work includes the integration of BA_kd tree into our prototype system. We confirmed that the BA_kd tree can be used with our combined SVD/PCA approach. The test results showed that BA_kdtree with our combined approach can achieve between 30 and 60 percent reduction in terms of average number of page access when compared to BA_kdtree with SVD.

A web based user interface has also been integrated into the system. HTML forms and Java applets are used to collect user query input, and CGI scripts are used to process the query.

7.2 Future Work

Further research work can be directed to the following areas:

- In this thesis, we use shot as the basic indexing unit. Introducing some other higher level indexing units can bring more flexibility to the system. So automatic organizing the shots into higher level units need to be further investigated.

- We selected a simple and commonly used visual feature, color feature, in our system. Support for other visual features such as texture, shape, motion, etc, are yet to be implemented.

- We have investigated in how to integrate probability rules in the video annotation system. However, more systematic approaches for generating probability rules need to be developed.

- More user friendly utilities that can help the user to generate predicates as query input for video annotation are also needed.

Bibliography

[ATe92] A. Akutsu, Y. Tonomura, and etc. Video indexing using motion vectors. In *Visual Communicatins and Image Processing*, pages 1522–1530. SPIE, 1992.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, Vol. 18(No. 9):509–517, 1975.

[BKSS90] Norbert Beckmann, Han-Peter Kriegel, Ralf Schnedier, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangle. In *Proceedigs of ACM SIGMOD Conference*, pages 322–331, 1990.

[BR96] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and Retrieval for Still Image and Video Databases IV*, pages 170–179. SPIE, February 1996.

[BYY96] R. M. Bolle, B. Yeo, and M. M. Yeung. Video query: Beyond the keywords. Technical Report RC20586, IBM, 1996.

[CB95] Mourad Cherfaoui and Christian Bertin. Temporal segmentation of videos: a new approach. In *Digital Video Compression: Algorithms and Technologies*, pages 38–47. SPIE, 1995.

[DLM+96] E. Deardorff, T.D.C. Little, J.D. Marshall, D. Venkatesh, and R. Walzer. Video scene decomposition with the motion picture parser. In *Digital Video Compression on Personal Computers: Algorithms and Technologies*, pages 44–55. SPIE, February 1996.

[DSS96] Asit Dan, Dinkar Sitaram, and Junehwa Song. Brahma: Browsing and retrieval architecture for hierachical multimedia annotation. In *Second*

International Workshop on Multimedia Information Systems, pages 25–29, Sep 1996.

[FEe94] C. Faloutsos, W. Equitz, and etc. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, Vol.3(No.3/4):231–262, 1994.

[FSZ95] B. Furht, S. W. Smoliar, and H. Zhang. *Video and Image Processing in Multimedia Systems.* Kluwer Academic Publisher, 1995.

[Fua93] Pascal Fua. A parrallel stereo algorithm that produces dense path maps and preserves image features. *Machine Vision and Applications*, 1993.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference*, pages 47–57, 1984.

[HJW94] A. Hampapur, R. Jain, and T Weymouth. Digital video segmentation. In *Proc. ACM Multimedia 94*, pages 357–364, San Francisco, CA, October 1994.

[Iok89] Mikihiro Ioka. A method of defining the similarity of images on the basis of color information. Technical Report RT0030, IBM Tokyo Research Lab, 1989.

[Jol86] I. T. Jolliffe. *Principal Components Analysis.* Springer-Verlage, 1986.

[Kat94] Ephraim Katz. *The film encyclopedia.* HarperCollins Publishers, 1994.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley and Sons, 1990.

[lYL95] Boon lock Yeo and Bede Liu. A unified approach to temporal segmentation of motion jpeg and mpeg compressed video. In *Proc of the Second Internation Conference on Multimedia Computing and Systems*, May 1995.

[LZ95] Hain-Ching H. Liu and Gregory L. Zick. Scene decomposition of mpeg compressed video. In *Digital Video Compression: Algorithms and Technologies*, pages 26–37, 1995.

[MJC95] J. Meng, Y. Juan, and S. Chang. Scene change detection in a mpeg compressed video sequence. In *Digital Video Compression: Algorithms and Technologies*, pages 14–25. SPIE, Feb 1995.

[NBe93] W. Niblack, R. Barber, and etc. The qbic project: Querying images by content using color, textur e, and shape. In *Proceedings of SPIE: Storage and Retrieval for Image and Video Database*, pages 173–187, Feb 1993.

[Ng94] Raymond Ng. Semantic, consistancy and query processing of empirical deductive databases. Technical Report TR94-11, Department of Computer Science, University of British Columbia, 1994.

[NH94] Raymond Ng and Jiawei Han. Efficient and effective clustering methods for spatial data min ing. In *Proceedings of the 20th International Conference on Very Large D atabases*, pages 144–155, 1994.

[NT92] A Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In E. Knuth and L. Wegner, editors, *Visual Database Systems II*, pages 113–127. Elsevier Science Publishers, 1992.

[Sam90] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* ADDISON-WESLEY, 1990.

[SB90] Michael J. Swain and Dana Ha. Ballard. Indexing via color histograms. In *Proceedings: Computer Vision*, pages 390–393, 1990.

[Sed96] Andishe Sedighian. A comparative analysis of multi-dimensional indexing structures for eigenimages. Master's thesis, University of British Columbia, 1996.

[SH93] H. Sawhney and J. Hafner. Efficient color histogram indexing for quadratic form distance function. Technical Report RJ9572, IBM, 1993.

[Sha95] B. Shahraray. Scene change detection and content-based sampling of video sequences. In *Doigital Video Compression: Algorithm and Technologies*, pages 2–13. SPIE, February 1995.

[SP95] I. K. Sethi and N. Patel. Statistical approach to scene change detection. In Wayne Niblack and Ramesh C. Jain, editors, *Storage and Retrieval for Image and Video Databases III*, pages 329–339. SPIE, February 1995.

[SRF87] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th VLDB Conference*, pages 507–518, 1987.

[Tam96] Dominic Tam. An analysis of multi-level filtering for high dimensioanl image data. Master's thesis, University of British Columbia, 1996.

[Ze94] H. J. Zhang and etc. Automatic parsing of news video. In *Proceeding of the International Conference on Multimedia Computing and Systems*, pages 45–54. IEEE, May 1994.

[ZKS93] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, Vol.1:10–28, 1993.

[ZMM93] R Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *Proc. ACM Multimedia 95*, pages 189–200, San Francisco, CA, November 1993.

Appendix A

Video Display and Processing

The format of digital videos used in our test experiment is MPEG-1. Following we describe how we digitize, display and process MPEG-1 videos in our system. Most of those MPEG tools and libraries we used can be obtained from the web site: "ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/mpeg/bmt1r1.tar.gz".

1. Video Digitizing

   In order to generate MPEG video stream, we first used the parallax video grabber to digitize the analog videos (taped from TV) into to JMOVIE format digital videos. Then, we used a tool named Compressor, which is included in the NVR Digital Media Development Kit (V2.0.2), to convert JMOVIE video to MPEG-1 video only stream.

2. Video Displaying

   We use MPEG_PLAY (V2.3) to display video streams. Developed at UC Berkeley, MPEG_PLAY is a software-only MPEG-1 video decoder. It decodes and displays an MPEG-1 video stream, but cannot handle audio streams. The

reason we choose this player is because it is widely used, and more importantly free of charge.

3. Video processing

   In order to perform shot boundary detection and other analysis, we need to first extract individual frames from the video streams. It is difficult to directly use the source code of MPEG_PLAY, since it is specifically geared towards displaying video streams in X window. Fortunately, the MPEG library, which is a collection of C routines to decode MPEG videos, directly provides the function that we need. As a front end to the Berkeley decoding engine, the MPEG library was developed by Greg Ward at Montreal Neurological Institute. Basically, it extracts the useful MPEG-related source code from the X11-specific and display related source of MPEG_PLAY, and provides a simple interface that allows a programmer to extract frames from an MPEG stream. The MPEG Library (version 1.2) used in our system is obtained from web site "ftp.mni.mcgill.ca/pub/mpeg".

   Following are three useful routines MPEG Library provides, and basically all we need to extract frames from video streams:

   - `SetMPEGOption()`: allows you to set a variety of options related to MPEG decoding, including the dithering mode which controls how YC_rC_b values are converted to RGB space.
   - `OpenMPEG()`: prepares an MPEG stream for decoding. It initializes internal data structures for decoding and dithering, and creates a color map.
   - `GetMPEGFrame()`: decodes the next frame from the movie. The decoded

frame is an array of pixels, each of which contains either rgb values or an index value in the color map depending on the different dither option set by `SetMPEGOption()` routine.

4. Dynamically generating MPEG video streams

   The result of query in our video database system will be a set of video sequences which at most times may not be the entire video streams. However, for the MPEG player we chose in our system, random access is not allowed (This is not due to a fundamental weakness with MPEG, but the nature of the decoding engine of MPEG_PLAY). That means if the part of video that a user wants is located at the end of the entire video stream, he probably has to wait a long time for decoder to uncompress the video from the very beginning before he can view the part he is interested in. More seriously, if the database system is distributed, returing the entire stream will waste a lot of network bandwidth.

   In order to solve the problem, for each video program we generate a set of Group of Pictures (GOPs) instead of having one MPEG stream. Whenever a query is performed, we calculate the number and positions of the corresponding GOPs according to start and end frame numbers of matched video sequence, and combine those GOP files together to from a MPEG file dynamically as the final query result. GOP is a roughly independently decodable sequence of frames. It starts with an I-frame which is followed by several B and P frames. The GOP files can be joined together at any time later. Since most compression work has already been done during the time GOPs were generated, the dynamic combine will not take much time.

   The software we use to generate GOP and combine GOPs is MPEG-1 video

encoder (version 1.5), which was written by Kevin L. Gong in the Berkeley Plateau Multimedia Research group. Specifically, we use the following two commands:

- `mpeg_encode -gop num param_file`: using this commend, we can encode a single GOP. Here parameter file should specify how many frames each GOP will contain (for MPEG-1 format video, the number of frames of every GOP is same), the IBP pattern (e.g. IBBBPBBBPBBBP), search window for motion vectors, search algorithm, etc.
- `mpeg_encode -combine_gops paramfile`: this command causes the encoder to combine some GOP files into a single MPEG stream. Here the parameter file need contain the YUV_SIZE value, an ouput file, and a list of input GOP files.

Appendix B

Color Spaces conversion

In our system, we use the 256 colors in the color map generated by the MPEG decoder as color bins. In order to compute the entries a_{ij} (formula 4.1) in the similarity matrix A used in color histogram difference caluculation (formula 2.1), we need to calculate the differences between any two of the 256 color bins we have chosen. Euclidean distance in $CIELUV$ space is used to measure the difference between two color bins, because this distance measure is close to human perception. Since frames extracted MPEG-1 stream contains color values in RGB space, we need to convert the colors into $CIELUV$ space. The conversion from RGB to $CIELUV$ also involves intermediate $CIEXYZ$ space. In the next two sections, we will give the transformation matrix used to convert a color from RGB space to $CIEXYZ$ space, and from $CIEXYZ$ to $CIELUV$ space.

RGB - CIE XYZ conversion

RGB values in a particular set of primaries can be transformed to CIE XYZ by a three-by-three matrix transform. We use the RGB system defined in CCITT ITU-R Recommendation BT 709 with CIE Illuminance D65 white point in our system.

Following is the transformation from 709 RGB space to CIE XYZ space:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The reference white color (X_n, Y_n, Z_n) in CIE XYZ space can be computed by transforming white point (1, 1, 1) in RGB using the above formula.

CIE XYZ - CIE LUV conversion

For each color, its L^*, u^* and v^* values in $CIELUV$ space can be calculated from its X, Y, Z values in $CIEXYZ$ space with (X_n, Y_n, Z_n) as reference white point by using following formula:

$$L^* = \begin{cases} 116 * (Y/Y_n)^{1/3}) - 16 & Y/Y_n > 0.008856 \\ 903.3Y/Y_n & Y/Y_n <= 0.008856 \end{cases}$$

$$u^* = 13 * L^* * (u^{'} - u_n^{'})$$

$$v^* = 13 * L^* * (v^{'} - v_n^{'})$$

where

$$u^{'} = 4 * X/(X + 15 * Y + 3 * Z)$$

$$v^{'} = 9 * Y/(X + 15 * Y + 3 * Z)$$

$$u_n^{'} = 4 * X_n/(X_n + 15 * Y_n + 3 * Z_n)$$

$$v_n^{'} = 9 * Y_n/(X_n + 15 * Y_n + 3 * Z_n)$$