Content-based Image Indexing and Retrieval for Visual Information Systems

Wing Wah Simon So

This thesis is presented in fulfilment of

the requirements for the degree of

Doctor of Philosophy

Computer and Mathematical Sciences

School of Communications and Informatics

Faculty of Engineering and Science

Victoria University of Technology

2000

Dedicated to the memory of my father, Yuk Sang So.

The death of my father at the start of my Ph.D. study is a tremendous loss to me. He was truly an indefatigable father who devoted all his life to the well being of his children. His perseverance and tenacity attribute to my success. I owe my life to him for all his love and sacrifices. He is my idol of a perfect father.

Dad, I miss you deeply !

Abstract

The dominance of visual data in recent times has made a fundamental change to our everyday life. Less than five to ten years ago, Internet and World Wide Web were not the daily vocabulary for the general public. But now, even a young child can use the Internet to search for information. This, however, does not mean that we have a mature technology to perform visual information search. On the contrary, visual information retrieval is still in its infancy. The problem lies on the semantic richness and complexity of visual information in comparison to alphanumeric information.

In this thesis, we present new paradigms for content-based image indexing and retrieval for Visual Information Systems. The concept of *Image Hashing* and the developments of *Composite Bitplane Signatures* with *Inverted Image Indexing and Compression* are the main contributions to this dissertation. These paradigms are analogous to the signature-based indexing and inversion-based postings for text information retrieval. We formulate the problem of image retrieval as a two-dimensional hashing as oppose to a one-dimensional hash vector used in conventional hashing techniques. Wavelets are used to generate the bitplane signatures. The natural consequence to our bitplane signature scheme is the superimposed bitplane signatures for efficient retrieval. Composite bitplanes can then be used as the low-level feature information together with high-level semantic indexing to form a unified and integrated framework in our inverted model for content-based image retrieval.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university. The material presented in this thesis is the product of the author's own independent research under the supervision of Professor Clement H. C. Leung.

Some of the materials presented in this thesis has been published in various publications. The thesis is less than 100,000 words in length.

Wing Wah Simon So

August 2000

A List of External Publications

1. C. H. C. Leung, S. So, A. Tam, D. Sutanto, P. Tse, "Visual Information Indexing for Content-Based Search and Retrieval," in Michael Lew (Ed.), *Multimedia Search: State of the Art* (To appear)

2. C. H. C. Leung and W. W. S. So, "Visual Information Systems," in Borko Furht (Ed.), *Handbook of Internet & Multimedia Systems & Applications*, Chapter 16, Boca Raton FL: CRC Press LLC, 1999, pp.361-374

3. Simon So and Clement Leung, "A New Paradigm in Image Indexing and Retrieval using Composite Bitplane Signatures," *IEEE International Conference on Multimedia Computing and Systems*, Florence, Vol. I, 7-11 June 1999, pp.855-859

4. Simon So, Clement Leung, and Philip Tse, "A Comparative Evaluation of Algorithms Using Compressed Data for Image Indexing and Retrieval," *International Conference on Computational Intelligence and Multimedia Applications*, Churchill, 9-11 February 1998, pp.866-872

5. W. W. S. So, and C. H. C. Leung, "Inverted Image Indexing and Compression," *SPIE Proc. Multimedia Storage and Archiving Systems II,* Dallas, Texas, Vol. 3229, 3-4 November 1997, pp.254-263

6. C. H. C. Leung and W. W. S. So, "Characteristics and Architectural Components of Visual Information Systems," in Clement H.C. Leung

(Ed.), *Lecture Notes in Computer Science*, Berlin Heidelberg: Springer-Verlag, Vol 1306, 1997, pp.1-12

7. W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Analysis and Evaluation of Search Efficiency for Image Databases," in Arnold Smeulders and Ramesh Jain (Eds.), *Series on Software Engineering and Knowledge Engineering*, Vol. 8, Singapore: World Scientific, 1997, pp.253-262

8. W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Search Space Reduction Strategies for Image Databases," *Proceedings of the First International Workshop on Image Databases and Multimedia Search*, Amsterdam, The Netherlands, August 22-23 1996, pp.179-186

9. W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "A Multi-paradigm Approach to Visual Information Systems Design," *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems,* Hong Kong, June 27-28 1996, pp.265-273

10. W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Picture Coding For Image Database Retrieval," *International Picture Coding Symposium,* Melbourne, March 1996, pp.69-74

Acknowledgments

My supervisor, Professor Clement H. C. Leung, deserves the utmost appreciation and warmest gratitude for providing guidance in all aspects which enable me to accomplish this work. Most importantly, he has always encouraged and believed in me. Many good ideas are nourished in this environment. Undoubtedly, he has provided me with a model of what a prominent professor and an excellent research supervisor should be.

I am grateful to the generous financial support provided by a departmental scholarship from the School and special thanks go to the Head of School, Associate Professor Neil Barnett, for being insisted on research support even in difficult financial constraints for recent years.

I have been very happy to be associated with many members of the Visual Information Systems Group, both past and present. To the staffs of Computer and Mathematical Sciences, I like to thank them for many hours of inspirational discussion and interaction. Special thanks go to Dr. Audrey Tam, Dr. Stephen Young, Kinh Nguyen, Ted Alwast, Tim Hunt, Dr. Yi Jiang, Grace Tan, Associate Professor Peter Cerone and many others. Philip Tse, Dr. Kevin Liu, Dr. David Taniar, Dwi Sutanto, Dr. Savitri Bevinakoppa, Dr. Pak Fai Tang, Ray Summit and Refyul Fatri have been a wonderful source of friendship and technical interaction. I would also like

to thank Emily Wark, a cheerful inter-library loans officer, for her efficient help to many of my requests. She is really my "window" of knowledge to the outside world.

For many of my personal friends and former colleagues from the University of Hong Kong living in Melbourne, I thank their company for nearly ten years. Weekend outings, dinners, and many activities enable me to leave my research problems behind and enjoy life a bit more. For Dr. T. Y. Chen, my former colleague at the University of Hong Kong and supervisor at the University of Melbourne, I cannot accord him with the customary gratitude of having been so supportive and caring. He has all my respects.

Finally, I would attempt the impossible task to express my gratitude to my family. Four years and two sons later, as I complete this work, I wholeheartedly thank my beloved wife, Shuk Fong, for her patience and understanding. She deserves an accolade of a honorary degree for putting up with everything; to bear the brunt of giving up a good life-style and sacrifice with me for my own personal goal, to forgive my dereliction of many duties as a father and a husband, and to endure a long period of uncertainty etc. For more reasons than there is room in these notes to list, I owe a debt of gratitude to her.

For my children, ABC (Alberta, Brian and Conrad), they have played a role in support of my research. Writing a paper for the deadline at the delivery room was indeed memorable. Notwithstanding the pressure and anxiety from both sides, it gave me a new perspective and balanced view of life. Also, for a house full of laughter and babyish cry, they stimulated my thoughts and prodded me occasionally to hurry up my study.

I thank my mother, Siu Kam, for her moral support. I am certain that her silent blessings from the other side of the hemisphere pay off. To my extended family, I thank my mother-in-law, Kwan Foon Wu, for her help during her stay in Melbourne. For my sister-in-law, Suk Yin Ho, I acknowledge her company in Melbourne for a few years and thank her for being my sources of everything in Hong Kong.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

The main objective of this thesis is to investigate indexing and retrieval paradigms for the management of visual information. The focus will be on the development of new indexing and retrieval strategies for image data by making use of techniques in information retrieval.

Two well-known text processing approaches in information retrieval are signature-based indexing and inversion-based postings. They have been the most popular retrieval techniques for textual data. However, while we witness the success of these paradigms to retrieve documents, these concepts have never been seriously explored for the retrieval of image data. In this thesis, we would like to develop these paradigms to efficiently index and retrieve images based on their visual contents. Our

aim is to exploit these concepts and define new directions in content-based image indexing and retrieval.

Image retrieval applications are often categorized into domain-specific and general purpose. In this research, we shall focus on the latter. It is true that domain-specific applications are easier to handle and can have a higher precision on retrieval. However, the lack of generality will prevent us to recover or discover images from large collections. This is particularly relevant to data sources such as those on the Internet. Therefore, we concentrate on the general purpose indexing and retrieval techniques and do not assume *a priori* knowledge on the image data.

Many retrieval techniques interact with low-level syntactic information such as color, texture and object shape, with high-level semantic information largely ignored. Another important objective of this thesis is to integrate high-level concepts with low-level features in our indexing techniques and present them in a unified framework. This also implies that our indexing schemes should operate on the level of regions in images rather than on the entire images.

In this thesis, we view Visual Information Systems as a natural extension of traditional Information Systems which include visual data. All aspects of applications (e.g. ESS, DSS and MIS) and databases will need to be improved and extended to cater for enriched media types as the fundamental elements.

1.2 Summary of Contributions

The contributions of this thesis in content-based image indexing and retrieval for visual information systems are summarized as follows:

- We present a model for Visual Information Management. Our framework for Visual Information Management is not only concerned with fragmented aspects of visual information retrieval but a complete paradigm for the management of visual information. Our treatment to Visual Information Systems is parallel to the traditional field of Information System.
- We exploit the principal indexing methods in information retrieval for text data and develop them for content-based image indexing and retrieval.
- We study the concept of image hashing, and we establish the data type for image hashing in the form of two-dimensional bit matrix.
- We develop Bitplane Signature as a form of image hashing. Bitplane Signature is a signature accessing strategy for content-based image retrieval. It takes on characteristics analogous to signature-based text retrieval methods using two-dimensional bitplanes. We use wavelet decomposition as one of the possible hashing functions to generate bitplanes.
- We establish Composite Bitplane Signature as the superimposed code for fast image retrieval. Furthermore, the framework of Hierarchical

Composite Bitplane Signature allows us to rapidly produce a partial ranking of images without the need to perform similarity predicates on every image in the collection.

- We develop the concept of Inverted Image Indexing to provide an integrated indexing framework for accessing image contents both semantically and syntactically.
- We develop an integrated query model for our inverted paradigm which allows us to pose Boolean or ranked queries with the possibility of intermixing high-level semantic information and low-level syntactic features.

1.3 Thesis Organization

This thesis is organized into 8 chapters. Figure 1.1 maps out the organizational structure and the research work addressed by this thesis. The relationships between chapters are depicted by the corresponding arrows.

Rather than to bundle the experimental results in each chapter, we present our research findings first and postponed the experimental results to the end of the thesis. Also, since we cover a number of indexing and retrieval paradigms in this thesis, we will not provide collective background analysis at the beginning of the thesis but instead most of the background materials and supporting concepts are presented at the start of each chapter. In particular, Section 3.2 deals with the background materials of image compression techniques; Section 4.2 introduces the traditional hashing

techniques; Section 5.2 outlines the paradigm of text retrieval using signatures; and Section 6.2 presents the inverted model for text retrieval.

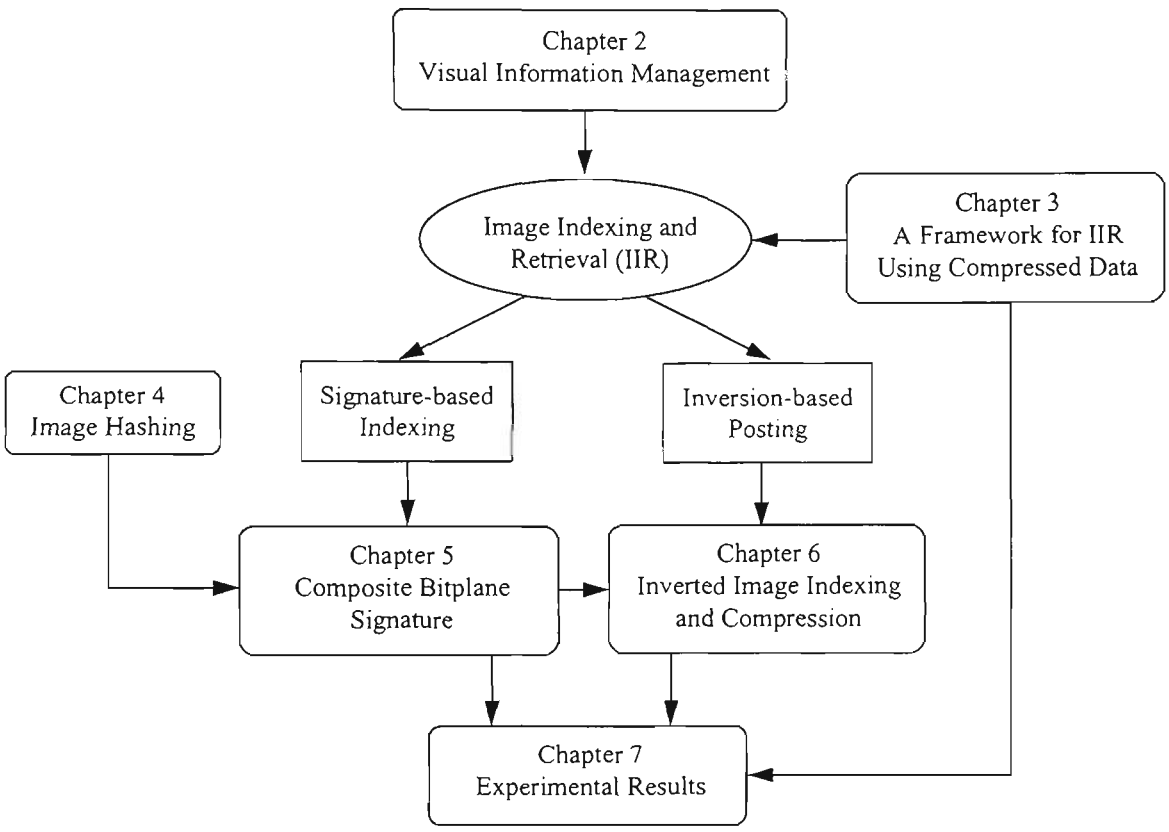


Figure 1.1 Thesis Organization

In Chapter 2, we analyze the many facets of Visual Information Systems and conclude that Visual Information Systems should be defined over the traditional field of Information System as an emerging field of study on its own right.

In Chapter 3, we identify that content-based image indexing and retrieval is an important element for the success of any visual information system. We specialize our investigation on the compressed domains for the purpose of indexing and retrieval. Representative techniques are analyzed from which we evaluate the common

approach and extract the desirable characteristics. We identify the problems related to this approach and suggest solutions in some aspects of the key areas.

In Chapter 4, we develop the concept of Image Hashing. This direction is motivated by the success of the hashing techniques in many areas of computing. We analyze traditional hashing methods and apply them to Image Hashing.

In Chapter 5, we present Composite Bitplane Signature as a signature-based image retrieval technique. We use wavelet decomposition as an example to generate the required bitplane signatures and superimpose bitplane signatures to form the composite bitplanes for fast image retrieval. We provide our unique ranking and searching algorithm which can significantly improve search efficiency. Furthermore, we introduce Hierarchical Composite Bitplane Signature as a mechanism to rapidly produce a partial ranking of images without the need to perform similarity predicates on every image in the collection.

In Chapter 6, We formulate a data and query model for accessing image contents using the concept of inversion-based postings. Our data model is capable of supporting high-level semantic indexing and low-level feature retrieval in a unified framework. Our inverted paradigm is a region-based data model with ability of indexing different visual contents within an image. Salient characteristics and meaningful visual contents of different areas within images are captured and organized using an inverted list. Our query model allows us to pose Boolean or ranked queries in an integrated environment for high-level contents and low-level features intermixed together in a query expression.

In Chapter 7, we provide experiments to reinforce different aspects of our research presented in previous chapters, and we conclude the thesis in Chapter 8.

Chapter 2

Visual Information Management

2.1 Introduction

The tremendous progress in computer hardware and software in recent times, particularly in multimedia, have pushed the dominance of visual information to an all time high. With the widespread use of digital images, videos and audios in various non-textual applications found in library, art galleries, museums, national defense, medicine, manufacture, security, scientific visualization and geographic applications, the need to efficiently manage, store, manipulate and retrieve visual contents is increasingly critical. The demand for flexible visual information access from the consumer side is compelling. This drives the research community to look beyond the traditional databases and information systems which are limited by the fact that they work well with only alphanumeric information. The fundamental problem is that

visual data are vastly different from alphanumeric data and require totally different approach and techniques of indexing and retrieval.

Future information system will be required to have the capability to process and retrieve visual information by content just as human beings routinely do. Next generation information systems will have a rich visual content, and there will be a shift in emphasis from a paradigm of pre-dominantly alphanumeric data processing to one of visual information processing. The traditional Information System (IS) will mutate to a new kind of information system: Visual Information System (VIS) [LEUN97a, LEUN97b, LEUN99, SO96b]. It is expected that VIS will maintain all the strengths and fulfilling all the functions of the former as well as open up a new horizon of enriched information processing. Not only can we correlate more meaningfully different kinds of information, the availability of previously untapped information sources will greatly enhance the effectiveness of an organization.

Although there has always been a demand for visual information, the technology for such systems was either immature or unavailable in the past. VIS is now becoming increasingly feasible because of a number of factors.

1. Advances in multimedia hardware for the efficient capture, storage, processing and delivery of visual information, which now pervades all categories of computer usage and is not just confined to certain specialized applications.

2. Ongoing improvement in software methodology for the effective handling of visual data and facilitated by the development of standards.

3. Advances in consumer demands such as the Internet and in digital communication such as FDDI, ATM and other high speed networking equipment, which by providing a significantly higher bandwidth, will enable the efficient transmission and delivery of visual information.

4. Widespread adoption of multimedia chips and general-purpose chips with multimedia functions.

There are many important issues in Visual Information Management that need to be addressed and, in this chapter, we will provide a unified perspective of these issues. This chapter is organized as follows. Section 2.2 briefly outline the characteristics of a conventional information system. The definition of Visual Information System is then provided in Section 2.3. Different aspects of Visual Information Systems are examined in Section 2.4. Section 2.5 describes different visual information contents and highlights the importance of inter- and intra-relationships of visual contents. The main features of a visual information system is then illustrated by an example. Section 2.6 examines the building block of a visual information management system. Finally, we conclude this chapter with a summary in Section 2.7.

2.2 Conventional Information Systems

The chief function of an information system is to provide information that can be easily understood and used. Such information needs to be provided not only in a timely fashion, but it must be presented in a form that is intelligible and natural so

that the information can be readily applied to solving management and decision problems. The efficient operation of an organization depends on people at various levels making correct or good decisions. While the decision making process requires a certain amount of judgmental input, a good decision always necessitate the availability of relevant information. Therefore, an *Information System* may be defined as an integrated, user-machine system for providing information to support the operations, management and decision-making functions in an organization [DAVI85]. At the structural level, it is made up of a set of components or subsystems that captures, processes, stores, analyzes, condenses, and disseminates information in various forms.

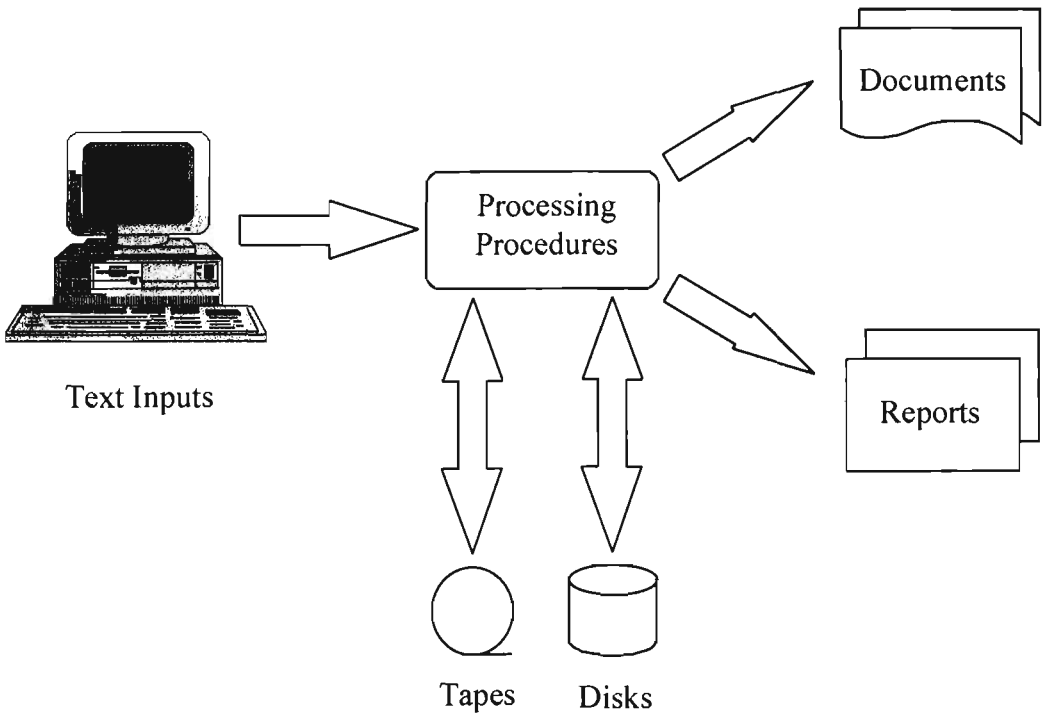


Figure 2.1 The Characteristics of A Conventional Information System

The typical characteristics of a conventional information system is shown in Figure 2.1. A conventional information system [ALTE96, SENN90, STAI96] is mainly based on alphanumeric input and alphanumeric output, which may be structured data fields or free text, and the type of processing involved is usually symbolic transformation or simple computation. A distinction is often made between data and information, where the former consist of unprocessed facts which may be useful, but most often are not readily useable. Useable information means that the associated data are organized and presented in such a way that the semantic richness of the information is clearly brought out. In addition, information that is useable at a particular level may not be readily useable at a different level. An example is data warehouse applications, where the availability of operational information used for online transaction (OLTP) typically do not permit data warehouse, time-related summary queries to be efficiently answered.

2.3 Definition of Visual Information Systems

Visual information systems are not just about the incorporation of new data types into existing information systems; rather, they require completely new ways of managing, using and interacting with information. It is true that the syntactics of the visual data are important. But more importantly, the semantic associations to the data are the most useable and naturally accessible information to users.

We must emphasize that visual information systems are not all about visual information retrieval. Other aspects not related to retrieval are equally important. It is

true that the retrieval aspects are fundamental and very important to the success of visual information systems. However, we believe the usability of the visual information within the context of man-machine interaction is far more important. The analogy to this is that the conventional information system is not solely concerned with the retrieval of data records. The proper retrieval of data records should be dealt with by databases. Information systems have a much broader scope than data retrieval as outlined in the previous section. Therefore, we must emphasize the usability of the visual information. Figure 2.2 shows the three levels of visual information retrieval.

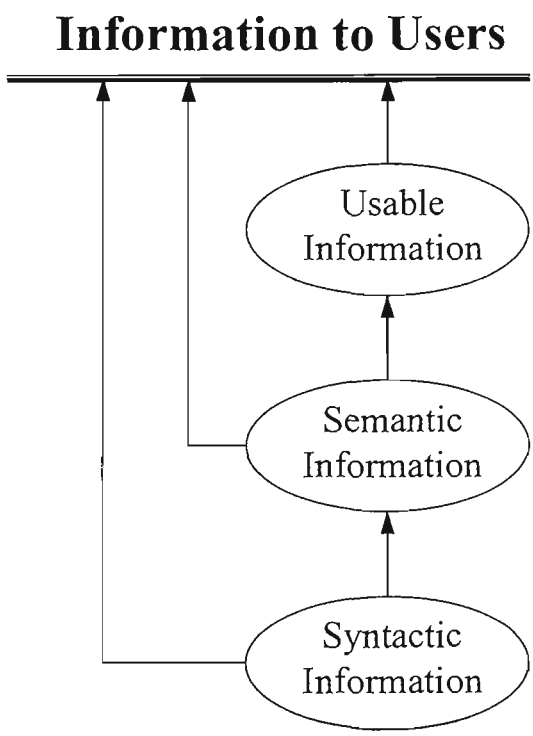


Figure 2.2 Three Levels of Visual Information Retrieval

It is interesting to note that the syntactic and semantic information is rather passive information. The usable information is the active information derived from the

inter-relationship of the visual information and supported by a framework that benefits the user. This filtered visual information enriches the information retrieval processes. Therefore, we stress the usability of the visual information and define Visual Information Systems as the following,

> *A Visual Information System is an integrated, user-machine system which is able to perform a number of functions on visual, multimedia and conventional information to support the operational, management and decision-making processes in an organization.*

This definition treats Visual Information System as an abstract field of study. This is parallel to the treatment of the traditional IS field. It emphasizes the applicability of usable visual information for the benefit of an organization. Also, we do not foresee an abrupt change from the traditional Information System to the Visual Information System defined above; it is a continuing and evolving process which depends on the maturity of a number of factors such as the indexing and retrieval techniques of visual information contents.

2.4 Aspects of Visual Information Systems

Similar to traditional information systems, the aspects and issues involved in visual information systems are very diverse and multifaceted [CHAN92, BERR93, DELB98]. In this section, we will provide views on various levels of visual information systems. These include (1) the high-level view of VIS which outline its

different components, (2) the system view of VIS in relation to the multimedia platform, (3) the computing platforms, and (4) the storage required to support VIS. Some of the major components in VIS such as the visual information contents, the multimedia database management system, the supporting tools and the applications will be further described in subsequent sections.

2.4.1 High-level View of Visual Information Systems

The high-level conceptual model of Visual Information Systems consists of five components: visual and meta-data, processing and supporting tools, a multimedia database management system, computer-based applications, and users. Figure 2.3 shows the high-level view of visual information systems from the prospective of an enterprise [LEUN98]. It is also a full system model of visual information systems involving visual data at all levels. The separation of visual information from the alphanumeric information is not necessary in our model. They are used by all types of users such as strategic management executives, operation managers and non-management employees in the hierarchy of an organization.

Figure 2.3 forms the skeleton of Visual Information Systems. It outlines the focal points for further discussion at various sections in this chapter. The characteristics and functionalities of each major component will be dissected and examined in detail.

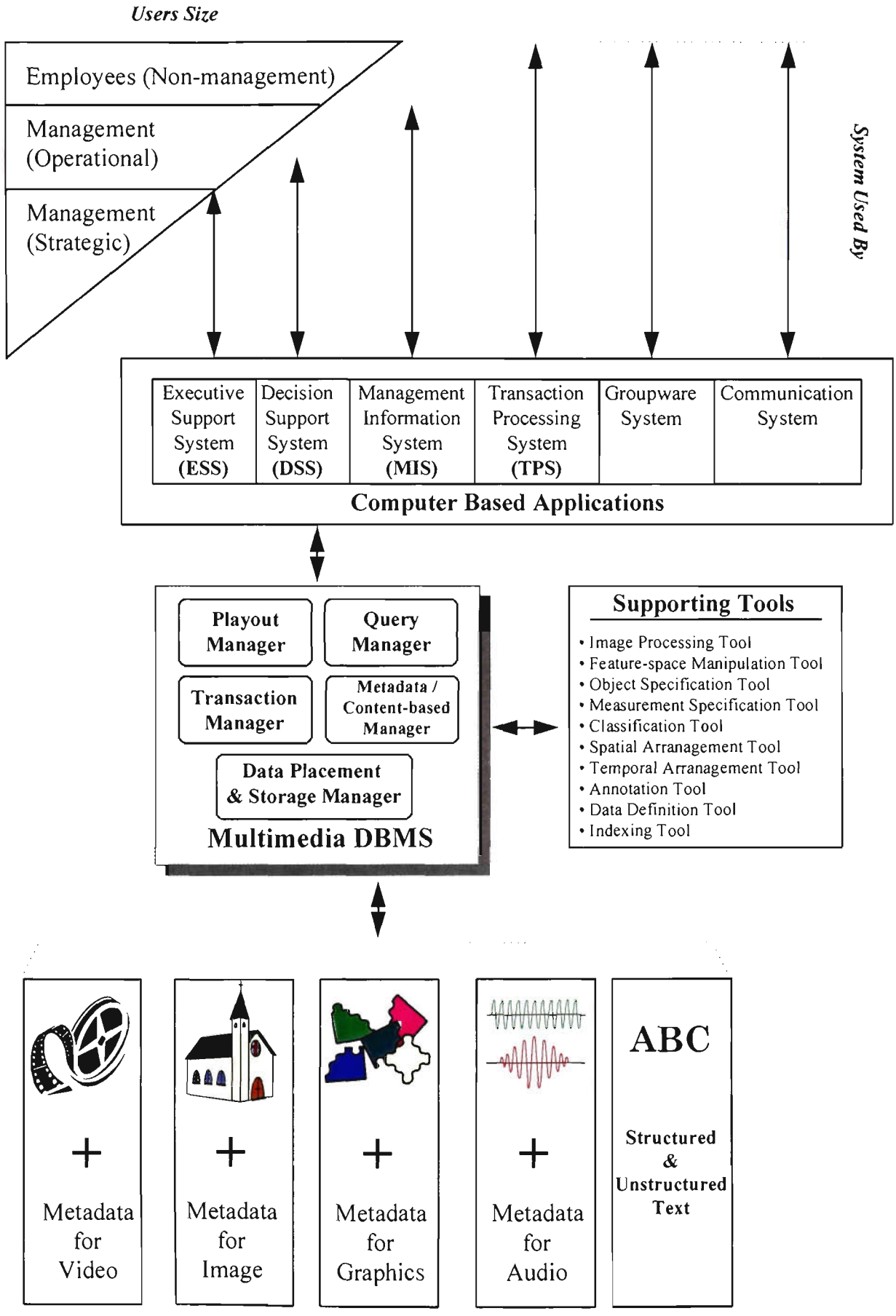


Figure 2.3 High-level View of a Visual Information System

2.4.2 System View of Visual Information Systems

The system view of Visual Information Systems comprises three layers as shown in Figure 2.4. They are the application layer plus the multimedia layers. The multimedia layers include a multimedia operating system (MMOS) interfaced with the multimedia hardware and resources; VIS is an application layer sitting on top of the multimedia layers. It is primarily concerned with the semantics, and possibly pragmatics, of multimedia information which occurs at a higher level much closer to the user. This sets it apart from the conventional wisdom and definition of Multimedia Information Systems. The latter has less clear separation between the low level multimedia aspects and the high level semantic information than our VIS definition.

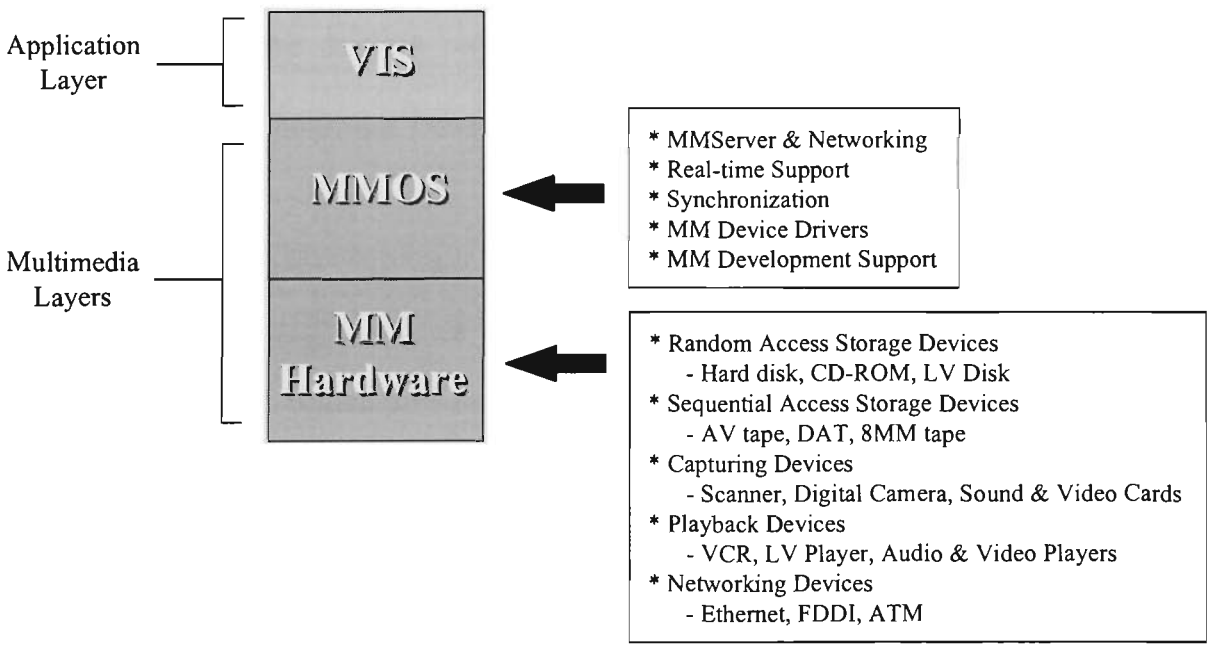


Figure 2.4 System View of Visual Information Systems

A multimedia operating system (MMOS) takes on the role similar to conventional operating system but with added facilities to support the sharing of multimedia resources. The multimedia operating system encompasses a large variety of multimedia devices with strict real-time constraints. The synchronization and real-time support of multimedia data streams are the prime supporting functions of MMOS. Multimedia server with networking capabilities [ANGE97] are needed to support multiple clients and data sources across a distributed computing environment. Sharing multimedia content in a distributed environment poses a unique challenge to the conventional servers [FURH94, FURH95]. Another important aspect of a multimedia operating system is the provision of a highly visual and standardized user interface so that the user is familiar across different applications within the visual information system as outlined in Figure 2.3.

The hardware devices required to support a VIS application is far more complex than a conventional IS application. We can roughly divide them into five categories:

1. *Random access storage devices.* These devices are characterized by the ability of organizing multimedia data on the storage medium to be read and/or written without a sequential search. Most of the random access storage devices such as CD-ROM and other laser video disks employed in multimedia applications are read-only. This is because cost considerations prohibit using conventional R/W devices such as hard disks to store a large amount of multimedia data for any practical use.

2. *Sequential access storage devices.* These devices are characterized by the sequential arrangement of multimedia data on the storage medium. They include any analog storage device such as VHS tapes and any digital storage device such as DAT. Because of the lower cost, they are best suited for back up as well as storing *cold* multimedia data.

3. *Capturing Devices.* These devices are used to take-in different external multimedia sources. They include scanners, cameras, sound and video capturing cards etc. A number of standard formats may be needed for various sources.

4. *Playback Devices.* These devices are used to playback any segment of the multimedia sources. Not all devices that can playback are appropriate. For example, household VCRs are not appropriate for large-scale production systems as they are usually lack of precise control and synchronization.

5. *Networking Devices.* These devices are important if a distributed environment is necessary. Because of the high bandwidth required by multimedia data, FDDI and high speed communication equipment such ATM are usually employed.

There are many other specialized devices needed for some VIS applications. For example, a geographical information system may employ high speed plotters and large size digitizers. Some of the aspects discussed in this section will be further re-visited at different contexts in the next sections.

2.4.3 High Performance Computing Platforms

In order to effectively process visual and multimedia information, powerful processing capacities are essential to realistically match user expectations and real-time constraints. Platforms based on multimedia chips or general-purpose high performance chips with rich multimedia instructions are typically deployed. This is often coupled with the use of high bandwidth networks, distributed or parallel servers, and operating systems geared to multimedia functions (see Figure 2.4).

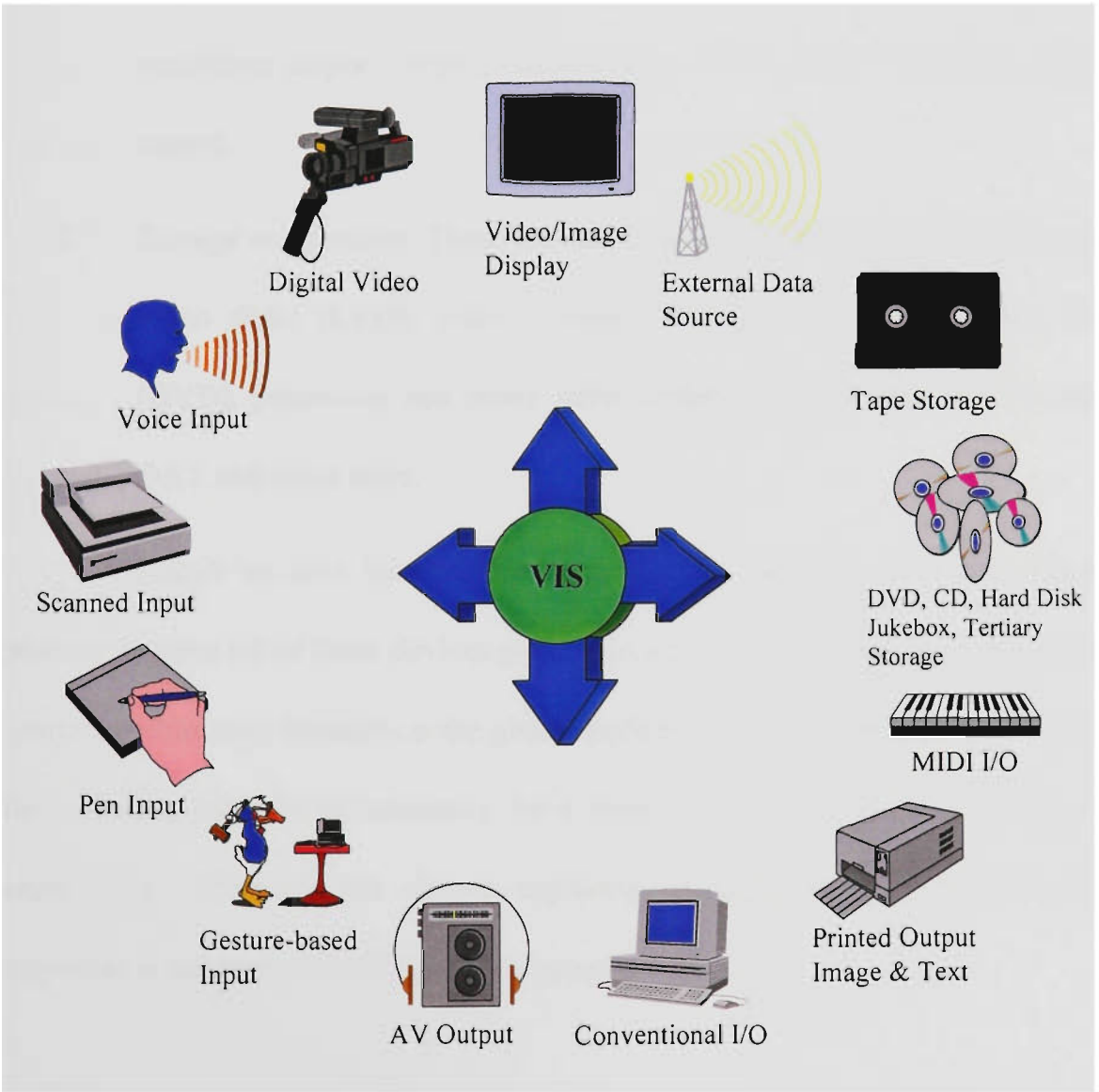


Figure 2.5 I/O and Storage Components of a Visual Information System

A wide range of I/O and storage devices is necessary to support a more natural and visual interaction with human users. Figure 2.5 outlines the different components vital to any visual information system. We can classify these components into three major categories:

1. *Input components*. These include conventional keyboard input, scanned input, voice or audio input, video input, MIDI input, pen-based input, gesture-based input and any external input.

2. *Output components*. These include screen output, printed output, high resolution display output, audio output, MIDI output and any external output.

3. *Storage components*. These include hard disks, compact disks (CD), laser video disks (LVD), video compact disks (VCD), digital video disks (DVD), jukeboxes and many other tertiary storages such as AV tapes, DAT and 8mm tapes.

Although we have listed a wide range of I/O and storage devices, it is not necessary to have all of these devices present in a particular system, and the relative importance ultimately depends on the given application. For example, audio input and output devices may not be necessary for a visual information system dealing with images only. Although not shown explicitly, a high performance processing component is necessary to underpin such systems.

2.4.4 Storage Consideration

As visual data objects are orders of magnitude larger than conventional structured records, large capacity storage systems are required to hold the data in a visual information system. To highlight the storage requirement, we provide a comparison of different common media types in Table 2.1 [BHAS97, HOFF97, RAO95]. To reinforce this point, let us consider an example using CD-ROM to store a video clip with broadcast quality. From Table 2.1, 720 x 480 resolution with 30 frames per second requires 31.1 Mbytes for a second of video clip if the data are not compressed. For a 650 Mbytes CD-ROM, only 21 seconds of video can be stored. Even with compression such as MPEG-1, only 74 minutes of VCR quality video can be stored in a single CD-ROM.

Besides the large capacity of storage requirement in visual information systems, we need to have sufficient bandwidth to deliver visual information in an efficient and timely manner. The use of extensive secondary and tertiary storage will be unavoidable in large visual information systems. In the long term, hierarchically organized storage systems covering a range of retrieval speeds, costs, and capacities will need to be managed [GEMM95]. Algorithms governing the staging and migration of multimedia data will need to be incorporated.

Media Types	Resolution	Bits per Unit	Storage Capacity*
Text:			
ASCII (a page)	~ 2000 chars	8 bits / chars	2 Kbytes / page
Image:			
Binary Resolution			
- Fax CCITT G3	A4 - 1728 x 2444	1 bits / pixel	0.5 Mbytes / page
Medium Resolution			
- VGA	640 x 480	8 bits / pixel	307 Kbytes / image
Detailed Resolution			
- SVGA	1024 x 768	24 bits / pixel	2.36 Mbytes / image
Audio:			
Voice			
-Telephone	8 Ksamples / s	8 bits / sample	64 Kbits / s
- Audio conference	8 Ksamples / s	16 bits / sample	128 Kbits / s
Digital Audio			
- CD quality	44.1 Ksamples / s	2 x 16 bits / sample	1.41 Mbits / s
- AC-3 as in HDTV	48 Ksamples / s	6 x 18 bits / sample	5.18 Mbits / s
Video:			
Low Resolution			
- VHS ≅ MPEG-1	352 x 240	12 bits / pixel, 30 fps	3.80 Mbytes / s
- CIF as in H.261	352 x 288	12 bits / pixel, 30 fps	4.56 Mbytes / s
Broadcast Video			
- CCIR601 NTSC	720 x 480	24 bits / pixel, 30 fps	31.1 Mbytes / s
- CCIR601 PAL	720 x 576	24 bits / pixel, 25 fps	31.1 Mbytes / s
HDTV (GA)			
- High Level	1920 x 1080	24 bits / pixel, 30 fps	186.6 Mbytes / s

* Uncompressed data only. Different compression schemes are used to reduce the storage and transmission requirement.

Table 2.1 Comparison of Different Media Types

2.5 Visual Information Contents

Data are raw facts that may or may not be meaningful to the users. Information is processed data that is meaningful for a particular use. In visual information systems, the basic data types and information contents are far more complex than in traditional information systems. We are no longer working with alphanumeric texts. The data modeling techniques using records, structured data and tables are hardly sufficient to capture the vastly different forms of visual contents. Content-based information retrieval techniques must be employed to address the central needs of visual information retrieval. Meta-data generated from the media types are used to provide the indexing and retrieval information. However, the accurate retrieval of visual information contents is a major problem today. Hence, content-based visual information retrieval will be an on-going research focus.

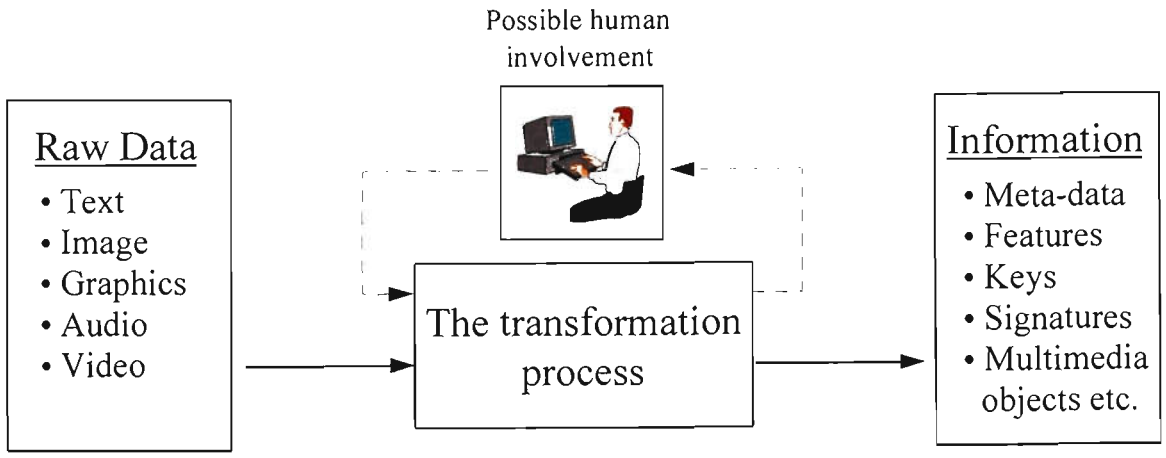


Figure 2.6 Transformation from Raw Data to Visual Information

In our view, there are five basic types of data in visual information systems. These include texts, audio, images, graphics and video. Each data type can be filtered, processed and transformed into its meaningful information content as shown in Figure 2.6. The processing of data can be performed automatically, semi-automatically, or manually. Generally, syntactic information can be processed automatically. For example, color histograms as a form of feature vectors in images can be computed automatically. On the other hand, semantic information is often extracted manually. For example, the emotion of an actor in a video clip is annotated manually. Figure 2.7 shows the general scenario of information processing which depends on the level of abstraction for the information content.

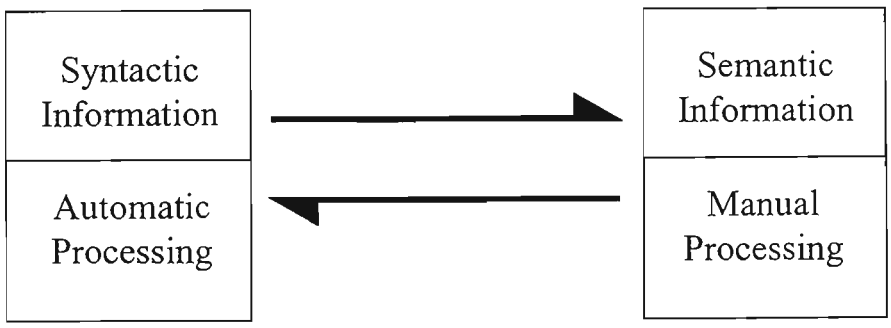


Figure 2.7 General Scenario of Information Processing

In this section, we will provide an overview of visual information contents. The importance of inter- and intra- media relationships is emphasized in Section 2.5.1. The main features of a visual information system in relation to different visual contents are illustrated by a specific example of a law enforcement application in Section 2.5.2. The rich data contents and the diverse media relationships for this type

of application reveals and demands the needs for an efficient and effective visual information system.

Text-based Content

Text-based content is the most widely used in traditional IS and comes in a variety of forms. It can be represented using keywords, free-text, structured records etc. Keywords and structured text can be indexed using established automatic indexing algorithms [SALT89]. Relational and object-oriented databases are the basic models for representing structured records. Although object-oriented databases offer greater scope for accommodating multimedia objects, they do not in general provide tools for their manipulation and identification. Free-text such as the annotation of images and videos can be indexed using free-text oriented databases. The ambiguity inherent in English can be minimized by using a thesaurus and classification methods.

Audio-based Content

Audio-based content [SCHA97] is the least researched area in content-based retrieval. It can be stand-alone content or tied to video-based content. Parsing of audio content can be done by signal processing and spectrum analysis. Although it is unlikely that automatic extraction is possible due to the multiplexing of different sources of sound in real-life scenario, some simple sound patterns such as door bells, hand clapping or engine vibration can be detected using model-based approaches if the background noise can be minimized. Voice analysis can be done to locate the

"signature" of any individual voice, and speech recognition programs may be employed to extract input semantics. Audio content may be used in conjunction with visual-based content to identify target image sets or video clips.

Graphics-based Content

Line arts, synthetic images and computer animation may be described by their geometrical properties. The line, polygon, surface patches and many mathematical entities can be indexed numerically, and this form of content can be easily manipulated and computed. The motion of objects in computer animation can be traced and the sequences of events can be indexed. In some situations, it may be more efficient to index the characteristics of the underlying graphics generation algorithms, since fast searching and identification may be done using text-based pattern matching procedures.

Image-based Content

Image-based content is the most researched area in content-based retrieval. Image content such as color, texture, object shape, sketch-based feature can be indexed [NIBL93]. Proximity among objects can also be represented. Color histogram, moment/centroid, segmentation, and other primitive features can be used to form the feature vectors, keys or signatures of the images. The information can then be indexed for similarity retrieval. Image compression techniques are useful not only for transmission and size reduction, but can also be used for indexing, identification,

and signature construction. Visual queries, sketches and fuzzy matches are some of the techniques that can be used for query formulation. Due to the imprecise nature of queries, browsing through miniature icons, compressed images or picture keys would seem to be indispensable operations in image retrieval systems [SO96a].

Image-based contents can be categorized into complex contents and primitive contents with different methods for their extraction. Primitive contents [LEUN95, ZHEN95] are low-level contents which can usually be extracted automatically by computers (eg. textures, colors, boundaries, and shapes). Complex contents [HIBL92, HIBL92, LEUN95] are extracted manually or human-assisted using computers, and corresponds to patterns within a picture which are considered as meaningful by human users and may be applications dependent or applications-independent. They cannot normally be automatically identified by the computer and are often qualitative in character (eg. a class room, a bride, a sports car).

Video-based Content

Video-based content retrieval has been attracting substantial research interests recently, possibly due to the demands arising from consumer electronics. In-house shopping, electronic kiosk, and video-on-demand are some of the commercial driving forces. Motion vectors, salient video stills, annotations and video partitioning are some of the methods employed for indexing and retrieval. The spatio-temporal aspect of objects between frames is an important property that needs to be indexed. A sufficiently fast index to support real-time retrieval and playback would appear to pose a considerable challenge.

2.5.1 Inter- and Intra- relationships of Visual Contents

The relational data model introduced by E.F. Codd in 1970 has been extremely successful among different classes of databases. This is largely due to the highly structured data organization and proven theories in such model. Although the relational paradigm works very well for text-based information, it is hardly sufficient for visual contents. Laying everything into table structures and using relationships to associate entries scantily met the needs to model the rich data contents and diverse media relationships.

Holding different categories of information will require storing the relationships between the different media. Intra-media data relationship will need to be stored, which signifies the relationship between data of the same media type. For example, the relationship between different entities in a conventional database are represented variously by joins, foreign keys, and aggregation hierarchies. Inter-media data relationship signifies the relationship between data belonging to different media types. Examples include the synchronization between video and sound, a still image and associated textual descriptions, and an individual's voice print and fingerprint.

Without better frameworks and data models for visual data, we would like to emphasize the importance of intra- and inter- media relationships using the notations from the popular ER modeling. This is shown in Figure 2.8. It is an abstract view of different media types in relationships with each others. For simplicity and without lost of generality, the inter-media relationships are represented in the degree of four. They are in many-to-many-to-many-to-many relationships involving text-based contents,

image-based contents, audio-based contents and video-based contents. It is, however, somewhat incorrect. Technically, they should be binary, ternary and/or 4-nary relationships among all the different media types depending on the specific requirement. For example, if only text-based and image-based contents are involved, only binary relationships are involved. The cardinality ratio should be M:N. Furthermore, the intra-media relationships represented in Figure 2.8 for each media type signifies the importance of relationships among the same media type. They are not the recursive relationships as in the traditional ER concepts.

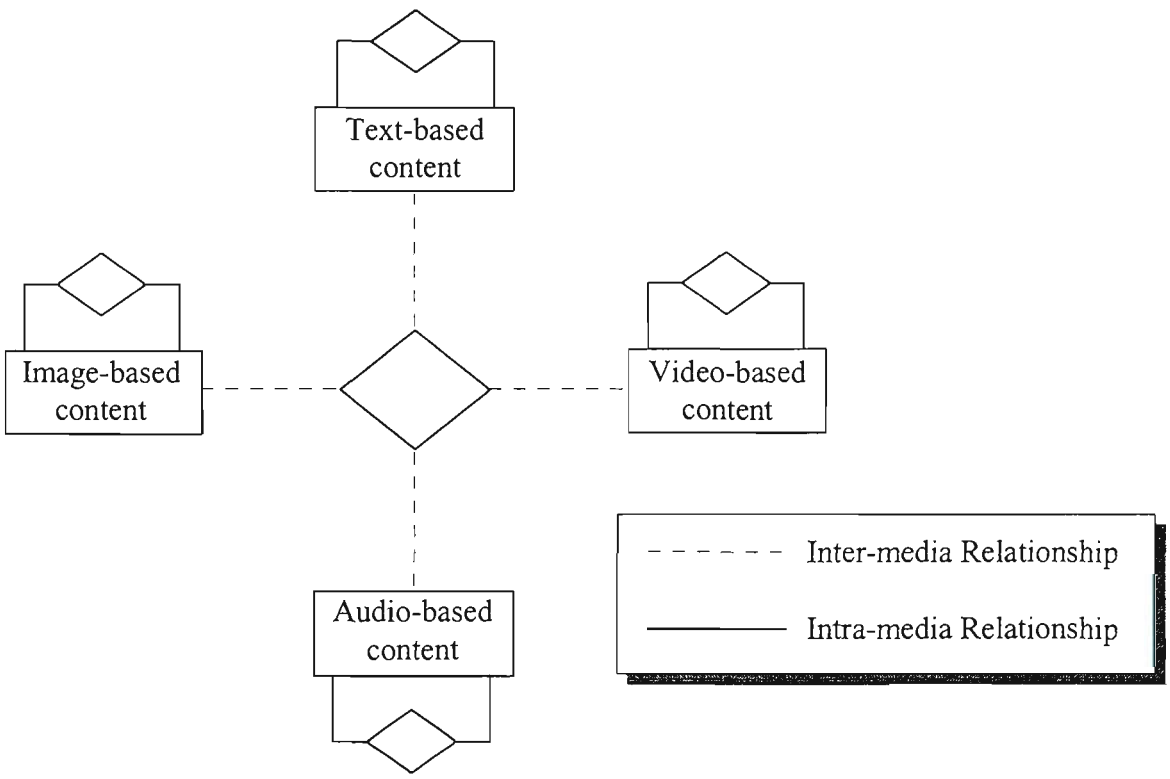


Figure 2.8 Inter-media and Intra-media Relationships

One of the major difficulties in representing intra- and inter- media data relationships is the imprecise specification of retrieval information. Hence, unique

keys or constraints are not always achievable. It is the reason that alternative data models beyond the classic relational data model are needed. Also, since the retrieval or query formation is mostly imprecise, techniques in classification, clustering, or relevant feedback are often used to facilitate the searching process [SO96c, SO97a].

2.5.2 An Example: VIS for Law Enforcement

For concreteness, we shall illustrate the main features of a visual information system by making use of a law enforcement application where a collection of visual and textual data for criminals must be managed. This provides a particularly appropriate example to highlight the effectiveness of VIS because of the integrated

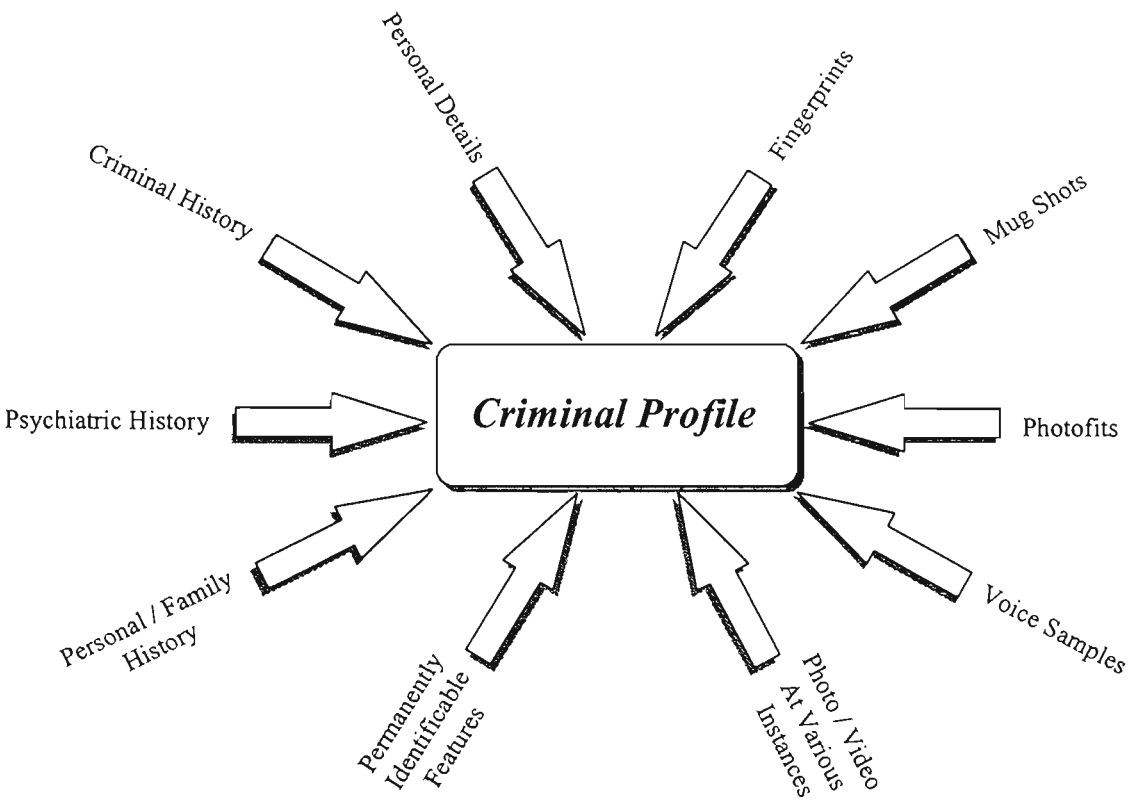


Figure 2.9 Components of Criminal Profile

nature of different data sources. Without the management by an integrated visual information system, the inter-relationship between different types of information will be difficult to extract from the fragmented and sometimes isolated documents or materials.

Relevant Data	*Media Type*	*Examples*
Personal Details	Structured Information	Age, sex, race etc.
Fingerprints	Image or Graphics	From crime sites, from previous offences etc.
Mug shots	Images	From previous offences, from family albums etc.
Photofits	Graphics	Composed by witness using Identikits
Voice Samples	Audio	From telephone taps, blackmail conversation etc.
Photo / Video Clips	Images / Video	From robbery, from banks' cameras, from tailing by police etc.
Permanently Identifiable Features	Text or Images	Tattoos (images), finger chopped off (text or images) etc.
Personal / Family History	Text	Description of childhood, parent profiles, habits etc.
Psychiatric History	Text	Classification of the criminal, patterns of behavior etc.
Criminal History	Text	Types and styles of previous offences, weapon used etc.

Table 2.2 The Media Types and Examples of Criminal Profile

There are many sources of information that needs to be managed in a visual information system for criminals. Personal details, criminal history, finger prints, for examples, are just some of the essential components. Figure 2.9 shows some of the components of a criminal profile to be used in building the visual information system, with the media of each component given in Table 2.2.

To use the system effectively, a number of paradigms may be adopted:

- Personal details can be retrieved using the conventional database and conventional query language such as SQL.
- Personal, psychiatric, and criminal historical information can be indexed using free-text indexing together with standard information retrieval techniques.
- Techniques from content-based image retrieval and content-based video retrieval will need to be employed for the indexing and retrieval of mug shots and video clips of criminals.
- Graphics contents can be scaled and modified numerically for similarity search.
- Automatic extraction of primitive contents such as mustaches, spectacles, contours of faces, voice features and other features of criminals.
- Voice characteristics may be matched and edited graphically.
- Complex contents from photos can be extracted and indexed with human assistance.

- Fingerprint information may be matched and retrieved using a combination of signature and classification schemes. Four typical groups - Arch, Loop, Whorl and Composite - are shown in Figure 2.10. Natural fingerprints may be composed of complex noise and disconnected virtual lines, and it is necessary to use some preprocessing techniques for image enhancements to improve their quality.

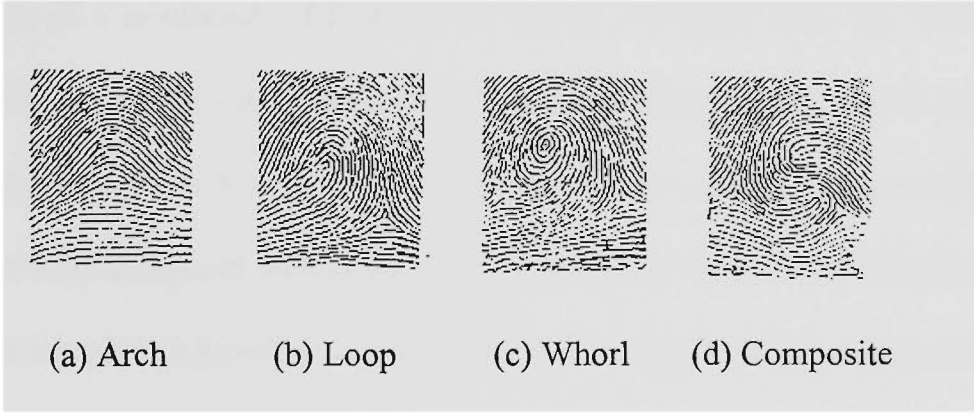


Figure 2.10 Fingerprint Classification

The effectiveness of this system is evident when an integrated query is performed. For example, a police detective may query the Criminal Profile System for "a male criminal who classified as extremely dangerous, has a tattoo of an eagle on his back and his left-thumb fingerprint is whorl". This query explores the *inter-media* relationship among personal details (eg. male), psychiatric history (eg. extremely dangerous) and permanently identifiable features (eg. tattoo and finger prints), which are represented as structured information, free-text and images respectively. The *intra-media* relationship of similarity search in tattoos (eg. eagle) and fingerprint (eg. whorl) will also be required.

2.6 Building Blocks of Visual Information Management Systems

The key components of visual information management systems consist of a complete and fully functional multimedia database management system supported by a number of processing tools catered specifically for the indexing and retrieval of visual information. In Section 2.6.1, we will present our view on the essential composition of a multimedia DBMS. It is largely consistent with the direction of many leading researchers in the areas of Multimedia DBMS and Visual Information Management. Since the handling of visual information requires exceptionally large numbers of supporting and processing tools, a list of the tools is provided in Section 2.6.2. Also, a number of applications in relation to the visual information management are discussed in Section 2.6.3.

2.6.1 Multimedia Database Management Systems

The rapidly growing research and development in multimedia computing has been quite significant in recent time. The demand of Multimedia DBMS is becoming critical for many applications. Although the history of multimedia or pictorial databases can be traced back to two decades ago [CHAN80, GROS92], they were mostly domain-specific using traditional database technologies such as the relational database as the backbone for their implementation. Due to the complexity of visual data, the short-coming of using relational databases for multimedia DBMS are evident. Hence, the adoption of hierarchical data models [GUPT91], object-oriented

databases [CARD93], object-relational databases [OGLE95] are proposed. Unfortunately, they are far from adequate. This is the key factor limiting the number of commercial multimedia database management systems available today. Therefore, research in Multimedia DBMS is actively pursued [ADJE97, GROS97a, GROS97b, MARC96, NARA96, NWOS96, NWOS97, PRAB97, SUBR96].

The difference between traditional databases and multimedia databases can be characterized by the following features:

1. Large storage sizes: The storage requirements for audio, image and video together with the meta-data such as feature vectors, signatures and motion vectors are extremely high in comparison to traditional textual databases. Large memories and tertiary storages are required to hierarchically manage the data at different levels. This leads to issues on performance and data placement to support the media requests.

2. Spatial and Temporal nature: The visual contents, for example, audio and video are time-dependent and continuous in nature. Image- and graphics-based contents are spatial in character. This is very different from standard alphanumeric records and tables in traditional databases where well-defined mathematical data models are used.

3. Coexistence of raw and semantic information: Raw media data are difficult to handle. They are awkward to retrieve, update and store. Meta-data which can capture the semantic information are generally used to retrieve them. Such retrieval are imprecise in nature and often resolve to

similarity matching. Hence, the browsing operation is a typical operation in multimedia databases.

To build a multimedia database system, some of the main properties and techniques used in traditional databases are desirable [ELMA94, GROS97b, SILB97]. For examples, the ACID properties (*Atomicity*, *Consistency preservation*, *Isolation*, *Durability* or *Permanency*) on transaction management are highly desirable. Concurrency control techniques, recovery techniques, security techniques and version control techniques are particularly useful for advanced multimedia database systems. Although some of the functions found in a multimedia database system resemble those in a traditional database, the complexity and characteristics in different visual contents demand new functionalities and techniques. For examples, it will be almost impossible to *roll back* a transaction that updates a large video object. Defining the right granularity for concurrency is difficult for continuous media objects. Scheduling and synchronization of different media streams under intense user interaction during *playout* is an added challenge.

The major components of Multimedia DBMS include 1) Playout Management, 2) Query Management, 3) Transaction Management, 4) Meta-data / Content-based Management, and 5) Data Placement and Storage Management as shown in Figure 2.3. Although there is no consensus on what components should be included in the Multimedia DBMS, the illustrated components are generic enough that any Multimedia DBMS should at least include them. It is true that not all features in each component are required, and may be eliminated for certain domain-specific

multimedia databases. For example, if video data are not involved, disk striping for data placement is not required.

Playout Management

The functionality of Playout Management [THIM95] is to orchestrate the multimedia presentation upon user's request and interaction. This is for lower level data stream handling and mainly interacts with the data placement and storage management component. The fundamental tasks for playout management include: 1) Device Management, 2) Data Stream Management, 3) Synchronization Enforcement, and 4) Support of Interactivity. Presenting multimedia data under real-time constraints in a distributed environment is not a trivial matter particularly with large objects such as video.

Query Management

Query Management is concerned with servicing the user's request. Unlike the traditional databases, query processing can involve a number of different contents and media types. Visual queries can be formulated using the following paradigms:

- text
- color
- texture
- shape
- volume
- spatial constraints
- temporal constraints

- objective attributes
- semantic similarity
- browsing
- domain concepts

It mainly interacts with the meta-data / content-based management component. Generally, the specification and interpretation of visual queries are imprecise and subjective. Ranking of target objects coupled with relevance feedback to refine queries are usually required. Also, an integrated visual language or interface is also needed [CARD93, CHAN90, CHAN96a, JOSE88].

Transaction Management

The functionality of Transaction Management is to ensure the proper execution of any transaction so that the integrity of data in the database is maintained. Transactions in multimedia database systems tend to be long and computational expensive. Updating visual objects or meta-data can place a heavy demand on system resources. Therefore, the execution plan has more stringent constraints than that for traditional databases. This is further complicated for a distributed and concurrent environment.

Meta-data / Content-based Management

Meta-data is derived data describing the contents, structure and possible the semantic of the data [PRAB97]. In Multimedia DBMS, meta-data for visual contents goes beyond titles, captions, authors, etc. of media objects. Useable information obtained from multimedia processing tools provides the semantic association to the

raw visual data. The extraction and organization of meta-data through the support tools are the functions of Meta-data Management. Indexing can be performed on the meta-data to speed up searching.

Data Placement and Storage Management

As visual objects are orders of magnitudes larger than conventional structured records, hierarchical storage organization involving large secondary and tertiary devices are required. Data migration from slower devices to faster devices can be a problem if the real-time requirement of some objects such as video are not satisfied. The problem is further compounded by the requirement of multiple data streams with multiple requests. Data placement such as multiple disks with striping, redundancy and interleaving are some of the techniques to provide simultaneous access of different data streams. Data compression schemes should also be used to reduce the storage and bandwidth requirement [SO96a].

2.6.2 Processing and Supporting Tools

One of the major differences between traditional databases and multimedia databases is that the latter requires a large number of processing and supporting tools in order to extract the required meta-data and supporting information. The purposes of these tools are to provide processing, manipulation and specification of visual data.

The collection of different tools should include at least the following items [GUPT97a]:

- *An image-processing tool.* This supporting tool includes techniques for image analysis and modification. Segmentation, texture extraction and shape recognition are some of the common techniques in image analysis. The techniques in color equalization and filtering are often needed for image modification. This tool would also be used interactively in the image querying and insertion processes.
- *A feature-space manipulation tool.* This supporting tool provides a better manipulation on the features and allows the specification of nearby images using such features. Users can explore the feature space by clustering, classification and projection. If the features are represented in n-dimensional vectors, some sorts of measures in distances are used to support the neighborhood search.
- *An object specification tool.* This supporting tool compliments the low-level shape recognition tool found in the image-processing tool. It allows the specification of objects using a high-level perspective. For example, the shapes of a kangaroo in various images may be quite different (eg. hopping, sleeping or boxing) but may be referred as the same object.
- *A measurement specification tool.* This supporting tool allows the measurement of objects or regions in size if such information is important.

For example, in geo-information systems, the area of a certain pattern will be used to determine the object's type.

- *A classification tool.* This supporting tool allows the grouping of different visual contents and objects using certain conditions and criterions. Classification can be general purposes or domain-specific. It can also be a high-level semantic classification or a low-level grouping of syntactic features.

- *A spatial arrangement tool.* This supporting tool allows the specification of spatial information related to objects in images or video. It is particularly useful in moving objects in which the interaction among objects are relevant. Very often, the locations of objects are normalized so that comparison can be performed relatively or absolutely.

- *A temporal arrangement tool.* This supporting tool allows the specification of temporal information related to the events in video. It captures the temporal information such as the transitions of objects, scene changes and camera cuts. Users can query the database using some specific temporal events and retrieve sections of video based on those properties.

- *An annotation tool.* This supporting tool allows the users to annotate the visual contents at different levels of abstraction. The annotation should be searchable using conventional database technologies. Thesauri are helpful to support ambiguous queries. Classification of terms can also be used for

search space reduction [SO96c, SO97a]. Annotation can be performed on: 1) the regions within an image, 2) a whole image, or 3) a group of images. Structured data model [LEUN95] for the annotation of visual contents are better than the use of keywords or captions in an uncontrolled searching environment.

- *A data definition tool.* This supporting tool allows the users to model the properties of the visual contents. Users can define the contents similar to a database schema which may contain domain-specific attributes or generic attributes. Besides the relational model, other forms of data modeling can also be used.

- *An indexing tool.* This supporting tool provides the facilities to organize meta-data, features and other searchable information into some forms of fast searching structures. Tree-based accessing structures are the most commonly used data structures for visual information and contents. This tool is important from the consideration of performance. Users will not tolerate a long response time for any given query.

Depending on the application, new tools may be required to assist the DBMS to acquire the appropriate information. In the object-relational multimedia database such as Illustra, the implementation of these tools can be done using the concept of Data Blade.

2.6.3 Applications of Visual Information Management Systems

The use of visual information is not just confined to standard OLTP applications, but pervades all application categories, which include DSS (Decision Support System), ESS (Executive Support System), MIS (Management Information System), TPS (Transaction Processing System), Groupware System, Communication System, Data Warehouse and Data Mining. In this section, we will present our view on the impact of visual information to these applications.

Decision Support System

DSS helps operational and senior managers to make decisions relating to unstructured and semi-structured problems arising from various business situations. DSS supports analytical work by providing flexible tools and models for analyzing data, identifying and evaluating alternatives, selecting best solution and implementing the solution. Visual data can enhance the understanding of a problem's dynamics and helps to pinpoint the cause of the problem and identify possible solutions. For example, in investigating the cause of an aircraft crash, video clips, images, sound clips and flight-recording data together with a number of hypothesis and mathematical models may be used to decide what is the likely cause of the crash. In turn, a solution may be selected for further simulation so that the problem can be rectified for existing operation and future aircraft designs.

Executive Support System

ESS provides information for executives and are mainly used by senior management for strategic planning and charting future company direction, which typically includes information both from internal and external sources. The use of visual content will provide a better understanding of problems and business trends, and is particularly essential where the business deals with commodities inherently visual in character. For example, a fashion company may need to obtain the latest trends in the industry from various sources and to predict sales of certain styles of outfits in different regions of the world. It can also provide links to the information sources of related industries such as shoes, cosmetics which may also be highly visual in character. In ESS, the visual content must be easy to use and highly interactive, since it is unreasonable to require executives to understand the specification and construction of meta-data and the formats of the visual contents.

Management Information System

MIS provides information for the day to day management of an organization, with regular reports generated for operational and senior managers. Multimedia management reports can replace the traditional reports, where visual contents can be incorporated to enhance the expressiveness and presentation of the associated management information. For example, the operation of a turbine in a processing plant can be monitored not only by the associated numerical data, but also by a video camera which continuously record the operation. Here, alphanumeric summaries together with snap shots of the operation can be provided for daily management.

Transaction Processing System

TPS collects, processes and store data into structured information. Specific processing steps of the transaction are pre-determined and repetitive operations. Normally, a large amount of computational efforts are required. This is particularly true where visual data are involved. The assimilation of raw visual data is usually domain-specific using different types of processing tools. For example, satellite images for the Bureau of Meteorology are continuously collected and processed using various image processing tools. Intervals of images together with analytical data are stored.

Groupware System

Groupware System provides the appropriate software to allow groups to work together on common data. It enables team members to collaborate on a task. Controlling workflows and sharing information are two major components of the groupware system. The richness of visual data has a significant impact on groupware and workflow systems. We are no longer dealing with alphanumeric data. Many issues such as security, authorship and bandwidth for visual data are becoming important issues. New paradigms are required to deal with them.

Communication System

Communication System in the context of Visual Information System is no longer restricted to conventional E-mail systems. Voice-mails, video-mails, video-

conferencing will become the norm of communication in the future. Communication Systems with multimedia capability will make the sending and receiving of visual data more transparent. Virtual whiteboards can pull people at remote locations to participate in an virtual environment. Internet is becoming the loosely-coupled channel for communication and the major information source.

Data Warehouse

Data warehouses typically contain the integration of relevant data obtained from diverse operational database systems, which are summarized at various levels of granularity over certain time periods. Firstly, in the case of conventional data, effective data visualization techniques can help with the clear presentation of trends and the identification of problem areas over a given time horizon, which will facilitate the making of sound management decisions. Secondly, in certain businesses, the use of visual information for operational and decision making purposes is inevitable. In such situations, the data warehouse must be able to incorporate visual information for meaningful decision making. For example, one might wish to visualize the evolution of a given product's size and shape over time, and determine to what extent these have impacted sales and market share.

Data Mining

Data mining problems are ones relate to the discovery of associations and classifications of entities and data items. Very often, one makes use of certain

available data to classify an entity and to estimate its likelihood of falling within a certain category with prescribed characteristics. For example, if a criminal is seen to inflict injury on his victims in a particular characteristic manner, then a certain profile of that criminal may be postulated. Another example will be the correlation of graphical trends in different stocks: if stock A is found to behave in a particular way, then stock B will likely to follow a characteristic pattern within a month.

2.7 Summary

Visual information is being collected and used in a wide variety of application domains. With the widespread use of visual information, there is a pressing requirement to efficiently manage, store, manipulate and retrieve visual information. The demand for flexible visual information management will be critical. In summary, some of the main characteristics of visual information management are the following.

1. The use of high performance computing platforms equipped with audio and video facilities, large disk storage and fast I/O for the effective handling of multimedia data, possibly based on multimedia chips or general-purpose chips with rich multimedia instructions, coupled with the use of high bandwidth networks together with high performance parrallel servers.

2. New storage and retrieval techniques are needed to be developed to cater the storage and real-time constraints of visual data. Disk stripping,

staging, migration, and hierarchically organized storage architectures are necessary and often needed to satisfy the constraints.

3. Visual information contents are derived from the raw data. Retrieval is mostly through the visual information contents. Inter- and intra-relationships of visual contents are important to characterize the very nature of different media contents.

4. Multimedia Database Management System is an essential element in Visual Information Management. It is quite different from the traditional database management because of the complexity to handle different visual data. A number of new issues, which cannot be found in the traditional databases, surface in multimedia databases and require new approaches to be addressed.

5. Flexible image, audio and video analysis and specification tools will be routinely available to the users. This will assist the users in formulating and refining visual queries for processing and information matching by the system.

6. The presentation and delivery of information is highly graphical. This in itself do not guarantee a system to be a VIS, as some conventional systems are also able to do this. Such presentation will involve the extensive use of visualization techniques, and will significantly enhance the value of usability of the underlying information.

7. A flexible query model involving different modes of inputs which supports both *visual information recovery* and *visual information discovery* will also be used. In additional to the common relational operators, a new set of spatial, temporal, and dynamic operators supported by visual languages are needed in composing queries.

8. The use of automatic and semi-automatic content-based indexing, retrieval, and feature extraction techniques will form an integral part of the multimedia database.

9. Visual information systems will incorporate the strengths of traditional information systems. Visual information can enhance the traditional IS applications such as Decision Support System, Executive Support System, Management Information System, Transaction Processing System, Groupware System, Communication System, Data Warehouse and Data Mining.

The increasingly widespread adoption of multimedia platforms has made visual information more available. This also highlights the limitations of current information systems, which are primarily based on a GUI-assisted alphanumeric paradigm. Although certain forms of visual information processing systems already exist, they tend to be restricted to highly specialized applications such as medical imaging and engineering design. As human end users form an important part of an information system, and as they often prefer non-alphanumeric multimedia interaction, traditional information systems are expected to be evolved into visual information systems. VIS will not only allow more natural interaction between human

and machine, but will pervade all applications areas and substantially increase the functionality and effectiveness of information systems, opening up a new horizon of previously untapped information sources.

Chapter 3

Evaluation and Framework for Image Retrieval Using Compressed Data

3.1 Introduction

Among different visual contents, image contents constitute a major component in Visual Information Systems. Uninterpreted visual data such as raw images take up considerable storage and difficult to retrieve. The ability to efficiently index, store and retrieve images based on their contents is crucial to the success of any visual information system. Over the years, many retrieval methods have been proposed and implemented ranging from color histogram, texture mapping, shape extraction, to fuzzy and neural implementations [NIBL93, OGLE95, PENT94 etc.]. Most of these techniques follow the general trend in image retrieval as shown in Figure 3.1. Images are analyzed by the pre-determined method. Meta-data are

produced either automatically, semi-automatically, or manually. These meta-data can be a set of feature vectors, keys or signatures extracted from the images. Textual information together with attributes may also be included into the meta-data to form the retrieval information. Relational, object-oriented, object-relational databases, tree-based or custom-made data structures are generally used to house all these data. Compression can then be used to reduce the storage occupied by the original images. To facilitate browsing, thumbnail images are often generated. Although only a single retrieval method depicted in Figure 3.1 is illustrated, in practice, it is possible to have a combination of methods to form the retrieval scheme.

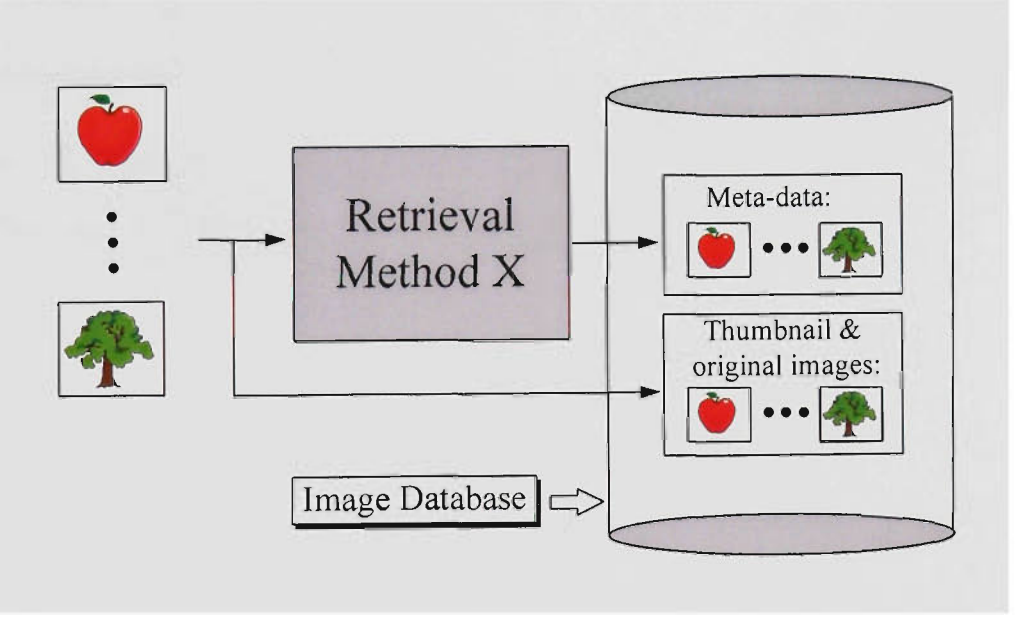


Figure 3.1 General Approach to Image Retrieval

Conceptually, the objective in image retrieval is to provide some sort of indices which can represent the details in a more organized fashion. The information used in the representation should be reasonably compact so that indexing can be performed. Figure 3.2 illustrates the general approach to the indexing and retrieval

problem. In most cases, the user does not usually have to know the intermediate representations as they are mainly performed by machines. However, the semantics of the query model and also the underlying data model will need to be understood by the users in order to effectively retrieve images based on their contents [GUDI95]. The intermediate representations generated from user query may not always precisely match with the intermediate representations generated from the image data. Similarity measure or approximation matching is often needed.

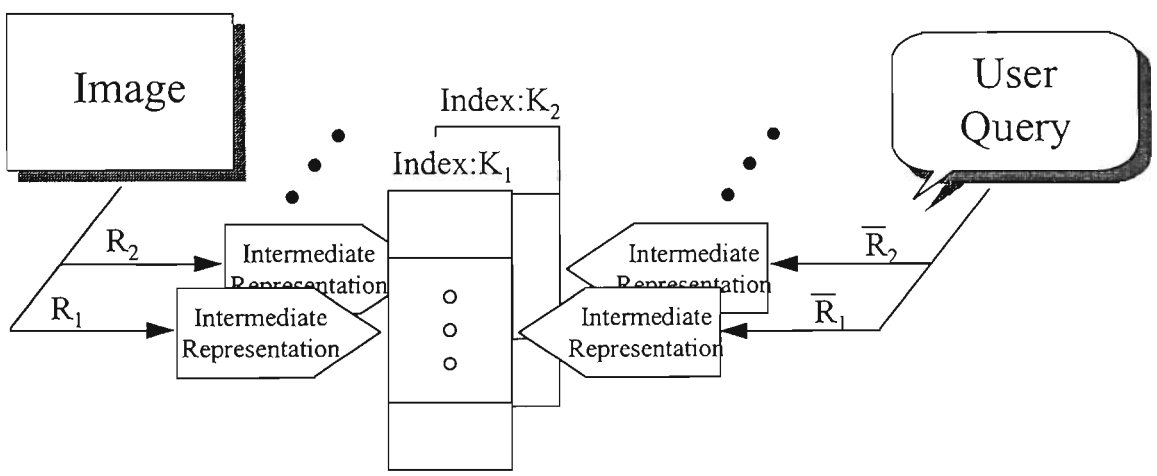


Figure 3.2 Image Indexing and Retrieval

Image indexing and retrieval techniques can be classified into two main categories; 1) spatial domain techniques, and 2) compressed domain techniques. Traditionally, the main stream approach to image indexing and retrieval is in spatial domain. They extract and produce indexing and retrieval information directly from the images. Color histogram, texture classification, and shape extraction are some of the examples in spatial domain. Recently, compressed domain techniques are emerging as

promising alternative to the indexing and retrieval problem. Instead of producing indexing and retrieval information directly from the images, such information is extracted from the compressed data. The obvious advantage to this approach is that many images are already stored in compressed formats. It eliminates the needs to decompress images for indexing purposes. One disadvantage to this approach is that the characteristics of the indexing information are mainly governed or restricted by the compression techniques themselves. For example, some invariant properties are not always achievable since the formation of compressed coefficients may hinder the translational, rotational, scaling, or color invariant properties. Other advantages and disadvantages of using compressed data for image retrieval will be thoroughly evaluated in subsequent sections.

In this chapter, a framework for image indexing and retrieval using compressed data is provided [SO98]. In order to analyze some of the indexing and retrieval methods in compressed domain, it is necessary to provide a thorough treatment of the most popular techniques in data and image compression. Therefore, the taxonomy of image compression techniques is outlined in Section 3.2. It was followed by the algorithms of various compression schemes. The descriptions are sufficiently comprehensive so as to provide a better understanding of different approaches in image compression. Three representative examples of using compressed data based on DCT, VQ and wavelets for the purpose of image indexing and retrieval are provided in Section 3.3. A comparative evaluation is provided in Section 3.4. The common approach is identified and the desirable characteristics are expatiated. Problems associated with this approach are also discussed.

3.2 A Taxonomy of Image Compression Techniques

The root of data compression can be traced back to the late 1940s when Claude Shannon at Bell Labs made a significant advance in the field of Information Theory. Practical compression techniques have been around for over half a century. The classic paper, which initiated the whole new avenues of data compression, was published by D.A. Huffman in 1952 and the associated technique has since come to be known as Huffman Coding. Originally, the advances of data compression were made for the coding of symbols such as texts. They were not specifically designed for image data. Since image data tends to exhibit a high degree of correlation, specialized methods have been developed over the years. Among all the image compression techniques, they can be categorized into two groups [DASA95]:

- Distortionless, or error-free, data compaction
- Error-prone, or lossy, data compression

Error-free data compaction permits a reconstruction resulting in an exact reproduction of the original image. These techniques are referred to as *lossless*, information-preserving or reversible methods of data compression. The existing techniques for the coding of symbols are often used for lossless image compression. However, a number of lossless techniques are reported in recent years which exploit the correlated nature of image data and specialize the compression schemes solely for images.

Under the category of error-prone data compression, the cost of the compression achieved includes not only the cost of the processes of compression and

decompression, but also a cost in terms of some distortion of the original image. These classes of compression schemes are referred to as *lossy*, non-information-preserving, or irreversible methods. Lossy compression with high compression ratio is acceptable in this instance as long as the compression images are legible. But, in some image databases (e.g. medical imaging applications), the undistorted full images must be kept for the final retrieval.

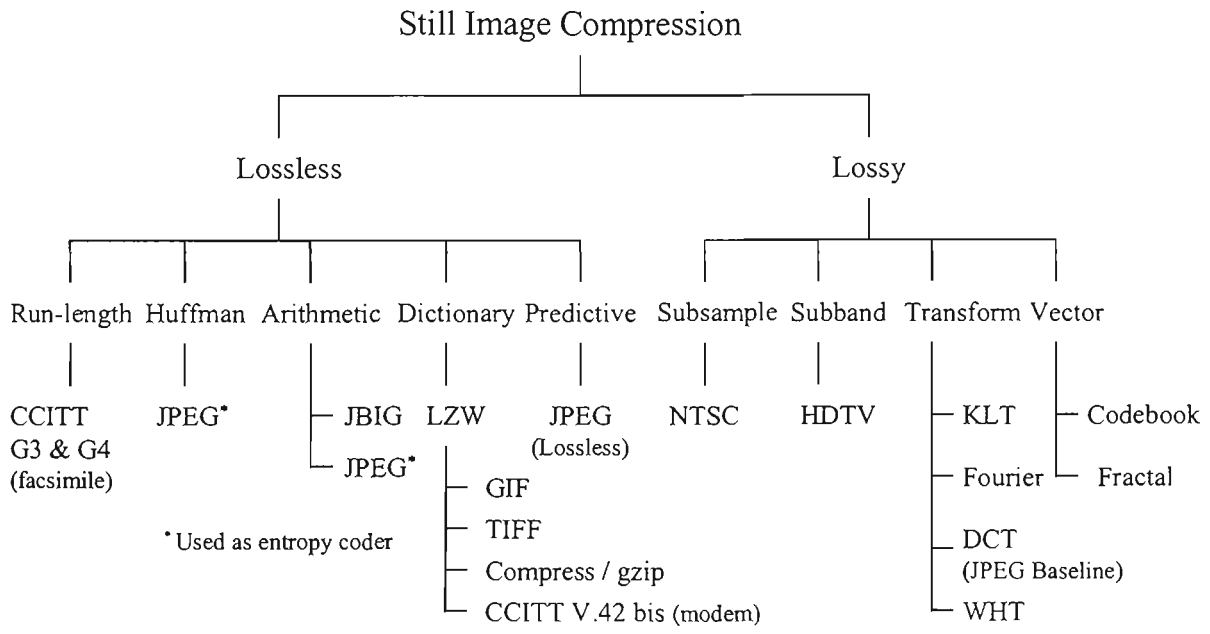


Figure 3.3 A Taxonomy of Image Compression Methods

There are many techniques existed for lossless and lossy compressions. Huffman Coding, Arithmetic Coding, Predictive Coding and Ziv-Lempel Coding are some of the lossless techniques. DCT (JPEG), Fractals, Wavelets, Vector Quantization (VQ) and Block Truncation Coding (BTC) are the major schemes in

lossy image compression today. Figure 3.3 illustrates the taxonomy of image compression methods.

By virtue of the complexities of all these compression methods and their variations, we intend to describe the fundamentals of the algorithms only. The implementation details are largely ignored. This approach is to provide a better understanding and insight of the algorithms and will be adopted for all the coding schemes described in this chapter.

3.2.1 Distortionless Data Compaction

Run-length Coding

Run-length coding is a simple and reversible technique for data compression. A run-length coder replaces a sequence of repeating symbols by the length of the run; once the sequence of identical symbols reaches a pre-defined level and such the replacement is advantageous to the overall compression. The idea is particularly suitable for black and white pixels such as Facsimile.

To illustrate run-length coding, we will consider two cases; 1) text compression used by IBM SNA™ (System Network Architecture) called the HDC (Hardware Data Compression) algorithm [HOFF97], and 2) facsimile compression standard specified by CCITT (now known as ITU-T) Group 3 and Group 4 - Recommendations T.4 and T.6 [GONZ92, JAIN89, SAYO96].

HDC (Hardware Data Compression)

The rules for the HDC algorithm are defined as follows:

- A sequence of consecutive blanks between 2 and 63 bytes long are substituted by a single control byte signifying the number of blanks.
- A sequence of consecutive characters other than blanks between 3 and 63 bytes long are substituted by 2 bytes with the first byte being the control bytes signifying the number of characters and the second byte containing the copy of the character.
- A sequence of non-repeating characters between 1 and 63 bytes long are prefixed by a single control byte signifying the number of non-repeating characters.

An example of the HDC algorithm is shown in Figure 3.4. Initially, the first eight blanks(b) are compressed into a single control byte, C_1, signaling the number of blanks is eight. Then the control byte, C_2, designates a string of 12 uncompressed characters, HA!bCOMPRESS, to be followed. Next in the compressed sequence is a single control byte, C_3, signaling the number of blanks is two. Then the control byte, C_4, designates a string of 2 uncompressed characters, MM, to be followed. Finally, eight consecutive E's are replaced by a control byte, C_5, and the letter, E. C_5 signifies the number of Es is eight.

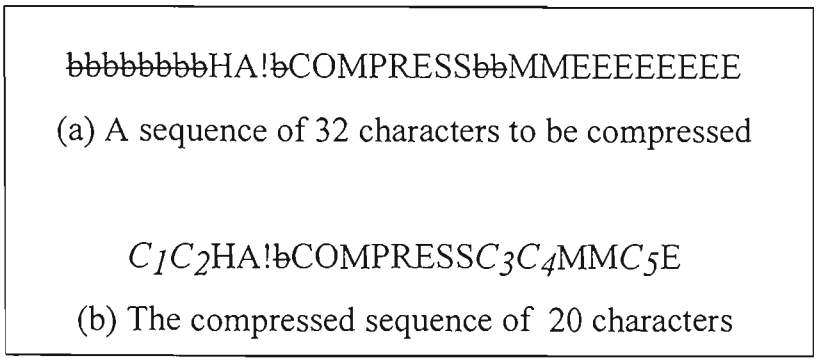


Figure 3.4 Example of the HDC Run-length Coding

Facsimile Coding Standards - CCITT Group 3 and 4

The CCITT has issued four recommendations to the Facsimile Coding Standards. Group 1 (T.2) and Group 2 (T.3) are using analog schemes to transmit documents over the phone lines. Group 3 (T.4) and Group 4 (T.6) are using digital schemes together with data compression to losslessly transmit a bitonal image over the phone lines. The recommendation for Group 3 has two coding schemes; a one-dimensional scheme and a two-dimensional scheme. One-dimensional scheme is to treat the image as a sequence of scan-lines and the coding of each line is performed independently. The two-dimensional scheme makes use of the pixel values in the previous line as the predictors for the current line. Group 4 drops the one-dimensional scheme and simplifies the two-dimensional schemes for more efficient transmission. The characteristics of CCITT Group 3 and Group 4 facsimile schemes are outlined in Table 3.1. To illustrate the basic concept of the run-length coding in facsimile transmission, we will describe the one-dimensional scheme only.

Algorithms	ITU-T Rec.	Brief Description
Modified Huffman Coding (MH) [one-dimensional]	ITU-T T.4 (CCITT G3)	A run-length description is first obtained and then followed by Huffman coding (separate codes for black and white runs).
Modified READ Coding (MR) [two-dimensional]	ITU-T T.4 (CCITT G3)	The pixels in the previous line are used as the predictors of the current line. MH coding is periodically inserted to MR coding to prevent error propagation.
Modified-modified READ Coding (MMR) [two-dimensional]	ITU-T T.6 (CCITT G4)	Error-protection mechanisms are eliminated from MR coding to maximize the overall compression efficiency. Error-free transmission is to be guaranteed by the channel.

Table 3.1 CCITT Group 3 and Group 4 Facsimile Schemes

For each scan-line in the one-dimensional coding scheme in Group 3, a series of alternating white and black runs are generated. Figure 3.5(a) illustrates an example of a line in an A4-size document which should have 1728 pixels per scan-line. The lengths for the alternating white and black runs are shown in Figure 3.5(b). Note that the first run is always assumed to be a white run. If the first pixel is black, then the length of the first white run is zero. A variable-length coding scheme is then used to represent the run-lengths just to take advantage of different probabilities of various lengths. Since the probabilities of white and black lengths are also different, CCITT uses separate Huffman codes for them (Huffman coding will be described in the next section). However, the numbers of possible lengths can be very large (between 0 and 1728 for a A4-size document). It is not easy to build a table of Huffman codes with so

many entries. Therefore, rather than using a Huffman code for each run-length, it is expressed by a terminating code if the run length is less than 63. If the run-length is greater than 63, a make-up code and a terminating code are used. The make-up codes are in multiple of 64 not exceeding the run-length. For example, a run-length of 129 is represented by the make-up code for 128 and the terminating code for 1. The Huffman codes for the example are shown in Figure 3.5(c). Note that the terminating codes for black and white runs (e.g. 4) are different. Also, a unique end-of-line (EOL) code word is inserted to signal the termination of the current line.

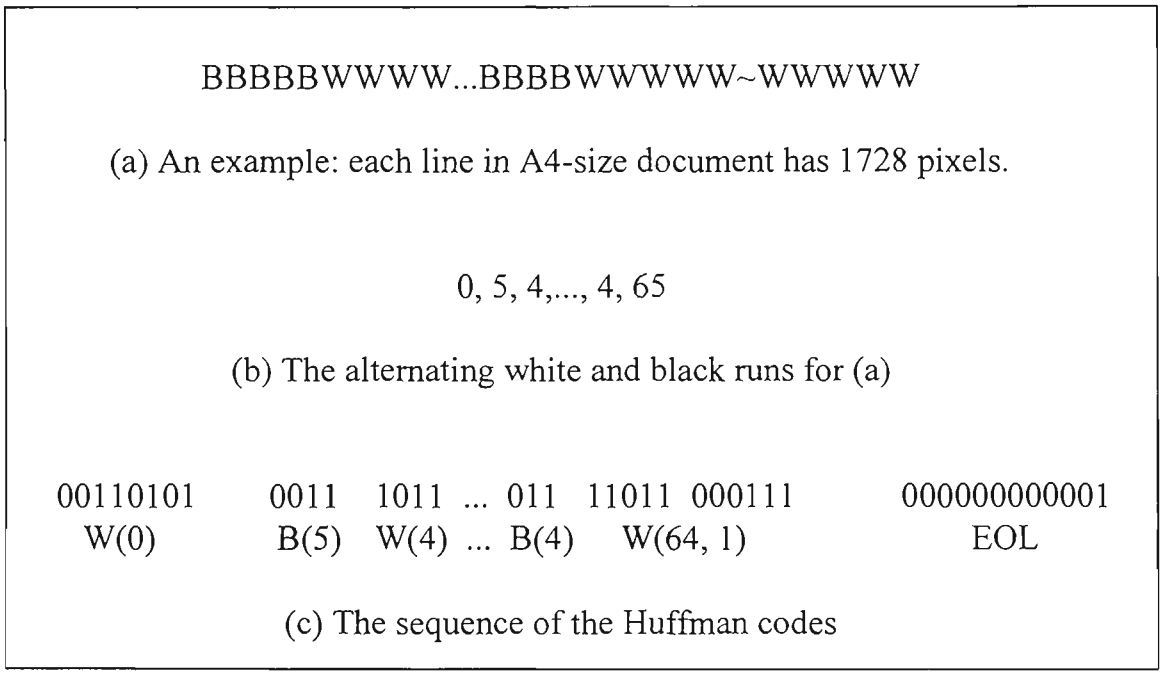


Figure 3.5 Example of the MH Coding (CCITT G3)

Huffman Coding

Huffman coding is one of the oldest methods in code construction with minimum redundancy. It was originally developed by David Huffman in 1952 as part

of a class assignment in the area of information theory; the first ever class taught by R. M. Fano at MIT [SAYO96]. Fano was also the inventor of the first well-known coding method commonly referred to as Shannon-Fano coding [NELS92]. Huffman coding belongs to the category of statistical coding. By knowing the probability of each symbol in a message, we can, ideally, construct a table of codes with the following properties:

1. Variable-length codes: different codes have different number of bits, but must be uniquely decoded. This is also referred to as the *prefix codes*. A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword [COVE91].

2. Symbols that occur more frequently should be represented by codewords with fewer bits than the symbols that occur less frequently. This is obvious if we want to compact the data.

Before we go on to describe the basic concept of Huffman coding, we need to iterate some of the basic definitions in information theory. Let consider a source A that generates random symbols $a_1, a_2, \ldots, a_n$. The probability of the occurrence for symbol a_i is $P(a_i)$, and such that,

$$\sum_{i=1}^{N} P(a_i) = 1 \tag{3.1}$$

The *self-information* for symbol a_i is then defined as

$$I(a_i) = \log \frac{1}{P(a_i)} \tag{3.2}$$

The *average self-information* or *entropy* of the source A is given by

$$H(A) = \sum_{i=1}^{N} P(a_i)\, I(a_i) = \sum_{i=1}^{N} P(a_i) \log \frac{1}{P(a_i)} \tag{3.3}$$

Note that the log is to the base 2 and entropy is measured in bits. Also, if $n(a_i)$ is the number of bits to code the symbol a_i, then the average codeword length is defined as

$$L_{avg} = \sum_{i=1}^{N} P(a_i)\, n(a_i) \tag{3.4}$$

Symbol a_i	Count	$P(a_i)$	$I(a_i)$	Code	$n(a_i)$	Total bits
E	3	1/4	2	10	2	6
D	2	1/6	2.585	001	3	6
S	2	1/6	2.585	010	3	6
C	1	1/12	3.585	011	3	3
O	1	1/12	3.585	110	3	3
M	1	1/12	3.585	111	3	3
P	1	1/12	3.585	0000	4	4
R	1	1/12	3.585	0001	4	4

Table 3.2 Information Contents and Codes for "DECOMPRESSED"

To illustrate the concept, let us consider the following sequence of symbols

DECOMPRESSED

We have a source with 8 distinct symbols { E, D, S, C, O, M, P, R }. Let assume we have only these symbols and the probability of each symbol corresponds to the

occurrence in the sequence. Table 3.2 summarizes the information contents and suggests a code table for the symbols. The entropy and the average codeword length are given in Eqs 3.5 and 3.6 respectively. The codes given in the table are in fact one of the Huffman codes which we will describe next.

The average self-information or entropy is

$$H(A) = \frac{1}{4} \times 2 + \frac{1}{6} \times 2.585 \times 2 + \frac{1}{12} \times 3.585 \times 5 = 2.855 \tag{3.5}$$

The average codeword length is

$$L_{avg} = \frac{1}{4} \times 2 + \frac{1}{6} \times 3 \times 2 + \frac{1}{12} \times 3 \times 3 + \frac{1}{12} \times 4 \times 2 = 2.917 \tag{3.6}$$

Note that the average codeword length is very close to the theoretical bound for this example. If we use a fixed-length coding scheme to represent the sequence such as the standard ASCII characters, we need 96 bits in comparison to 35 bits using the variable-length Huffman coding scheme.

Huffman is one of the best known method to construct the optimal prefix code if the probability of each symbol in the source is given. Now let us look at the algorithm to construct the Huffman code:

1. Arrange the symbols in decreasing order according to their probabilities.
2. Merge the two symbols with the smallest probability to form a new symbol having the sum of two probabilities.
3. Repeat Step 1 and 2 until there is only one member.

To see how it is done, we refer to our example and generate the steps in Figure 3.6. For each step, we merge the two symbols with the smallest probability to form a new

probability for the merged symbol. We arbitrarily assign 0 and 1 to each pair. In this way, the codewords corresponding to these two symbols will differ by this bit. The list of the symbols with one less than the previous step is further sorted and the process is repeated. The code of each symbol can be traced from the root node to the individual symbol. For example, the code for 'E' will be 1 - 0 - - - - (i.e. the code '10'). Therefore, the sequence of "DECOMPRESSED" will be encoded as "00110011110111000000011001001010001" using our derived Huffman codes.

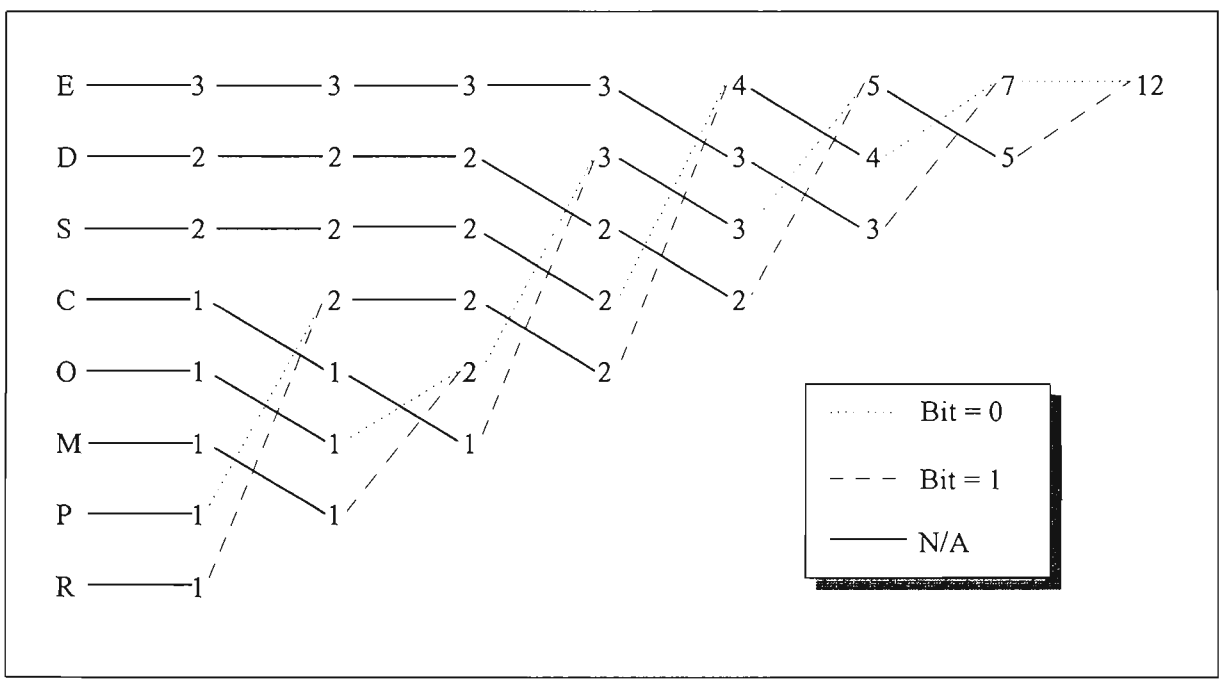


Figure 3.6 Generating Huffman Codes for "DECOMPRESSED"

Given a table of Huffman codes, there are two popular ways to decode any arbitrary length of encoded messages:

- Implemented by a binary tree - Since the Huffman codes are prefix codes, we can construct a binary tree with the leaf nodes corresponding to the

symbols. The encoded message can be parsed one bit at a time. The traversal of the tree is dictated by the arrival of bits until the leaf node is reached. The symbol is then output and the process is repeated until the message is exhausted. Note that this implementation is not very efficient as each bit required the traversal of one link in the tree.

- Implemented by a table - A table can be constructed with 2^N entries where N is the number of bits in the longest Huffman code. For example, the longest code is 4 for our previous codes (i.e. 'P' and 'R'). All the entries in the table will be filled and duplicated by the symbols having the prefix code (i.e. all the entries with '10xx' will be filled with 'E'). The decoding process is then by reading N bits at a time. Since any combination of the actual code will point to the same symbol, the symbol can be quickly located. If the retrieved symbol has only L bits, the last N-L bits of the current N bits are retained to form the next N bits for next round. Readers may already realize that it is advantageous to generate Huffman codes with minimum variance. Otherwise, the number of entries will increase significantly. The way to generate minimum variance Huffman codes is to put the combined probability as high as possible in the list. Figure 3.6 is an example of this.

Since the probability of each symbol affects its length, it is sometime impractical to generate a table with large entries. Constrained or modified Huffman coding such as the CCITT G3 described previously are commonly used. Also, the algorithm of Huffman coding depends on the probabilities of the symbols to be known

a priori. Two passes (collection of statistics and the actual encoding) are required. To combine the two passes into one, Adaptive Huffman coding schemes are developed to generate an encoded message based on the statistics of the symbols already encountered [NELS92, SAYO96].

Arithmetic Coding

The major problem of Huffman coding is that symbols must be coded by integral number of bits. Hence, the theoretical optimal can not be achieved unless the probabilities of all symbols are in the integral powers of 1/2. Arithmetic coding is a better statistical coder by avoiding the need to translate symbols into specified codes. It treats the entire input stream of symbols as one unit and replaces the symbols with a single floating-point number. More digits (or bits) are generally needed to extend the precision of the floating-point number as the stream is getting longer. This floating-point number is in the interval between 0 and 1 which can uniquely represent the entire message.

The main idea behind arithmetic coding is fairly simple and we will illustrate the algorithm through an example. Let assume $A = \{ a_1, a_2, a_3, a_4 \}$ is the alphabet for a discrete source with $P(a_1) = 0.5$, $P(a_2) = 0.2$, $P(a_3) = 0.1$, and $P(a_4) = 0.2$. At the start of the encoding process, each symbol is assigned a portion of the interval between 0 and 1 according to its probability. The cumulative probabilities are used so that the sub-intervals are disjointed. Table 3.3 shows one such arrangement. The half-open sub-interval represents the lower and upper limits of the symbol. For non-adaptive arithmetic coding, these proportions are fixed through out the coding process.

Symbol a_i	$P(a_i)$	Range
a_1	0.5	[0.0, 0.5)
a_2	0.2	[0.5, 0.7)
a_3	0.1	[0.7, 0.8)
a_4	0.2	[0.8, 1.0)

Table 3.3 Example of Static Model for { a_1, a_2, a_3, a_4 }

Now, imagine we have a sequence beginning with $a_2\ a_3\ a_4\ a_1\ a_3$. We would encode the sequence using non-adaptive arithmetic as shown in Figure 3.7.

After seeing the first symbol a_2 in the sequence, any floating number in [0.5, 0.7) will uniquely decode the symbol. Once this range is determined, the subsequent symbols will further restrict the range using the same proportions of the original subdivision. Figure 3.7 illustrates the narrowing process of our example. To uniquely represent the first five symbols, any floating point number in [0.6564, 0.6576) will suffice. The mid-value or the lower bound of the range is a popular choice for such a number. It is necessary, however, for the decoder to know the end of the sequence. Therefore, a special end-of-file symbol is needed to append at the end of the input sequence so that the decoder can properly terminate the decoding process.

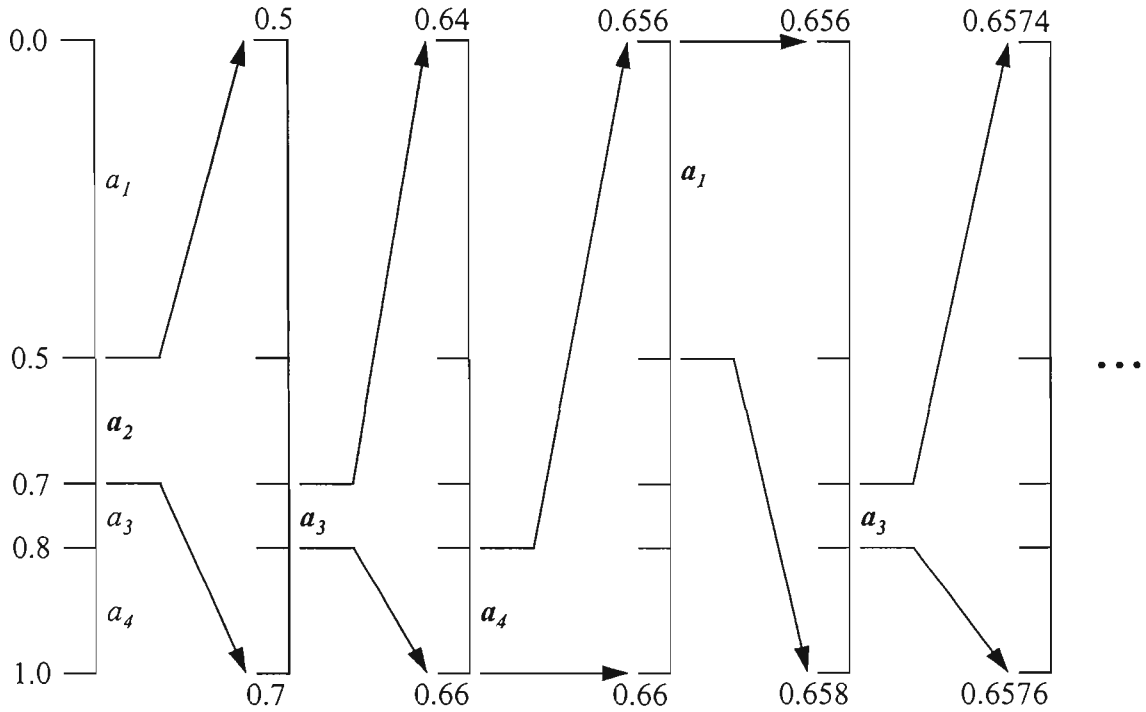


Figure 3.7 Non-adaptive Arithmetic Coding with Input Sequence; $a_2\ a_3\ a_4\ a_1\ a_3$...

Given any floating point number, the sequence can be traced by narrowing the ranges. For example, the decoder can immediately deduce the first symbol is a_2 given any floating number in [0.6564, 0.6576) since it completely lies within [0.5, 0.7). It is also within [0.64, 0.66) to be decoded as a_3 and so forth. At the first glance, it is conceptually possible to encode a sequence using this algorithm. However, because of the finite precision in floating-point arithmetic, it is impractical to implement without any modification in producing the floating point number. Another problem to this algorithm is that one has to encode the entire sequence of symbols

before any release of the compressed bit stream. Therefore, it is necessary to modify the original algorithm so that it has the following desirable properties:

- *Fixed-precision Arithmetic*

  Using integer arithmetic with finite precision of 16 or 32 bits, one can use simple operations such as shifts and additions which is much better than floating-point arithmetic. For example, an implicit decimal point can be assumed on the left side of the integer. One can then have a fixed-precision to represent a floating point number. Since it will eventually run out of precision, the value is required to rescale.

- *Incremental Encoding and Decoding*

  Rather than waiting for the entire sequence to be processed before transmitting the first encoded bit, it is desirable to transmit portions of the code as the sequence is being observed. The decoder, on the other side, can start the decoding process as soon as sufficient codes are received.

The concept of encoding sequence using floating-point arithmetic has been known for some time. Practical implementations using fixed-precision arithmetic and operating incrementally were discovered in 70s. It was based on the fact that once the most significant digits in the limits are the same. Any subsequent symbol will not alter their values. For example, the subinterval range after the second symbol, $a3$, is always started with the value 0.6 and will not change for the rest of the sequence. Hence, we can transmit the most significant digits once the upper and lower limits are the same. To recover the precision, the values can then be rescaled by shifting to the

left. Although we use an example in decimal, the principle is the same in binary and the decimal can be implicit. Once the transmitted bits are sufficient enough to decode, such as 0.6 in our example, the decoder can immediately determine the first symbol is a_2 since it is in [0.5, 0.7).

A number of design issues such as the problems of overflow and underflow in arithmetic are thoroughly described in [BELL90, WITT87]. The integer implementation can also be found in these references. Since the prescribed probability for each symbol is not changed throughout the sequence, the proportions of the subintervals are fixed. Adaptive model can also be used by allowing the variation of cumulative probabilities after each symbol is observed.

Predictive Coding

The concept of predictive coding is very simple. If the data is highly correlated such as images, we can predict the value from its neighboring data. The difference between the predicted value and the actual value is called the prediction residual. All we need to keep is the residual in order to faithfully reproduce the original data. Since the range and the variance of the residual are generally much smaller than the original data, the entropy should be lower and, hence, better compression can be achieved. We will illustrate the predictive coding using the lossless mode of JPEG as an example. A more detailed treatment of the JPEG compression schemes will be given in Section 3.2.2.

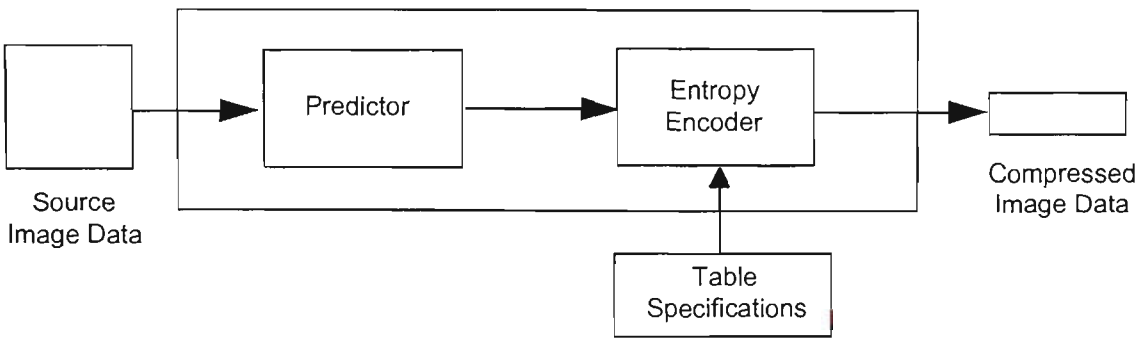


Figure 3.8 The Encoding Steps for the Sequential Lossless Mode of JPEG

The overall processing steps for the sequential lossless mode are shown in Figure 3.8. The neighboring pixels are used to form the prediction of the current pixel. This prediction is then subtracted from the actual value and the residual is encoded by either Huffman or Arithmetic coding.

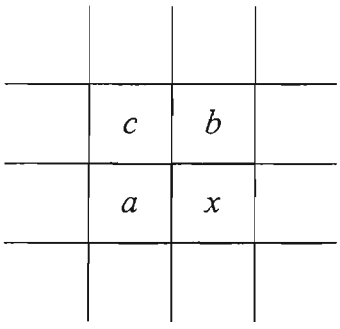


Figure 3.9 The Prediction Kernel

Figure 3.9 shows the prediction kernel for Figure 3.8. R_a, R_b, and R_c are defined to be the reconstructed pixels immediately to the left, immediately above, and

diagonally to the left of the current predicted pixel P_x. A total of 8 allowable predictors, one of which is selected in the header, are listed in Table 3.4.

Selection-value	Prediction
0	no prediction
1	$P_x = R_a$
2	$P_x = R_b$
3	$P_x = R_c$
4	$P_x = R_a + R_b - R_c$
5	$P_x = R_a + ((R_b - R_c) / 2)$
6	$P_x = R_b + ((R_a - R_c) / 2)$
7	$P_x = (R_a + R_b) / 2$

Table 3.4 Predictors for the Lossless Coding

Selection-value 0 should only be used for differential coding in the hierarchical mode of operation (see Section 3.2.2). Selections 1, 2, and 3 are one-dimensional predictors and selections 4, 5, 6 and 7 are two dimensional predictors. The first line of an image is always coded using the one-dimensional horizontal predictor (i.e. Selection 1) since there is no reconstructed pixels existed for the first line of the image.

The residual values are not directly encoded by Huffman or Arithmetic coding. They are calculated modulo 2^{16}. In the case of using Huffman coding for the entropy coding, it is expressed using *category* and *magnitude*. The table for the 17

categories (0 to 16) and the magnitudes can be found in the ISO DIS 10918-1 document [ISOD1, PENN93].

Ziv-Lempel Coding

Jacob Ziv and Abraham Lempel are two Israelli Scientists who developed two famous dictionary-based compression techniques in 1977 and 1978 and triggered a flood of research activities into the universal coding technique. A *universal code* is referred to a code that is designed to work with an arbitrary source distribution. The two techniques by Ziv and Lempel are commonly known as LZ77 and LZ78 [ZIV77, ZIV78].

LZ77 and LZ78 are two distinctively different approaches to adaptive dictionary compression. Both are very practical and easy to implement. LZ77 uses a sliding window technique. It assumes that a given pattern will occur close together. Hence, the dictionary is simply a portion of the previously encoded sequence. References are made within the window for the occurrence of the current coding sequence. After a year of the landmark paper published in 1977, Ziv and Lempel put forward a completely new approach for building a dictionary. Instead of using the sliding window over a fixed-length sequence, LZ78 builds a dictionary by adding each new entry using an existing dictionary entry concatenated with one new symbol.

LZ77 and LZ78 have given rise to a number of variations such as LZSS from LZ77 and LZW from LZ78. By far the most well-known modification is the LZW algorithm by Terry Welch in 1984 [WELC84]. It has been found in many modern

applications such as UNIX `compress`, GIF and V.42 bis. We will describe the algorithms of LZ77, LZ78 and LZW next.

LZ77

In the LZ77 approach, data previous seen is used as a dictionary. A sliding window of N characters composes of two parts: the *search buffer* with M characters and *the look-ahead buffer* with $N - M$ characters. A search buffer contains the recently encoded characters used as the dictionary. A look-ahead buffer contains the characters being encoded. Figure 3.10 shows an example of a sliding window with $N = 12$ and $M = 8$. In practice, the size of the sliding window is much larger and in the order of a few thousand characters.

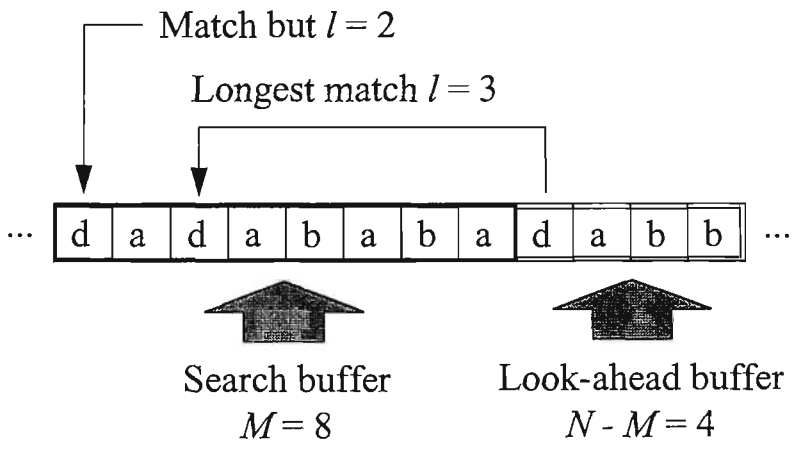


Figure 3.10 Sliding Window with $N = 12$

Initially, the search buffer is filled with some pre-determined characters such as spaces or zeros. To encode the sequence in the look-ahead buffer, the first M

characters are searched for the longest match. The longest match is then encoded with a triple $< o, l, a >$, where o is the offset from the look-ahead buffer, l is the length of the match, and a is the character that follows. In our example, the triple is encoded as $< 6, 3, b >$. The window is then "slid" to the right $l + 1$ characters.

Using this scheme, it is possible that the longest match is overlapped into the look-ahead buffer. It does not create any problem for the decoder if we impose the maximum length of the match to be N-M characters. Since only the first M characters are used, it is guaranteed that the longest match cannot be the look-ahead buffer itself. The reason to attach the explicit character in the triple is to provide a mechanism for the scenario that no match is found. In this case, $< 0, 0, c >$ is used where c is the first character in the look-ahead buffer.

LZ78

One year after the publication of LZ77, Ziv and Lempel developed a totally different scheme to adaptive dictionary compression. This scheme is commonly referred to as LZ78. The LZ78 algorithm abandons the use of search buffer and generates an explicit dictionary dynamically. The dictionary contains all the previously seen phrases. Each phrase is comprised of a previously seen phrase followed by an extra character. The decoder is thus output a double $< i, a >$ where i is an index of the previous phrase and a is the character.

We illustrate the encoding process using LZ78 by an example with the following sequence,

bbabbbabbaabbbbabb

Initially, the dictionary contains only a null phrase indexed by the value 0. The first character will be encoded as < 0, *b* > and the dictionary will now have a phase "*b*". The second character is a *b* which is found in the dictionary. Hence, the next character is appended to the phrase and formed a new phase "*ba*". The processing of the sequence is given in Table 3.5.

Output	Index	Phrase
	0	{}
<0, *b*>	1	*b*
<1, *a*>	2	*ba*
<1, *b*>	3	*bb*
<2, *b*>	4	*bab*
<2, *a*>	5	*baa*
<3, *b*>	6	*bbb*
<4, *b*>	7	*babb*

Table 3.5 The Output of our Example Using LZ78

It is obvious that the dictionary grows quickly if we do not restrict its size. This is due to that LZ78 asymptotically approaches to theoretical entropy as the size of the input increases. In practice, when the number of entries reaches a pre-defined

level, one option is to clear all entries and the dictionary is rebuilt. Another option is to stop adding new phrases once it is full.

LZW

LZW is the most important modification of LZ78. It dispenses the necessity of an explicit character in the double < *i*, *a* >. The encoder sends only the index of the dictionary. This is accomplished by initially filling the dictionary with every character in the character set. The input sequence is then parsed into a new phrase containing the longest seen phrase concatenated with a character. This last character of the new phrase is used as the first character of the next phrase. Only the index of the known phrase is transmitted.

Let us examine the LZW algorithm using the example in the previous section. Assuming we have only two characters { *a, b* } in the alphabet. The dictionary is initialized with two entries as shown in Table 3.6. In practical situation, 256 entries corresponding to the ASCII set are initialized. Initially, the new phrase parsed in is "*bb*" since it composes of the existing known phrase "*b*" concatenated with the character, *b*. Therefore, the index, < 2 >, corresponding to "*b*" is output and the new phrase "*bb*" is added to the dictionary to form the index < 3 >. The next phrase parsed in is "*ba*" which is again corresponding to "*b*" concatenated with the character, *a*. The index, < 2 >, is output and the new phrase "*ba*" is added to the dictionary to form the index < 4 >. The complete output of the encoding sequence is given in Table 3.6.

Output	Index	Phrase
	1	*a*
	2	*b*
< 2 >	3	*bb*
< 2 >	4	*ba*
< 1 >	5	*ab*
< 3 >	6	*bbb*
< 4 >	7	*bab*
< 3 >	8	*bba*
< 1 >	9	*aa*
< 5 >	10	*abb*
< 6 >	11	*bbba*
< 10 >	12	-

Table 3.6 The Output of our Example Using LZW

In order to faithfully reproduce the sequence, we must construct the dictionary in the same way as the encoder. For example, the output, 2 2 1 3 4 3 1 5 6 10, is parsed one index at a time. The current phrase is concatenated with the first character of the next phrase and the new phrase is added to the dictionary. The first index is 2 which corresponds to "*b*". This phase is then concatenated with the first character of the next phase which happens to be *b* again. So we add an entry with the new phrase "*bb*" into the dictionary. The process is then repeated.

The decoding process described above is very accurate with one complication. There is a situation where the next phrase has not been constructed in the dictionary so the first character of that phrase must be interpreted. To see how this happens, let assume we have another sequence, *babbababa*. The output from the encoder is 2 1 2 3 6. The decoding process begins with the first phrase which is "*b*". It concatenated with the first character of the next phrase which is the character, *a*. A new phrase, "*ba*", is added to the dictionary. The process is continued until it reaches the index 3. We know it corresponds to "*ba*". However, when we construct the new phrase from the next index, we found that the entry referred by 6 is not yet defined. Although we may not have this entry, we do know that it must start with *ba*. Hence, we can interpret this phrase starting with the character, *b*, and the new phrase would then be "*bab*".

3.2.2 Irreversible Image Compression

DCT (JPEG)

JPEG, or Joint Photographic Experts Group, is a collaborative effort between CCITT (International Telegraph and Telephone Consultative Committee) and ISO (International Standards Organization). This group has been working to establish an international compression standard for both color and gray-scale still images since 1986. As JPEG committee includes members from both CCITT and ISO, the committee made all technical decisions on the basis of unanimous agreement and carried out the selection process by competitive contests. JPEG reached a consensus

from twelve proposals that the Adaptive Discrete Cosine Transform (ADCT) approach for the lossy compression and the Different Pulse Code Modulation (DPCM) for the lossless compression are the best. Two published documents in 1992 (ISO DIS 10918-1 and ISO DIS 10918-2) [ISOD1, ISOD2] became the milestone of JPEG Compression Standard. Part 1 of ISO DIS 10918 describes the requirements and guidelines of JPEG. Part 2 is the compliance testing for Part 1 which sets out tests for determining whether implementations comply with the requirements for the various encoding/decoding processes. In August 1994, JPEG published its extensions (ISO DIS 10918-3) [ISOD3], which provides the requirements and guidelines for coding/decoding extensions to processes defined in Part 1. Interested readers should refer to Pennebaker and Mitchell [PENN93] for the in-depth description of JPEG International Standard. The complete ISO DIS 10918-1 and ISO DIS 10918-2 documents can also be found in the Appendix A and B of the book respectively, and the short technical paper by Gregory Wallace [WALL92] offers an excellent introduction. Briefly, JPEG offers four modes of operation:

1. Sequential DCT-based encoding: each image component is encoded in a single left-to-right, top-to-bottom scan;

2. Progressive DCT-based encoding: the image is encoded in multiple scans by either spectral selection or successive approximation so that the decoded image builds up in multiple coarse-to-clear passes;

3. Lossless encoding: the image is encoded using predictive coding technique to guarantee an exact reproduction; and

4. Hierarchical encoding: the image is encoded at multiple resolutions so that it provides for progressive coding with increasing spatial resolution between progressive stages. Upsampling by interpolation to increase the spatial resolution is necessary at each hierarchical stage.

The DCT-based sequential compression method is the most popular one. Figure 3.11 shows the key steps for all encoding processes based on DCT. Briefly, an input image is broken into 8x8 blocks of pixels. All 64 points are shifted from the range $[0,2^P-1]$ to the range $[-2^{P-1}, 2^{P-1}-1]$ and then transforms into the frequency domain using Forward Discrete Cosine Transform (FDCT) as shown in Eq. 3.7. The resulting 64 coefficients are uniformly quantized using a 64-elements quantization table. This is the step which makes this encoding mode lossy.

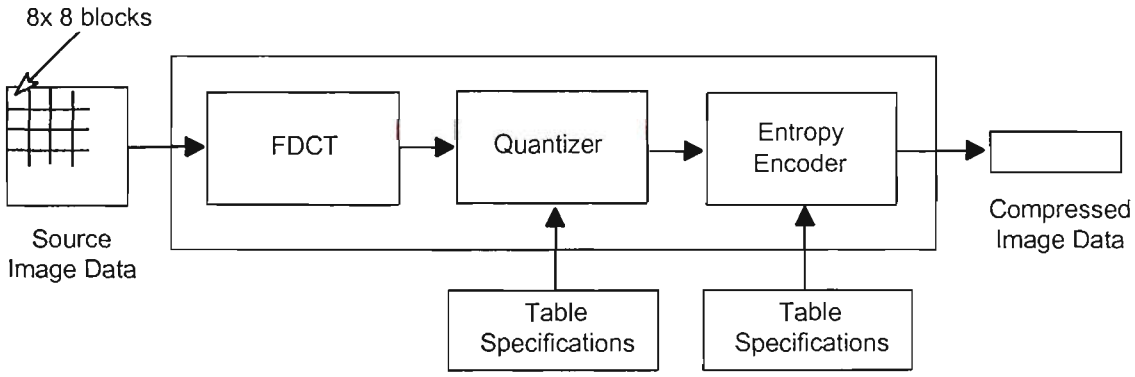


Figure 3.11 DCT-Based Encoder

8x8 Forward DCT (FDCT):

$$F(u,v) = \frac{1}{4} C(u)\, C(v) \left[\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \right] \qquad (3.7)$$

where $0 \le u, v \le 7$, $C(0) = 1/\sqrt{2}$, and $C(i) = 1$ for $1 \le i \le 7$

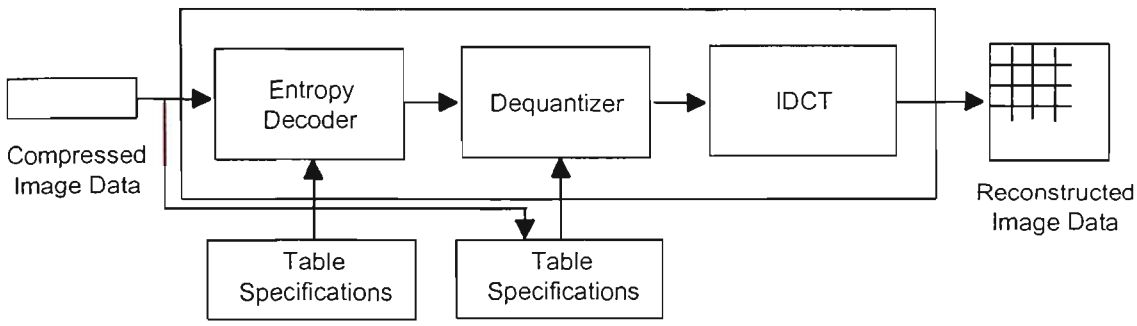


Figure 3.12 DCT-Based Decoder

8x8 Inverse DCT (IDCT):

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)\,C(v)\,F(u,v)\,\cos\frac{(2x+1)u\pi}{16}\,\cos\frac{(2y+1)v\pi}{16}\right] \tag{3.8}$$

where $0 \le x, y \le 7$, $C(0) = 1/\sqrt{2}$, and $C(i) = 1$ for $1 \le i \le 7$

After quantization, the DC coefficient and the 63 AC coefficients are prepared for entropy encoding, as shown in Figure 3.13. The DC coefficient (0,0), which represents the mean pixel value for each block, is encoded as the difference from the DC term of the preceding block in left-to-right, top-to-bottom scan order. The remaining 63 coefficients are converted into a one-dimensional "zig-zag" sequence. Each non-zero AC coefficient is coded by the "runlength" of preceding zeros followed by its size and coefficient value. The sequence of data values are then entropy-encoded by either Huffman or arithmetic coding.

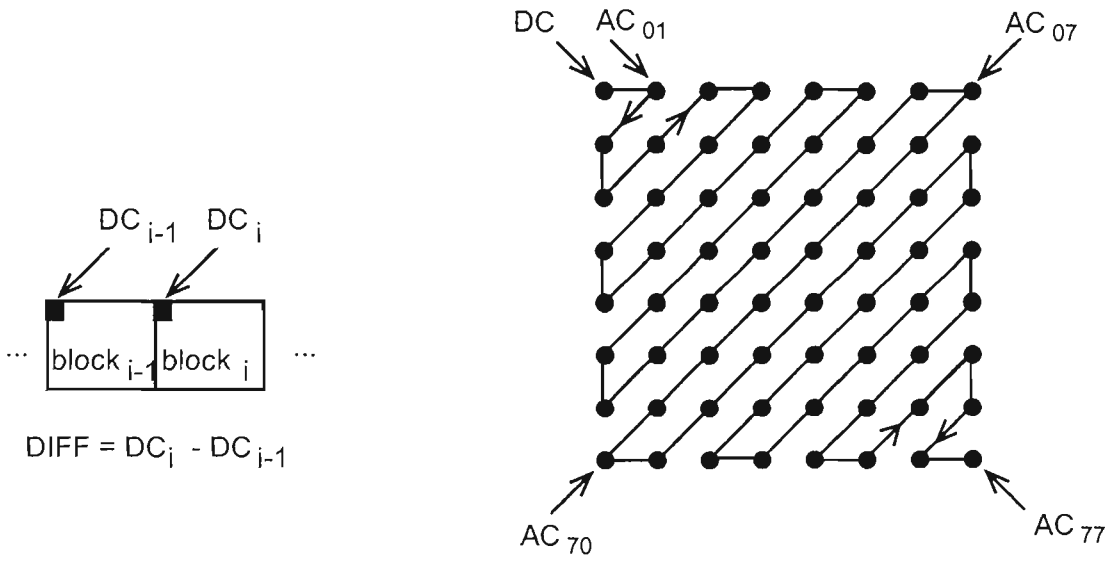


Figure 3.13 (a) Differenced DC Coefficients, and (b) Zig-zag Encoded AC Coefficients

For decoding, the process is just the opposite. The steps for the DCT-based decoding are shown in Figure 3.12 with the Inverse Discrete Cosine Transform (IDCT) given in Eq. 3.8. Since the computational cost for the encoding and decoding processes are roughly equal, the JPEG baseline codecs is considered as a symmetrical algorithm. Also, one of its attractions is that higher compression ratio does not require higher codecs time. To obtain higher compression factor only requires changing the quantisation table, while the computational cost remains the same [SO96a].

Although we are primarily dealing with still images, the work of MPEG is also relevant. MPEG-7 aims to create a standard for describing multimedia contents. Hence, it is formally named as "Multimedia Content Description Interface". The emphasis of MPEG-7 is to develop standard forms to represent and describe audio-visual information regardless of the types of storage, coding and the underlying

technologies. The work of the MPEG-7 committee is significantly shifted from the traditional work of encoding data streams such as MPEG-1 and MPEG-2 or even the object-based representations of MPEG-4. The overall standard has the following parts:

1. MPEG-7 Systems
2. MPEG-7 Description Definition Language
3. MPEG-7 Audio
4. MPEG-7 Visual
5. MPEG-7 Generic Entities and Multimedia Description Schemes
6. MPEG-7 Reference Software
7. MPEG-7 Conformance

The proposed MPEG-7 Standard is largely based on the concept of Descriptors (D) and Description Schemes (DS). A Description Definition Language (DDL) similar to XML is established to create Ds and DSs. Figure 3.14 outlines the scope of MPEG-7.

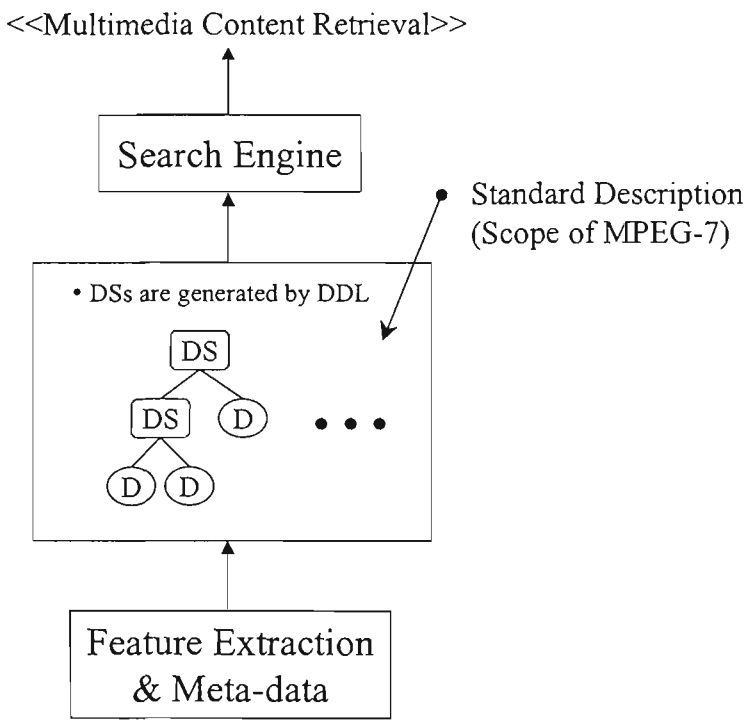


Figure 3.14 The Scope of MPEG-7

The scope if MPEG-7 is limited to standardize the descriptors and description schemes. A Descriptor is a representation of a Feature. A Descriptor defines the syntax of such representation. A Description Scheme specifies the structure and semantics of the relationships between its Descriptors and, possibly, its Description Schemes.

Fractals

The main idea of fractal compressions is to find the affine transformations of an image which can satisfy the so-called Collage Theorem developed by Michael Barnsley [BARN88, BARN93]. The Collage Theorem states what an Iterated Function System must be like in order to represent an image. The amazing property of an Iterated Function System is that a unique image emerges from an arbitrary image if the IFS are contractive. The latent image is called the fixed point or attractor of the IFS. Mathematically[1], let A_0 be the initial image and $W = \bigcup_{i=1}^{n} w_i$ is the transformation functions. Then if we apply W repeatedly:

$$A_1 = W(A_0)$$

$$A_2 = W(A_1) = W(W(A_0)) = W^2(A_0)$$

$$A_3 = W(A_2) = W(W(A_1)) = W(W(W(A_0))) = W^3(A_0)$$

$$\vdots$$

$$A_n = W^n(A_0) \tag{3.9}$$

[1] For rigorous definitions and proofs, refer to [BARN93]

The Contractive Mapping Fixed Point Theorem states that if X is a complete metric space and W: X->X is contractive, then W has a unique fixed point A_∞ regardless of the initial A_0,

i.e. $$A_\infty = \lim_{n\to\infty} W^n(A_0) \quad \text{for any initial } A_0 \tag{3.10}$$

If we want to know whether W is contractive, we have to define a distance between two images such that, for any two points P_1 and P_2,

$$d(W(P_1), W(P_2)) < s\, d(P_1, P_2) \quad \text{where } 0 < s < 1 \tag{3.11}$$

There are many metrics (e.g. Haussdoriff metric) to choose from.

The hardest part is how to obtain the transformation functions which can produce the desired fixed-point image. The very first practical fractal image compression is by Arnaud Jacquin [JACQ92, JACQ93] who developed the Partitioned Iterated Function Systems (PIFS). The idea of PIFS is to partition an original image and looks for similar pieces that can be paired and related by an affine transformation of the form (for gray scale images):

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

where s_i controls the contrast and
o_i controls the brightness (3.12)

To encode an image f, we want to find a collection of maps $W = \bigcup^n w_i$ such that

$$f' \cong f = W(f') \cong W(f) = w_1(f) \cup w_2(f) \cup \ldots \cup w_n(f) \tag{3.13}$$

with $\delta(f', f)$ small.

Here is a simple illustrative example,

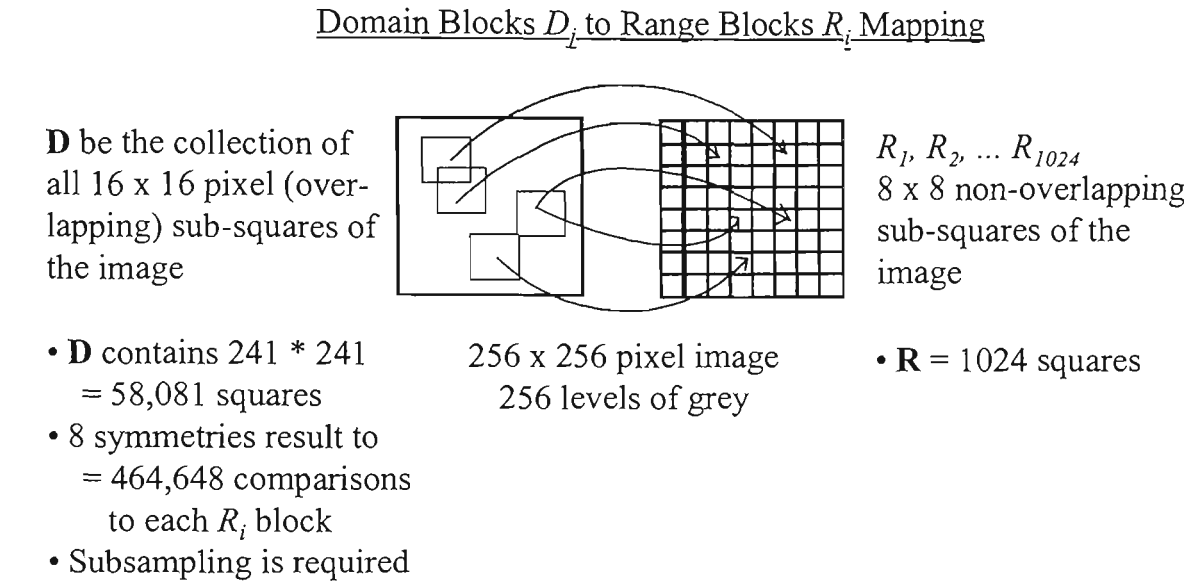


Figure 3.15 An Example of Fractal Compression

Although the example uses square partitions, this need not be the case as it could be triangulated, or decomposed into irregular rectangles as reported in advanced literatures [FISH92, FISH95]. The important point is that every pixel in non-overlapping range blocks (small blocks) is covered by one and only one block to ensure that the Collage Theorem applies. Some of the domain blocks (big blocks) may freely overlap.

Vector Quantization

The main idea of Vector Quantization in the context of image coding is a mapping from a k-dimensional Euclidean space R^k to a finite subset of R^k [NASS88]. This finite set Y is called a VQ codebook or VQ table. Figure 3.16 illustrates the block

diagram of a simple vector quantizer. The objective is to select an optimal codebook Y which results in the lowest possible distortion among all possible codebooks of the same size. By proper design of indexes and codebooks, we can explore the redundancy in an image so that the coding can be significantly more compact than the original image.

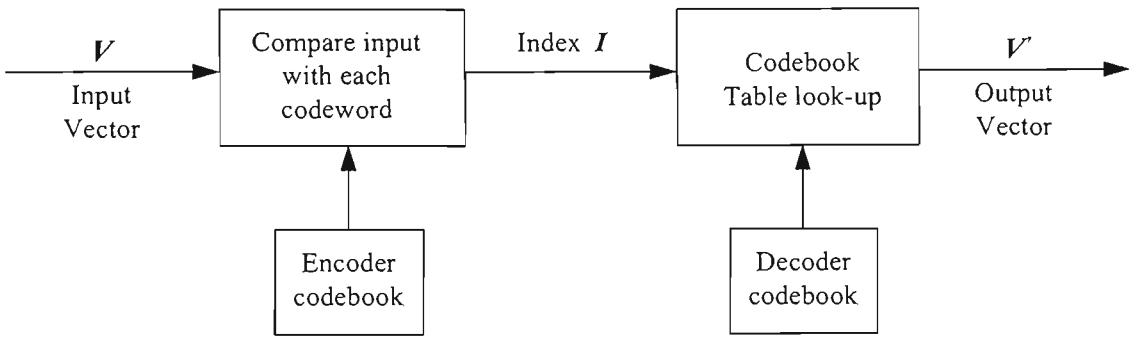


Figure 3.16 Block Diagram of a Simple Vector Quantizer

There are many codebook design methods. The first group of methods constructs codebook based on empirical data (training set). This approach is pioneered by Linde, Buzo, and Gray [LIND80], and their algorithm is now commonly referred to as the LBG algorithm. The second group of methods constructs codebook based on mathematical models. In this approach, it usually needs a probabilistic model of the image vector and requires a huge codebook for good performance.

Given a codebook, the encoding process is to search for the codevector in the codebook to find the best match (least distortion) for each input vector of an image. Some of the distortion measures are:

MSE $$d(V,V_i^{'}) = \frac{1}{k}\sum_{m=1}^{K}[V(m) - V_i^{'}(m)]^2 \quad (3.14)$$

MAE $$d(V,V_i^{'}) = \frac{1}{k}\sum_{m=1}^{K}\left|V(m) - V_i^{'}(m)\right| \quad (3.15)$$

L_n Norm $$d(V,V_i^{'}) = \left[\sum_{m=1}^{K}\left|V(m) - V_i^{'}(m)\right|^n\right]^{1/n} \quad (3.16)$$

Weighted $$d(V,V_i^{'}) = \sum_{m=1}^{K} w_m[V(m) - V_i^{'}(m)]^2 \quad (3.17)$$

or

$$d(V, V_i^{'}) = \sum_{m=1}^{K} w_m\left|V(m) - V_i^{'}(m)\right| \quad (3.18)$$

where $m = 1, 2, ..., K$, K = Dimension of the input vector, and $V(m) = m^{th}$ components of the input vector.

There are many variations of Vector Quantizers: Spatial Vector Quantizers (SVQ), Transform Vector Quantizers (TVQ), Tree Search VQ, Pyramid VQ, Finite State VQ, Predictive VQ, Entropy Constrained VQ, Sub-band VQ and Fine-Coarse VQ etc. I refer the interested readers to [NASS88], [GERS93], and [BARL96] (a special issue on VQ).

BTC

Block Truncation Coding was first published by Delp and Mitchell [DELP79]. The basic algorithm of BTC used in [DELP79] is to segment an image into small blocks (of, say 4 x 4 or 8 x 8) and a binary representation of the block is created by applying a threshold to the intensity values within the block [DASA95]. The threshold, the two quantization levels and/or the number of pixels above and below the threshold are determined to ensure the preservation of the first and second moments of the block. The statistical calculations of BTC are the following:

Let $m = n^2$ (n x n blocks) and let $X_1, X_2, \ldots X_m$ be the values of the pixels in a block of the original image. Then the first and second sample moments and the sample variance are, respectively

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i \tag{3.18}$$

$$\overline{X^2} = \frac{1}{m}\sum_{i=1}^{m} X_i^2 \tag{3.19}$$

$$\sigma^2 = \overline{X^2} - \overline{X}^2 \tag{3.20}$$

As with the design of any one bit quantizer, the objective is to find a threshold X_{th}, and two output levels, a and b, such that

$$\begin{aligned} &\text{if } X_i \geq X_{th} \text{ output } = b \\ &\text{if } X_i < X_{th} \text{ output } = a \\ &\text{for } i = 1, 2, \cdots, m \end{aligned} \tag{3.21}$$

If we choose $X_{th} = \overline{X}$, the output levels a and b are found by solving the following equations:

Let q = number of X_i 's greater than or equal to X_{th} $(= \overline{X})$

then to preserve $\overline{X}$ and $\overline{X}^2$

$$m\,\overline{X} = (m - q)\,a + q\,b \tag{3.22}$$

and

$$m\,\overline{X}^2 = (m - q)\,a^2 + q\,b^2 \tag{3.23}$$

Solving for a and b:

$$a = \overline{X} - \sigma\sqrt{\frac{q}{m-q}} \tag{3.24}$$

$$b = \overline{X} + \sigma\sqrt{\frac{m-q}{q}} \tag{3.25}$$

Each block is then described by the values of $\overline{X}, \sigma$ and n x n bit plane consisting of 1's and 0's indicating whether pixels are above or below X_{th}.

Wavelets

Wavelets are topics from pure mathematics which are a mathematical tool for hierarchically decomposing functions. They allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow [STOL95a, STOL95b]. The very term "wavelet" comes from the fact that they are "waving" above or below the x-axis. There are several different families of wavelets such as Daubechies, Haar, Coiflet, and Symmlet. Wavelets have shown a great potential in

many fields such as image compressions [ANTO92, DEVO92, FOUR94, HILT94]. A wide variety of wavelet-based image compression schemes exist ranging from adaptive transforms, tree encodings, edge-based encodings to the very simple entropy encodings.

In order to understand wavelets, I will initially present the most simple wavelets, the Haar basis, and see how it can be used for image compressions. This section is ended with the block diagrams of 2D forward wavelet transforms commonly used in wavelet image compressions [HILT94].

As in Fourier analysis, one of the most common approach to analyze a function $f(x)$ is to represent it as a weighted sum of basis functions,

$$f(x) = \sum_i c_i \psi_i(x) \tag{3.26}$$

where the collections of $\psi_i(x)$ are the basis functions and c_i are the coefficients[2]. To simplify things, let us constrain all of the basis functions to be the dilated (scaled) and translated versions of the same *mother wavelet*, $\psi(x)$. This is accomplished by

$$\psi(2^j x - k) \tag{3.27}$$

where $(j,k) \in R^+ \times R$

For Haar wavelets, the basis functions are given by

$$\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k) \quad , \quad k = 0, ...,2^j - 1 \tag{3.28}$$

[2] The main difference between Fournier analysis and wavelets is that Fournier basis functions are localized in frequency but not in time. Small changes in frequency domain will induce changes every where in the time domain. Wavelets are local in both frequency / scale (via dilations) and in time (via translations).

$$\text{where } \psi(x) = \begin{cases} 1 & \text{for } 0 \le x < 1/2 \\ -1 & \text{for } 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

It is easy to verify the orthogonality of $\psi_{jk}(x)$. The constant $2^{j/2}$ is to make this basis orthonormal (by $1 = \int \psi^2$). Now, one can represent the original function on [0,1) by

$$f(x) = c_{00}\phi(x) + \sum_{j=0}^{n-1}\sum_{k=0}^{2^j-1} d_{jk}\psi_{jk}(x) \tag{3.29}$$

$$\text{where } \phi(x) = \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

For example, let $\hat{y}$ = (9, 7, 3, 5) be a one-dimensional "image" of four pixels. We can compute the Haar coefficients by the following:

$$\begin{bmatrix} 9 \\ 7 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \sqrt{2} & 0 \\ 1 & 1 & -\sqrt{2} & 0 \\ 1 & -1 & 0 & \sqrt{2} \\ 1 & -1 & 0 & -\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \end{bmatrix}$$

The solution is

$$\begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

The goal of compression is to express an initial set of data using some smaller set of data, either with or without loss of information. For example, if we have a function as a weighted sum of basis function,

$$f(x) = \sum_{i=1}^{m} c_i \psi_i(x) \tag{3.30}$$

We can find an approximated function,

$$\tilde{f}(x) = \sum_{i=1}^{\tilde{m}} \tilde{c}_i \tilde{\psi}_i(x) \tag{3.31}$$

where $\tilde{m} < m$ such that for some norm

$$\left\| f(x) - \tilde{f}(x) \right\| \leq \varepsilon \tag{3.32}$$

In general, we could construct another function (different basis function) with fewer coefficients to provide a good approximation. If we want to fix the same basis function, then we can drop some of the coefficients to satisfy the user-specified error tolerance. Therefore, let σ be a permutation of 1,2,...,m and let $\tilde{f}(x)$ be a function that uses the coefficients corresponding to the first $\tilde{m}$ numbers of the permutation σ.

$$\tilde{f}(x) = \sum_{i=1}^{\tilde{m}} c_{\sigma(i)} \psi_{\sigma(i)}(x) \tag{3.33}$$

The square of the L_2 error with this approximation is

$$\left\| f(x) - \tilde{f}(x) \right\|_2^2 = \left\langle f(x) - \tilde{f}(x) \middle| f(x) - \tilde{f}(x) \right\rangle$$

$$\left\| f(x) - \tilde{f}(x) \right\|_2^2 = \left\langle \sum_{i=\tilde{m}+1}^{m} c_{\sigma(i)} \psi_{\sigma(i)}(x) \middle| \sum_{j=\tilde{m}+1}^{m} c_{\sigma(j)} \psi_{\sigma(j)}(x) \right\rangle$$

$$\left\| f(x) - \tilde{f}(x) \right\|_2^2 = \sum_{i=\tilde{m}+1}^{m} \sum_{j=\tilde{m}+1}^{m} c_{\sigma(i)}\, c_{\sigma(j)} \left\langle \psi_{\sigma(i)}(x) \middle| \psi_{\sigma(j)}(x) \right\rangle$$

$$\left\| f(x) - \tilde{f}(x) \right\|_2^2 = \sum_{i=\tilde{m}+1}^{m} \left(c_{\sigma(i)} \right)^2 \tag{3.34}$$

The last step follows from the assumption that the basis is orthonormal (i.e. $\left\langle \psi_{\sigma(i)} \middle| \psi_{\sigma(j)} \right\rangle = \delta_{ij}$ (Kronecker Delta). We conclude that to minimize the error for

any given $\tilde{m}$, the best choice for σ is the permutation that sorts the coefficients in order of decreasing magnitude; that is, σ satisfies $|c_{\sigma(1)}| \geq \cdots \geq |c_{\sigma(m)}|$.

The generalization of using Haar Wavelets to 2D image compression can be found in [STOL95a, STOL95b]. Obviously, this method is far from ideal. More serious method in image compression using Daubechie's *W6* wavelet is shown in Figure 3.17 and Figure 3.18 where H is a lowpass filter and G is a highpass filter. $\overline{H}$ and $\overline{G}$ are the impulse responses of H and G respectively. One can use *W6* as the mother wavelet basis function. The filter coefficients for H and G as well as the coefficients of the impulse responses $\overline{H}$ and $\overline{G}$ for *W6* can be calculated.

In Figure 3.17, an image is decomposed into an average image (f_{LL}) and three images (f_{LH}, f_{HL}, f_{HH}) which are directionally sensitive. To obtain higher compression, the average image will be further transform using the same scheme. Details of this wavelet image compression technique can be found in [HILT94].

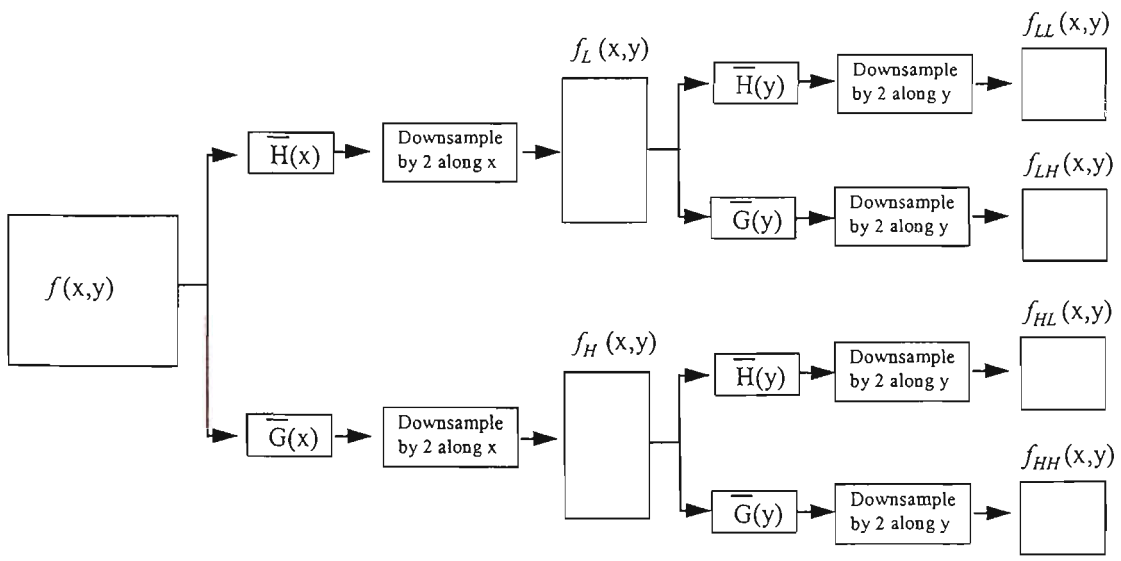


Figure 3.17 Block Diagram of the 2-D Forward Wavelet Transform

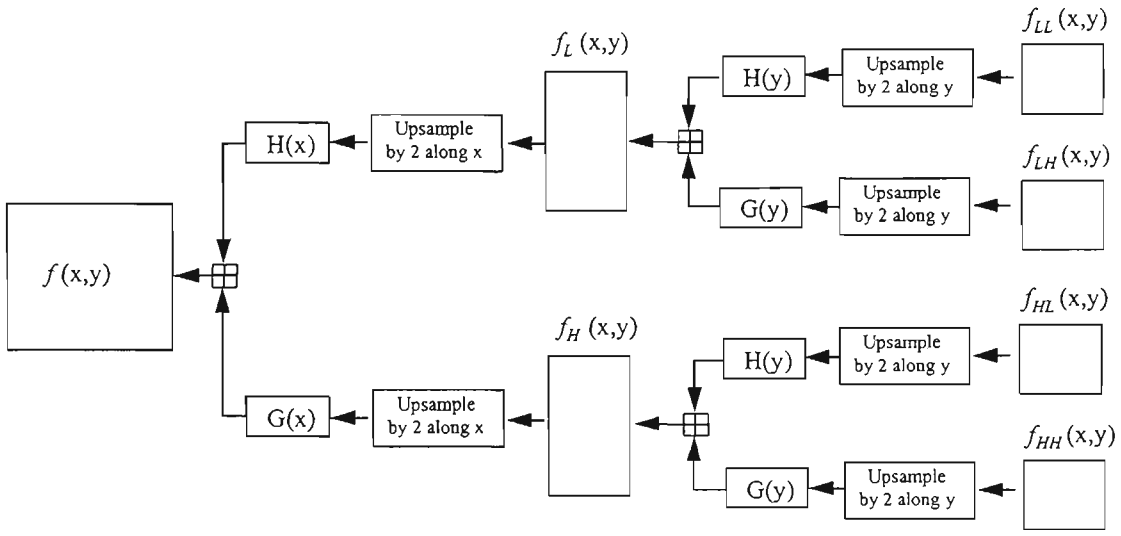


Figure 3.18 Block Diagram of the 2-D Inverse Wavelet Transform

3.3 Image Indexing and Retrieval in Compressed Domain

As described in Section 3.2, the primary objective of any compression technique is to reduce the data size for the consideration of transmission and storage requirement. Using compressed data for the purpose of image indexing and retrieval is a natural extension to image compression techniques as shown in Figure 3.19. Two approaches are possible into this direction as follows:

- The indexing and retrieval information can be produced during the compression process. This approach is only suitable for an integrated system in which the designer of the image database has a total control of all processes in the system.

- The indexing and retrieval information is sought from the already compressed data. This is particularly suitable for the existing image files compressed into certain formats and standards.

These two approaches are illustrated in Figure 3.19 where the extraction scheme, Y, is either driven by the compression scheme, X, or derived the indexing and retrieval information after the compression scheme.

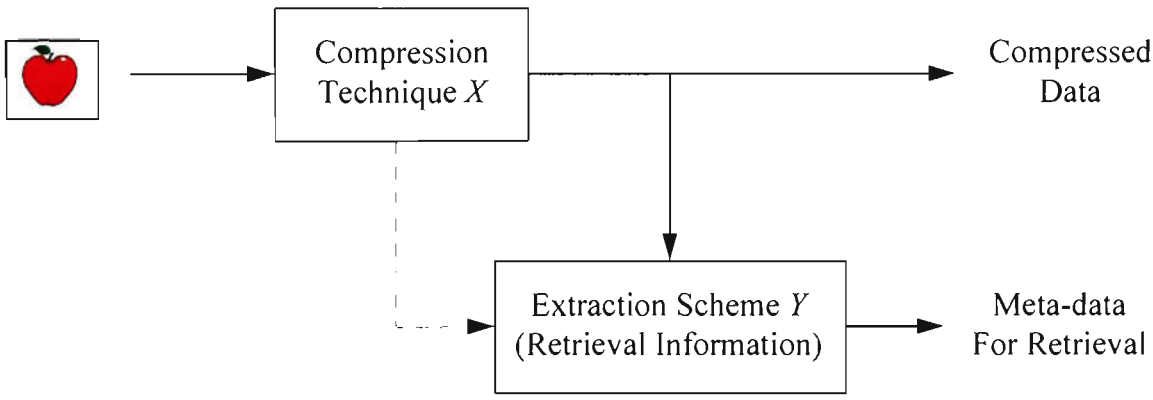


Figure 3.19 Extraction of Retrieval Information from Compressed Data

Before we analyze the advantages and disadvantages of using compressed data for image indexing and retrieval in Section 3.4, we will present three recent examples based on DCT, VQ and wavelets in here. These are just the representative examples in the proliferation of indexing and retrieval techniques using compressed data in the literature.

3.3.1 Using DCT(JPEG) for Image Indexing and Retrieval

The study of image indexing and retrieval using the compressed data of JPEG is certainly warranted because of its popularity. Many images are now stored in this format. Although there are four modes of operation in JPEG, we will describe an algorithm, which exploits the DCT-based compression scheme, by Shneier and Abdel-Mottaleb [SHNE96] for this purpose. Their prime objective is to create index keys so that images in similar contents to a given image can be retrieved. For examples, by giving an image of a person's face, they want to retrieve other images of people's faces from the image database. If an image of a document is given, the aim is to find other images which are also documents. Instead of describing their algorithm in plain texts and point forms, we will restructure their algorithm mathematically so that it is more concise and easier to understand.

Let us select a set of sub-images or windows ($W_1, \ldots, W_{2k}$) from the image as in Figure 3.20. These windows can be non-overlapping areas covering the entire image or can be a smaller sample of the image data. The window size will be the multiple of 8 in each dimension to coincide with the boundary of JPEG 8x8 blocks. They randomly pair up the windows in such a way that each window has only one partner. The pairs of windows should be far from each other to avoid both windows being in the same region. This particular pattern of pairing scheme will be used throughout the database and the query image. Let define each pair of windows as $P_1, \ldots, P_k$. Refer to Figure 3.11, the quantized 8x8 DCT coefficients in each block can be obtained by undo the entropy coding. The algorithm makes use of the already computed DCT coefficients to provide a set of keys for indexing and retrieval.

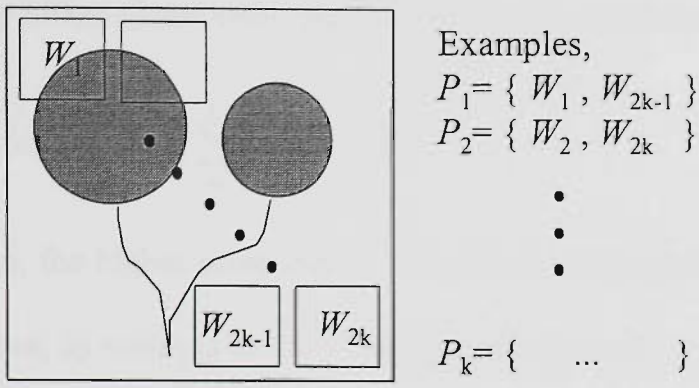


Figure 3.20 Pairs of Windows

For each window (W_i ; $1 \le i \le 2k$), we have a number of quantized 8x8 DCT blocks $\{ B_l ; l = 1,\cdots,n \}$. Each block has 64 coefficients $\{ C_j^l ; j = 0,\cdots,63 \}$. We compute the average for each corresponding DCT component by

$$A_j = \frac{1}{n}\sum_{l=1}^{n} C_j^l \tag{3.35}$$

where $0 \le j \le 63$

For examples, A_0 and A_{63} are the averages of all the DC components and the highest AC components in W_i respectively.

For each pair of windows (P_i ; $1 \le i \le k$), we compare the corresponding A_j as follows,

$$I_{ij} = \begin{cases} 1 & \text{if the 1st window } A_j > \text{ the 2nd window } A_j \text{ of } P_i \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

where $i = 1,\cdots,k$ and $j = 0,\cdots,63$

If we organize the I_{ij} into a set of indexing keys, we have 64 keys of k bits.

The similarity measure is by counting the total number of differences in I_{ij} between the query image Q and each target image T in the database. i.e.,

$$\text{Score } (Q,\ T) = \sum_{i=1}^{k} \sum_{j=0}^{63} \left| I_{ij}^{Q} - I_{ij}^{T} \right| \tag{3.37}$$

With this measure, the higher score means the weaker match between Q and T. The ranking is, therefore, by ordering the images in ascending score.

They performed a number of experiments; 16- and 32- windows (i.e. $k = 8$ and 16), different window sizes, and DC coefficients only. The conclusion is that the best arrangement is by using all 64 coefficients with the 16-window scheme.

3.3.2 Using VQ for Image Indexing and Retrieval

By using the histogram of pixels for image indexing is probably one of the oldest scalar methods in image retrieval. With the maturity of VQ compression techniques, using the histogram of VQ labels for image indexing is the natural extension from the scalar counterpart. After all, the frequency of each label used in VQ can reveal the characteristics of the image in similar fashion.

The study of VQ for image indexing and retrieval is reported by Idris and Panchanathan in [IDRI95, IDRI96, IDRI97]. They explore not only the usual histogram of labels as mentioned above. Their idea of "Usage Map" represents a very simple and elegant way for the purpose of generating indices. A usage map of an image is a signature of N bits. Each bit indicates the corresponding codevector has

been used. Therefore, let T be the image. The representation of a usage map with N codevectors is the following,

$$\text{UM} = \{\, u(T,i)\ ;\ 0 \le i \le N-1 \,\} \tag{3.38}$$

$$\text{where } u(T,i) = \begin{cases} 1 & \text{if } T \text{ has at least one codevector, } i. \\ 0 & \text{otherwise} \end{cases}$$

Figure 3.21 illustrates the usage map generation. The bit pattern of the usage map characterizes the major features of the image. Also, if the codebook is large, the signature will be reasonably unique.

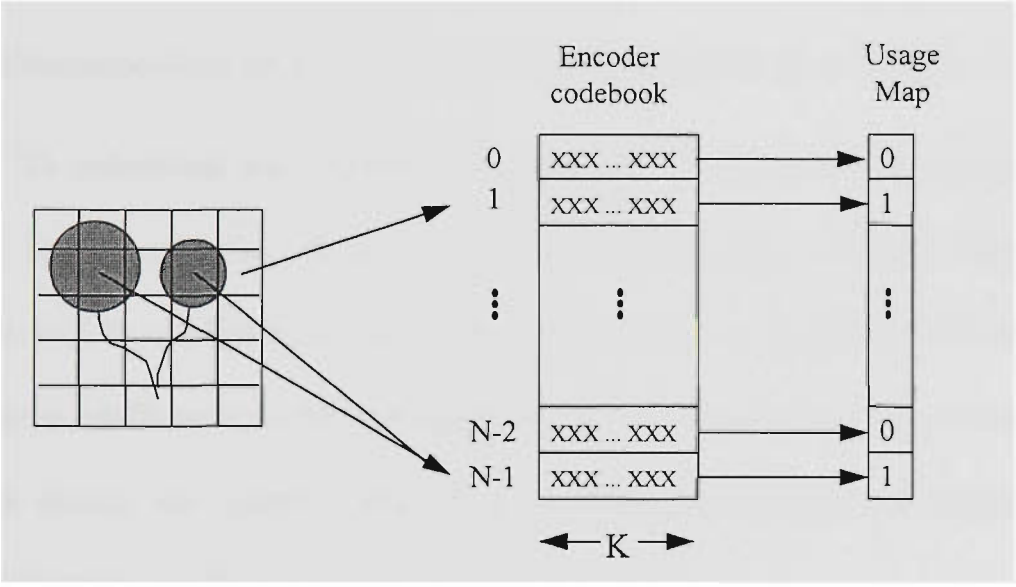


Figure 3.21 Usage Map Generation

The similarity between the target and query images, T and Q, is measured by the number of different bits using Eq. 3.39. Again, the smallest score means the closest match:

$$\text{Score}(Q,T) = \sum_{i=0}^{N-1} \left(u(T,i) \oplus u(Q,i) \right) \tag{3.39}$$

where $\oplus$ is the XOR operator

3.3.3 Using Wavelets for Image Indexing and Retrieval

The study of image indexing and retrieval using Haar wavelet decomposition is pioneered by Jacobs, Finkelstein and Salesin [JACO95]. The multiresolution nature of wavelets provides a good foundation for them to exploit this possibility. Their experiments are by far the most comprehensive among the algorithms described in this chapter. Hundreds of queries in databases of 1,000 and 20,000 images are used. Three types of queries, namely painted queries, scanned queries and memory queries (from users), are extensively studied. From the experiments, they found the standard 2D Haar decomposition on the color space of YIQ works best for their data.

To understand their algorithm, we must first introduce their "image querying metric". Instead of using all the wavelet coefficients (e.g., 128x128 image will generate 16,384 coefficients), the metric is designed to drastically truncate and quantize the coefficients so that the significant features in the images are retained and the fine details are ignored. According to them, this appears to improve the discriminatory power for image querying.

For each color channel, let Q and T are the wavelet decomposition of the query and target images. $Q[0,0]$ and $T[0,0]$ are the scaling function coefficients corresponding to the overall average intensity. $\tilde{Q}[i,j]$ and $\tilde{T}[i,j]$ represent the $[i,j]$-th truncated, quantized wavelet coefficients of Q and T; these values are either -1 (large negative coefficient), 0, or +1 (large positive coefficient). Their "L_q" image querying metric $\|Q,T\|_q$ is given by

$$w_0 \left| Q[0,0] - T[0,0] \right| + \sum_{i,j:\tilde{Q}[i,j]\neq 0} w_{bin(i,j)} \left(\tilde{Q}[i,j] \neq \tilde{T}[i,j] \right) \qquad (3.40)$$

where the expression $(\tilde{Q}[i,j] \neq \tilde{T}[i,j])$ is evaluated to 1 if $(\tilde{Q}[i,j] \neq \tilde{T}[i,j])$, and 0 otherwise. The weights, w_0 and $w_{bin(i,j)}$, are obtained experimentally. The function $bin(i,j)$ is to group terms into a small number of bins. They use only six weights $(w_0, w_1, ..., w_5)$ for their experiments. Also, not all truncated coefficients are used. The m largest-magnitude coefficients (again, m is determined experimentally) of Q and T are involved in the database preparation and query formation.

The algorithm performs a standard 2D Haar wavelet decomposition of all images in the database, and store the overall average color ($T[0,0]$) and the signs and indices of the m largest-magnitude coefficients of each color channel. For a given query image Q, the same process of preparation is performed. Equation 3.40 is then used to compute the score for each image in the database. Finally, the smallest scores are considered to be the closest matches. They let the users to browse through the top 20 target images.

3.4 Evaluation of the Approaches

From the studies of the three examples in previous section, we can analyze the common approach of using compressed data for image indexing and retrieval, and their advantages and disadvantages can be highlighted. In Section 3.4.1, the desirable characteristics in the approach are explained. Furthermore, problems related to the approach and image retrieval in general are identified in Section 3.4.2.

3.4.1 Common Approach and Desirable Characteristics

Uncompressed image data take up a large amount of secondary storage, and for many database applications, the use of compression techniques is unavoidable. Also, for distributed applications over low bandwidth networks, compression techniques can, to a certain extent, alleviate some of the bandwidth limitation. These are the obvious reasons. Another important reason for using compression techniques in image retrieval applications, particularly lossy compression, is the support of browsing operation. The search resulting from a query specification may not always produce a precise collection of images. Hence, the visual inspection out of the ranked images tends to be the most frequent operation in user-machine interaction.

Most of the compression techniques are computationally expensive. If we can make use of the compression process and produce the indexing and retrieval information, it is an added bonus. Images, which are already in compressed files, can also benefit from this approach. Instead of completely decoding the compressed image and then generating the indexing information, the retrieval system can work on the compressed data and will likely result in a faster operation. A good example of this scenario is searching images on the Internet.

As observed from the above algorithms, the common approach to image indexing and retrieval is to generate some manageable signatures or indexing keys. These keys are designed to capture the major features of the images without excessively concern with fine details. Using fine details for similarity measure is likely to be false matches. Also, in many cases, the query formulation is imprecise

anyway. Using major features to generate indexing keys is probably a better choice. In other words, we trade the recall ratio to the precision ratio (i.e. the desired image surfaced a little bit later is better than never at all).

In image retrieval, the properties of rotation, scale, translation and color-shift invariants are desirable. Unfortunately, if compressed data are used for indexing, it is most likely dictated by the compression schemes. The total freedom in the algorithm design of indices is not available. For example, whether one likes it or not, one needs to make use of the block-by-block nature of JPEG if such compressed data is chosen. Also, one has to be careful not to heavily rely on the localities of blocks. Otherwise, the rotation and translation invariants will be jeopardized.

One must justify on the computational costs of using compressed data for image indexing and retrieval. However, we must separate the indexing part and the retrieval part when we consider the computational costs. For well-defined image database applications, the indexing information is generated when the images are entered into the system. The time and computational costs required to generate such indices may not be too critical. This is because the scanning process, the cataloging routine and the house-keeping interaction are likely to be the bottleneck. On the other hand, the retrieval mechanism must be reasonably fast. Users will not tolerate a long delay on browsing through a set of target images. The pre-computed indexing keys and the similarity matching are related issues but must be separately dealt with. For ill-defined image repository and remotely accessed databases such as those on the Internet, both the indexing and retrieval processes must be very efficient. This is because (1) we do not have the luxury to pre-compute keys before hand and, therefore,

dynamic indexing is the only option, and (2) the querying/retrieval process requires the images to be transmitted over the network which can make the searching impracticable. From this viewpoint, using compressed data for indexing and retrieval offers a distinct advantage.

In all of the algorithms described above, the similarity measure tends to measure the difference between two images. In ranking the images, the lowest score means the best match. Using the cardinality of positive numbers for the implicit ranking is an efficient method particularly for unknown numbers of target images. Also, determining the similarity measure or metric is the second most time critical operation next to the browsing operation. If each metric is applied to all the images in a large database, the design of such metric must be very efficient. Therefore, the metrics such as L_1 and L_2 (Eq. 3.16) are very expensive, if not ineffective, on the querying side.

Many indexing techniques, particularly in spatial domain, use numeric values for the feature vectors. When two images are compared, distance metric or similarity measure are generally used to rank the target images. The drawbacks to this approach are:

1. The computational cost is very high when multi-dimensional vectors are involved. If the collection of images is very large, the performance on the querying side would be seriously affected. Generally, users will not sympathize with the sheer size of the database and will always demand a near instantaneous response to their queries. Although tree-based approach to organize feature vectors is certainly better than sequential search, the

computational cost of using multi-dimensional feature vectors, particularly on the querying side, still remains a major issue in image retrieval.

2. To compute and rank images using multi-dimensional feature vectors, the process of using weights for each dimension is often used. The weights are normally chosen either from the properties of the retrieval methods or from experiments. To a certain extent, it is somewhat superficial and counter-productive with respect to retrieval efficiency.

3. As mentioned before, using coarse features for image retrieval improve the recall rates. However, using higher dimension in the construction of the feature vectors will generally have an adverse effect on recall rates but may improve on the precision rates.

3.4.2 Problems to be Solved

We believe the avenue of using compressed data for image indexing and retrieval is promising. However, one must study the effect of the compression ratio to the precision of the retrieval. Although fine details of the compressed data are normally ignored in generating indices which coincide with the approach of many compression schemes themselves, the danger of over truncated and quantized processes is evident.

Currently, there are many compression schemes and file formats available. For well-defined image database applications, a particular compression scheme will be adopted and indices can be generated as a by-product. But, for other applications, it

will be far-fetched to hope for an algorithm to work on many different schemes. Hence, the only option is to convert from other compression schemes to its own scheme. This defeats the purpose of using compressed data for image indexing and retrieval.

One may already observe that we did not directly compare the recall rates, the precision rates, and the efficiency of all the algorithms. The reason is very simple; it is not fair to do so. We do not have a well-organized testbed and benchmark in image databases. In contrast to other related disciplines, the test images for the image processing field are Lena, Peppers and Boats etc. Miss America, Flower Garden and Mobile & Calendar sequences are the popular test video clips for the video processing field. Although testbeds and benchmarks may not always be completely objective and fair for each individual algorithm, we do at least have a common ground for comparison if the methods fall into the same category and aspects. To cater for a large community of image database researchers, we suggest that the collection of images for general-purpose image databases should at least include 1) line-art images, 2) images from natural scenes, 3) images of man-made objects, and 4) images related to people, faces and events etc. Also, for each category, we should include the rotated, translated, scaled, and color-shifted variations of the images. A very similar sequence of images, perhaps from a short video clip, should also be included for the testing of similarity retrieval. Captions for the images should be provided for those algorithms which make use of the textual information associated with the images. If the standard test images are available, we can also supply a suite of pre-defined queries for testing. The recall rate and the precision rate can then be measured for comparison. The

computational costs can also be measured using standard benchmark techniques or some form of complexity analysis.

3.5 Conclusion

In this chapter, the overall principle of image indexing and retrieval in compressed domain is provided. A comprehensive evaluation of the most contemporary compression techniques is undertaken in this chapter. They include Run-length Coding, Huffman Coding, Arithmetic Coding, Predictive Coding and Ziv-Lempel Dictionary Coding as the representative techniques in the area of the distortionless data compaction. Discrete Cosine Transform (DCT), Fractals, Vector Quantization (VQ), Block Truncation Coding (BTC) and Wavelets are used to demonstrate the irreversible techniques in image compression.

The general approach to image indexing and retrieval using compressed data is studied through examples reviewed in the recent literature. The first possibility involves using DCT coefficients in JPEG to construct indexing information. Since JPEG is a popular compression standard, the merit of using DCT coefficients for the extraction is certainly justified. The second possibility uses the references of the codebook in VQ as the indexing information. It is a simple but efficient way to characterize the major feature of an image. Using the coefficients of wavelets to generate indexing information is our third example. The properties of wavelets, particularly the multi-resolution one, are advantageous to image indexing and retrieval.

Using the studied examples, the common approach and desirable characteristics are identified. The strength and weakness of using compressed data for image indexing and retrieval are analyzed. The problem associated with this approach is also provided. We concluded that using compressed data is workable and provides significant scope for image indexing and retrieval. The rest of this thesis will be built from the concepts developed in this chapter.

Chapter 4

Image Hashing

4.1 Introduction

The idea of hashing was originated in the 50's. Despite the term, *hashing*, had already become the common jargon in the 60's, researchers were reluctantly to use the term in print at that time. This is partly due to the meaning of the term in English. According to the *Webster's* Dictionary, the meanings of "hash" are `1) to chop (meat or vegetables) into pieces for cooking, 2) to make a mess or botch of; bungle.` This undignified meaning drove researchers away from using it. Not until the mid-60's, it became the standard terminology for key transformation. This interesting history of hashing in computing has been documented by Donald Knuth in *the Art of Computing Programming* [KNUT73]. Today, the term is very popular and has been used in different areas of computing. Some of these areas include the *static hashing* for text retrieval or file organization [KNUT73], the

extendible hashing for relational databases [ELMA94, SILB97, FOLK92], the *geometric hashing* for machine vision [WOLF97], and the *secure hashing* for applications in authentication [PIEP93].

The very essence of hashing is to provide O(1) access to an arbitrary data item regardless of the data size. In other words, we would ideally like the underlying retrieval mechanism to locate the data item in constant time and the access does not depend on the size of the collection. In practice, we would satisfy with the retrieval in near constant time. For image data, the fundamental concept of hashing is very much applicable. The reasons are the following:

- *Data size:* Image data are voluminous. If we can hash image data into some forms of hash values, we can then organize images into categories or buckets and retrieve images via these hash values. Of course, hash values should be much smaller than the original to be effective.
- *Ordered access:* Hashing is inherently not suitable for ordered access. This major weakness for hashing does not really concern image data, since the order of images in an image collection is generally not important. Given an image, there is no apparent meaning to locate the "next" image if we are not dealing with sequence of images. In general, all the relational operators with the exception of $\{=, \neq, \approx\}$ are not meaningful to image data unless defined by the applications.
- *Similarity retrieval:* In many cases, images are retrieved based on their visual contents. If we can hash similar images into similar hash values, the

hash information can be used for similarity grouping and retrieval. Although our primarily goal for image hashing is for efficient access and not for similarity retrieval, it should be beneficial if the hashing function exhibits this property.

Recently, the motivation in efficient access to multimedia data drives the proliferation of research findings in visual information retrieval. Surprisingly, the concept of *Image Hashing* based on the visual contents of image data is not well researched. We can only see the application of hashing in image databases through the textual or object information about the images [RABI90, BHAT94]. This does not satisfy our perception of image hashing. We would like to take the paradigm of hashing and directly apply to image data. In other words, rather than using the meta-data for images to generate the hash information, the transformation from image data to hash information should be directly and completely based on the contents of images. This view is parallel to the key transformation in textual information.

In this chapter, we will develop and promote the concept of image hashing. To understand our motivation behind image hashing, we provide a brief review and analysis of the traditional hashing techniques and their relationship to image hashing in Section 4.2. In Section 4.3, we present our complete view of image hashing. These include the motivation, the possible approaches and our definition of image hashing in Section 4.3.1, Section 4.3.2 and Section 4.3.3 respectively. We conclude that two-dimensional bitplanes are the better form of hash information for image data.

4.2 Evaluation of Traditional Hashing Techniques

4.2.1 Preliminaries

Hashing is one of the oldest searching techniques which allows one to determine the presence or absence of an arbitrary element in a table of entries. In the absence of a priori statistical information, hashing is both conceptually simple and also very efficient. It has a better average behavior than other linear or binary searching techniques which, at the best, has a search time of $O(\log_2 n)$. The search times for hashing techniques can be independent of the number of entries in a table. It is achieved through a transformation of an element X using some arithmetic function, *f*. $f(X)$ gives the address where X should be placed. If X is a key in the key space, this key-to-address transformation provides a mapping mechanism from the key space (K) into an address space (A). Very often, the key space is usually several orders of magnitude larger than the address space, many keys will be mapped on to the same address. Such a many-to-one mapping results in collisions. For image hashing outlined in subsequent sections, we can see that, because of the extremely large image spaces for image data, this problem is more severe.

There are many hashing methods available [KNUT73, TREM79]. The most classic one is the division method. Others include the mid-square method, the folding method, the digit-analysis method and the length-dependent method etc. For the division method, an integer key, X , out of n possible keys is mapped using a hash function $f(X) = X \bmod m$ such that $\alpha * m > n$, where α is the loading factor. According

to Knuth, m should be a prime number to avoid substantial bias in the hashing. In general, the desired properties of any hashing method are: 1) it is easy to compute, 2) it minimizes the number of collisions, and 3) it does not result in a biased use of the hash values. These desirable properties are equally applicable to image hashing.

Ideally, a hash function should avoid collisions by mapping a set S of n keys to distinct locations. If this property of the hash function holds, it is a *perfect hash function*. Best of all, if a perfect hash function achieves a loading factor of 1, it is called a *minimal perfect hash function* (MPHF). MPHF is not only optimal in performance and guaranteed one probe access, but also optimal in space utilization of a given hash table.

In reality, MPHF is difficult to find. Collisions are often occurred in hashing schemes. Hence, the inevitability of collisions has led to the selection of methods for collision resolution. The objective of a collision-resolution technique is to place the colliding element elsewhere. In general, there are two broad classes of collision-resolution techniques; open hashing and chaining.

With open hashing, if a key X is mapped to a location i and this location is filled, then other locations in the hash table are examined until a free slot is found for this key. One simple solution to obtain the free slot is by the following sequence of locations for a table of m entries: $i, i+1, \ldots, m-1, 0, 1, \ldots, i-1$. This collision-resolution technique is called linear probing. Alternatively, a random sequence of positions rather than an ordered sequence is generated. Such a technique is called random probing. This random sequence should consist of every location exactly once. One example to generate a random sequence of position numbers is the following,

$$y \leftarrow (y + c) \bmod m \tag{4.1}$$

where y is the initial position number. The values of c and m are chosen such that they are relatively prime to each other. For example, if we assume m = 7 and c = 3, the random sequence for an initial position number of 2 will be 5, 1, 4, 0, 3, 6 and back to 2.

One advantage of open hashing is that no additional space is required. However, it suffers a number of drawbacks. Firstly, deletion is troublesome. Once an element is deleted from the table, a special marker is often used to distinguish it from an empty slot. Otherwise, compaction is needed. For the random probing, the problem for deletion is much more severe. If many deletions are required, the random probing should not be employed. Secondly, the clustering effects are usually occurred particularly for linear probing. Long sequence of occupied slots tends to become longer. Hence, the number of probes is increased and the performance is deteriorated. Thirdly, when the table is nearly full, the probing is increased exponentially. Moreover, if the entire table is full, reorganization to a large table is the only option. With these drawbacks, open hashing is of little use for relational databases as we will describe in the next section. It is mostly used in the construction of symbol tables for compilers where insertion and lookup operations on the symbol tables are the only operations during compiling. Also, since single value hashing is not useful for image data, the approach of open hashing is of limited use for image hashing.

With chaining, collision is avoided by linked-allocation techniques. Overflow elements are handled by a separate overflow area which provides additional storages for chaining. In other words, the colliding elements are chained into a special

overflow area which is different from the original prime area of the hashing table. A linked list is used for each set of overflow elements. For some applications, it may be advantageous to have all colliding elements in the list ordered. There are many other forms of chaining. For example, a more efficient representation involves the use an intermediate hash table which the prime area contains only pointers. Each pointer is linked to a list of elements in blocks to minimize the storage spaces to set up the linked lists. Since the basic idea of chaining is to cluster data elements using linked lists, this approach is universally applicable to organize different types of data elements such as images through image hashing techniques.

Generally, sequential search is generally required after the hash key is computed. Therefore, the search time for a particular element is mostly independent of the number of entries in the hash table. It depends on the times taken to evaluate the hashing function and to perform the data operation. However, it is desirable to keep a uniform distribution of elements among the linked lists and the number of elements in each linked list should be small. Otherwise, if all elements are mapped into the same list, it is no more efficient than a linear search and the load is heavily skewed. In practice, it is not a trivial matter to obtain an optimal balance of load in each linked list, since the size of the colliding elements in each linked list depends on the keys being used.

The performance analyses for open hashing (linear probing) and chaining are thoroughly discussed in [KNUT73]. We will state without proof the results as follows:

The average number of probes for chaining are:

$$U_n \approx e^{-\alpha} + \alpha \qquad \text{(unsuccessful search)} \qquad (4.2)$$

$$S_n \approx 1 + \tfrac{1}{2}\alpha \qquad \text{(successful search)} \qquad (4.3)$$

The average number of probes for linear open hashing are:

$$U_n \approx \tfrac{1}{2}\left(1 + \left(\tfrac{1}{1-\alpha}\right)^2\right) \qquad \text{(unsuccessful search)} \qquad (4.4)$$

$$S_n \approx \tfrac{1}{2}\left(1 + \tfrac{1}{1-\alpha}\right) \qquad \text{(successful search)} \qquad (4.5)$$

where $\alpha = n / m$ is defined as the loading factor of the hash table.

4.2.2 Hashing for Relational Databases

One of the important issues in relational databases is the performance in accessing records randomly through key or non-key fields. Particularly for large relational databases, fast random access to the desired records in a given table is essential. Hence, supporting data structures are often used to facilitate the searching. Generally, there are two main approaches to build access structures to speed up the retrieval of records; namely *indexing* and *hashing*. These two approaches are sometimes inter-mixed to build better access structures. For our concept of image hashing, we can also combine with tree-based indexing techniques to provide efficient access structures for image data.

Given a table of records on disk, an index structure is an alternative search path to gain fast access to records based on a particular search field. This search field can be the primary key or the secondary keys of the table. There are many types of indexing structures available ranging from single-level indices to multi-level indices. Single-level indices are characterized by one-level ordered indices. If the records of a

table are ordered sequentially by its search field, a sparse index can be used to provide a search time better than O($\log_2 n$). Otherwise, a dense index for every search value (key or non-key) is needed. Single-level indices are of limited use in modern relational databases for the following reasons:

- A binary search is always required and is still very inefficient for large tables. In other words, too many seeks are needed in binary searching. The problem is further complicated if the searching cannot be done on the main memory and the index must be kept on disk.
- Even if we use sparse index, the index itself can become very large and, hence, impractical for efficient searching.
- Insertion and deletion can be costly unless flexible data structures are in place to shrink and expand the index easily.

Therefore, multi-level indices are often adopted to improve the efficiency of searching. Multi-level indices are characterized by tree-like structures. Multi-level indices can be implemented in the form of binary trees, AVL trees, B-trees or B^+-trees. Since the fan-out of binary and AVL trees are less than B- and B^+-trees, the heights of binary trees are often higher and, hence, more disk probes are required. Even with the balanced nature of AVL trees, B- or B^+-trees are generally preferred over binary trees in relational databases. B- and B^+-trees are height-balanced trees with the following characteristics:

- Each nonleaf node has between $\lceil p/2 \rceil$ and p children where p is the order of the tree. Since the worst case of the node is half empty, it is

inevitable that spaces are wasted. The trade-off between performance and space overhead is acceptable in B- and B^+-trees.

- The tree is balanced through node splitting, concatenation, and redistribution from the bottom up. Through the use of two-to-three splitting and of redistribution, a greater storage efficiency is achieved.
- For higher values of p, the shape of the tree is likely to be broad and shadow. Seeks to secondary stores can be reduced. Furthermore, it is also suitable to be implemented by virtual tree structures through paging.
- The restructuring after each insertion or deletion, if needed, maintains the performance at an acceptable level.

The major difference between B- and B^+-trees is that every value of the search fields together with a data pointer to the record appears once at some nodes in the B-trees. Consequently, we may find the record we are looking for at any level of the B-tree. This differs from B^+-trees which requires the traversal of the tree down to the leaf nodes for the associated information or data pointers. The internal nodes repeat some of the search values in the table as the guide for the traversal. Given a fixed node size, an internal node of a B^+-tree can hold more entries than a B-tree. In other words, the fanout of B^+-trees can be higher and shadower. Also, since all the data pointers in B^+-trees are ordered in the leaf nodes, sibling pointers can be used to link the leaf nodes so that ordered access to the table can be efficiently performed. In practice, B^+-trees are preferred over B-trees in relational databases.

Retrieval of records based on *hashing* is another important technique for relational databases. Contrast to multilevel indices, the desired record can be directly retrieved by computing the address of the record using a hash function on the search field. As described in Section 4.2.1, it is important that a good hash function is chosen for the data set. Ideally, the distribution should be both uniform and random. This is difficult to achieve due to the fact that we do not know at design time precisely the statistical information of the data set. The problem can be somewhat alleviated by reducing the loading factor of the hash table. But storage spaces are wasted as the results. Alternatively, open hashing or chaining can be used at the expense of performance. Or better yet, self-adjusting hash structures can be used to overcome the drawback of static hashing and to allow growth and shrinkage of the databases dynamically. Hashing techniques such as *dynamic hashing*, and *extendible hashing* are some of these algorithms.

The general approach to flexible hashing techniques is outlined in Figure 4.1. Instead of using a fixed hash table to buckets, a use-more-as-you-need-more approach is the underlying concept of flexible hashing. To support this concept, the fixed hash table must be replaced by a more flexible data structure or directory in order to accommodate the dynamic nature of most relational tables where insertions and deletions of records are constantly performed. The leading bits of the hash value are used to guide the distribution of records among the buckets. If a bucket is overflow, it is splitted into two buckets and the records in the bucket are re-distributed by using one more bit of the hash value. The corresponding directory or data structure is updated if necessary. Similarly, if a bucket is empty or below a pre-defined level, it is

coalesced with the neighboring bucket to form a single bucket. Since only localized buckets are involved in the reorganization, the overall performance is predictable. Also, the binary representation of the hash value plays an important role to build the directory or data structure. It is desirable that the address space for the hash value is made to as large as possible. For example, if 32 bits are used for the hash value, we can potentially have 4.3 billions of distinct buckets. Obviously, the chosen hash function to generate the string of bits ought to be random and uniform. In some cases, it is desirable to reverse the hash value and use the trailing bits first.

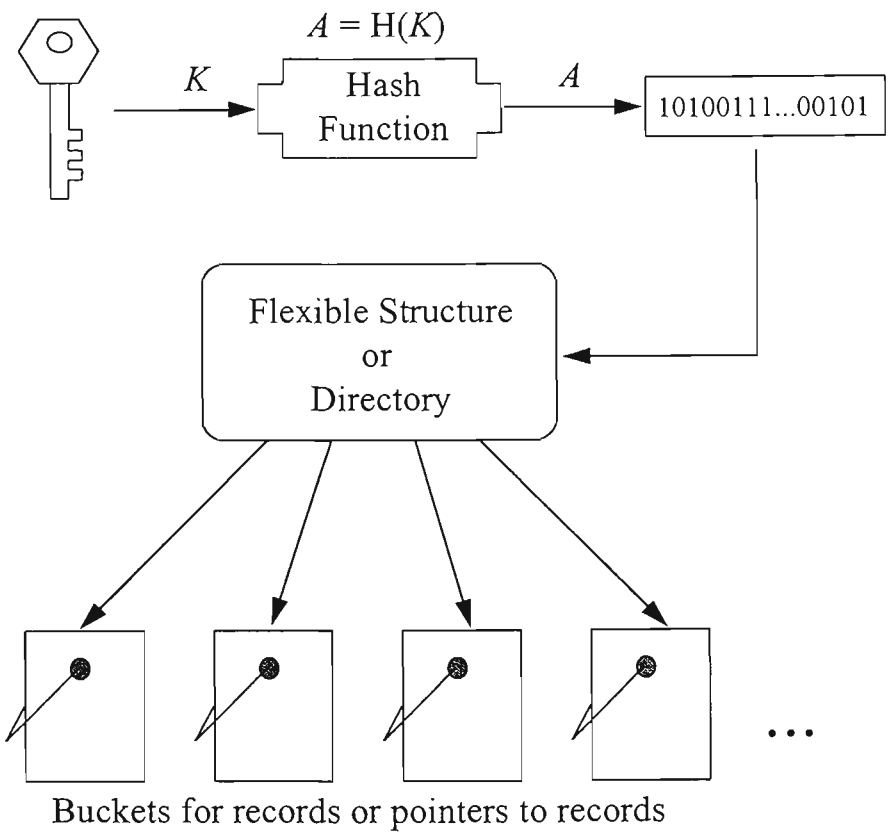


Figure 4.1 General Approach to Flexible Hashing Techniques

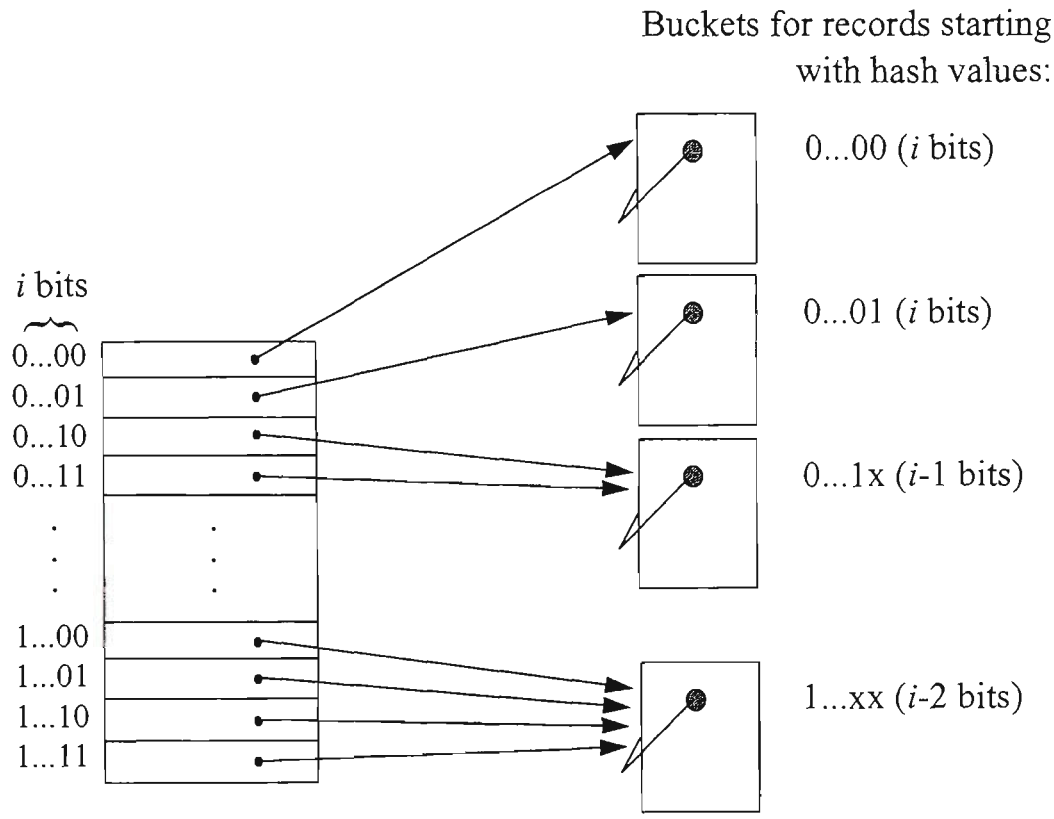


Figure 4.2 Extendible Hash Structure

In *extendible hashing*, a directory of 2^i pointers is maintained at any one time, where $0 \leq i \leq n$ and n is the word length of a hash value. Not all the pointers are pointing to distinct buckets. Several consecutive directory locations having the common hash prefix may point to the same bucket. Figure 4.2 illustrates an extendible hash structure. For each bucket, it is necessary to keep a local value j , where $1 \leq j \leq i$. For example, the last bucket in Figure 4.2 has $j = i - 2$. When a record is inserted into the designated bucket according to the leading bits of the computed hash value, there are three possible scenarios:

1. The bucket is not yet full and the record or a pointer to the records is inserted. We do not need to modify the directory.

2. The bucket is full and $j < i$. The bucket is splitted into two buckets. Since the bucket contains records with the same hash prefixes up to the initial j bits, we can separate the records into two buckets using j+1 bits. The directory is updated to reflect the new buckets. We increment the value of j by 1.

3. The bucket is full and $j = i$. Doubling the size of the directory is needed. Effectively, we increment the value of i by 1. Each entry in the directory is then replaced by two entries and pointing to the original bucket. Step 2 is then performed to insert the record.

To delete a record, we take the first i bits of the hash value and locate the bucket by just one probe. We can then remove the record or the pointer from the bucket. If the bucket becomes empty, buckets with the same hash prefixes can be coalesced. It is possible that the size of the directory is reduced by half as the result of the deletion. In general, grouping records through partial hash keys are useful concepts and certainly applicable to our model of image hashing.

The approach for *dynamic hashing* is very similar to *extendible hashing*. But a linked data structure rather than a directory is used to keep track of the buckets. The linked structure is implemented as a radix 2 trie. Buckets are splitted or coalesced if necessary. Space utilization and performance are very similar for both techniques.

4.3 Image Hashing

4.3.1 Motivation

Images are bulky in nature. We usually retrieve images by their file names. Within an operating system, the unique labeling of the image files by given a unique file name is working to a point that the content of the image can be irrelevant to the file name. Although it is very efficient to locate images by file names, this does not really allow us to group or cluster images related to their contents. To look at this problem from a different perspective, if an image is said to be within a large collection of images and the file name is not known, how can we efficiently locate the image without resorted to pixel-by-pixel comparison for all images. Even if we need to perform any elaborate search, we would like to drastically eliminate a large portion of images prior to any comparison. Similarity measures or distance predicates are reasonable solutions to solve this problem. However, our main goal is to provide an efficient mechanism to locate the image. This is similar to the hashing concept in text retrieval for which a record can be quickly retrieved through the key-to-address transformation.

The main motivation for image hashing is to parallel the success of traditional hashing and to genuinely provide O(1) access to images based on their visual contents. In this chapter, we develop the concept of image hashing and exploit the possible avenues in the transformations from image data to hash information.

4.3.2 Approaches to Image Hashing

Given an image, we would like to generate the appropriate hash information as shown in Figure 4.3 so that we can place the image into its bucket or cluster collectively. The hash information may not have to derive from the visual attributes such as colors or shapes. As long as the transformation to obtain the hash information can be tracked or reproduced, it can be used for image hashing. After all, we are merely interested in the efficient access of the images in a collection. Nevertheless, it is advantageous for the hash information to exhibit the property of similarity retrieval in addition to our goal for efficient access.

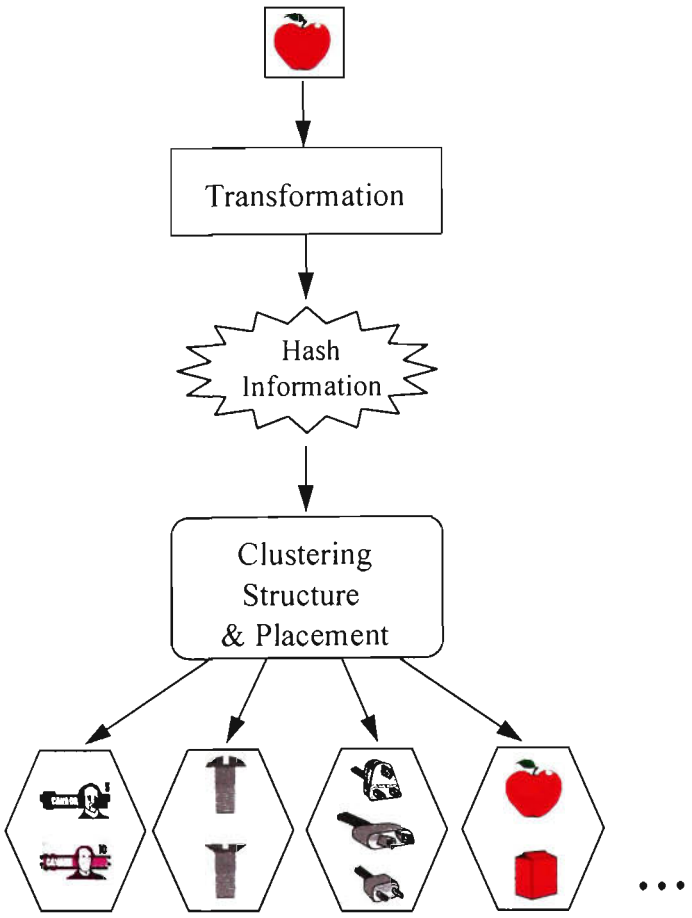


Figure 4.3 Concept of Image Hashing

Without lost of generality, we examine the categories of different hash information from the possible data characteristics. We aim to hold a general view for image hashing and do not specifically associate a particular algorithm for the transformation or mapping in each class of hash information. We also suggest some possible avenues to obtain the required hash information.

Hash to a Single Numeric Value

This category of mapping is to hash an image into a single real or integral value. Some of the possible ways to generate such hash information are the following:

- We can compute the average value of the pixels.
- We can take the pixel value at the center.
- We can take the first byte or the upper left corner of the image.

If an integral value is used, the bucket can be directly correspond to each cluster of images using the division method examined in Section 4.2. The mapping ratio will be very high. Although the handling of the hash value is very easy and the comparison is extremely fast, content-based similarity retrieval is not really possible. Hence, the clustering of images will be widely spread. Also, this approach is very sensitive to errors.

Hash to a Bit Vector

This category of mapping is to hash an image into a bit vector. Similar to hashing in text retrieval, the bit vector is expected to be short. Some of the possible ways to generate such hash information are the following:

- A block partition of an image is used to set a corresponding bit in the bit vector.
- A histogram-based algorithm is computed and some kind of threshold are used to set the corresponding bit.
- The 1 bits are selected by a pre-defined algorithm such as significant color regions, chain codes of a significant object and many others. A random seeking method is also possible.

The mapping ratio, however, is still high. The handling of the bit vector is reasonably easy if the bit pattern is not too long. The comparison should be relatively fast as it may be performed using low-level bitwise operators. Content-based similarity retrieval is possible but weak in nature. The main advantage of a bit vector approach is that we can make use of the conventional hash structures such as extensible hashing to organize the image clusters. Also, superimposed coding can be used.

Hash to a Numeric Vector

This category of mapping is to hash an image into a numeric vector. This is a common approach for similarity retrieval. Image processing is performed to extract the salient features of an image. The numeric value of each dimension represents each

image feature. The multi-dimensional feature vector is the basis for similarity comparison using some forms of metric. Some of the possible ways to generate such hash information are the following:

- Features are extracted from color histograms.
- Features are computed from spectrum analyses such as DFT or DCT.
- Texture analysis is performed to extract the features.

The mapping ratio will depend on the dimension of the feature vectors. If the dimension of the feature vector is high (say 20), the handling of the numeric vector is very clumsy. This is particularly true for large image collections. The comparison can also be very inefficient as the metric computation is likely to involve floating point arithmetic. Images can be clustered by space partitioning or closest neighboring vectors.

Hash to a Bit Matrix

This category of mapping is to hash an image into a bit matrix. Most spatial and geographical data are dealing with this kind of information [SAME90a, SAME90b]. Some of the possible ways to generate such hash information are the following:

- The bit matrix can be obtained from a binary projection of an image.
- Locations of some salient features are registered using a bitmap.
- The signs of significant coefficients of a transformation are extracted.

The handling of the bit matrix is good if the bit matrix is sparse and can be highly compressed. The cost for comparison should not be excessive as it may be performed using low-level bitwise operators. Content-based similarity retrieval is good. Superimposed coding is possible. Conventional clustering data structures can be applied.

Hash to a Numeric Matrix

This category of mapping is to hash an image into a numeric matrix of real or integral values. Some of the possible ways to generate such hash information are given in the following:

- The average values such as the DC coefficients of DCT are partitioned into macro-blocks for the entries of the matrix.
- We can select the significant coefficients of a transformation or the threshold in the spatial domain.
- We can use the projected image.

The handling of the numeric matrix is extremely poor if the number of rows and columns for the numeric matrix are high. Also, the cost for comparison is high unless partial matching is achievable under the specific algorithm. Content-based similarity retrieval may be good but sensitive to variations in small changes in the image contents. Hence, images are difficult to cluster into groups. This type of mapping is counter-productive to our goal.

4.3.3 Feature Vector vs. Bit Matrix

Among the five types of hash information outlined above, we believe that only the feature vectors or the bit matrices are the likely candidates for image hashing. The considerations are based on the following facts:

1. *Computational cost:* The most important property of hashing for image data is to provide a good guess as to where the image can be located within a clustering structure. If the guess is very expensive to compute, it does not meet our goal of efficient access.

2. *Memory and storage consideration:* Unlike the conventional hashing, the hash information for image data is mostly computed beforehand due to the fact that the cost to compute the hash information is very often too high for real-time generation. It makes sense to store the pre-computed hash information. Therefore, the hash information should be reasonably compact to reduce the storage requirement and to ease the demand on memory. It also helps the efficient handling and fetching of hash information between storage media.

3. *Ability to provide content-based similarity retrieval:* Although similarity retrieval is not the direct goal of image hashing, it helps to provide a mean of clustering images. If similar images can hash to similar hash information, we can make use of this fact and organize the images into a clustering structure by visual contents.

Between feature vectors and bit matrices, we are in favor of bit matrices for the purpose of image hashing. Feature vectors are expensive to compare. Also, the nature of bit matrices is more in line with the traditional hashing concepts and behaviors for textual information. For example, we can still make use of the superimposed concept on bit matrices. This is an important aspect on which our composite bitplane signature scheme in Chapter 5 is based upon. On the other hand, feature vectors do not exhibit this property. Furthermore, bit matrices allow us to develop suitable organizational structures similar to the development of *extendible hashing* in relational databases.

4.3.4 Our Focus in Image Hashing

From previous argument outlined in Section 4.3.3, we take the narrower view for image hashing and adopt the hash information as being the bit matrix. In our approach, we will also use the term, *bitplane*, instead of *bit matrix*.

Formally, the *image-to-bitplane transformation* is defined as a mapping or a hashing function, h , which maps the image space (I) into a bitplane space (B); $h : I \mapsto B$. That is, given an image value (i.e. a point in I), a mapping function h produces a unique bitplane of the image. We expect that the mapping function is surjective to reduce the comparison requirement and accept the fact that, as in conventional hashing, collisions will inevitably occur. Also, to facilitate content-based retrieval, similar images should ideally hash into similar bitplanes.

In practice, the hash information may consist of more than one bitplane as shown in Figure 4.4. For example, we need at least three bitplanes to represent color images. Also, the hash information for each color channel may collectively require multiple bitplanes. However, it is desirable to keep the number of bitplanes as minimum as possible. Otherwise, the effectiveness of using bitplanes for image hashing will be eroded.

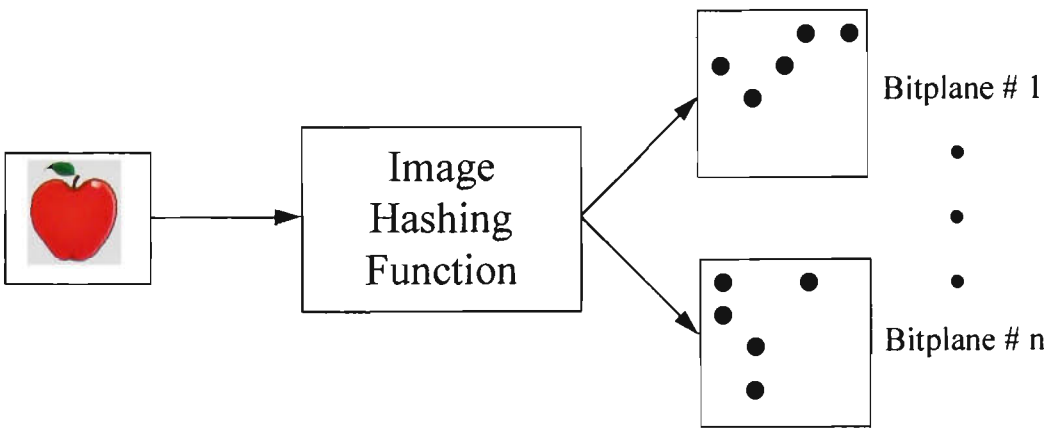


Figure 4.4 Using Bitplanes for Image Hashing

For a given image collection, there are a number of design parameters for image hashing. These include 1) the number of bits required to set the bitplane, 2) the dimension of the bitplane, and 3) the number of bitplanes to represent an image. Generally, these parameters are dictated by the underlying algorithm. The algorithm will determine the distribution of the bit patterns and, in turn, will affect the behavior of the image hashing scheme. Furthermore, these parameters are unlikely to be altered during the life span of a given system. Otherwise, all the images will need to undergo "re-hashing" and the exercise can be expensive.

4.4 Summary

In this chapter, the concept of image hashing is presented. Although hashing techniques are very popular for information retrieval in modern computing, the application of hashing in multimedia particularly to image databases is still not well researched. From the conceptual point of view, the basic principle of hashing is very suitable for image retrieval. We are motivated by this observation and develop the abstract framework of image hashing. This framework will be used in subsequent chapters in this thesis.

The main objective of hashing is to provide an efficient means to access an arbitrary data item regardless of the data size. It allows one to determine the presence or absence of the data item in constant time. Ideally, the distribution of the data items in the organizational structure should be as uniform and random as possible, although this property is hard to achieve in practice. Furthermore, we need to take into account other considerations such as the possibility of similarity retrieval.

The reported techniques on using hashing for image or multimedia databases are only applied to the textual or object information deduced from the images. This falls short of our expectation. We expect the hash information is directly derived from the visual contents of the images. In this way, we can attempt to establish a parallel between the traditional hashing paradigm of key transformation and achieve our perception of image hashing from images to hash information.

We also examine the categories of different hash information from the possible data characteristics. We conclude that bit matrices or bitplanes are the most

suitable form of hash information. The nature of bitplanes is more in line with bit vectors for text retrieval. The considerations are based on computational costs, memory and storage consideration and similarity retrieval. Superimposed coding in two-dimensional basis is also possible for the bitplane arrangement. Therefore, our approach to image hashing is focused on *image-to-bitplane transformation*.

We emphasize that there is no specific algorithms for image hashing proposed or defined in this chapter. A more concrete design in the form of Composite Bitplane Signature using wavelets is studied in Chapter 5. The study presented in this chapter is to gain a broader insight into image hashing in relation to the traditional hashing techniques and build up the concepts with similar properties.

Chapter 5

Composite Bitplane Signature

5.1 Introduction

Signature-based text retrieval methods have been the popular accessing techniques for text databases or documents [FALO84, FALO85, FALO87, FALO92, SALT89, WITT94]. Signature accessing strategy is based on representing every indexed term of a document in a binary string or bit vector. A signature file is created from either every indexed term or a block of terms using superimposed coding. The idea behind superimposed coding for textual data is particular attractive from the image indexing and retrieval point of view. We apply the signature approach to picture data and formulate the concept of Bitplane Signature. Bitplane Signature is a two-dimensional bit matrix in comparison to the one-dimensional bit vector of the text counterpart. Naturally, the superimposed coding concept can then be extended to the

bitplane signatures and form a two-dimensional superimposed bit matrix which we call it Composite Bitplane Signature.

In this Chapter, we will demonstrate the method of Composite Bitplane Signature for image indexing and retrieval [SO97b, SO99]. We briefly present the signature concept for text retrieval in Section 5.2. A simple example is used to illustrate the various aspects of signature retrieval. The underlying mechanism is very useful for the formation of Bitplane Signature. Bitplane Signature is developed in Section 5.3; this includes the outline of desirable properties for Bitplane Signature in Section 5.3.1. The scheme of using wavelet coefficients for signature generation is studied in Section 5.3.2. We also highlight other possible schemes for signature generation in Section 5.3.3. In Section 5.4, the complete framework of Composite Bitplane Signature is presented. The formation of Composite Bitplane Signature is illustrated in Section 5.4.1. In Section 5.4.2, our unique searching and ranking mechanism is presented. As a concrete illustration, we provide a simple example to demonstrate the formation of Composite Bitplane Signature and how to pose a query to it. The framework of Hierarchical Composite Bitplane Signature is then formulated. It allows us to rapidly produce the partial ranked list of images without the need to perform similarity retrieval on every image in the database. We finally conclude this Chapter with a summary in Section 5.5.

5.2 Signatures for Text Retrieval

Before we look at Bitplane Signature for image retrieval, let us investigate the signature retrieval for the text counterpart. This provides us an interesting contrast to our Bitplane Signature for image retrieval. This also leads to the development of Composite Bitplane Signature for fast similarity retrieval in two-dimensional bit matrix. Furthermore, the idea of image hashing from Chapter 4 is very much applicable to the generation of Bitplane Signature similar to the hashing techniques used to generate signatures for words in text databases.

The common approach to text retrieval is to divide a document into a number of logical blocks. Each block consists of a pre-defined number of distinct words. Hashing techniques are used to generate the signature for each word. The word signatures in a logical block are superimposed using the bitwise-Or operator to form the block signatures. The block signatures are concatenated to form the document signature. The number of 1 bits in each word signature and the word length itself are design parameters. However, it is desirable to control the number of 1 bits in the block signatures so that the false drop probability is maintained to the lowest [FALO92, SALT89]. It has been proved that the superimposed coding with 50% of 1 bits is an optimal design.

To illustrate the signature concept for text retrieval, let us look at a simple example. Table 5.1 contains four logical blocks for a hypothetical document. To simplify the illustration, we assume each word signature is a 12-bit vector which is too small for real applications. It is assumed that no more than 3 bits are set for each

word signature. There are two ways to set the 1 bits; 1) three hashing functions with the given word as the key are used to generate values in 0 to 11 and setting the corresponding bits, and 2) a random generator is used to pick the three positions. Overlapping is allowed for both of the schemes. The advantage to use hashing functions is that there is no need to store a table of word signatures. This is because the bit patterns are predetermined by the hashing functions and can be computed on the fly. The disadvantage is that the distribution of bits may not be random and uniform as we may hope for. This leads to unnecessary collisions. The advantage and disadvantage for using a random generator are just the opposite. Instead of providing details for three hashing functions, we chose the second scheme just to simplify the illustration.

Block	Text
1	Visual Information System
2	Multimedia Database Management
3	Visual Information Content
4	Multimedia Content Store

Table 5.1 Sample Document with Four Logical Blocks

The word signatures for Table 5.1 are shown in Table 5.2. The bit positions are generated using a random number function, `random()`, in UNIX. Initially, the seed of the random number generator is set to a randomly picked prime number, 19937 (i.e. `srandom(19937)`). The random numbers are converted into the values

between 0 and 11 using modulus arithmetic. They are then applied in sequence to set the 12-bit vectors with 3 positions for each word signature. The least significant bit of the word signature is designated as position 0.

Word	Bit Positions	Word Signature
Content	1, 10, 7	010010000010
Database	6, 7, 5	000011100000
Information	6, 11, 11	100001000000
Management	2, 11, 7	100010000100
Multimedia	5, 8, 3	000100101000
Store	3, 3, 2	000000001100
System	0, 10, 0	010000000001
Visual	9, 10, 4	011000010000

Table 5.2 Word Signatures for Table 5.1 Using Random Number Generator

All word signatures in a logical block are then superimposed into a block signature using the logical OR operation. The four block signatures are shown in Table 5.3. To search for a word in the block signatures, any block signature is a potential candidate if it consists of all the corresponding bit positions as the query word. This, however, does not guarantee the existence of the word in the block signature. All the required bit positions may well be set accidentally by different words in the logical block. If this is the case, it is called a *false match*. Any potential candidate is then sequentially searched to make sure that the word is existed. For

examples, a query with the word "Visual" requires the 1 bits set at positions {4,9,10} in the logical blocks. Blocks 1 and 3 are the potential candidates. Further scanning on the blocks indicates that both are valid answers and not false matches. On the other hand, a query "Information" produces a false match on the second block after scanning the potential candidates of the first three blocks. It is noted that a conjunctive query on the block level is easy to perform. For example, to search for "Visual Information", one has to superimpose the word signatures of "Visual" and "Information" and becomes a query signature to the block signatures. Since the query is more specific, less signatures can be the potential candidates. The retrieval is faster as a result. This is a desirable property.

Block	Block Signature
1. Visual Information System	111001010001
2. Multimedia Database Management	100111101100
3. Visual Information Content	111011010010
4. Multimedia Content Store	010110101110

Table 5.3 Block Signatures for Table 5.1

Sequential search to the signature file can be slow particularly for large text databases. Many methods have been suggested to accelerate the response time. Compression, vertical partitioning and horizontal partitioning are the main approaches to speed up the access which we will briefly describe next.

Compression

We can view the signature file as a $n \times b$ matrix where n is the number of logical blocks and b is the size of the word signature in bits. Normally, we are not able to control the value n, but we can, for sure, make the value b larger to reduce collisions. This will make the word signatures much sparser and increase the compressibility of the matrix.

Many lossless compression techniques described in Chapter 3 are suitable to compress the signature matrix. Run-length encoding is one of them. Since our Bitplane Signature described in the next section is also sparse in nature, compression is very much suitable for Bitplane Signature as well.

Vertical partitioning

The idea behind vertical partitioning is that the organization of the signature matrix can be done vertically. If each bit position of the signature matrix is stored separately, it is only necessary to retrieve m bit files which correspond to the 1 bits in the query signature. This is much better than performing sequential search on the signatures in row order. This technique is called *bit-slicing*. Table 5.4 shows the bit-sliced files for the block signatures of Table 5.3. There are a total of 12 bit files. Each slice consists of four bits. Generally, the slices are much longer than these. To look for a word such as "Visual", we need to fetch only bit files 4, 9 and 10. The potential candidates can then be easily identified by bitwise **and** operations on the slices. In

other words, 1010 ^ 1010 ^ 1011 = 1010 reveals quickly that the first and third logical blocks contain the potential candidates.

Bit	Slice	Bit	Slice
11	1110	5	0101
10	1011	4	1010
9	1010	3	0101
8	0101	2	0101
7	0111	1	0011
6	1110	0	1000

Table 5.4 Bit-sliced Files for Block Signatures in Table 5.3

Bit-slicing is not without problems. It is very expensive to update and delete words in documents if signatures are organized in bit-slices. One has to fetch all the bit files and changes the corresponding bits for minor updating operations. Hence, bit-slicing is only suitable for static documents involving little updating. Another problem is the size of each bit-slices. Large text databases will generate long slices. The performance issues of retrieving and comparing long slices cannot be ignored. Hence, methods such as frame-sliced signature files and blocked signature files are suggested. Compression can also be used to compact long slices.

Horizontal partitioning

The idea behind horizontal partitioning is that the signature matrix is partitioned horizontally by grouping similar signatures together. This will improve the searching time as non-similar signatures are avoided initially. There are a number of methods to partition the signatures horizontally. Methods commonly used in relational databases can be used as the accessing data structures.

Similar to extendible hashing in relational databases, the common prefix approach can be used to partition the signatures. Buckets of signatures are dynamically expanded and coalesced. Depending on the signature characteristics, the performance of better than *O(N)* search time is certainly expected.

Another common technique to group signatures is by tree structures. B- or B^+- trees can be used to organize the signatures. Although the performance is theoretically at the best of $O(\log_k N)$ for B- or B^+- trees of order k, the advantage of grouping similar signatures into the same subtree eliminates unnecessary disk access. Hence, the practical consideration of searching large numbers of signatures can not be ignored.

For image retrieval, the Hierarchical Composite Bitplane Signature exploited in Section 5.4.4 takes the direction of horizontal partitioning described in here. We would like to use tree-based data structures to avoid the sequential scanning of Composite Bitplane Signature.

5.3 Bitplane Signature

To introduce Bitplane Signatures for image retrieval, we shall use the metaphor of signing a signature on a document (e.g. a cheque). Every individual has his/her own signature, which can be an easily recognizable or obscure pattern. The purpose of signatures is to uniquely associate an individual to his/her signature. If we project a signature onto a rectangular grid, the ON bits are roughly on the same locations every time the signature is signed.

For image retrieval, we can use a pre-defined size of bitplane as the "signature canvas". A hash function or transformation is chosen so that we can translate the image into a set of bit patterns and onto the bitplane. It serves as the signature of the image. In fact, the signature does not have to be directly related to the geometrical properties such as edges and shapes within the image. Ideally, the signature is an injective mapping from the image source to the bitplane. Different images will give rise to different signatures. However, if a $n \times m$ color image is transformed to a $n \times m$ bitplane, an injective mapping is theoretically impossible. Therefore, near injection is the best we can hope for. One may argue that, to reduce collisions, we can use a $n' \times m'$ bitplane where $n << n'$ and $m << m'$. Unfortunately, this makes the bitplane much more difficult to handle and also against our goal of using signatures for fast image retrieval.

Another fundamental issue for Bitplane Signature is the behavior of the transformation from images into bitplanes. In text retrieval, we want the hashing functions to be as random and uniform as possible. This, however, may not be the

case for image data. For example, we may not want to directly generate the random patterns for the bitplane signatures as in the text counterpart. Unless we can keep track of the randomly generated bitplanes, this solution is clearly unworkable. It is true that we want to avoid collisions among bitplane signatures in a given image database as much as possible. This, however, cannot undermine the importance of similarity retrieval for image data. Hence, the transformation scheme must be able to captures the salient characteristics of the images.

There are a number of desirable properties which we want the bitplane signature scheme to exhibit. They are exploited in Section 5.3.1. Having the goal for the desirable signature scheme, we propose and design our signature scheme using wavelet decomposition in Section 5.3.2. Using wavelet decomposition for image retrieval has been proved in literature that it is indeed a good way for similarity retrieval [JACO95, STOL96]. The method has been adopted and modified by us to fit into our overall model. In Section 5.3.3, we explore other possibilities for signature generation.

5.3.1 Desirable Properties for Bitplane Signature

Many of the desirable properties for Bitplane Signature are not found in text signatures. For example, content-based image retrieval is important for pictorial data. Therefore, we would like the transformation to provide the capability of similarity retrieval. Ideally, we would like the signature scheme to exhibit the following properties:

1. A precise retrieval of an image if the corresponding signature is used to query the database. Otherwise, the particular signature scheme is not useful at all.

2. The signature scheme will produce a similar signature if similar images are given. This is very important for the organization and retrieval of images.

3. The computational cost for comparing two signatures should be very efficient. Otherwise, given a query image, the overall performance of retrieving the target images will suffer if the database is very large.

4. The data structure to organize the signatures should be highly searchable and easily manipulated.

5. Collisions in signatures are unavoidable. Ideally, we would like the hash patterns evenly distributed. However, this is proven by our experiment in Chapter 7 that it is generally not possible as in other hashing contexts. This is because the nature of distribution is dictated by the chosen hashing scheme. Once a hashing algorithm is chosen to optimize similarity retrieval, the nature of the resultant collisions will be determined by the underlying distribution, collisions will be resulted in trend with the distribution.

6. If the compressibility of the signatures is high, we may be able to store all the signatures in memory for a moderate size of image collections.

Furthermore, it is much easy to handle the signatures if they are highly compressed.

Admittedly, we may not able to design a bitplane signature scheme with all the desirable properties listed above. However, we should satisfy the first three properties in order to make the designed scheme useful.

5.3.2 Using Wavelet Coefficients for Signature Generation

The selection of progressively better signature scheme is an ongoing research. Basically, we are looking for a good hashing scheme which can translate an image into a set of signature bitplanes. As long as the scheme generates a reasonably unique and sparse signature bitplane and possesses the properties described in Section 5.3.1, we can use it for the purpose of generating image signatures. The technique of retaining the signs of the m largest magnitude of wavelet coefficients proposed in [JACO95, STOL96] provides us a good start for signature generation. They originally suggested that the multiresolution nature of wavelets could be used as an effective scheme for image retrieval. Their technique of image retrieval using wavelets can be found in Section 3.3.3. Indeed, using wavelet decomposition for the purposes of signature generation offers a number of distinct advantages as follows:

1. A good image approximation can be achieved by just a few coefficients. To demonstrate this point, a simple example is illustrated in Figure 5.1. The famous image, Lena, is scaled to 128x128 and restored from substantially reduced coefficients. It is noted that 10 to 20 percent of the

largest magnitude of wavelet coefficients can reproduce Lena with reasonable clarity.

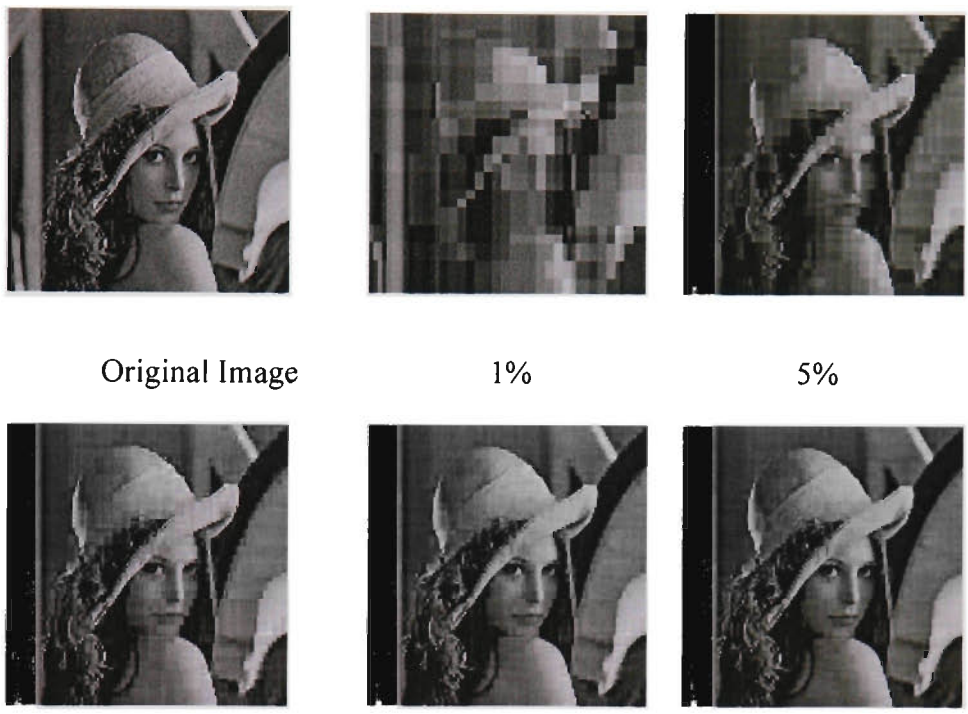


Figure 5.1 Representing Lena by Fewer Wavelet Coefficients

2. It is very simple to implement and reasonably fast to compute. This is particularly true for the wavelet decomposition using Haar basis functions. Although the mathematical foundation of wavelets is quite involved, the filter bank is very simple to understand as described in Chapter 3. Furthermore, the heart of the wavelet decomposition and composition can be programmed with just a simple C++ program as follows:

```
#define root2     1.414213562
#define root2div2 0.707106781
// ********** Haar Wavelet Decomposition **********
void DecompositionStep(float *C, int n) { // C[1..n]
     int i;
     float *T;
     T = new float [n];
     for(i=1; i <= n/2; i++) {
        T[i] = ( C[2*i-1] + C[2*i] ) / root2;
        T[n/2+i] = ( C[2*i-1] - C[2*i] ) / root2;
     }
     for(i=1; i <= n; i++) C[i] = T[i];
     delete T;
}
void Decomposition(float *C, int n) { // C[1..n]
     int i;
     for(i=1; i <= n; i++) C[i] = C[i] / sqrt(n);
     while (n > 1) {
        DecompositionStep(C,n);
        n = n / 2;
     }
}
// ********** Haar Wavelet Composition **********
void CompositionStep(float *C, int n) { // C[1..n]
     int i;
     float *T;
     T = new float [n];
     for(i=1; i <= n/2; i++) {
        T[2*i-1] = ( C[i] + C[n/2+i] ) * root2div2;
        T[2*i] = ( C[i] - C[n/2+i] ) * root2div2;
     }
     for(i=1; i <= n; i++) C[i] = T[i];
     delete T;
}
void Composition(float *C, int n) { // C[1..n]
     int i;
     i = 2;
     while (i <= n) {
        CompositionStep(C,i);
        i = i * 2;
     }
     for(i=1; i <= n; i++) C[i] = C[i] * sqrt(n);
}
```

Figure 5.2 C++ Source Codes for HWT

The *standard decomposition* for two-dimensional image data is by repeatedly calling `Decomposition(...)` to process each row of pixels. Next, the transformed rows are treated as an image and the

columns are again processed by repeatedly calling `Decomposition(...)`. The programming effort is amazingly simple and the computational cost is reasonably fast in comparison with frequency domain transformation.

3. Not only wavelet decomposition has shown great potential in image compression, it has been proven to be an effective way for image retrieval in recent years. Because of its multiresolution nature, the coarser image can be represented by very few coefficients as illustrated previously. Hence, the coefficients can be used to capture the salient features of the images and provides a good starting point for image indexing and retrieval.

Figure 5.3 outlines the processing steps of our signature scheme. It is our view that the raw data should undergo a pre-processing step before any wavelet decomposition takes place. Not only the usual conversion of different color spaces are performed, but most importantly, this step should include measures to achieve higher degrees of translational, rotational and scale invariance [GEVE96]. For example, a pre-processing step of bin-packing the color histogram [VELL95] is geared toward this goal. A pre-processing step for our signature scheme is scaled different sizes of images into a standard size, n_x x n_y. Scale invariance is being taking care of using a fixed size image before any signature is generated. The scaled images are then converted to YIQ color spaces. YIQ color spaces are used instead of RGB color spaces as the latter is not color invariance. The scaled images are then undergone standard Haar wavelet decomposition for each color channel, and a set of 6 bitplane

signatures are generated as a result of retaining the signs of the m largest magnitude of coefficients. m, n_x and n_y are design parameters. These parameters are directly affected by the behavior of wavelet coefficients and are the subjects of investigation in our experiments in Chapter 7. Once the parameters are chosen, they will not be altered for the duration of the database. This design decision is similar to the text signatures. Also, this is for sure that $m << n_x * n_y$ in order to make the bit matrices very sparse.

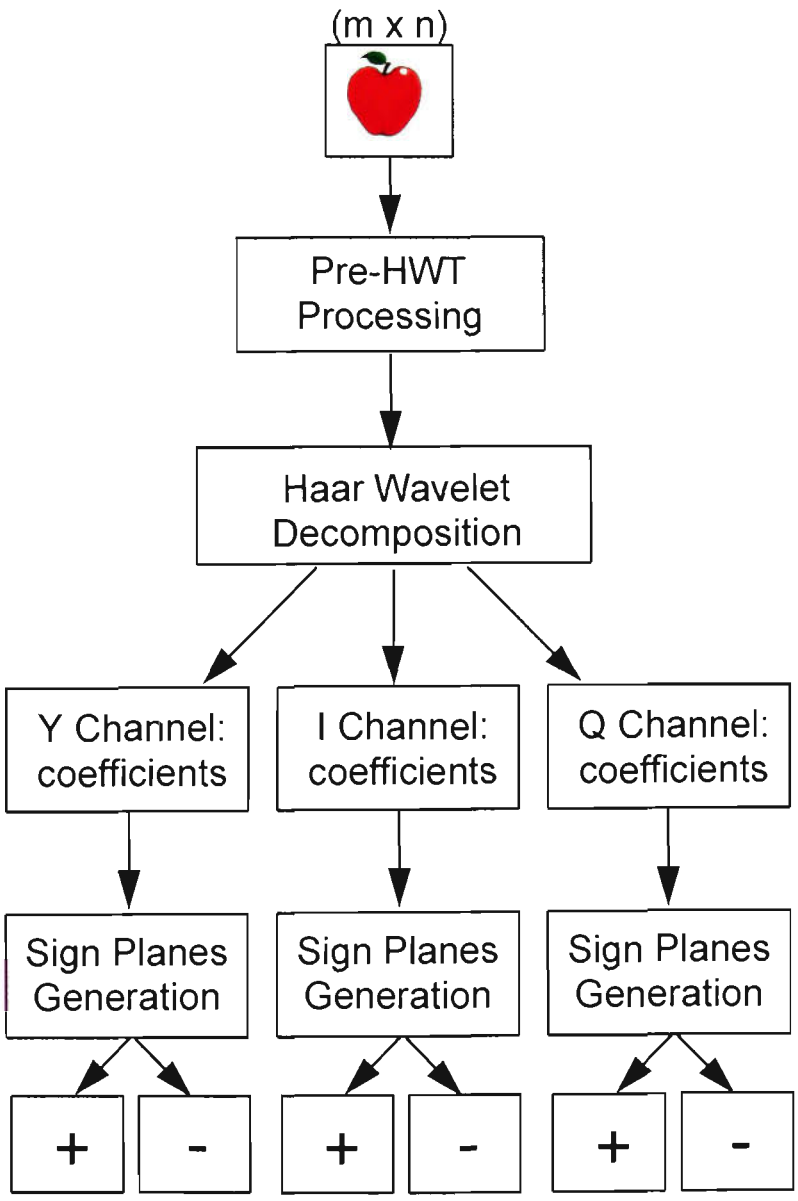


Figure 5.3 Bitplane Signature Scheme Using Haar Wavelet Decomposition

On the surface, in numbers of bits, the size of the bitplanes appears to be prohibitively large. This is not true at all. On the contrary, the storage requirement for our bitplane signatures is amazingly small. As determined by experiment, only approximately 150 bytes are needed for a 128 x 128 sign bitplane (with m = 128) in compressed form. Hence, we can store all six sign planes of an image with just a few hundred bytes.

5.3.3 Other Possible Schemes for Signatures Generation

Bitplane signatures can be constructed from many methods. Many hashing schemes, which can produce two-dimensional bitplanes, are possible for signature generation. One obvious scheme is to convert from color images to bi-level images. This scheme, however, has far too many bits on a bitplane and sensitive to noise. Other schemes based on transformation such as FFT or cosine transforms are possible candidates. As a matter of fact, many content-based image retrieval techniques are also derived from this approach. Although the frequency domain does not resemble the spatial domain, the spectrum can be used to deduce the signatures. After all, signatures are for machines to read and they do not need to be human readable.

We must stress that signature generation using wavelet coefficients in previous section is only one of many possible schemes. Because of the multiresolution nature of the coefficients, it provides a convenient way to abstract the images as well as fitted into our overall concept of bitplane signatures. Nevertheless, it is ongoing research to look for a better signature scheme.

5.4 Composite Bitplane Signature

As we have seen in Section 5.2, we can superimpose word signatures into block signatures. This concept is equally applicable to bitplane signatures. The motive of doing this is similar to the text counterpart. That is we do not want to search the bitplane sequentially. We would like to search the bitplanes in groups so that better than linear search can be achieved. The formation of Composite Bitplane Signature is described in Section 5.4.1.

In fact, most of the image retrieval algorithms and searching engines reported in the literature are based on *ranking* of similar images. Very often, not all the images are needed to be completely ranked. Only a partial ranking is required for the top most candidates. For example, the user may only request the top 50 images which can satisfy the similarity query. Therefore, it is wasteful in producing a complete list of ranked images. Furthermore, if the ranking system can extract the top ranked images without performing the similarity measure to the entire image collection, this is superior in performance to those systems which are required to compare and traverse all the images in the collection. Our unique ranking system for Composite Bitplane Signature, described in Section 5.4.2, allows us to produce the target set of images without the need to search the entire database. Regardless of the different composition of bitplane signatures, the consistency of our ranking system is always guaranteed if false matches are being examined. Moreover, if the Hierarchical Composite Bitplane Signature in Section 5.4.4 is implemented, the searching mechanism is likely to be much faster for partial ranking.

To appreciate the concept of Composite Bitplane Signature, a simple illustrative example is given in Section 5.4.3. It outlines the processes of generating Bitplane Signature using wavelet decomposition, the formation of Composite Bitplane Signature and the querying.

5.4.1 Formation of Composite Bitplane Signature

Given a set of bitplane signatures with similar patterns, we can overlay the bitplanes and form a composite bitplane signature as shown in Figure 5.4. If similar patterns are grouped together, we can superimpose more signatures than from the signatures of non-similar patterns. This makes the Composite Bitplane Signature more compact. In fact, it is possible to overlay non-similar patterns as long as we do not populate the overall bitplane excessively. Otherwise, the uniqueness of the composite bitplane signature will be eroded.

As in the block signatures for text retrieval, the bitwise OR operation is used to superimpose the bitplane signatures. For the gray-scale images, only the plus and minus composite bitplane signatures are resulted. For color images, we will produce the six composite bitplane signatures corresponding to the color channels of the images, $\{X^+, X^-, Y^+, Y^-, Z^+, Z^-\}$ where $\{X, Y, Z\}$ represents the color scheme of the user's choice.

For our signature scheme, the bits for the plus and minus bitplanes tend to be concentrated around the scaling coefficient (i.e. the coarsest subband) due to the multi-resolution nature of wavelets. Collisions are unavoidable. This characteristic

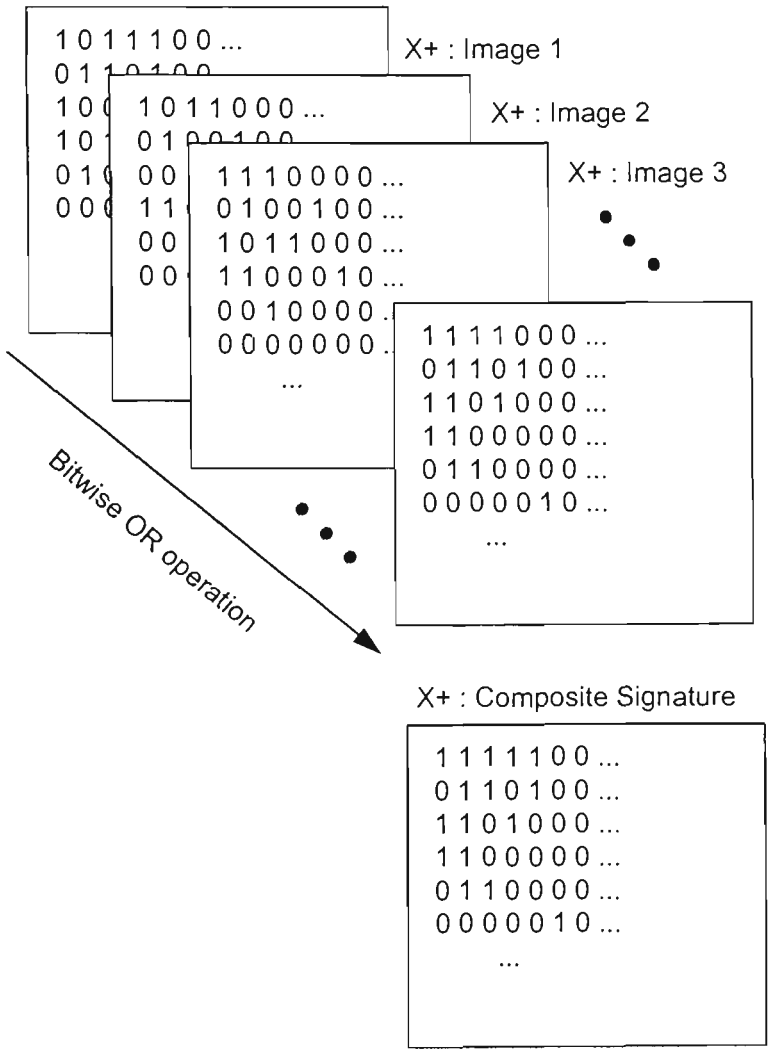


Figure 5.4 Formation of Composite Bitplane Signature

will limit our freedom to add too many signatures into any composite bitplane. Therefore, we would like to find out the behavior of forming Composite Bitplane Signature under the influence of wavelet decomposition. However, the analytical determination of behavior is difficult. Hence, we will observe this by experiments as given in Chapter 7.

5.4.2 Searching and Ranking

To search a particular signature embedded in a composite bitplane signature, we rank all the composite bitplane signatures not by the number of matched bits but by the *least* difference of 1 bits from the query signature. The logic behind this searching mechanism is that the likelihood of finding the target image in a composite bitplane signature is higher in those composite bitplane signatures having least difference. Therefore, it would be logical to search the individual bitplane signatures from those composite bitplane signatures having a higher chance of success first. Moreover, false matches are being taken care of in the searching process.

Expressed mathematically, the composite bitplanes $\{C^+, C^-\}$ for a set of sign planes $\{ I_k^+, I_k^-;\ 1 \le k \le N \}$ are defined as,

$$C^+(x,y) = \bigoplus_{k=1}^{N} I_k^+(x,y), \tag{5.1}$$

$$C^-(x,y) = \bigoplus_{k=1}^{N} I_k^-(x,y), \tag{5.2}$$

where $\oplus$ is a bitwise inclusive OR operator.

The individual score between the sign planes of the query image Q and the target image T for all three color channels are computed by the following equations,

$$S_i^+ = \sum_{x=0}^{m} \sum_{y}^{n} Q_i^+(x,y) \wedge \sim T_i^+(x,y), \tag{5.3}$$

$$S_i^- = \sum_{x=0}^{m} \sum_{y}^{n} Q_i^-(x,y) \wedge \sim T_i^-(x,y), \qquad 1 \le i \le 3. \tag{5.4}$$

The overall score for all three-color channels is the weighted sum of the individual scores,

$$S = \sum_{i=1}^{3} \left(a_i S_i^+ + b_i S_i^- \right). \tag{5.5}$$

With this measure, a higher score means a weaker match between the query image and the target image. We rank the target images by the least scores. Also, we can place different emphases on the color channels by assigning different weights to a_i and b_i . For example, if the color scheme is YIQ, ranking by intensity only (i.e. ignoring colors) can be performed by setting the weights for I and Q scores to zero.

5.4.3 Illustration

As a concrete illustration of Composite Bitplane Signature, let us assume that we have three 8x8 gray-scale images with data shown in Figure 5.5. Since they are gray-scale images, there are no need to generate all six bitplanes as it would be for color images. Only two bitplane signatures corresponding to the plus and minus bitplanes are generated. At first glance, the first and third images are very similar. Therefore, we will expect that the bitplane signatures are similar as we will see later on.

Under our scheme, we need to perform Haar Wavelet Transformation on the image data. We use the standard HWT decomposition. The wavelet coefficients are shown in Figure 5.6. They are round to the nearest integers. It is noted that we should perform pre-HWT processing according to our scheme described in Section 5.3. However, for this illustration, we omit the processing for clarity. But the processes of

0	0	0	64	64	0	0	0
0	0	72	255	255	72	0	0
0	80	255	128	128	255	80	0
88	255	0	255	255	0	255	88
255	0	0	255	255	0	0	255
0	255	0	255	255	0	255	0
0	0	255	128	128	255	0	0
0	0	0	255	255	0	0	0

(a) Image #1

3	7	11	15	19	23	27	31
7	15	23	31	39	47	55	63
11	23	35	47	59	71	83	95
15	31	47	63	79	95	111	127
19	39	59	79	99	119	139	159
23	47	71	95	119	143	167	191
27	55	83	111	139	167	195	223
31	63	95	127	159	191	223	255

(b) Image #2

0	0	0	64	0	0	0	0
0	0	128	255	255	0	0	0
0	128	255	128	128	255	0	0
128	255	0	255	255	0	255	0
255	0	0	255	255	0	0	255
128	255	0	255	255	0	255	0
0	128	255	128	128	255	0	0
0	0	128	255	255	0	0	0

(c) Image #3

Figure 5.5 Data for Three 8x8 Gray-scale Images

scaling and color translation are definitely required and performed in all our experiments described in Chapter 7.

It is clear from Figure 5.6 that the multiresolution nature of the wavelet transform is appeared in the coefficients. Larger magnitudes of coefficients are concentrated around the coarsest subband. This further confirms our observation that we cannot over-populate composite bitplane signatures. The m-largest magnitude of

97	0	-27	27	-8	-32	32	8
-6	0	1	-1	-8	8	-8	8
-30	0	-6	6	11	-5	5	-11
17	0	20	-20	0	-17	17	0
-16	0	12	-12	0	7	-7	0
-8	0	-17	17	5	24	-24	-5
0	0	0	0	32	0	0	-32
8	0	-6	6	0	24	-24	0

(a) HWT for Image #1

80	-36	-13	-13	-5	-5	-5	-5
-36	16	6	6	2	2	2	2
-13	6	2	2	1	1	1	1
-13	6	2	2	1	1	1	1
-5	2	1	1	0	0	0	0
-5	2	1	1	0	0	0	0
-5	2	1	1	0	0	0	0
-5	2	1	1	0	0	0	0

(b) HWT for Image #2

101	13	-24	28	-8	-26	32	8
-15	1	-1	0	-8	6	-8	8
-30	-1	-10	-4	11	-3	6	-11
14	-3	24	-20	11	-23	17	0
-18	-2	14	-11	0	4	-16	0
-8	0	-17	17	0	24	-24	-16
-4	-4	-6	0	24	0	0	-32
8	0	6	6	-8	16	-24	0

(c) HWT for Image #3

Figure 5.6 HWT Using Standard Decomposition for Figure 5.5

coefficients are selected to generate the corresponding plus and minus bitplanes. We chose m to be 8 which makes the bitplane signatures reasonably unique. The truncated coefficients are shown in Figure 5.7.

The selected coefficients are then used to generate the plus and minus bitplane signatures. The bitplane signatures for the three images are shown in

97	0	-27	27	0	-32	32	0
0	0	0	0	0	0	0	0
-30	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	32	0	0	-32
0	0	0	0	0	0	0	0

(a) Selected Coefficients for Image #1

80	-36	-13	-13	0	0	0	0
-36	16	0	0	0	0	0	0
-13	0	0	0	0	0	0	0
-13	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b) Selected Coefficients for Image #2

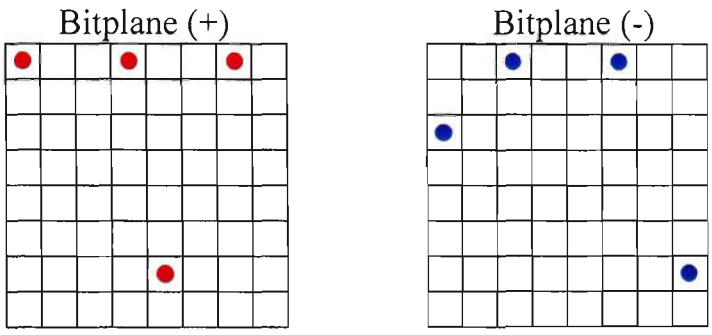
101	0	-24	28	0	-26	32	0
0	0	0	0	0	0	0	0
-30	0	0	0	0	0	0	0
0	0	24	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-32
0	0	0	0	0	0	0	0

(c) Selected Coefficients for Image #3

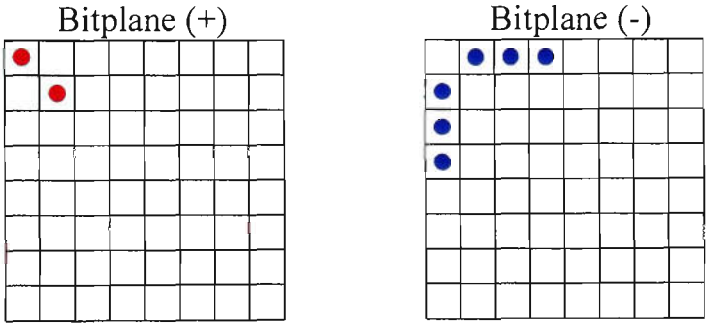
Figure 5.7 8-largest Magnitude of Coefficients from Figure 5.6

Figure 5.8. It is clear from Figure 5.8 that similar images will have similar bitplane signatures. This characteristic is very useful for similarity retrieval.

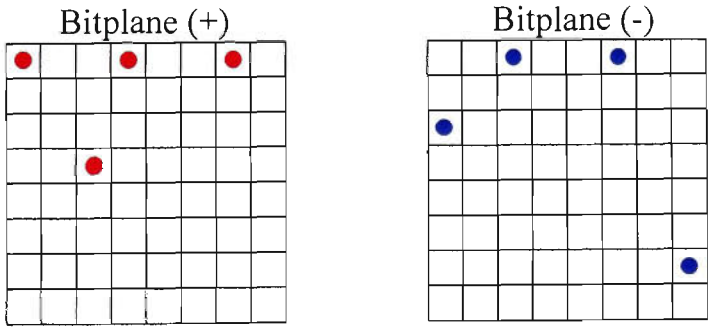
Given the bitplane signatures, we would like to illustrate the formation of composite bitplane signatures. Let us assume we intend to form composite bitplane signatures from the first and second images and use the third image as a query image



(a) Bitplane Signatures for Image #1



(b) Bitplane Signatures for Image #2



(c) Bitplane Signatures for Image #3

Figure 5.8 Bitplane Signatures for Figure 5.5

to the composite bitplanes. Figure 5.9 shows the formation of composite bitplane signatures by bitwise-ORing the corresponding plus and minus bitplanes.

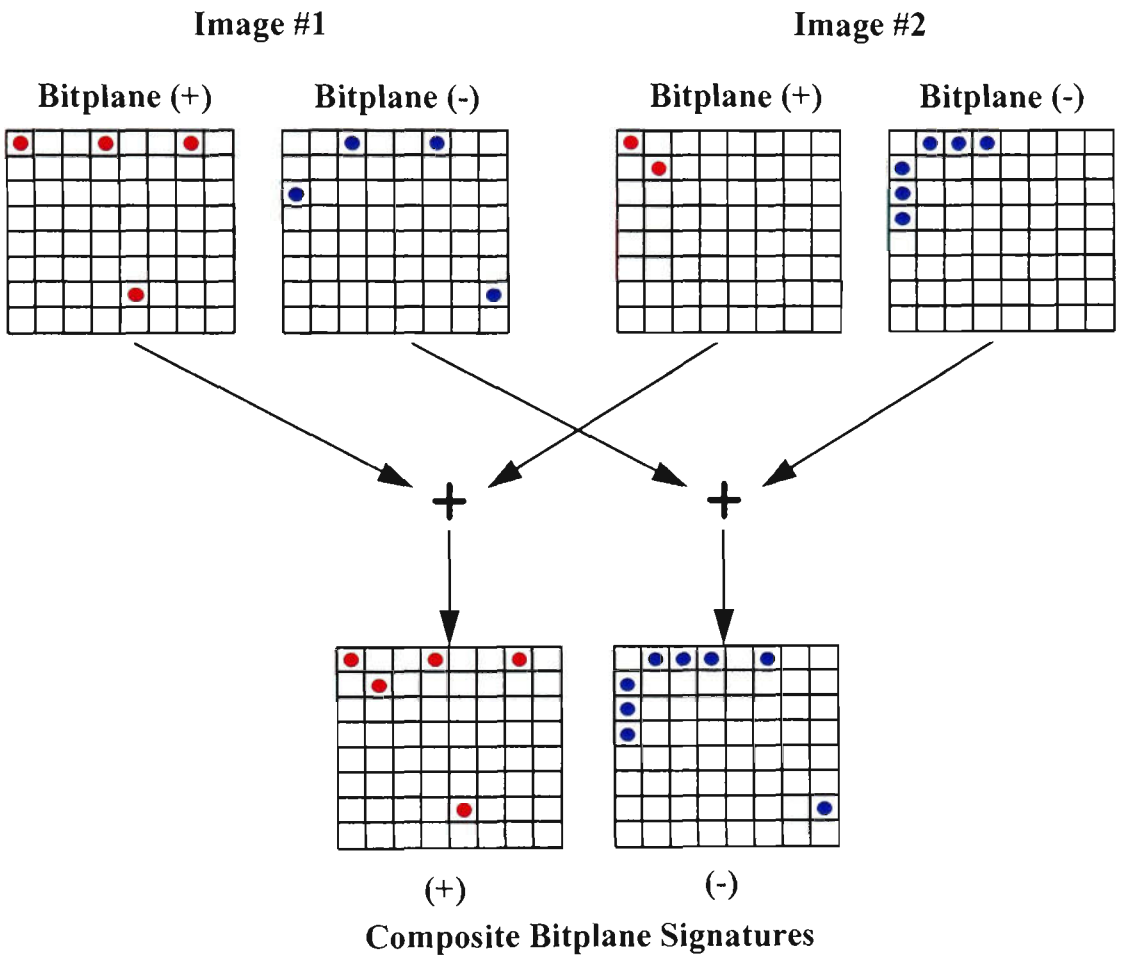


Figure 5.9 Formation of Composite Bitplane Signatures Using Image #1 and Image #2

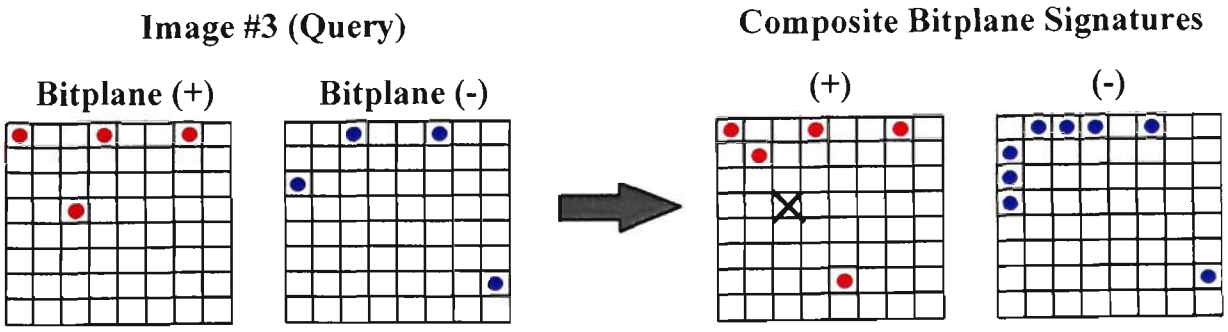


Figure 5.10 Query the Composite Bitplane Signatures in Figure 5.9

To query the composite bitplane signatures, we count the number of least difference of 1 bits to the query image. As shown in Figure 5.10, a total of only 1 bit is different in the plus bitplane and none in the minus bitplane.

From this example, a number of important properties can be observed from the bitplane signatures:

1. If a precise image is given, a precise match will be obtained. The composite bitplane signature, which has the signature embedded, will definitely come out in the top rank.
2. Similar images have similar signatures. For example, Image #1 and Image #3 are very similar. The bitplane signatures are also similar. This provides a good way to group/cluster images together.
3. The querying operation is very simple and efficient. For large image databases, it is extremely advantageous.
4. The data structure to hold the bitplane signatures is also very simple. The bitplanes can be packed into an array of words for storage.
5. To maintain the uniqueness of the bitplane signatures, we cannot over-populate the composite bitplane signatures.
6. Since the bitplane signatures are very sparse, it is ideal for compression.

This example is for illustration purposes only. The characteristics and behaviors of Composite Bitplane Signature using wavelet coefficients are the subjects of the experiments in Chapter 7.

5.4.4 Hierarchical Composite Bitplane Signature

The composite bitplane signature discussed above eliminates the need to examine individual bitplane signatures in the entire image repository. For any query image, we still have to search through the composite bitplane signatures sequentially. Although it is much better than searching the entire collection of images as many existing retrieval methods do, performance will suffer if the image database contains a large number of composite bitplane signatures. Because of our unique ranking scheme, a hierarchical accessing structure of composite bitplane signatures can be constructed for extremely fast image searching as shown in Figure 5.11. A partial ranked list of images can be rapidly located through traversal and back tracking. We call this Hierarchical Composite Bitplane Signature.

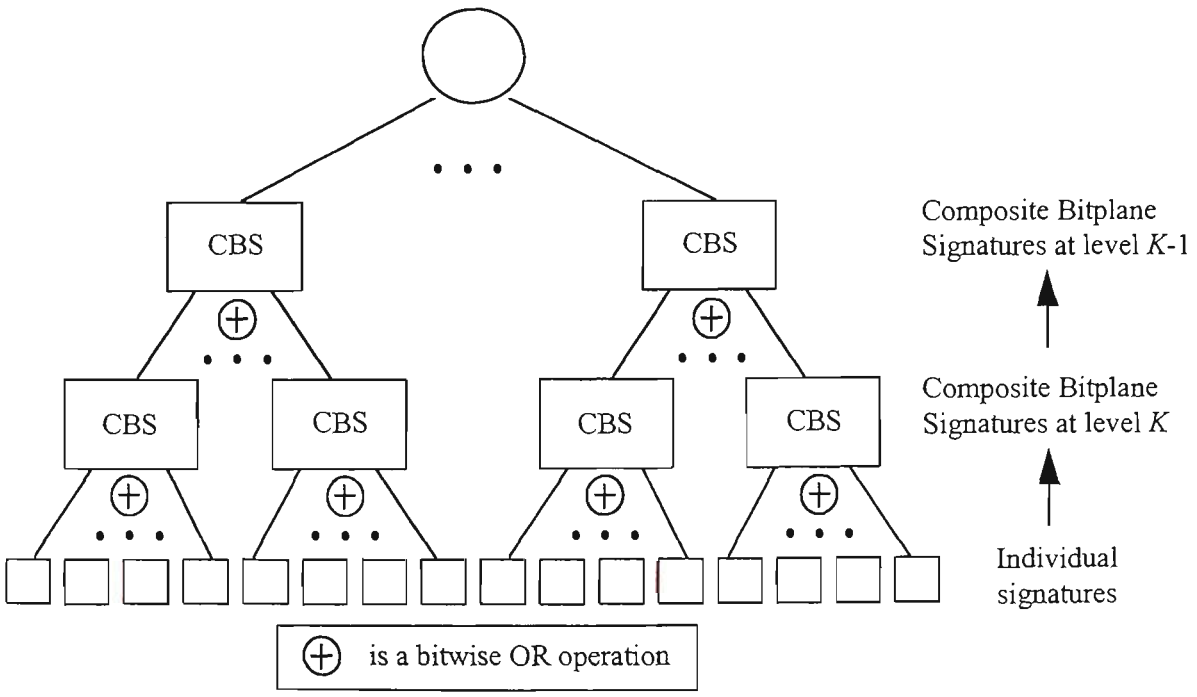


Figure 5.11 Hierarchical Composite Bitplane Signature

It is possible that other data structures can be used to organize the composite bitplane signatures. Data structures such k-d trees, quad-trees and R-trees are some of the possible spatial structures [SAME90a, SAME90b, SILB97]. These data structures can also take advantage of the subband nature of wavelet coefficients for fast retrieval.

5.5 Summary

In this chapter, the concept of Composite Bitplane Signature is presented. Composite Bitplane Signature is an approach for fast image indexing and retrieval. The basic building block of Composite Bitplane Signature is Bitplane Signature. It encapsulates the elegance of conventional signature-based text retrieval. This is accomplished by extending one-dimensional bit vectors for text signatures into two-dimensional bit matrices designed for image signatures.

Specifically, our design for Bitplane Signature enables us to retrieve images based on their visual contents. It is achieved by using the signs of m-largest magnitude of wavelet coefficients as the scheme for signature generation. Under our scheme, similar images will possess similar signatures. This fundamental characteristic is important for content-based similarity retrieval.

Composite Bitplane Signature is a superimposed coding technique for Bitplane Signatures. It avoids the need to individually examine every signature. Given a query image, the least difference of the composite bitplane signatures to the query signature is sought. They are the likely candidates containing the images similar to the query. Therefore, they should be examined first. Certainly, false matches are possible

and can be detected by further comparison on the individual signature composing the composite bitplane. This is similar to the resolution used in text retrieval.

Hierarchical Composite Bitplane Signature takes the concept of Composite Bitplane Signature one step further. It allows us to rapidly produce the partial ranked list of images by traversing a tree-like structure with back-tracking. This accessing structure is extremely useful for a large database. A simple comparison for the search efficiency is summarized in Table 5.5.

Signature Scheme	Search Efficiency
Bitplane Signature	N (N = no. of signatures)
Composite Bitplane Signature	$N/G + G$ (G = no.of signatures in a group)
Hierarchical Composite Bitplane Signature	$log_m N + m$ (if a balanced tree of order m is used)

Table 5.5 A Simple Comparison of our Image Retrieval Schemes

Our unique ranking and searching mechanism can efficiently produce a partially ranked target set of images without the need to search the entire database. This offers significant advantage over those retrieval algorithms which require the comparison of all the images in the database.

Chapter 6

Inverted Image Indexing and Compression

6.1 Introduction

Indexing of information content is vital for voluminous data. It is next to impossible to locate a piece of information if it is not properly indexed. For example, it is necessary to look up terms in the index of a book when searching for important terms. If an index is not provided, readers have to search the headings of chapters and sections and may eventually locate the information by a sequential scan to paragraphs. This is often a time-consuming and frustrating process. Therefore, the concept of *postings* provides a useful solution [HARM92, SALT89, WITT94].

Information indexing and retrieval, particularly alphanumeric information, have been heavily studied for the last 40 years. The most noticeable contributor in the

field is the late Gerard Salton from Cornell University where he and his students persistently worked on the areas of automatic indexing and information retrieval. The usefulness of automatic indexing for image data is prompted by the recognition of its importance in the text domain. Many paradigms and techniques in text retrieval can be exploited and improved upon. Hashing and signature schemes described in previous chapters are such examples. In this chapter, we take the paradigm of inverted files and apply to image data.

For image retrieval, however, using title, caption, or description of images for indexing purposes are not sufficient. In many cases, it is difficult to describe the visual effect by words. For instance, if a picture of a designer's dress is indexed by words, the descriptions for the material texture, shape, cut, feature and style would be lengthy. Hence, we must also index the contents visually, which is commonly the subject of Content-Based Image Retrieval (CBIR) [GUDI95]. However, it is also difficult and next to impossible to represent higher level concepts and semantic information using just lower level features and syntactic information. The model described in this chapter bridges the deficiency of both sides and provides a unified framework for accessing image data both semantically and syntactically.

To understand our motivation of applying the inverted paradigm to image data, a simple analogy of text and image contents is given in Table 6.1. If we view the counterpart of characters are pixels, then words in a document are equivalent to areas of pixels in an image. This comparison is rather crude but very appropriate in helping to establish the indexing and storage technique in this chapter; namely Inverted Image Indexing and Compression (IIIC). To proceed further, we can look at the formation of

queries in document databases and image databases. Users poses a query to a document database by supplying a list of words. Similarly, a query to an image database using visual contents is by sample pictures (Query By Picture Examples) to any possible area in images.

Text content	Image content
Document	Image
Word	Area
Character	Pixel
Word position	Area coordinates

Table 6.1 A Simple Analogy of Text and Image Contents

Many content-based image retrieval techniques are applied to images as the whole pictures. In many applications, multiple objects in images are required to be indexed separately. For example, a picture of a woman cuddling a baby contains two indexable objects [GROS97b]. Content-based feature extraction on the whole picture is neither appropriate nor meaningful with respect to an individual object. In other words, each image can have multiple areas which are perceived to be meaningful visual contents. Each area must be indexed on its own and related to similar objects in the entire image collection. This is analogous to text documents where each word is a searchable item. Inverted text files are used to group them together. Recent researchers recognize that it is important to consider multiple areas within images [CHAN97, GROS97b, SMIT97].

In this chapter, we firstly look at a simple example using an inverted file for textual documents in Section 6.2. This provides an interesting contrast to our inverted file for pictorial documents. In Section 6.3, the components of the IIIC model are presented in details. These include picture keys, the Ternary Fact Model representation (TFM), the Composite Bitplane Signature (CBS) and the image coordinates. In Section 6.4, the query model for the inverted paradigm is presented. This allows us to provide Boolean and ranked queries in a unified framework. Compression consideration for various components of the IIIC model is discussed in Section 6.5. Lastly, we conclude this chapter with a summary in Section 6.8.

6.2 Inverted Files for Text Retrieval

Indexing techniques for large text databases are pursued mainly in two directions. They are signature-based indexing and inversion-based postings. Of these two approaches, the inverted paradigm is the most natural way of searching information. We are very much used to this paradigm in our daily life. The obvious example is the concordance of a book. Another example is the street directory for which the inverted paradigm is applied to pictorial and spatial information. This is very much similar to our inverted schemes for images.

For completeness, we will examine a simple example using an inverted file for textual documents. This provides an interesting contrast to our inverted organization for pictorial information. Table 6.2 is an example of some proverbs and famous phases. To simplify the illustration, we assume each document consists of

only one line. The inverted file for the documents is shown in Table 6.3. The lexicon of the inverted file consists of all the words in the documents. We do not consider many usual transformations of words in producing the inverted file. In practice, a number of automatic indexing techniques [SALT89] would be applied to the text in order to improve the retrieval performance as well as the precision or recall. For examples, (1) *case-folding*: uppercase letters or words such as "Life" can be folded to lowercase, (2) *stop-words*: high frequencies of common grammatical words such as "the", "a", "but" and other words deemed to be irrelevant can be eliminated, and (3) *stemming*: words such as "Men" and "saved" can be reduced to their morphological roots.

Document	Text
1	Life is ours to be spent, not to be saved
2	I must be cruel only to be kind
3	Life is nothing but a dream
4	Men are cruel but man is kind
5	I have a dream

Table 6.2 Sample Documents

Term	{Document: Words}	Term	{Document: Words}
I	{2:1}, {5:1}	kind	{2:8}, {4:7}
Life	{1:1}, {3:1}	man	{4:5}
Men	{4:1}	must	{2:2}
a	{3:5}, {5:3}	not	{1:7}
are	{4:2}	nothing	{3:3}
be	{1:5,9}, {2:3,7}	only	{2:5}
but	{3:4}, {4:4}	ours	{1:3}
cruel	{2:4}, {4:3}	saved	{1:10}
dream	{3:6}, {5:4}	spent	{1:6}
have	{5:2}	to	{1:4,8}, {2:6}
is	{1:2}, {3:2}, {4:6}		

Table 6.3 Word-level Inverted File for Table 6.2

For each entry in Table 6.3, it consists of two main columns. The first column includes all the distinct and searchable words in the entire database. A number of organizational methods for these words are possible; (1) O(1) access using hashing if some sorts of word-to-address transformations or hashing functions are used, (2) O($\log_2 n$) access using binary search if these words are sorted, and (3) O($\log_k n$) access using balanced multiway trees such as B- or B^+- trees of order k. All these searching methods have their advantages and disadvantages. The ultimate selection is also dependent on other considerations such as storage.

The second column in Table 6.3 contains one or more lists of pointers organized in the form of { d: w_1, ... w_n } where d is the document number and w_i is the position of the given word in ascending order within the document d. This is a word-level inverted list which may be too expensive to perform on some large databases. Coarser indices referred only to the document numbers will reduce the storage spaces significantly but at the expense of scanning the entire text in the document sequentially. Also, proximity queries may result in false match at coarser indices. For example, the search for "Bill Clinton" may result from a document containing both "Bill Cosby" and "Hilary Clinton". Furthermore, fetching and scanning documents to satisfy basic conjunctive and disjunctive queries on multiple terms are likely to be slower.

Compression can play a major role in reducing the storage required by inverted files. Particularly, the list of word positions for an entry can be very long for large documents. Differential coding for each list followed by entropy coding can be used to compress the list. This leads to the so-called compressed inverted files.

6.3 Image Indexing and Retrieval Using Inverted Paradigm

Information retrieval, in principle, is based on the similarities between queries and stored contents. Regardless of the media types, this principle holds for any type of retrieval. Also, information retrieval can be broadly classified into two categories; namely high-level semantic discovery and low-level syntactic recovery.

Naturally, the results from semantic queries are less precise than syntactic queries. Image retrieval is no exception in this. For example, to answer a low-level query of locating images with certain color decomposition, the query can be easily satisfied and should be reasonably accurate. On the other hand, the semantic queries are much more difficult to answer. Human intelligence and intervention are often needed. Therefore, a retrieval model should ideally include mechanisms for high-level semantic indexing and low-level syntactic feature retrieval. Figure 6.1 outlines the conceptual image retrieval model which includes mechanisms for high-level semantic indexing and low-level feature extraction. This model is in closely resemblance to the text retrieval model given in [SALT89].

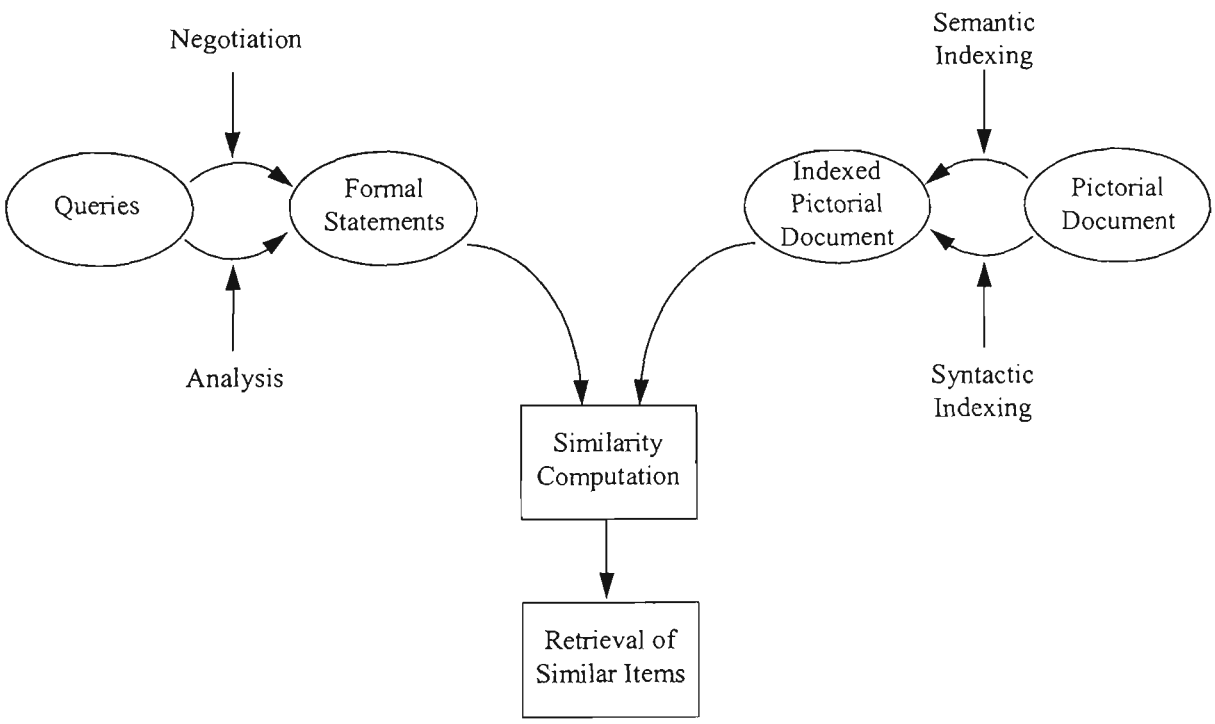


Figure 6.1 Conceptual Image Retrieval Model

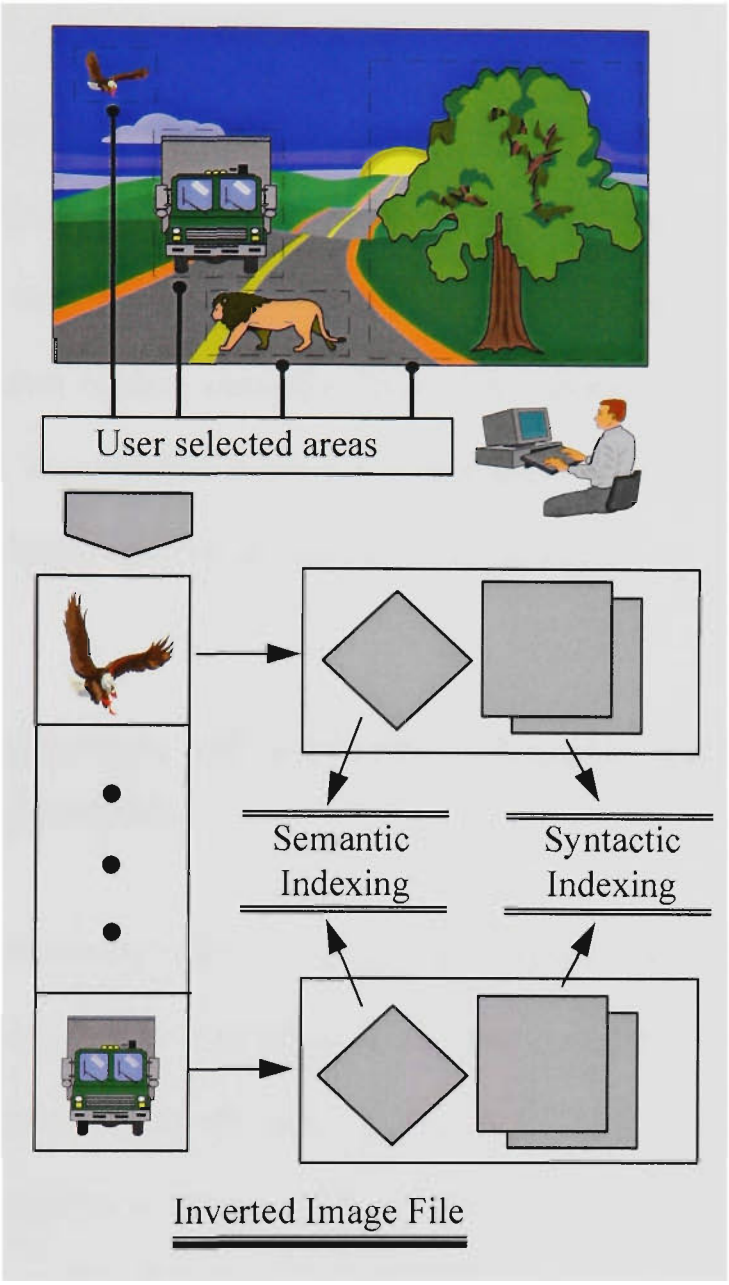


Figure 6.2 Image Indexing Using Inverted Paradigm

By recognizing the importance of semantic and syntactic indexing, our inverted paradigm for pictorial documents must include mechanisms to satisfy these needs. Furthermore, each pictorial documents can have multiple areas which are perceived to be meaningful visual contents. These areas are selected by users and undergo content specification at the semantic level and signature generation at the

feature or syntactic level. Figure 6.2 shows our view of image indexing using inverted paradigm.

It is shown in Figure 6.2 that the selected areas are limited to rectangular shapes. This restriction is for efficiency in handling and computation only. In theory, irregular regions can be used for better representation. However, it is tedious and impractical to sketch regions manually. Even with current technology in Machine Vision, automatic segmentation is proved to be a difficult task for arbitrary shapes. Also, overlapping areas are allowed in our inverted paradigm.

6.4 Components of Inverted Image Indexing and Compression

Unlike the inverted text file which has a complete and precise lexicon, we cannot directly use each area in an image as a searching 'vocabulary' for the inverted file. Instead, we must capture the salient feature of the area and translate it into a searchable representation. In our approach, we use two forms of representation; high-level specification using Ternary Fact Model [LEUN95] and low-level signature generation using Haar Wavelet Transform [JACO95, STOL96]. We use TFM because of its canonical nature in description. Hence, the searching mechanism is highly structured. With regard to the signature generation, we adopt the wavelet decomposition using the Haar basis. Composite Bitplane Signature described in Chapter 5 is also used for efficient searching. To illustrate the components of our Inverted Image Indexing and Compression model [SO97b], let us look at a simple

example of three pictures in Figure 6.3. The inverted file for image contents is given in Table 6.4. There are four major components of our model; the compressed area (picture key), the Ternary Fact Model representation, the Composite Bitplane Signature and the coordinates of each area.

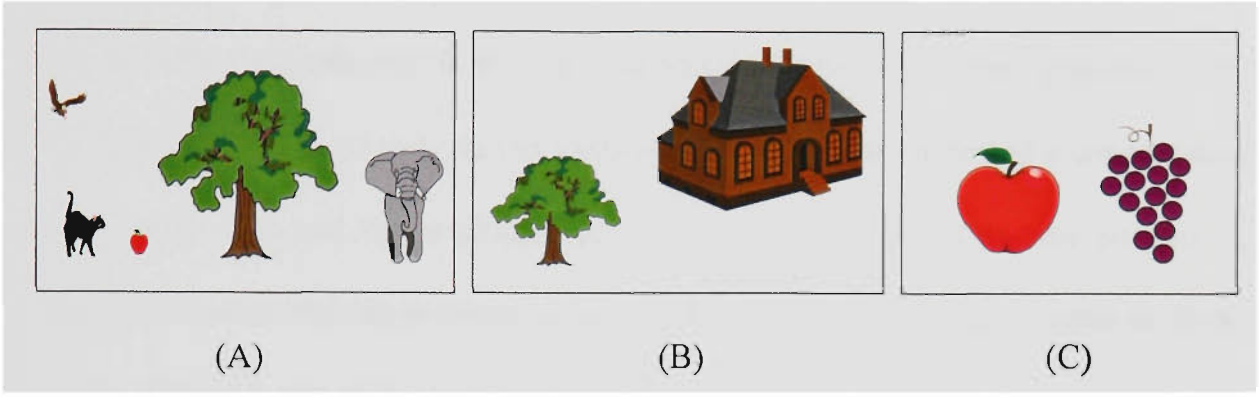


Figure 6.3 Three Sample Images

Area (compressed)	Ternary Fact Model	Composite Bitplane Signature	{ Image: Coordinates }
	red APPLE	0,1, ...	{A:(x_1,y_1)-(x_2,y_2)}, {C:(x_3,y_3)-(x_4,y_4)}
	black CAT	1,0, ...	{A:(x_5,y_5)-(x_6,y_6)}
...	...	...	...
	TREE	1,1, ...	{A:(x_k,y_k)-(x_l,y_l)}, {B:(x_m,y_m)-(x_n,y_n)}

Table 6.4 A Simplified View of Inverted Image File for Figure 6.3

The inverted image file described in Table 6.4 is for illustrative purposes. We will discuss how to construct the components of each entry next.

6.4.1 Picture Key

For each entry of the IIIC model, we need to provide a visual representation to signify what are included in the entry. In the simplest case, the entry has only one item which corresponds to an area from the original image. Otherwise, this image serves as a representative item among the collections within the entry. There is no need to have a precise replica of image contents in order to put them into the entry. The sole objective is to provide the users with the salient features and characteristics of the entry. We call this a picture key. For example, one can put the portrait of President Clinton into the picture key area. Pictures, which have Bill Clinton in them, can be cropped out and inserted into the entry. These pictures may include his childhood, as a Governor, with Monica Lewinsky, or with some other politicians.

The picture keys are stored in compressed form. For the purpose of picture keys, the compression can be a lossy one. As they are served as a representative item within the entry, they are often used for browsing or thumbnail purposes. Hence, the compression scheme should be reasonably fast and universally accepted. We select JPEG as the media for storage purposes.

6.4.2 Ternary Fact Model

Instead of using unstructured text to describe the image contents, a canonical description called Ternary Fact Model is used to capture the semantics of the images [LEUN95]. The semantic information is captured through manual indexing. A data model such as TFM is able to provide a set of homogenous structured data items to

facilitate searching. The basic building block of the model consists of discrete *facts*. Conceptually, image facts may be classified into five types:

1. *Elementary Facts*. An elementary fact merely states the presence of a particular item in the image.

   Examples: CHAIR, GRAPE, TREE.

2. *Modified Facts*. These are the elementary facts augmented with descriptive properties through the use of modifiers.

   Examples: red APPLE, black CAT.

3. *Outline Facts*. These outline the abstraction of the images.

   Examples: EXPLOSION, WEDDING.

4. *Binary Facts*. These are the facts linking together exactly two elementary or modified facts. Such links may correspond to verbs.

   Examples: BOY Rides ELEPHANT, WOMAN Holds CHILD.

5. *Ternary Facts*. These are the facts linking together exactly three elementary or modified facts together.

   Examples: BOY Hitting CAT With RACKET, WOMAN Giving CHILD BISCUIT.

For the examples given above, the following convention is adopted:

- Elementary facts and outline facts are indicated by upper case letters.
- Modifiers are indicated by lower case letters.

- In binary and ternary facts, the link is expressed using an initial upper case letter.

The above convention is important to facilitate the searching process if the facts are stored in free-text forms. Using free-text to store the canonical description has its advantages: (1) its data structure is much simpler as only strings of characters are needed for the storage, (2) text databases can be used to provide efficient searching, and (3) it is more natural to the users. The obvious disadvantage is that, in comparison with housing all the facts in different attributes of a relational database, it may require additional processing efforts at query time. Furthermore, it is time-consuming to process the strings if weighted sums or ranking of different facts are used. Querying issues will be examined in Section 6.4.

Pre-processing of facts are often needed to improve the precision or recall rates. Although the data model itself greatly alleviates the ambiguity of English-language texts, it is necessary to perform some forms of transformation as follows:

- Words are reduced to their morphological roots. This will narrow the queries and, hence, improves the precision rate. For examples, "Hitting" and "Rides" are converted to "Hit" and "Ride" respectively.

- Thesauri may be invoked if the exact matches are not found. This will broaden the queries which, in turn, enhances the recall rate. For example, if "LORRY" is not found in the facts, it will be substituted by "TRUCK" for further searching.

- Subject classification schemes similar to the Library of Congress can be used as the search space reduction strategies [SO96c, SO97a]. It is likely to reduce the set of images considerably. Whether it will improve the precision or the recall of the query really depends on the classification itself.

Elementary facts, modified facts and outline facts are used more often when the image contents are relatively simple. This is particularly true for our experiments which mainly consists of images with simple objects. We also decided to use slightly generalized facts for our inverted files so that a number of image areas can be inserted into a single entry. This provides an implicit grouping of areas with similar contents.

6.4.3 Composite Bitplane Signature

For each entry in the inverted image file, the inverted list may include a number of image areas. These areas may come from a single image or multiple images. To facilitate the low-level searching, each image area is required to undergo signature generation. A set of YIQ bitplane signatures are computed for each image area in the list using the wavelet decomposition of the Haar basis. They are then combined into composite bitplane signatures or added to the existing composite bitplane signatures if the entry is already existed. Composite bitplane signatures are thoroughly explained in Chapter 5 and we will not repeat in here.

The composite bitplane signatures provide a quick searching facility for all the image areas in each entry. The best ranked composite bitplane signatures indicate

that the likelihood of the target image is embedded in the entry. Individual bitplanes are further analyzed to see whether a false match has occurred or not. Depending on the user's requirement, we can cease searching the composite bitplane signatures as soon as the number of ranked images are reached. Due to our unique ranking technique described in Chapter 5, the searching process will always lead to the closest entry first. For example, if an identical image area or a very close resemblance is given to test all of the composite bitplane signatures in the inverted file, the top ranked image areas will have a score of zero (i.e. perfect match). For integrated query mixing TFM and CBS in a query, this characteristic is very useful for the querying of the image database.

Although the image areas are conceptually related and put together in a group under a single entry, the low-level feature signatures may be quite different. This does not really create much problems to the signature retrieval. After all, it is unnecessary to produce a complete ranked list of images. Only the top ranked images under user's specification are retrieved. On the other hand, if too many image areas are included in each inverted list, false matches will increase. In this case, the performance will degrade. Hierarchical composite bitplane signatures described in Chapter 5 can be used to improve the search performance.

6.4.4 Image Coordinates

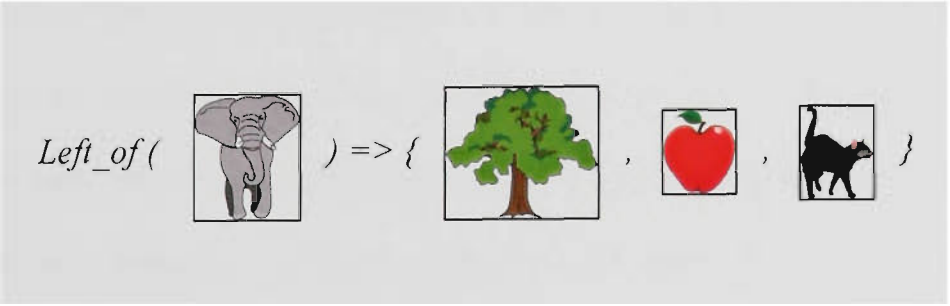
The coordinates in each entry of the inverted file indicate where the image area comes from. It takes the form $\{ I: R_1 \ldots R_n \}$ where I is the image identifier (ImageID) and R_i is the bounding box of each area within an image. Each image may

consist of a number of related areas. For example, two or more apples in an image is cropped out and entered into a single entry. All coordinates in this component are measured relative to the upper left corner (0,0) of the original image. If the visual content covers the entire image, the coordinates will reflect the width and height of the image. Image identifiers can either be the unique frame numbers or the file names within the image database. If the image collection spans across different platforms or systems, image identifiers must be appropriately designed. In our experiment, we just use the file names.

There are a number of practical uses of the coordinates in our model. Proximity, spatial relationship, foreground and background relationship are some of the useful information about the image areas. Some of the information deduced from the coordinates are:

1. Relationship among visual contents.

   We can deduce many spatial relationships by using the orientation of the visual contents. For example,



   This relationship can be computed by using the centroid from the coordinates. Of course, the criteria and tolerance to a spatial relationship are just a matter of programmable specification.

2. Foreground and background relationships.

   Each area in the inverted file represents the visual content of interest. We may consider these areas as the foreground contents. Different compression ratios can be applied to the foreground and background contents over the entire images to conserve storage spaces. For example, *Complex Tiling* defined in JPEG Extension [ISOD3] could be used to implement this kind of multiple compressions requirement within an individual image.

We can make use of additional mechanisms such as 2D-String [CHAN96b] to enrich the deduction of spatial relationships (e.g. nearness, overlapping, surrounding and many others). Furthermore, conjunctive queries, disjunctive queries and even relational joins are possible.

6.5 The Query Model

For any database whether it is for alphanumeric or multimedia information, a query model is an important component next to the data model. The query model works hand-in-hand with the data model to complete the conceptual framework of a database. For any advanced database, particularly multimedia ones, two broad categories of queries should be supported. The first category of queries are exact in nature. It is also referred to as Boolean queries and commonly used in relational databases. For example, we can ask for any image in the database containing "black CAT **and** red CHAIR". The answer ought to be very precise if such conjunctive query

is posed to the database. The second category of queries are ranked queries. It is also referred to as similarity queries. Using our previous example, any image closely relevant to CAT and CHAIR will be retrieved as the ranked candidates. One can view a Boolean query as really a special case of a similarity query. In other words, if the data set resulted from a Boolean query is not empty, it should be ranked at the top for any similarity query to be reasonable and meaningful. In image databases, the majority of queries are similarity queries due to the fact we do not have a robust technique to formulate and identify image contents.

There are numerous types of queries in image retrieval. Description-based, content-based, context-based, example-based (QBE), spatial-based, or constraint-based queries are some of the types. Regardless of the type, Boolean and ranked queries are the fundamental forms of queries and should be supported. For the data model which are able to support mixed queries such as our inverted model, it is not unusual to pose an integrated query with high-level concept and low-level feature matching on the basis of exact or similarity retrieval. Furthermore, since our inverted model can handle multiple entities within an image, it allows us to submit intricate queries to the data model. For example,

((brown CAT **or** red APPLE **or** *S*) **and** TABLE)

where *S* is the signature of a mouse. This query is looking for a cat, an apple or a mouse together with a table in an image. For ranked retrieval, a picture with a brown leopard standing on the table, for example, can be a potential candidate for this query.

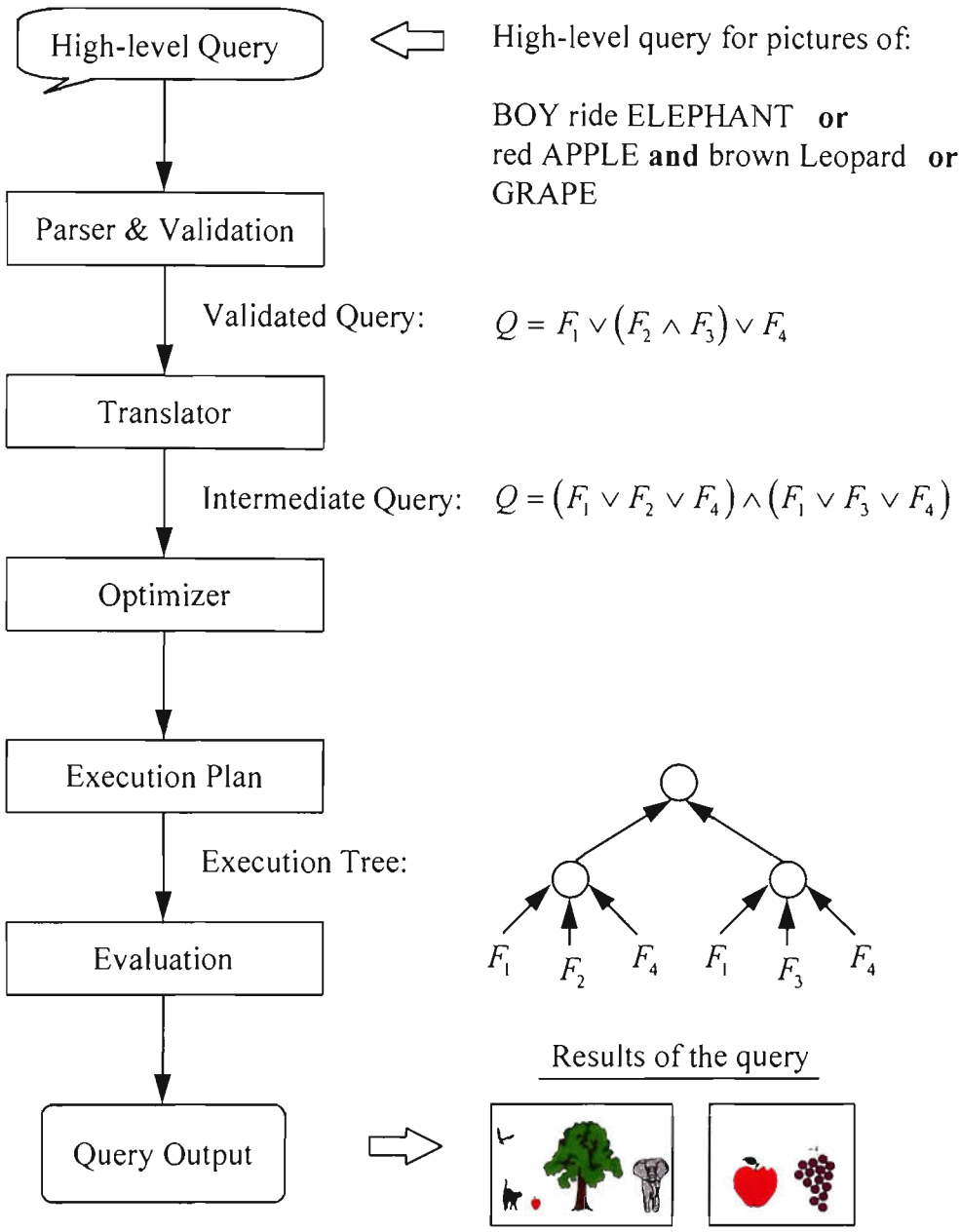


Figure 6.4 Processing Steps for our Inverted Model with a Simplified Illustration

Similar to conventional query processing, the steps involved in processing an image query include: (1) parsing and validating an query, (2) translating and transforming into an intermediate query, (3) optimizing the intermediate query, (4) formulating the execution plan, and lastly (4) performing the query. Figure 6.4 shows

the processing steps for our inverted model with an illustrative example. While it is not much of a problem to pose an alphanumeric query, it is harder to formulate query involving visual contents. Undoubtedly, an integrated user interface is needed for the query processing of image data. This is a research area in its own right and we do not pursue this further here.

The rest of this Section is organized as follows. The query processing steps will be thoroughly explained in Section 6.5.1. These include how to compose Boolean and ranked queries, to formulate intermediate queries for the execution plan, and to construct an execution tree. In Section 6.5.2, the selection criteria for ranked queries are specified. This is one of the possible heuristics which satisfies our expectation. A query example is shown in Section 6.5.3 to reinforce the concept. In Section 6.5.4, we outline the constraint-based queries using the spatial relationships of the image coordinates as an interesting extension to our inverted image model.

6.5.1 Query Processing

To begin our illustration for each processing step, we need to be more formal in defining our query. Let F_i be either an TFM fact T_i, or a signature S_i, in a general expression of a query after the parsing and validating processes and presented in the form of an Boolean equation, $Q(F_1, F_2, \ldots, F_n)$. For example, $Q = ((T_1 \wedge S_2) \vee (T_1 \wedge T_3) \vee S_4)$ for $n = 4$. Allowable Boolean operators include **and** ($\wedge$), **or** ($\vee$), **not** ($\neg$) together with parentheses. It is noted that Q is specified over an image. If **and** operator is used, both must be true for an image. An example is "brown

CAT **and** red APPLE" where both of these entities must exist in an image for this expression to be true. If we want any brown CAT or red APPLE in any image, the **or** operator is used instead. The **not** operator is rather subtle and not often used. An example is "**not** brown CAT". This means any image containing brown CAT is not selected.

The general expression is required to undergo translation and transformation into an intermediate query. It is not possible to evaluate the query with arbitrary levels of parentheses and terms. It must translate into an acceptable form so that the execution plan can be formulated. There are two forms we can use. The first one is the *product-of-sums*. It is also referred to as a *conjunction of disjunctions* or a *maxterm expression*. Any general Boolean equation can be translated into the following form,

$$Q(F_1, F_2, \ldots, F_n) = (F_w \vee \ \ldots \ \vee F_x) \wedge \ \ldots \ \wedge (F_y \vee \ \ldots \ \vee F_z) \qquad (6.1)$$

where $w < x$ and $y < z$ such that $w, x, y, z \in \{1 \ldots n\}$. The advantage in using this form is that it helps the execution plan to do lazy evaluation for Boolean queries. The second form is the *sum-of-products*. It is also referred to as a *disjunction of conjunctions* or a *minterm expression*. It is in the following form,

$$Q(F_1, F_2, \ldots, F_n) = (F_w \wedge \ \ldots \ \wedge F_x) \vee \ \ldots \ \vee (F_y \wedge \ \ldots \ \wedge F_z) \qquad (6.2)$$

where $w < x$ and $y < z$ such that $w, x, y, z \in \{1 \ldots n\}$. The advantage in using this form is that parallel execution of the image query is possible. These two forms are acceptable intermediate queries, but we would take the first one to standardize our execution. The reasons to choose this form are: 1) the logical integrity is maintained for the disjunctive terms when we relax the disjunctive terms to include less precise

target images, 2) this is a commonly used form for text retrieval and, hence, we may be able to borrow the research results from them, and 3) we do not expect a long list of disjunctive expressions for image queries. The obvious disadvantage is that each disjunctive expression can produce a large number of intermediate candidates. Unnecessary intersections may result.

It is worth noting that both Boolean and ranked queries are performed the same operations up to this point. The next step is optimization. Since the time required to evaluate each term is different, it is wise to do terms that are most efficient to generate results. For example, we would do terms that contains only TFM facts first because it is much easier to perform than searching signatures. Also, the real performance issue is not on searching TFM facts. Even with a large image collection, the total number of TFM facts is expected to be manageable. If an additional structure such as a B^+-tree is used, the searching process can be very efficient. Furthermore, given the memory capacity of modern computers, we can comfortably fit thousands of TFM facts with an average of modified facts or binary facts in memory for searching. Since each entry in the inverted file is unlikely containing a long sequence of pointers to the areas, the union and intersection operations are not too excessive. Nevertheless, choosing a disjunction with the least maxterms to evaluate first is a better choice.

Once the order of execution for each disjunction is set, the execution plan is formulated. An execution tree is used to propagate the intermediate results. Figure 6.5 illustrates an execution tree for the product-of-sums configuration. The leaf nodes of the tree consist of each disjunctive term in the query expression. An example of performing intersection and union on image sets is also shown in Figure 6.5. Our

algorithm to obtain the final set of target images is similar to harvest a crop. The crop is going over a number of times. We pick the best at each run until we have enough. At each run, we relax our selection criteria so that the weaker crop can be picked.

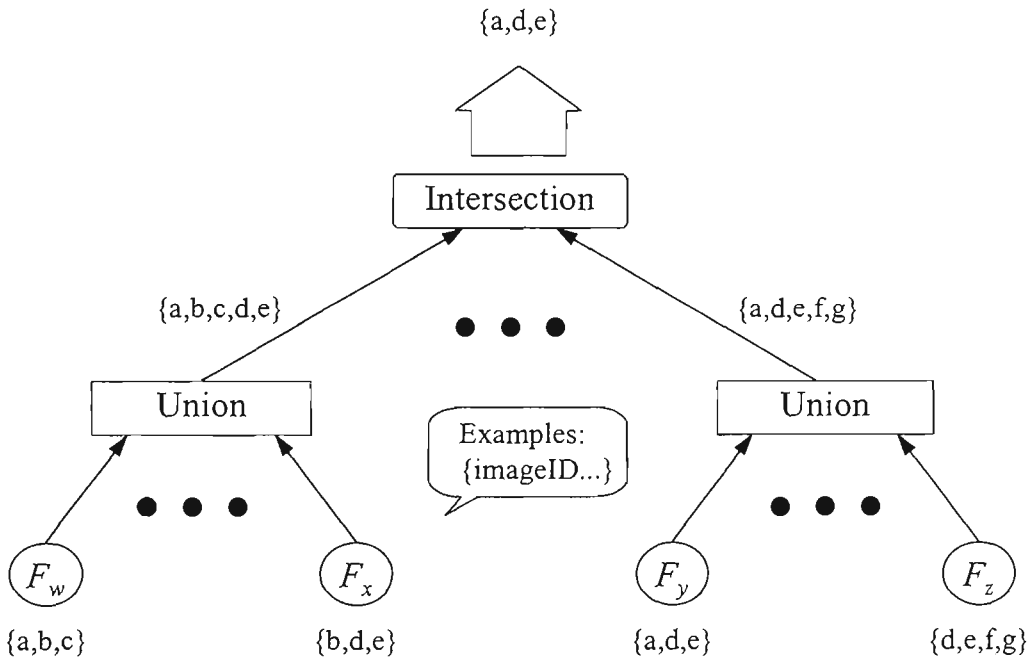


Figure 6.5 Execution Tree (product-of-sums) for Target Images with a Selection Criteria

There are a number of possible heuristics to determine the selection criteria for each pass of the tree. The heuristics described in the next section does it without resorted to weighting.

6.5.2 Selection Criteria for Ranked Queries

The selection criteria should be natural enough so that the results are consistent to meet our expectation. In other words, the results from the Boolean query

must surface first. These images are ranked at the top of anything else. We then relax the criteria step-by-step to accommodate less precise answers. Figure 6.6 depicts the heuristics to meet our expectation. Initially, the set of the target images is empty. For each step, a set of possible candidates is selected based on the logic of the execution tree. These intermediate candidates are appended to the end of the target set if they are not already in the set. Implicitly, ranked candidates are produced with the most precise images at the top. We can terminate the selection process once the number of target images is reached. Also, if only a precise answer is required, we can stop at the first step.

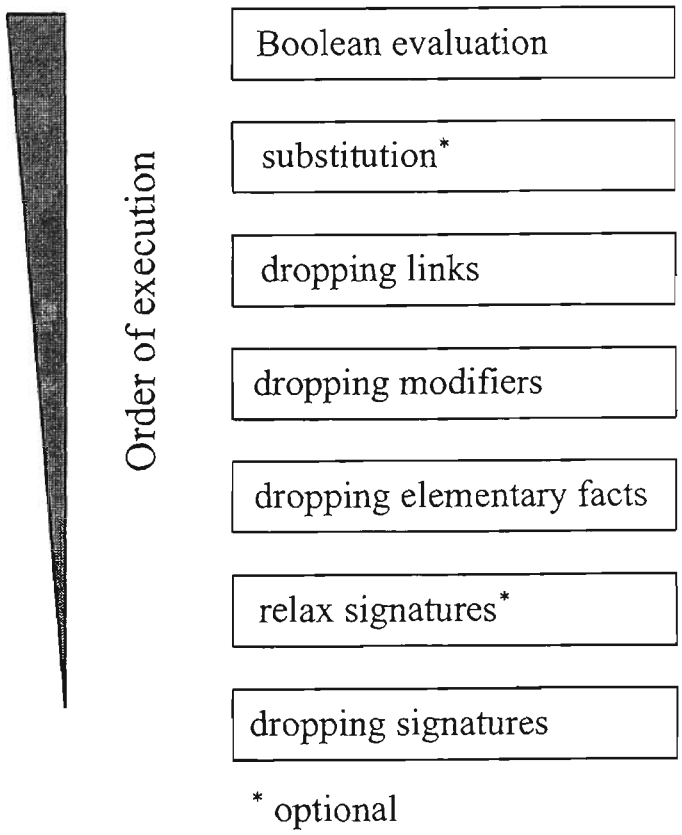


Figure 6.6 Heuristics for Selecting Target Images

The criteria are briefly explained in the following,

- *Boolean evaluation:* This step provides the precise solution with a given query expression. Ternary facts are matched as they are shown in each term of the expression. Exact signatures are also required if they are included in the query.
- *Substitution:* The purpose of this step is to replace the facts in the query expression using additional tools such as thesauri or knowledge-based tools. This will increase the chances if the exact matching is failed.
- *Dropping links:* For any binary facts or ternary facts, the links are removed. These will give rise to two individual elementary or modified facts. These two facts are considered under the disjunctive relationship.
- *Dropping modifiers:* To relax further, the modifiers for all modified facts are deleted and become the elementary facts. All various forms of facts containing the newly established elementary facts will be selected. This will further increase the chances of matching.
- *Dropping elementary facts:* All of the elementary facts are removed. Signatures are remained if they are involved in the query expression.
- *Relaxation of signatures:* Since we use signatures for feature retrieval, it is unlikely that a perfect zero score occurs in the ranking of signatures unless an exact image is given. Hence, we can relax the consideration by choosing the closest matches. Depending on the reliability of signatures, this step can be employed earlier in the selection criteria.

- *Dropping signatures:* If we still do not have enough ranked images, we can selectively drop some of the signatures out of the query expression.

It is conceivable that we can vary the above heuristics to cater for different precision and recall rates. Also, additional criteria can be supplemented to further improve our needs and expectations.

6.5.3 A Query Example

To reinforce the query processing concept for our inverted paradigm, let look at an example using the illustration in Figure 6.3. Let assume we want to locate images with the following,

(BOY Ride ELEPHANT **or** (GRAPE **and** green APPLE) **or**  **or** HOUSE)

We have a binary fact, a modified fact, a few elementary facts and a QBE image intermixed in a logical expression. For argument sake, let assume that the signature matching for the QBE image is not a precise one. Also, we do not have thesauri or knowledge-based tools to expand the facts. As a quick glimpse for the Boolean evaluation, only the image with the house (i.e. ImageID: B) satisfies the query. For the Boolean and ranked query, let us look at each stage of the complete evaluation.

Parsing and validating an query:

The query is parsed and validated. Any QBE image is processed and turned into signatures. A good user interface is very useful in formulating and parsing the query. Let assume the query is parsed into the following,

F_1 = BOY Ride ELEPHANT (a binary fact)

F_2 = GRAPE (an elementary fact)

F_3 = green APPLE (a modified fact)

F_4 =  (a signature of )

F_5 = HOUSE (an elementary fact)

The parsed query becomes,

$$Q(F_1, F_2, F_3, F_4, F_5) = (F_1 \vee (F_2 \wedge F_3) \vee F_4 \vee F_5) \tag{6.3}$$

Transforming into an intermediate query:

The query is then transformed into an intermediate query so that it can be easily processed. The expression in the product-of-sums form is adopted. Theorems of Boolean algebra are needed. The intermediate query becomes,

$$Q(F_1, F_2, F_3, F_4, F_5) = (F_1 \vee F_2 \vee F_4 \vee F_5) \wedge (F_1 \vee F_3 \vee F_4 \vee F_5) \tag{6.4}$$

Formulating an execution plan:

Given the query expression in Eq. 6.4, the execution plan can be formulated. An execution tree is constructed with the leaf nodes consisted of two disjunctive subexpressions. Figure 6.7 shows the execution tree for the query specified in Eq. 6.4.

Each pass of the execution tree will produce a set of incremental candidates to be inserted into the final target set of images, T_q . The target set is somewhat ordered with the most likely candidates at the top.

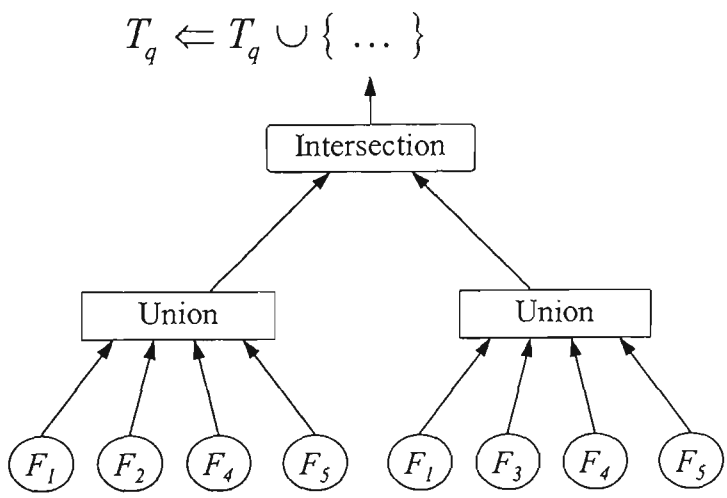


Figure 6.7 Execution Tree for Equation 6.4

Performing the query:

All the steps except the substitution outline in Figure 6.6 are performed to obtain the target set of images. Let assume $Q_1 = (F_1 \vee F_2 \vee F_4 \vee F_5)$ and $Q_2 = (F_1 \vee F_3 \vee F_4 \vee F_5)$ are the two disjunctive terms in Q. Also, $T_q = \varnothing$ initially.

The intermediate results are tabulated in Table 6.5. The order of the candidates, {B,A,C}, is a logical answer and within our expectation. Admittedly, if we have more images with longer inverted lists, the power of our inverted model will be more evident. Nevertheless, this example illustrates that Boolean and ranked queries are nicely integrated to provide a ranked sequence of images.

	Boolean evaluat-ion	Dropping links	Dropping modifiers	Dropping element-ary facts	Relax signatures	Dropping signatures
F_1	∅	{A} Note #1	{A}	N/A	N/A	N/A
F_2	{C}	{C}	{C}	N/A	N/A	N/A
F_3	∅	∅	{A,C} Note #2	N/A	N/A	N/A
F_4	∅	∅	∅	∅	{A,B} Note #3	∅ Note #4
F_5	{B}	{B}	{B}	N/A	N/A	N/A
Q_1	{B,C}	{A,B,C}	{A,B,C}	∅	{A,B}	∅
Q_2	{B}	{A,B}	{A,B,C}	∅	{A,B}	∅
Q	{B}	{A,B}	{A,B,C}	∅	{A,B}	∅
T_q	{B}	{B,A}	{B,A,C}	{B,A,C}	{B,A,C}	{B,A,C}

Note #1: The link, "Ride", is dropped. ELEPHANT is matched.

Note #2: The modifier, "green", is dropped. APPLE is matched on red APPLE.

Note #3: Assuming the relax signature for the tree is matched.

Note #4: No signature is left if the signature for the tree is dropped.

Table 6.5 Results for the Query Specified by Equation 6.3 Using Rules in Figure 6.6

6.5.4 Constraint-based Queries Using Spatial Relationships

With the coordinates specified for image areas, it is possible to deduce many spatial relationships among them as mentioned in Section 6.4.4. We can extend our queries to incorporate many of these relationships. The extensions can be in the formed of functions and integrated into Eq. 6.2. For example, let assume we have a function, `Surround(`F_i`)`, which is able to return the facts surrounding F_i in an image. The results can then be part of the query for further searching. Using our three sample images, `Surround(` HOUSE `)` will return the elementary fact, TREE. If we pose a Boolean query to the sample images such as `(` ELEPHANT **and** `Surround(` HOUSE `))` will precisely return the image set {A}, via the connection of TREE.

Constraint-based queries using the information provided by the coordinates are not limited to the Boolean and ranked queries. They can be used in advanced queries such as gathering statistical information of the image collections, complex proximity search and many others. Certainly, supporting data structures and additional programming efforts are needed to implement these queries.

6.6 Compression Consideration

The main bulk of the data in the inverted text file is the sequence of word pointers. Hence, compression is used to reduce the long list of pointers with the fact that decompression is needed every time a query hit the entry. On the surface, this may seem to be a disadvantage. With the current CPU processing capability, the cost

in decompression can easily outweigh the I/O cost in retrieving the uncompressed data, especially for distributed data. This is even more true for image data. Compression is always needed to reduce the high demand of storage for image data. For our IIIC model, the main area of compression is not on the image coordinates. This is because the sequence of coordinates for each inverted list is expected to be not very long. We do not see the advantage to compress the coordinates. The main areas of compression in the IIIC model would be on the picture keys and the composite bitplane signatures. The nature of these two areas require different consideration.

As mentioned before, the picture keys are used to capture the salient features of the inverted lists. Lossy compression is acceptable. They are mainly used as the thumbnail pictures for the entry. Initially, we integrate the compression with the Haar decomposition for the signature generation. Performing one decomposition for dual purposes seems to be a good idea to start with. However, the compatibility of our "home-made" format to the outside world is a problem. We finally store the picture keys in JPEG so that we can integrate our experiments with third party software, especially user interface software.

For the composite bitplane signatures, the advantages of using compression for both storage and retrieval consideration are apparent. Firstly, it is not unusual to have thousands of signatures in the databases. The storage requirement is excessive and wasteful if the bitplanes are not stored in compressed forms. For example, a n x m bitplane will take at least $\lceil nm/8 \rceil$ bytes to store, which is over 2K bytes for a 128 x 128 bitplane. If the bitplane is stored using lossless compression, the size is reduced to approximately 150 bytes as verified by our experiment. This is because the bitplane is

a sparse matrix with many zeros in it. Compression is ideal for this kind of data. Secondly, if the bitplane is in compressed form, it is possible to store them in memory if the database size is moderate. For example, if the image collection is of the order of a few thousand, keeping a few megabytes of the compressed bitplanes in memory is not unreasonable. Decompression on demand can be very efficient if everything is already in memory. Lossless compression such as LZW described in Chapter 3 is ideal for the compression of bitplane signatures.

We must emphasize that it is essential to use compression in our IIIC model. Without compression, it is awkward to handle some components of the IIIC model. This will hinder the searching and retrieval process.

6.7 Summary

In this chapter, the concept of Inverted Image Indexing and Compression is presented. The inverted model for text retrieval is well known but the paradigm has never been applied to image data. The constituents of an image is distinguished into Area, Pixel and Area Coordinates, which correspond to Word, Character and Word Position respectively. This provides us with an analogy in formulating our inverted image model.

Many of the existing content-based image retrieval techniques are concerned with the low-level visual features. It is our view that low-level syntactic indexing must be accompanied by high-level semantic indexing to provide a complete indexing structure. Users can search the inverted image file through the semantic indices or the

syntactic feature indices. The semantically rich description of Ternary Fact Model is used for our high-level indexing. For the low-level indexing, Composite Bitplane Signatures are used for content-based similarity retrieval.

Another attractive property of our inverted model, unlike many of the existing indexing methods, is the ability of indexing different visual contents within an image. Each image can have multiple areas which are perceived to be meaningful visual contents. They must be indexed separately in the framework that relationships among them can be exploited for meaningful retrieval. This important observation is also echoed by recent developments in image retrieval.

A good data model must be accompanied by a good query model. In general, we must support precise Boolean queries as well as the ranked queries for similarity retrieval. The query model provided in this chapter not only can be used for our inverted model, but can also be applied to other content-based image retrieval methods. The query model is characterized by harvesting the potential target images from the precise solution to the more relaxed solutions under the logical rule and specification of the query expression. Advanced queries using the spatial relationships of the image areas are possible in our inverted model.

Compression plays an important role for our inverted model. Without compression, it is awkward to handle some of the components. Due to the fact that our bitplane signatures are very suitable for compression, the advantage of using compression is evident. This also becomes an important property of our inverted model.

Chapter 7

Experimental Results

7.1 Introduction

In this chapter, we devote our attention to the experimental aspects of our work. In Section 7.2, we outline the environment of our software developments throughout this research project. The nature of our test images is also described. Most of the experiments are using over a thousand of test images. In Section 7.3, we present all the experiments collectively. These include the following:

1. *The nature of wavelet compression*. Since we use wavelet decomposition for our bitplane signatures, we would like to verify that we can indeed use a small portion of coefficients to generate a good approximation of the original image. Hence, this experiment provides a rough estimation of the compressibility. Although this is not a direct verification or evidence to

prove our signature scheme, it just illustrates the correctness of this direction.

2. *Characteristics of wavelet subbands.* The nature of wavelet coefficients is characterized by the subbands. Our experiment reveals the characteristics of the directionally sensitive features of the wavelet coefficients. Since we use the signs of the m largest magnitude coefficients for the bitplane signature generation, our experiment provides insight into how the coefficients are picked.

3. *Distribution of the signs of wavelet coefficients.* Ideally, we would like the image hashing to be as uniform and random as possible with respect to the bitplane signatures. This is not achievable in practice for our wavelet signatures as the distribution of the bit patterns are dictated by the nature of the subbands. Hence, our experiment also reveals the limitation of using wavelet signatures for signature generation.

4. *Performing Bitplane Signature generation.* We demonstrate the generation of bitplane signatures for our test images. The process of generation is followed by the procedure set out in Chapter 5. Examples of bitplane signatures are shown.

5. *Image retrieval through Composite Bitplane Signatures.* Our experiment illustrates the effectiveness of our image retrieval scheme through Composite Bitplane Signatures. The ability to retrieve similar images

using our bitplane signatures embedded in Composite Bitplane Signatures is evaluated in this experiment.

6. *Analysis of superimposed bitplane signatures.* In this study, we superimpose bitplane signatures to form a composite bitplane. The purposes of these experiments are to analyze the behavior of composite bitplanes. We investigate how the composite bitplanes are populated. This will give us an estimate to the maximum number of images to be superimposed. We investigate the behavior using different image arrangements and settings.

7. *Inverted Image Retrieval.* We perform image indexing and retrieval using our inverted model. Inverted lists are generated using high-level semantic TFM specifications and low-level bitplane signatures for all the test images. Examples of searching through TFM and bitplane signatures are provided.

In Section 7.4, we conclude this chapter with a summary where the main observations and experimental results are discussed.

7.2 Environment

7.2.1 Platforms

Most of the software are developed in UNIX platforms. The hardwares include a Pentium 100 Mhz PC and a multiprocessing SPARC Station. The Pentium is equipped with 16M memory, 2 Gbytes hard disk and Diamond Multimedia Card etc. It is mainly used for the implementation of the testing software. The operating system is Linux. Third party software used regularly in our development and testing include `XV`, `gzip`, `ImageMagick`, `pbmplus` and `cjpeg`. The main programming language is GNU C++.

7.2.2 Test Images

The collection of our test images is from MasterClips 35,000™. It consists of 1001 color images. The images are divided in 28 groups ranging from automobiles to wild-life animals. Each group contains approximately 35 images of various sizes. Most of the images are in 384 x 288 pixels. Table 7.1 summarizes the groups and their contents. The first and last images of each group can be found in Appendix A. Also, all 35 images of the first group, "auto", are shown in Appendix B. The illustrations of these sample images are needed for verification purpose for some of the experiments conducted in this chapter.

Groups	No of Images	Descriptions
auto	35	automobiles, engines, parts etc.
birds	36	birds such as parrots, pelicans, swans, gooses etc.
bkgrnds	36	background patterns such as mud, sands, roofs etc.
boats	35	boats, ships, tankers, yachts, sailboats etc.
buildngs	35	buildings, offices, hospital, churches, colleges etc.
children	35	babies, children at various activities etc.
citytwn	35	cities, streets, towns, bays, freeways etc.
coasts	36	coasts, cliffs, beaches, surf, shores etc.
dom_an	36	domestic animals such as horses, cats, dogs etc.
dsrtcyn	32	canyons, deserts, sandstone peaks, gorges etc.
fields	30	barns, pastures, crops, fields etc.
food	38	food items such as fruits, pies, seafood, nuts etc.
lakerivr	36	lakes, rivers, streams, pools, banks etc.
lifestyl	35	life styles, sports, concerts, leisure etc.
mountain	35	sceneries related to mountains etc.
objects	48	assorted objects such as computers, sea shells etc.
people	38	athletes, workers, guards, portraits of women etc.
planes	36	aircrafts such as planes, fighters, jets, biplanes etc.
plants	38	flowers, ferns, leaves, vines, thistles etc.
skycloud	36	sceneries related to clouds etc.
space	36	spaces launches & walks, astronauts, planets etc.
speclocsn	28	special occasions such as Christmas, fireworks etc.
sunset	36	sceneries related to sunsets etc.
texture	35	textures for walls, bricks, rocks, water drops etc.
treeleav	36	forests, pines, trees, orchards, branches etc.
underwtr	35	underwater creatures such as coral, fishes, crab etc.
watrfall	36	water falls, cascades, falls and spray etc.
wildanim	36	wild-life animals such as tigers, lions, bears etc.

Table 7.1 1001 Test Images From MasterClips 35,000™

7.3 Experiments

7.3.1 Wavelet Compression

The signature scheme proposed for our Composite Bitplane Signature is built using the wavelet decomposition. Before we investigate the effectiveness of our signature scheme for image retrieval, we would like to verify that a good image approximation can be achieved by just a small portion of coefficients. In other words, we would study the compressibility of such method. The mathematical derivation of the Haar wavelet decomposition and composition for the purpose of lossy compression is outlined in Section 3.2.2. It has been proven that we can retain the largest magnitudes of coefficients to minimize the errors introduced by the truncation.

We arbitrarily pick an image, "auto001", from our collection to test the compression. The image is scaled to 128 x 128 prior to the wavelet decomposition using Haar basis functions. It is the nature of wavelet decomposition shown in Figure 3.17 and Figure 3.18 that each dimension of the images should be in 2^j pixels for the subbands to work. Therefore, the images are scaled to a manageable size of 2^7 x 2^7 (i.e. 128 x 128). We retain the largest coefficients by percentages for each color channel and restore the image based on the truncated coefficients. The RMS errors and PSNR for the compression scheme in RGB-colors are plotted in Figure 7.1 and Figure 7.2 respectively. For completeness, we provide the equations in Eqs. 7.1 and 7.2 for the two error measures - Root Mean-Squared Error (RMS Error) and Peak Signal-to-Noise Ratio (PSNR). These definitions assume 8 bits per color channel for

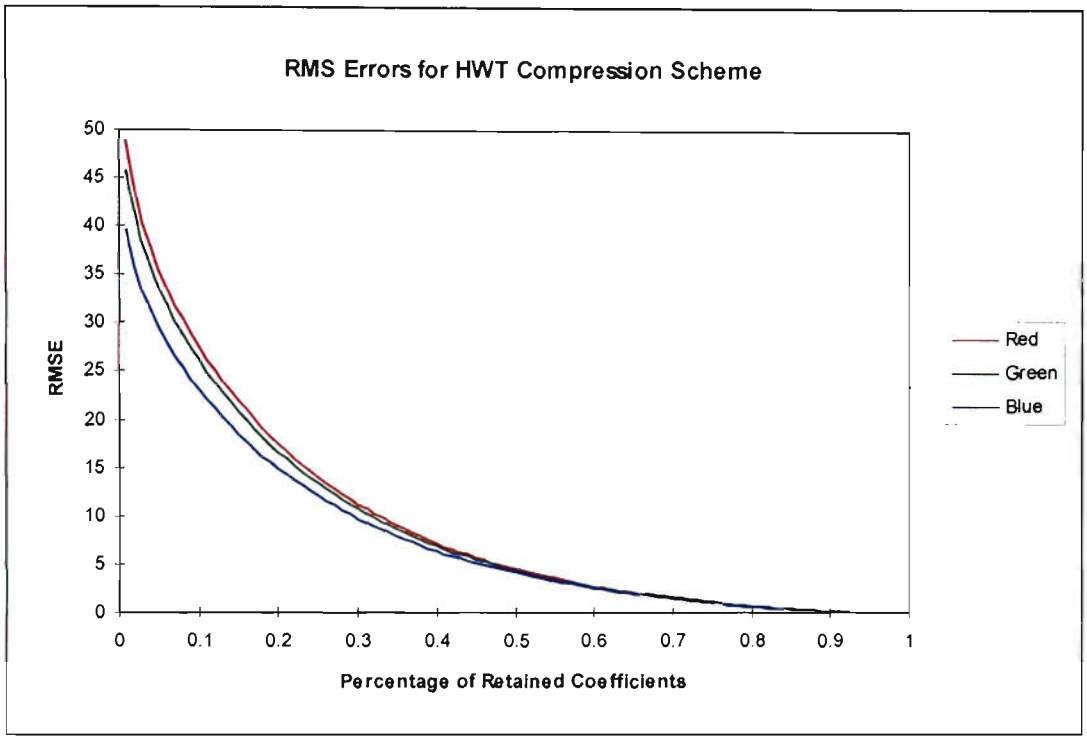


Figure 7.1 RMS Errors for "auto001" in RGB-colors

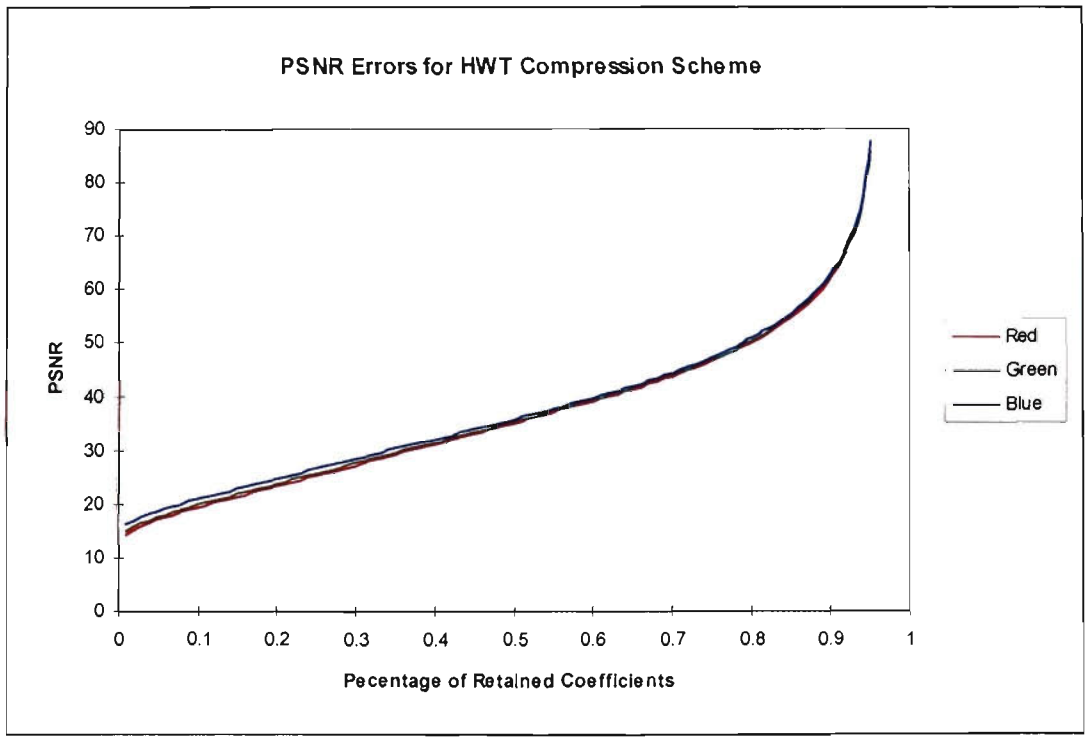


Figure 7.2 PSNR for "auto001" in RGB-colors

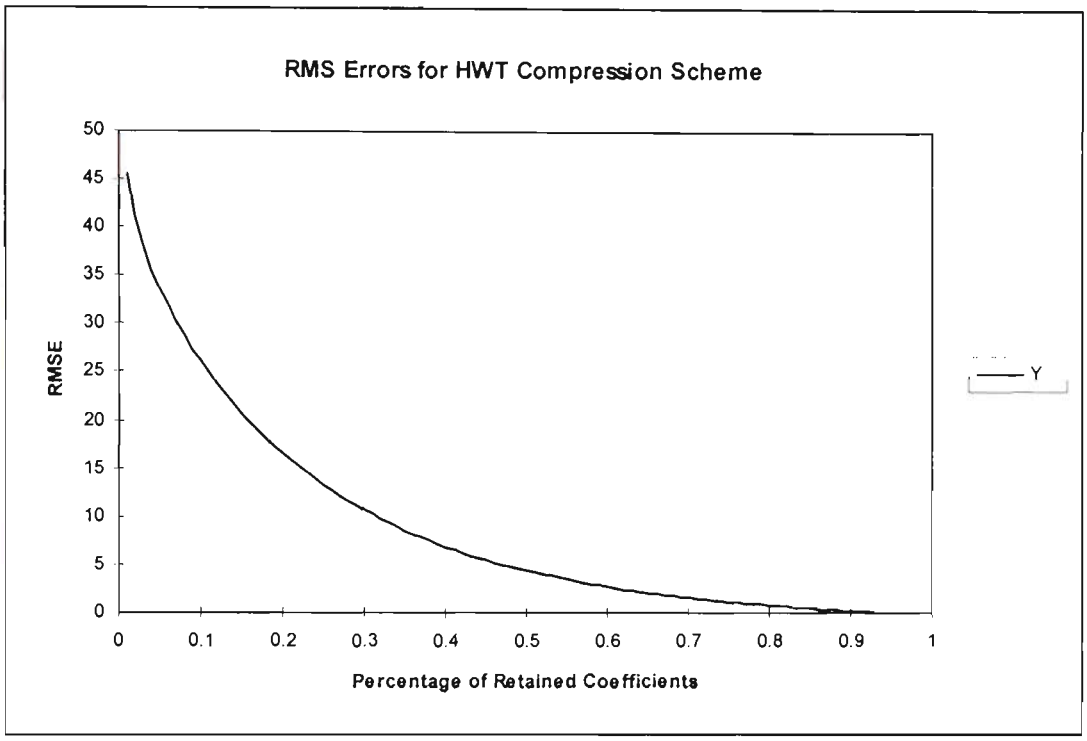


Figure 7.3 RMS Errors for "auto001" in Y Channel

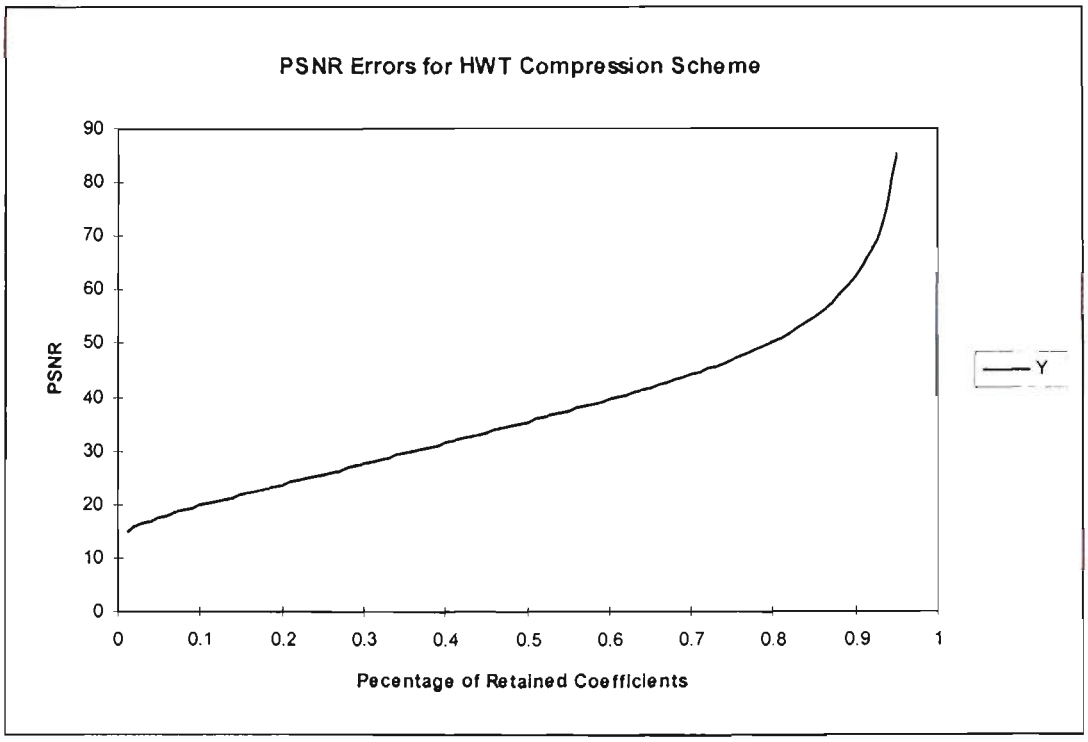


Figure 7.4 PSNR for "auto001" in Y Channel

each pixel. Hence, we use 255 for the numerator (i.e. $\max|F(x,y)|$) in Eq. 7.2 for the RGB colors and only the gray scale images (i.e. the Y channel) in YIQ colors are plotted. The original image and the reconstructed image, F and $\tilde{F}$, are N x M in width and height.

$$RMSE = \sqrt{\frac{1}{NM}\sum_{x=1}^{N}\sum_{y=1}^{M}\left[F(x,y) - \tilde{F}(x,y)\right]^2} \tag{7.1}$$

$$PSNR = 20 \log_{10} \frac{\max|F(x,y)|}{RMSE} \quad \text{(in decibels; dB)} \tag{7.2}$$

The conversion from RGB to YIQ is defined as:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{7.3}$$

The same experiment is repeated for YIQ-colors. The RMS errors and PSNR for Y- channel are plotted in Figure 7.3 and Figure 7.4 respectively. As expected, the RMS and PSNR plots confirmed that the image deteriorates gradually initially and we do not see any abrupt change in errors for both color schemes. Sometime, it is meaningless to quantitatively measure the errors whereas the visual deterioration is a different matter. Therefore, we inspect the deterioration visually. Figure 7.5 illustrates the reconstructed images using a given percentage of the total coefficients. We are confident to say that 20 percent of the largest magnitudes of coefficients can reproduce the original image with a reasonable degree of clarity.

Figure 7.5 Compressed "auto001" at Various Percentages of HWT

7.3.2 Characteristics of Wavelet Decomposition

The nature of wavelet coefficients is characterized by the subbands. From Figure 3.17, the 2D forward wavelet decomposition is accomplished by two separate 1D wavelet decomposition. For each level of decomposition, an average (low-pass) sub-image together with three directionally sensitive (high-pass) sub-images are generated. The process is repeated on the average sub-image to produce a higher level of decomposition. Figure 7.6 shows the process of decomposition. That is, at each level, an image f is filtered into four subbands; f_{LL}, f_{LH}, f_{HL}, and f_{HH}. As we can verify from the experiments later, f_{LH} emphasizes the horizontal image features, f_{HL} the vertical features, f_{HH} the diagonal features and f_{LL} is being the average image.

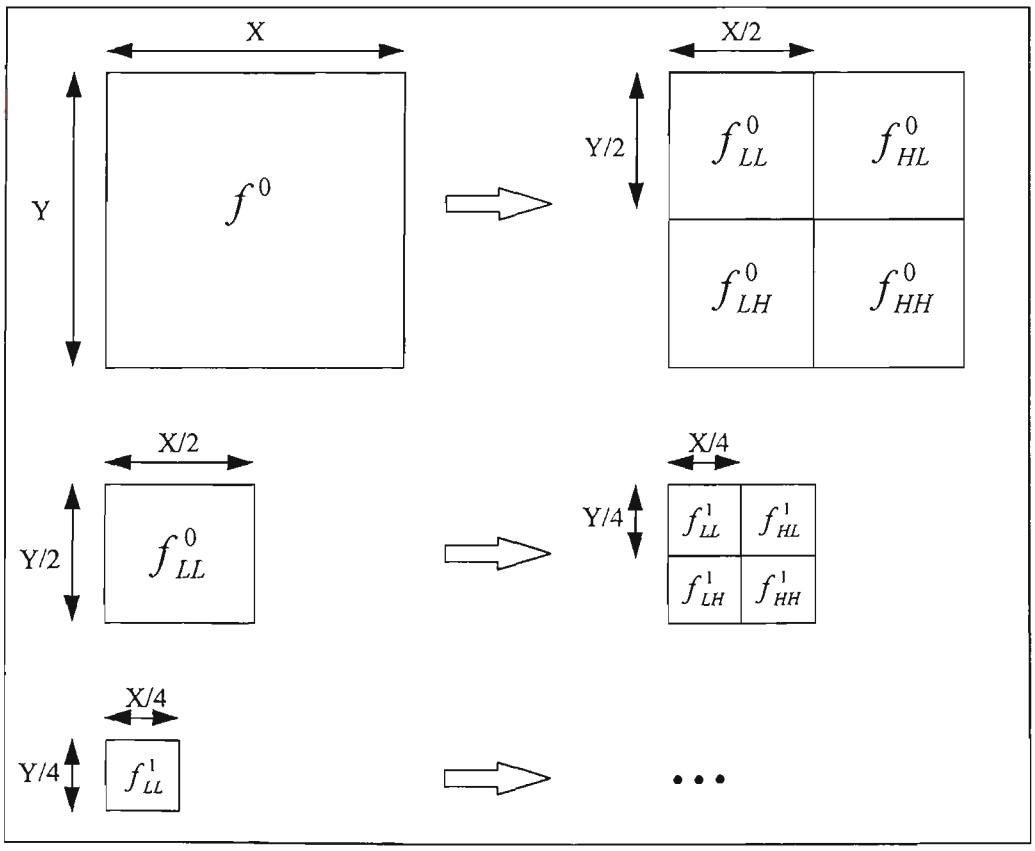


Figure 7.6 Process of Wavelet Decomposition

If Haar basis functions are used for the wavelet decomposition, the low-pass and high-pass sub-images represent the averaging and differencing coefficients respectively. To illustrate this, we use the non-standard decomposition without normalization (refer to Section 3.2.2 for details) to produce the different levels of decomposition. Two images, "auto001" and "lena", are used to demonstrate the decomposition. Figure 7.7 shows the whole process of decomposition for "auto001". The image is initially scaled to 256 x 256 and, hence, the decomposition has 8 levels. To amplify the high-pass image features, the absolute values of the coefficients for the sub-images fLH, fHL, and fHH are multiplied by a small factor and the sub-images are printed in reverse.

We also repeat the experiment with the popular image, "lena". Since "lena" has many horizontal, vertical and diagonal edges, it is ideal to show the directionally sensitive features of fLH, fHL, and fHH. Figure 7.8 shows a two-level wavelet decomposition of "lena". It is clear from Figure 7.8 that the vertical features (e.g. hair, mirror's edge) and the horizontal features (e.g. eye-brows, lips) are picked up by the subbands.

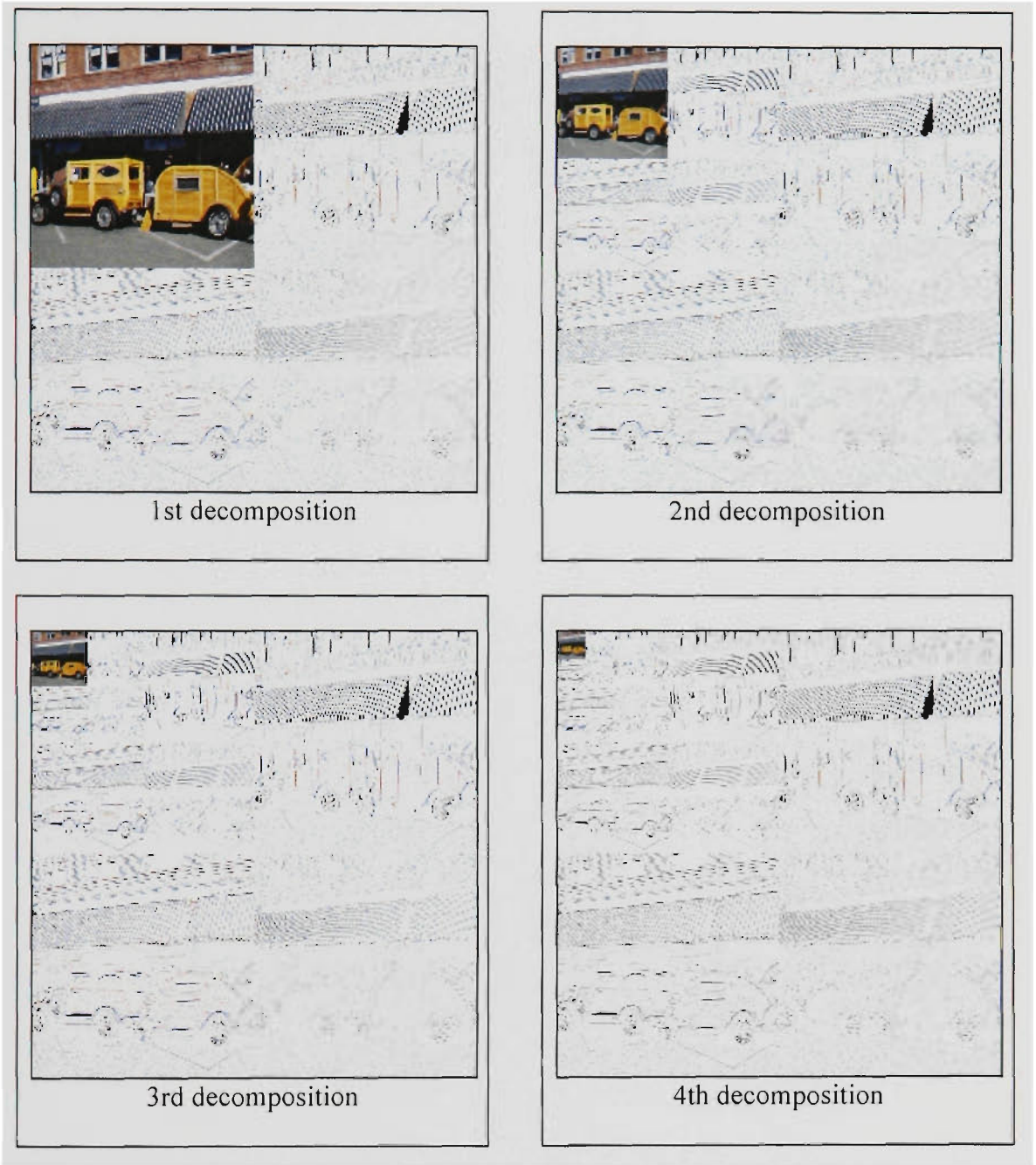


Figure 7.7 Continued

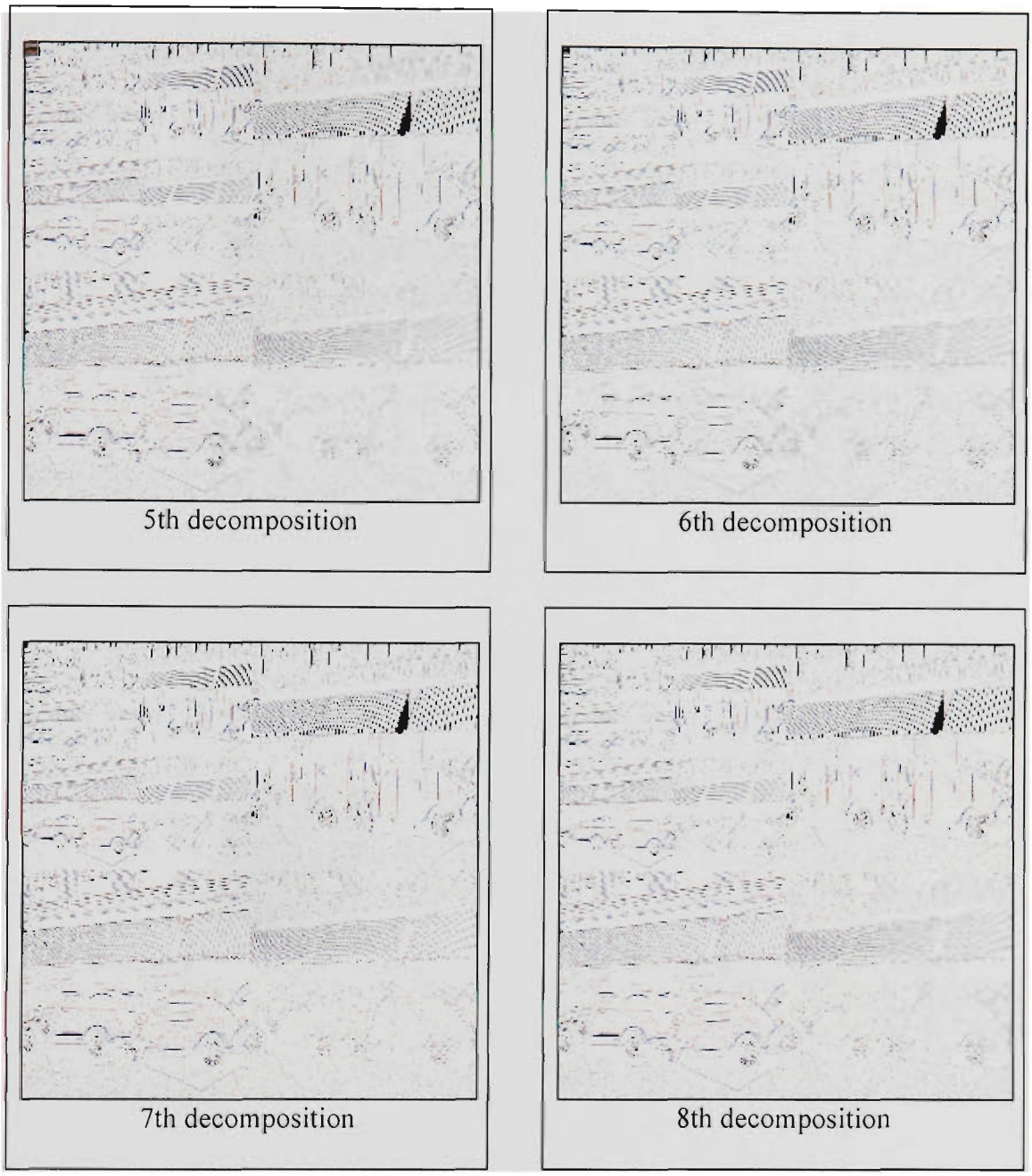
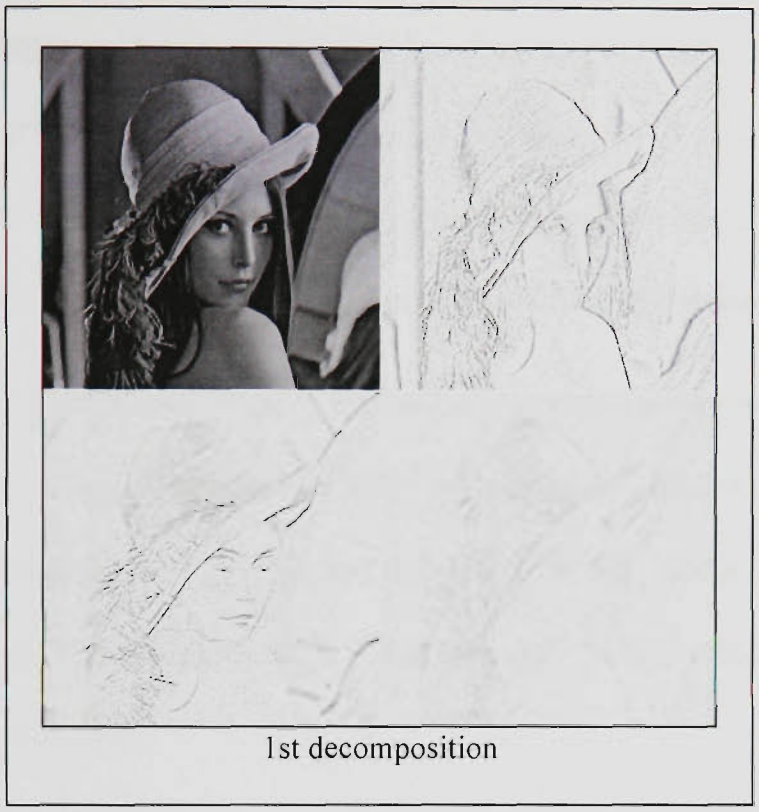
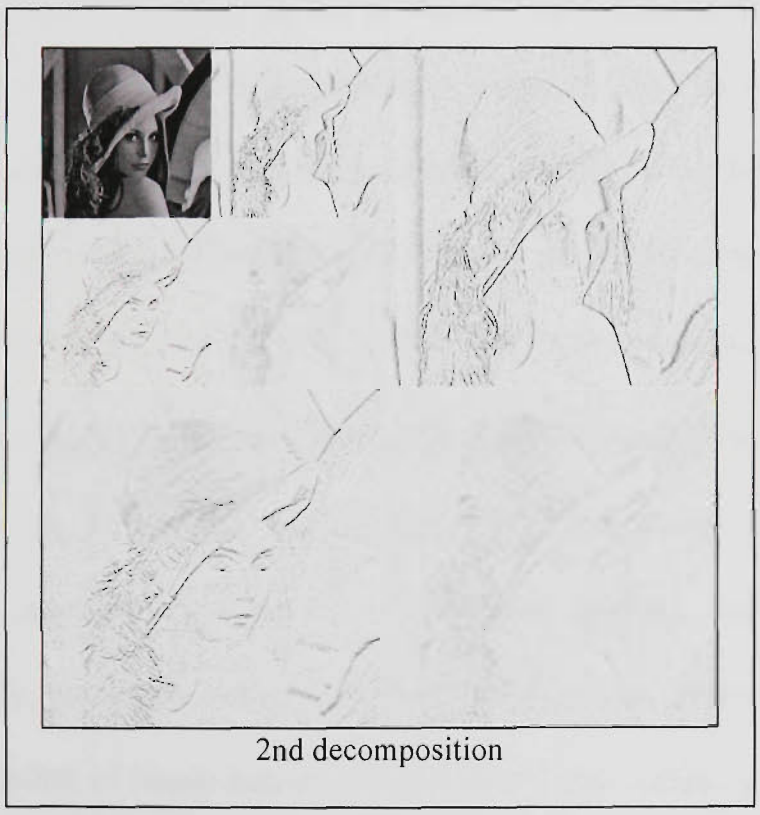


Figure 7.7 Wavelet Decomposition of "auto001"



1st decomposition



2nd decomposition

Figure 7.8 Wavelet Decomposition of Lena (Only the 1st and 2nd Decomposition Shown)

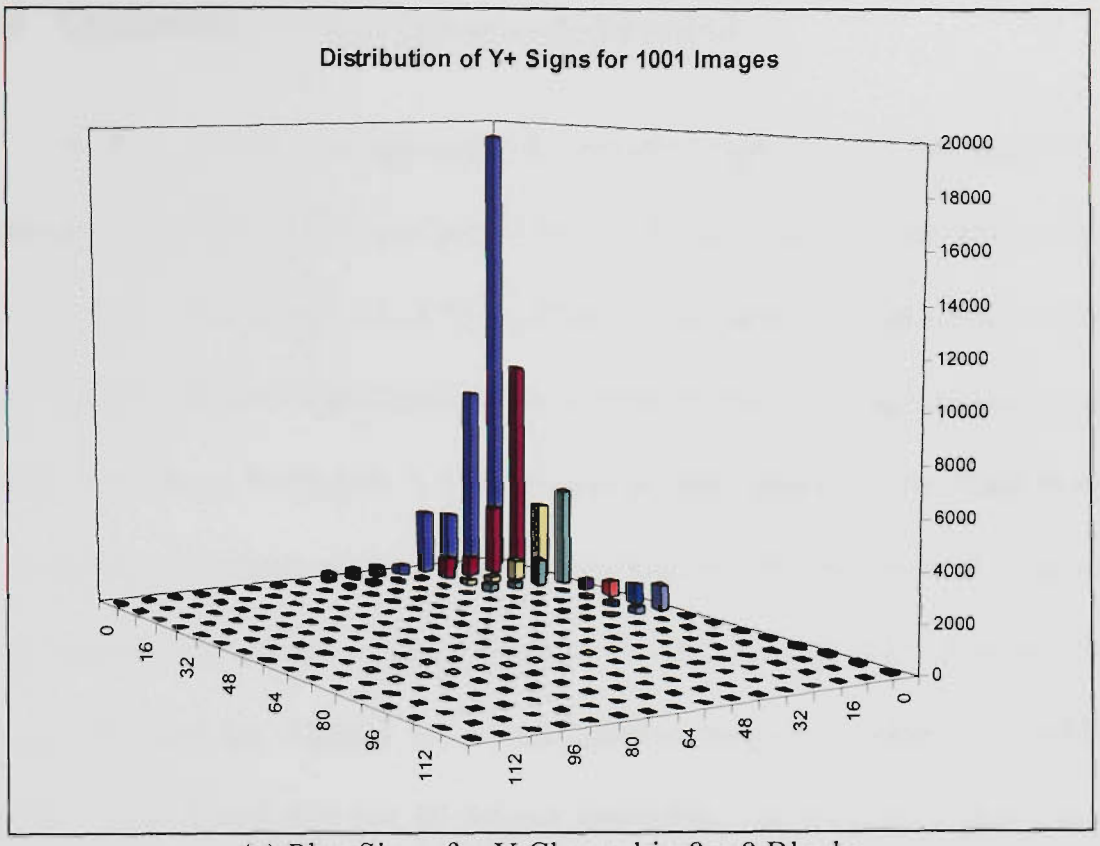
7.3.3 Distribution of Wavelet Coefficients

Our signature scheme uses the signs of the m largest magnitudes of wavelet coefficients to generate the image bitplanes. In this section, we will study the distribution of the largest wavelet coefficients using Haar basis functions. This will, in turn, dictate the distribution of the bit patterns for our bitplane signature scheme.

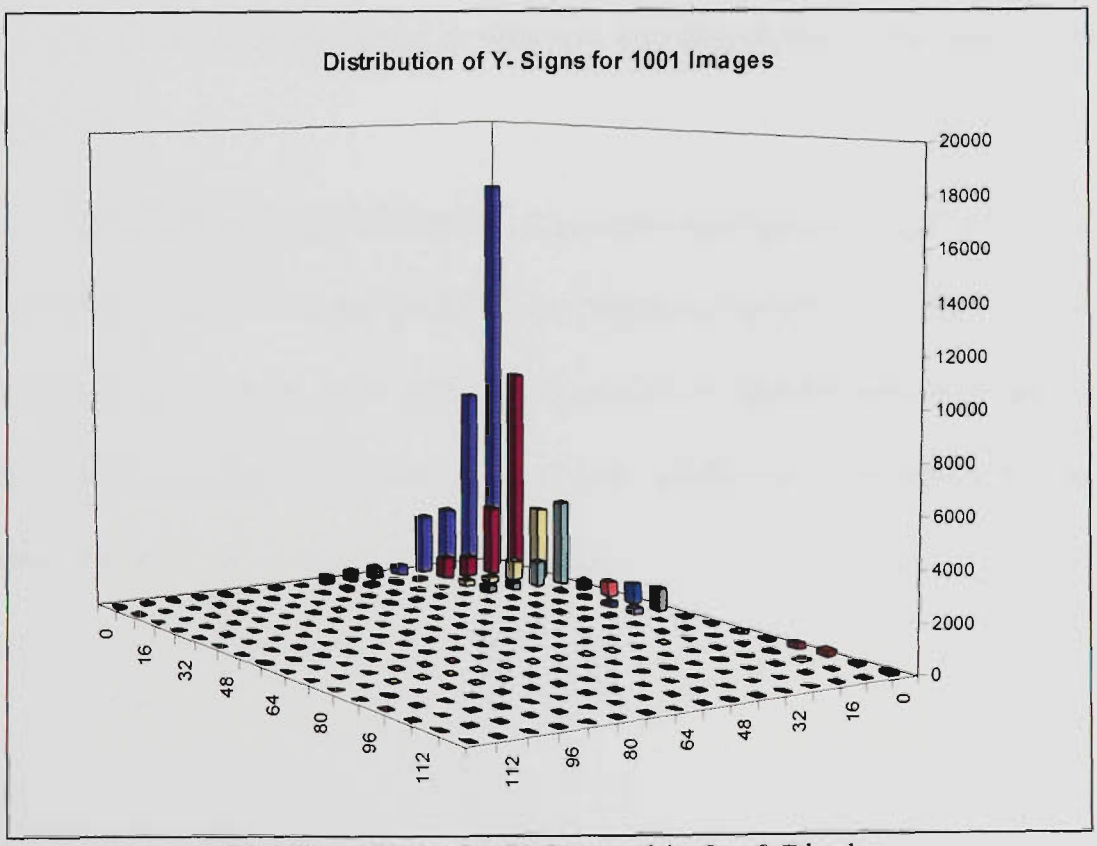
As observed from Section 7.3.2, we expect that the largest magnitudes of wavelet coefficients are concentrated around the scaling coefficient. This is because the scaling coefficient is really the average value of an image. For any given decomposition, the low-pass subband f_{LL} will give rise to higher absolute values than other high-pass subbands.

In this experiment, all images in the Masterclips™ collections are used. They are scaled to 128 x 128 and the color space is converted to YIQ. We then use m = 128 to generate the wavelet signatures of all the images. The plus and minus signs for each of the m coefficients are counted. Figure 7.9 shows the distribution of signs for the 1001 images. To make the graphs easier to read, we plot the counts in 8 x 8 blocks.

The experimental results confirmed our intuitive observation. We must assert that the distribution is not ideal. This characteristic will hinder the generation of unique bitplane signatures and limit our freedom to perform composite bitplane signatures. Ideally, we would like the distribution as uniform and random as possible so that our perception of image hashing is attainable. Nevertheless, this short-coming is acceptable if we trade this for the effectiveness of similarity retrieval. This nature is further explored later in this chapter.



(a) Plus Signs for Y Channel in 8 x 8 Blocks



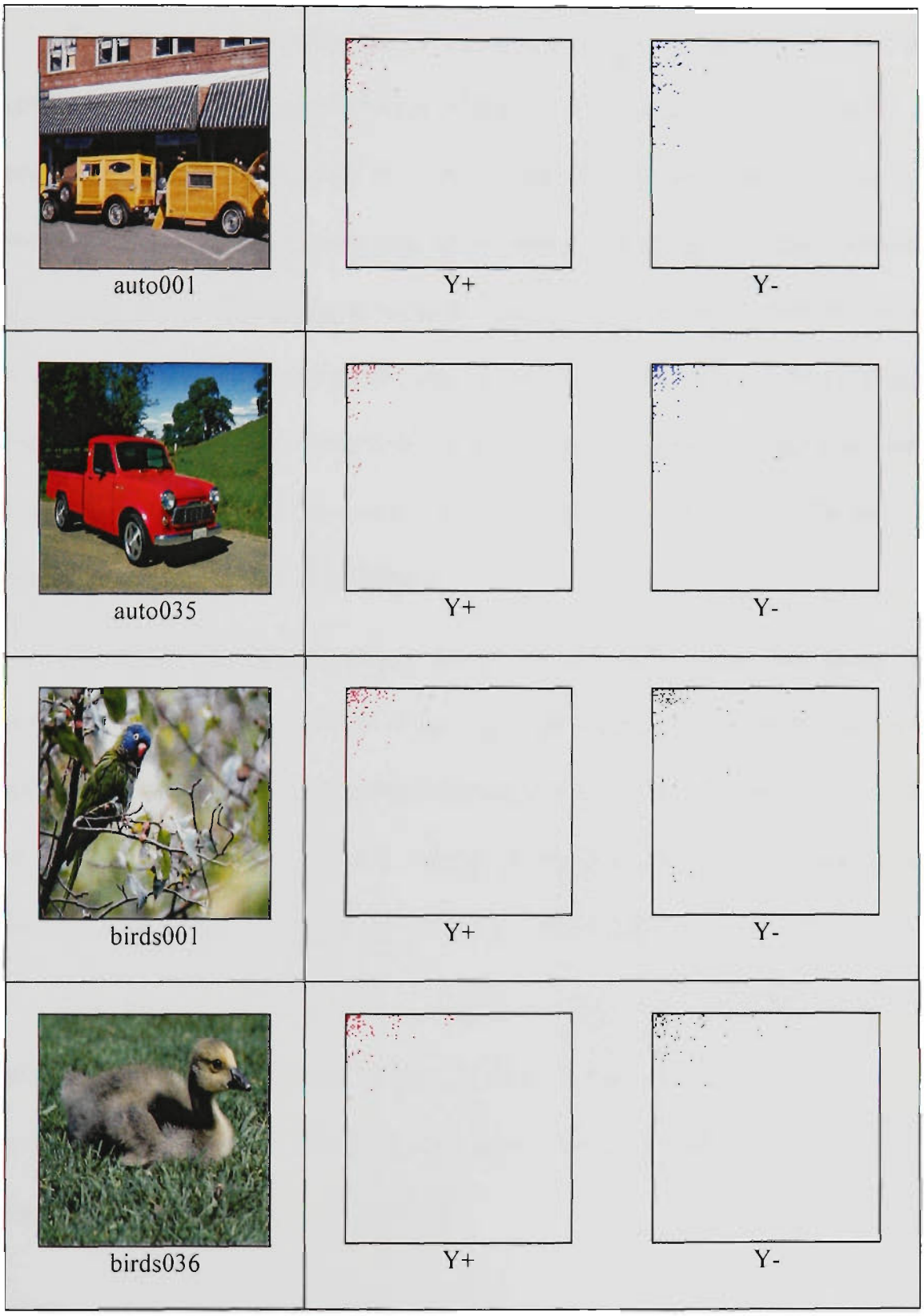
(b) Minus Signs for Y Channel in 8 x 8 Blocks

Figure 7.9 Distribution of Signs for Y Channel with m = 128 for 1001 Scaled Images

7.3.4 Generation of Bitplane Signatures

In this section, we generate the bitplane signatures to be used by the subsequent experiments. As mentioned before, all the images in our collection are scaled to 128 x 128. These allow the consistent generation of signatures for all the images in the collection. The images are converted to YIQ color spaces according to our scheme outlined in Chapter 5. The images are then subject to the Haar Wavelet decomposition. The signs of the m largest magnitude coefficients are used to generate the six bitplane signatures. Again, we use m = 128 for each color channel. These design parameters are initially set to the similar research findings in [JACO95, STOL96]. They found that the 60 largest magnitude coefficients in each channel worked best for painted queries, while 40-coefficients worked best for scanned queries. The effect of changing the m values is also the subject of the experiments in Section 7.3.6.

Some examples of the bitplane signatures selected from our collection are shown in Figure 7.10. They are the first four images in Appendix A. Only the plus and minus signatures of the Y color channel are shown. As demonstrated in Figure 7.10, the bits tend to be concentrated around the coarser subbands. Once again, the results are consistent with our findings in Section 7.3.3.



(a) Sample Pictures (b) Y Color Channels

Figure 7.10 Samples of Bitplane Signatures

7.3.5 Image Retrieval Using Composite Bitplane Signatures

To test the effectiveness of our composite bitplane signatures, the bitplane signatures of all 1001 images are used. Using the natural grouping of images, we generate the composite bitplanes for each group. This is equivalent to having a composite bitplane signature containing approximately 35 images. We then randomly pick an image in the collection and generate some distorted variances. Not only do we want to evaluate the robustness of our scheme, we would like to know the ranking of similar images. The image, "auto008", is picked and distorted by applying some filtering techniques such as *blur*, *sharp*, *oil paint*, *pixelize* and *spread*. Figure 7.11 shows the original image and its variances.

Intuitively, the original image, "auto008", is embedded in the first group. We expect that it should be ranked first if such an identical image is used for the query. Otherwise, our scheme for the composite bitplane signatures is incorrect or not useful at all. The variances of the original image are really to verify the power of the similarity retrieval using the signs of the wavelet coefficients.

The ranking of 28 groups is shown in Table 7.2. It illustrates that the distorted variances of the original image are able to select the first group as the top rank. Now, within the first group, the expressiveness of our signature scheme for similarity retrieval is our next investigation.

Figure 7.11 Test Images

Groups	A. original	B. blur	C. sharp	D. oil paint	E. pixelize	F. spread
auto	1	1	1	1	1	1
birds	6	5	8	4	4	7
bkgrnds	4	6	4	5	4	3
boats	15	16	17	16	13	15
buildngs	14	13	15	13	11	13
children	9	4	8	5	5	10
citytwn	15	16	18	14	14	18
coasts	17	15	20	17	16	15
dom_an	10	6	7	9	5	8
dsrtcyn	7	12	6	9	7	11
fields	19	19	23	20	20	21
food	5	4	4	7	5	4
lakerivr	13	11	13	10	12	12
lifestyl	16	13	16	18	14	14
mountain	15	14	19	12	15	16
objects	2	2	2	2	2	2
people	11	12	12	10	10	12
planes	17	18	21	15	17	19
plants	3	4	3	3	3	6
skycloud	17	17	18	18	18	17
space	9	8	8	7	8	9
speclocsn	18	18	22	19	19	20
sunset	20	20	24	21	21	22
texture	8	9	9	8	6	8
treeleav	10	7	10	8	9	7
underwtr	10	10	11	9	9	12
watrfall	12	11	14	11	12	11
wildanim	5	3	5	6	3	5

Table 7.2 Ranking Using Composite Bitplane Signatures for the Test Images in Figure 7.11

The results of similarity retrieval are given in Table 7.3. From the images in Appendix B, "auto006" and "auto007" bear close resemblance to "auto008" and selected as the top three ranked images. The images are repeated in Figure 7.12. Even the distorted variances of the original image can select "auto008" as the top ranked

image. This shows the similarity retrieval of using wavelet coefficients indeed works well.

File	A. original	B. blur	C. sharp	D. oil paint	E. pixelize	F. spread
auto001	22	19	23	24	23	21
auto002	5	5	8	7	5	5
auto003	6	6	6	6	6	7
auto004	8	6	10	8	8	6
auto005	24	20	25	25	25	22
auto006	2	2	2	2	2	2
auto007	3	3	3	3	3	3
auto008	1	1	1	1	1	1
auto009	23	21	24	22	24	22
auto010	10	6	7	9	9	8
auto011	12	10	10	8	10	12
auto012	20	17	21	18	22	19
auto013	17	15	18	19	17	16
auto014	15	12	13	11	14	15
auto015	19	16	18	16	19	17
auto016	18	16	17	20	17	17
auto017	21	18	22	23	21	20
auto018	9	7	8	10	9	8
auto019	15	12	15	14	15	14
auto020	16	13	16	15	16	14
auto021	13	11	14	12	12	13
auto022	15	12	12	14	14	13
auto023	10	8	9	9	8	8
auto024	8	7	10	8	8	9
auto025	17	15	19	18	17	16
auto026	9	9	9	9	9	11
auto027	11	7	11	12	11	10
auto028	18	14	18	18	18	16
auto029	14	10	14	13	13	12
auto030	10	9	10	10	8	11
auto031	7	6	5	5	7	9
auto032	20	17	20	21	20	18
auto033	10	6	7	8	8	6
auto034	17	15	19	17	17	16
auto035	4	4	4	4	4	4

Table 7.3 Ranking Within "Auto"

auto008

(*i*) Top Ranked Target Image

auto006

(*ii*) Second Ranked Target Image

auto007

(*iii*) Third Ranked Target Image

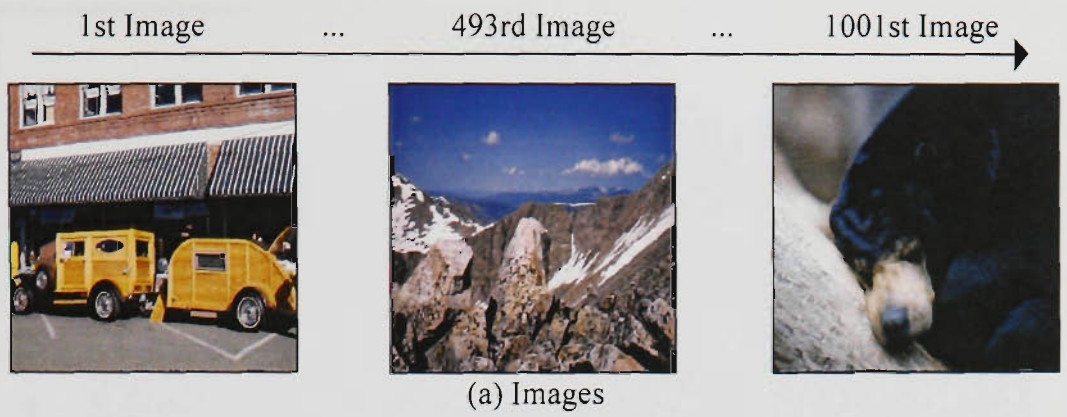
Figure 7.12 Top Three Ranked Images for the Test Images in Figure 7.11

7.3.6 Analysis of Composite Bitplane Signatures

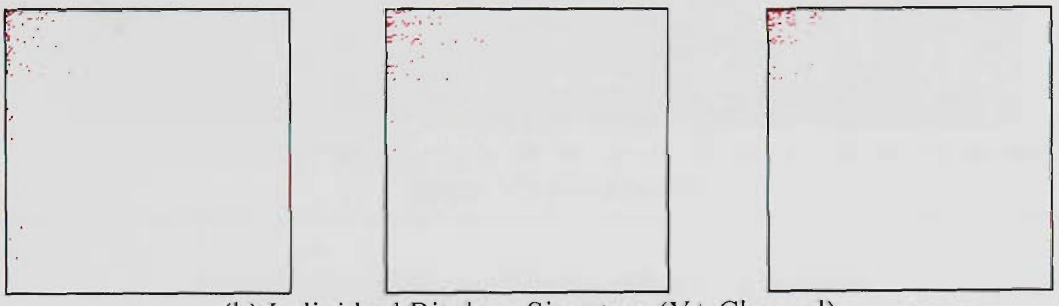
Two most important issues in Composite Bitplane Signature are: 1) the robustness of the adopted signature scheme for similarity retrieval, and 2) the behavior of the composite signatures. If the wavelet decomposition is used to generate the bitplane signature, the results from Section 7.3.5 indicate that using the signs of m largest magnitude coefficients is indeed a good way to generate bitplane signatures. In this Section, we will focus on the second issue.

As shown in previous sections, the bits tend to be concentrated around the scaling coefficient (i.e. the coarsest subband) due to multi-resolution nature of wavelets. This characteristic will limit our freedom to add too many signatures into any composite bitplane signature. Therefore, in order to study the behavior of forming a composite bitplane signature from a sequence of images, we separate the study into three parts.

Firstly, we use the natural ordering of the images and add all the bitplane signatures successively with $m = 128$. Figure 7.13 provides a snapshot of such action for the first image, the 493rd image, and the last image. Only the Y color channel of the individual signatures and the composite bitplane signatures are shown. From this experiment, the distributions of bits into subbands are emerging after a number of insertions.

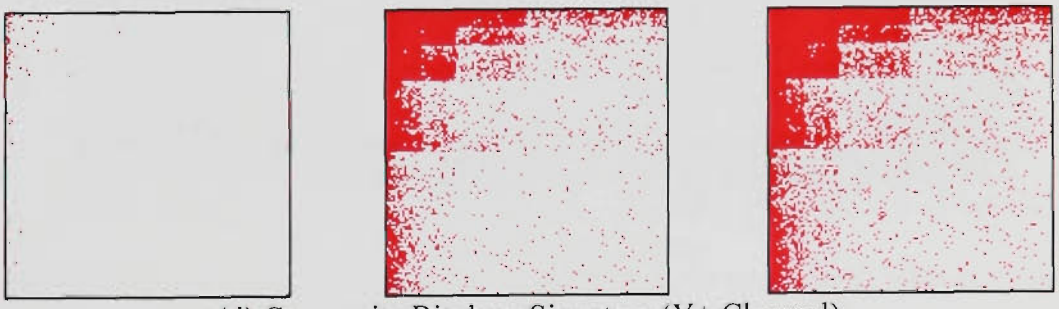


(a) Images

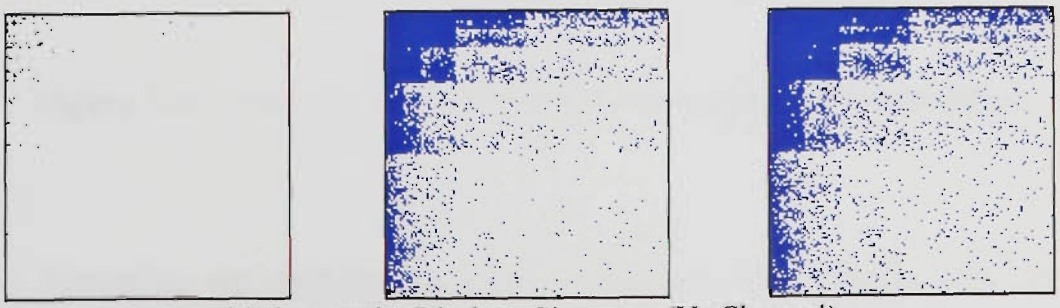


(b) Individual Bitplane Signature (Y+ Channel)

(c) Individual Bitplane Signature (Y- Channel)

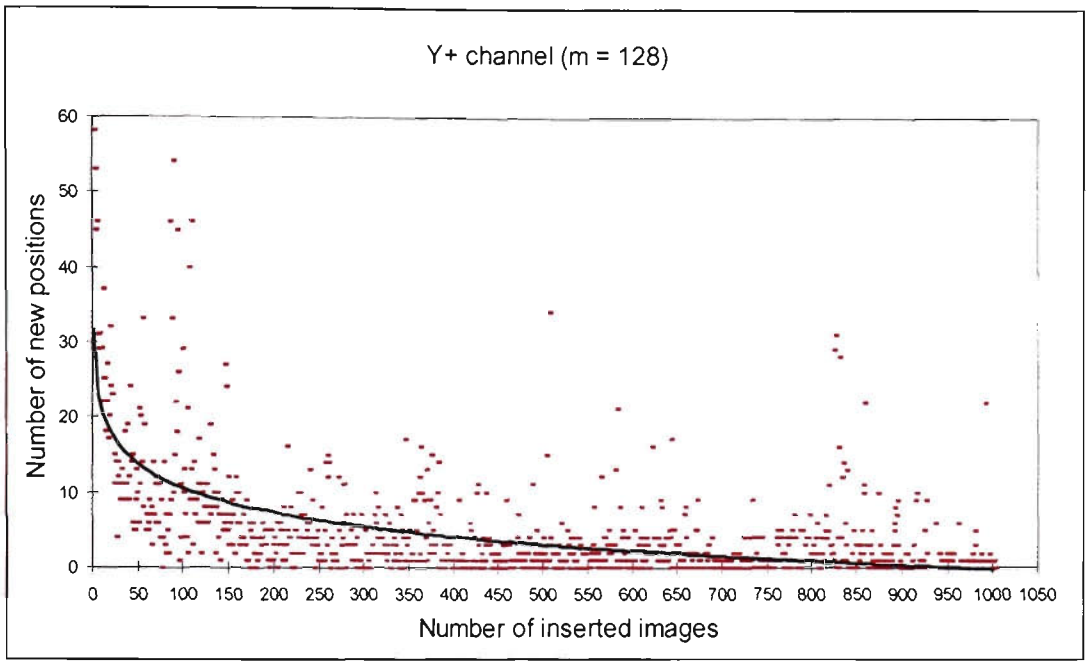


(d) Composite Bitplane Signature (Y+ Channel)

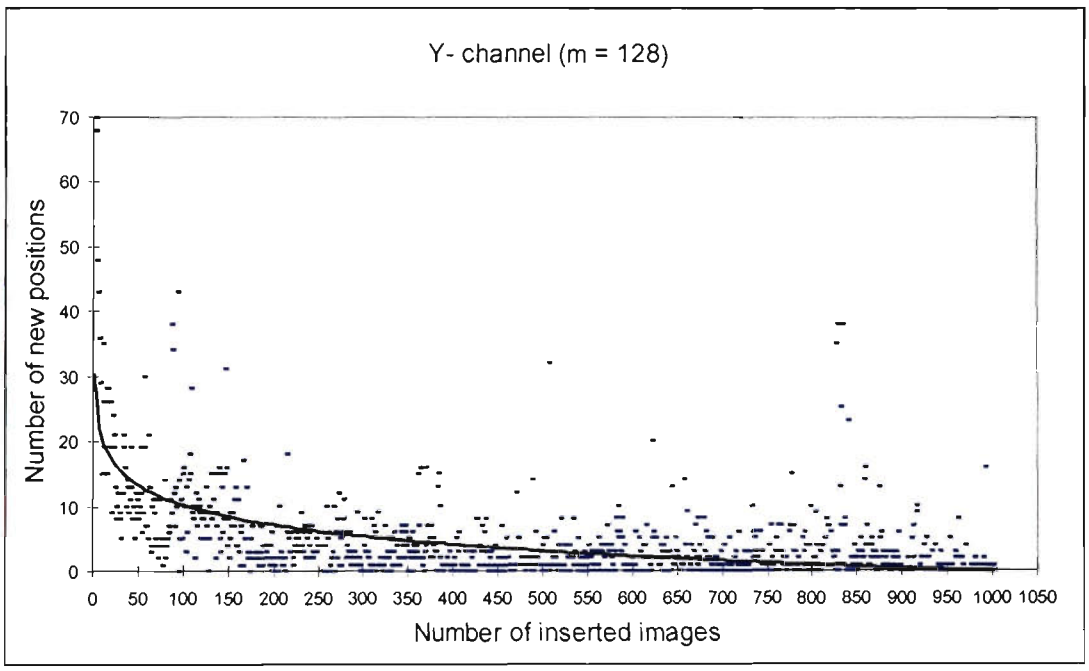


(e) Composite Bitplane Signature (Y- Channel)

Figure 7.13 A Snapshot of Overlaying Bitplane Signatures



(a) Number of Non-colliding Bits (Y+ Channel)



(b) Number of Non-colliding Bits (Y- Channel)

Figure 7.14 Analysis for the Successive Insertion of 1001 Signatures

Secondly, we analyze the total number of non-colliding bits for each successive insertion of the above experiment. Since the bit patterns for the composite

bitplanes are not randomly determined, we can expect the number of non-colliding bits to diminish quickly after a certain threshold is reached. The trends of Y+ and Y- channels are shown in Figure 7.14. The result indicates that the total number of non-colliding bits decreases rapidly at around 25 insertions. This is an estimation only and should be used as a rough guideline. In other words, if a bitplane of 128 x 128 matrix with m = 128, a suitable number of images from which we can form a composite bitplane signature using the wavelet scheme should not exceed 25 images. However, if a set of similar images is given to form a composite bitplane signature, this number can increase. This is because similar images should have similar signatures. We can accommodate more images without over-populating the bitplane.

Lastly, since our previous experiment used only m = 128 and in order to thoroughly study the effect with different m for a given sequence of images, we repeat our experiments for m = 32, 64, 128 and 256. The first 50 insertions are reported in Table 7.4. Also, we shuffle the images randomly and repeat all the experiments. The results are shown in Table 7.5. Again, only the first 50 insertions are reported. We observe from the results that, despite the matrix is sparser for smaller m, the number of non-colliding bits decreases as m decreases. This is largely due to the fact that the m largest magnitude coefficients are most likely selected near the scaling coefficient. Also, the collision is less likely to occur for the random ordering of images as expected. Hence, as the rule of thumb observed from the results, 20% of m is roughly the maximum number of images to be inserted into a composite bitplane signature.

	Original sequence							
	32		64		128		256	
	Y+	Y-	Y+	Y-	Y+	Y-	Y+	Y-
1	14	18	27	37	58	70	112	144
2	8	20	21	35	45	68	100	126
3	16	13	28	24	53	48	104	101
4	7	15	21	23	46	43	99	66
5	10	12	17	20	31	36	61	83
6	8	8	19	12	29	36	60	64
7	8	5	15	10	31	29	54	67
8	6	3	13	5	22	15	45	50
9	8	7	17	17	29	35	50	72
10	8	9	17	14	37	26	62	65
11	7	5	8	14	25	19	67	54
12	7	7	12	15	25	26	57	58
13	2	3	10	8	18	28	50	66
14	5	3	11	10	22	15	52	41
15	4	5	12	13	27	28	45	45
16	0	2	2	12	17	26	47	45
17	4	5	8	14	20	26	42	43
18	8	5	16	8	32	19	57	45
19	4	4	10	4	24	9	34	37
20	3	4	10	12	23	24	41	46
21	3	2	13	4	23	19	38	47
22	6	3	7	10	15	21	55	45
23	3	4	5	4	11	13	23	32
24	4	4	8	4	14	8	30	24
25	1	4	3	4	4	10	22	38
26	5	3	4	6	14	12	35	22
27	1	2	1	2	12	8	22	27
28	0	2	3	4	11	8	36	23
29	1	2	3	4	9	9	24	31
30	5	1	5	7	9	5	17	22
31	0	3	3	7	13	12	38	28
32	4	2	4	8	11	21	25	28
33	1	2	8	8	11	19	31	39
34	2	4	5	9	18	16	27	27
35	1	2	2	3	9	10	16	25
36	4	0	9	1	11	14	36	31
37	3	4	8	3	19	13	38	18
38	1	3	0	5	9	9	30	28
39	3	4	8	5	12	11	32	22
40	5	11	10	14	24	19	50	40
41	6	4	13	4	14	13	34	38
42	5	1	7	1	15	8	30	14
43	1	1	2	4	6	10	12	11
44	1	1	5	2	5	10	25	29
45	2	0	3	1	5	5	12	7
46	2	3	4	6	9	12	24	22
47	3	1	2	3	13	9	26	18
48	4	1	7	1	10	11	18	19
49	2	2	5	2	6	8	15	16
50	2	1	1	4	11	10	37	40

Table 7.4 The First 50 Insertions for $m = 32$, 64, 128 and 256 Using the Natural Grouping of the Image Collection

	Random sequence							
	32		64		128		256	
	Y+	Y-	Y+	Y-	Y+	Y-	Y+	Y-
1	17	15	33	31	64	64	129	127
2	13	15	25	31	51	59	112	112
3	10	15	26	24	54	41	103	80
4	11	8	25	18	52	36	90	84
5	8	9	19	24	36	42	84	86
6	9	8	21	16	37	33	75	71
7	12	9	23	13	41	25	70	67
8	4	8	8	11	26	21	63	48
9	9	4	11	8	22	24	46	48
10	4	5	13	22	38	37	79	64
11	3	10	9	6	25	26	47	53
12	3	5	9	7	14	16	21	41
13	6	3	10	7	16	20	47	51
14	5	7	14	10	26	21	53	44
15	5	3	5	6	19	14	33	43
16	5	6	11	10	26	27	80	55
17	7	4	15	7	18	19	35	28
18	12	12	25	24	38	54	57	102
19	2	0	4	6	18	9	52	32
20	4	1	6	8	23	9	32	25
21	3	2	7	4	10	9	20	24
22	4	3	10	12	24	26	63	54
23	1	5	6	10	7	20	22	31
24	5	3	5	6	13	14	26	27
25	7	3	8	6	12	20	14	30
26	3	3	5	6	13	17	35	46
27	1	2	4	6	8	7	19	22
28	2	3	5	9	10	15	18	21
29	7	6	16	20	40	44	93	85
30	1	1	5	8	10	19	28	45
31	6	7	12	8	15	9	26	22
32	2	3	2	8	6	10	17	17
33	1	4	6	4	17	6	30	16
34	0	2	4	7	15	15	29	37
35	8	1	5	3	11	11	20	20
36	4	1	2	5	14	10	24	29
37	0	3	1	4	7	3	9	18
38	0	2	5	5	9	7	23	17
39	2	2	6	2	11	8	23	16
40	4	2	8	6	14	10	32	21
41	1	3	3	4	6	9	10	17
42	4	3	11	5	15	12	31	37
43	3	0	5	3	6	12	14	14
44	6	2	9	8	15	17	46	34
45	2	0	3	1	7	9	28	20
46	4	3	17	8	35	22	83	48
47	2	4	6	9	11	20	31	42
48	5	5	8	9	15	20	34	46
49	1	1	5	2	10	5	22	24
50	5	9	12	11	24	16	35	29

Table 7.5 The First 50 Insertions for $m = 32$, 64, 128 and 256 Using the Random Shuttle of the Image Collection

7.3.7 Inverted Image Retrieval

To test the IIIC model, we perform the TFM specifications on the entire image collection. Using the sample images in Appendix A, we illustrate the corresponding TFM specifications in Table 7.6. The facts are prescribed using the conventions specified in Chapter 6. Once the facts are specified, we perform pre-processing steps to reduce words into their morphological roots. Since we do not use any tools to perform this task, this step is done manually. For a complete implementation, we need lexical tools to check against stopwords and to perform stemming. Entries to our inverted list are then formed using the TFM specifications and the bitplane signatures.

Images	TFM Specifications
auto001	woody CHEVY
auto035	red PICKUP
birds001	green PARROT Clings TREE
birds036	little GOOSE Sits GRASS
bkgrn001	decorative DRAGON
bkgrn036	brown ROOF
boats001	busy MARINA
boats035	SAILBOAT
build001	angled BUILDING
build035	COLLEGE
child001	BLANKET Wraps happy BABY
child035	CHILD Plays TOYS At POOLSIDE
city001	aerial VIEW Over CITY
city035	central FLORENCE

Table 7.6 Continued

coast001	aerial VIEW Over small HARBOUR
coast036	DRIFTWOOD On BEACH
doman001	appaloosa HORSE
doman036	GOAT Stands ROCKS
dsrt001	CANYON
dsrt034	GORGE
field001	old BARN Over HILL FIELD
field030	young FIRS
food001	STANDS With FRUITS In MARKET
food038	APPLE With ORANGE
lakes001	LAKE At SUNRISE
lakes036	RIVER In WINTER
life001	PEOPLE Sit ROCKS FISHING
life035	PEOPLE Enjoy SUN-BATHING At LAKE
mount001	BOULDERS On MOUNTAIN
mount035	TREE-LINE Of MOUNTAIN
objec001	COMPUTERS
objec048	troll SCULPTURE Under BRIDGE
peopl001	MAN Climbs TRAFFIC-LIGHT
peopl038	WOMAN Sitting BOATDOCK At LAKE
plane001	PLANE On RUNWAY
plane036	TAIL Of TUNDERBIRDS
plant001	FLOWERS On MEADOW
plant038	asserted FLOWERS
cloud001	cumulus CLOUDS
cloud036	CLOUDS Between TREES
space001	ROCKET Launching clear SKY
space036	SHUTTLE Landing CHUTE
specl001	PEOPLE Watching BASEBALL At STADIUM
specl028	PEOPLE Walking STREET In RAIN

Table 7.6 Continued

sunst001	MOON Over desert PLATEAU
sunst036	WILLOW With BOATS
txtur001	adobe WALL
txtur035	DROPS Of WATER
tree001	green FOREST
tree036	white-trunked ASPENS
water001	red CORAL
water035	CRAB
wtrfl001	big FALLS
wtrfl036	FALLS On narrow CLEFT
wldan001	TIGER
wldan036	BEAR Sleeps LOG

Table 7.6 TFM Specifications for Images in Appendix A

We also crop out some of the related images to form additional entries. Figure 7.15 illustrates a search using a binary fact: CHILD, BICYCLE, Ride (Fact1, Fact2, Link).

child008: (92,133) - (188,302)

child008 (157,145) - (296,435)

child009 (22,39) - (199,287)

child009 (176,45) - (331,283)

Figure 7.15 Search Results Using Binary Fact; CHILD, BICYCLE, Ride

We also searched for entries with astronauts in space suit through signatures by a picture example using the first picture. The resulting list is shown in Figure 7.16 (For clarity, the ImageID and coordinates are omitted here).

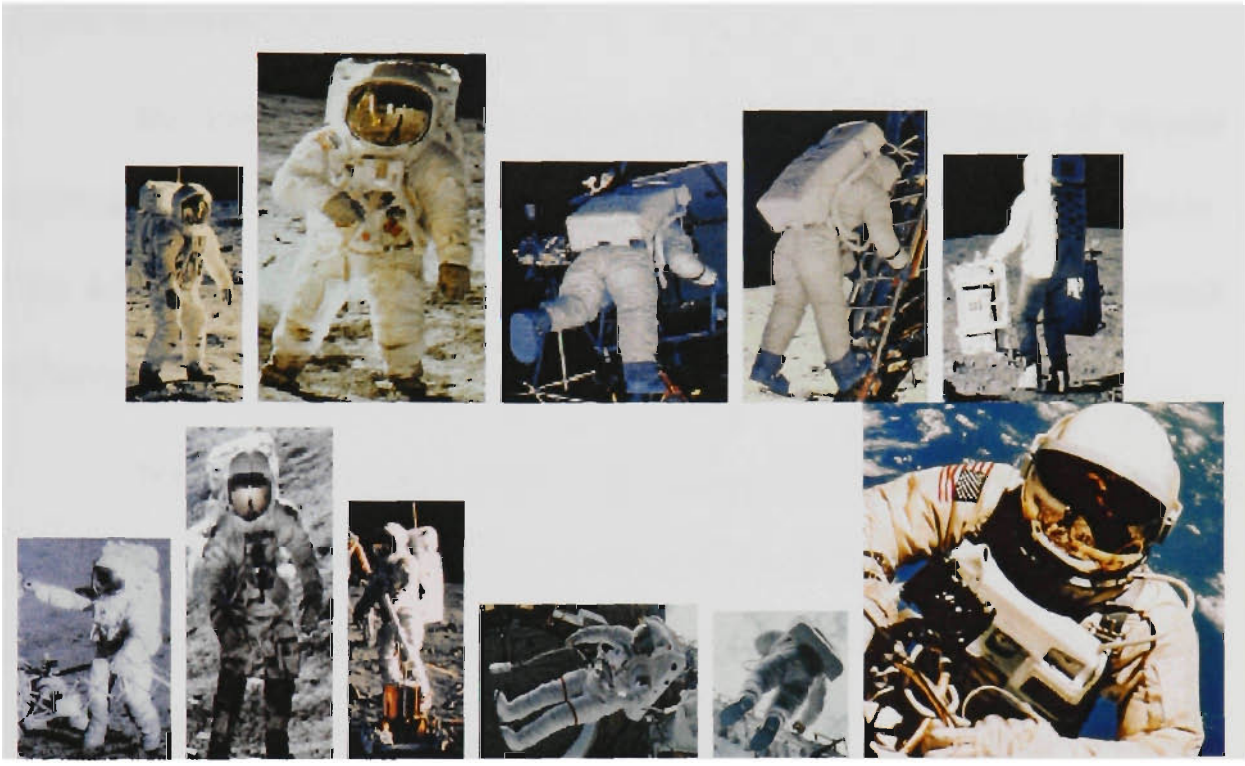


Figure 7.16 Astronauts in Space Suit

Compression is used to substantially reduce the storage requirement for the bitplane signatures. Typically, we can compress a bitplane with over 2K bytes into 150 bytes using off-the-shelf compression tools such as `gzip`. The compressibility is very good. This is largely due to the fact that our bitplane matrices are sparse in nature.

7.4 Summary

In this chapter, we perform a number of experiments to support the image indexing and retrieval paradigms described in previous chapters. The experimental results are summarized in this section.

We verified that a small portion of the largest magnitudes of wavelet coefficients at round 20% can reproduce an image with reasonable degree of clarity. This demonstrates the possibility of using the signs of those coefficients to generate bitplane signatures.

We determined that wavelet coefficients are concentrated around the scaling coefficient. This exhibits the multi-resolution nature of wavelet subbands with the coarser coefficients at the low-pass subbands. Also, directionally sensitive features are revealed in the subbands.

We proved that our wavelet signature scheme exhibits a good degree of similarity retrieval. It is quite robust for similar or distorted images. Using the sign bits of the wavelet decomposition are indeed a good way to generate signatures. However, the bits tend to be concentrated on the coarser coefficients. This limits our freedom to insert too many bitplane signatures into any composite bitplane signature. Otherwise, the uniqueness of the bitplane signatures is eroded. We have determined experimentally that 20% of m is roughly the maximum number of signatures to be inserted into a composite bitplane. For example, if m is 128, this should not exceed 25 images. This condition can be relaxed if a set of similar images are used to form a composite bitplane signature.

We demonstrated the feasibility of our inverted model. The inverted lists are constructed using the whole area of the images as well as the selected regions of the images. The high-level semantic indexing is done using the Ternary Fact Model. The low-level feature information is extracted using Bitplane Signature. The selected regions for any entry of the inverted lists are superimposed and a composite bitplane is formed. In this way, we can retrieve images or regions of images both semantically and syntactically.

Compression is an important issue to our indexing schemes. Since the bitplane of any image is very sparse, the compressibility is excellent. This opens an opportunity for us to perform image retrieval completely in memory, which will be increasingly essential for large scale visual information systems. Comparison of bitplane signatures can then be done by expanding the bitplane signatures without the need to fetch from hard disks. Furthermore, handling the compressed form of bitplanes can reduce the bandwidth for retrieval.

In this chapter, we demonstrated a number of experimental results for the different concepts developed in this thesis. Admittedly, further experimentations will be useful, and additional implementations such as a query processing and a good user interface will be beneficial. However, we believe that we have clearly illustrated our contributions outlined in this thesis through our experiments.

Chapter 8

Conclusion

8.1 Summary of Contributions

Indexing and retrieval of visual data is one of the most important functions of Visual Information Systems. In this thesis, we presented new paradigms for content-based image indexing and retrieval. These paradigms have a strong analogy to the conventional signature-based indexing and inversion-based postings. We are able to capture the essence of these concepts and develop them for image data. The success of using these paradigms for pictorial information opens a new horizon for the indexing and retrieval of multimedia data.

We summarize the major points and contributions of this thesis as follows.

- **Visual Information Management**

We presented our complete view on the management of visual information. Our definition of Visual Information Systems emphasizes the applicability of usable visual information for the benefit of an organization. The high-level components of Visual Information Systems include: 1) users, 2) computer-based applications such as ESS, DSS, MIS, TPS and many others, 3) Multimedia DBMS with supporting tools, and 4) the visual information and meta-data. The usable visual information is characterized by the inter- and intra- relationships of media types. We identify Multimedia DBMS should consist of: 1) Playout Manager, 2) Query Manager, 3) Transaction Manager, 4) Meta-data / Content-based Manager, and 5) Data Placement and Storage Manager.

- **Compressed Domain Indexing and Retrieval Techniques**

Effective and efficient retrieval of visual data is necessary to underpin any visual information system. Content-based image indexing and retrieval can be classified into two categories; 1) spatial domain techniques, and 2) compressed domain techniques. Recently, compressed domain techniques are emerged as a promising alternative to the indexing and retrieval problem. In this thesis, we analyzed this approach through comparative evaluation of the representative methods in JPEG, VQ and Wavelets. The advantages and disadvantages of using compressed data for image indexing and retrieval can then be drawn. Problems associated with this approach are also explored.

Many research developments in this thesis are related to compression issues and techniques. For example, we use wavelets for our Composite Bitplane Signature. To facilitate the investigation, we provided a concise and comprehensive treatment of the most popular techniques in data and image compression by way of a taxonomy. These include: 1) Distortionless data compaction - Huffman Coding, Arithmetic Coding, Predictive Coding and Ziv-Lempel Coding, and 2) Irreversible image compression - JPEG, Fractals, Wavelets, Vector Quantization and Block Truncation Coding.

- **Image Hashing**

Hashing has been used in different areas of computing and is an efficient paradigm for information retrieval. For image data, the concept of image hashing is shown to be highly applicable. We have analyzed the categories of different hash information from the possible data characteristics. The hash information can be in the form of: 1) a single numeric value, 2) a bit vector, 3) a numeric vector, 4) a bit matrix, and 5) a numeric matrix. Among these five types, the bit matrix is found to be the best candidate for image hashing, and we establish Image Hashing as the process of *image-to-bitplane transformation.*

- **Composite Bitplane Signature**

Signature-based indexing is well known in document or text retrieval. We have applied the signature-based approach to visually index image data and developed

Bitplane Signature as a form of image hashing. Bitplane Signature is a two-dimensional bit matrix in contrast to the one-dimensional bit vector of the text counterpart. We use the signs of the m largest magnitudes of wavelet coefficients to generate the bitplane signatures. A number of bitplane signatures can then be superimposed to form *Composite Bitplane Signature*. In this way, if a bitplane signature of an image is embedded into a composite bitplane signature, then we can locate the image without the need to examine the individual signatures. Since our signature scheme is able to produce similar signatures with similar images, content-based similarity retrieval is possible in our signature-based paradigm.

Many indexing structures in image retrieval are using feature vectors to perform filtering, but this is not what is required in visual information retrieval. Ranking, instead of filtering, is more appropriate for image retrieval. Our indexing scheme using composite bitplanes is shown to be far more superior. Firstly, it is a ranking system without expensive computation at retrieval. Secondly, we can produce a partial ranking of images without the need to compare and traverse all the images in the collection. This property is far better than many existing methods. This is achieved by ranking all the composite bitplane signatures not by the number of matched bits but by the least difference of 1 bits from the query signature. Because of our unique ranking scheme, a hierarchical accessing structure of composite bitplane signatures can also be constructed for extremely fast image searching using tree traversal and back tracking.

In our experiments, we studied our bitplane signature scheme under wavelets. These confirm that using the signs of m largest coefficients is indeed a good way to

generate bitplane signatures. We have also analyzed the behavior of the composite bitplane signatures. Since wavelet signatures are chosen for the hashing method to generate the bitplane signatures, we expect that the distribution of bits is dictated by the subband nature of wavelets, with the bits concentrating around the scaling coefficient (i.e. the coarsest subband) due to the multi-resolution nature of wavelets. We observed from our experimental results that, as a rule of thumb, 20% of m is roughly the maximum number of images to be inserted into a composite bitplane signature under wavelets.

- **Inverted Image Indexing and Compression**

Inversion-based indexing is another popular technique in text retrieval. In this thesis, we took the inverted paradigms and applied them to image data.

Information retrieval can be broadly classified into high-level semantic discovery and low-level syntactic recovery. Many image retrieval methods deal with low-level information such as color, texture and shape, which can be computed automatically. High-level semantic information such as events, scenes, actions and objects are normally deduced manually or at least semi-automatically. Our inverted model is capable of providing a unified framework for accessing image data both semantically and syntactically.

Many content-based image indexing and retrieval techniques are applied to only images as a whole picture, which has severe limitations. Very often, multiple objects or regions within an image perceived to be meaningful visual contents and

should be indexed separately. These regions may be related to other regions in the image collection. Therefore, an inverted list is an appropriate data structure to organize this information. This is similar to the words from different documents organized using an inverted file, and our inverted model for image data is developed from this paradigm.

The data model consists of four major components: 1) Picture Key, 2) Ternary Fact Model, 3) Composite Bitplane Signature, and 4) Image Coordinates. A picture key is a visual representation to signify the contents in each entry. This image serves as a representative item among the collections within the entry. The picture keys are stored in compressed form and often used for browsing or thumbnail purposes. The high-level semantic information for each entry is captured by a data model called Ternary Fact Model. The basic construct of Ternary Fact Model consists of discrete facts. Image facts may be classified into five types: 1) Elementary Facts, 2) Modified Facts, 3) Outline Facts, 4) Binary Facts, and 5) Ternary Facts. Each fact is a highly structured and homogeneous data item to facilitate searching. This canonical description is very useful to capture the semantics of the images in each entry. For low-level syntactic information, we superimpose the bitplane signatures of the image items in the entry and form a Composite Bitplane Signature for group searching. Since the likelihood of obtaining the target images can be found from the best ranked composite signatures, individual signatures are then analyzed for false matching. The component of Image Coordinates is to keep track of the coordinates of each image region in the entry. Some of the useful information such as proximity, spatial

relationship, foreground and background relationship can be obtained from the image coordinates.

Compression plays an important role for our inverted image model. Picture Keys and Composite Bitplane Signatures are two main areas where compression is certainly needed. If the picture keys are compressed, we can reduce the storage and alleviate the demands for the browsing operations. Bitplane signatures ought to be compressed for efficient retrieval. Since bitplanes are sparse in nature, the storage requirement can be significantly reduced by lossless compression. For an uncompressed bitplane taking over 2K bytes to store under our wavelet signature scheme, the size can be reduced to approximately 150 bytes using lossless compression such as LZW. Hence, we can easily fit a few thousand bitplanes in memory for searching. This property is an important factor to consider for large image databases.

- **Integrated Query Model**

We also developed an integrated query model for our inverted model. Boolean or ranked queries are supported in a unified framework. Users can pose an integrated query with high-level concept and low-level feature matching on the basis of exact or similarity retrieval. The processing steps for our query model include: 1) parsing and validating a query, 2) translating and transforming into an intermediate query, 3) optimizing the intermediate query, 4) formulating the execution plan, and 5) performing the query. Allowable logical operators for any query expression include **and**, **or**, **not** together with parentheses. We translate any general query into an

intermediate query with the form of *a conjunction of disjunctions*. An execution tree is built to evaluate the disjunctive terms at the bottom. A conjunction is then performed on the intermediate results. In order to provide a consistent result for both Boolean and ranked queries, we established a general approach to relax the selection criteria step-by-step to accommodate less precise answers. This provides a list of ranked images with the most probable candidates first.

8.2 Limitations

Content-based image indexing and retrieval is widely accepted to be a hard problem. This is largely due to the richness and relative imprecision of formulating meta-data and retrieval information to satisfy users' queries. The problem is further complicated if the design of the indexing scheme is geared toward a general-purpose image retrieval application.

Looking for a "swiss-knife" to solve many problems in visual information retrieval is rather unrealistic. Although we developed an integrated paradigm to retrieve high-level semantic information and low-level features within a unified framework, there are limitations.

- Our extraction of the salient characteristics from images is manually performed. Although the low-level indexing information is automatically extracted, the high-level semantic information to construct the TFM facts are entered manually. This is time consuming.

- Given a query expression, we translate the query into an intermediate expression in the form of a conjunction of disjunctions. Since the disjunctive expressions are evaluated first at the bottom of the execution tree, the number of intermediate candidates are likely to increase under the union operations. The union sets can be very large. This situation is far from ideal.

- Any content-based image retrieval technique should include measures to achieve higher degrees of translational, rotational, scaling and color invariance. For our wavelet signature scheme, it is our view that the raw data should undergo a pre-processing step before any wavelet decomposition takes place. Color space translation and size scaling are our two main steps geared towards this goal. However, we still feel that these measures are not sufficient.

- Hashing collisions limit our ability to uniquely identify signatures embedded in composite bitplane signatures. However, this problem is no different from any content-based retrieval techniques where feature vectors are used for indexing and retrieval purposes.

- The image indexing and retrieval techniques developed in this research project are designed for general-purpose retrieval systems. No domain specific knowledge is used in our signature-based and inversion-based indexing techniques. In some applications, we can make use of domain specific knowledge to improve the recall or precision of the underlying retrieval system.

- Throughout the thesis, we are concerned with the construction of signatures and inverted lists. The focus is on the insertion of indexing information into the data structures, and not much attention has been paid to other operations such as deletion. Although these operations are not frequent for image collections in comparison to relational databases, the impact of these operations should not be neglected in the overall implementation of the retrieval system.

8.3 Future Directions

We list below some of the possible directions directly related to the different aspects of our research.

Investigation of other schemes for signature generation. We only use wavelet decomposition for our signature-based indexing. To achieve more uniform and better distribution of bits, further exploration of signature schemes is needed.

Automatic extraction of semantic information. As mentioned before, it is difficult to automate the extraction of high-level information. However, we should attempt to gather as much information as possible related to the images automatically or semi-automatically. These include: 1) analysis of voice data associated with the image, 2) natural language processing on the captions or descriptions, and 3) knowledge-based inference of existing ternary facts.

Visual language and Interfaces. A good visual interface is needed to enter the components of our inverted model. Also, integrated queries for high-level semantic

and low-level syntactic information require special attention to the construction of a suitable text-oriented query language and visual user interface.

Many researchers around the world including ourselves believed that visual information retrieval is still in its infancy. Many areas are waiting to be explored. The indexing and retrieval of visual data is probably the most critical area which requires major research efforts. We are happy to contribute a small advance toward this important area. Let me conclude this thesis with a Chinese proverb, [生 命 有 涯 , 知 識 無 涯] ~ "Life is limited. Knowledge is unlimited".

Bibliography

[ADJE97] Donald A. Adjeroh, and Kingsley C. Nwosu, "Multimedia Database Management - Requirements and Issues," *IEEE Multimedia*, Vol. 4, No. 3, July-September 1997, pp. 24-33

[ALTE96] Steven Alter, *Information Systems: A Management Perspective*, 2nd Ed., Menlo Park CA: Benjamin/Cummings, 1996

[ANGE97] Marios C. Angelides, and Schahram Dustdar, *Multimedia Information Systems*, MA: Kluwer Academic Publishers, 1997

[ANTO92] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies, "Image Coding Using Wavelet Transform," *IEEE Trans. on Image Processing,* Vol. 1, No. 2, April 1992, pp.205-220

[ASLA99] Y. Alp Aslandogan, and Clement T. Yu. "Techniques and Systems for Image and Video Retrieval," *IEEE Trans. on Knowledge and Data Engineering,* Vol. 11, No. 1, 1999, pp.56-63

[BARL96] Michel Barlaud, Philip A. Chou, Nasser M. Nasrabadi, David Neuhoff, Mark J.T. Smith, and John W. Woods, "Guest Editorial: Introduction to the Special Issue on Vector Quantization," *IEEE Trans. on Image Processing*, Vol. 5, No. 2, February 1996, pp.197-201

[BARN88] Michael F. Barnsley, and Alan D. Sloan, "A Better Way to Compress Images," *BYTE*, January 1988, pp.215-223

[BARN93] M. F. Barnsley, and L. P. Hurd, *Fractal Image Compression,* Wellesley MA: AK Peters, 1993

[BELL90] Timothy C. Bell, John G. Cleary, and Ian H. Witten, *Text Compression,* Enhlewood Cliffs NJ: Prentice Hall, 1990

[BERR93] P. Bruce Berra, Forouzan Golshani, Rajiv Mehrotra, and Olivia R. Liu Sheng, "Guest Editors' Introduction: Multimedia Information Systems," *IEEE Trans. on Knowledge and Data Engineering,* Vol. 5, No. 4, August 1993, pp.545-550

[BHAS97] Vasudev Bhaskaran, and Konstantinos Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures,* 2nd Ed, MA: Kluwer Academic Publishers, 1997

[BHAT94] Sanjiv K. Bhatia, and Chaman L. Sabharwal, "A Fast Perfect Hash Function for Image Databases," *IEA/AIE'94 Proceedings of the Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Yverdon, Switzerland: Gordon and Breach Science Publishers, 1994, pp.337-346

[CARD93] Alfonso F. Cardenas, Ion Tim Ieong, Ricky K. Taira, Roger Barker, and Claudine M. Breant, "The Knowledge-based Object-Oriented

PICQUERY+ Language," *IEEE Trans. on Knowledge and Data Engineering,* Vol. 5, No. 4, August 1993, pp. 644-657

[CAWK94] A.E. Cawkell, *A Guide to Image Processing and Picture Management.* Aldershot Hampshire: Gower, 1994

[CHAN80] Ning San Chang, and King Sun Fu, "Query-by-Pictorial-Example," *IEEE Trans. on Software Engineering,* Vol. SE-6, No.6, November 1980, pp.519-524

[CHAN90] Shi Kuo Chang, *Principles of Pictorial Information Systems Design.* Englewood Cliffs, NJ: Prentice Hall, 1990

[CHAN92] Shi Kuo Chang, and Arding Hsu, "Image Information Systems: Where Do We Go From Here? " *IEEE Trans. on Knowledge and Data Engineering,* Vol. 4 No. 5, October 1992, pp.431-442

[CHAN96a] Shi Kuo Chang, "Towards Multidimensional Languages", *Proc. International Conference on Visual Information Systems*, Melbourne, February 1996, pp. 9-19

[CHAN96b] Shi Kuo Chang, and Erland Jungert, *Symbolic Projection for Image Information Retrieval and Spatial Reasoning.* London UK: Academic Press, 1996

[CHAN97] Shih-Fu Chang, John R. Smith, Mandis Beigi, and Ana Benitez, "Visual Information Retrieval from Large Distributed Online Repositories," *Communications of the ACM*, Vol. 40, Number 12, December 1997, pp.63-71

[COVE91] Thomas M. Cover, and Joy A. Thomas, *Elements of Information Theory*, NY: John Wiley & Sons, 1991

[DASA95] Belur V. Dasarathy, *Image Data Compression: Block Truncation Coding*. Los Alamitos, CA: IEEE Computer Society Press, 1995

[DAVI85] G. B. Davis, and M. H. Olson, *Management Information Systems: Conceptual Foundations, Structure, and Development*, 2nd Ed., NY: McGraw-Hill, 1985

[DELB98] Alberto Del Bimbo, "A Perspective View on Visual Information Retrieval Systems," *IEEE International Workshop on Content-based Access of Image and Video Libraries,* 1998, pp.108-109

[DELP79] Edward J. Delp, and O. Robert Mitchell, "Image Compression Using Block Truncation Coding," *IEEE Trans. on Communications,* Vol. COM-27, No. 9, September 1979, pp.1335-1342

[DEVO92] R. DeVore, B. Jawerth, and B. Lucier, "Image Compression Through Wavelet Transform coding," *IEEE Trans. on Information Theory,* 38(2):719-746, March 1992

[ELMA94] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, 2nd Ed., CA: Benjamin / Cummings, 1994

[FALO84] Christos Faloutsos and Stavros Christodoulakis, "Signature Files: An Access Method for Document and Its Analytical Performance Evaluation," *ACM Transactions on Office Information Systems,* Vol. 2, No. 4, October 1984, pp.267-288

[FALO85] Christos Faloutsos, "Access Methods for Text," *ACM Computing Surveys,* Vol. 17, No. 1, March 1985, pp.49-74

[FALO87] Christos Faloutsos and Stavros Christodoulakis, "Description and Performance Analysis of Signature File Methods for Office Filing," *ACM Transactions on Office Information Systems,* Vol. 5, No. 3, July 1987, pp.237-257

[FALO92] Christos Faloutsos, "Signature Files," in W.B. Frakes and R. Baeza-Yates (Eds.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ:Prentice Hall, 1992, pp.44-65

[FISH92] Yuval Fisher, "Fractal Image Compression," *ACM SIGGRAPH '92 Course Notes.*

[FISH95] Yuval Fisher (Ed.), *Fractal Image Compression: Theory and Application.* NY:Springer-Verlag, 1995

[FOLK92] Michael J. Folk, and Bill Zoellick, *File Structures,* 2nd Ed., MA:Addison-Wesley, 1992

[FOUR94] Alain Fournier (Ed), *Wavelets and Their Applications in Computer Graphics,* ACM SIGGRAPH '94 Course Notes.

[FURH94] Borko Furht, "Multimedia Systems:An Overview," *IEEE Multimedia,* Spring 1994, pp.47-59

[FURH95] Borko Furht, Stephen W. Smoliar, and HongJiang Zhang, *Video and Image Processing in Multimedia Systems.* MA:Kluwer Academic Publishers, 1995

[GEVE96] T. Gevers, and A. Smeulders, "Evaluating Color- and Shape-invariant Image Indexing for Consumer Photography", *Proc. International Conference on Visual Information Systems, Melbourne*, February 1996, pp.293-302

[GEMM95] D. J. Gemmell, H. M. Vin, D. D. Kandlur, R. V. Rangan, and L. A. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, May 1995, pp.40-49

[GERS93] Allen Gersho, and Robert Gray, *Vector Quantization and Signal Compression*. MA:Kluwer Academic Publishers, 1993

[GONZ92] Rafael C. Gonzalez, and Richard E. Woods, *Digital Image Processing*, MA:Addison-Wesley, 1992

[GROS92] W. Grosky, and R. Mehrotra, "Image Database Management," in M.C. Yovits (Ed.), *Advances in Computers,* Vol. 34, San Diego, CA: Academic Press, 1992

[GROS97a] William I. Grosky, Ramesh Jain, and Rajiv Mehrotra (Eds.), *The Handbook of Multimedia Information Management,* New Jersey:Prentice Hall PTR, 1997

[GROS97b] William I. Grosky, "Managing Multimedia Information in Database Systems," *Communications of the ACM*, Vol. 40, Number 12, December 1997, pp.73-80

[GUDI95] Venkat N. Gudivada, and Vijay V. Raghavan, "Content-Based Image Retrieval Systems," *IEEE Computer,* September 1995, pp.18-22

[GUPT91] Amarnath Gupta, Terry Weymouth, and Ramesh Jain, "Semantic Queries with Pictures: The VIMSYS Model," *Proceedings of the 17th International Conference on Very Large Data Bases,* Barcelona, September 1991, pp.69-79

[GUPT96] Amarnath Gupta, "Visual Information Retrieval Technology: A Virage Perspective," Virage Inc., May 8, 1996

[GUPT97a] Amarnath Gupta, and Ramash Jain, "Visual Information Retrieval," *Communications of the ACM*, Vol. 40, Number 5, May 1997, pp.70-79

[GUPT97b] Amarnath Gupta, Simone Santini, and Ramash Jain, "In Search of Information in Visual Media," *Communications of the ACM*, Vol. 40, Number 12, December 1997, pp.35-42

[HARM92] Donna Harman, Edward Fox, Ricardo Baeza-Yates, and W. Lee, "Inverted Files," in W.B. Frakes and R. Baeza-Yates (Eds.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ: Prentice Hall, 1992, pp.28-43

[HELD96] Gilbert Held, *Data and Image Compression: Tools and Techniques*, 4th Ed., West Sussex: John Wiley & Son, 1996

[HIBL92] J.N. David Hibler, Clement H.C. Leung, and Keith L. Mannock, and Njagi K. Mwara, "A System for Content-based Storage and Retrieval in An Image Database," *IS&T/SPIE Proceedings on Image Storage and Retrieval Systems,* San Jose CA, Vol. 1662, February 1992, pp.80-92

[HILT94] Michael L. Hilton, Bjorn D. Jawerth, and Ayan Sengupta, "Compressing Still and Moving Images with Wavelets" *Multmedia Systems*, Vol.2, No. 3. 1994

[HOFF97] Roy L. Hoffman, *Data Compression in Digital Systems*, NY: Chapman & Hall, 1997

[IDRI95] F. Idris, and S. Panchanathan, "Storage and Retrieval of Compressed Images," *IEEE Trans. on Consumer Electronics,* Vol. 41, No. 3, Aug 1995, pp.937-941

[IDRI96] F. Idris, and S. Panchanathan, "Algorithms for the Indexing of Compressed Images," *Proc. International Conference on Visual Information Systems,* Melbourne, February 1996, pp.303-308

[IDRI97] F. Idris, S. Panchanathan, "Storage and Retrieval of Compressed Images Using Wavelet Vector Quantization," *Journal of Visual Languages and Computing*, Vol. 8, No. 3, Academic Press, June 1997, pp.289-301

[ISOD1] (ISO DIS 10918-1) ISO Draft International Standard: *Digital Compression and Coding Of Continuous-Tones Still Images. Part 1: Requirements and Guidelines.*

[ISOD2] (ISO DIS 10918-2) ISO Draft International Standard: *Digital Compression and Coding Of Continuous-Tones Still Images. Part 2: Compliance Testing.*

[ISOD3] (ISO DIS 10918-3) ISO Draft International Standard: *Digital Compression and Coding Of Continuous-Tones Still Images. Part 3: JPEG Extension (26-8-94).*

[JACO95] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin, "Fast Multiresolution Image Querying," Proc. SIGGRAPH 95 (Los Angeles, CA, August 6-11, 1995). *In Computer Graphics Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH,* pp.277-286

[JACQ92] Arnaud E. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," *IEEE Trans. on Image Processing,* Vol. 1, No. 2, January 1992, pp.18-30

[JACQ93] Arnaud E. Jacquin, "Fractal Image Coding: A Review," *Proceedings of the IEEE,* Vol. 81, No. 10, October 1993, pp.1451-1465

[JAIN89] Anil K. Jain, *Fundamentals of Digital Image Processing,* Englewood Cliffs NJ: Prentice Hall, 1989

[JAIN93] Ramesh Jain (Ed.), *NSF Workshop on Visual Information Management Systems, ACM SIGMOD RECORD,* Vol.22, No. 3, September 1993, pp.57-75

[JAIN97] Ramesh Jain, "Guest Editor's Introduction: Visual Information Management," *Communications of the ACM*, Vol. 40, No. 12, December 1997, pp.30-32

[JOSE88] Thomas Joseph, and Alfonso F. Cardenas, "PICQUERY: A High Level Query Language for Pictorial Database Management," *IEEE Trans. on Software Engineering,* Vol. 14, No. 5, May 1988, pp.630-638

[KNUT73] Donald E. Knuth, *The Art of Computer Programming, Sorting and Searching,* Volume 3, MA: Addison-Wesley, 1973, pp.508-542

[KOUL95] Thomas M. Koulopoulos, and Carl Frappaolo, *Electronic Document Management Systems: A Portable Consultant*, NY: McGraw Hill, 1995

[LEUN90] C.H.C. Leung, and D. Hibler, and N. Mwara, "Picture Retrieval by Content Description," *Journal of Information Science,* Vol. 18, Elsevier, 1990, pp.111-119

[LEUN95] C.H.C. Leung, and Z.J. Zheng, "Image Data Modelling for Efficient Content Indexing," *Proc. International Workshop on Multimedia Database Management Systems,* New York, August 1995, IEEE Computer Society Press, pp.143-150

[LEUN97a] Clement Leung, "Guest Editor's Introduction: Visual Information Systems," *Journal of Visual Languages and Computing*, Vol. 8, No. 3, Academic Press, June 1997, pp.261-263

[LEUN97b] C. H. C. Leung and W. W. S. So, "Characteristics and Architectural Components of Visual Information Systems," in Clement H.C. Leung (Ed.), *Lecture Notes in Computer Science*, Berlin Heidelberg: Springer-Verlag, Vol. 1306, 1997, pp.1-12

[LEUN99] C. H. C. Leung and W. W. S. So, "Visual Information Systems," in Borko Furht (Ed.), *Handbook of Internet & Multimedia Systems & Applications*, Chapter 16, Boca Raton FL: CRC Press LLC, 1999, pp.361-374

[LIND80] Yoseph Linde, Andres Buzo, and Robert M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communications*, Vol. COM-28, No.1, January 1980, pp.84-95

[MARC96] Sherry Marcus, and V.S. Subrahmanian, "Foundations of Multimedia Database Systems," *Journal of the ACM*, Vol. 43, No. 3, May 1996, pp.474-523

[NARA96] A. Desai Narasimhalu, "Multimedia Databases," *ACM Multimedia Systems*, Springer-Verlag, Vol. 4, No. 5, October 1996, pp.226-249

[NASR88] Nasser M. Nasrabadi, and Robert A. King, "Image Coding Using Vector Quantization: A Review," *IEEE Trans. on Communications*, Vol. 36, No. 8, August 1988, pp.957-971

[NWOS96] Kingsley C. Nwosu, Bhavani Thuraisingham, and P. Bruce Berra (Eds.), *Multimedia Database Systems: Design and Implementation Strategies,* MA: Kluwer Academic Publishers, 1996

[NWOS97] Kingsley C. Nwosu, Bhavani Thuraisingham, and P. Bruce Berra, "Guest Editor's Introduction: Multimedia Database Systems - A New Frontier", *IEEE Multimedia*, Vol. 4, No. 3, July-September 1997, pp.21-23

[NELS92] Mark Nelson, *The Data Compression Book*, San Mateo CA: M&T Books, 1992

[NIBL93] Wayne Niblack (et. al.), "The QPIC Project: Querying Images By Content Using Color, Texture, and Shape," *IS&T/SPIE Proc. Storage and Retrieval for Image and video Databases,* San Jose CA, Vol. 1908, February 1993, pp.173-187

[OGLE95] Virginia E. Ogle, and Michael Stonebraker, "Chabot: Retrieval from a Relational Database of Images," *IEEE Computer,* September 1995, pp.40-48

[PENN93] William B. Pennobaker, and Joan L. Mitchell, *JPEG Still Image Data Compression Standard.* NY: Van Nostrand Reinhold, 1993

[PENT94] A. Pentland, R. W. Picard and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *IS&T/SPIE Proc. Storage and Retrieval for Image and video Databases II,* San Jose CA, Vol. 2185, February 1994, pp.34-47

[PIEP93] Josef Pieprzyk, and Babak Sadeghiyan, *Design of Hashing Algorithms, Lecture Notes in Computer Science*, Berlin Heidelberg: Springer-Verlag, Vol. 756, 1993

[PRAB97] B. Prabhakaran, *Multimedia Database Management Systems*, MA: Kluwer Academic Publishers, 1997

[RABI90] F. Rabitti, and P. Zezula, "A Dynamic Signature Technique for Multimedia Databases," *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1990, pp.193-210

[RAO95] K.R. Rao, *Workshop on Digital Image/Video/Audio Coding: Principles, Algorithms and International Standards,* Vol. I-III, Monash University, 10th-12th July 1995

[SALT89] G. Salton, *Automatic Text Processing*, Reading MA: Addison-Wesley, 1989

[SAME90a] Hanan Samet, *The Design and Analysis of Spatial Data Structures*, Reading MA: Addison-Wesley, 1990

[SAME90b] Hanan Samet, *Applications of Spatial Data Structures*, Reading MA: Addison-Wesley, 1990

[SAYO96] Khalid Sayood, *Introduction to Data Compression*, San Francisco CA: Morgan Kaufmann Publishers, 1996

[SCHA97] Peter Schäuble, *Multimedia Information Retrieval: Content-Based Information Retrieval from Large Text and Audio Databases*, MA: Kluwer Academic Publishers, 1997

[SENN90] J. A. Senn, *Information Systems in Management*, 4th Ed., Belmont CA: Wadsworth, 1990

[SHNE96] Michael Shneier, and Mohamed Abdel-Mottaleb, "Exploiting the JPEG Compression Scheme for Image Retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 18, No. 8, August 1996, pp.849-853

[SILB97] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, *Database System Concepts*, 3rd Ed., NY: McGraw Hill, 1997

[SMIT97] John R. Smith, and Shih-Fu Chang, "Visually Searching the Web for Content," *IEEE Multimedia,* Vol. 4, No. 3, July-September 1997, pp.12-20

[SO96a] W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Picture Coding For Image Database Retrieval," *International Picture Coding Symposium,* Melbourne, March 1996, pp.69-74

[SO96b] W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "A Multi-paradigm Approach to Visual Information Systems Design," *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems,* Hong Kong, June 27-28 1996, pp.265-273

[SO96c] W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Search Space Reduction Strategies for Image Databases," *Proceedings of the First International Workshop on Image Databases and Multimedia Search*, Amsterdam, The Netherlands, August 22-23 1996, pp.179-186

[SO97a] W. W. S. So, C. H. C. Leung, and Z. J. Zheng, "Analysis and Evaluation of Search Efficiency for Image Databases," in Arnold Smeulders and Ramesh Jain (Eds.), *Series on Software Engineering and Knowledge Engineering*, Vol. 8, Singapore: World Scientific, 1997, pp.253-262

[SO97b] W. W. S. So, and C. H. C. Leung, "Inverted Image Indexing and Compression," *SPIE Proc. Multimedia Storage and Archiving Systems II,* Dallas, Texas, Vol. 3229, 3-4 November 1997, pp.254-263

[SO98] Simon So, Clement Leung, and Philip Tse, "A Comparative Evaluation of Algorithms Using Compressed Data for Image Indexing and Retrieval," *International Conference on Computational Intelligence and Multimedia Applications*, Churchill, 9-11 February 1998, pp.866-872

[SO99] Simon So and Clement Leung, "A New Paradigm in Image Indexing and Retrieval using Composite Bitplane Signatures," *IEEE*

International Conference on Multimedia Computing and Systems, Florence, Vol. I, 7-11 June 1999, pp.855-859

[STAI96] Ralph M. Stair, *Principles of Information Systems: A Managerial Approach*, 2nd Ed., MA: Boyd & Fraser Publishing, 1996

[STOL95a] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin, "Wavelets For Computer Graphics: A Primer, Part I" *IEEE Computer Graphics and Applications,* 15(3):76-84, May 1995

[STOL95b] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin, "Wavelets For Computer Graphics: A Primer, Part II" *IEEE Computer Graphics and Applications,* 15(4):75-85, July 1995

[STOL96] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*, San Francisco CA: Morgan Kaufmann Publishers, 1996

[SUBR96] V. S. Subrahmanian, and Susil Jajodia (Eds.), *Multimedia Database Systems: Issues and Research Directions*, Berlin Heidelberg: Springer-Verlag, 1996

[THIM95] Heiko Thimm, and Wolfgang Klas, "Playout Management - An Integrated Service of a Multimedia Database Management System," *Proc. International Workshop on Multimedia Database Management Systems,* New York, August 1995, IEEE Computer Society Press, pp.38-47

[TREM79] Jean-Paul Tremblay, and Richard B. Bunt, *An Introduction to Computer Science: An Algorithmic Approach,* NY:McGraw-Hill, 1979, pp.531-549

[VELL95] A. Vellaikal, and C.C. Jay Kuo, "Content-Based Image Retrieval Using Multiresolution Histogram Representation", *SPIE Proc. on Digital Image Storage and Archiving Systems*, Vol. 2606, 1995, pp.312-323

[WALL92] Gregory K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Trans. on Consumer Electronics,* Vol. 38, No. 1, February 1992

[WELC84] Terry Welch, "A Technique for High-Performance Data Compression," *IEEE Computer,* Vol. 17, No. 6, June 1984, pp.8-19

[WITT87] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, Vol. 30, No. 6, June 1987, pp.520-540

[WITT94] I. Witten, A. Moffat, and T. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, NY: Van Nostrand Reinhold, 1994

[WOLF97] Haim J. Wolfson, and Isidore Rigoutsos, "Geometric Hashing: An Overview," *IEEE Computational Science and Engineering*, October-December, 1997, pp.10-21

[ZHEN95] Z. J. Zheng, and C. H. C. Leung, "Quantitative Measurements of Feature Indexing for 2D Binary Image of Hexagonal Grid for Image Retrieval," *IS&T/SPIE Proc. Storage and Retrieval for Image and Video Databases III,* San Jose, California, Vol. 2420, 1995, pp.116-124

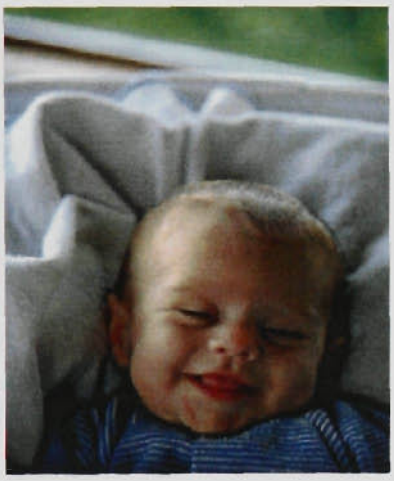
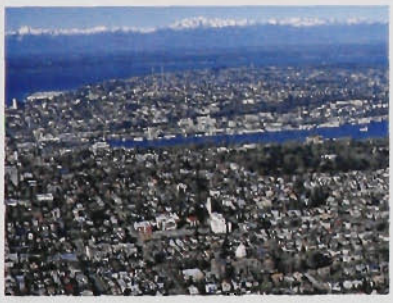
[ZIV77] Jacob Ziv, and Abraham Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory,* Vol. IT-23, No. 3, May 1977, pp.337-343

[ZIV78] Jacob Ziv, and Abraham Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Trans. on Information Theory,* Vol. IT-24, No. 5, September 1978, pp.530-536

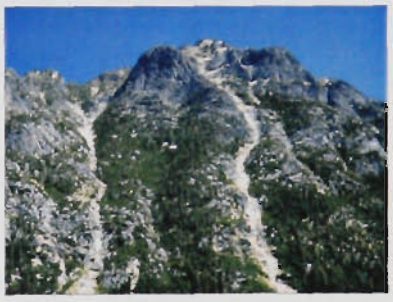
Appendix A: Test Images in MasterClips 35,000™

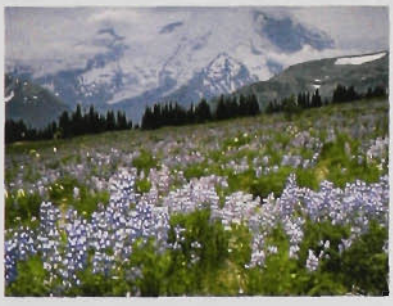
* Only the first and last images of each group in the collection are shown.

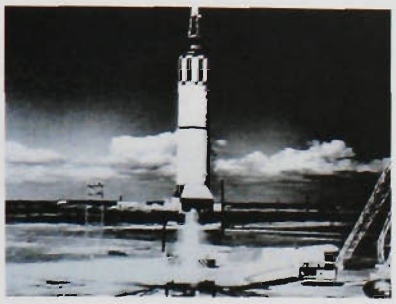
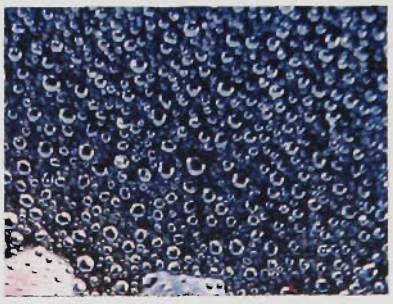
	**auto** ◁ auto001 ≡ . . . ≡ auto035 ▷	
	**birds** ◁ birds001 ≡ . . . ≡ birds036 ▷	
	**bkgrnds** ◁ bkgrn001 ≡ . . . ≡ bkgrn036 ▷	
	**boats** ◁ boats001 ≡ . . . ≡ boats035 ▷	

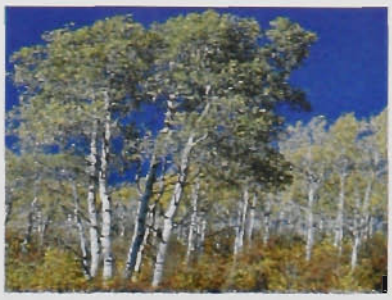
	**buildngs** ◁ build001 ≡ . . . ≡ build035 ▷	
	**children** ◁ child001 ≡ . . . ≡ child035 ▷	
	**citytwn** ◁ city001 ≡ . . . ≡ city035 ▷	
	**coasts** ◁ coast001 ≡ . . . ≡ coast036 ▷	

	**dom_an** ◁ doman001 ≡ . . . ≡ doman036 ▷	
	**dsrtcyn** ◁ dsrt001 ≡ . . . ≡ dsrt034 ▷	
	**fields** ◁ field001 ≡ . . . ≡ field030 ▷	
	**food** ◁ food001 ≡ . . . ≡ food038 ▷	

	**lakerivr** ◁ lakes001 ≡ . . . ≡ lakes036 ▷	
	**lifestyl** ◁ life001 ≡ . . . ≡ life035 ▷	
	**mountain** ◁ mount001 ≡ . . . ≡ mount035 ▷	
	**objects** ◁ objec001 ≡ . . . ≡ objec048 ▷	

	**people** ◁ peopl001 ≡ . . . ≡ peopl038 ▷	
	**planes** ◁ plane001 ≡ . . . ≡ plane036 ▷	
	**plants** ◁ plant001 ≡ . . . ≡ plant038 ▷	
	**skycloud** ◁ cloud001 ≡ . . . ≡ cloud036 ▷	

	**space** ◁ space001 ≡ . . . ≡ space036 ▷	
	**speclocsn** ◁ specl001 ≡ . . . ≡ specl028 ▷	
	**sunset** ◁ sunst001 ≡ . . . ≡ sunst036 ▷	
	**texture** ◁ txtur001 ≡ . . . ≡ txtur035 ▷	

	**treeleav** ◁ tree001 ≡ . . . ≡ tree036 ▷	
	**underwtr** ◁ water001 ≡ . . . ≡ water035 ▷	
	**watrfall** ◁ wtrfl001 ≡ . . . ≡ wtrfl036 ▷	
	**wildanim** ◁ wldan001 ≡ . . . ≡ wldan036 ▷	

Appendix B: Test Images In Group - AUTO


auto001


auto002


auto003


auto004


auto005


auto006


auto007


auto008


auto009


auto010


auto011

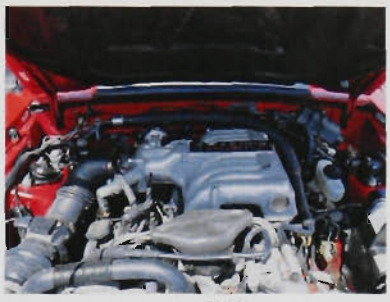

auto012

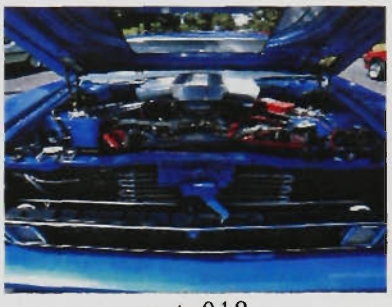

auto013


auto014


auto015


auto016


auto017


auto018


auto019

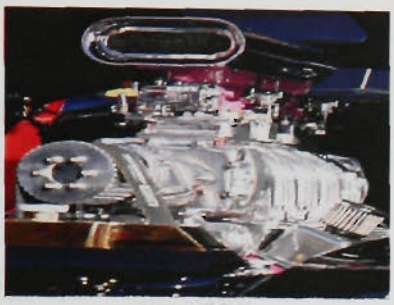
auto020

auto021


auto022


auto023


auto024


auto025

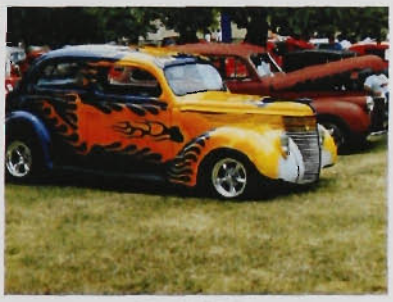

auto026


auto027


auto028


auto029


auto030


auto031


auto032

auto033


auto034


auto035