EFFICIENT VIDEO IDENTIFICATION BASED ON LOCALITY SENSITIVE HASHING AND TRIANGLE INEQUALITY

Yang Zixiang

NATIONAL UNIVERSITY OF SINGAPORE

2005

Name:	Yang Zixiang
Degree:	Master of Science
Dept:	Computer Science
Thesis Title:	Efficient Video Identification Based on Locality Sensitive Hashing and Triangle Inequality

Abstract

Searching for duplicated version video clips in large video database, or video identification, requires fast and robust similarity search in high-dimensional space. Locality sensitive hashing, or LSH, is a well-known indexing method for efficient approximate similarity search in such space. In this thesis, we present a highly efficient video identification method for transcoded video content based on locality sensitive hashing and triangle inequality. To store large volume of videos, we design a small feature dataset and index the dataset using improved locality sensitive hashing. In addition, we employ triangle inequality to further enhance the system efficiency. Experimental results demonstrate that once the features of a given 8s query video are extracted, it takes about 0.17s to retrieve it from a 96-hour video database. Furthermore, our system is robust to the changes of the query videos on frame size, frame rate and compression bit-rate.

Keywords: video identification
video search
video hashing
locality sensitive hashing

EFFICIENT VIDEO IDENTIFICATION BASED ON LOCALITY SENSITIVE HASHING AND TRIANGLE INEQUALITY

Yang Zixiang

B. Eng. (Hons), XJTU, P. R. China

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

I sincerely thank my supervisors, Dr. Sun Qibin and Dr. Ooi Wei Tsang, who have guided and supported me throughout my postgraduate years. Their suggestions for improvement and faith in my work have strengthened my confidence. I have benefited tremendously both technically and personally from their guidance and supervision.

I send my sincere regards to all my colleagues who I worked with during my academic years, for their valuable suggestions: Dr Tian Qi, Dr. Heng Wei Jyh, Dr. Gao Sheng, Dr. Zhu Yongwei, Dr. He Dajun, Mr. Zhang Zhishou. In addition, many friends who have contributed in one way or another: Mr. Yuan Junsong, Mr. Yang Xianfeng, Mr. Wang Dehong, Mr. Ye Shuiming, Mr. Zhou Zhicheng, Mr. Li Zhi, to name a few, for their help and encouragement.

Finally, special thanks to my family members who gave me their continual moral support to complete this course.

Publications

Zixiang Yang, Wei Tsang Ooi and Qibin Sun, "Hierarchical, non-uniform locality sensitive hashing and its application to video identification," in *Proceedings of International Conference on Multimedia and Expo*, Jun 2004, Taipei, Taiwan.

Wei Jyh Heng, Yu Chen, Zixiang Yang and Qibin Sun, "Classroom assistant for real-time information retrieval," in *Proceedings of International Conference on Information Technology: Research and Education*, pp.436-439, Aug 2003, Newark, New Jersey, USA.

Contents

List of Figures

List of Tables

Summary

The problem of content-based video identification concerns identifying the duplicated version of a given short query video clip in a large video database based on content similarity. Video identification has many applications, including news report tracking on different channels, video copyright management on the internet, detection and statistical analysis of broadcasted commercials, video database management, etc. Three key steps in building a video database for video identification are (i) video segmentation and feature extraction to represent the video clips; (ii) similarity measuring between the query video and the videos in database; (iii) indexing of the feature vectors to allow efficient search of similar video.

In this thesis, we present a highly efficient video identification system at transcoding level for a large video database by systematically taking “feature extraction”, “feature indexing” and “video database construction” together into consideration. The selected feature is robust to the changes on frame size, frame rate and compression bitrate. Principal components analysis (PCA) and improved locality sensitive hashing (LSH, an index structure in database area) are then used to reduce the dimensions of feature space and generate the index code. Considering that the original LSH is only good for indexing uniformly distributed high-dimensional data points and can be improved for video identification where data points may be clustered. We therefore give two improvements to LSH to distribute the points more evenly. First, by building a hierarchical hash table, we adapt the number of hashed dimensions to the density of the data points. Second, we choose the hashed dimensions carefully in such a way that the points are more evenly hashed, thus making the hash table more uniformly distributed and reducing the miss rate. We further apply triangle inequality on the resulted buckets by LSH to skip some redundant match operations. In terms of system design, to save the storage of the video database's feature dataset, we slide the search window on the query video rather than the videos in database.

Experimental results verify that our improved LSH is much better than original LSH in terms of both efficiency and accuracy when applied on the video feature dataset for similarity search. For video identification, our system is robust to the transcoding level noise, i.e. changes on frame size, frame rate and compression bit-rate. We greatly reduce the search space and redundant match operations by incorporating improved LSH with triangle inequality to improve the efficiency. We further demonstrate the promising system performance by comparing our algorithm with NTT's "active search" algorithm. The use of LSH with triangle inequality and sliding search window on the query video are two main contributions of this research work.

Chapter 1

Introduction

We live in a world of information. Information was first delivered to the general public through broadcasting media such as newspapers, radio, and eventually television. Later, the computer was invented. Computers allow information to be compiled in digital form, and make it possible for people to search for required information. Furthermore, information could be selectively retrieved when required, which is quite useful when querying huge database. Looking at the great success of text search engines, such as Google and Yahoo, researchers started to wonder if the same concept could be applied to videos because recently digital videos become increasingly popular with the development of hardware and video compression standard like MPEG. There are a wide range of applications for content-based video search. For example, you may be interested in a historic event or a scene involving a movie star, but only have few materials about it. With an effective video retrieval system, you can find more detailed video content. For some video producers, they may be interested in how their publications are spread in the world. They can find if there are some illegal copies via a video identification system. A video search system is also useful for video editors. They can search for useful video clips with a simple query instead of spending hours browsing unrelated video content. For video database management, videos with similar content could be clustered to facilitate browsing. In [1], Hong-Jiang Zhang summarized the state-of-the-art technologies, directions, and important applications for research on content-based video retrieval. Some applications are:

- Professional and educational applications
    - Automated authoring of web video content
    - Searching and browsing large video archives
    - Easy access to educational video material
    - Indexing and archiving multimedia presentations
    - Indexing and archiving multimedia collaborative sessions
- Consumer domain applications
    - Video overview and access
    - Video content filtering
    - Enhanced access to broadcast video

While video is widely accepted as a form of broadcasting media, the ability to search through video contents has only recently been investigated. The search for text in documents simply looks for matching words and it achieves great success. Therefore, a straightforward approach to index video database is to represent the visual contents in textual form (e.g. keywords and attributes). These keywords serve as indices to access the associated visual data. This approach has the advantage that visual databases could be accessed using standard query languages (SQL). However, this approach needs too much manual annotation and processing. More seriously, these descriptive data are not reliable because they do not conform to a standard language. So they are inconsistent and might not capture the video content. Thus the retrieval results may not be satisfied since the query is based on the features that have been inadequately represented. Actually, the search of content within video sequence is much more complicated. There are different kinds of inputs and requirements for different video search applications. We can classify the video search systems into “query by keywords” and “query by video clip” based on the inputs, or classify it into video retrieval and video identification based on the results. We will give more details about these different categories in next section.

1.1 Classification for Video Search Systems

1.1.1 “Query by Keywords” and “Query by Video Clip”

We can classify video search systems into “query by keywords” and “query by video clip” based on their inputs. For example, we give the video search system several keywords to find a category of video clips, i.e. query by keywords, and these returned video clips are ranked by their similarity to these query keywords. Here, the keywords not only refer to text, but also include some other properties that describe the video content, such as shape, color, etc. “Query by keywords” is a semantical level video retrieval application [2, 3, 4] which works just like the text search engine. The advantage is that it is easy for the users because they only need to give the system some keywords or some descriptions to search what they want. However, since text can not well represent the content of video, the returned results may not be satisfied. Another case is using an example video clip as the query to search the similar videos, i.e. query by video clip, which also has been actively researched [5, 6, 7]. This kind of system is suitable when the user can not clearly describe what they want in keywords, or the text index structure for the video database is unavailable, or they just want to search some specified video clips like pirated video copy detection. Compared with “query by keywords”, “query by video clip” provides a more flexible method to search the video database because usually a well-built text index structure is unavailable for a large video database. It is quite laborsome to manually label the whole video database while the performance of automatically indexing the video database is poor. For “query by video clip”, the query clip could be a sub-shot, a shot or several shots, based on the requirements of the users. Since the query clip is usually a logical story unit which contains cohesive semantical meaning, “query by video clip” is a more natural way for users to access and search the video database. The application of “query by video clip” comprises video copyright management [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20], video content identification in broadcast video [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36], and similar video content search by given example [6, 5, 7, 37, 38, 39, 40, 41].

1.1.2 Video Retrieval and Video Identification

We can classify video search systems into video retrieval and video identification based on their results. For video retrieval, we measure the similarity between the query and the video clips in database. The resulted video clips will be ranked by their similarities and returned to the users. The users will browse these results and decide which one is exactly they wanted, just like the text search engine. Thus, video retrieval is a measurement problem. For video identification, the system need to decide whether a video clip in database is a duplicated version or not based on the similarity matrix, so video identification is a decision problem. Video identification is a relatively new area compared to video retrieval. The topic of video retrieval has been extensively researched for more than ten years, but only recently has video identification been proposed as a new topic. The two areas are similar in some aspects. Some of the main research issues in video retrieval including video content representation and indexing are shared by video identification. Video identification can inherit many techniques from video retrieval. For example, those representation schemes used in video retrieval systems, such as key frame representation, color histogram feature, motion histogram, etc., are also used in some video identification systems [11, 24]. However, video retrieval and video identification are different:

Firstly, the query is different. The query of video retrieval could be text, shape, color or other properties that describe the video content; also it could be a query video clip. For video identification, the query must be a query video clip. Therefore, video identification definitely belongs to “query by video clip”, while “query by video clip” also includes some video retrieval systems which use the video clip as the query.

Secondly, video retrieval aims to search video clips that somehow look similar to the query, such as contain similar objects as the query, while video identification is to identify video clips that are perceptually the same, except for quality differences or the effects of various video editing operations. The results in video retrieval are similar to the query in semantical level, but for video identification they may be false alarms. Thus, the features for video identification need to be far more discriminatory, but they do not necessarily need to be semantical which is used for video retrieval.

Thirdly, video retrieval generally has the loop of relevance feedback in which user interaction is incorporated, i.e. users will decide which one is better in the returned video clips, but for video identification the system will output the final results. That is to say, generally video retrieval needs more manually work like in feature extraction, data supervision and training, etc., due to the poor performance of artificial intelligence on semantical level applications in current stage.

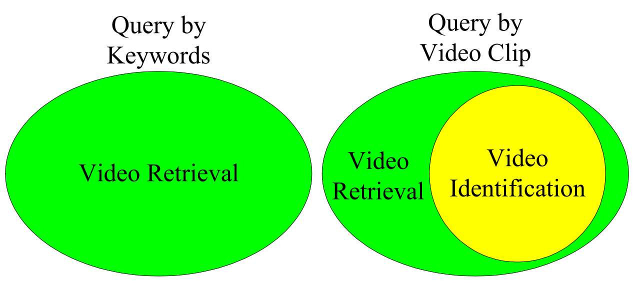


Figure 1.1 Two types of classifications for video search systems

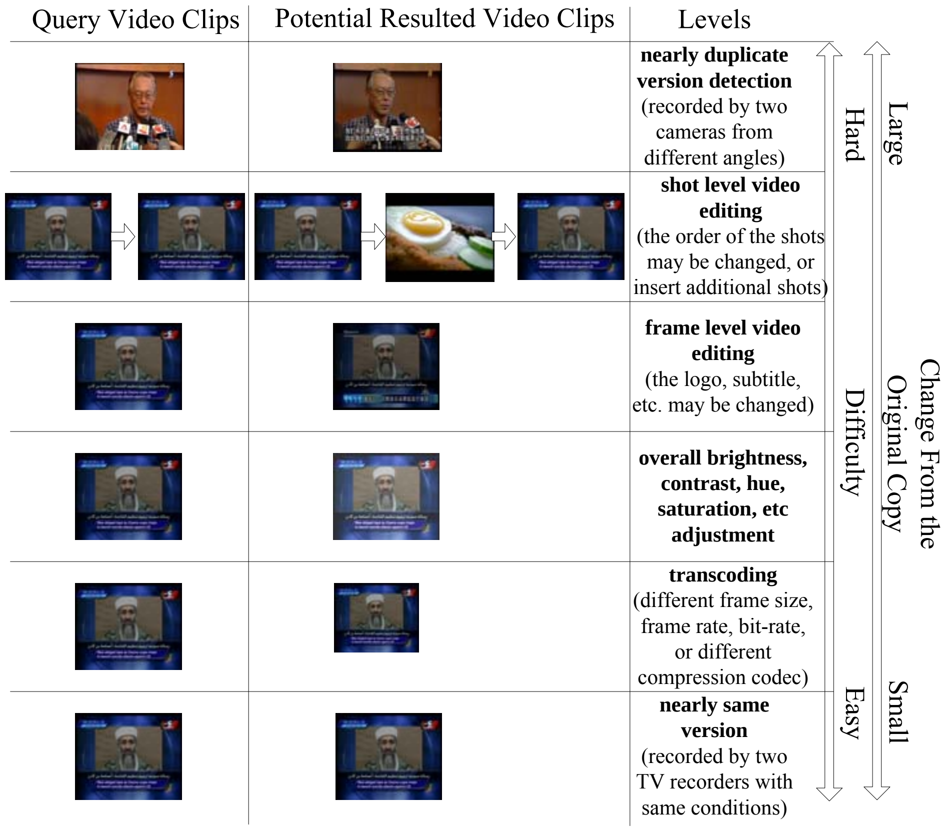
Figure 1.1 shows the relation of the above classifications for video search systems. Since the topic of this thesis is video identification, we will not discuss with "query by keywords" any more. For the case of "query by video clip", the differences between video retrieval and video identification result in different considerations and emphases on the system framework, although video retrieval and video identification have the same term of "similarity video search". For video retrieval, the task of retrieving similar video clips of the query at the concept level is associated with the challenge of capturing and modeling the semantical meaning inherent to the query. With an appropriate semantical model and similarity definition, video clips (a shot or several shots) with a similar concept as the query can be found [42]. However, for video identification, as the recognition task is relatively simple, complex concept level content modeling is usually unnecessary to identify and locate the duplicated versions of the query, but the prospective features or signatures are expected to be compact and robust to some variations, e.g. different frame size, frame rate, compression bit-rate and color shifting, brought by digitization, coding and post editing.

Furthermore, the methods and intentions to organize and manage the video database are different when targeting video retrieval and video identification tasks. Both of the tasks need to organize and index the video database, but their purposes are fundamentally different, even though they may apply the same term of "video indexing". For video retrieval, "video indexing" refers to annotating the video contents and classifying them into different concepts or semantical classes. By doing this, it could help the user to browse and retrieve the video content more effectively. On the other hand, "video indexing" mentioned in the video identification means to apply some basic database index techniques to organize the feature dataset extracted from the video contents, e.g. using a tree structure or hash index [43, 44]. Such a database index structure aims to provide an efficient method to accelerate the search speed. The nodes of the basic database index structure do not contain semantical level meaning, which is just the case for video retrieval indexing, to facilitate the video content browsing.

Finally, the search speed requirements are different for video retrieval and video identification. When doing video retrieval, normally we are not concerned with the search speed since the performances on precision and recall are not good enough. The bottleneck against a promising performance is the gap between low-level perceptual features and high-level semantical concepts. However, for video identification, the search speed is a big concern, because its applications are usually oriented to a very large video database or a time-critical online environment. On the other hand, compared with video retrieval, the task of video identification is relatively simple. Generally, video identification can achieve quite high precision and recall, which making efficient search possible.

Video identification and video retrieval are research issues on different levels. In fact, even inside video identification itself there are different level research problems. We will show different level video identification problems in next section.

1.2 Different Levels of Video Identification



Query Video Clips
Potential Resulted Video Clips
Levels
nearly duplicate version detection (recorded by two cameras from different angles)
shot level video editing (the order of the shots may be changed, or insert additional shots)
frame level video editing (the logo, subtitle, etc. may be changed)
overall brightness, contrast, hue, saturation, etc adjustment
transcoding (different frame size, frame rate, bit-rate, or different compression codec)
nearly same version (recorded by two TV recorders with same conditions)
Hard
Difficulty
Easy
Large
Change From the Original Copy
Small

Figure 1.2 Different levels of video identification

We divide the video identification problems into 6 levels based on the noise between the original and the duplicated version video clips. Figure 1.2 illustrates these different level problems of video identification. The systems for high level or semantical level video identification problems have to be robust to large noise, like recorded by cameras on different angles, different shot orders, various video editing operations, etc. These systems concern more on the performance of precision and recall than the search speed. Usually they need to apply some models and semantical level features to achieve acceptable results, which is a relatively difficult task. Compared with high level video identification, low level or exact match level video identification problems are easier. They only have small noise, like frame shift, transcoding, overall brightness adjustment, etc. Since nearly 100% of the performance on precision and recall can be achieved, low level video identification systems have more concerns on the search speed and scalability. Usually they will not apply models and their features do not necessarily need to be semantical, but have to be far more discriminatory. More details and some typical research works about each level are listed here:

1) Nearly duplicated version detection: The duplicated version video clip may be recorded by cameras from different angles. Some objects may be obstructed while some other objects may be reappeared because of the different view angles. Dong Qing Zhang et al. [36] presented a part-based image similarity measure derived from the stochastic matching of Attributed Relational Graphs that represent the compositional parts and part relations of image scenes. They compared this model with several prior similarity models, such as color histogram, local edge descriptor, etc. This presented model outperforms the prior approaches with large margin.
2) Shot level video editing: The order of the shots in duplicated version video clip may be different, or the duplicated vision can insert/delete shots into/from the original version. Victor Kulesh et al. [25] presented an approach for video clip recognition based on HMM and GMM for modeling video and audio streams respectively. Their method can detect the new shorter version of video clip which is produced by removing some shots from the original one.
3) Frame level video editing: The video editing operation is limited to frame level. The logo, subtitle, etc., may be changed. Timothy C. Hoad et al. [14] presented the shot-length comparison method for video identification. This method is found to be extremely robust to changes in the video, including alterations to the colors as well as changes in frame size, frame rate, bit-rate, and introduction of analogue interference, because the feature is not related to the content of a single frame.

4) Overall brightness, contrast, hue, saturation, etc. adjustment: This is common in different standard TV programs (like PAL, NSTC) conversion. Color (brightness) ordinal feature is useful for this kind of video identification [28, 33, 37], since ordinal measure is non-sensitive to uniform color shifting.
5) Transcoding level: The duplicated version video clip is transcoded from the original version. It may be different on frame size, frame rate, bit-rate or compression codec. Oostveen et al. [17] proposed a new hashing solution (i.e., perceptual/robust hash or fingerprints) and a database index strategy for video identification. Their fingerprints are robust to the above transcodings. Unfortunately, they did not report their performance on search speed. Our work in this thesis is also in this level.
6) Nearly same version level: The duplicated version video clip may be captured from real-time TV broadcasting using other TV recordings (in same conditions) which are different from their original version. There is only a little frame shift noise between the duplicated and original version video clips. Kunio Kashino et al. [31] proposed a quick search method for audio and video signals based on histogram pruning. They tested their algorithm on a 48h video database and get good performance on search speed.

1.3 Different Tasks of Video Identification

Besides the above 6 levels, there are 3 different tasks of video identification:

1) Task 1 is to find the identical video clips by comparing the query video with the videos in database [15]. The video database comprise of many short video clips. This task does not need to locate a short query video in a long video in database.
2) Task 2 is to identify the reoccurrences of some specified video segments in a long video clip [29]. The noise of task 2 is quite small because these reoccurrence video

segments are in the same video clip, i.e. the query videos have no distortions like changes on frame size, frame rate and compression bit-rate for a normal video identification application.

3) Task 3 is to search and locate a short query video clip in a large video database, which comprises of many long video clips [17, 31]. This is a general case for video identification, which is more difficult than the above two cases. Our work in this thesis is in this category.

1.4 Objectives

Our work in this thesis is located in the second lowest level of video identification problems, i.e. transcoding level. The task is to search and locate a transcoded version short query video clip in a large video database which comprises of many long video clips. That is to say, our objective is to build a highly efficient content-based video identification system which is robust to the transcoding level noise, i.e. changes on frame size, frame rate and compression bit-rate.

1.5 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 gives a broad survey about content-based video identification. Some backgrounds about similarity search in high-dimensional database and locality sensitive hashing (LSH) are also provided since they are closely related to this thesis. Chapter 3 presents our highly efficient video identification system for a large video database based on improved locality sensitive hashing and triangle inequality. Chapter 4 evaluates our system performance. Finally, chapter 5 concludes the thesis and points out the future work.

Chapter 2

Background and Related Work

In this chapter, some backgrounds and related work are provided. Firstly, we will give a survey of related issues to video identification which include “feature extraction”, “similarity measuring” and “index structures”. Some profound surveys about video search can be found in [1, 45, 46, 47]. Secondly, we will give some backgrounds about efficient similarity search in high-dimensional space via database index structures, which is closely related to this thesis. Finally, we will introduce locality sensitive hashing (LSH), a highly efficient index structure applied in our work.

2.1 Content-Based Video Identification: A Survey

2.1.1 Architecture of a Video Storage and Identification System

A systematical video database used for video identification has two main processes: storage and identification. The storage process extracts features from videos and organizes these feature vectors for storage in the database. In the identification process, an input query is represented by the appropriate features, and a search is formed on the stored feature vectors to find the closest videos. A similarity metric is used to measure the similarities between the query video and the videos in database. The feature vector indexing structure can improve the search efficiency. Figure 2.1 shows the architecture of a video storage and identification system.

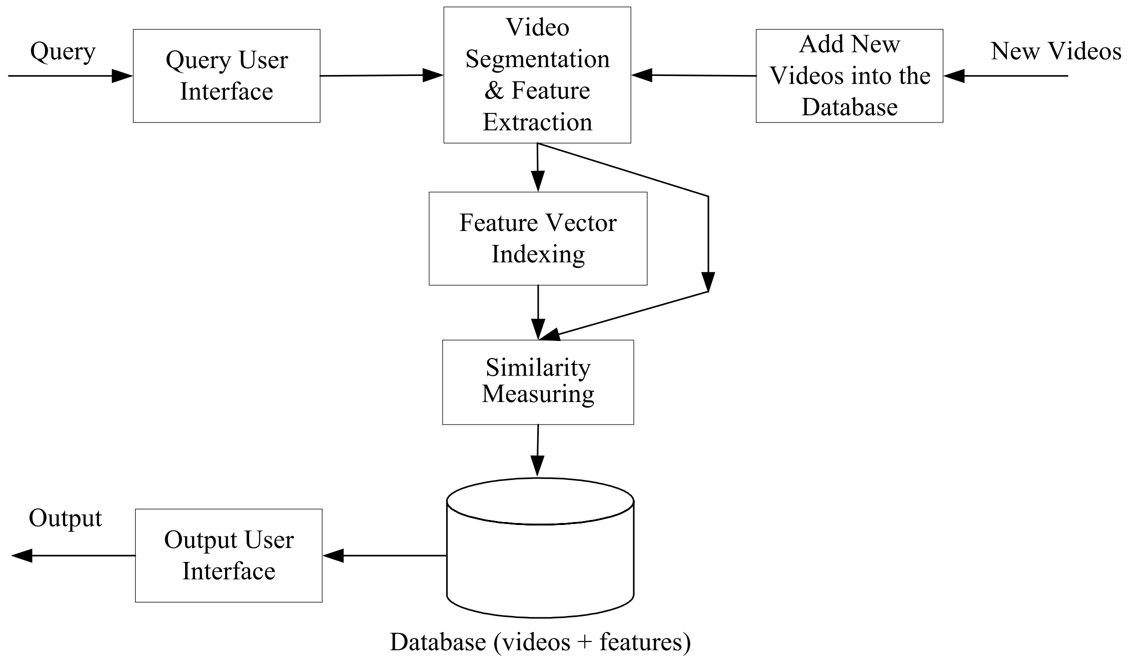


Figure 2.1 Architecture of a video storage and identification system

In the above system, there are 3 key modules: (i) video segmentation and feature extraction; (ii) similarity measuring; (iii) feature vector indexing. Some high level or semantical level video search systems do not have module "feature vector indexing", which is useful for increasing the search speed, because they only care the performance on precision and recall in current stage.

2.1.2 Video Segmentation and Feature Extraction

This module is the main part of the whole video search system. Lots of research work has been done for this module [48]. Figure 2.2 shows how to extract features to represent a video clip. Video has both spatial and temporal dimensions and hence a good video index should capture the spatiotemporal contents of the scene. Normally, a video is first segmented into elemental video segments (scenes or shots). For some video databases which only comprise short video clips (e.g. task 1 in section 1.3), this step may be skipped and the whole video clip is treated as one video segment. These video segments are regarded as the basic units for indexing and search. Next, the module extracts feature vectors for every video segment. These feature vectors may be spatial features such as color, texture, sketch and shape from key frames, or temporal features such as object motion and camera operation, or some features based on the video segment itself, like the length of the video segment. For all these features, some are on semantical level and often used for video retrieval applications like camera operation, objection motion, spatial relation, etc., while other low level features are more suitable for video identification applications.

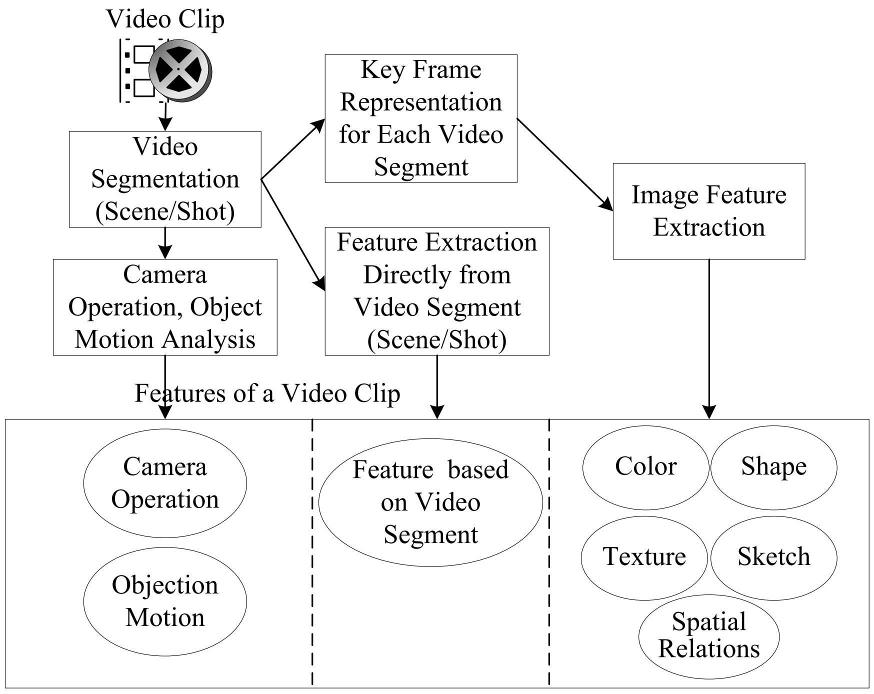

Video Clip
Video Segmentation (Scene/Shot)
Key Frame Representation for Each Video Segment
Image Feature Extraction
Camera Operation, Object Motion Analysis
Feature Extraction Directly from Video Segment (Scene/Shot)
Features of a Video Clip
Camera Operation
Objection Motion
Feature based on Video Segment
Color
Shape
Texture
Sketch
Spatial Relations

Figure 2.2 Structure of video segmentation and feature extraction module

Color histogram is often used for video identification because its simplicity and relatively good robustness and discriminability. Cheung et al. [15] used HSV color histogram of the key frames to represent a short video clip. Naphade et al. [22] applied color histogram intersection to compute the similarity between two clips. They verified that color histogram intersection is an effective and fast method for video sequence matching. Ferman et al. [39] used robust color histogram descriptors called alpha-trimmed average histogram to represent a group of frames (GoF). This is a generalized version of the average histogram and the median histogram. Unless strong luminance and/or chrominance variations are observed throughout a GoF, the average histogram (i.e. $\alpha = 0$) can be used to provide a reliable representation of the GoF color content, with minimal computational overhead. Otherwise, a non-zero value for the trimming parameter will be adopted to reduce or eliminate the effects of these variations.

Color (brightness) ordinal feature is also applied for video identification [28, 33, 37]. Since ordinal measure is non-sensitive to uniform color shifting, which is a kind of typical color distortions in TV program, the formed ordinal representation can represent key frames robustly.

Texture-based methods are similar to the color histogram methods. Instead of using a feature vector based on color, similarity is computed based on a feature vector that represents the contrast, grain, and direction properties of the image [49]. This method has the efficiency performance problems, as texture histograms are generally more expensive to produce than color histograms. This method would also be sensitive to encoding artifacts and changes in encoding bit-rate, as texture information is often lost at low bit-rate. That is to say, texture-based features are not quite robust to transcodings on bit-rate.

Timothy C. Hoad et al. [14] presented the shot-length comparison method for video identification. This method is found to be extremely robust to changes in the video, including alterations to the colors as well as changes in frame size, frame rate, bit-rate, and introduction of analogue interference, because the feature is not related to the content of a single frame. However, there are some limitations when it is applied to certain content. Queries that contain only a small number of shots could not be reliably identified. Similarly, errors in cut-detection lead, in some case, to considerable reduction in query effectiveness.

Arun Hampapur et al. [50] compared the performance of a number of image distance measures (color histogram intersection, image difference, edge matching, edge orientation histogram intersection, invariant moments and Hausdorff distance) for comparing video frames for the purpose of video copy detection. In their experimental results, the local edge measure proposed in [10] has good performance. However, the number of bits of indexing information required for one frame is quite large, and the computational complexity is heavy to compute local edge representation for each frame in a video clip.

2.1.3 Similarity Measuring

After effectively represent the given query clip and the clips in video database by features, the next step is similarity measuring. Current video searching methods based on representative images matching can be summarized into three main categories: frame sequence matching [21, 37, 22, 31], key-frame based shot matching [24, 14] and subsampled frame matching [5, 38, 26].

Although frame sequence matching attained certain level of success in [21, 37, 22], the common drawback of these techniques is the heavy computational cost of exhaustive search. [31] improved on this by skipping unnecessary steps during the search, while guaranteeing exactly the same search result as exhaustive search.

Key-frame based shot matching is another popular method [24, 14] for video identification and retrieval. When applied to short video clip searching, this method, however, has some drawbacks. First, the performance of shot representation strongly depends on the accuracy of shot segmentation and characteristics of the video content. For example, if the given clip has blurry shot boundaries or very limited number of shots, shot-based searching will not produce good results. Second, shot resolution, which could be a few seconds in duration, is usually too coarse to accurately locate the instances in the video stream.

Some other methods [5, 38, 26] consider sub-sampled frame matching for video stream searching. Although search speed can be accelerated by using coarser temporal resolution, these methods may suffer from inaccurate localization. When the sub-sampled frames of the given clip and that of the matching window are not well aligned in temporal axis, it will affect the matching result. [26] partially overcomes this sub-sampled frame shifting problem and is robust to video frame rate change. However, feature extraction in [26] is time consuming, therefore not suitable for on-line processing and large video database search.

2.1.4 Feature Vectors Indexing

In the above research work, they try different kinds of content-based features and similarity measuring methods to achieve better performance on precision and recall. Among all these methods for video identification applications, only a few concerned the speed performance and have been tested on a large video database:

Cheung et al. [15] summarized each video with a small set of sampled frames, called the Video Signature, and then extracted the HSV color histograms of these frames as the features. They tested their method on a collection of 46,356 video sequences. However, their method can only judge if two short video clips are identical or not, that is to say, their method cannot detect and locate the short query video in a large video database.

Oostveen et al. [17] proposed the concept of video fingerprinting and a database index strategy for video identification. Fingerprints, also named as perceptual/robust hashes, are defined as follows: a fingerprint function is the function that (i) maps (usually bitwise large) audiovisual objects to (usually bitwise small) bitstrings (fingerprints) such that perceptual small changes lead to small differences in the fingerprints and (ii) such that perceptually very different objects lead with very high probability to very different fingerprints. With fingerprints, an index structure can be constructed to achieve efficient video identification. Unfortunately their hash table will be not efficient if the entries are not evenly distributed which is just the case for most videos.

Kok Meng Pua et al. [29] presented a real time repeated video sequence identification system based on video sequence hashing. Color moments are used to extract the hash bitstring. They evaluated their system on a 32h video continuous stream and get real time performance, but they also face the problem of non-uniform distribution for the hash table. Moreover, since the hash table is not robust enough, their method is only limited to search repeated video segments inside a large video database, where the query videos have no distortions like changes on frame size, frame rate and compression bit-rate for a normal video identification application.

Kunio Kashino et al. [31] proposed a quick search method for audio and video signals based on histogram pruning. They used the histogram of a set of consecutive frames' color distribution as the feature, and gave an "active search" algorithm to skip the redundant match operations, where a match operation is a computation on the distance between two feature points and the number of total match operations (CPU time) is used to measure the performance. They tested their algorithm on a 48h video database and get good performance. However, their feature dataset may be too large to be fit in the main memory, which introduces additional I/O cost, and the efficiency could be further increased by applying some index structure.

2.1.5 Some Well-known Video Search Systems

We will introduce some well-known video search systems in this subsection:

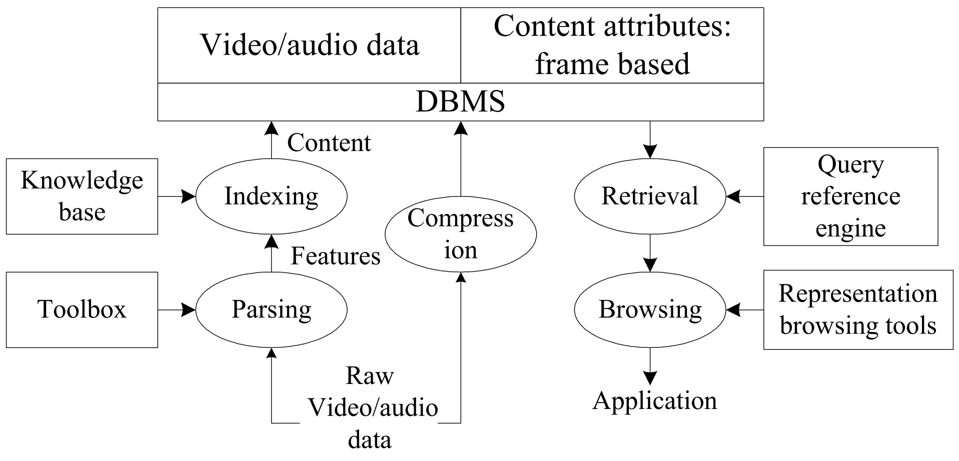


Figure 2.3 Architectural diagram of a video retrieval system

(Figure is adapted from "S. W. Smoliar, H. J. Zhang, "Content-based video indexing and retrieval," in IEEE Multimedia, vol.2, no.1, pp.63-75, Summer 1994")

Stephen W. Smoliar et al. [51, 4] presented a content-based video indexing and retrieval system. Figure 2.3 summarizes this system in an architectural diagram. The heart of the system is a database management system containing the video and audio data from video source material that has been compressed wherever possible. The DBMS defines attributes and relations among these entities in terms of a frame-based approach to knowledge representation. This representation approach, in turn, drives the indexing of entities as they are added to the database. Those entities are initially extracted by tools that support the parsing task. In the opposite direction, the database contents are made available by tools that support the processing of both specific queries and the more general needs of casual browsing.

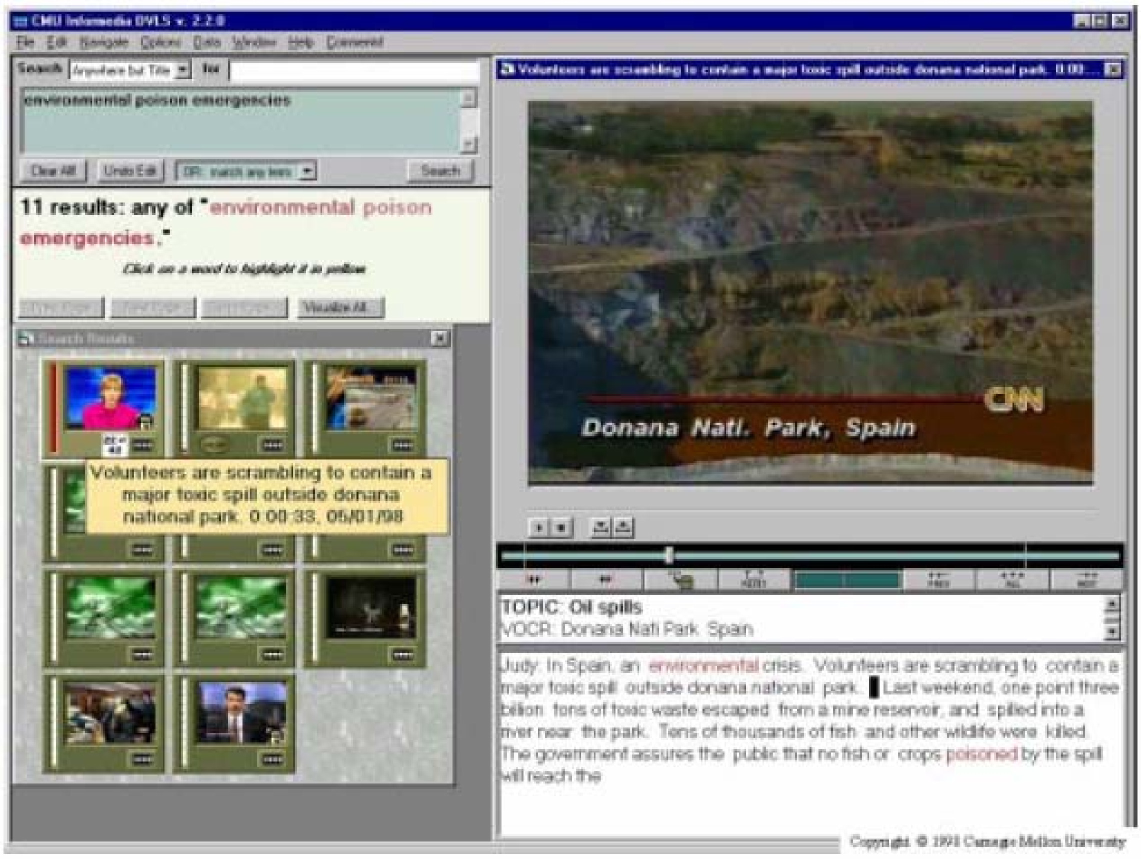
Myron Flickner et al. [3] presented the famous QBIC (query by image and video content) system. QBIC allows queries on large image and video database based on

- example images,
- user-constructed sketches and drawings,
- selected color and texture patterns,

- camera and object motion,
- other graphical information.

Two key properties of QBIC are (i) its use of image and video content – computable properties of color, texture, shape, and motion images, videos, and their objects – in the queries, and (ii) its graphical query language in which queries are posed by drawing selecting and other graphical means. QBIC has two main components: database population (the process of creating an image database) and database query. During the population, images and videos are processed to extract features describing their content – colors, textures, shapes, and camera and object motion – and the features are stored in a database. During the query, the user composes a query graphically. Features are generated from the graphical query and then input to a matching engine that finds images or videos from the database with similar features.

Howard D. Wactlar et al. [2] presented the Informedia digital video library project. The Informedia system provides "full-content" search and retrieval of current and past TV and radio news and documentary broadcasts. The system implements a fully automatic intelligent process to enable daily content capture, analysis and storage in on-line archives. The library consists of approximately 2,000 hours, 1.5 terabyte library of daily CNN news captured over the last 3 years and documentaries from public television and government agencies. This database allows for rapid retrieval of individual "video paragraphs" which satisfy an arbitrary spoken or typed subject area query based on a combination of the words in the soundtrack, images recognized in the video, plus closed-captioning when available and informational text overlaid on the screen images. There are also capabilities for matching of similar faces and images, generation of related map-based displays. Figure 2.4 shows an interface of Informedia system.


CMU Informedia DVLS v. 2.2.0
File Edit Navigate Options Data Window Help Comment
Search Anywhere but Title for
Clear All
Undo Edit
OR: match any term
Search
Visualize All
Search Results
CNN
Donana Nati. Park, Spain
Copyright © 1998 Carnegie Mellon University

environmental poison emergencies

11 results: any of "environmental poison emergencies."

Click on a word to highlight it in yellow

Volunteers are scrambling to contain a major toxic spill outside donana national park. 0:00:33, 05/01/98

Volunteers are scrambling to contain a major toxic spill outside donana national park. 0:00...

TOPIC: Oil spills
VOCR: Donana Nati Park Spain

Judy: In Spain, an environmental crisis. Volunteers are scrambling to contain a major toxic spill outside donana national park. Last weekend, one point three billion tons of toxic waste escaped from a mine reservoir, and spilled into a river near the park. Tens of thousands of fish and other wildlife were killed. The government assures the public that no fish or crops poisoned by the spill will reach the

Figure 2.4 Interface of Informedia system

(Figure is adapted from "H. D. Wactlar, T. Kanade, M. A. Smith and S. M. Stevens, "Intelligent access to digital video: Informedia project," in IEEE Computer, vol.29, no.3, pp.46-52, May 1996")

2.2 Similarity Search via Database Index Structure

For large video database applications, the system efficiency (e.g. search time, database size, etc.) could be a big issue. Just as high-speed and high-volume text search engines have been widely used, we believe that the quick search algorithms on large video dataset may soon become the basic technologies for handling large volume video data. Thus, besides "Feature Extraction" and "Similarity Measuring", "Feature Vector Indexing" is an important module for a video identification system on a large video database, since it can significantly reduce the search space to improve the search speed.

There are mainly two kinds of similarity search problems in database indexing area, i.e. nearest neighbor search and ε-range search. Here are the definitions:

Definition: Nearest Neighbor Search

Given a set P of n objects represented as points in a normed space l_p^d, preprocess P so as to efficiently answer queries by finding the point in P closest to the query point q.

Definition: ε-Range Search

Given a set P of n objects represented as points in a normed space l_p^d, preprocess P so as to efficiently answer queries by finding all the points in P that the distances between these points and the query point q is lower than the threshold ε.

There are many well-known index structures which support the above similarity search problems, such as K-D-B tree [52], R^*-tree [53, 54], TV-tree [55], X-tree [56], SS-tree [57], SR-tree [58], etc. Some researchers in database fields have started studying how to efficiently and accurately index multimedia such as image, video database. However, these index structures do not work well for high-dimensional multimedia data. Roger Weber [59] showed in theory and in practice that all above space- and data-partitioning methods will suffer from the *dimensional curse*, which means their performance will degrade to linear search as the number of dimensions increases (above 20 dimensions). In fact, these index structures insist too much on the indexing accuracy (e.g., finding the exactly nearest feature point to locate to the single video frame) by assuming that an accurate and robust feature set can be obtained by means of some multimedia analysis tools. Such assumption is very hard or even impossible to be realized in practice because hundreds of consecutive video frames may look very similar in a video. On the other hand, exactly locating to a single frame may not be necessary for most video-related applications, since in multimedia applications, the meaning of “exact” is highly subjective. Because of the nature of these applications it is usually not very meaningful to pursue exact answers in such applications. Moreover, the features themselves are approximate representations of the real world entities. They model the real data, but not always with 100% accuracy. Therefore, some researchers think about time-quality tradeoff. They apply approximate similarity search to achieve better performance with a little cost of accuracy. Locality sensitive hashing (LSH, see next subsection) [60] is one of such methods.

Hash table is a highly efficient index structure for large database. While traditional hashing methods are not robust to some kinds of noise which is common in video-related applications, researchers try to find the robust video hashing solutions [17, 29]. A general way to generate the hash index bitstring from features is quantization. However, the hash bitstring generated from the feature point is not robust if this feature point is near to the quantization threshold. A little noise may make the point cross the threshold and generate different hash bitstring. Locality sensitive hashing is more robust because it uses the random quantization thresholds and multiple hash functions/tables, and the robustness will be increased as we increase the number of hash functions/tables. Therefore, LSH is suitable for video hashing to achieve efficient video identification. We will give more details about LSH in next subsection.

2.3 Introduction to Locality Sensitive Hashing

Aristides Gionis, Piotr Indyk and Rajeev Motwani [60] proposed locality sensitive hashing (LSH) for highly efficient approximate similarity search. Traditional hashing functions are used to build several hash tables as the index structure. The principle is that the probability of collision of two points p and q is closely related to the distance between them. Specifically, the larger the distance, the smaller the collision probability. For one hash table, they first partition the space randomly into high-dimensional cubes.

Then, they use bitstrings to represent every cube, and all the points in the same cube have the same bitstring. Finally, they apply traditional hashing function to map all these points (bitstrings) into a hash table, so the points in the same cube are mapped into the same bucket in the hash table. Several hash tables are used to prevent missing the near neighbors. Figure 2.5 illustrates LSH more clearly.

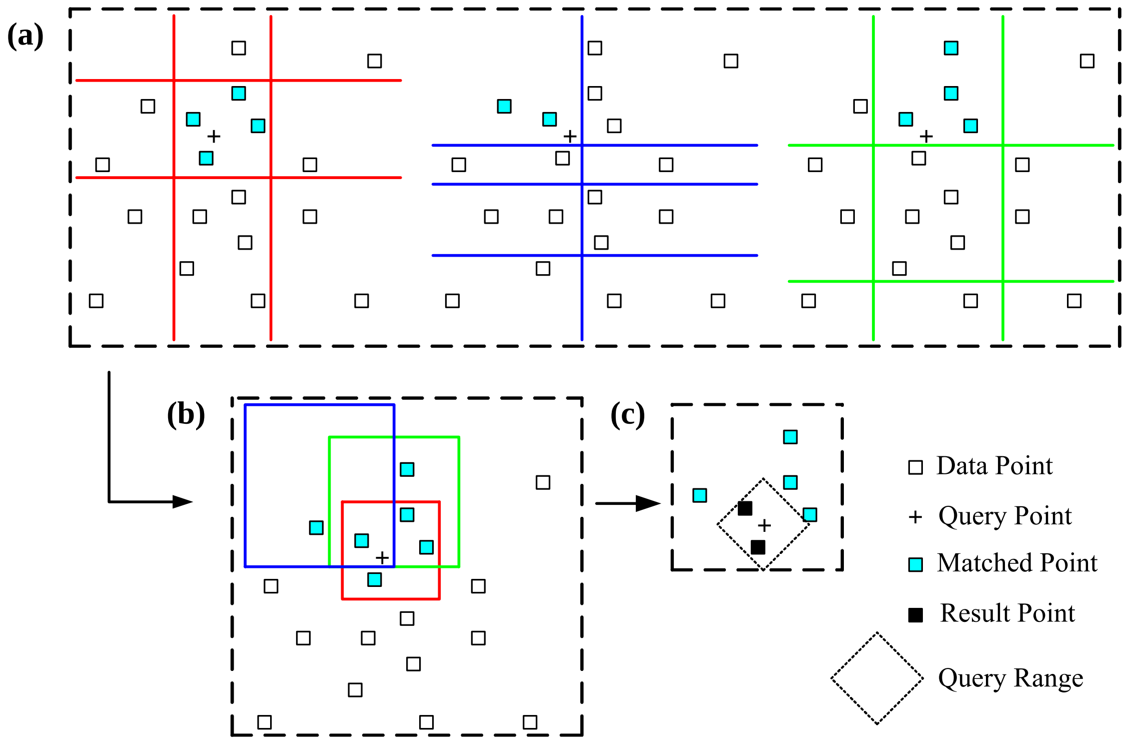


Figure 2.5 A 2D example of merging the results from multiple hash tables

Figure 2.5 shows a 2D example of hash tables in LSH. In this example, we have 3 hash tables. We build these hash tables by randomly partitioning the space into cubes and mapping all the points into hash tables. For a query point, we also map it into all hash tables and return all the buckets in which it is located. In Figure 2.5(b) we merge the points in these returned buckets to build the candidate set. In Figure 2.5(c) we search the candidate set linearly to find the near neighbors that satisfied the condition.

With LSH, we can reduce the query time significantly. The query time is increased sub-linearly with the size of the database $O(dn^{1/(1+\varepsilon)})$ and the preprocessing cost polynomial in n and d, i.e. $O(n^{1+1/(1+\varepsilon)} + dn)$. Figure 2.6 is a disk accesses comparison between LSH and SR-tree, another well-known similarity search index structure.

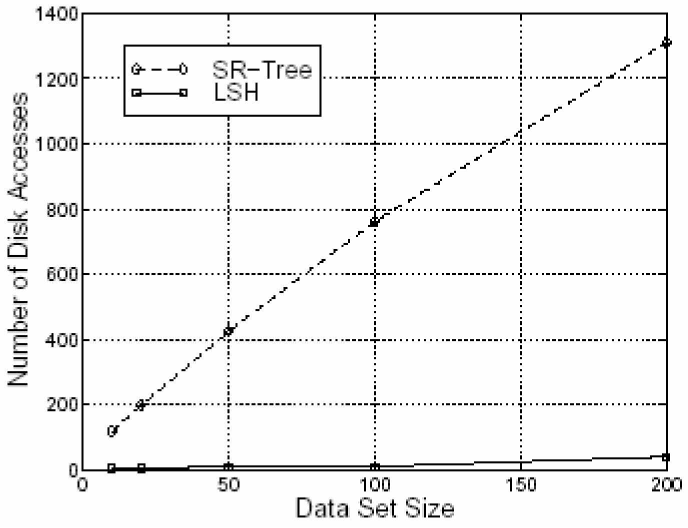


Figure 2.6 Disk accesses comparison between LSH and SR-tree

(Dimension , dataset size from 10,000 to 200,000)

(Figure is adapted from “A. Gionis, P. Indyk and R. Motwani, “Similarity search in high dimensions via hashing,” in Proceedings of International Conference on Very Large Data Bases, pp.518-529, Sep 1999, Edinburgh, Scotland”)

Chapter 3

Efficient Video Identification Based on Locality Sensitive Hashing and Triangle Inequality

In this section, we present an efficient video identification system for a large video database by systematically taking “feature extraction”, “feature indexing” and “video database construction” together into consideration. The selected feature is robust to the changes on frame size, frame rate and compression bit-rate. Principal components analysis (PCA) and improved locality sensitive hashing (LSH) are then used to reduce the dimensions of feature space and generate the index code. Considering that the original LSH is only good for indexing uniformly distributed high-dimensional data points and can be improved for video identification where data points may be clustered. We therefore give two improvements to LSH to distribute the points more evenly. First, by building a hierarchical hash table, we adapt the number of hashed dimensions to the density of the data points. Second, we choose the hashed dimensions carefully in such a way that the points are more evenly hashed, thus making the hash table more uniformly distributed and reducing the miss rate. We further apply triangle inequality on the resulted buckets by LSH to skip some redundant match operations. In terms of system design, to save the storage of the video database's feature dataset, we slide the search window on the query video rather than the videos in database.

The rest of this section is organized as follows. Section 3.1 presents the system overview. Section 3.2 explains how to slide the search window on the query video to reduce the feature dataset size and formulates the video identification problem as a ε-range search problem in the high-dimensional space. Section 3.3 describes the LSH and our improvements. Section 3.4 introduces employing triangle inequality on the resulted buckets by LSH to skip some redundant match operations. Section 3.5 focuses on selecting and extracting the robust features for video identification.

3.1 System Overview

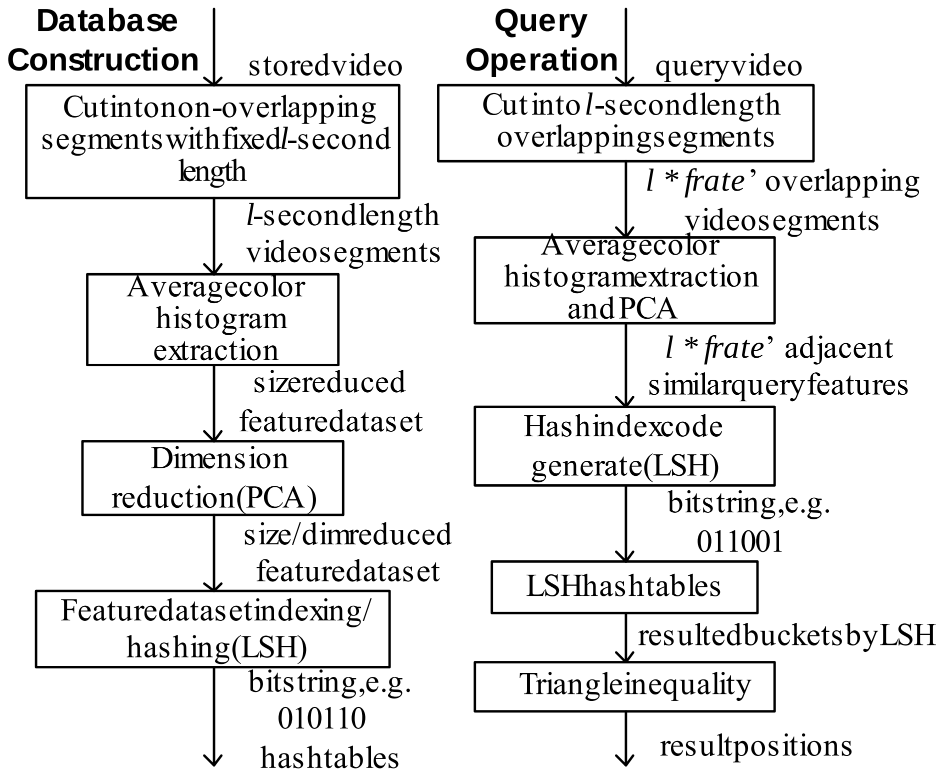

Database Construction
storedvideo
Cutintonon-overlapping segmentswithfixedl-second length
l-secondlength videosegments
Averagecolor histogram extraction
sizereduced featuredataset
Dimension reduction(PCA)
size/dimreduced featuredataset
Featuredatasetindexing/ hashing(LSH)
bitstring,e.g. 010110
hashtables
Query Operation
queryvideo
Cutintol-secondlength overlappingsegments
l * frate' overlapping videosegments
Averagecolor histogramextraction andPCA
l * frate' adjacent similarqueryfeatures
Hashindexcode generate(LSH)
bitstring,e.g. 011001
LSHhashtables
resultedbucketsbyLSH
Triangleinequality
resultpositions

Figure 3.1 System overview

Figure 3.1 gives a brief overview of our system.[1] During the database construction period, we first extract the average color histogram feature for each l-second video segment of the video in database (stored video), and then we apply principal components analysis (PCA) to reduce the dimensions of the extracted features. After we obtained a size and dimension reduced feature points dataset, we use locality sensitive hashing (LSH) to generate the index hash code for this dataset. For the query operation (given a short query video clip), we extract similar features from l-second length overlapping video segments. Then, we generate the hash index code via LSH and get the resulted buckets. We further employ triangle inequality on the resulted buckets by LSH to reduce some redundant match operations during the final linear search within these buckets.

3.2 Slide Search Window on Query Video

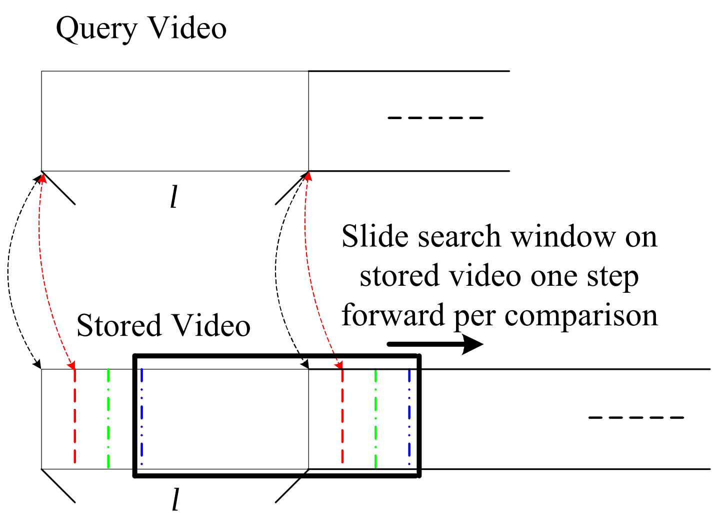

Query Video
l
Slide search window on stored video one step forward per comparison
Stored Video
l

Figure 3.2 A usual video search algorithm

Figure 3.2 gives a framework for a usual video search algorithm. Firstly, it cuts the query video into l-second length segments and extracts the feature vector for each

[1] We cut the stored (database) video and the query video by different ways to reduce the stored video's feature dataset size (see section 3.2).

segment. Here we use the l seconds instead of frames as the length of each video segment because the query video and the video in database may be differ in frame rate, result in different number of frames in one segment. Then we use one of these query features to search in the same type of features of the video in database (stored video), which can be computed offline. Once a candidate position found, the whole query video will be compared to the same length video segment of the stored video at this position to decide whether it is a duplicated version. The moving step of the stored video's compare window can be adjusted to do trade-off between robustness and speed. To reduce the temporal frame shift noise, we set the step to 1 frame, which is the minimal value.

A problem of this algorithm is that the size of the stored video's feature dataset may be too large to fit in the main memory, so additional I/O cost will be introduced. For example, we use 96-hour videos as the stored video for experiments. 96-hour videos with the frame rate of 29.97 fps have 10,357,632 frames and we have one feature point per frame. Suppose we use 120-dimension feature vectors and each dimension takes 4 bytes, thus the whole feature set size is 4,971,663,360 = 4.9G bytes.

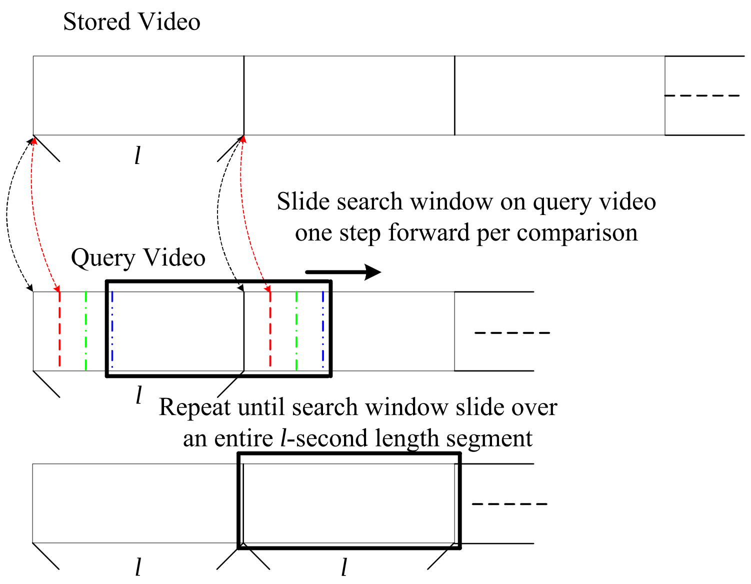

Stored Video
l
Slide search window on query video
one step forward per comparison
Query Video
l
Repeat until search window slide over
an entire l-second length segment
l
l

Figure 3.3 Slide search window on query video

To reduce the size of the feature dataset, we cut the stored video into the non-overlapping segments with fixed l-second length and then extract the feature points from these segments to build the feature dataset, i.e. one feature point per l-second video ($l * frate$ frames). This is shown in Figure 3.3. Furthermore, instead of sliding the search window on the stored video we slide it on the query video for the comparison. Note that to maintain the nearly same accuracy as the previous method, we have to slide the search window over an entire l-second length segment on query video because definitely there is a l-second length overlapping segment in any continuous $2l$-second length query video which is exactly aligned to one segment in the stored video. These two aligned video segments may have little temporal frame shift if the query video and the stored video are differ in frame rate. Finally, we get a size reduced feature dataset which is $1/(l * frate)$ of the original dataset size, and $l * frate'$ more queries which the adjacent queries are very similar. The similarity of these queries will benefit the system performance by employing triangle inequality. Here $frate$ and $frate'$ are the frame rates of the stored video and the query video respectively. These two advantages also bring us an additional constraint on the length of the query video clip, i.e. the query video has to be longer than 2 segments (i.e. $2l$ seconds) while the previous method only requires 1 segment. Actually such constraint can be easily met by carefully selecting l (e.g. $l = 4$s) for the tasks such as searching news and commercial videos because a typical length of news and commercial videos is usually longer than 8s. In this case, the feature dataset size of the 96-hour videos is reduce to 1/120 of the original size, i.e. 40M bytes compared to 4.9G bytes and we have 120 similar queries if the query video and the stored video have the same frame rates.

We use the average color histogram [39] as the features, i.e. averaging the color histogram of every frame in one segment. We employ histogram intersection to measure the similarity [22], which is equivalent to the L_1 distance measure [61]. Finally, we formulate the problem as a ε-range search problem in high-dimensional space:

We have a list of query points $q_1, q_2, ... q_m$ searched on the points dataset $P = \{p_1, p_2, ..., p_n\}$, where q_i, p_j are D-dimension feature points in a norm space L_1^D. We wish to find all points in P for every q_i, that the L_1 norm distance is lower than the threshold ε, i.e.

$$\underset{p_j \in P}{Dis}(q_i, p_j) < \varepsilon \qquad i = 1, 2, ..., m \tag{3.1}$$

We count one computation for the distance between two points as one match operation and use the number of total match operations to measure the cost of the algorithm. Here, the adjacent queries q_i are similar. Dimension D is large for the features of video application. Threshold ε is small because for duplicated version video search, the difference is low. Therefore, we can apply locality sensitive hashing (LSH) to index this high-dimensional dataset with the low miss rate because of the small threshold ε.

3.3 Improvements to Locality Sensitive Hashing

3.3.1 Description of Locality Sensitive Hashing

In section 2.3, we have introduced locality sensitive hashing (LSH). The idea behind LSH is rather simple. It randomly partitions a high-dimensional space into high-dimensional cubes. Each cube is a hash bucket.[2] A point is likely to be located in the same bucket as its near neighbors. Given a query point, we determine which bucket the point is located in, and perform linear search within this bucket to check the distances

[2] In practice, we may hash the bitstring representing the cube using traditional hash functions, resulting in multiple cubes in a bucket.

of these candidate points. The hash function is therefore a mapping from the high-dimensional point to the bitstring representing the bucket the point is in. It is possible that we may miss some points whose distances are lower than the threshold ε if these points have been hashed to a different bucket than the query point (e.g. point A in Figure 3.4, left). To reduce the likelihood of this, LSH maintains multiple hash tables, hashing a point multiple times using different hash functions. The probability that a query point and its near neighbors are hashed into different buckets by these hash functions can be reduced by reducing the number of buckets or increasing the number of hash tables. In theory, the miss probability will be reduced exponentially as the number of hash tables increasing because the hash tables are independent. We will show this result in our experiments. Finally, the buckets that the query point is hashed into for all hash tables will be merged together to build the candidate set C for the final linear search.

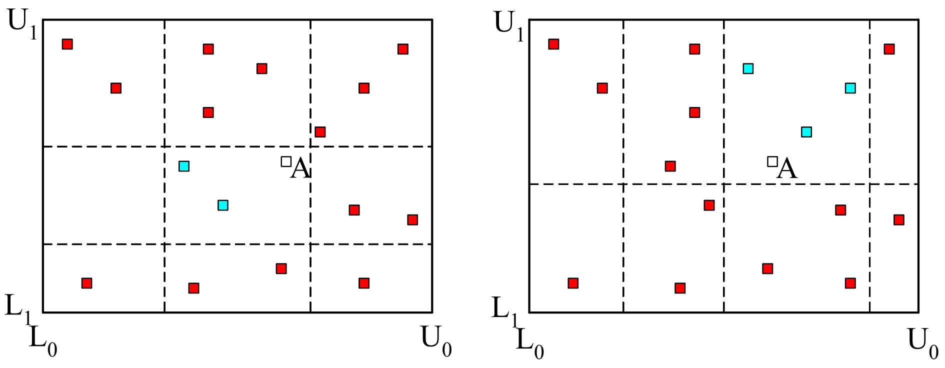


Figure 3.4 Locality sensitive hashing

We can now describe LSH more formally. Let D be the dimension of the vector space, and $[L_i, U_i]$ be the range of possible values for dimension i. Each hash table in LSH is parameterized by k, the number of hashed dimensions; $d = \langle d_0, d_1, ..., d_{k-1} \rangle$, the hashed dimensions; and $t = \langle t_0, t_1, ..., t_{k-1} \rangle$, a quantization threshold vector. d_i is chosen uniformly at random from $[0, D-1]$ while t_i is randomly chosen from $\left[L_{d_i}, U_{d_i}\right]$.

Given a point $p = \langle p_0, p_1, ..., p_{d-1} \rangle$, we hash it into a k-bit bitstring $b = \langle b_0, b_1, ..., b_{k-1} \rangle$, representing the bucket, where

$$b_i = \begin{cases} 1 & p_{d_i} > t_{d_i} \\ 0 & p_{d_i} \le t_{d_i} \end{cases} \qquad i = 0,1,...,k-1 \tag{3.2}$$

LSH builds N such hash tables, each with different d and t. The values of N and k can be tuned to change the probability that we miss the points whose distances are lower than threshold ε. Figure 3.4 illustrates LSH in 2-dimensional space with $k = 4$ and $N = 2$.

3.3.2 Improvements to Locality Sensitive Hashing

The major factor that determines the efficiency of LSH is the size of the bucket the query points hashed to. Since for each query point, we need to check through all points in the same bucket to find if their distances are lower than threshold ε or not. We would like the points to be evenly distributed among the buckets. However, LSH does not always give such distribution, especially for multimedia dataset. In this subsection, we illustrate two such problems with LSH and propose two improvements to it.

a) Hierarchical LSH

Currently, LSH partitions the space without considering the distribution of points. In most cases, the image/video dataset is not uniformly distributed [58, 62]. For example, in Figure 3.5(a), we see that the number of points in the middle bucket is large. Check the near neighbors of point A will involve many match operations, thus reducing the efficiency of LSH. One way to solve this problem is to increase k, the number of hashed dimensions. The resulting partitions are shown in Figure 3.5(b). While this reduces the number of points in each bucket, it reduces the accuracy as well since some query points in sparse area such as point B will miss the near neighbors. Another problem for the fixed bucket size is that buckets according to cubes in sparse area may be nearly empty while buckets according to cubes in dense area are already full and can not accept new points. Thus, the hash table is inefficient and hard to expand.

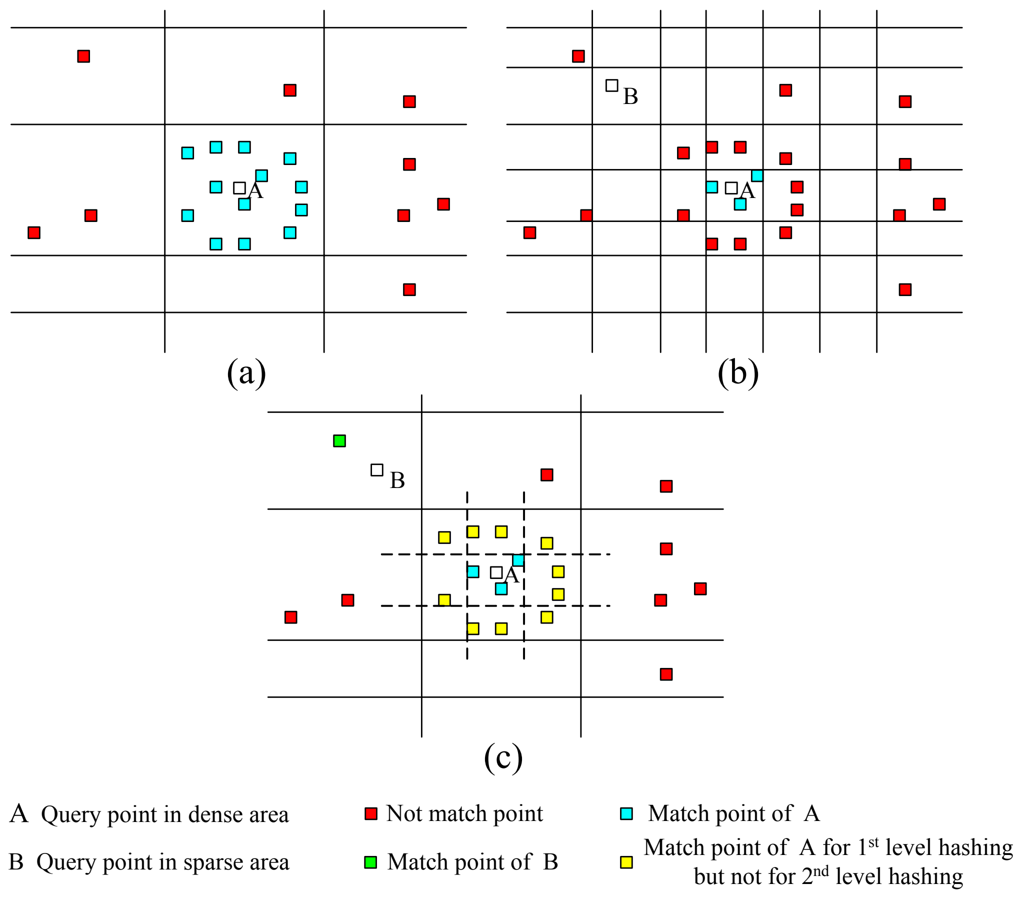

A
B
A
(a)
(b)
B
A
(c)
A Query point in dense area
B Query point in sparse area
Not match point
Match point of B
Match point of A
Match point of A for 1st level hashing but not for 2nd level hashing

Figure 3.5 Hierarchical partitioning in locality sensitive hashing

Our solution to this problem is illustrated in Figure 3.5(c). When the number of points hashed to a bucket exceeds a threshold, we repartition the bucket, and rehash all points in this bucket. This scheme establishes a hierarchy of hash tables in dense area. It reduces the size of the candidate set C for linear search while keeping the miss probability low.

b) LSH with Non-uniform Partition

Another problem of LSH is that the space is partitioned randomly using uniform distribution. This works well when the values of each dimension are evenly distributed. In image/video dataset, points may be denser in one dimension compared to another. For example, in the case of video identification, some features may be more sensitive than others in differentiating videos. Figure 3.6(left) illustrates the problem.

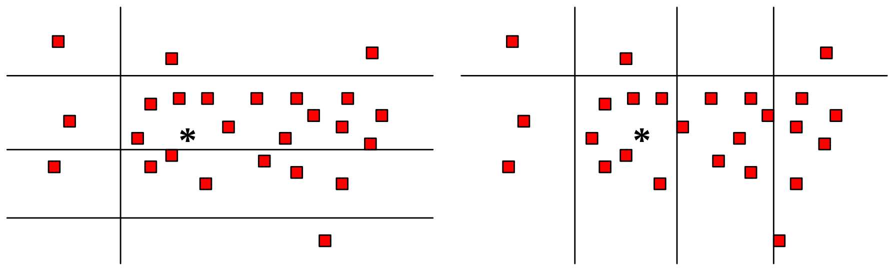


Figure 3.6 Non-uniform selection of partitioned dimensions in locality sensitive hashing

To solve the second problem, we should choose the partition dimensions d_i according to the distribution of values in that dimension. Densely distributed dimensions should be chosen with lower probability while dimensions with uniformly distributed values should be chosen with higher probability. In the example shown in Figure 3.6, it is better to partition the horizontal dimension compared to the vertical dimension.

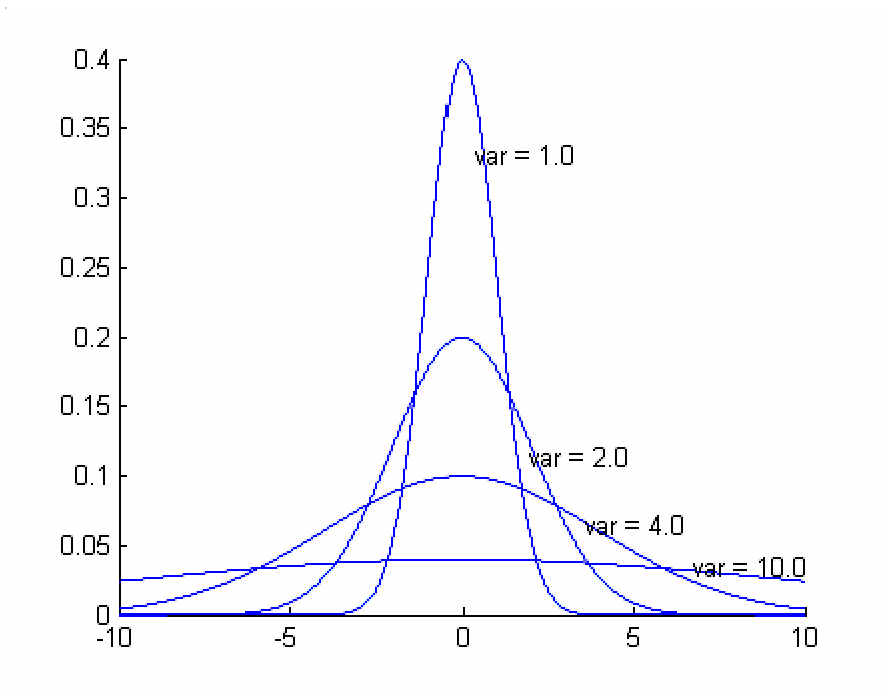


Figure 3.7 PDF of Gaussian distributions for different variances

We can prove that to reduce the probability that we miss the points whose distance is lower than the threshold ε, we should partition the dimensions whose values' distribution is closer to uniform distribution with higher probability. However, maintaining the distribution of every dimension is too costly. We choose to use the standard deviation σ as a criterion. Normally, for nearly unimodally distributed dataset, if the distribution of one dimension is close to uniform distribution, its variance is large. We give an example of Gaussian distributions for different variances in Figure 3.7. Therefore, we set the probability of selecting dimension j in proportion to the standard deviation σ of its distribution, i.e.

$$p\{choose\ j\} = \frac{\sigma_j}{\sum_{i=0}^{d-1} \sigma_i} \tag{3.3}$$

where the denominator is the sum of the standard deviation for all D dimensions, i.e.

$$\sigma_j = \sqrt{\sigma_j^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(p_{ij} - m_j\right)^2} \tag{3.4}$$

where m_j is the mean of values for all points on dimension j.

We can easily calculate the new standard deviation σ_j after the dataset is updated based on some statistical results from the old dataset. Firstly, we calculate and save 3 results for the old dataset: (i) Dataset size $size(P)$; (ii) Sum of the values for every dimension j, i.e. $sum(p)_j$; (iii) Sum of the square of the values for every dimension j, i.e. $sum(p^2)_j$. So

$$\sigma_j = \sqrt{\sigma_j^2} = \sqrt{E(p^2)_j - E^2(p)_j} = \sqrt{\frac{1}{N} sum(p^2)_j - (\frac{1}{N} sum(p)_j)^2} \tag{3.5}$$

After we add or remove some video clips (denoted as the changing dataset U) into or from the original dataset (denoted as the original dataset V), we get the new dataset W . We first calculate the above 3 results for the changing dataset U , then the 3 results for the new dataset W are just sums or differences of the 3 results for the original dataset V and the changing dataset U , i.e.

(i) $size(W) = size(V) \pm size(U)$ **(3.6)**

(ii) $sum_W(p)_j = sum_V(p)_j \pm sum_U(p)_j$ **(3.7)**

(iii) $sum_W(p^2)_j = sum_V(p^2)_j \pm sum_U(p^2)_j$ **(3.8)**

Thus, we can easily calculate the new standard deviation σ_j (via Equation 3.5) and the probabilities $P\{choose\ j\}$ for the new dataset W .

If the probabilities $P\{choose\ j\}$ of the new dataset W do not change much from the original dataset V , we can keep our hash tables; otherwise, we need to update the hash tables to keep the high accuracy and efficiency. However, it is difficult to dynamically update one hash table according to the changing part of the dataset, and it costs too much to rebuild all the hash tables. Thus, we should rebuild some hash tables while keep the other hash tables to maintain an acceptable performance. The number of the hash tables to be rebuilt is determined by the changing scale of the probabilities $P\{choose\ j\}$.

We call our improved LSH, "hierarchical, non-uniform locality sensitive hashing", or HNLSH. With this index structure, we can greatly reduce the search space and decrease the number of match operations for one query feature point. In Figure 3.8, we give an example to illustrate how to apply HNLSH for video identification. The 6th query feature point of the query clip (with bitstring HEX 774A3458) is mapped into bucket C342 in the first level hash table. Since the bucket C342 is full, it is rehashed to bucket 0010 in the second level hash table and we find 3 points in this bucket. We linearly check these 3 candidate points and the first candidate point satisfies the condition, i.e. the distance is lower than ε. Finally, we get a candidate position located in clip 2.

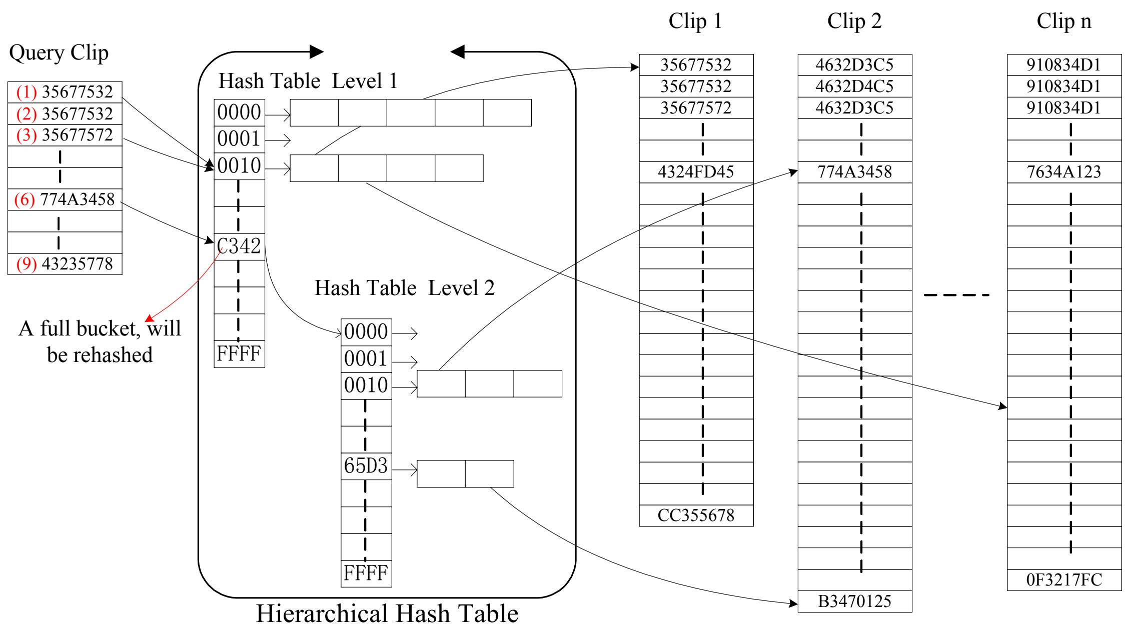


Figure 3.8 Illustration of HNLSH for video identification

3.4 Skip Redundant Match Operations by Triangle Inequality

From the above discussion, we can reduce search space for one query via HNLSH. In our problem formulation (see section 3.2), we have $l * frate'$ query points and the adjacent queries are similar, triangle inequality therefore can be employed to skip redundant match operations while keep the exactly same search result.

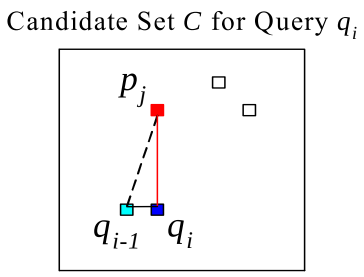


Figure 3.9 Skip redundant match operations by triangle inequality

To make it clear, we give an example in Figure 3.9. Suppose q_{i-1} and q_i are two adjacent queries searched on the points dataset $P=\{p_1,p_2,...,p_n\}$. We get the candidate search sets for q_{i-1} and q_i via HNLSH. Since q_{i-1} and q_i are similar, they are likely to be located in the same hashing bucket, resulted in the same points in their candidate search sets. That is to say, the dataset point p_j which needs to be checked for query q_i likely has already been checked by query q_{i-1}. Therefore, we compute the distance between q_{i-1} and q_i first, if the lower bound of the distance between q_{i-1} and p_j subtract the distance between q_{i-1} and q_i is still larger than the threshold ε, i.e. the lower bound of the distance between q_i and p_j is larger than the threshold ε, we need not to check the distance between q_i and p_j any more. Here is the mathematical verification: (from triangle inequality)

$$Dis(q_i,p_j)\geq Dis_{Low}(q_i,p_j)\triangleq Dis_{Low}(q_{i-1},p_j)-Dis(q_{i-1},q_i)\geq\varepsilon \quad \textbf{(3.9)}$$

We record the lower bounds of the previous query q_{i-1}, and update the lower bounds of the current query q_i, then iterate these operations for all queries. Here is a brief description for the algorithm:

```
for query $q_i$
    1. $Dis_{Low}(q_i, p_j) \leftarrow 0$,    $j = 1, 2, ..., n$;
    2. compute $Dis(q_{i-1}, q_i)$;
    3. get the candidate set $C$ of query $q_i$ via HNLSH;
    4. for any $p_j \in C$
        if $Dis_{Low}(q_{i-1}, p_j) - Dis(q_{i-1}, q_i) \geq \varepsilon$
        then $Dis_{Low}(q_i, p_j) \leftarrow (Dis_{Low}(q_{i-1}, p_j) - Dis(q_{i-1}, q_i))$;
                                                // no need to compute $Dis(q_i, p_j)$;
        else compute $Dis(q_i, p_j)$, if $Dis(q_i, p_j) \geq \varepsilon$
                                      then $Dis_{Low}(q_i, p_j) \leftarrow Dis(q_i, p_j)$;
                                      else output one answer $p_j$ for query $q_i$;
    5. repeat step 4 for all points in the candidate set $C$
```

Since q_{i-1} and q_i are similar, i.e. $Dis(q_{i-1}, q_i)$ is small, most of match operations are redundant and can be skipped. Therefore, triangle inequality can significantly reduce the number of total match operations for a batch of similar queries. This will be further proven in our tests.

3.5 Feature Extraction

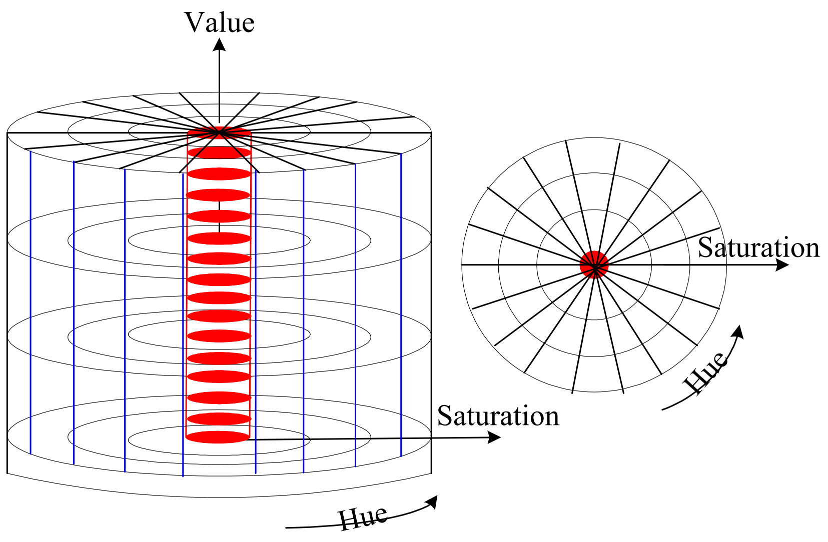


Figure 3.10 Quantization of the HSV color space

Color histogram is widely used in image/video retrieval applications. Color histogram intersection, which is equivalent to the L_1 distance [61], is proven to be an effective measurement for video identification [22]. We use the average color histogram as the feature for each l-second length video segment [39], i.e. the average of all the frames' histograms in one video segment. We represent each frame by three 178-bin color histograms on the Hue-Saturation-Value (HSV) color space. The quantization of the color space used in the histogram is shown in Figure 3.10, which is similar to the one used in [63, 15] with a slight change. The saturation (radial) dimension is uniformly quantized into 3.5 bins, with the half bin at the origin. The hue (angular) dimension is uniformly quantized at 20^{o}-step size, resulting in 18 sectors. The quantization for the value dimension (vertical) is a bit more complicated. In fact, the saturation and the hue dimensions do not make sense when the value dimension is very small. That is to say, the color is always black no matter what hue and saturation are. Similarly, the color is in gray-scale when saturation is small, i.e. hue is useless in this case. Therefore, when value dimension is small (<1/16) we will regard it as one bin for black color. For those colors with the saturation values near zero, a finer quantizer of 15 bins is used to better differentiate between gray-scale colors. For the rest of the color space, the value dimension is uniformly quantized into 3 bins. So the number of total bins for this quantization is 3 * 3 * 18 + 16 = 178. The histogram is normalized such that the sum of all the bins equals one. In order to incorporate spatial information into the frame feature, each frame is partitioned into three horizontal stripes, and each stripe has its own color histogram (shown in Figure 3.11). As a result, the number of total dimensions of one single feature vector becomes 178 * 3 = 534.

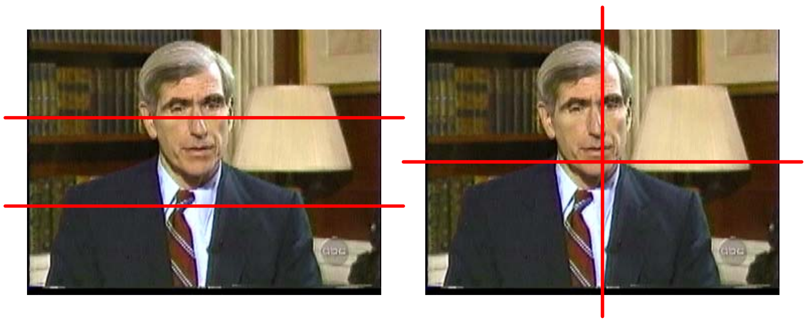


Figure 3.11 Frame partition

The reason we partition each frame into three horizontal stripes instead of four quadrants like in [15] is that we believe in image or video, the color information is more symmetric and homogenous on horizontal dimension than vertical dimension, i.e. the color variance on vertical dimension is larger than horizontal dimension. Incorporate spatial information in vertical dimension will make the features more discriminative. Another reason is that, in video, usually two consecutive frames have more changes on horizontal dimension than vertical dimension because there are far more pan operations than tilt operations in video recording. This may cause more temporal frame shift noise on horizontal dimension. Therefore, our histograms do not embed the spatial information on this dimension. The distance between two feature points is L_1 norm distance:

$$Dis(x,y) = \sum_{i=1}^{534} |x_i - y_i| \tag{3.10}$$

Chapter 4

Experimental Results and Discussion

In this chapter, we will first compare our improved LSH (HNLSH) with the original LSH. Then we will show promising system performance of our video identification scheme based on HNLSH and triangle inequality, and compare our method with NTT's "active search" [31].

4.1 Feature Dataset of the Video Database

We use 192 TREC2003 ABC and CNN news video clips to construct our video database, which are recorded from Feb.7, 1998 to Jun.25, 1998. Each video clip is about 30 minutes. So the total video length is 96 hours. All these video clips are in MPEG format, with frame size 352 * 264, frame rate 29.97 fps and bit-rate 1600k bps. We set the video segment length $l = 4$s, and therefore we need the query video clips to be longer than $2l = 8$s, which can be easily satisfied for ABC and CNN news and commercial video clips. Each video segment has 120 frames. We use the average of these 120 frames' HSV color histograms as the feature for this segment. The raw feature is 178 * 3 = 534-bin color histogram. We apply PCA on each 178-bin histogram to do dimension reduction respectively and keep 85% energy of the original data after we reduce the number of dimensions to 40 * 3 = 120. Finally, we get 81992 feature points with dimension 120 for this 96-hour video database.

4.2 Query Video Datasets

We randomly extract 40 video segments (including news and commercial videos, 20 from ABC and 20 from CNN) from the video database as the raw query video clips and all are equal or longer than 8s. We do some transformations and use these edited versions as our queries for the experiments.

Query dataset 1: We use this query dataset to compare our performance with NTT's "active search" algorithm [31], which is a quick search method searched for a large video database (48 hours). NTT's query videos are directly captured from real-time TV broadcasting using other TV recordings (in same conditions) which are different from their stored video. To simulate this, we generate our query videos by applying linear interpolation with random weights on the raw query videos, i.e.

$$F_{edt}(n) = \alpha F_{raw}(n) + (1-\alpha)F_{raw}(n+1) \quad \textbf{(4.1)}$$

where, $\alpha \in [0,1]$ is a random factor, $F_{raw}(n)$ is the nth frame of the raw query video, $F_{edt}(n)$ is the nth frame of the edited version.

Query dataset 2: We constructed the second query dataset using 3 kinds of transcodings, i.e. changes on frame size, frame rate and compression bit-rate. We use "Ulead VideoStudio 7" [64] to transcode the raw query videos from frame size 352 * 264, frame rate 29.97 fps, bit-rate 1600k bps to 320 * 240, 24 fps, 1200k bps to build this query dataset. We then use this query dataset to test the robustness of our system for different types of transcodings.

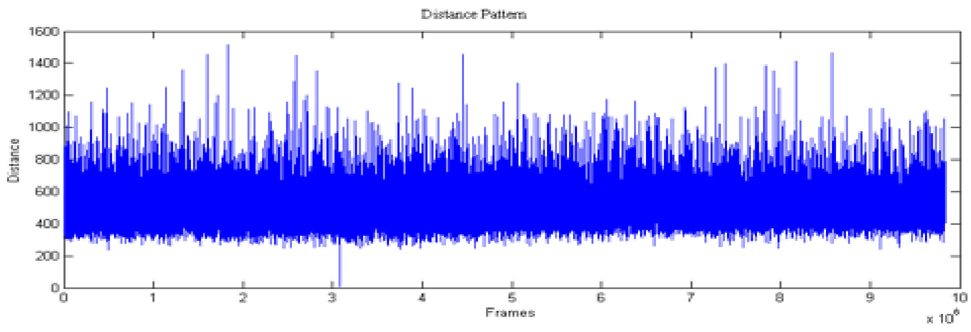


Figure 4.1 A distance pattern between the query video and the videos in database

(Note that there is a duplicated version located around 3*10^6)

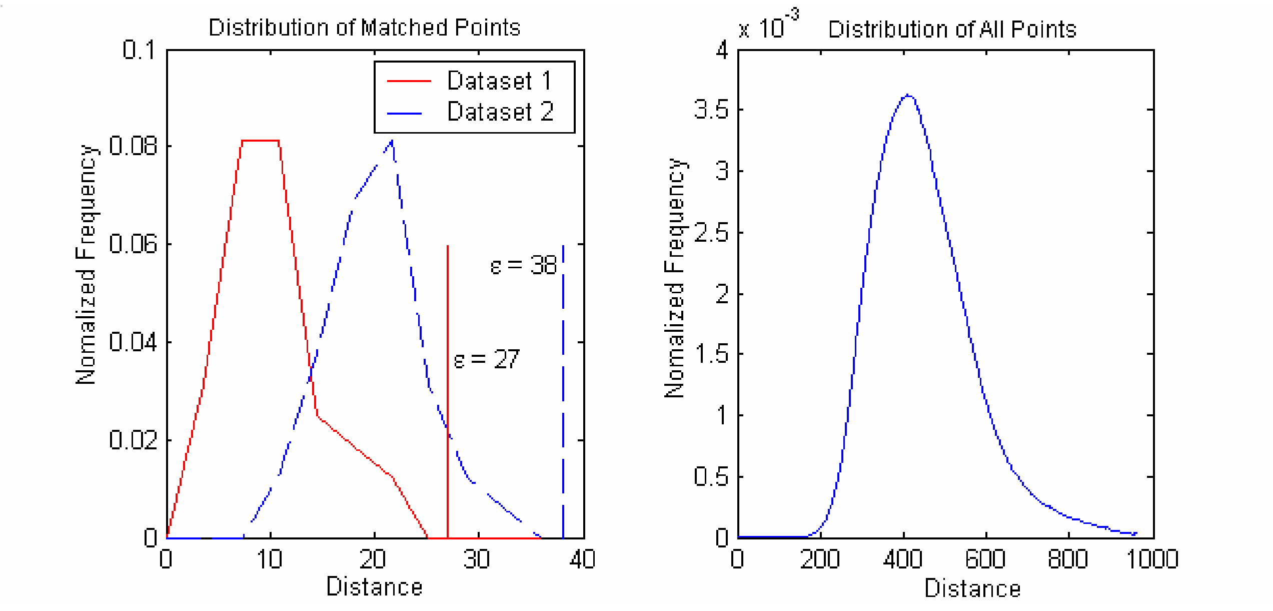


Figure 4.2 Distance distribution of the query video and the videos in database

Figure 4.1 gives a typical distance pattern between the query video in query dataset 1 and the videos in video database. The query dataset 2 has the similar distance pattern. In this figure, the horizontal axis is the time and the vertical axis is the distance. In the left figure of Figure 4.2, we present the distance distribution of the query videos and the duplicated version video segments in video database, i.e. the distance distribution of matched points; in the right figure, we present the distance distribution of the query videos and all the video segments in video database. We can see that the distances of matched points are all lower than 40 and nearly all the distances of unmatched points are larger than 200. The gap can be set the threshold ε is quite large, which means our feature is robust and discriminative.

The threshold ε should be larger than all the distances of the duplicated version videos to avoid missing the correct detection. However, the speed performance of HNLSH and triangle inequality will be decreased if we choose a too large threshold. To get a reasonable choice for the threshold ε, we suppose the distances between the query videos and the duplicated version videos have Gaussian distribution (left figure in Figure 4.2) and let the threshold $\varepsilon > \mu + 3\sigma$ based on the 3σ rule. Therefore, we set $\varepsilon = 27$ for the query dataset 1 and $\varepsilon = 38$ for the query dataset 2.

4.3 Performance of HNLSH

In [20], we have tested that HNLSH works better than original LSH on approximate nearest neighbor search problem for a video dataset. To evaluate the performance of HNLSH on ε-range search, we randomly create 200 queries that for any query q, there is a point p in the dataset whose distance from q is ε, i.e. $Dis(q,p) = \varepsilon$. We get the candidate set C of query q, if p is not in this candidate set, i.e. $p \notin C$, we regard it as a miss. Here, we choose $\varepsilon = 38$ for the case of query dataset 2. The size of candidate set C, i.e. the number of points to be searched, is used as a measurement for efficiency. We apply the original LSH plus the improvements on the above 81992 points feature dataset, each indexing structure consists of 4 hash tables ($N = 4$). For original LSH, the number of hashed dimensions k is varied from 10 to 30. For hierarchical LSH, the maximum hashing level is 4, and the number of hashed dimensions for each level k is varied from 6 to 15. The rehash threshold is decided by the maximum hashing level and hashed dimensions for each level k to let the LSH can hold the whole dataset. We build each kind of hash tables 200 times and the performance is measured by average.

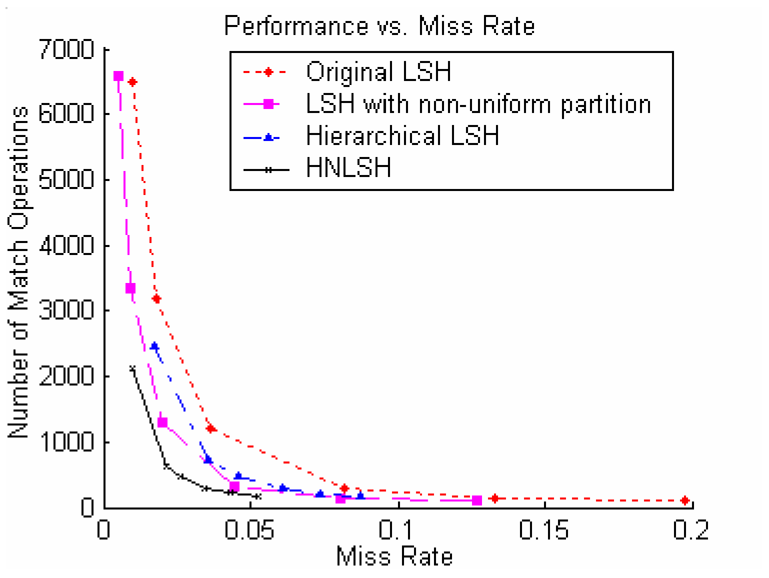


Figure 4.3 Performance of HNLSH

We compare the candidate set size (i.e. number of match operations) and the miss rate for 4 different implementations of LSH: the original, LSH with non-uniform selection of partitioned dimensions, LSH with hierarchical partitioning, and LSH with both improvements combined (HNLSH). Figure 4.3 shows our results. Compared with LSH, HNLSH is much better in terms of both performance and accuracy.

Although our feature dataset is only 40M bytes and can be fit in the main memory, someone may still interested in the I/O cost, i.e. number of page access. In fact, for one hash table of HNLSH, the number of points in the return bucket is always less than the rehash threshold. Therefore the I/O cost is predictable even for the worst case. For example, in the above experiment, if the maximum hashing level is 4 and the number of hashed dimensions for each level $k = 10$, then the rehash threshold is 320. So the return bucket size of one hash table is less than 320 for any query points. Suppose each page contains 40 points, the page access for this hash table will be 8 pages. Therefore the total page access will not exceed 32 pages for 4 hash tables.

Now we wish to get a suitable choice for the number of hash tables N for HNLSH. We fix the number of hashed dimensions for each level $k=10$ in hierarchical hashing and vary N to get the miss rate. In theory, the miss rate will be dropped exponentially with the increase of the number of hash tables N, i.e. the miss rate is r^N for N hash tables. Therefore, when the threshold ε changed, we can use fewer hash tables to increase the performance or build more hash tables to maintain the low miss rate without changing the existent hash tables. Table 4.1 shows the miss rate in experiment and theory for both query datasets. The results of simulation verified that the miss rate will be dropped exponentially. However, the simulation consistently gives higher miss rate than the theoretical analysis, because we can not get the miss rate in theory for 1 table, so we use the simulation value instead. Therefore, the error of the miss rate will be accumulated for the cases of 2 tables, 4 tables, 6 tables, etc.

Table 4.1 Number of hash tables *N* vs. miss rate

	Query dataset 1 ($\varepsilon=27$)		Query dataset 2 ($\varepsilon=38$)	
Tables N	Miss Rate (Experiment)	Miss Rate (Theory)	Miss Rate (Experiment)	Miss Rate (Theory)
1 table*	31.8628% r	------ r	41.3857% r	------ r
2 tables	10.5925%	10.152% r^2	18.1075%	17.1278% r^2
4 tables	1.2775%	1.0307% r^4	3.44%	2.9336% r^4
6 tables	0.19%	0.10464% r^6	0.76%	0.50246% r^6
8 tables	0.025%	0.010624% r^8	0.17%	0.08606% r^8
10 tables	0.0025%	0.001079% r^{10}	0.035%	0.01474% r^{10}
12 tables	0	0.000110% r^{12}	0.0075%	0.002525% r^{12}

*For the case of 1 hash table, we build HNLSH 2000 times to get more reliable result (represented by r) and use it for theory analysis. For the other cases, we build HNLSH 200 times.

The miss rate for the points within the distance ε from the query is definitely less than the miss rate listed in Table 4.1. Therefore, we can get very low miss rate for ε-range search both in query dataset 1 ($\varepsilon=27 \quad N=10$) and query dataset 2 ($\varepsilon=38 \quad N=12$). We will present this result in the next experiment.

4.4 Performance of Video Identification

We will test the performance of HNLSH for the real query dataset, i.e. query dataset 1 and query dataset 2 created above, which totally including 40 query clips. We also employ triangle inequality on the return candidate set C of HNLSH, to skip redundant match operations. The maximum hashing level is 4; the number of hashed dimensions for each level $k = 10$ and the rehash threshold is 320. We vary the number of hash tables N and show the total number of match operations for one query video clip vs. miss rate in Figure 4.4. Each set of hash tables are built 200 times and the performance is measured by average.

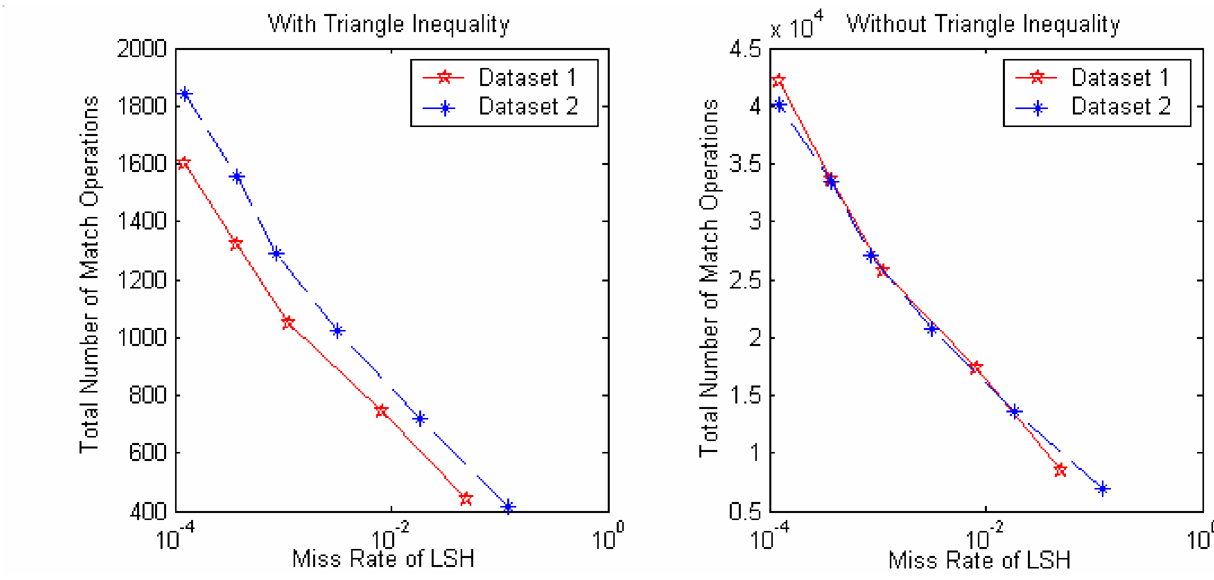


Figure 4.4 Performance of video identification

For the real query video dataset, the miss rate is much lower than the miss rate of randomly created query dataset which listed in Table 4.1, because the distances of the real query points are much lower than the threshold ε. In fact, the miss rate of HNLSH dropped to 0 when we use more than 6 hash tables ($N = 6$) for query dataset 1 and 7 hash tables ($N = 7$) for query dataset 2. Compared the left figure and right figure in Figure 4.4, we can see that triangle inequality significantly reduces lots of redundant match operations. The number of total match operations is reduced to 4% with triangle inequality.

Table 4.2 Summary of the performance for video identification

Query Dataset	Th ε	Tables N	Miss Rate	Before Triangle Inequality	Total Match Operations	Reduction Ratio	Time (s)
1	27	10	<0.0025%	83240	2970	1/3312	0.158
2	38	12	<0.0075%	78803	3434	1/2292	0.170

In Table 4.2, we give a summary of the system performance for video identification. To be conservative, we still use 10 hash tables for query dataset 1 ($N = 10$) and 12 hash tables for query dataset 2 ($N = 12$). The miss rate of HNLSH is low enough and can be ignored in practice, because in theory we always have miss probability for video identification if we set a high threshold ε. Refer to Table 4.2, the number of total match operations of a linear search algorithm for query dataset 1 and query dataset 2 are 4s * 30fps * 81992 = 9839040 and 4s * 24fps * 81992 = 7871232 respectively. Compared with those linear search algorithms, the reduction ratios of total match operations of our algorithm are 2970/9839040 = 1/3312 and 3434/7871232 = 1/2292 respectively. Note that, besides HNLSH, triangle inequality also plays an important role in our system. For example, for query dataset 1, HNLSH reduce the number of total match operations from 9839040 to 83240, where the reduction ratio is 1/118, and triangle inequality further reduce the number of total match operations from 83240 to 2970, where the reduction ratio is 1/28. We run these experiments on the Pentium 4, 1.7GHz machine with 384MB memory. We build the HNLSH for the feature dataset and extract features from the query video offline. Once the features of the query video extracted, the total running time is 0.158s for query dataset 1 and 0.170s for query dataset 2.

4.5 Comparison with NTT's "Active Search"

We compare our algorithm with NTT's "active search" algorithm [31], which is a quick search method for video signals. The query dataset 1 is a simulation for NTT's query dataset. The comparison results are shown in Table 4.3. Our video database is twice as large as NTT's database, which is 96 hours. Since we move the compare window on the query, the query video clip has to be longer than 2 segments, i.e. $2l = 8$s. For accuracy, NTT's "active search" insures the exactly same search result as the linear search, whereas HNLSH is an approximate search method with the miss rate near to 0, and its miss rate can be decreased exponentially with more hash tables. In fact, all the hash tables can be implemented in a parallel way on distributed systems because they are independent. On the other hand, we gain 11 times improvement on the speed performance. Note that the running time they reported is only the CPU time (i.e., excluding I/O time) while ours is the total time because our dataset can be fit in the main memory.

Table 4.3 Comparison of our algorithm and NTT's "active search"

Methods	Database Size	Query Length	Miss Rate	Total Match Operations	Reduction Ratio	Time (s)
NTT's	48h	≥ 4s	0	17793	1/291	0.17 (CPU)
Ours	96h	≥ 8s	<0.0025%	**2970**	**1/3312**	**0.158 (total)**

Chapter 5

Conclusions and Future Work

5.1 Conclusions

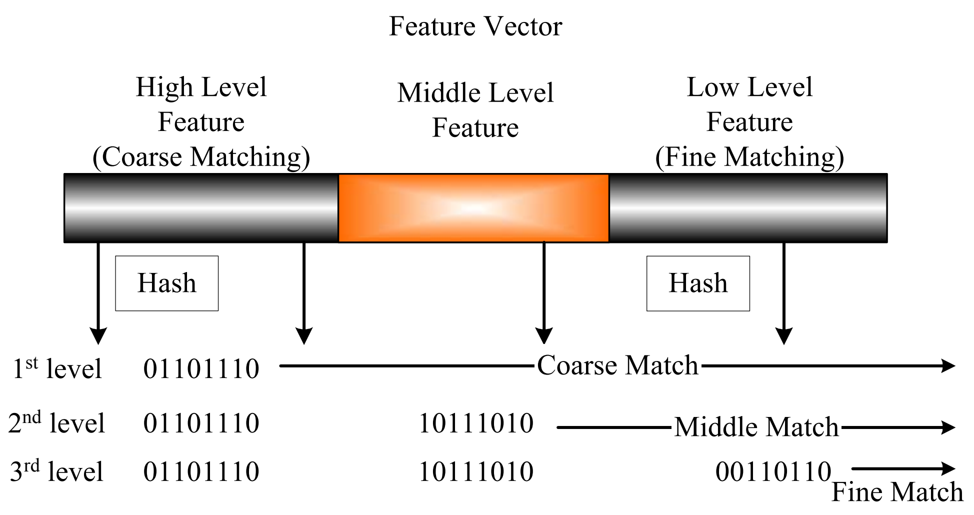
The problem of content-based video identification concerns identifying the duplicated version of a given short query video clip in a large video database based on content similarity. In this thesis we first divide video identification problems into 6 levels based on the noise between the original and the duplicated version video clips. High level (semantical level) and low level (exact match level) video identification systems have different concerns on search speed. Our task is to build a highly efficient video identification system which is robust to transcoding level noise, i.e. different frame size, frame rate and compression bit-rate.

In our video identification system, to save the storage of the video database's feature dataset, we slide the search window on the query video rather than the videos in the database. Thus, we can fit the whole feature dataset into the main memory to save additional I/O cost. We introduce locality sensitive hashing (LSH), and explain why it is suitable for video hashing to achieve highly efficient video identification. We improve the original version of LSH on two weaknesses to make it more suitable for video datasets. Triangle inequality is employed on the resulted buckets by improved LSH to further enhance the system efficiency.

Experimental results verify that our improved LSH is much better than original LSH in terms of both efficiency and accuracy when applied on the video feature data-set for similarity search. For video identification, we greatly reduce the search space and redundant match operations by incorporating improved LSH with triangle inequality to improve the efficiency. We further demonstrate the promising system performance by comparing our algorithm with NTT's "active search" algorithm. The use of LSH with triangle inequality and sliding search window on the query video are two main contributions of this research work.

5.2 Future Work

Our video identification system represents the query video and the videos in database as feature vector sequence, and then does feature vectors matching. We use the average color histogram as the feature vector. To build the hierarchical hash tables, we choose some bins of the color histogram for the first level hashing and then some other bins for the second level hashing. We believe that more improvements could be achieved if we can find some *hierarchical features* to incorporate with our hierarchical hash tables. *Hierarchical features* are the feature vectors whose first dimensions stand for high level features for coarse matching and last dimensions stand for low level features for fine matching. Incorporating *hierarchical features* with hierarchical hash tables, we can do first level hashing using high level features and second level hashing using low level features accordingly. Thus, we do coarse search first to greatly reduce the search space and then do fine search if necessary. Figure 5.1 gives an example.



Feature Vector
High Level Feature (Coarse Matching)
Middle Level Feature
Low Level Feature (Fine Matching)
Hash
Hash
1st level 01101110
Coarse Match
2nd level 01101110 10111010
Middle Match
3rd level 01101110 10111010 00110110
Fine Match

Figure 5.1 Incorporate hierarchical feature vectors with hierarchical hash tables

As mentioned before, how to dynamically update one hash table in the HNLSH is another problem for the future research. Since the performance of HNLSH will be decreased with the changes of the dataset distribution, while it cost too much to rebuild a hash table, therefore, how to maintain an acceptable high performance of the hash table for the new dataset without rebuilding it will be an interesting problem.

For special domain video search, such as news video search, "abstraction" and "structure analysis" are helpful for video indexing. Some domain knowledge (e.g. subtitle, anchor person, lead-in for news video) will benefit every process in the system. Figure 5.2 shows the overall system.

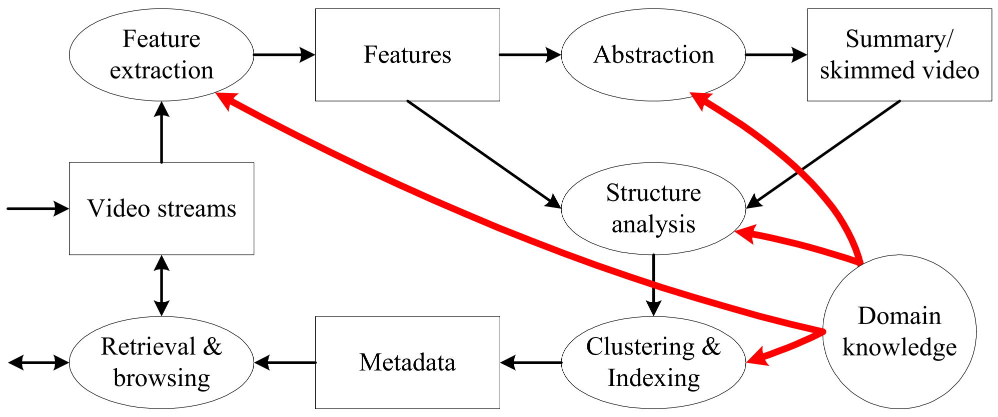


Figure 5.2 Process diagram for special domain video indexing

Bibliography

[1] N. Dimitrova, H. J. Zhang, B. Shahraray, M. I. Sezan, T. Huang and A. Zakhor, "Applications of video-content analysis and retrieval," in *IEEE Multimedia*, vol.9, no.3, pp.42-55, Jul-Sep 2002.

[2] H. D. Wactlar, T. Kanade, M. A. Smith and S. M. Stevens, "Intelligent access to digital video: Informedia project," in *IEEE Computer*, vol.29, no.3, pp.46-52, May 1996.

[3] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang and B. Dom, "Query by image and video content: The QBIC system," in *IEEE Computer*, vol.28, no.9, pp.23-32, 1995.

[4] H. J. Zhang, C. Y. Low, S. W. Smoliar and J. H. Wu, "Video parsing, retrieval and browsing: An integrated and content-based solution," in *Proceedings of ACM Multimedia*, Nov 1995, San Francisco, CA.

[5] A. K. Jain, A. Vailaya and W. Xiong, "Query by video clip," in *Multimedia Systems*, vol.7, issue.5, pp.369-384, Sep 1999.

[6] R. Lienhart, W. Effelsberg and R. Jain, "VisualGREP: A systematic method to compare and retrieve video sequences," in *SPIE, Storage and Retrieval for Image and Video Database VI*, vol.3312, Jan 1998, San Jose, CA.

[7] Y. P. Tan, S. R. Kulkarni and P. J. Ramadge, "A framework for measuring video similarity and its applications to video query by example," in *Proceedings of International Conference on Image Processing*, Oct 1999, Kobe, Japan.

[8] G. I. P. Indyk and N. Shivakumar, "Finding pirated video sequences on the internet," in *Stanford Infolab Technical Report*, Feb 1999.

[9] E. Y. Chang, J. Z. Wang, C. Li and G. Wiederhold, "RIME: A replicated image detector for the world-wide web," in *Proceedings of SPIE, Symposium of Voice, Video, and Data Communications*, pp.58-67, Nov 1998, Boston, MA.

[10] A. Hampapur and R. M. Bolle, "Feature based indexing for media tracking," in *Proceedings of International Conference on Multimedia and Expo*, Aug 2000, New York, NY.

[11] A. Hampapur, R. M. Bolle, "VideoGREP: Video copy detection using inverted file indices," *Technical Report*, IBM Research.

[12] A. Hampapur, K. Hyun and R. Bolle, "Comparison of sequence matching techniques for video copy detection," in *SPIE, Storage and Retrieval for Media Databases*, vol.4676, pp.194-201, Jan 2002, San Jose, CA.

[13] T. C. Hoad and J. Zobel, "Fast video matching with signature alignment," in *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pp.263-269, 2003, Berkeley, CA.

[14] T. C. Hoad and J. Zobel, "Video similarity detection for digital rights management," in *Proceedings of Australasian Computer Science Conference*, pp.237-245, 2003, Adelaide, Australia.

[15] S. C. Cheung and A. Zakhor, "Efficient video similarity measurement with video signature," in *IEEE Transaction on Circuits and System for Video Technology*, vol.13, no.1, pp.59-74, 2003.

[16] S. C. Cheung and A. Zakhor, "Fast similarity search and clustering of video sequences on the world-wide-web," in *IEEE Transactions on Multimedia*, vol.7, no.3, pp.524-537, 2005.

[17] J. Oostveen, T. Kalker and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems*, pp.117-128, Mar 2002, Hsin Chu, Taiwan.

[18] A. Joly, C. Frelicot and O. Buisson, "Robust content-based video copy identification in a large reference database," in *Proceedings of International Conference on Image and Video Retrieval*, pp.414-424, Jul 2003, Urbana-Champaign, IL.

[19] A. Joly, C. Frelicot and O. Buisson, "Feature statistical retrieval applied to content-based copy identification," in *Proceedings of International Conference on Image Processing*, pp.681-684, Oct 2004, Singapore.

[20] Zixiang Yang, Wei Tsang Ooi and Qibin Sun, "Hierarchical, non-uniform locality sensitive hashing and its application to video identification," in *Proceedings of International Conference on Multimedia and Expo*, Jun 2004, Taipei, Taiwan.

[21] R. Lienhart, C. Kuhmünch and W. Effelsberg, "On the detection and recognition of television commercials," in *Proceedings of International Conference on Multimedia Computing and Systems*, pp.509 - 516, Jun 1997, Ottawa, Canada.

[22] M. Y. M. Naphade and B. L. Yeo, "A novel scheme for fast and efficient video sequence matching using compact signatures," in *SPIE, Storage and Retrieval for Media Databases*, vol.3972, pp.564–572, Jan 2000, San Jose, CA.

[23] L. P. Chen and T. S. Chua, “A match and tiling approach to content-based video retrieval,” in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp.417-420, Aug 2001, Tokyo, Japan.

[24] J. M. Sanchez and X. Binefa, “Shot partitioning based recognition of TV commercials,” in *IEEE Transactions on Multimedia Tools and Applications*, vol.18, no.3, pp.233-247, 2002.

[25] V. Kulesh, V. Petrushin and I. Sethi, “Video clip recognition using joint audio-visual processing model,” in *Proceedings of International Conference on Pattern Recognition*, pp. 500-503, Aug 2002, Quebec City, Canada.

[26] N. Diakopoulos and S. Volmer, “Temporally tolerant video matching,” in *SIG IR Multimedia Information Retrieval Workshop*, Aug 2003, Toronto, Canada.

[27] E. Kasutani and A. Yamada, “An adaptive feature comparison method for real-time video identification,” in *Proceedings of International Conference on Image Processing*, vol.2 , pp.5-8, Sep 2003, Barcelona Spain.

[28] X. S. Hua, X. Chen and H. J. Zhang, “Robust video signature based on ordinal measure,” in *Proceedings of International Conference on Image Processing*, pp.685-688, Oct 2004, Singapore.

[29] K. M. Pua , J. M. Gauch , S. E. Gauch and J. Z. Miadowicz, “Real time repeated video sequence identification,” in *Computer Vision and Image Understanding*, vol.93, no.3, pp.310-327, Mar 2004.

[30] A. Kimura, K. Kashino, T. Kurozumi and H. Murase, “A quick search method for multimedia signals using feature compression based on piecewise linear maps,” in *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, vol.4, pp.3656-3659, May 2002, Orlando, Florida.

[31] K. Kashino, T. Kurozumi and H. Murase, “A quick search method for audio and video signals based on histogram pruning,” in *IEEE Transactions on Multimedia*, vol.5, no.3, pp.348-357, Sep 2003.

[32] K. Kashino, A. Kimura and T. Kurozumi, “A quick video search method based on local and global feature clustering,” in *Proceedings of International Conference on Pattern Recognition*, Aug 2004, Cambridge, UK.

[33] J. Yuan, Q. Tian, Ranganath and S. Ranganath, “Fast and robust search method for short video clips from large video collection,” in *Proceedings of International Conference on Pattern Recognition*, Aug 2004, Cambridge, UK.

[34] J. Yuan, L. Y. Duan, Q. Tian and C. Xu, “Fast and robust short video clip search using an index structure,” in *ACM Multimedia Workshop on Multimedia Information Retrieval*, pp.61-68, Oct 2004, New York, NY.

[35] L. Y. Duan, M. Xu , Q. Tian and C. Xu, “Mean shift based video segment representation and applications to replay detection,” in *Proceedings of International Conference on Acoustic, Speech, and Signal Processing*, May 2004, Montreal Quebec Canada.

[36] D. Q. Zhang and S. F. Chang, “Detecting image near-duplicate by stochastic attributed relational graph matching with learning,” in *Proceedings of ACM Conference of Multimedia*, pp.877-884, Oct 2004, New York, NY.

[37] R. Mohan, “Video sequence matching,” in *Proceedings of the International Conference on Audio, Speech and Signal Processing*, vol.6, pp.3697-3700, May 1998, Seattle, Washington.

[38] S. H. Kim, R. H. Park, “An efficient algorithm for video sequence matching using the modified Hausdorff distance and the directed divergence,” in *IEEE*

Transactions on Circuits and Systems for Video Technology, vol.12, pp.592-596, Jul 2002.

[39] A. M. Ferman, A. M. Tekalp and R. Mehrotra, "Robust color histogram descriptors for video segment retrieval and identification," in *IEEE Transaction on Image Processing*, vol.11, no.5, pp.497-508, May 2002.

[40] D. DeMenthon and D. Doermann, "Video retrieval using spatio-temporal descriptors," in *Proceedings of ACM Conference of Multimedia*, pp.508-517, Nov 2003, Berkeley, CA.

[41] C. Y. Cho, Y. T. Chuang, P. C. Chu, S. Y. Huang and J. S. Wang, "Efficient motion-vector-based video search using query by clip," in *Proceedings of International Conference on Multimedia and Expo*, Jun 2004, Taipei, Taiwan.

[42] D. Zhong and S. F. Chang, "An integrated approach for content-based video object segmentation and retrieval," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol.9, no.8, pp.1259-1268, Dec 1999.

[43] C. Bohm, S. Berchtold, and D.A.Keim, "Searching in high-dimensional space: Index structures for improving the performance of multimedia databases," in *ACM Computing Surveys*, vol.33, no.3, pp.322-373, Sep 2001.

[44] E. Tuncel, H. Ferhatosmanoglu and K. Rose, "VQ-index: An index structure for similarity searching in multimedia databases," in *Proceedings of ACM Multimedia*, pp.543-552, Dec 2002, Juan Les Pins, France.

[45] F. Idris and S. Panchanathan, "Review of image and video indexing techniques," in *Journal of Visual Communication and Image Representation*, vol.8, no.2, pp.146-166, Jun 1997.

[46] G. Ahanger and T. D. C. Little, “A survey of technologies for parsing and indexing digital video,” in *Journal of Visual Communication and Image Representation*, vol.7, no.1, pp.28-43, Mar 1996.

[47] Y. A. Aslandogan and C. T. Yu, “Techniques and systems for image and video retrieval,” in *IEEE Transactions on Knowledge and Data Engineering*, vol.11, no.1, pp.56-63, Jan 1999.

[48] H. Wang, A. Divakaran, A. Vetro, S. F. Chang and H. Sun, “Survey of compressed-domain features used in audio-visual indexing and analysis,” in *Journal of Visual Communication and Image Representation*, vol.14, no.2, pp.150-183, Jun 2003.

[49] Y. Rui, T. S. Huang and S. F. Chang, “Image retrieval: Past, present and future,” in *Proceedings of International Symposium on Multimedia Information Processing*, Dec 1997, Taipei, Taiwan.

[50] A. Hampapur and R. Bolle, “Comparison of distance measures for video copy detection,” in *Proceedings of International Conference on Multimedia and Expo*, Aug 2001, Tokyo, Japan.

[51] S. W. Smoliar and H. J. Zhang, “Content-based video indexing and retrieval,” in *IEEE Multimedia*, vol.2, no.1, pp.63-75, Summer 1994.

[52] J. T. Robinson, “The K-D-B-tree: A search structure for large multi-dimensional dynamic indexes,” in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.10-18, Apr 1981, Ann Arbor, Michigan.

[53] A. Guttman, “R-trees: A dynamic index structure for spatial searching”, in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.47-57, Jun 1984, Boston, Massachusetts.

[54] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, “The R*-tree: An efficient and robust access method for points and rectangles,” in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.322-331, May 1990, Atlantic City, NJ.

[55] K. Lin, H. V. Jagadish and C. Faloutsos, “The TV-tree: An index structure for high-dimensional data,” in *VLDB Journal*, vol.3, no.4, pp.517-542, 1994.

[56] S. Berchtold, D. A. Keim and H. P. Kriegel, “The X-tree: An index structure for high-dimensional data,” in *Proceedings of the 22nd International Conference on Very Large Databases*, pp.28-39, Sep 1996, Bombay, India.

[57] D. A. White and R. Jain, “Similarity indexing with the SS-tree,” in *Procceedings of the 12th International Conference on Data Engineering*, pp.516-523, Feb 1996, New Orleans, Louisiana.

[58] N. Katayama and S. Satoh, “The SR-tree: An index structure for high-dimensional nearest neighbor queries,” in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.369-380, May 1997, Tucson, Arizona.

[59] R. Weber, H. J. Schek and S. Blott, “A quantitative analysis and performance study for similarity search methods in high-dimensional spaces,” in *Proceedings of the 24th International Conference on Very Large Data Bases*, pp.194-205, Aug 1998, New York City, New York, USA.

[60] A. Gionis, P. Indyk and R. Motwani, “Similarity search in high dimensions via hashing,” in *Proceedings of International Conference on Very Large Data Bases*, pp.518-529, Sep 1999, Edinburgh, Scotland.

[61] M. J. Swain and D. H. Ballard, "Color indexing," in *International Journal of Computer Vision*, vol.7, no.1, pp.11-32, 1991.

[62] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. E. Abbadi, "Vector approximation based indexing for non-uniform high dimensional data sets," in *Proceedings of ACM CIKM International Conference on Information and Knowledge Management*, Nov 2000, Washington, DC, USA.

[63] J. Smith and S. F. Chang, "Tools and techniques for color image retrieval," in *Proceedings of SPIE Storage and Retrieval for Image and Video Databases IV*, vol.2670, pp.426-437, Jan-Feb 1996, San Diego/La Jolla, CA.

[64] http://www.ulead.com/