Université d'Ottawa · University of Ottawa

Image and Video Indexing Using Vector Quantization

by

Fayez M. Idris

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Ph.D. in Electrical Engineering

Ottawa-Carleton Institute of Electrical Engineering
Department of Electrical and Computer Engineering
Faculty of Engineering
University of Ottawa
September, 1999

I hereby declare that I am the sole author of this thesis.

Fayez M. Idris

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Fayez M. Idris

I further authorize the University of Ottawa to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutionsor individuals for the purpose of scholarly research.

Fayez M. Idris

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please, sign below and give address and date.

Abstract

Visual database systems require efficient indexing to enable fast access to the images and video. In addition, the large memory capacity and channel bandwidth requirements for the storage and transmission of visual data necessitate the use of compression techniques. Future multimedia applications are likely to increasingly store and transmit the visual information in compressed form. Hence indexing the visual content in compressed domain is expected to result in significant savings in computational complexity. Vector quantization (VQ) is an efficient technique for low bit rate image and video compression. In addition, the lower complexity of the decoder makes VQ attractive for low power systems and applications which require fast decoding. Most importantly, VQ is naturally an indexing technique, where a block of pixels is compactly represented using an index (label) which corresponds to a codebook.

In this thesis, we propose the novel concept of using VQ for joint compression and indexing of images and video. The images/image frames are compressed using VQ and the labels and codewords are employed in indexing the visual content. First, we present a review of image/video compression and indexing. We then propose two techniques in the VQ compressed domain for image indexing. In the first technique, the histogram of codewords weighted by the number of labels is used as feature vector for indexing. In the second technique, the histogram of the labels, which are used to represent an image, is used as an index. We also propose a new technique based on adaptive wavelet VQ, which provides an improvement in coding and retrieval performance. Here, the images are decomposed using wavelet transform followed by VQ of the transform coefficients. A usage map of codewords is generated for each image and is stored along with the image. In the retrieval process, the usage map of the query image (VQ encoded) is compared with the corresponding usage maps of the target images in the database.

Since video has both spatial and temporal dimensions, a straightforward extension of the image indexing techniques for video indexing is inefficient. We propose to employ both the spatial and temporal features for efficient indexing of video clips. The video sequence is partitioned into shots using the label maps of the individual frames and the camera operations and motion within each shot are then determined by further processing the label maps. Each shot is then represented using a *spatio-temporal* index. The spatial index represents the content of the key frame (image) of a shot, while the temporal index represents the motion and camera operations within the shot. Detailed simulations have been carried out using a large database of images and video sequences. Simulation results demonstrate the excellent retrieval performance of the proposed techniques at a significantly reduced computational complexity.

Acknowledgments

First of all, I wish to express my gratitude and appreciation to my supervisor, Prof. S. Panchanathan. His guidance, encouragement and support were the driving force for the successful completion of my thesis work.

The co-operation of the members of the Multimedia Communications Research Laboratory is appreciated.

Thanks to the staff of the Department of Electrical Engineering, University of Ottawa, for their cheerful help.

The financial support from the Canadian International Development Agency and the Jordan University of Science and Technology for my Ph.D. program is gratefully acknowledged.

Finally, I am thankful for the encouragement and help of my wife and my parents. It would not have been possible to finish this work without their constant support and understanding.

Contents

List of Figures

List of Tables

1

Introduction

Multimedia Information Systems are becoming increasingly important with the advent of broadband networks, high-powered workstations and compression technologies. There are several applications including distance learning, telemedicine, interactive television, digital libraries, multimedia news and geographical information systems which are already dominated and are expected to be increasingly populated by visual (images and video) information. Since, visual media involves large amounts of memory and computing power for storage and processing, the problems of flexible acquisition, processing and access have become more important.

It can be seen from the experience of textual media based information retrieval, for example over the World-Wide Web (WWW), that content-based indexing play a crucial role in the location and retrieval of the required information. The success of visual media search engines will therefore rely on the development of sophisticated tools and techniques for content-based indexing.

1.1 Problem Definition

Previous approaches to indexing of visual media have taken two directions. The first direction is a straightforward extension of textual databases. In this approach, the visual

contents are represented in textual form using keywords and attributes such as scene description, actor's name, director's name, etc. The keywords and attributes serve as indices to access the associated visual data. The Aggregate Data Manager (ADM) which is an interactive database system is an example of the extension of conventional databases to handle images [1]. ADM is based on a relational database model and uses the Structured English Query Language (SEQUEL) to query the stored images. This approach has the advantage that visual databases can be accessed using standard query languages such as SQL (Structured Query Language). However this entails extra storage and needs a large amount of manual processing. A more serious consideration from the point of view of reliability is that the descriptive data (i) do not conform to a standard language, (ii) are inconsistent, and (iii) might not capture the image/video content. Thus the retrieval results may not be satisfactory since the query is based on features that have been inadequately represented. The second approach to indexing visual data is to apply image analysis/understanding techniques. Image pattern recognition techniques are first used to classify an image/video into one of several categories. Interpretation to each class is then provided using knowledge bases. For example, the shape features in a scene may be mapped into symbols which represent elementary shapes such as circles, rectangles, etc. Semantics of the scene is developed by interpreting the collection of symbols. The interpretation is accomplished by using visual models and rules that imitate the human understanding. The Multi-sensor Image Database System (MIDAS) of Carnegie-Mellon University is a good example of this approach [2]. Two file types are distinguished in MIDAS: data files and description files. The data files contain the images while the description files (text files) contain a hierarchical symbolic representation of a scene. In addition, the description files contain relational tables which describe the

interrelations between the text files. This approach has two disadvantages: (i) the use of pre-defined categories limits the application of the database system and (ii) this is a computationally intense and complex task. As a result, there has been a new focus on developing image/video indexing techniques which (i) have the capability to retrieve visual data based on their contents, (ii) are domain independent, and (iii) can be automated.

The storage of uncompressed visual data requires considerable capacity. For example, the data rate for a 704×576 full motion video at 24 bits/pixel and a frame rate of 30 frames/second (4 Common Intermediate Format) is 276 Megabits/second. This implies that a DVD-ROM (write once, read many Digital Versatile Disc) with a storage capacity of 17 gigabytes can only store 8.4 minutes of video. Similarly, the transmission of uncompressed image/video data over digital networks requires a high bandwidth. For example, the transmission of one second of video over FDDI (Fiber Distributed Data Interchange) at a rate of 100 Megabits/second and 20% throughput rate for a shared network involves 13 seconds. Hence, it is necessary to use efficient image/video compression techniques to provide cost-effective solutions for the storage and transmission of visual data.

Many compression algorithms have been reported in the literature to reduce the storage and transmission requirements in image/video applications [3]-[27]. The International Standards Organization (ISO) has proposed the Joint Photographic Experts Group (JPEG) and the Moving Pictures Experts Group (MPEG) standards for image and video compression, respectively [23]-[25]. Recently, the Moving Pictures Experts Group has developed an audiovisual compression standard, referred to as MPEG-4, to support access and processing of audiovisual data at very low bit rates [26]-[27]. The scope of MPEG-4 is future delivery and storage systems allowing for high compression (10-64 kbits/sec), interactivity, scalability

of video and audio content, and support of natural and synthetic audio and video content. Recently, MPEG has initiated a new standard, called *Multimedia Content Description Interface* (MPEG-7). MPEG-7 will specify a standard set of descriptors that can be used to describe various types of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching of visual data.

Typically indexing and compression has been pursued independently as shown in Figure 1.1. Compression algorithms are concerned mainly with the optimization of distortion, bit rate and complexity without focusing on content accessibility. On the other hand, indexing techniques are usually designed ignoring the fact that it is very likely that images and video may be stored in the compressed form. Image and video indexing in the uncompressed domain (Figure 1.1) has the following disadvantages: First, processing of the uncompressed data is time consuming since image/video data are voluminous. Second, there is an auxiliary storage requirement to store the decompressed data. This reduces the overall system performance and storage efficiency.

To eliminate the problems of indexing in the uncompressed domain, it is necessary to combine image/video indexing and compression as shown in Figure 1.2. Image and video indexing in the compressed domain has two advantages. First, there is a reduction in computational cost as image/video are represented in the compressed form. Second, we note that many compressed bit streams typically contain information, such as motion vectors, which can help in deriving content-based indices.

Research in the area of combined image/video indexing and compression can be pursued in two directions: (i) to generate content-based indices in the compressed domain for existing compression techniques, and (ii) to develop novel compression techniques that are

optimized not only in terms of signal distortion, bit rate and complexity but also provide the feature of indexing.

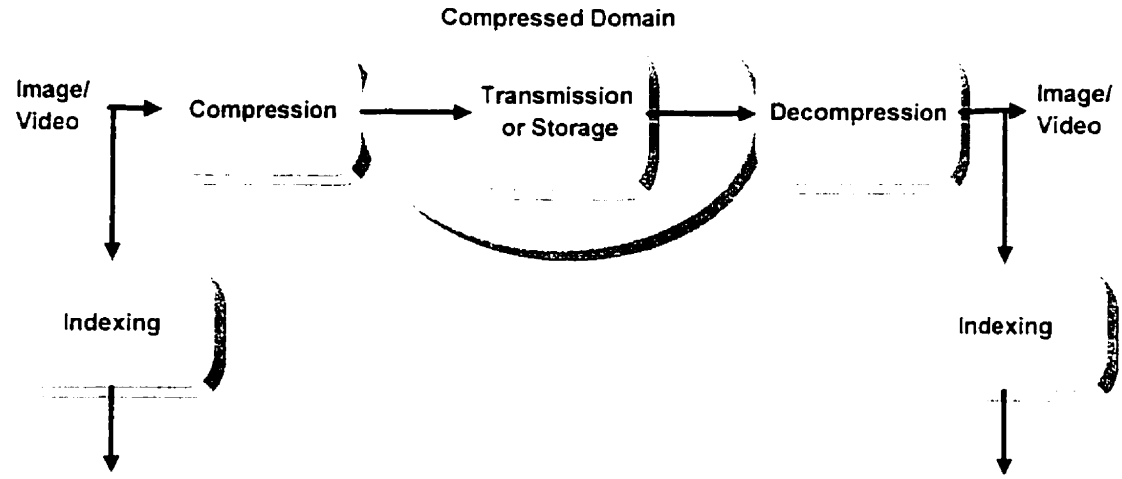


Figure 1.1: Indexing in the uncompressed domain.

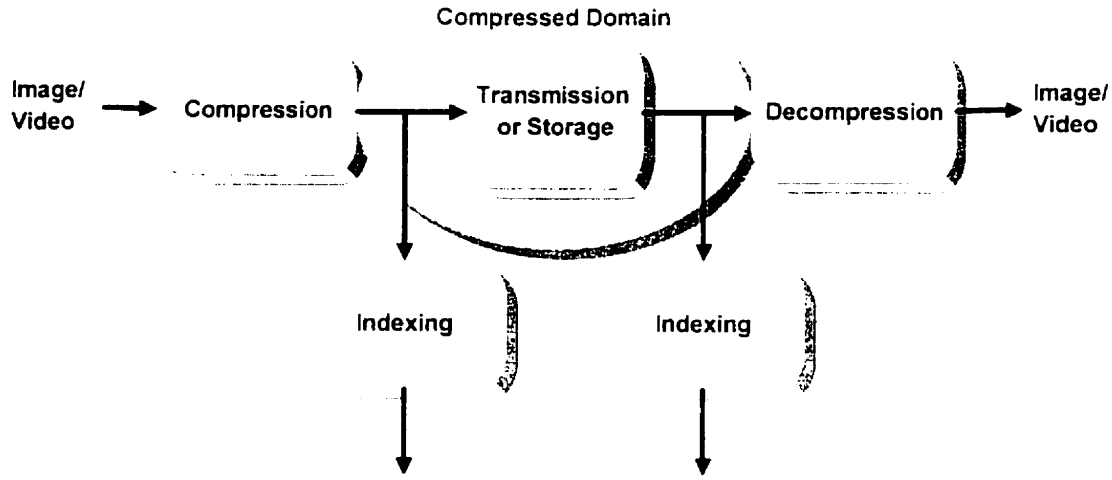


Figure 1.2: Indexing in the compressed domain.

1.2 Investigated Approach

Vector quantization (VQ) is an efficient technique for very low bit rate image and video compression [7]-[15]. Recently, VQ has been used to simplify image processing algorithms, such as enhancement, edge detection and reconstruction, by performing them simultaneously with the compression [18]. In VQ [7], the image to be compressed is decomposed into L-dimensional vectors. Using a nearest neighbor rule, each input vector is mapped onto the label of the closest codeword. The labels of the codewords are used to represent the input image. Image reconstruction is implemented by a simple table look-up procedure, where the label is used as an address to a table containing the codewords. In other words, VQ is naturally an indexing technique, where each subimage (vector) is mapped into an index (label). In addition, VQ has the following advantages:

- Fast decoding which makes it attractive for systems based on software only playback of video such as Intel's Indeo, Apple's QuickTime and Microsoft's Video.
- Reduced hardware requirements due to the simplicity of the decoder which makes it attractive for low power applications such as portable video-on-demand in wireless communications [13],[28].

Hence, VQ is a promising approach for combining compression with indexing.

In this thesis, we propose novel algorithms based on VQ to index images and video in the compressed domain. The proposed techniques combine compression and indexing, and provide excellent performance at both high and low compression ratios.

1.3 Summary of Contributions

The fundamental contribution of this thesis is the proposal of VQ as an efficient technique for joint compression and indexing of images and video. Following is a list of the individual contributions:

1. A critical survey of existing image and video indexing techniques [151]-[152].
2. Two new algorithms for the indexing of compressed images using vector quantization [154].
3. Algorithm for the integration of indexing and compression of vector quantized images using VQ and wavelet VQ [155].
4. Algorithm for the segmentation of video sequences in VQ domain..
5. Algorithm for the detection of camera operations in the VQ domain [156].
6. Spatio-temporal indexing of video sequences [157].

1.4 Thesis Outline

The thesis is organized as follows. Chapter 2 presents a review of image and video indexing techniques. Chapter 3 follows with a review of content-based image and video indexing techniques in the compressed domain.

In chapter 4, we propose two novel techniques for indexing images using VQ. In the first technique, for each codeword in the codebook, a histogram is generated and stored along with the codeword. We note that the superposition of the histograms of the codewords, which are used to represent an image, is a close approximation of the histogram of the image. This

histogram is used as an index to store and retrieve the image. In the second technique, the histogram of the labels of an image is used as an index to access the image.

In chapter 5, we propose a new technique based on adaptive vector quantization which integrates the index of an image within the compressed bit stream. Here, the index is generated at compression time and hence the proposed technique eliminates the need for a separate structure to store the indices. The performance of this technique is investigated in the spatial and transform domains.

In chapter 6, we propose an algorithm for video indexing using a *spatio-temporal* index. The spatial index represents the spatial content of the representative frame of a shot, while the temporal index represents the temporal content of the shot. The spatial index is based on the codewords used to compress the representative frame, while the temporal index is based on motion and camera operations within the shot. We present two algorithms for the detection of shot boundaries and camera operations, respectively. The proposed techniques are executed entirely in the compressed domain. This entails significant savings in computational and storage costs resulting in faster execution.

Finally, the summary and future research directions are presented in chapter 7 followed by the references.

2

Review of Image and Video Compression

Image and video data are voluminous: compression is essential for storage and transmission. The goal of image/video data compression is to reduce the number of bits required to represent an image/video signal while maintaining an acceptable fidelity [3]-[4]. Image/video compression is essentially a redundancy removal process. Efficient compression is achieved by exploiting the spatial, temporal and psychovisual redundancies. Spatial redundancy refers to the dependency between the pixel values in a local region of an image. Spatial redundancy is typically removed by employing intraframe coding techniques such as predictive coding, transform coding, etc. Temporal redundancy refers to the correlation between the successive frames in a video sequence and is usually removed by employing interframe compression techniques such as motion estimation/compensation, frame replenishment, etc. Most image/video frames contain psychovisual redundancies: that is, some information may be removed without sacrificing the subjective image quality. For example, two properties of the human visual system that may be exploited to a great

advantage are (i) lower sensitivity to faster moving objects and (ii) lesser perception of distortion at higher spatial frequencies.

In this section we present a review of image and video compression techniques. First, the concepts of lossless image/video data compression are presented in section 2.1. This is followed in section 2.2 by an overview of lossy compression techniques. In section 2.3, we present a review of vector quantization techniques. This is followed in section 2.4 by a review of wavelet transform. The JPEG, MPEG, and MPEG-4 standards are then discussed in sections 2.5, 2.6, and 2.7, respectively. We note that emphasis is placed on the review of vector quantization (VQ) methods as the focus of this thesis is essentially based on VQ. The summary is presented in section 2.8.

2.1 Lossless Compression

Lossless compression is concerned with minimizing the average number of bits per pixel without any loss in image quality; i.e. the decoder should reconstruct the exact input image from the encoded image. Information theory states that the source can be exactly encoded with H bits/pixel, where H is the source entropy. For a source with 2^b possible independent symbols with probabilities p_i, i=1,2, ... ,2^b, the *zeroth-order* entropy is given by:

$$H = -\sum_{i=1}^{2^b} p_i \log_2 p_i \qquad (2.1)$$

This results in a variable length code (VLC) where shorter codewords are assigned to more frequent pixel values and longer codewords are assigned to infrequent pixel values. Huffman and Arithmetic coding are the most popular approaches for lossless coding. Arithmetic coding achieves higher compression ratios than Huffman coding, but it is more difficult to

implement. Another technique for lossless coding is run length coding [3] in which a reduction in the bit rate is achieved by sequentially transmitting or storing the pixel values (run) followed by the number of its repetitions (length). Significant compression is possible if the input image is characterized by long runs.

2.2 Lossy Compression

Lossy or minimum distortion compression results in a reconstruction that is not identical to original (uncompressed) image. The purpose of lossy compression is to minimize the bit rate for a given average distortion or equivalently, to minimize the average distortion for a given bit rate. For an image source, the average distortion D is defined as

$$D = E\{d(A,\hat{A})\} \tag{2.2}$$

where $d(A,\hat{A})$ is a distortion measure between the source A and its reproduction $\hat{A}$. Clearly, the design of such an encoding scheme depends on the statistics of A and the characteristics $d(A,\hat{A})$. Rate distortion theory [5] provides the theoretical lower bound on the bit rate of any quantizer. For the source A with a probability function $p_A(A)$, the rate distortion function is defined as

$$R(D^*) = \min\{I(A,\hat{A})\} \tag{2.3}$$

where the minimum is taken over all the encoding schemes which result in average distortion D less than some value D^*, and $I(A,\hat{A})$ is the average mutual information between A and $\hat{A}$ defined as

$$I(A,\hat{A}) = \sum_{a}\sum_{\hat{a}} p_A(a)p_{\hat{A}/A}(\hat{a}/a)\log\frac{p_{\hat{A}/A}(\hat{a}/a)}{p_{\hat{A}}(\hat{a})} \qquad (2.4)$$

where $p_{\hat{A}/A}(\hat{A}/A)$ is the conditional probability for $\hat{A}$ given A. This definition of the rate distortion function indicates that for a given average distortion D, the minimum transmission rate is $R(D)$. Shannon's coding theorem states that it is possible to design an encoding system which achieves the average distortion D at a transmission rate arbitrarily close to $R(D)$.

Although the theory does not detail the design of such an encoding system, it is valuable in comparing the performance of different encoding schemes.

The techniques for lossless image/video compression can be classified into predictive, transform, wavelet/subband, vector quantization, fractal, and model-based coding. We now present a brief overview of each technique. Detailed description of the techniques can be found in [3]-[29].

Predictive Coding: Predictive compression exploits the mutual redundancy between neighboring pixels. Rather than encoding the pixel intensity directly, its value is first predicted from the previously encoded pixels. The predicted pixel value is then subtracted from the actual pixel value and the difference (prediction error) is quantized and coded for transmission. The quantized prediction error is used at the receiver to reconstruct the image.

Transform Coding: The basic concept in transform coding is to concentrate the important information in a few transform coefficients, which are then quantized, coded and transmitted. For still images, the input image is first divided into non-overlapping blocks $X_{m1,m2}$ of $M_1 \times M_2$ pixels. In order to decorrelate the image data, a two dimensional transform is then applied to $X_{m1,m2}$ as shown in The transformation maps $X_{m1,m2}$ into a two dimensional

array $Q_{u,v}$ of transform coefficients with the same dimension. Mathematically this operation is given by:

$$Q_{u,v} = \sum_{m1=0}^{M_1} \sum_{m2=0}^{M_2} X_{m1,m2} A_{u,v,m1,m2} \tag{2.5}$$

where $A_{u,v,m1,m2}$ is the transform kernel. The resulting coefficients $Q_{u,v}$, u=1,2,...,M_1 v=1,2,...,M_2 are then quantized, coded and transmitted. At the receiver, an inverse transform operation is applied to the quantized coefficients $Y_{u,v}$ to reconstruct the image:

$$Z_{m1,m2} = \sum_{u=0}^{M_1} \sum_{v=0}^{M_2} Y_{u,v} A^{-1}_{u,v,m1,m2} \tag{2.6}$$

$A^{-1}{}_{u,v,m1,m2}$ is the inverse transform kernel.

An optimum transform should result in statistically independent coefficients. Karhunen-Loeve transform is an optimum transform in terms of both the mean square error and subjective quality. However, it requires a large number of operations to compute; and is hence usually replaced by sub-optimal transforms such as Fourier transform, discrete cosine transform (DCT), or Hadamard transform. Although there are many transform techniques, DCT is widely used in practice because of its simplicity and it is performance which is close to the optimal Karhunen-Loeve transform [33]. DCT has been adopted in image and video coding standards, such as JPEG, MPEG and H.261.

Subband Coding: In subband coding, the input signal is filtered to create a set of subimages or subbands, each of which contains a limited range of spatial frequencies. The resulting subbands are downsampled to preserve the data rate. The subbands are then separately quantized, with attention being paid to bit allocation. Decompression is performed

by upsampling the decoded subbands, applying appropriate filters and adding the reconstructed subbands together.

Fractal Coding: A fractal is a geometric form where irregular details recur at different scales and angles which can be described by a transformations (*e.g.* an affine transformation). Fractal image compression is the inverse of fractal image generation, *i.e.* instead of generating an image from a given formula, fractal image compression searches for sets of fractals in a digitized image which describe and represent the entire image. Once the appropriate sets of fractals are determined, they are reduced to very compact fractal transform codes or formulas. In block fractal coding, an image is partitioned into a collection of non-overlapping regions known as range blocks. For each range block, a domain block and an associated transformation are chosen so that the domain block best approximates the range. These transformations are known as *fractal codes*. While the pixel data contained in the range and domain blocks are used to determine the code, they are not part of the code, resulting in a high compression ratio

Model-Based Coding: Model-based coding can be classified into 2-D and 3-D model-based coding. In 2-D model-based compression, the input image is segmented into regions exhibiting common features. For example, an image might be partitioned homogeneous regions and encodes their shapes and intensities. In 3-D model based compression are based on structural model of scenes. There are two approaches to 3-D model based compression. The first makes use of surfaces of the object modeled by general geometric models such as planes or smooth surfaces. Here, information such as surface structure and motion information is estimated from image sequences and utilized in compression. The second approach utilizes parameterized model of the object such as parameterized facial models.

Here, the parameterized models are usually given in advance. Model-based compression have achieved some of the highest compression ratios, however, they have high computational complexity.

Vector Quantization (VQ): Details of VQ are presented in section 2.3.

Wavelets: Details of wavelet coding will be presented in section 2.4.

These categories of compression techniques often overlap, combined (hybrid techniques) and they are often combined with lossless compression.

2.3 Vector Quantization

In VQ [10], a training set of representative images is decomposed into L-dimensional vectors. An iterative clustering algorithm such as the LBG algorithm is used to generate a codebook, $\{U_1, U_2, \ldots, U_N\}$, where N is the number of codewords in the codebook and $U_i=\{u_{i1}, u_{i2}, \ldots, u_{iL}\}$. The codebook is then made available at both the transmitter and the receiver. In the encoding process, the image to be compressed is decomposed into L-dimensional vectors. Each vector $V_i=\{v_{i1}, v_{i2}, \ldots, v_{iL}\}$ is mapped into another vector U_j. The mapping process is based on a minimum distortion or nearest neighbor rule; Compression is achieved by transmitting the label j corresponding to U_j. Image reconstruction is implemented by a table look-up, where the label j is used as an address to a table containing the codewords.

The steps involved in VQ as applied to image/video compression are vector formation, codebook design, and quantization.

2.3.1 Vector Formation

The first step in VQ is to decompose the input image into vectors. The image is partitioned into two dimensional blocks of equal or variable sizes. The features or values are extracted from the blocks and then rearranged into vectors. Various vector formation schemes have been proposed [7]-[10]. For example, the vectors can be formed from the original pixel values of the blocks; the transform coefficients of a block of pixels; the prediction error of a block; the pixel values of a block normalized by the average and the color components of a pixel.

2.3.2 Codebook Design

Linde *et al.* [8] have presented an algorithm for codebook design based on the two conditions for optimality, referred to as the generalized Lloyd or the LBG (Linde, Buzzo, and Gray) algorithm. The LBG algorithm is a variant of the K-means (C-means) clustering algorithm. In this algorithm, given an initial codebook, each training vector is assigned to its nearest neighbor codeword. Each codeword is then modified to minimize its distortion relative to the vectors assigned to it. This process continues iteratively until the change in distortion between two successive iterations is within a threshold of acceptance. The algorithm is described as follows:

1. Given an initial codebook, $C_0=\{W_i, i=1,2, \ldots, N\}$, a threshold $\varepsilon \geq 0$ and a training set $\{V_i, i=1,2, \ldots, K\}$, set m to 0 and D_{-1} to ∞.
2. Assign each input vector to its nearest neighbor codeword:

$$q(V_i) = W_{i,m} \quad iff \quad d(V_i, W_{j,m}) \leq d(V_i, W_{k,m}) \ \ for \ \ all \ \ j \neq k \tag{2.7}$$

3. Find C_{m+1} by computing the centroids of the training vectors assigned to each codeword:

$$W_{i,m+1} = \frac{1}{M_{i,m}} \sum_{V_j \in W_{i,m}} V_j \qquad i = 1,2,\ldots,N \tag{2.8}$$

where $M_{i,m}$ is the number of vectors assigned to $W_{j,m}$.

4. Compute the average distortion:

$$D_m = \frac{1}{N} \sum_{W \in C_m}^{N} D(V_j, W) \tag{2.9}$$

if D_m relative to D_{m-1} is less than ε, then stop; otherwise, go to step 2.

To obtain the initial codebook C_0, one possible approach is to select the first, or the evenly spaced N vectors as an initial codebook. Alternatively, one might use the splitting algorithm [8], where the centroids of the training set is calculated and split into two codewords. The LBG algorithm is applied to yield a codebook of two codewords. Each codeword is then split into two codevectors to yield a codebook of four codewords. This procedure is repeated until an N-level codebook is constructed.

Note that the LBG algorithm may converge to a local minimum. Furthermore, the solution is not unique and depends upon the initial codebook. Simulated annealing is a procedure which introduces randomness in each iteration of the LBG algorithm can be used to avoid the local minima. However, this procedure is computationally intensive. Recently, techniques based on neural networks and the pairwise nearest neighbor (PNN) algorithm has been reported as alternatives for codebook design [7]. In the neural network approach, the weights (which represent the codewords) between the neurons are adaptively adjusted after the presentation of each vector. The main advantage of neural network is that it converges to an asymptotic value faster than the LBG algorithm. In the PNN approach, each of the K

training vectors is considered as a separate cluster. We start with the K clusters and merge together the two clusters which result in the minimum increase in average distortion. This yields K-1 clusters. The merging process continues until only N clusters remain. PNN is faster than the LBG algorithm, however the LBG algorithm results in a codebook that satisfies the two conditions for optimality. In addition, it has been shown that for a practical codebook size and training set, the computational efficiency of the LBG algorithm is higher than the PNN algorithm [9].

2.3.3 Vector Quantization using an Universal Codebook

Vector quantization techniques can be broadly classified, with respect to training and codebook generation, as universal and adaptive. In this section, image coding using universal VQ is reviewed, while adaptive VQ is discussed in the next section. Universal VQ employs a fixed codebook generated using a large set of training vectors selected from different types of images. To ensure good image fidelity the codebook must be large, which in turn increases both the bit rate and the coding complexity. The codebook size can be reduced using techniques which exploit the local image statistics. Examples include classified VQ, predictive VQ, finite state VQ, multi-stage VQ, fast search VQ, address VQ, and fast search VQ.

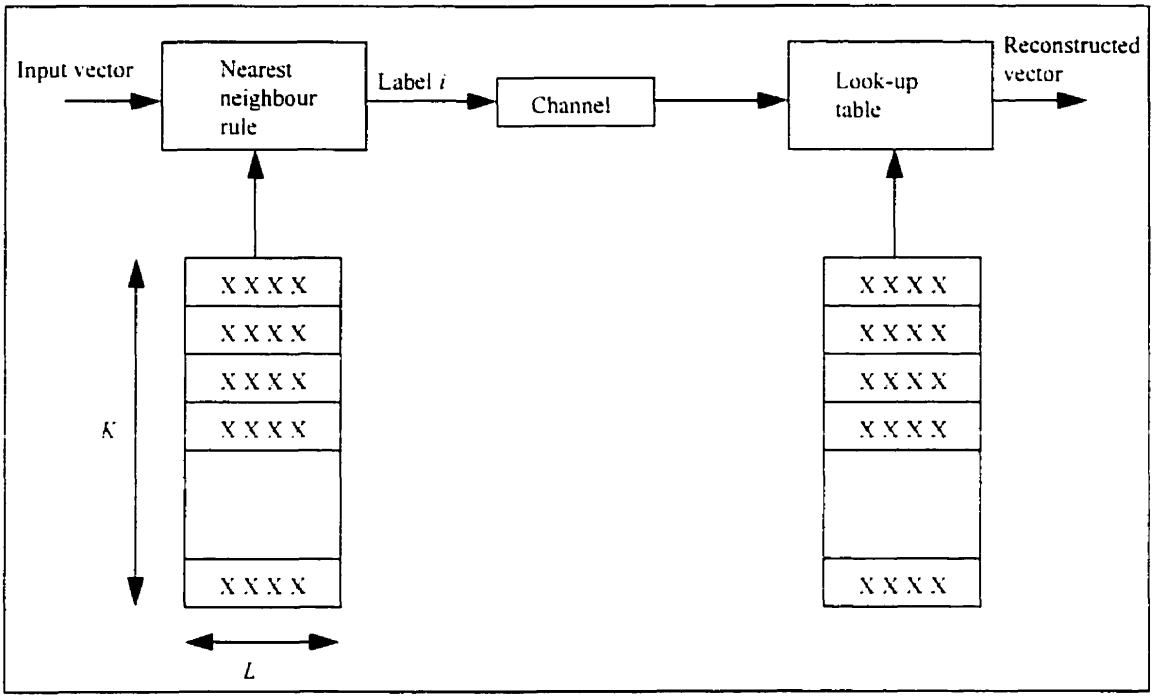


Figure 2.1: Vector quantization

2.3.4 Vector Quantization using an Adaptive Codebook

In adaptive VQ, the codebook is adapted in order to match the local image statistics. For example, a new codebook can be generated using the vectors of the input image as a training set. The new codebook is transmitted, followed by the labels corresponding to the vectors of the image. Goldberg *et al.* [13] have presented an adaptive VQ scheme for coding monochrome and color images. In their scheme, the image is partitioned into non-overlapping sub-images, and for each sub-image a separate 16-64 level codebook is created. These codebooks better match the local image statistics, however, the improvement in coding performance is achieved at the expense of the overhead incurred for transmitting the new codebooks. An alternative scheme, which reduces the overhead, is to update or replenish part of the codebook. For example, Gersho [14] have proposed a technique where the distortion of

each input vector is monitored, and if it is larger than a predetermined threshold, the codeword with the "largest time since use" is replaced by the input vector.

We note that in the above techniques, the major drawback is that the improvement in image quality is achieved at the expense of increasing the computational complexity.

2.4 Wavelets

Wavelet transform decomposes a signal into a weighted sum of basis functions called wavelets [23]. The unique feature of the wavelet transform is that the wavelets are all dilated and translated versions of a single function, the so called "mother wavelet". Mathematically, this is expressed as follows (one dimension):

$$\psi^{\alpha,\beta}(t) = |\alpha|^{-0.5}\psi(\frac{t-\beta}{\alpha}) \tag{2.10}$$

and by discretizing the values for α_0^m and $\beta=n\beta_0\alpha_0^m$, we obtain the following discrete wavelet decomposition for a function $g(.)$:

$$g = \sum c_{m,n}(g)\psi_{m,n} \tag{2.11}$$

where $\psi_{m,n}(t)=\alpha_0^{(-m/2)}\ \psi(\alpha_0^{-m}t-\beta_0)$. The usual choice is $\alpha_0=2$ and $\beta_0=1$, which results into dyadic grid. Because of the orthonormality of $\psi_{m,n}$ we get

$$c_{m,n}(g) = <\psi_{m,n}, g> \tag{2.12}$$

The wavelet coefficients can be calculated iteratively using a two channel filterbank [23]. A 2-D wavelet transform is implemented using a separable approach.

Figure 2.2 shows a 3-level wavelet decomposition of an image S^0 of size $X\times Y$ pixels. In the first level of decomposition, one low pass sub-image S^1 and three orientation selective

high pass sub-images ($W^{1,h}$, $W^{1,v}$, $W^{1,d}$) are created. In the second level, the low pass sub-image is further decomposed into one low pass and three high pass sub-images ($W^{2,h}$, $W^{2,v}$, $W^{2,d}$). This process is repeated on the lowpass subimage to form a higher level wavelet decomposition. The inverse wavelet transform is calculated recursively, where the higher resolution images are generated starting from the lower resolution sub-images.

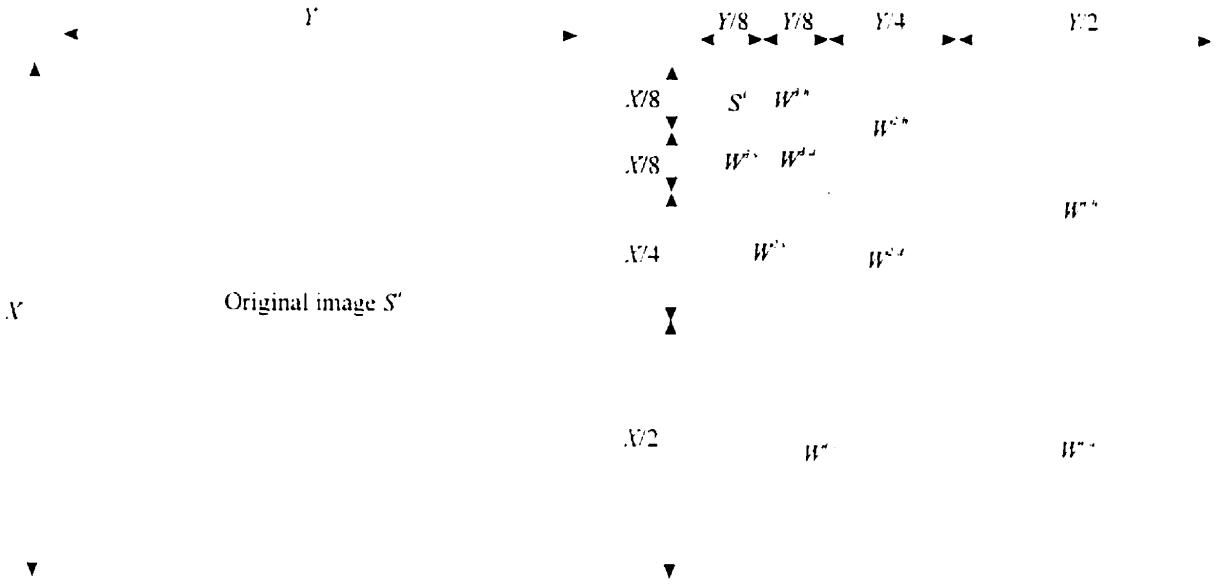


Figure 2.2: Representation of 2-dimensional wavelet transform.

2.5 JPEG Compression Standard

Recently, the International Standards Organization (ISO) has proposed a standard for image compression known as JPEG (Joint Picture Experts Group) [25]. JPEG provides a framework for compression of gray level and color images for a wide rage of applications. The JPEG standard has four modes of operations:

- Baseline sequential: each image is compressed in a single left-to-right, top-to bottom;
- Progressive coding: the image is encoded in multiple scans for applications in which transmission time is long and the user prefers to view the image building up in multiple coarse-to-clear passes.
- Hierarchical coding: the image is encoded at multiple resolutions, so that lower resolution versions may be accessed without the need to decompress the full resolution image.
- Lossless compression: the image is encoded using DPCM-based losless approach.

The baseline sequential is a DCT-based algorithm. The progressive and hierarchical modes use a modified version of the baseline algorithm. The DPCM-based lossless algorithm is independent of the DCT. We now present a brief description of the baseline sequential mode.

In the baseline mode, image compression is carried out in three steps: DCT computation, quantization and variable length coding. The image to be coded is first partitioned into non-overlapping blocks of size 8×8 pixels as shown in Figure 2.3. In order to decrease the average energy of the image pixels, each pixel is level-shifted by 2^{n-1}, where n is the number of bits required to represent each pixel value (for example, n is equal to 8 for images with 256 gray levels). Each block then undergoes a 2-dimensional DCT. The DCT coefficients are quantized using a visually adapted quantization table. The DC coefficients are differentially encoded, while the AC coefficients are scanned along the zigzag lines shown in Figure 2.4 and encoded using an entropy encoder. The decoder is simply the inverse of the encoder.

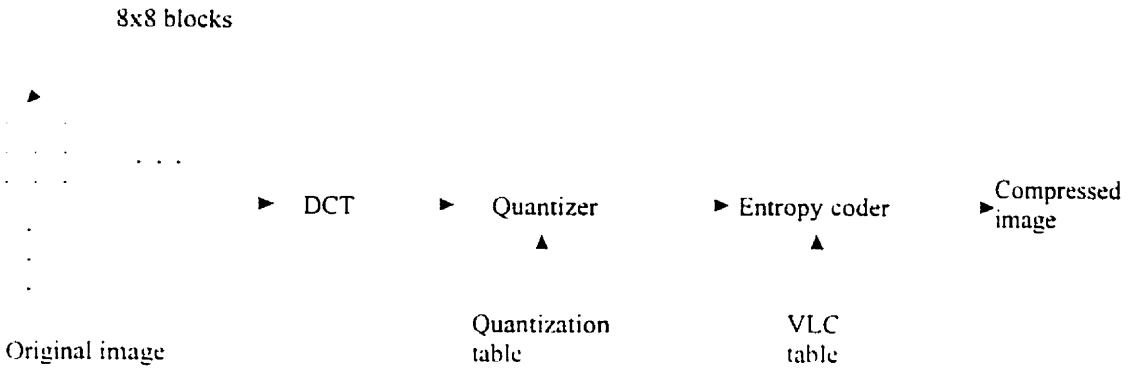


Figure 2.3: Baseline JPEG encoder.

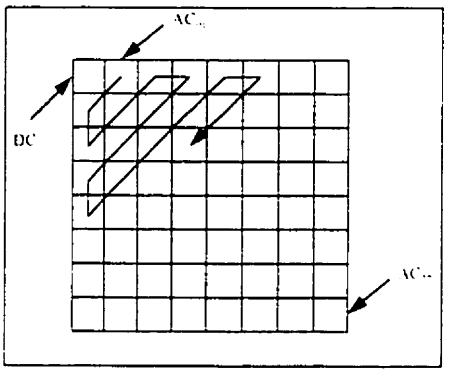


Figure 2.4: Zigzag scanning of the DCT coefficients

2.6 MPEG Compression Standards

Recently, the International Standards Organization (ISO) have proposed standards for video compression known as MPEG (Moving Picture Experts Group [26]. In addition, the Consultative Committee on International Telephony and Telegraphy (CCITT) has recommended a standard for videotelephony called the H.261 at p×64 Kbits/s [27]. In this section we review the MPEG video compression algorithm.

In the MPEG video compression standard, a block based motion estimation/compensation is employed to remove the interframe correlation and discrete

cosine transform (DCT) for the removal of the intraframe correlation. Block diagrams of the MPEG encoder is shown in Figure 2.5. Here, a group of pictures approach is used instead of the frame by frame coding. A group of pictures is typically a combination of one or two intra-pictures (I), predicted picture (P), and the rest of bi-directional pictures (B) as shown in Figure 2.6. The I-frames provide random access points and are also used as a reference for P-frames. The I-frames are coded using DCT on 8×8 blocks. The DC coefficient is differentially encoded using variable length codes (VLC). The AC coefficients are zigzag scanned as shown in Figure 2.4 and ordered into {RUNLENGTH, AMPLITUDE} pairs. A variable length coder (VLC) is used to encode each pair. The P- and B-frames are decomposed into 16× 16 blocks and the motion vector for each block is calculated. The motion vectors are also variable length coded and transmitted. The motion compensated difference frame is partitioned into 8× 8 blocks which then undergo a 2-dimensional DCT. The DC and AC coefficients are quantized, ordered along the zigzag scan line into {RUNLENGTH, AMPLITUDE} pairs and coded using a VLC.

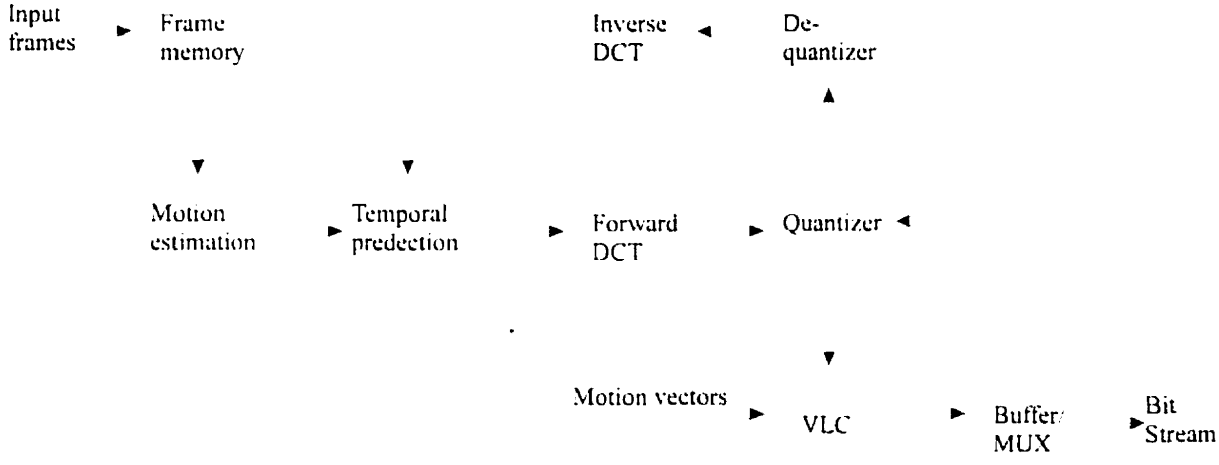


Figure 2.5: MPEG video coder

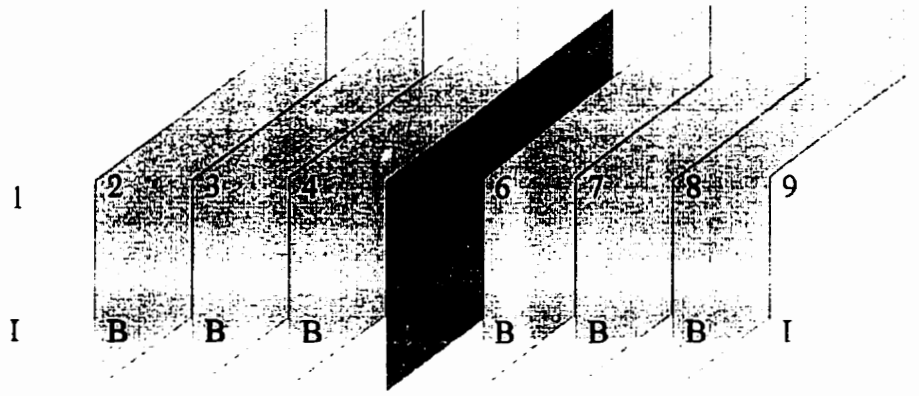


Figure 2.6: Example of a group of pictures used in MPEG.

2.7 MPEG–4

In MPEG-1 and MPEG-2 the video information is assumed to be rectangular of fixed size displayed at fixed interval. In MPEG-4 the concept of Video Object (VO), Video Object Layer (VOL) and Video Object Plane (VOP) have been introduced. A VO can be viewed as the MPEG-4 equivalent of a GOP in the MPEG-1 and -2 standards. VOP represents instances of a given VO. The VOP can have arbitrary shape. For example, the video frame shown in Figure 2.7a, can be segmented into two VOP's: VOP_1 for the background and VOP_2 for the foreground, as shown in Figure 2.7a and 2.7b, respectively. At the encoder side, together with the VOP, composition information is sent to indicate where and when each VOP is to be displayed. At the decoder side the user may be allowed to change the composition of the scene displayed by interacting on the composition information. The shape, motion and texture information of the VOP's belonging to the same VO is encoded into a separate video object layer (VOL).

Figure 2.7: Example of VOP. (a) One frame from a scene, (b) VOP_1, and (c) VOP_2.

For each VO, the shape, motion, and texture information of VOP's are coded. The shape information is referred to as alpha planes. The techniques to be adopted by the MPEG-4 will provide lossless coding of alpha planes and lossy coding of shapes and transparency information; thus, allowing for tradeoffs between bit rate and accuracy of shape representation. Furthermore, intra-and inter-shape coding functionalities employing motion compensated shape prediction is envisioned so as to allow efficient and random access operations as well as efficient compression of shape and transparency information for diverse applications.

Temporal redundancies between video content in separate VOP's within a VO are exploited using block based motion estimation and compensation. The intra VOP's as well

as errors after motion compensated prediction, are coded using DCT on 8×8 blocks, in a manner similar to hat employed in MPEG. The compressed alpha plane, motion vectors, and DCT coded information are multiplexed into a VOL bit stream by coding the shape information followed by motion and texture coded data.

2.8 Summary

In this chapter we have presented a review of image and video compression techniques. First, we have explained the concepts of lossless image/video data compression. We have then presented an overview of lossy compression techniques including predictive coding [3]-[5], transform coding [6], subband coding, fractal coding [18]-[19], and model-based compression techniques [17]. We have emphasized the review of vector quantization (VQ) [7]-[15] and wavelet [20]-[24] methods as the focus of this thesis is essentially based on VQ and wavelet-VQ. Finally, we have presented a review of JPEG, MPEG, and MPEG-4 compression standards [25]-[30].

3

Review of Image/Video Indexing

We recall from chapter 1 that, one of the key features required in a visual database is efficient indexing to enable fast access to the images/video in the database. In order to overcome the limitations of keyword based systems and classification/interpretation techniques, indexing techniques based on automatic feature extraction have been reported in the literature. Typically, indices based on feature vectors derived from images and video are used as indices to search and retrieve the image(s)/video of interest. In this chapter we present a detailed review of image and video indexing techniques in the uncompressed domain [151].

This chapter is organized as follows. An overview of a visual storage and retrieval system is presented in section 3.1. Image and video indexing in pixel (uncompressed) domain are reviewed in sections 3.2 and 3.3, respectively. Compressed domain image and video indexing techniques are reviewed in sections 3.4 and 3.5, respectively. This is followed by a review of MPEG-7 standards in section 3.6. Finally, the summary is presented in section 3.7.

3.1 Visual Storage and Retrieval System

Visual database access has two main components: storage and retrieval. In the storage process, images and video are processed to extract features which describe their semantics. The extracted features are then represented, organized and stored in the database. In the retrieval process, the system analyzes a query, extracts the appropriate feature vector and a search process is performed. The search process is carried out by computing the "similarity" between the feature vector of the query and those of the candidate images and video stored in the database. The retrieved images and video are presented to the user in the descending order of the similarity to the query.

Several image and video database systems have been proposed in the literature [34]-[37]. An architecture of a generic image/video database system is shown in Figure 3.1. It consists of the user interface, content-based retrieval, organization, and database management modules. A functional description of each module is presented below.

1. *User Interface*: In visual information systems, user interaction plays an important role in almost all of its functions (e.g., semi-automatic and manual feature extraction, navigation, selection, and refinement). The user interface consists of a query processor and a browser to provide the interactive graphical tools and mechanisms for querying and browsing the database, respectively. The query processor provides the means to retrieve images and video using a variety of methods and interfaces. A query can range from a simple keyword-based query to a complex one where the user specifies a sketch or an object track. In contrast to textual database systems, image and video databases are required to evaluate properties of the data specified in a query. For example, to retrieve all images

similar to an image based on color, the color attributes (e.g., color histogram) of the input image has to be calculated. After retrieving all similar images/video. the browser is used to display the results. The browser allows users to navigate through the database visually and to further refine the search.

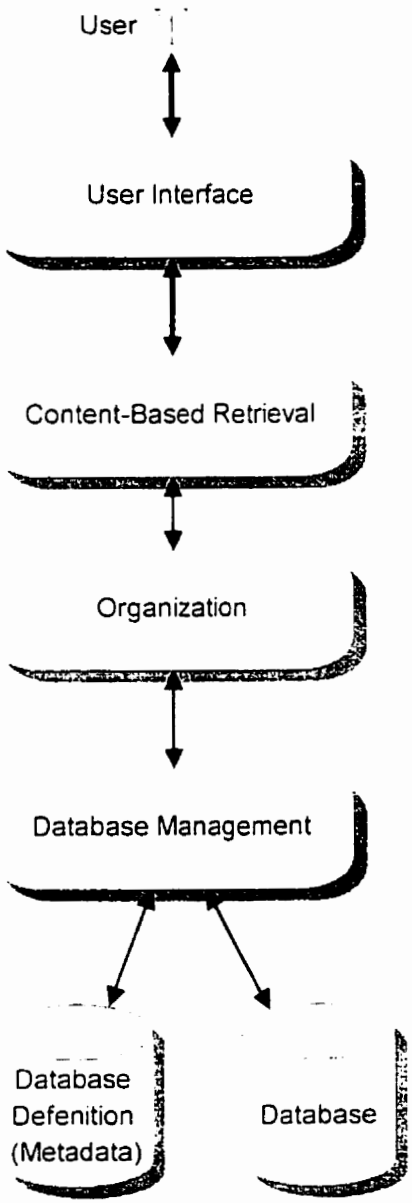


Figure 3.1: Storage and retrieval of images and video.

1. *Content-based Retrieval Module*: As shown in Figure 3.2, the content-based retrieval module consists of the following:

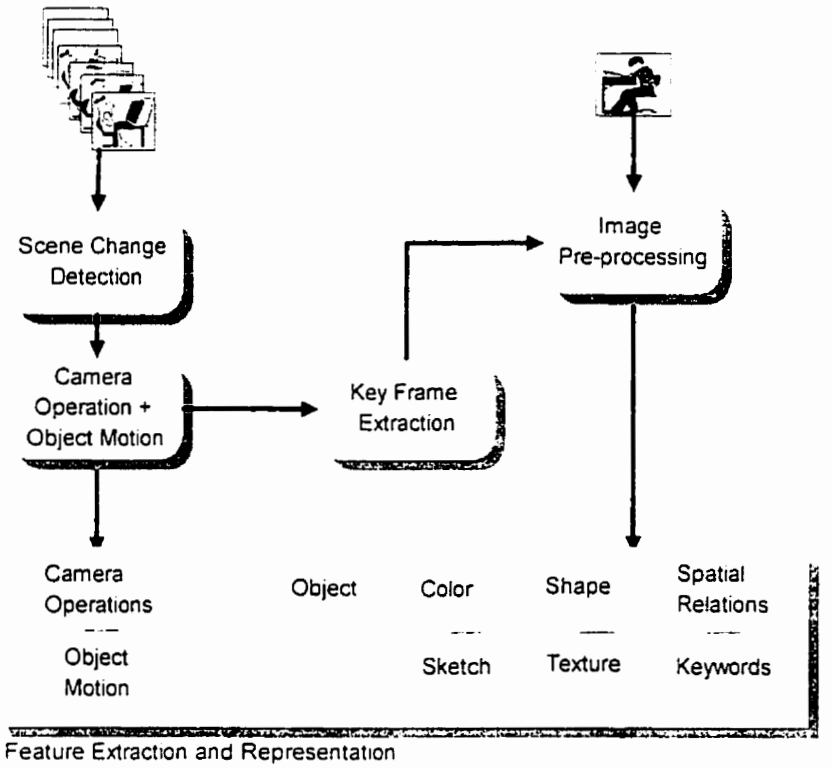


Figure 3.2: Content-Based Retrieval Module

- *Scene Change Detection*: Prior to storage in a database, a video sequence is first segmented into elemental scenes called shots. A shot is a sequence of frames generated during a continuous operation and therefore represents a continuous action in time or space [127]. The purpose of the segmentation process is to partition the video stream into a set of meaningful and manageable segments which then serve as basic units for indexing. Once a video sequence is segmented into shots, a set of key frames is then

selected to represent the shot [111],[112]. Each shot is represented using spatial or temporal features. The spatial features refer to the spatial content of the key frames of a shot, while the temporal features refer the temporal content of the shot. The key frames of a shot are fed to the image pre-processing stage in order to generate the spatial features of the shot, while the shots are analyzed in order to extract the temporal features.

- *Image Pre-processing*: The image is first processed in order to extract the features which describe its contents. The processing might involve decompression, enhancement, filtering, normalization, segmentation, and object identification. If the input image is in the compressed form, decompression is required to facilitate execution of the pixel domain algorithms. The output of the image pre-processing stage is typically a collection of objects and regions of interest.

- *Feature Extraction and Representation*: In this stage, the semantics of image/video content are extracted and represented. The basic philosophy is the transformation of the data-rich pixel representation of image and video space to compact and semantic-rich representation of visual characteristics (color, texture, ..., etc.) in feature space. Features (of the objects, regions, and/or the whole image) such as texture, color, etc. are used to describe the content of a still image. For video, the spatial features are generated using still image techniques [111]-[112], while the temporal features are extracted based on motion and/or camera operations within the shot [122]. Image and video features can be classified into primitive and logical features [39]. Primitive features such as color, shape centroids, etc. are quantitative in nature and can be extracted automatically or semi-automatically. Logical features are qualitative in nature and provide abstract

representations of visual data at various levels of detail. Typically, logical features are extracted manually. One or more features can be used in a specific application. For example, in a satellite information system, the texture features are important, while shape and color features are more important in trademark registration systems. Once the features have been extracted, the textual, numerical, alphanumerical, etc., index keys are derived.

3. *Organization*: Efficient query processing necessitates the organization of image/video indices such that efficient search strategies can be used. We note that image/video indices are approximately represented, may have inter-related multiple attributes and may not have an embedded order [125]. Therefore, conventional indexing structures like B-tree, hashing, etc., cannot be used for the organization of image/video indices. Flexible data structures should be used in order to facilitate storage/retrieval in visual information systems. Structures such as R-tree family [128], R^*-tree [129], quad-tree [130] and grid file [131] are commonly used. Each structure has its advantages and disadvantages; some have limited domains and some can be used concurrently with others. Niu *et al.* [35] have discussed some issues concerning novel indexing structures for image retrieval. Ahanger *et al.* [36] have also presented a review of indexing structures for video.
4. *Database Management Module*: The database management module provides internal level physical storage structure and access path to the database. The database management module has the following characteristics: (i) provides insulation between programs and data, (ii) provides users with a conceptual representation of the data, (iii) supports multiple views of the data, and (iv) ensures data consistency.

3.2 Image Indexing In Pixel Domain

Recently, image indexing techniques based on color, texture, sketch, shape and spatial relationships have been reported in the literature [40]-[45]. Color, texture and shape allow users to retrieve images that contain objects which have similar attributes. Sketch allows images to be retrieved based on a rough outline of the object(s) in a query image. Spatial relationships facilitate retrievals based on features such as adjacency, overlap, and containment among the objects in a scene. We now present a review of image indexing techniques.

3.2.1 Color

Color is an important attribute for image representation. The color distribution of an image is typically represented using the image histogram. The histogram of an image f_n is an N-dimensional vector $\{H(f_n, i): i=1, 2, \ldots, N\}$, where N is the number of colors (bins) and $H(f_n, i)$ is the number of pixels having color i. Histograms are invariant to image rotation, translation and viewing axis [41]. In image indexing using histogram [40]-[43], the histograms are the feature vectors used as image indices. We note that a similarity measure is used in the histogram space to measure the similarity of two images.

Given a pair of images, f_n and f_m, the similarity between the two images may be measured using the normalized intersection of their histograms

$$\frac{\sum_{i=1}^{N} \min(H(f_n, i), H(f_m, i))}{\sum_{i=1}^{N} H(f_m, i)} \tag{3.1}$$

It has been shown that this metric eq. (3.1) is fairly insensitive to changes in image resolution, histogram size, occlusion, depth and view point [40]. However, histogram intersection does not consider the perceptual similarity between the different bins. A metric which takes into account the similarity between the bins is defined as follows :

$$\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\left[H(f_n,i)-H(f_m,i)\right]\left[H(f_n,j)-H(f_m,j)\right] \tag{3.2}$$

where the weight a_{ij} denotes the cross correlation between the colors corresponding to bins i and j. We note that the metric in eq. (3.2) has a higher computational complexity than the histogram intersection (eq. (3.1)). However, it is closer to human judgment of color similarity.

The color histogram requires additional storage space and a large amount of processing. For an image of size $X \times Y$, the histogram calculation requires $O(XY)$ additions and $O(XY)$ increments. In addition, $O(N)$ operations are required to compare a pair of histograms. To decrease the computational complexity, the number of bins should be reduced. The first approach to reduce the number of bins is to represent the red (R), green (G) and blue (B) components using the RG, BY, and WB color axes as follows [40]:

$$\begin{aligned} RG &= R - G \\ BY &= 2 \times B - R - G \\ WB &= R + G + B \end{aligned} \tag{3.3}$$

This representation allows the intensity (WB) to be more coarsely quantized than RG and BY. A histogram of 2048 bins is obtained if RG and BY are divided into 16 sections, while WB is divided into 8 sections.

The second approach to reducing the number of bins is based on the observation that a small number of bins capture the majority of pixel counts in a histogram.

Therefore, only the bins with the largest counts are compared. Experiments have shown that this approach results in a marginal degradation in performance [41,42].

An alternative approach to reducing the computational complexity in color indexing, is to use the dominant features of a histogram. A color distribution of an image is interpreted as a probability distribution which can be characterized by its moments [44]. If the first three moments of each color component are used, only 9 floating point numbers are required to represent each index. The similarity metric is a weighted sum of the absolute differences between the corresponding moments. This approach outperforms the approaches based on reducing the number of bins in terms of storage requirements, retrieval speed and robustness [44]. However, the use of low order moments is sensitive to changes in illumination.

Another approach to reduce the computational complexity is to use a lower dimensional histogram or a lower complexity metric to filter a large fraction of the database. A higher-dimensional histogram or a higher complexity metric can then be applied to the small set of retrieved images. Vellaikal *et al.* [48] have proposed to represent the color histogram at different resolutions. Here, the histogram is decomposed using the three-dimensional Haar wavelet basis functions. In the search process, the top W wavelet coefficients of the histogram of the query image are compared. Hafner *et al.* [53] have proposed the use of a lower dimensional and computationally simple distance measure based on (3.2). This technique not only reduces the complexity of the search process, but can also be implemented in a hierarchical manner, where a finer match can be obtained by increasing the number of coefficients employed in the search process.

We note that in the previous techniques a histogram describes the entire image content, without taking into account the location of the colors. This may result in

unsatisfactory retrievals (e. g., false positives). For example, an image with a white car on the left might be matched to an image with white birds on the right. This problem can be eliminated by incorporating spatial information in the content representation. Gong *et al.* [42] have proposed the use of local histograms. Here, an image is partitioned into 9 (3×3) sub-images. For each sub-image, the color histogram is generated. The content of the image is represented by the histogram of entire image and the histograms of the sub-images. This technique captures the locality of colors in an image. Stricker *et al.* [49] have proposed a technique where the image is partitioned into 5 overlapping fuzzy regions. The histogram for each region is calculated and is represented by the first 3 moments. The 15 moments are used to represent the image. The fuzziness of the regions makes the feature vector insensitive to small rotations of an image. In addition, a similarity function which exploits the spatial arrangement of the 5 regions is employed resulting in invariance of retrieval results with respect to rotations of 90 degrees around the center of an image.

Including spatial information in the color representation of an image will not only improve the retrieval rate but also allow user queries based on the color of a sub-image. For example, queries such as "retrieve all images with white birds on the right" could be answered. However, this technique requires the use of efficient segmentation and representation of the sub-images.

3.2.2 Texture

Texture is an important feature of a visible surface where repetition or quasi-repetition of fundamental pattern occurs. Texture features such as contrast, uniformity, coarseness, roughness, regularity, frequency, density and directionality provide significant information

for scene interpretation and image classification [54]. Texture modeling and classification techniques can be grouped into structural, statistical and spectral methods. In this section, a survey of texture modeling and classification techniques that have been employed in image databases is presented.

Picard *et al.* [59] have presented a technique based on Wold decomposition. If an image is assumed to be a homogeneous and regular 2-D random field, then the 2-D Wold like decomposition is a superposition of three orthogonal components: a purely-indeterministic field *u(n,m)*, a generalized evanescent field *v(n,m)* and a harmonic field *w(n,m)*. This model provides a description of textures in terms of periodicity, directionality and randomness. Each one of these attributes is associated with the prominence of a different component. The conspicuous components in periodic, directional and less structured textures are *w(n,m)*, *v(n,m)* and *u(n,m)*, respectively [59]. Simulations on about a 1000 test images (cropped from 112 Bordatz texture images [62]) have shown that the Wold-based parameters are closer to human judgment of texture similarity than the principal components approach.

Tamura *et al.* [61] have proposed the use of coarseness, contrast and directionality as texture features. *Coarseness* is a measure of the granularity of the texture. *Contrast* can be measured based on the gray-level distribution. *Directionality* determines whether or not the image has a certain direction. A modified set of the Tamura features has been used in the QBIC project [43].

Zhang *et al.* [66] have proposed a technique based on features derived from a multiresolution representation of the texture image. Here, the image pattern is decomposed into a set of different resolution sub-images using multiresolution simultaneous autoregressive model (MR-SAR). The MR-SAR parameters associated with each sub-image

are used to construct the index. The combination of MR-SAR model with Tamura's coarseness features and gray level histogram gives better performance than MR-SAR [66]. However, the improvement in retrieval rates is at the expense of increasing the size of the feature vector which increases the complexity of indexing and searching.

Retrieval by texture is useful when the user is interested in retrieving texture images which are similar to the query image. However, the use of texture features in a general visual database system requires texture segmentation (which remains a challenging problem) and the combination of texture features with spatial information.

3.2.3 Sketch

Another approach to describing the content of an image is by using a sketch. A sketch is an abstract image which contains the outline of objects. In this technique, for each image to be stored in the database, a sketch image is generated and stored. Typically, a sketch is created by using edge detection, thinning and shrinking algorithms. The sketch is used as a key to retrieve the desired images from the database. The similarity of two images is measured by using the similarity of their sketches.

A technique for sketch based image retrieval has been proposed by Kato *et al.*[80] and is implemented in the QBIC [43]. Each color image is converted into luminance and chrominance components. An edge operator is applied to the luminance component to compute the binary edge image. The edge image is reduced to 64×64 pixels and the reduced image is thinned. Query processing is performed by matching the user drawn sketch to the sketches stored in the database. The matching process between the query image and a

candidate image is executed as follows. The query sketch is first reduced to 64×64 pixels and divided into blocks of 8×8 pixels. Each block in the query image is correlated with the blocks within a search area in the candidate image. The size of the search area is 16×16 pixels and is centered around the block.

The main disadvantage of this approach is that it is orientation and scale dependent. Similar images with different orientation or scale will not be retrieved when compared with the query image. This problem can be eliminated by using sophisticated edge representation and matching algorithms.

3.2.4 Shape

The shape of an object refers to its profile and physical structure. Shape features are fundamental to systems such as medical image databases, where the color and textures of objects are similar. In general, shape features can be represented using traditional shape analysis such as invariant moments, Fourier descriptors, autoregressive models and geometry attributes [69]- [71]. However, in image storage and retrieval applications, shape features can be classified into global and local features.

Global features are the properties derived from the entire shape. Examples of global shape features are roundness or circularity, central moments, eccentricity and major axis orientation. In general, global features are robust to distortion, however, they cannot handle occluded shapes. Eakins *et al.* [78] have developed the ARTISAN shape retrieval system (Automatic Retrieval of Trademark Images by Shape Analysis). Edge detection techniques are applied to the trade mark images to derive a set of region boundaries. The boundaries are

approximated as a sequence of straight line and circular arc segments. The boundaries are then grouped into families using proximity and shape similarity criteria. Eight global features are used, namely: circularity, aspect ratio, discontinuity angle irregularity, length irregularity, complexity, right anglenness, sharpness, directedness.

Local features are those derived by partial processing of a shape and do not depend on the entire shape. Examples of local features are size and orientation of consecutive boundary segments [76], points of curvature, corners and turning angle. Gary *et al.* [75] have presented a shape similarity technique based on local boundary features encoded as multi-dimensional points. Although local features can be used for occluded shapes, they are noise sensitive. Ang *et al.* [79] have presented multidimensional feature measures of object shapes and feature blobs for retrieval of ceramic artifacts. Object shape is represented by boundary eccentricity and region compactness, moment and convexity. High detailed regions are represented by number of blobs, dispersion of blobs, and central moment of blobs, and total blob size.

Retrieval by shape similarity is a difficult problem because of the lack of mathematically exact definition of shape similarity which accounts for the various semantic qualities that humans assign to shapes. The majority of such techniques have been aimed at retrieving simple 2-D image objects capable of being represented by a single shape boundary. Recently, Scassellati *et al.* [74] have studied shape similarity using algebraic moments, spline curve distances, cumulative turning angle, sign of curvature and Hausdroff distance and compared it to human perception. It has been shown [74] that turning angle, sign of curvature and algebraic moments most closely match human judgment.

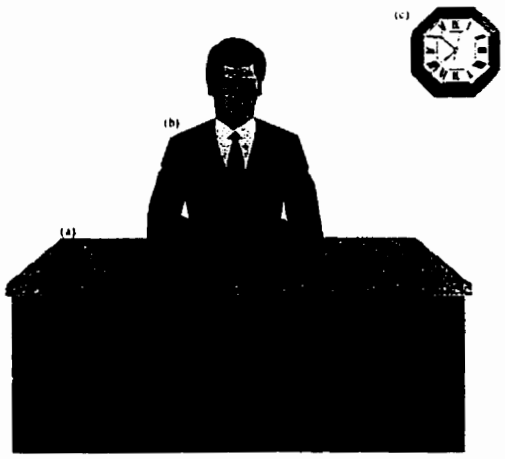
3.2.5 Spatial Relationships

In this technique, objects and the spatial relationships among objects in an image are used to represent the content of an image. First, each image is converted into a symbolic picture. The symbolic pictures are then encoded typically using 2-D strings which are stored in the database [81]-[87]. Queries are expressed in the same 2-D string notation. The problem of image retrieval thus becomes a problem of 2-D sequence matching.

The basic algorithm for image indexing using spatial relationships was presented by Chang *et al.* [81]. To start with, the objects in an image are segmented and recognized. The image is converted into a symbolic picture, where the objects are represented using a set of symbols S. The position of an object in the symbolic picture is determined by the object's centroid. For example, the symbolic picture corresponding to the image in Figure 3.3a is shown Figure 3.3b. The relationships among the objects in an image are expressed using the set of operators $A=\{<,=,:\}$. The symbol "<" denotes the left-right or below-above spatial relationship. The "=" stands for "at the same spatial location as" and the symbol ":" denotes the relation "in the same set as". A 2-D string over S is defined as

$$(x_1y_1x_2y_2\ldots x_ny_n, x_{p(1)}z_1x_{p(2)}z_2\ldots x_{p(n)}z_n) \qquad (3.4)$$

where $x_1x_2x_3 \ldots x_n$ is a 1-D string over S ($n\geq 0$ and $x_i\in S$), $p(.)$ is a permutation over $\{1, \ldots, n\}$. $y_1y_2y_3 \ldots y_n$ and $z_1z_2z_3 \ldots z_n$ are 1-D strings over A. For example, in Figure 3.3a, a and b are to the left of c, c is above b and b is above a, the image can be represented using the 2-D string $\{a=b<c, a<b<c\}$.



a)

c

b

a

b)

Figure 3.3: Example of image representation using 2-D string. a) An image containing 3 main objects. b) Symbolic picture which represents the image (a).

Matching 2-D strings is based on a ranking scheme for the object symbols in the strings. The matching algorithm is simple, however, for images with large number of objects, (i) the 2-D string representation is complex, and (ii) the spatial operators ($A=\{<,=,:\}$) are not sufficient to give a complete description of the spatial relationships among the objects.

3.3 Video Indexing in Pixel Domain

A video sequence is a series of images sequentially ordered in time. Prior to storage in a database, a video stream is segmented into elementary units (shots) to be identified and indexed. Since video data consists of still images (frames), all the techniques presented in section 2.2 are applicable to the individual frames of a video sequence (spatial features). In addition, video has temporal properties such as motion, camera operations, sequential composition and interframe relationships. In this section, we present a review of video segmentation methods followed by video indexing using spatial and temporal features.

3.3.1 Scene Change Detection

A number of algorithms for scene change detection in both the uncompressed and compressed domains have been reported in the literature [94]-[115]. The different algorithms in the uncompressed domain can be broadly classified into five categories: template matching, histogram-based, block-based, twin-comparison, and model-based techniques.

3.3.1.1 Intensity/Color Template Matching

Scene change can be detected by emphasizing the spatial similarity between two frames [94]-[46]. The simplest way to measure the spatial similarity between two frames f_m and f_n is using template matching, where each pixel at the spatial location (i,j) in f_m is compared with the pixel at the same location in f_n. Typically, the difference magnitude, $D(f_m,f_n,i,j)$, of f_m and f_n is used for comparison where

$$D(f_m, f_n, i, j) = \begin{cases} |P(f_m, I, i, j) - P(f_n, I, i, j)| & \text{for gray level images} \\ \sum_{l=1}^{3} |P(f_m, C_l, i, j) - P(f_m, C_l, i, j)| & \text{for color images} \end{cases} \quad (3.5)$$

where $P(f_m, I, i, j)$ is the intensity of the pixel at (i,j) and $P(f_m, C_1, i, j)$, $P(f_m, C_2, i, j)$, $P(f_m, C_3, i, j)$ are the color components C_1, C_2, and C_3 of the pixel (i,j) respectively. For example, in case of using the RGB color coordinate system, C_1, C_2 and C_3 are equal to R, G and B, respectively. Nagasaka *et al.* [46] have proposed the use of the sum of the difference magnitude

$$S_1(f_m, f_n) = \sum_{i=1}^{N} \sum_{j=1}^{N} D(f_m, f_n, i, j) \quad (3.6)$$

A scene change is declared whenever $S_1(f_m, f_n)$ exceeds a prespecified threshold.

Zhang *et al.* [95] have presented an algorithm based on the number of changed pixels. A pixel is changed if $D(f_m, f_n, i, j)$ is greater than a certain threshold. A cut is detected if the percentage of the changed pixels is greater than a threshold.

We note that, using the previous metrics, it is difficult to distinguish between a small change in a large area and a large change in a small area. Therefore, template matching methods are sensitive to noise, object motion and camera operations (e.g. panning and zooming) since they result in false detections.

3.3.1.2 Histogram Based Techniques

We recall from section 2.1, that the intensity/color histogram of a gray/color image f is an N-dimensional vector $\{H(f,i): i=1, 2, \ldots, N\}$ where N is the number of levels/colors and $H(f,i)$ is the number of pixels of level/color i in the image f. The rationale behind histogram based approaches is that two frames that exhibit minor changes in the background and object

content will also show insignificant variations in their intensity/color distributions. In addition, histograms are invariant to image rotation and change slowly under the variations of viewing angle, scale and occlusion [41]. Hence, this technique is less sensitive to camera operations and object motion compared to template matching based techniques.

Tonomura [97] has proposed a technique based on the gray level histogram difference. Let the histograms of frames f_m and f_n be denoted by $H(f_m, i)$ and $H(f_n, i)$, respectively. The sum of the histograms difference magnitude is defined as:

$$S_2(f_m, f_n) = \sum_{i=1}^{N} |H(f_m, i) - H(f_n, i)| \tag{3.7}$$

A cut is declared if $S_2(f_m, f_n)$ is greater than a threshold.

Histogram based techniques tend to lose scene changes which have small variations in their intensity distribution [98]. In addition, histogram comparisons may not reflect the content difference [100]. Hence, histogram based techniques are not necessarily superior to template matching approaches.

3.3.1.3 Block Based Techniques

We note that the techniques presented in sections 3.1.1.1 and 3.1.1.2 use global attributes of images such as point by point differences, and intensity or color histograms. Block based techniques [100]-[101] use local attributes to reduce the effect of noise and camera flashes. Here, each frame f_m is partitioned into a set of r blocks. Rather than comparing a pair of frames, every subframe in f_m is compared with the corresponding subframe in f_n. The similarity between f_m and f_n is measured using

$$S_4(f_m, f_n) = \sum_{i=1}^{r} C_i \times S_p(f_m, f_n, i) \tag{3.8}$$

where C_i is a predetermined weighting factor and $S_p(f_m,f_n,i)$ is a partial match obtained by comparing the ith region in f_m and f_n. Kasturi *et al.* [94] have presented a metric based on statistical characteristics of the intensities of subimages. Corresponding blocks in two frames are compared using a likelihood ratio,

$$\frac{\left(\frac{\mu_{m,i} + \mu_{n,i}}{2} + \frac{\sigma_{m,i}^2 - \sigma_{n,i}^2}{2}\right)^2}{\sigma_{m,i}^2 \times \sigma_{n,i}^2} \tag{3.9}$$

where $\mu_{m,i}$ and $\sigma_{m,i}^2$ are the mean and variance, respectively, of block i in frame f_m. If the likelihood ratio is greater than a threshold, $S_p(f_m,f_n,i)$ is set to 1. Otherwise, $S_p(f_m,f_n,i)$ is set to 0. A scene change is declared whenever the number of changed blocks is large enough (i.e., whenever $S_4(f_m,f_n)$ is greater than a given threshold and C_i is 1 for all i). Compared to the intensity/color template matching, this approach reduces the number of over detected (incorrectly) camera breaks. This reduction is a result of the increased tolerance to slow camera and object movements. However, cuts may be misdetected between two frames that have similar pixel values, but different density functions.

Video segmentation is the identification of two types of segment boundaries (abrupt changes and gradual transitions) which take place over a sequence of frames. The previous techniques are based on a single threshold and lack the power of detecting gradual scene changes, since the frame to frame difference in a gradual transition is smaller than the threshold. Lowering the threshold results in both false detections and misdetections. We now review a technique for the detection of gradual scene changes.

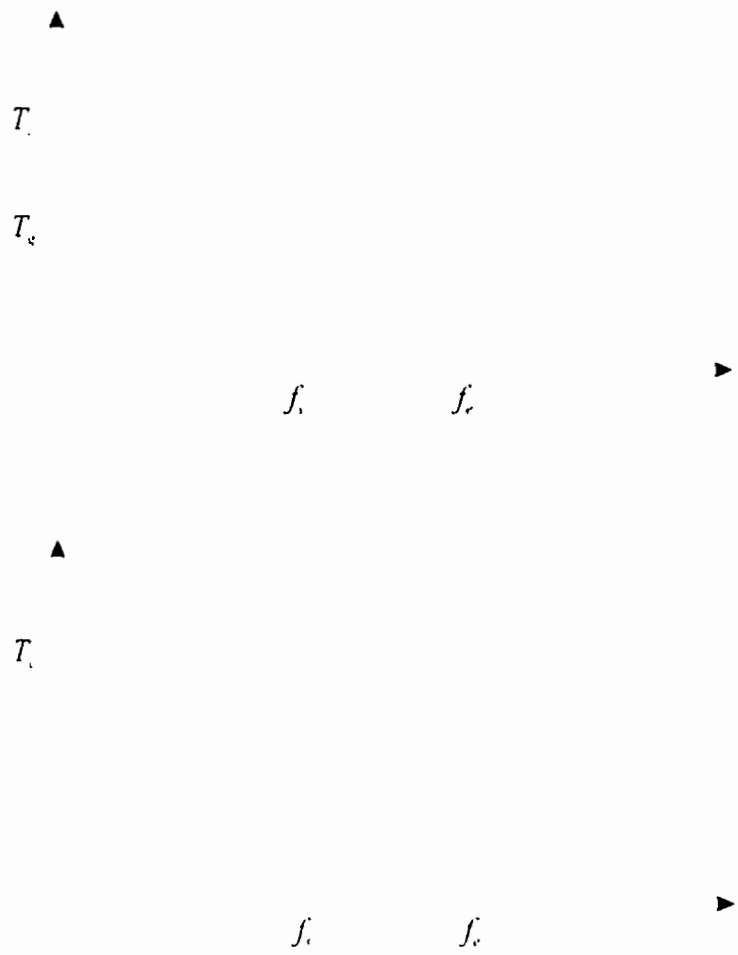


Figure 3.4: Twin-comparison.

3.3.1.4 Twin-Comparison

Twin-comparison [95] has been proposed for the detection of gradual scene changes using a dual threshold. In the first pass, a high threshold T_c is employed to detect abrupt scene changes. In the second pass, a reduced threshold T_g is used to identify the potential starting frame f_s of a transition as shown in Figure 3.4. Once f_s is identified, it is compared with subsequent frames, measuring the accumulated difference instead of the frame to frame difference. The end frame f_e of the transition is detected when the difference between successive frames decreases to less than T_g while the accumulated difference becomes larger

than T_c. If the consecutive frame difference falls below T_g before the accumulated difference exceeds T_c, then the potential starting point f_g is dropped, and the search continues for other gradual transitions. Although the twin-comparison approach is effective in detecting gradual scene changes, however the type of scene change (fade, wipe, ... etc.) cannot be identified.

3.3.1.5 Model-Based Segmentation

In a video sequence, gradual transition from a scene to another is the result of the editing process. Editing has direct influence on how the viewers respond to the video material, their interpretation, and their emotional reaction. Film editing is to be considered as not only the glue between shots, but also an essential contribution to the meaning conveyed by the video. Hence, it is not only important to identify the transition position, but also the type of the transition.

In model-based techniques, the problem of video segmentation is viewed as the process of locating the edit boundaries within the video sequence. Here, different edit types, such as cuts, translate, wipes, fades and dissolves are modeled by mathematical functions. Hampapur *et al.* [114] have presented a model for the video edit. Let $S_1(x,y,t)$ and $S_2(x,y,t)$ be two shots that are being edited, and $S(x,y,t)$ the edited shot. All the chromatic processes can be described as a linear pixel intensity manipulation:

$$S(x,y,t) = S_1(x,y,t)(1-\frac{t}{l_1}) + S_2(x,y,t)(\frac{t}{l_2}) \qquad (3.10)$$

where l_1, l_2 are the length (in number of frames) for which the scaling of each of the two shots lasts. This technique is efficient for detecting chromatic edits which results from scaling the color space. However, it cannot be used in detecting other types of chromatic translations,

rotation, etc. Aigrain *et al.* [113] have presented a technique for the detection of scene boundaries based on a differential model of motion picture. The algorithm is based on an estimation of the density function for the difference between two frames.

After the segmentation of a video stream, features within each shot such as content, length and camera operations are used for indexing proposes. Two approaches for video representation are distinguished. The first approach is based on image indexing techniques while the second is based on temporal features. We now present video indexing techniques based on spatial features. Temporal-based indexing techniques are presented in section 2.3.3.

3.3.2 Spatial Features

A set of representative (reference) frames is selected to represent each shot to be stored in the database. Image indexing techniques (section 2.1) are then applied on the reference frame. Arman *et al.* [111] have proposed a technique where each video shot is represented using the shape and color features of a reference frame. The reference frame is the 10*th* frame in the shot. The shape and color properties are represented using moments (the mass and the moments of inertia around the horizontal and vertical axes) and color histogram, respectively. Zhang *et al.* [112] have presented an algorithm where the reference frame(s) is first segmented based on prominent color. In addition, the reference frame is partitioned into 9 subframes (3×3). Each frame is indexed using the size, color, shape, and location of the segmented regions and the color histograms of the frame and the 9 subframes.

The major drawback of spatial-based video indexing techniques is that video sequences are treated as still images, thus the semantics contained in a sequence are lost. This

results in restricting the user queries. Temporal features allow the user to specify queries that involve the exact positions and trajectories of the objects in a shot.

3.3.3 Temporal Features

The apparent motion in a video sequence can be attributed to camera or object motion. In this section, we present a review of video indexing techniques using motion information and camera operations.

3.3.3.1 Motion

Here, image sequences are indexed based on the motion properties of objects within the sequence. The goal of the system is to be able to retrieve a ranked set of sequences which have object motions similar to that specified by the query.

Ioka *et al.* [120] have presented a method for retrieving sequences using motion information as a key. To start with, each frame is partitioned into rectangular blocks. Motion vectors are derived from the image sequences using block matching. These vectors are mapped into spatio-temporal space and the motion of each block is then represented as a single vector in the feature space. The vectors are clustered and a representative trajectory is generated for each group of vectors. A representative trajectory of a cluster is the closest to the mean vector of the cluster and has the longest life time. The representative trajectories are stored in the database. Queries can be specified using an interactive query specification mechanism which allows the user to enter a motion trajectory. The specified trajectory is matched with the trajectories of the sequences in the database using a distance measure and

the sequences with the smallest distance are retrieved. Although this technique does not address the problem of correspondence of trajectories, it can be incorporated as a low level tool into a complete video data management system for raw feature-based retrieval.

Lee *et al.* [121] have presented a video indexing technique based on the motion representation for the track of a moving object (using optical flow for motion extraction). Object motion is represented using a combination of the following 16 primitive motion types:

1. *Translation*: North, north east, east, south east, south, south west, west, north west.
2. *Translation in depth*: close to the camera, away from the camera.
3. *Rotation*: clockwise, counterclockwise.
4. *Rotation in depth*: rotate to left, rotate to right, rotate upward, rotate downward.

3.3.3.2 Camera Operations

The seven basic camera operations are: fixed, panning (horizontal rotation), tracking (horizontal transverse movement), tilting (vertical rotation), booming (vertical transverse movement), zooming (varying the focusing distance) and dollying (horizontal lateral movement) as shown in Figure 3.5. Camera operations include the basic operations and all the different possible combinations [115].

Akutsu *et al.* [115] have used motion vectors and their Hough transforms to identify the seven basic camera operations. The motion vectors pattern is characterized physically and spatially by i) the magnitude of the motion vectors, ii) the divergence/convergence point. For example, in case of a simple zoom in, pan right and tilt up at a constant speed, the motion vectors are shown in Figure 3.6a, Figure 3.6b, and Figure 3.6c, respectively. The algorithm has two stages. The first stage employs block matching to determine the motion vectors

between successive frames. In the second stage the motion vectors are transformed to the Hough space. The Hough transform of a line in the spatial domain is just a point in the Hough space. A group of lines in the spatial domain are represented by:

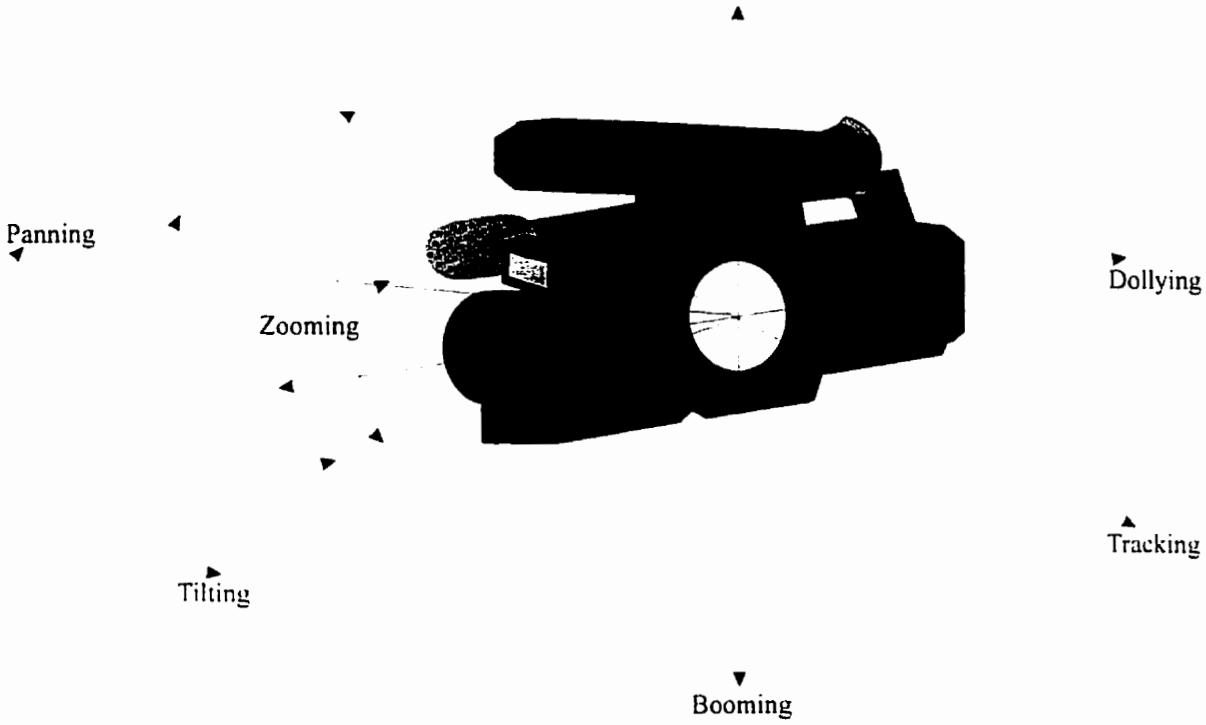


Figure 3.5: Basic camera operations.

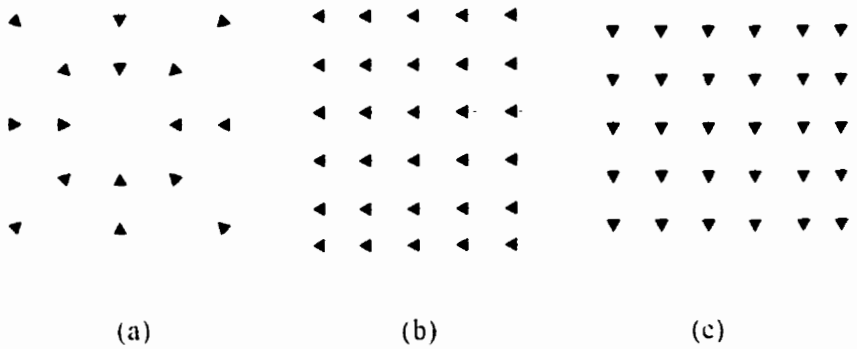


Figure 3.6: Motion vectors. zoom sequence, (b) pan sequence, and (c) tilt sequence

$$\rho = x_0 \cos(\varphi) + y_0 \sin(\varphi) \tag{3.11}$$

in the Hough space, where (x_0, y_0) is the point of divergence/convergence. The least squares method is used to fit the transformed motion vectors to the curve represented by eq. (3.11) Seven categories of camera operations have been estimated: pan, zoom, tilt, pan and tilt, pan and zoom, tilt, zoom and pan. We note this technique based on motion vectors is noise sensitive and has a high computational complexity.

Camera operations can be detected by examining what are known as the X-ray images [116]. An edge detection is first performed on all the frames within a shot. A horizontal X-ray image is then obtained by taking a weighted integral of the edge frames in the horizontal direction. Similarly, a vertical X-ray image is obtained by taking a weighted integral of the edge frames in the vertical direction. Camera operations are obtained by approximating the spatial distribution of the edge angles of the horizontal and vertical X-ray images. We note that performing edge detection on all frames in the sequence is time consuming.

We note that in all the previous techniques , only a subset of the camera operations are extracted. In addition, it is not possible to distinguish tracking from panning, and booming from tilting. Recently, Srinivasan *et al.* [117] have proposed a technique based on optical flow in order to distinguish tracking from panning, and booming from tilting. This technique is based on the idea that if the components of the optical flow due to camera rotation and zoom are subtracted from the optical flow, the residual flow will be parallel.

We note that in all these techniques for the detection of camera operations, it is assumed that there is no large moving object dominating the visual field in the video sequences. In case of the presence of a large moving object dominating the visual field, false detection of a camera operation may occur.

We recall from chapters 1 and 2 that image and video data are voluminous, hence, the visual data in future multimedia databases is expected to be stored in the compressed form. In order to avoid the unnecessary decompression operation in the searching process, it is efficient to index the image and video in the compressed form. Recently, compressed domain image and video indexing techniques based on compression parameters have been reported in the literature [88]-[149]. In sections 3.4 and 3.5 we present a review of image and video indexing in the compressed domain, respectively.

3.4 Image Indexing in the Compressed Domain

Compressed domain image indexing techniques are generally transform domain techniques and can be classified into four categories: discrete Fourier transform (DFT), Karhunen-Loeve transform (KLT), discrete cosine transform (DCT), and multiresolution-based techniques such as subbands and wavelets. We now present a review of compressed domain image indexing techniques.

3.4.1 Discrete Fourier Transform (DFT)

Fourier transform is very important in image and signal processing. DFT employs complex exponential basis functions and provides a good coding performance since it has good energy compaction property. We note that DFT has the following properties which are useful in indexing: (i) the magnitude of the DFT coefficients are translation invariant, and (ii) the

spatial domain correlation can be efficiently computed using DFT coefficients. We now present a review of Fourier domain indexing techniques.

Stone *et al.* [135] have proposed an image retrieval algorithm in Fourier domain. The algorithm has two thresholds that allow the user to independently adjust the closeness of a match. One threshold controls an intensity match while the other controls a texture match. The thresholds are correlation values that can be computed efficiently using the Fourier coefficients and are particularly efficient when the Fourier coefficients are mostly zero. Augustejin *et al.* [136] have studied the retrieval performance of satellite images based on the radial and angular distribution of Fourier coefficients. We note that the radial distribution is sensitive to texture coarseness whereas the angular distribution is sensitive to directionality of textures. It was observed that the radial and angular measures provide a good performance when a few dominant frequencies are present. The statistical measures provide a satisfactory performance in the absence of dominant frequencies. Celantano *et al.* [137] have evaluated the performance of angular distribution of Fourier coefficients in image indexing. Here, the images are first pre-processed with a lowpass filter and the FFT is calculated. The FFT spectra is then scanned by a revolving vector exploring 180^{o} range. The angular histogram is calculated by computing the sum of image components contribution for each angle. While calculating the sum, only the middle frequency range is considered as they represent visually important image components. The angular histogram is used as the feature vector for indexing. The feature vector is independent of translation in spatial domain while the rotation in spatial domain corresponds to a circular shift in the histogram.

3.4.2 Karhunen-Loeve Transform (KLT)

Karhunen-Loeve transform or principal component analysis, is based on the statistical properties of an image. Here, the basis functions are the eigenvectors of the autocorrelation matrix of the image. KLT provides maximum energy compaction and is statistically the optimum transform.

A technique based on the principal components analysis combined with DFT has been applied in the Photobook [60]. Let $\mathbf{x}_i$, i=1,2,....,N be the vector representing the DFT magnitude of the *i*th image in a training set of *M* images. The covariance matrix, **C**, of the training set is estimated. Each principal component is an eigenvector of the covariance. The computational complexity is reduced by noting that **C** can have at most *M* eigenvectors. The KLT coefficients of the images to be stored in the database, are obtained using the *M* eigenvectors. The transform coefficients are the features used for comparing textures. KLT has two advantages. First, vector components are decorrelated and second, components are compressed into a small number of coefficients. In addition, the use of DFT magnitudes makes the transform coefficients invariant to spatial translation. However, the performance may be degraded for images outside the training set.

The projection to Karhunen-Loeve space extracts the *Most Expressive Features* (MEF's) of an image. However, an eigenfeature may represent aspects of the imaging process, such as illumination direction, which are unrelated to recognition. An increase in the number of eigenfeartures does not necessarily lead to an improved success rate. To address this issue, Swets *et al.* [138] have proposed a Discriminant Karhunen-Loeve (DKL) projection where KLT is followed by a discriminant analysis to produce a set of *Most*

Discriminating Features (MDF's). In DKL projection, between-class scatter is maximized, while the within-class scatter is minimized. The authors have reported an improvement of 10-30% using DKL technique (over KLT) on a typical database.

KLT has been studied extensively for image compression applications [4]. Although KLT is optimal, it is not widely used due to its high computational complexity. We note that KLT is employed in analyzing and encoding multispectral images [22] and has therefore a potential for indexing in remote sensing applications.

3.4.3 Discrete Cosine Transform (DCT)

The DCT employs real sinusoidal basis functions [4] and has energy compaction efficiency close to the optimal KLT. As a result, the international image and video compression standards, such as JPEG, MPEG, H.261, and H.263, are based on DCT. We now present a review of the DCT-based indexing techniques that have appeared in the literature.

Chang *et al.* [88] have proposed a texture based indexing technique in the DCT/Mandala domain. The energies of the subbands are used to define the texture feature sets. For an $N \times N$ DCT/Mandala transform, N^2 bands are obtained. In order to reduce the search complexity, Fisher Discriminant analysis is used to reduce the texture feature vector. We note that Fisher Discriminant analysis generates a family of linear composites from the original feature vectors that provide for maximum average separation among training classes. The transform domain feature elements of the input image are mapped to a set of

eigenvectors with the maximum reparability significance. The Mahalanobis distance in the transformed feature space is used to measure the similarity between two images.

Shneier *et al.* [89] have proposed a technique for image based on the mutual relationship between the DCT coefficients of unconnected regions in both the query image and target image. Here, a set of $2m$ windows is selected. The windows are randomly paired, with the constraint that each window has only one partner. For each pair of windows, a bit is allocated in the m bit index. For each window the average of each DCT coefficient is computed resulting a 64-dimensional feature vector. For each feature value and each window pair, the index is computed by comparing the values of the first window with those of the second window. If the difference is greater than a threshold the corresponding bit is set to 1 otherwise it is reset to 0. The similarity between two images is measured by computing the similarity between their keys. In contrast to the technique proposed by Chang *et al.* [88] which is based on texture similarity, the similarity in this technique has no semantic meaning.

Smith *et al.* [42] have proposed a DCT based method where the image is divided into 4x4 blocks and the DCT is computed for each block resulting in 16 coefficients. The variance and the mean absolute values of each of these coefficients are calculated over the entire image. The texture of the entire image is then represented by this 32 component feature vector. Reeves *et al.* [43] have proposed a DCT-based texture discrimination technique which is similar to that of Smith *et al* [42]. Here, the image is divided into 8x8 blocks. A feature vector is formed with the variance of the first 8 AC coefficients. The technique does not employ the mean absolute value of the DCT coefficients, as in [42]. The technique assumes that the first AC coefficients have the most discriminating features, and thus avoids

discriminant analysis used in [42]. The run-time complexity of this technique is smaller than that of [42], since the length of the feature vector is small.

3.4.4 Multiresolution-Based Techniques

Recently, techniques based on image decomposition into a set of different resolution subimages using subbands, wavelets and Gabor have become popular in image coding and indexing applications [23]-[24]. Here, an image is passed through a set of lowpass and highpass filters, recursively, and the filter outputs are decimated in order to maintain the same data rate resulting in a multiresolution representation. Subband coding is generally implemented using quadrature mirror filters (QMFs) in order to reduce the aliasing effects arising out of decimation. In wavelet transform, the lowpass output is recursively filtered. Gabor transform is similar to wavelet transform, where the basis functions are Gaussian in nature and hence Gabor Transform is optimal in time-frequency localization. Since most of the energy in the subband domain is represented by a few lowpass coefficients, high compression ratio is achieved by discarding the high frequency coefficients. We note that the entire data is passed through the filters, and there is no blocking of data as in JPEG. Image decomposition has several advantages in coding - i) multiresolution capability, ii) better adaptation to nonstationary signals, iii) high decorrelation and energy compaction efficiency, and iv) reduced blocking artifacts and mosquito noise.

Chang *et al.* [139] have proposed a texture analysis scheme using irregular tree decomposition where the middle resolution subband coefficients are used for texture matching. In this scheme, a J dimensional feature vector is generated consisting of the energy

of J most important subbands. Indexing is done by matching the feature vector of the query image with those of the target images in a database. For texture classification, superior performance can be obtained by training the algorithm. Here, for each class of textures, the most important subbands and their average energy are found by the training process. A query image can then be categorized in one of the texture classes, by matching the feature vector with those of the representative classes.

Jacobs *et al.* [145] have proposed an indexing technique based on direct comparison of wavelet coefficients. Here, all images are rescaled to 128×128 pixels followed by a wavelet decomposition. The average color, the sign (positive and negative) and indices of M (a value of 40-60) largest magnitude transform coefficients of each image are calculated. The indices for all of the database images are then organized into a single data structure for fast image retrieval. A good indexing performance has been reported in the paper. However, the index is dependent on the location of transform coefficients. Hence, the target images which are translated and rotated versions of the query image, may not be retrieved using this technique.

Wang *et al.* [146] have proposed a technique which is similar to that of Jacob *et al* [145]. Here, all images are rescaled to 128×128 pixels followed by a four stage wavelet decomposition. Let the four lowest resolution subimages, which are of size 8×8, be denoted by S_L (lowpass), S_H (horizontal band), S_V (vertical band), and S_D (diagonal band). Image matching is then performed using a three step procedure. In the first stage, 20% of the images are retrieved based on the variance of S_L band. In the second stage, a fewer number of images will be selected based on the difference of S_L coefficients of query and target images. Finally,

the images will be retrieved based on the difference of S_L, S_H, S_V and S_D coefficients of query and target images. For color images, this procedure will be repeated on all three color channels. The complexity of this technique is small due to hierarchical matching. The authors have reported an improvement of performance over Jacob's technique [145]. However, as in Jacob's technique, the indexing performance is not robust to translation and rotation.

Mandal *et al.* [141] have proposed to compare the histograms of directional subbands to find a match with the query image. It has been shown that the histograms of wavelet bands of similar images, with limited camera operations, are similar. The complexity of direct comparison of the histograms of all the subbands is reduced by matching the distribution parameters of the subbands. The *pdfs* (or histograms) of highpass wavelet subbands can be modeled using generalized Gaussian density (GGD) function [142] which is expressed in terms of two parameters σ (standard deviation) and γ (shape parameter). Hence, the dissimilarity between a target and query image can be expressed in terms of the difference of the band parameters. The images which have minimum distance are retrieved from the database.

Mandal *et al.* [143] have proposed a histogram-based technique in the wavelet domain which is robust to changes in illumination. In this technique, the change in the illumination level is estimated using scale invariant moments of the histogram. The subband parameters σ and γ of each subband of the target image are then changed appropriately to counter the effect of illumination change.

An indexing technique using Gabor wavelets was proposed by Manjunath *et al.* [144]. Here, each image is decomposed into four scales and six orientations. A feature vector, of

dimension 48, is then formed using the mean (μ) and standard deviation (σ) of each subband. The similarity of the query image and a target image is determined by the similarity of their feature vectors. In this technique, the number of orientations are more, i.e., six, compared to three orientations (horizontal, vertical and diagonal) in the wavelet domain. Hence, better directional discrimination is achieved with this technique. However, the Gabor wavelets are computationally expensive compared to dyadic wavelets.

3.5 Video Indexing in the Compressed Domain

3.5.1 Scene Change Detection

Recently, several algorithms for video segmentation in the compressed domain have been reported [102]-[109]. According to the type of information used, the algorithms for video segmentation in the compressed domain are divided into four classes, namely, segmentation using DCT coefficients, motion vectors, motion/DCT and subband decomposition.

3.5.2 DCT Coefficients

The standards for image and video compression (JPEG, MPEG and H.261) are DCT-based techniques [25]-[27]. The transform coefficients in the frequency domain are related to the spatial domain. Therefore, the DCT coefficients can be used for scene change detection in compressed video sequences.

Arman *et al.* [102]-[103] have proposed a technique for scene change detection in motion JPEG using DCT coefficients. For each compressed frame f_m^I, B blocks are first chosen *apriori* from R connected regions in f_m^I. A set of randomly distributed coefficients $\{c_v$

c_y, c_z, ...} is selected from each block where c_x is the xth coefficient. A vector Vf_m^I={c_1, c_2, c_3, ...} is formed by concatenating the sets of coefficients selected from the individual blocks in R. The vector Vf_m^I represents f_m^I in the transform domain. The normalized inner product is used as a metric to judge the similarity of frame f_m^I to frame f_n^I

$$\psi = 1 - \frac{Vf_m^I \bullet Vf_n^I}{\left|Vf_m^I\right\|Vf_n^I\right|} \tag{3.12}$$

A scene transition is detected if Ψ is greater than a threshold. In case of false positives, which result from camera and object motion, f_m^I and f_n^I are decompressed and their color histograms are compared to detect camera breaks [103]. Zhang *et al.* [104]-[105] have presented a pair-wise comparison technique in the transform domain similar to template matching techniques in the uncompressed domain. Here, the pair wise normalized absolute difference $D(f_m^I, f_n^I, i, j)$of the (i,j)block in two frames f_m^I and f_n^I is determined using

$$D(f_m^I, f_n^I, i, j) = \frac{1}{64}\sum_{k=1}^{64} \frac{\left|c(f_m^I, k, i, j) - c(f_n^I, k, i, j)\right|}{\max(c(f_m^I, k, i, j), c(f_n^I, k, i, j))} \tag{3.13}$$

where $c(f_m^I, f_n^I, i, j, k)$ is the kth coefficient of block (i,j) in f_m^I. If the difference $D(f_m^I, f_n^I, i, j)$ is larger than a threshold, the block (i,j) is considered to be changed. If the number of changed blocks exceeds certain threshold, a scene change in the video sequence from frame f_m^I to frame f_n^I is declared. Compared to the technique by Arman *et al.* [102], the processing time of this technique is less, however, it is more sensitive to gradual changes [105].

We note that the previous two algorithms are applied on video sequences compressed using motion JPEG. In case of MPEG video, only I-frames are compressed with DCT coefficients and hence the previous two techniques cannot be directly applied to the B- and P-frames. In addition, the techniques based on I-frames may result in false positives. To

overcome these problems, Yeo *et al.* [106] have proposed a unified approach for scene change detection in motion JPEG and MPEG. This algorithm is based on the use of only the DC coefficients. To start with, a DC frame f_m^{DC} is constructed for every frame in the sequence. The DC coefficients in JPEG and I-frames in MPEG are obtained directly from each block. For B- and P-frames in MPEG video the DC coefficients are estimated. The sum of the difference magnitude of the DC frames f_m^{DC} and f_n^{DC} is used as a measure of similarity between two frames, i.e.,

$$S_5(f_m^{DC}, f_n^{DC}) = \sum_{i=1}^{X/8} \sum_{j=1}^{Y/8} \left| P(f_m^{DC}, I, i, j) - P(f_n^{DC}, I, i, j) \right| \qquad (3.14)$$

where $P(f_m^{DC}, I, i, j)$ is the DC coefficient of block (i,j). A scene change from f_m to f_n is declared if: (i) $S_5(f_m^{DC}, f_n^{DC})$ is the maximum within a symmetric sliding window and (ii) $S_5(f_m^{DC}, f_n^{DC})$ is *2-3* times the second largest maximum in the window. Although this technique is fast, cuts may be misdetected between two frames which have similar pixel values but different density functions. A metric for gradual transition has also been proposed [106] based on temporal subsampling where one in every 20 frames is tested rather than successive frames. This technique is sensitive to camera flashes and variations in scene that typically occur before scene changes.

3.5.3 Motion Vectors

The apparent motion in a video sequence can be attributed to camera or object motion. Motion estimation/compensation plays an important role in video compression. The objective is to reduce the bit rate by taking advantage of the temporal redundancies between adjacent frames in a video sequence. Typically, this is accomplished by estimating the displacement

(motion vectors) of uniformly sized blocks between two consecutive frames. In general, motion vectors exhibit relatively continuous changes within a single camera shot, while this continuity will be disrupted between frames across different shots .

In MPEG, B- and P-frames contain the DCT coefficients of the error signal and motion vectors. Liu *et al.* [108] have presented a technique based on the error signal and the number of motion vectors. A scene cut between a P-frame f_m^P and a past reference P-frame f_n^P increases the error energy. Hence, the error energy provides a measure of similarity between f_m^P and the motion compensated frame f_n^P.

$$S_6(f_m^P, f_n^P) = \frac{\sum_{i=1}^{F_p} E_i}{F_p^2} \tag{3.15}$$

where E_i is the error energy of macroblock i and F_p is the number of forward predicted macroblocks. For the detection of scene changes based on B-frames, the difference between the number of forward predicted macroblocks F_p and backward predicted B_p is used. A scene change between a B-frame and its past reference B-frame will decrease F_p and increase B_p. A scene change is declared if the difference between F_p and B_p changes from positive to negative.

Zhang *et al.* [104] have proposed a technique for cut detection using motion vectors in MPEG. This approach is based on the number of motion vectors M. In P-frames, M is the number of motion vectors. In B-frame, M is the smaller of the counts of the forward and backward non-zero motion. Then $M < T$ will be an effective indicator of a camera boundary before or after the B- and P-frame, where T is a threshold value close to zero. However, this method yields false detection when there is no motion. This is improved by applying the

normalized inner product metric (eq. (3.12)) to the two I-frames on the sides of the B-frame where a break has been detected.

3.5.4 Hybrid Motion/DCT

Meng *et al.* [109] have presented a segmentation algorithm based on motion information and the DC coefficients of the luminance component. To start with, the DC coefficients in the P-frames are reconstructed. The variance of the DC coefficients $|\Delta\sigma^2|$ for the I- and P-frames is then computed. Three ratios are computed, namely:

$$R_p = \frac{\text{Number of intra compressed macroblocks}}{\text{Number of blocks with motion compensation}}$$

$$R_b = \frac{\text{Number of backward motion vectors}}{\text{Number of forward motion vectors}}$$

$$R_f = \frac{1}{R_b}$$

A two pass algorithm is applied. In the first pass, suspected scene change frames are marked. A P-frame and B-frame are suspected frames if R_p and R_b, peaks, respectively. An I-frame is a suspected frame if $|\Delta\sigma^2|$ peaks and R_f of the B-frames in front of them peaks. In the second pass, all suspected frames which fall in a dissolve region are unmarked. All the marked frames are then examined. If the difference between the current marked frame and the last scene change exceeds a threshold, then the current marked frame is a true scene change.

3.5.5 Segmentation Using Subband Decomposition

Lee *et al.* [110] have presented a scene detection algorithm where the temporal segmentation is applied on the lowest subband in subband compressed video. Four metrics have been investigated, namely:

- *Difference of histograms*: measures the absolute sum of the histograms of f_m^L and f_n^L:

$$S_1(f_m, f_n) = \sum_{i=1}^{N} \left| H(f_m^L, i) - H(f_n^L, i) \right| \tag{3.16}$$

This metric is insensitive to object motion, however, it is sensitive to camera operations such as panning and zooming.

- *Histogram of difference frame*: is the histogram of the pixel to pixel difference frame and measures the change between two frames f_m and f_n. The degree of change between f_m and f_n is large if there are more pixels distributed away from the origin.

$$S_2(f_m, f_n) = \frac{\sum_{i \notin [-\alpha, \alpha]} H(f_m^L - f_n^L, i)}{\sum_{i=-N}^{N} H(f_m^L - f_n^L, i)} \tag{3.17}$$

where α is a threshold for determining the closeness to zero. The histogram of the difference frame (Eq. 24) is more sensitive to object motion than the difference of histograms (Eq. 23) [110].

- *Block histogram difference*: the lowest subband is divided into R blocks and the sum of the absolute differences of the blocks defined as

$$S_3(f_m, f_n) = \sum_{i=1}^{R} \sum_{j=1}^{N} \left| H(f_m^L, i, j) - H(f_n^L, i, j) \right| \tag{3.18}$$

is used as a metric for scene change detection. This metric is sensitive to local object motion.

- *Block variance difference*: instead of using the histogram, the variance of the block is used, i.e.

$$S_{10}(f_m, f_n) = \sum_{i=1}^{R}\sum_{j=1}^{N}\left|\sigma^2(f_m^l, i, j) - \sigma^2(f_n^{l'}, i, j)\right| \qquad (3.19)$$

This metric is block-based and hence it is sensitive to local object motion.

After the segmentation of a video stream, features within each shot such as content, length and camera operations are used for indexing proposes. Two approaches for video representation are distinguished. The first approach is based on image indexing techniques while the second is based on temporal features. We now present video indexing techniques based on spatial features. Temporal-based indexing techniques are presented in section 3.3.2.

3.5.6 Video Indexing Using Motion

Dimitrova *et al.* [122] have proposed a technique based on the motion compensation component of the MPEG video encoder. The trajectory of a macroblock is computed from the forward and backward motion vectors that belong to the macroblock. The position of a macroblock in a P-frame is computed using block coordinates and forward motion vectors. The position of a macroblock in a B-frame is computed by averaging the positions obtained from (i) the next predicted block coordinates and the backward motion vector and (ii) the previous block coordinates and forward motion vector. Each trajectory can be thought of as an n-tuple of motion vectors. The macroblock trajectories are the feature vectors used for indexing.

3.6 MPEG-7

We recall that MPEG compression standards have addressed primarily the storage and transmission aspects of audiovisual materials. MPEG-4 will extend the functionality of the underlying data representations and will also maintain some backward compatibility with MPEG-1 and MPEG-2. Recently, MPEG proposed to specify a new standard, called *Multimedia Content Description Interface* and referred to as MPEG-7. MPEG-7 will specify a standard set of descriptors that can be used to describe various types of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching for audiovisual material. In addition to having a description of the content, the MPEG-7 description may include other types of information about multimedia, such as coding scheme used, conditions for accessing the material, classification and links to other relevant material.

We note that MPEG-7 descriptions do not depend on how the described content is coded or stored. For example, visual information could be compressed using MPEG-4, JPEG, or VQ. MPEG-7 will allow different granularity in its descriptions, offering the possibility to have different levels of discrimination. This implies that the same material can be described using different types of features, tuned to the area of application. For example, in visual material, a lower abstraction level might be a description of shape, texture, color, and/or motion, while for audio, mood, tempo, and tempo changes might be used.

There are many potential applications that will benefit from the MPEG-7 standard. Examples include digital libraries, multimedia directory services, broadcast media selection,

and multimedia editing. It is anticipated that the MPEG-7 will become an international standard by the end of the year 2000.

3.7 Summary

In this chapter we have presented a review of image and video indexing techniques in both the compressed and uncompressed domains. First, we have presented an overview of a visual storage and retrieval system. This was followed by a review of image and video indexing techniques in the uncompressed (pixel) domain. A summary of the reviewed techniques is shown in Table 3.1.

The advent of compression techniques has led to the introduction of compressed domain indexing techniques based on compression parameters such as transform coefficients, motion vectors, etc. In this chapter we have presented a review of compressed domain image and video indexing techniques. A summary of the different techniques is shown in Table 3.2. We note that it is difficult to compare the performance of various indexing techniques. KLT, although statistically optimal, is computationally intensive. In addition, the basis images need to be stored, which reduces the compression efficiency. The block DCT in JPEG provides a good coding and indexing performance. However, the block structure was not originally intended for indexing. It has been shown in [42] that the wavelet transform outperforms the DCT/Mandala transform in image classification.

In conclusion, it is efficient to index image/video (visual data) in compressed form for the following reasons: (i) the advent of visual compression standards is expected to result in visual data being increasingly stored in compressed form [3]-[30], (ii) indexing in the

compressed domain eliminates the need to decompress the visual data and apply pixel-domain indexing techniques. (iii) many compressed bit streams contain information, such as motion vectors, which can be used in deriving content-based indices [102]-[106],[122], and (iv) in compressed domain there is a reduction in computational cost as the visual data is compactly represented [153]-[157].

In the next chapters we present novel techniques for combined image/video indexing and compression in the VQ compressed domain.

Problems	Methodologies	References
Image Indexing	Color	[40]-[53]
	Texture	[43],[54]-[59],[61]-[66]
	Sketch	[43],[80]
	Shape	[69]-[71]-[79]
	Spatial Relationships	[81]-[87]
Detection of Scene Change	Intensity/Color Template Matching	[46],[94],[95]
	Histogram-based Techniques	[97],[98],[100]
	Block-Based Techniques	[94],[100],[101]
	Twin Comparison	[95]
	Model-Based Segmentation	[113],[114]
Video indexing	Spatial features of key frames	[111],[112]
	Motion	[120],[121]
Detection of Camera Operations	Motion vectors	[115]
	X-ray images	[116],[117]

Table 3.1: Summary of image/video indexing techniques in the pixel domain.

Problem	Methodology	References
Image Indexing	Discrete Fourier Transform (DFT)	[135]-[137]
	Karhunen-Loeve Transform (KLT)	[22],[60],[138]
	Discrete Cosine Transform (DCT)	[88],[89]
	Multiresolution-Based Techniques	[23],[139]-[146]
Scene Change Detection	DCT Coefficients	[102]-[106]
	Motion Vectors	[104],[108]
	Hybrid Motion/DCT	[109]
	Subband Decomposition	[110]
Video indexing	Motion Vectors	[122]

Table 3.2: Summary of the image and video indexing techniques in the compressed domain.

4

Image Indexing Using Vector Quantization

4.1. Introduction

We recall from chapter 2, that several algorithms for image indexing have been reported in the literature. However, these techniques require a large amount of processing and additional storage space to compute and store the indices, respectively. A more serious problem is that these algorithms may not be applicable to images stored in the compressed form. We also recall from chapter 3, that several image compression algorithms have been reported in the literature to reduce the storage and transmission requirements in image applications. The International Standards Organization has proposed the JPEG [23] and MPEG [24] standards for image and video compression, respectively. Compressed domain image and video indexing techniques based on compression parameters such as DCT coefficients, subband coefficients, motion vectors, etc. have been reported in the literature [112]. We note that, at low bit rates, DCT based techniques suffer from both blocking effects and mosquito noise. Mosquito noise results from the quantization error of the high frequency components, which exist at the edge of an object but spans across the block in transform domain [20].

Vector quantization (VQ) is an efficient technique for low bit rate image and video compression [7]. In addition, VQ has the following advantages (i) fast decoding which makes it attractive for systems based on software only playback of video such as Intel's Indeo, Apple's QuickTime and Microsoft's Video, and (ii) reduced hardware requirements due to the simplicity of the decoder which makes it attractive for low power applications such as portable video-on-demand in wireless communications [106]. More importantly, VQ is naturally an indexing technique [155], where each subimage (vector) is mapped into an index (label). Hence, VQ is a promising approach for combining compression with indexing.

In this chapter, we propose two efficient techniques based on VQ that provide fast access to the images a database at a lower computational complexity. Most importantly, these techniques combine image compression and indexing. The proposed techniques provide fast access to the images in the database and have lower storage requirements.

This chapter is organized as follows. In section 4.2, we explain why VQ can be used to combine compression and indexing of images and video. Indexing using the histogram of codewords weighted by the number of labels and the histogram of the labels are presented in sections 4.3 and 4.4, respectively. Simulation results are reported in section 4.5, followed by the summary in section 4.6.

4.2. Indexing Using VQ

In VQ [10], a training set of representative images is decomposed into L-dimensional vectors. An iterative clustering algorithm such as the LBG algorithm is used to generate a codebook, $C=\{W_1,W_2, \ldots ,W_N\}$, where N is the number of codewords in the codebook and $W_i=\{w_{i1}, w_{i2}, \ldots ,$

$w_{iL}\}$. The codebook is then made available at both the transmitter and the receiver. In the encoding process, the image to be compressed is decomposed into L-dimensional vectors. Each vector $V_i=\{v_{i1}, v_{i2}, \ldots, v_{iL}\}$ is mapped into another L-dimensional vector W_j

$$q: V_i \longrightarrow W_j \tag{4.1}$$

where $W_j \in C$. In other words, vector quantization involves the partitioning of the L-dimensional space into N decision regions $\{\rho_i, i=1, 2, \ldots, N\}$, each containing one of the N reproduction vectors or codewords W_i. The vector V_i is quantized as W_j if it is in the region$\{\rho_j\}$, that is

$$q: V_i \longrightarrow W_j \quad \text{if } V_i \in \rho_j \tag{4.2}$$

which implies that the mapping is completely characterized by the partition ρ_i. Here, the selection rule is a minimum distortion or nearest neighbor rule; i.e. :

$$q(V_i) = W_j \quad \text{iff} \quad d(V_i, W_j) \le d(V_i, W_k) \ \text{ for all } k \tag{4.3}$$

where $q(.)$ is the quantization operation and $d(V_i, W_j)$ is a distortion measure which represents the error when V_i is reproduced by W_j.

Thus, VQ involves a clustering and mapping operations. The two operations make VQ a natural indexing technique due to the following:

- The clustering process involved in the generation of the codebook, implies that vectors having "similar" properties are grouped together.
- Input vectors which generally have much in common, are likely to have to the same label.

A visual illustration of encoding and decoding processes is shown in Figures 4.1 and 4.2 using a codebook of size 16 codewords and 16×16-dimensional vectors. It can be seen that the codewords, which can be regarded as a set of subimages, can be used to derive content-based features. We also note from Figures 4.1 and 4.2 that similar vectors map to similar codewords.

We conclude from the above discussion that features derived from codewords, which represent an image, have the potential to be content-based indices. In order to provide fast and efficient retrieval in a database system environment, these features have to satisfy several requirements including: (a) simple to derive and represent, (b) can be compared using a similarity measure which involves low computational complexity, and (c) provide excellent retrieval rate. We have selected two features: (a) the histogram of codewords weighted by the number of labels and (b) the histogram of the labels. Simulations demonstrated that the two features provide fast access to the images in a database with lower computational complexity compared to other techniques.

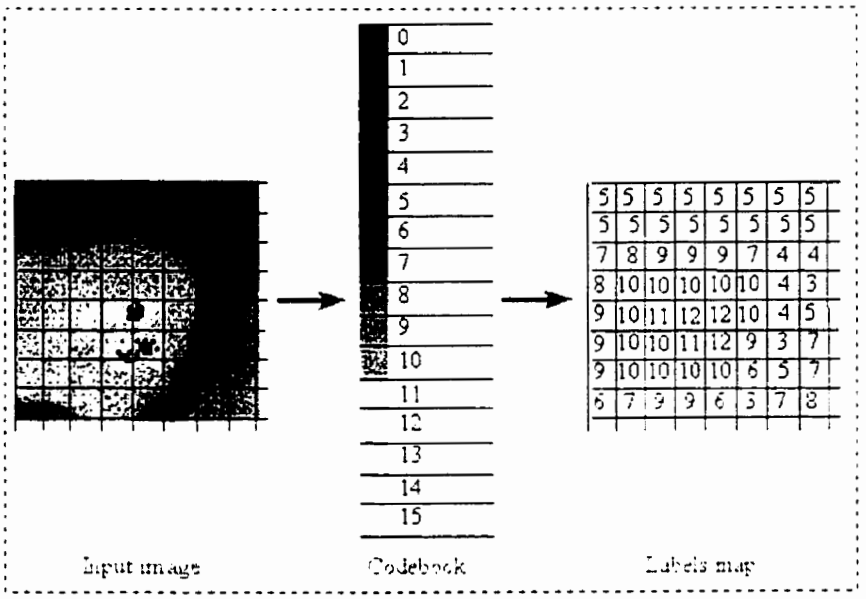


Figure 4.1: VQ encoding.

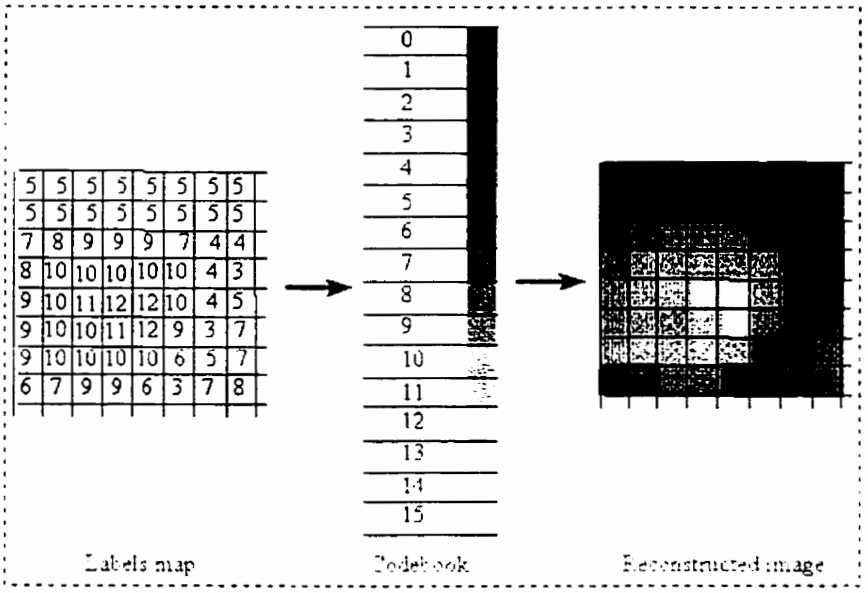


Figure 4.2: VQ decoding.

4.3. Histogram Of The Codewords Weighted By The Frequency Of The Labels

The histogram of the pixels of an image refers to the probability density function of the image intensities. For color images the histogram refers to the joint probability distribution of the three color channels. In image indexing using histogram, the images are scaled to the same number of pixels, and the histograms are the feature vectors which are used as image indices. A distance measure is used in the histogram space to measure the similarity of two images. We refer to this technique as the direct histogram of the pixels technique (H-PX).

The advantage of H-PX is that it is invariant to image rotation, translation and viewing axis [20]. However, it requires a large amount of processing and additional storage space. For an image of size $X \times Y$, the calculation of the histogram of each color channel requires $O(XY)$

additions and $O(XY)$ increments. In addition, $O(P)$ operations are required to compare a pair of histograms, where P is the number of bins in the histogram. The total number of operations required to calculate the histogram for images of various sizes are tabulated in column II of Table 4.1. Hence, the use of H-PX on compressed image has the following disadvantages: (a) requires decompression before feature extraction which entails large storage, and (b) has high computational complexity. We now present an algorithm for the indexing of compressed images using VQ which has the advantages of H-PX at lower storage and computational requirements.

The histogram of the codewords weighted by the frequency of the labels is calculated as follows. To start with, for each codeword, j, in the codebook, the histogram of the pixels, $\{w_{i,j}; i=1,2, \ldots, P\}$, is generated and stored along with the corresponding codeword. Let $m_1, m_2, \ldots, m_N$ be the frequency of the labels $l_1, l_2, \ldots, l_N$, respectively. The summation of the histograms of the codewords weighted by the frequency of the labels is a close approximation of the histogram of the image as illustrated in Figure 4.3. In other words, the histogram of an image $\{H(f_m,i) ; i=1, 2, \ldots, P\}$ is approximated by

$$H(f_m,i) = \sum_{j=1}^{N} m_j \times w_{i,j} \qquad (4.4)$$

for $i=1, 2, \ldots, P$. We refer to this approach by H-CL. For example, the histograms of pixels of the Lena image (Figure 4.4) calculated using H-PX and H-CL (N =512, L =16) are shown in Figure 4.5. The histogram of the codewords weighted by the frequency of labels is used as an index to store and retrieve the image.

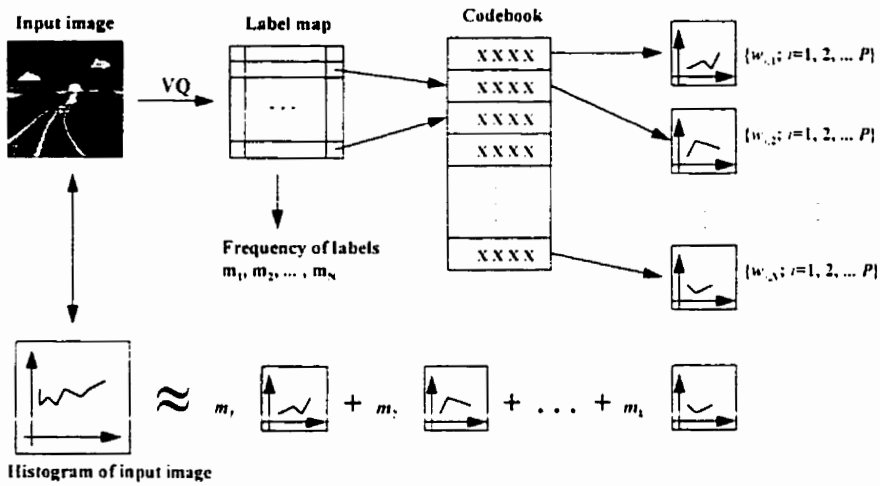


Figure 4.3: Calculation of histogram of the codewords weighted by the frequency of labels

For an image of size $X \times Y$ pixels, the calculation of the histogram of the pixels in H-CL requires the same number of additions and comparisons as the H-PX algorithm, however it reduces the number of increments to $O(XY/L)$ operations. Comparing columns II and III of Table 4.1, it can be seen that for L=16, the number of operations required in H-CL is approximately 50% of that in H-PX which results in a faster execution.

Figure 4.4: Lena image.

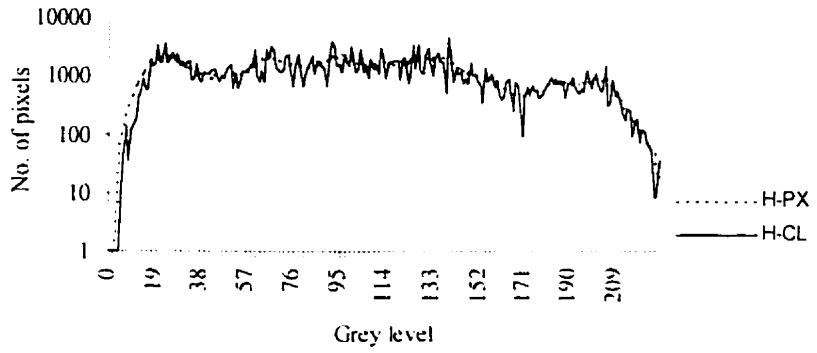


Figure 4.5: The histograms of pixels of the Lena image.

4.4. Histogram Of The Labels

VQ is a mapping from a vector in L-dimensional space into a finite set (codebook) of reproduction vectors (codewords). We note that the information conveyed by a set of quantized vectors, is also encoded in the set of codeword labels. To illustrate this, consider the example shown in Figure 4.6. Here, the luminance component (Y) of a sequence of 5 images with various camera operations are shown in Figure 4.6a-Figure 4.6e. The images are compressed using VQ at a compression ratio of 16:1 (N=256,L=16)). The codebook is arranged in the ascending order of the average and standard deviation of the codewords. By ordering the codebook, similar vectors map to neighboring labels and hence the label map of an image produces a scaled version of the image as shown in Figure 4.6f-Figure 4.6j. This suggests the use of feature vectors derived from the labels as indices for the database. Here, the histogram of the labels of an image f_m is a K-dimensional vector $\{H(f_m,i) : i=1, 2, \ldots, K\}$, where $H(f_m,i)$ is the number of labels i in the compressed image and K is the number of codewords in the codebook. The histograms of the

labels are the feature vectors used as image indices. We refer to this algorithm as the direct histogram of the labels (H-LB).

a) Original image

f)

b) Rotation to the right

g)

c) Rotation to the left

h)

d) Pan left and boom up

i)

e) Zoom out

j)

Figure 4.6: Five images and their label maps.

For an image of size $X \times Y$ pixels, H-LB requires $O(XY/L)$ additions, $O(XY/L)$ increments and $O(N)$ operations for comparing a pair of histograms. We note that for L=16, H-LB requires only 6.25% of the number of operations required by H-PX as can be seen from Table 4.1.

I Image size	II (H-PX)	III (H-CL)	IV (H-LB)
256×256	4.0×10^5	2.0×10^5	2.5×10^4
352×288	6.1×10^5	3.2×10^5	3.8×10^4
512×512	1.6×10^6	8.4×10^5	9.8×10^4
720×576	2.5×10^6	1.3×10^6	1.6×10^5

Table 4.1: The number of operations required to calculate the histogram of the pixels (H-PX), histogram of the codewords weighted by the number of labels (H-CL), and the histogram of labels (H-LB).

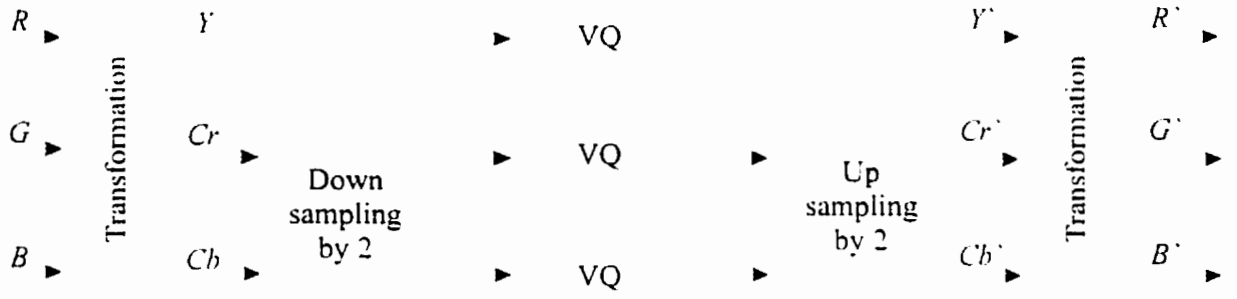


Figure 4.7: Block diagram of the compression algorithm.

4.5. Simulation Results

We recall from chapter 3 that several VQ algorithms for the compression of still image have been reported in the literature [7]. In our simulations, we have adopted the configuration shown in Figure 4.7. Here, the color image is first transformed from *RGB* to the *YCrCb* format. The *Y* component represents the luminance of a color pixel, while the *Cr* and *Cb* represent the two

chrominance components. The chrominance components have a lower signal energy, and hence they can be spatially subsampled without degrading the overall coding performance. The *Cr* and *Cb* components are subsampled by a factor of two in both the horizontal and vertical directions. Each of the three components (*Y*, *Cb* and *Cr*) is compressed separately using VQ through the same process for monochrome pictures. Several *Y*, *Cb* and *Cr* codebooks were pre-generated separately using the LBG algorithm as described in section 3.1.3. We note that the codebooks are generated using the same training set. These codebooks are used to compress the test images used in the simulations presented in the rest of this sections. Let N_Y, N_{Cb}, and N_{Cr} be the sizes of the *Y*, *Cb* and *Cr* codebooks, respectively. The bit rate in bits/pixel is calculated as follows:

$$\frac{\log_2 N_Y}{L_Y} + \frac{\log_2 N_{Cr}}{4L_{Cr}} + \frac{\log_2 N_{Cb}}{4L_{Cb}} \qquad (4.5)$$

where L_Y, L_{Cb}, and L_{Cr} are the vector dimensions of the *Y*, *Cb* and *Cr* components, respectively.

The use of vectors of uniform dimension offers good matching for hardware implementation, hence, in our experiments, we have used $N_Y=N_{Cb}=N_{Cr}=N$ and $L_Y=L_{Cb}=L_{Cr}=L$. The VQ parameters (values of *N* an *L*) and the corresponding compression ratios are tabulated in Table 4.2.

Image retrieval is performed as follows. First, the index of the query image is determined. The index of the query is then compared with the indices of the images in the database. The comparison process is the computation of the "similarity" between the two images. The images whose histograms are closer to that of the query image are then retrieved. This process is illustrated in Figure 4.8.

codebook size N	Vector dimension L	Compression ratio
16	16	64
32	16	51
64	16	43
128	16	37
256	16	32
512	16	28
16	64	256
32	64	205
64	64	171
128	64	146
256	64	128
512	64	114

Table 4.2: Compression ratios for various codebook sizes and vector dimensions.

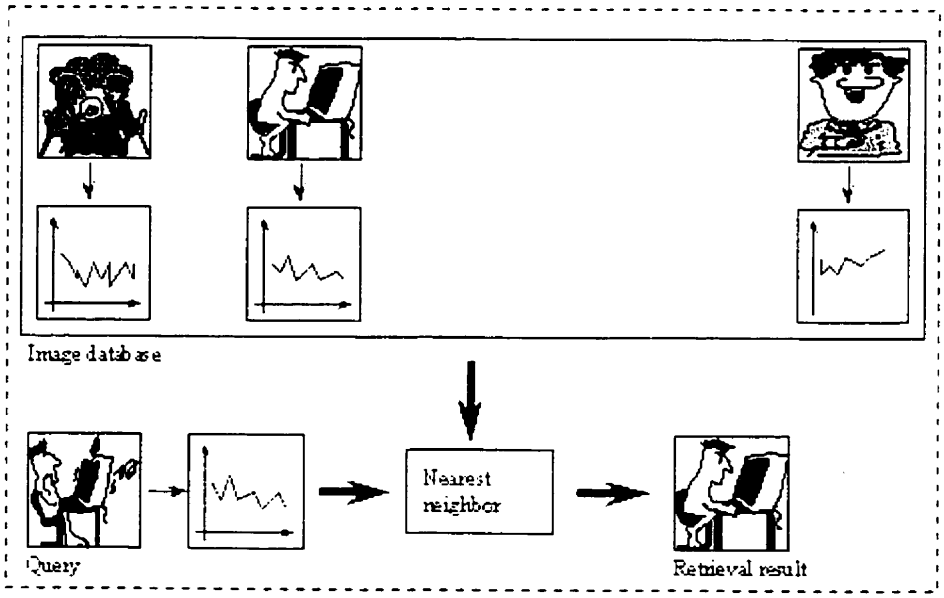


Figure 4.8: Image retrieval.

We use the retrieval results of H-PX as a baseline for comparison. For each image three histograms (one for each color channel C) are obtained. Given a query image f_m, and an image f_n in the database, the similarity between the two images is measured using the distance between their histograms. In our experiments two distance measures are used. The first is the sum of the intersections of the corresponding histograms (INTR):

$$\sum_{c}\left[\frac{\sum_{i=1}^{P}\min\left(H_c(f_m,i),H_c(f_n,i)\right)}{\sum_{i=1}^{P}H_c(f_n,i)}\right] \tag{4.6}$$

where $C=\{C_1,C_2,C_3\}$ is the three color channels. For example, in case of using the RGB color coordinate system, C_1, C_2 and C_3 are equal to R, G and B, respectively. The second distance measure which is used to evaluate the similarity between two images is the sum of the Euclidean distances between the corresponding histograms (EUCL):

$$\sum_{c}\sum_{i=1}^{P}\left|H_c(f_m,i)-H_c(f_n,i)\right|^2 \tag{4.7}$$

For each query, let M be the number of similar images in the database. Let T be the number the retrieved images (the number of relevant and non-relevant images retrieved in response to a query). We define the retrieval rate of a query image j, R_j, as:

$$R_j=\begin{cases} r_j / M & M \le T \\ r_j / T & M > T \end{cases} \tag{4.8}$$

where r_j is the number relevant images retrieved. The retrieval rate, R, is the average of the query retrieval rates over the total number of queries, i.e.,

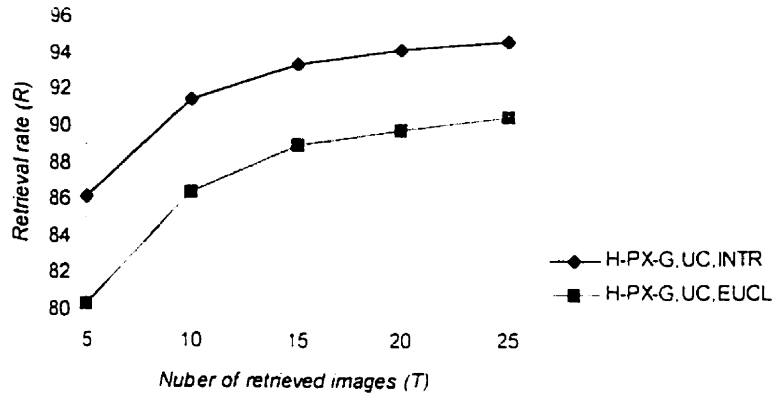
$$R = \frac{1}{N_q} \sum_{i=1}^{N_q} R_i \tag{4.9}$$

where N_q is the total number of queries.

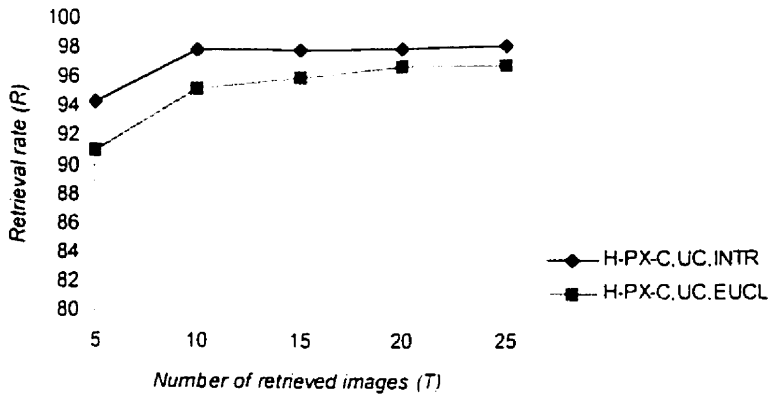
To evaluate the retrieval performance of H-PX, we have performed simulations using approximately 1000 images each of size 248×256 pixels. The images are taken from various image classes including people, natural scenes, buildings, animals, etc. The images are stored in the *YCbCr* coordinate system. We refer to this database (of the uncompressed images) as UC.

Five sets of experiments were performed. In the first set, the histogram of the Y channel is used as an index (H-PX-G). The retrieval rates of H-PX-G are graphed in Figure 4.9a. For example at T=10, the retrieval rate of H-PX-G using the histogram intersection (INTR) and Euclidean (EUCL) are 95.27% and 86.41%, respectively. This means that on the average, 95.27% of the similar images are present in the retrieved images using INTR. The corresponding rate for EUCL is 86.41%. In the second set of experiments, the histograms of the three color channels are used as an index (H-PX-C). The retrieval rates of H-PX-C are graphed in Figure 4.9b. It can be seen from Figure 4.9a and Figure 4.9b that INTR has higher retrieval rate than that of EUCL. For H-PX-G, INTR outperforms EUCL by 1%-4%, while for H-PX-C, INTR outperforms EUCL by 4%-6%. This is because, in EUCL, the large error components dominate the small error components resulting in a lower performance. By comparing Figure 4.9a and Figure 4.9b, it can be seen that H-PX-C out performs H-PX-G by 3%-8% and 6%-11% using INTR and EUCL, respectively. However, the complexity of H-PX-G is 1/3 of H-PX-C.

The test images were compressed at different compression ratios as shown in Table 4.2. We refer to each database (of compressed images) by VQ(*compression ratio*). For example, VQ(32:1) refers to the image database where the images are compressed at 32:1 using the codebook with N=256 and L=16 as shown in the fifth entry of Table 4.2.



(a)



(b)

Figure 4.9: Retrieval rate as a function of T using H-PX on UC: (a) Histogram of Y and (b) Histograms of Y, Cr and Cb.

In the third set of experiments, simulations were carried out using H-CL on the databases VQ(32:1) and VQ(28:1),VQ(128:1), and VQ(114:1). the performance of H-CL using the Y channel only (H-CL-G) and using the three color channels (H-CL-G) are evaluated in the databases VQ(32:1) and VQ(28:1). Retrieval rates of H-CL-G and H-CL-C are graphed in Figure 4.10a and Figure 4.10b, respectively. It can be seen from Figure 4.10 that the retrieval rate decreases as compression ratio increases (bit rate decreases). This is due to the fact that a histogram computed using eq. (**4.4**) approaches the original histogram at lower compression ratios. For example, for T=25 using INTR, at compression ratios of 32:1 and 28:1, the retrieval rates are 85.45% and 89.66%, respectively. The corresponding rates using EUCL are 71.70% and 74.77%, respectively. It can also be seen from Figure 4.10, that retrieval rates using INTR are higher than the rates of EUCL. Comparing Figure 4.10a and Figure 4.10b, it can be seen that H-CL-C outperforms H-CL-G by 7%-15% and 13%-25% using INTR and EUCL, respectively. We note that all the test images are outside the training set and therefore further improvements in retrieval performance can be expected for images inside the training set.

To investigate the effect of using a larger vector dimension retrieval rates of H-CL-G and H-CL-C on VQ(128:1) and VQ(114:1) are determined. Retrieval results are graphed in Figure 4.11. It can be seen from Figure 4.11 that by increasing the vector dimension from 16 to 64 the retrieval rate of H-CL-G decreases by 2%-6% and 3%-5% for codebooks of size 256 and 512 codewords, respectively. It can also be seen from Figure 4.11 that the corresponding decrease in retrieval rates for H-CL-C are 3%-4% and 2.5%-3%, respectively. Similar decrease in retrieval rates have been observed using EUCL. Using codebooks of sizes 256 and 512, increasing the

vector dimension from 16 to 64, for H-CL-G the retrieval rates decrease by 7%-8% and 1%-3%, respectively. The corresponding decrease for H-CL-C are 3%-6% and 4%-5%, respectively. This is because the distance between a histogram computed using eq. (**4.4**) using a larger vector dimension while keeping the codebook size and the histogram of the original will increase. The effect of using a larger vector dimension can be decreased by increasing the codebook size

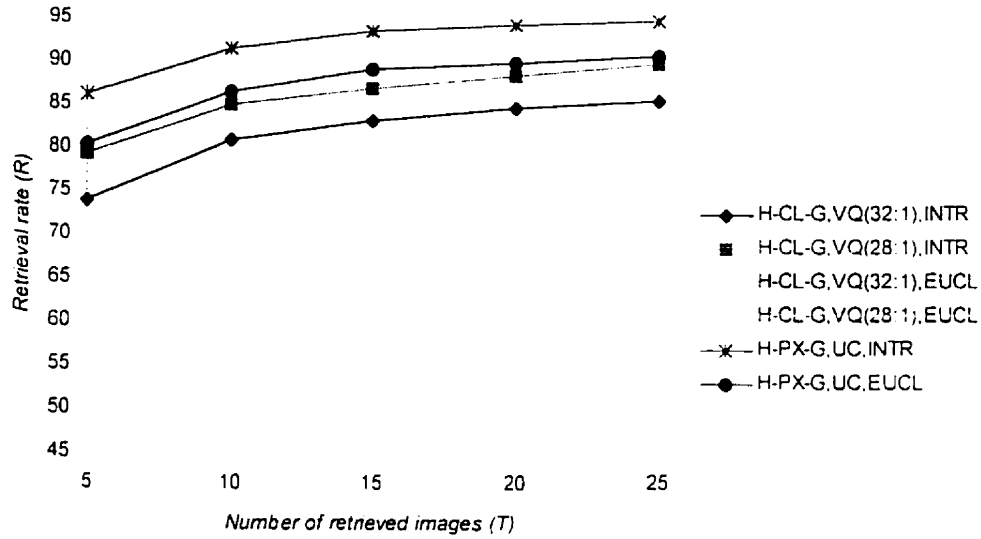
Retrieval rates were obtained using H-LB on VQ(64:1), VQ(51:1), VQ(43:1), VQ(37:1) VQ(32:1) and VQ(28:1). The rates are graphed in Figure 4.12. It can be seen from Figure 4.12a, that using INTR the retrieval rate increases as the bit rate increases (compression ratio decreases). It can be seen from Figure 4.12b that using EUCL the retrieval rate increases as the compression ratio decreases from 64:1 to 32:1. However, using EUCL the retrieval rate at a compression ratio of 28:1 is less than that at a compression ratio of 32:1. This is because the number of empty bins in the histogram of labels increases with increasing codebook size (not all the codewords are used in the compression). This results in large error components which dominate smaller errors and hence reducing the retrieval rate. It can be seen from Figure 4.12 that H-LB outperforms H-PX for VQ(37:1), VQ(32:1), and VQ(28:1) by 0.1%-3%, 1%-5% and 1%-5%. For VQ(43:1), VQ(51:1), and VQ(64:1), H-PX outperforms H-LB by 0-1% and 1%-4%. Comparing Figure 4.12a and Figure 4.12b, it can be seen that INTR outperforms EUCL, therefore, only the results using INTR are reported in subsequent experiments.

To investigate the effect of using larger vector dimension, the test images are compressed using a vector dimension of 64 (8×8 block). Retrieval rates of H-LB-G on VQ(256:1), VQ(205:1), VQ(171:1), VQ(146:1), VQ(128:1), and VQ(114:1) were obtained. The results are shown in Figure 4.14. It can be seen from Figure 4.14, that increasing the vector dimension from 16 to 64 (reducing the bit rate by a factor of 4) using a codebook of size 16

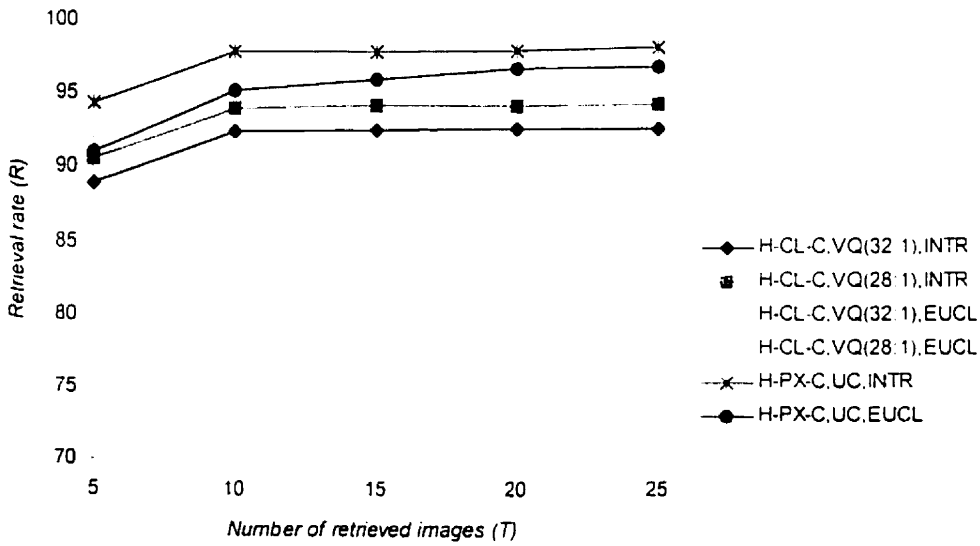
codewords reduces the retrieval rate by 0%-1.5%. However, for using a larger codebook size (32, 64, 128, 256 and 512), the retrieval rates using a vector dimension of 64 are slightly higher than the corresponding rates using a vector dimension of 16.

In the last set of experiments, the performance of the proposed H-LB technique is compared with a technique based on the histogram of the DC coefficients. Here, the test images are also compressed using JPEG to form two databases: JPEG(32:1) and JPEG(64:1). The JPEG(32:1) and JPEG(64:1) images are compressed using JPEG at a compression ratio of approximately 32:1 and 64:1, respectively. The histogram of the DC coefficients (H-DC) is used as a feature vector to access the images in the databases. Retrieval rates are graphed in Figure 4.15. It can be seen from Figure 4.15, that at compression ratios of 32:1 and 64:1, H-LB outperforms H-DC by 3%-6% and 20%-30%, respectively.

From the retrieval results presented in this section, it can be seen that H-LB outperforms H-LC at both high and low compression ratios, however, form the visual retrieval results (see Figures 4.16 and 4.17).We notice that H-CL out performs H-LB in retrieving rotated or translated images using the original image as a query.

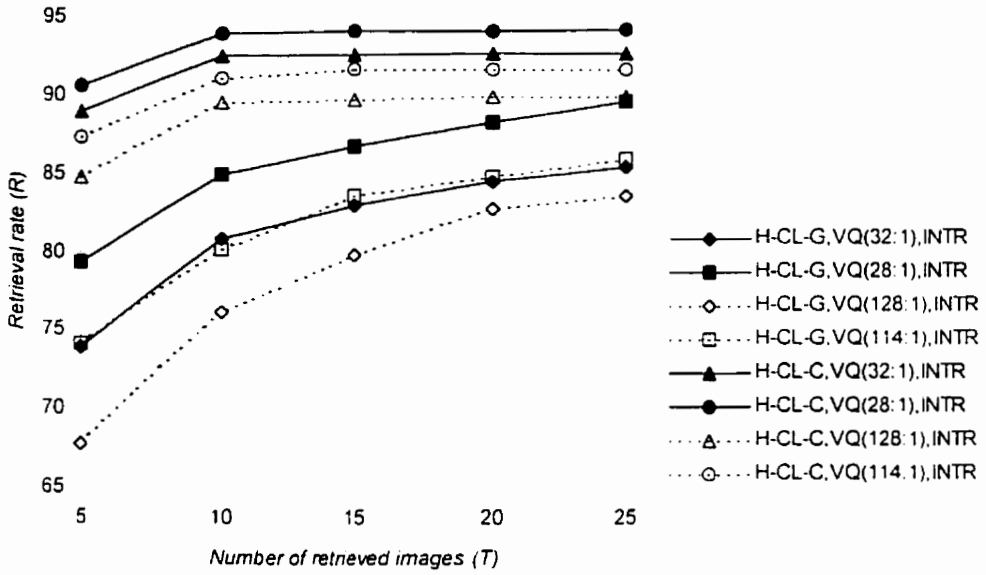


(a)

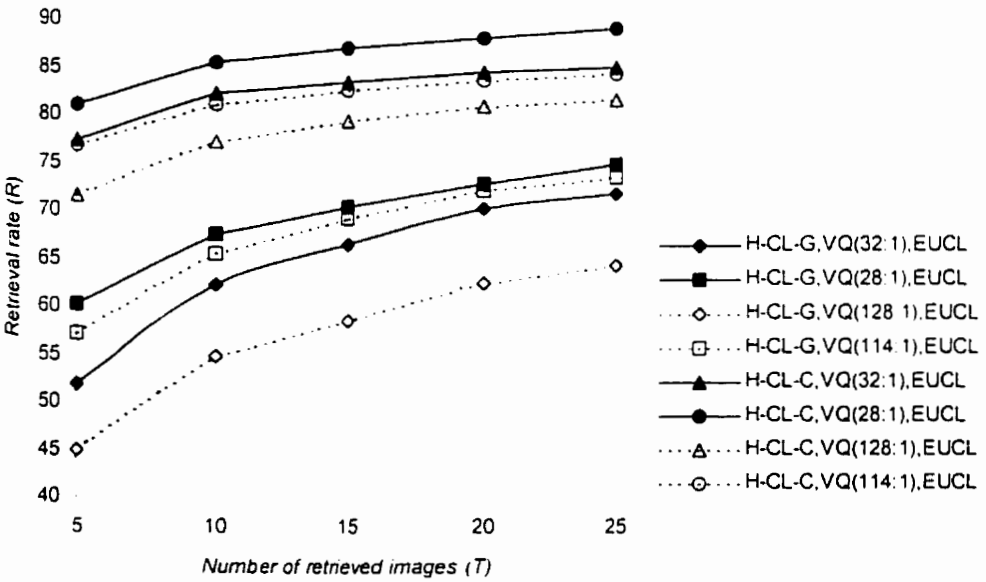


(b)

Figure 4.10: Retrieval rates using H-CL on VQ(32:1) and VQ(28:1): (a) Using only Y channel (b) Using the three color channels.

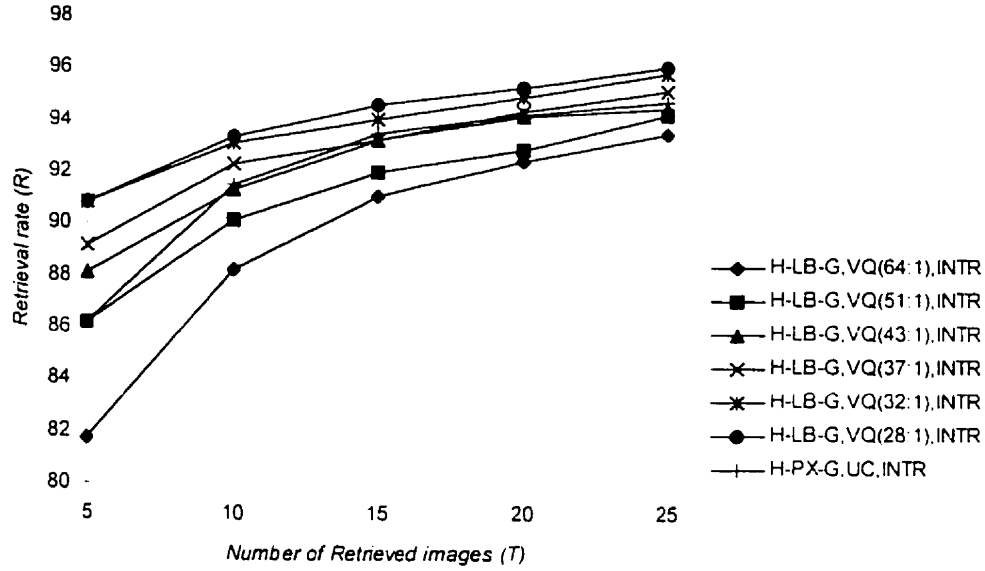


(a)

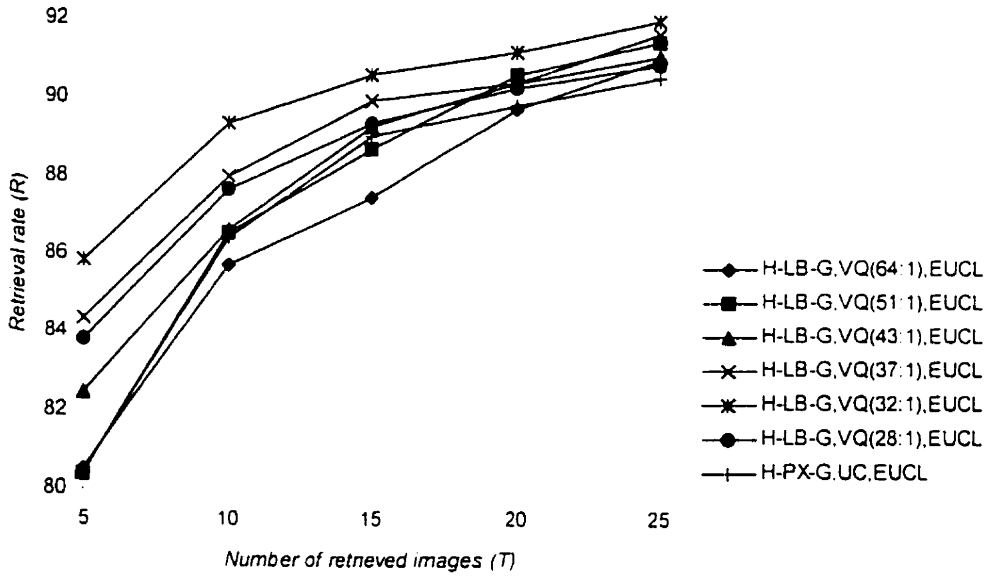


(b)

Figure 4.11: Effect of vector dimension on retrieval rates using H-CL (a) INTR (b) EUCL



a)



b)

Figure 4.12: Retrieval rates using H-LB-G on VQ(64:1), VQ(51:1), VQ(43:1), VQ(37:1) VQ(32:1) and VQ(28:1). using: a) INTR, and b) EUCL.

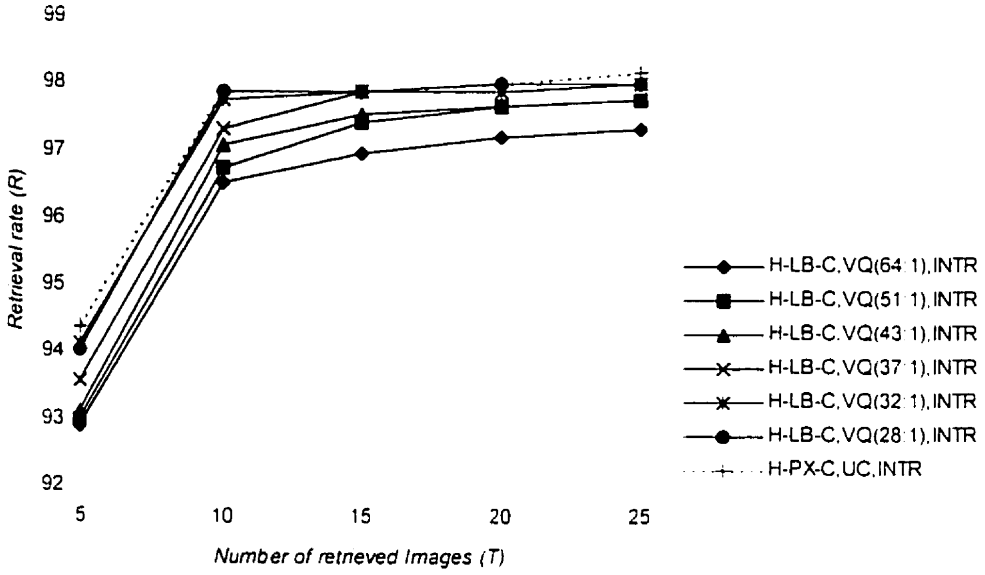
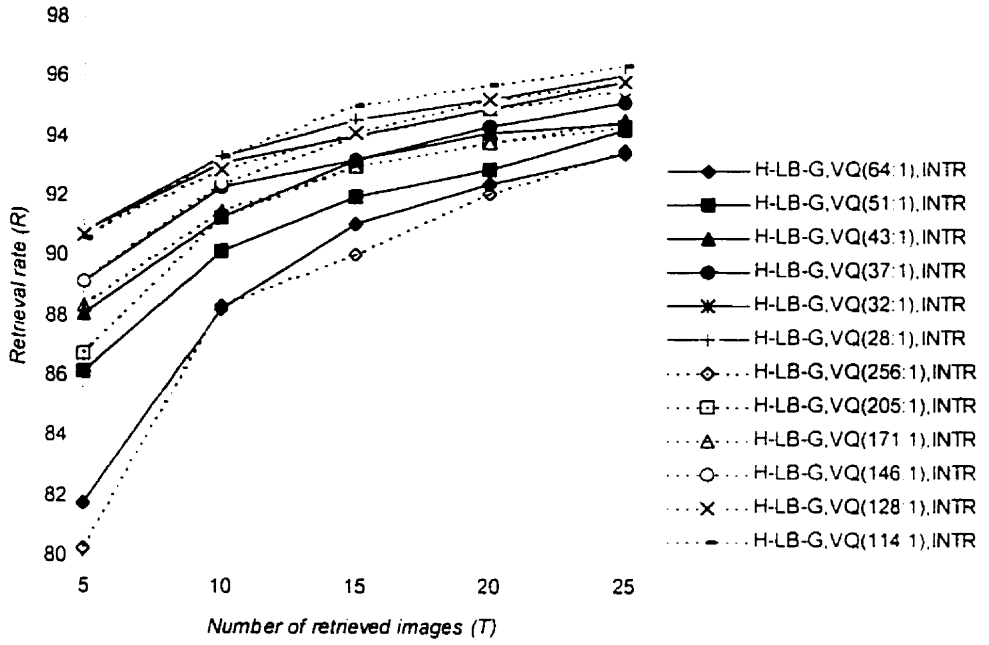
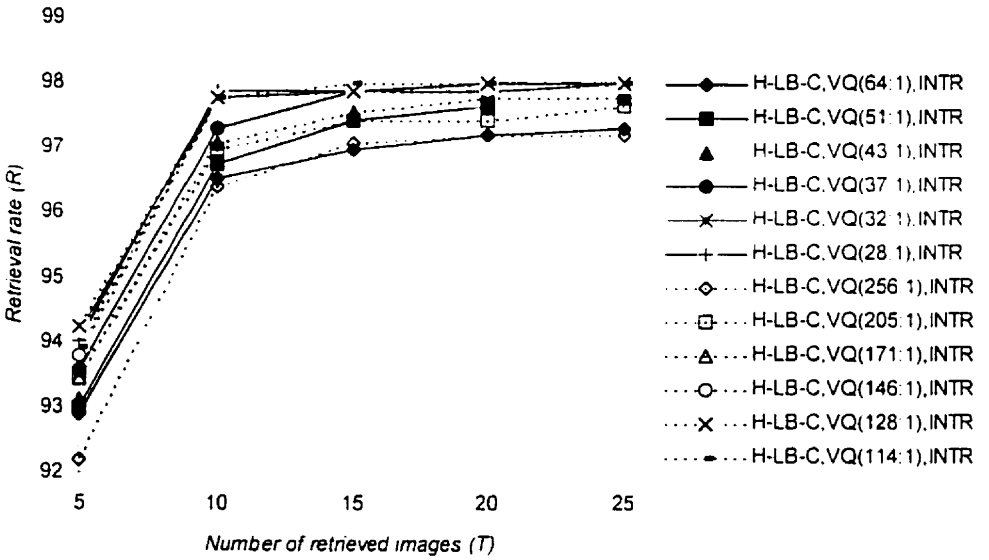


Figure 4.13: Retrieval rates using H-LB-C on VQ(64:1), VQ(51:1), VQ(43:1), VQ(37:1) VQ(32:1) and VQ(28:1), using: a) INTR, and b) EUCL.



(a)



(b)

Figure 4.14: Effect of vector dimension on the retrieval of H-LB-C (a) gray images (b) color images.

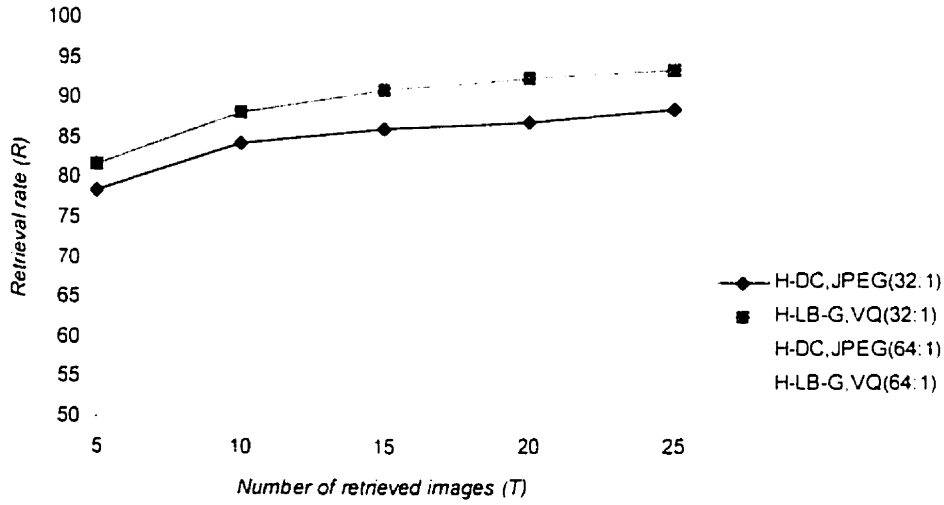


Figure 4.15: Retrieval rates of H-DC on JPEG(32:1) and JPEG(64:1).

(a)

(b)

(c)

Figure 4.16: (a) Query image. First three retrieved images (b) H-CL-C (c) H-LB-C

(a)

(b)

(c)

Figure 4.17: (a) Query image. First three retrieved images (b) H-CL-C (c) H-LB-C

4.6. Summary

In this chapter, we have demonstrated that VQ is an efficient technique for joint image compression and indexing. We have presented novel algorithms for the indexing compressed image using VQ. In the first technique (H-CL), the histogram of the codewords weighted by the number of labels is used as an index to store and retrieve an image. In the second technique (H-LB), the histogram of the labels of an image is used as an index. Simulation results demonstrate that H-CL has a similar performance to H-PX at a compression ratio of 32:1, while H-LB outperforms H-CL and H-PX at high as well as low compression ratios. In addition, H-CL is best suited for retrieving rotated or translated images using the original image as a query. In terms of computational complexity, H-CL and H-LB require only 50% and 6.25% of the number of operations required by H-PX. The performance of H-LB was also compared with the histogram of the DC coefficients in JPEG compressed images (H-DC). At a compression ratio of 32:1 and 64:1, H-LB outperforms H-DC by 3%-6% and 20%-30%, respectively.

We note that H-CL can be applied to adaptive VQ compression techniques which are based on codebook replenishment, while H-LB can not be applied directly to those techniques. In chapter 5, we propose an indexing techniques for adaptive VQ based n the usage map in the spatial and wavelet domains.

5

Image Indexing Using Adaptive VQ

5.1. Introduction

In chapter 4, we have presented two image indexing technique based on VQ. In these techniques, the indices are derived from compressed images which eliminates the need for decompression and hence reduces the computational and storage requirements. However, the index is associated with the corresponding compressed bit stream, thereby reducing the storage efficiency. In this chapter, we propose a technique based on adaptive vector quantization which integrates the index of an image within the compressed bit stream. This integration has two advantages. First, the index is generated automatically at compression time, which avoids the unnecessary decompression and/or processing operations. Secondly, it does not require additional memory for storing the indices.

This chapter is organized as follows. The employed compression technique for adaptive wavelet VQ is detailed in section 5.2. The indexing algorithm is presented in section 5.3. Simulations are presented in section 5.4. Finally, the summary is presented in section 5.4.

5.2. Image Compression Using Wavelet Vector Quantization

It has been shown that wavelet transform based coding outperforms DCT-based coding, since it is free from both blocking effects and mosquito noise. We recall from chapter 3, that wavelet transform provides a tool for decomposing a signal into a weighted sum of basis functions called wavelets. For image applications, wavelet transform decomposes an image into a set of different resolution subimages corresponding to the various frequency components of the original image, resulting in a multiresolution representation.

The wavelet coefficients can be quantized and encoded using a variety of techniques. The combination of wavelet transform and VQ has been shown to be very efficient for very low bit rate image coding [16]-[17].

Here, the image to be compressed is first decomposed using wavelet transform. To start with 64-D vectors (v_0-v_{16}) are formed by combining the corresponding blocks in the 10 subimages as shown in Figure 5.1. The vector v_0 corresponds to the lowpass subimage, while (v_1-v_3), (v_4-v_{15}) and (v_{16}-v_{63}) correspond to the subimages at levels 3, 2, and 1, respectively. A block of size 2^{3-m} (m=1, 2,3) is used for each horizontal, vertical and diagonal orientation subimages at the mth level. In other words, the sizes of the blocks are scaled corresponding to the level of the wavelet pyramid. With this structure, the number of blocks in the subimages is constant and hence there is a one to one correspondence between a block at one resolution level in the wavelet pyramid and a block at the same position at a different resolution level. Hence, there is significant amount of redundancies among the various subimages.

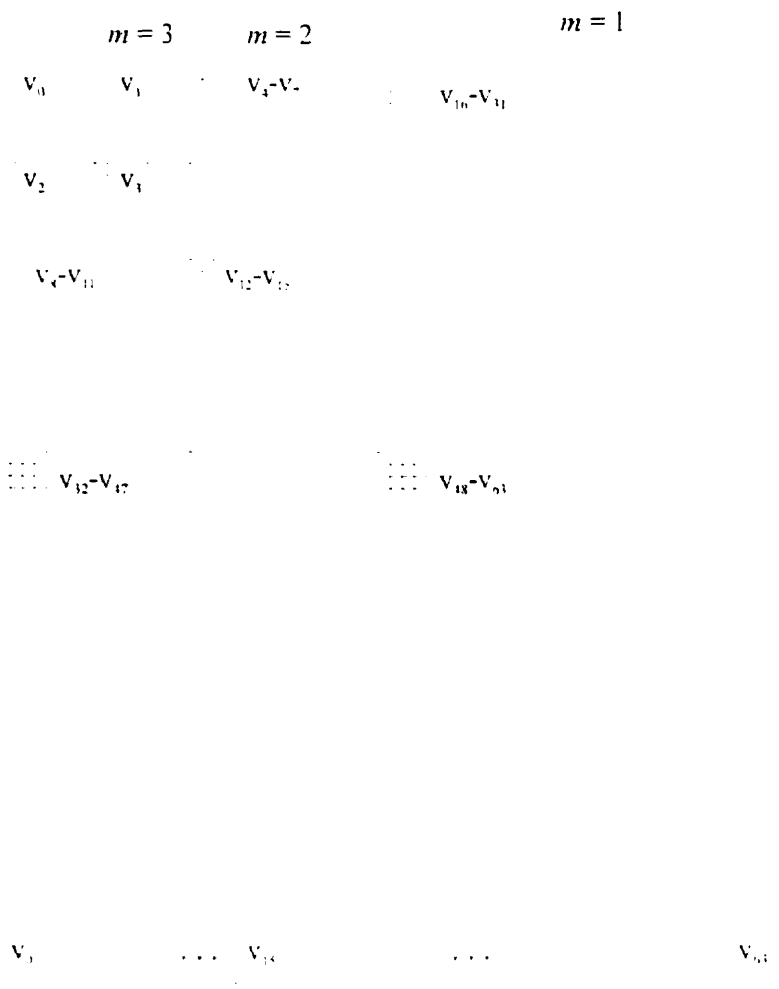


Figure 5.1: Vector formation.

To exploit the correlation among the various subimages, the 64-D vectors are encoded using nonlinear interpolative VQ (NIVQ). Here, 16-D feature vectors (v_0-v_{15}) are generated by taking the first 16 elements from lower resolution subimages within the corresponding 64-D vectors (v_0-v_{63}) as shown in Figure 5.1. VQ is performed on the 16-D vectors. We note that the codebook at the encoder (16-D codebook) differs from that at the decoder (64-D

codebook). The codebooks are designed by partitioning the 16-D and 64-D vector spaces into corresponding subspaces. The 16-D codebook is first generated by clustering the 16-D vector space into subspaces. The 64-D codebook is then generated by projecting the resultant subspaces in the 16-D vector space into the 64-D vector space. The projection between the 16-D and 64-D vectors is a one to one mapping operation.

This technique reduces the computational complexity in the conventional VQ process. In addition, it preserves the high frequency subimages even at high compression ratios (very low bit rate). This results in a superior coding performance. Good quality reconstructed images can be obtained at compression ratios of 40:1 for gray-level images and 100:1 for color images.

We recall from chapter 3, that VQ a large codebook must be used in order to ensure a good image fidelity, which in turn increases both the bit rate and the coding complexity. Typically, adaptive VQ is employed to eliminate the need for a large universal codebook. In adaptive VQ, the codebook or part of it is modified in order to match the local image statistics resulting in higher fidelity. We note that the improvement in image quality is achieved at the expense of increasing the computational complexity.

In the following section we present a codebook adaptation and indexing technique which results in lower bit rate for label encoding and provide indexing features.

5.3. Indexing using adaptive wavelet VQ

In the proposed technique, a large universal codebook of size N codewords is first generated off-line as described in section 5.2. The codebook is generated using a set of representative

images. For each image to be stored in the database, the image is decomposed using wavelet transform. The transform coefficients are then encoded using NIVQ. A usage map $\{u(f_m,j);\ 0 \le j \le K-1\}$, where $u(f_m,j) \in \{0,1\}$ is generated to indicate the used codewords as shown in Figure 5.2. The used codewords constitute a reduced codebook of size M codewords, where $M < N$. The labels corresponding to the reduced codebook and the usage map are stored in the database. We note that the reduced codebook corresponding to an image reflects the content of the image and similar images have similar reduced codebooks. The usage map corresponding to an image constitutes a feature vector which is used as an index to store and retrieve the image.

The similarity between two images f_m and f_n is measured using the following equation:

$$S(f_m, f_n) = \sum_{i=0}^{N-1} \left(u(f_m, i) \oplus u(f_n, i) \right) \qquad \textbf{(5.1)}$$

where $\oplus$ is the XOR operation. Using this metric, the comparison of two indices requires $O(N)$ bit operations, where N is the codebook size. The number of bits required to store an index is $O(N)$ bits. This technique doesn't require any additional operations to calculate the indices as the usage map is generated during compression of the images. We refer to this technique as indexing using the usage map (IUM). The proposed technique provides fast access to the compressed images in the database and has lower storage requirements. In addition, the lowest resolution subimages resulting from the wavelet decomposition can be used as visual icons for browsing purposes.

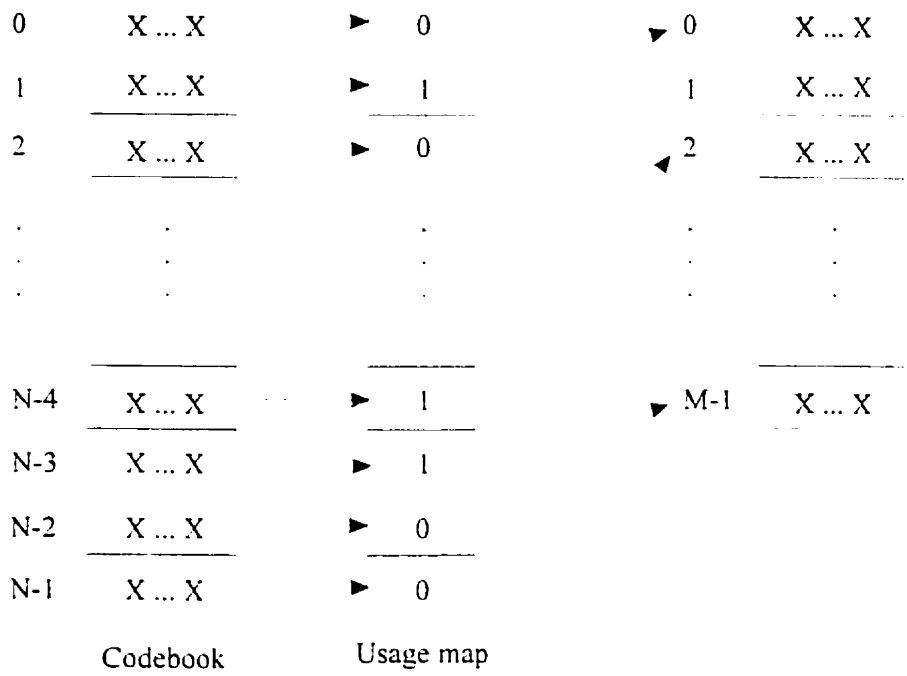


Figure 5.2: Usage map generation.

5.4. Simulation Results

Simulations were carried out using a database containing 500 gray level images, each of size 512×512 pixels. The database contains images of different classes based on textures, natural scenes, buildings, animals, etc. The images are first decomposed using a bi-orthogonal wavelet transform with 3/5 taps [16]. The corresponding coefficients of the wavelet filter bank are shown in Table 1. The transform coefficients are compressed using NIVQ as described in sections 5.2 and 5.3. Each image is represented by a usage map and a set of labels. We note that codebook of size 4096 codewords is generated using a training set of 20 images. The hierarchical indexing approach is used as a baseline for comparison .

In hierarchical indexing technique [107], a multiresolution indexing technique based on subband decomposition. The images are decomposed using m subband filters and the histogram of each subimage is generated. The histogram of a subimage is an n-D vector $\{H(i) : i=1, 2, \ldots, n\}$, where n is the number of gray levels and $H(i)$ is the number of pixels of gray level i in the image and serves as the index of the image. The histograms (indices) are compared at different resolutions in a hierarchical manner. To start with, the histogram of the lowest subimage is used as an index for image retrieval. If the retrieval result is not satisfactory, the histogram of the next higher resolution subimage is used as an index. The process is repeated until the images of interest are retrieved. We refer to this method as the hierarchical indexing approach (HIA). We note that HIA is computationally intensive procedure.

Retrieval rates of the proposed technique are evaluated using the query by example approach. Here, a sample image (query) is given and the task of the system is to retrieve images which are "similar" to the sample image. For this purpose, the system extracts the usage map (index) of the query image and matches it against all the usage maps (indices) of images stored in the database. The matching process is carried out by computing the "similarity" between the index of the query and those of the images in the database. The similarity between two images f_m and f_n is measured using equation 2. The retrieved images are arranged in the order of increasing value of the similarity metric.

	Analysis		Synthesis	
	Lowpass	Highpass	Lowpass	Highpass
n = -3	0.0	0.0	0.0	0.0
n = -2	-0.125	0.0	0.0	0.0
n = -1	0.25	0.0	0.25	0.125
n = 0	0.75	0.25	0.5	0.25
n = 1	0.25	-0.5	0.25	-0.75
n = 2	-0.125	0.25	0.0	0.25
n = 3	0.0	0.0	0.0	0.125

Table 5.1: Coefficients of bi-orthogonal wavelets.

For each query, let T be the number the retrieved images (the number of relevant and non-relevant images retrieved in response to a query). Let S be the number of similar images in the database We define the retrieval rate of a query image j, R_j, as:

$$R_j = \begin{cases} r_j / S & S \le T \\ r_j / T & S > T \end{cases} \qquad \textbf{(5.2)}$$

where r_j is the number relevant images retrieved. The retrieval rate, R, is the average of the query retrieval rates over the total number of queries, i.e.,

$$R = \frac{1}{N_q} \sum_{i=1}^{N_q} R_i \qquad \textbf{(5.3)}$$

where N_q is the total number of queries.

The retrieval rates are shown in Figure 5.3. It can be seen from Figure 5.3, that the rank of the similar image is among the first 5 (1% of the total database population) retrieved images at a rate of 87.2%. For example, the first three retrieved images in response to the queries shown in Figures Figure 5.4a and Figure 5.6a are shown in Figure 5.4b and Figure 5.6b, respectively.

The same queries were performed on the database using HIA. Here, index matching is a hierarchical approach, where the histograms of the subimages are compared at different resolutions. Retrieval results are shown in Figure 5.3. It can be seen from Figure 5.3 that using IUM and HIA, the rank of the retrieved images where among the first 15 images at a rate of 93.2% and 92%, respectively. It can also be seen that using IUM and HIA, the rank of the retrieved images where among the first 25 images at a rate of 94.5% and 94.4%, respectively. Hence, HIA and IUM perform comparably. However, the computational complexity of IUM is much less than that of HIA which results in faster execution.

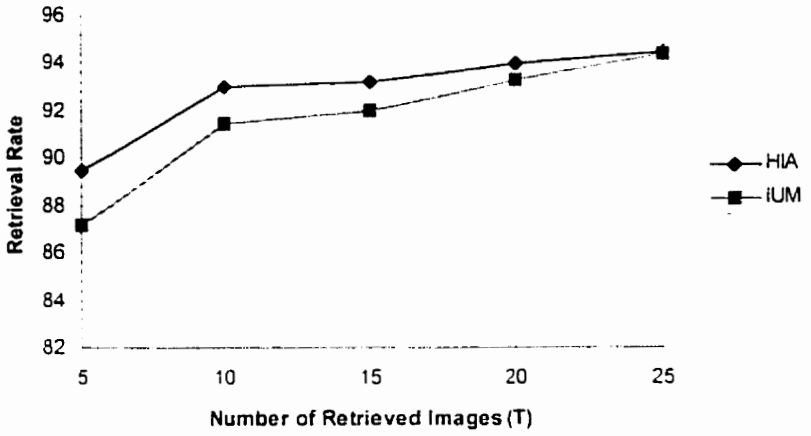
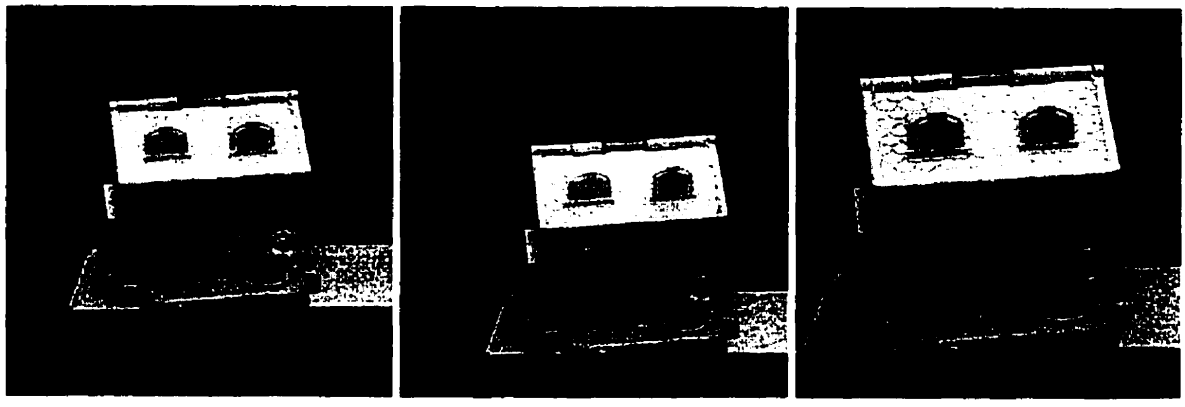


Figure 5.3: Retrieval rate as a function of the number of retrieved images.

(a)

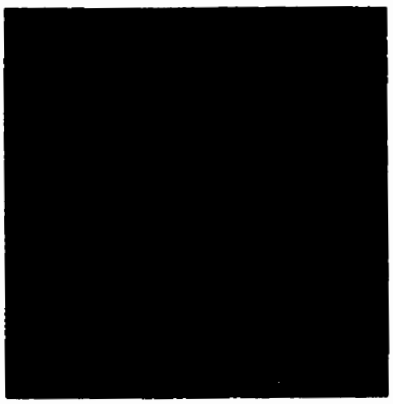


(b)

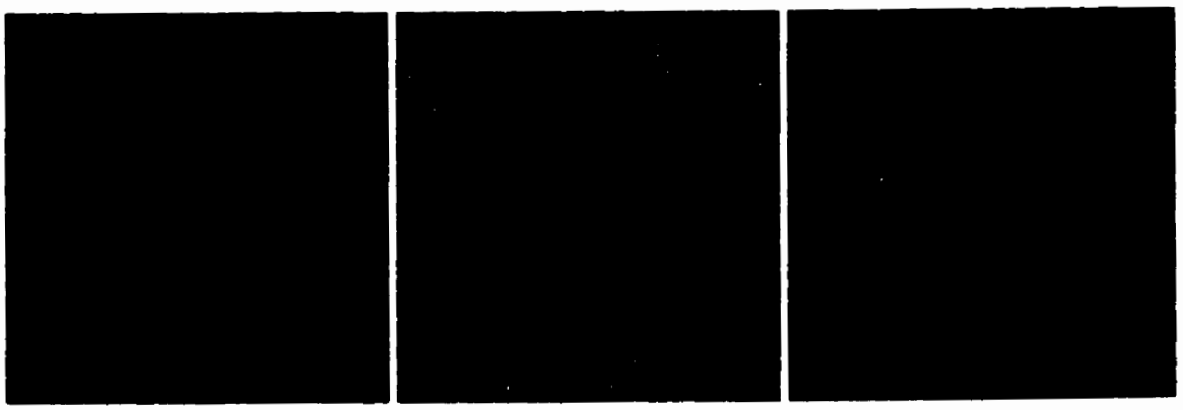
Figure 5.4: (a) Query image. (b) Retrieval results.

The number of operations and the number of bits required to calculate and store a single index in HIA and IUM for some typical image sizes are shown in Table 5.2 and Table 5.3, respectively. It can be seen from Table 5.2 and Table 5.3, that IUM has much lower computational and storage requirements. IUM requires only 32%-38% of the number of bits required to store a single index using HIA. A more serious consideration from implementation point of view is the computational complexity involved in calculating and comparing the indices. It can be seen from Table 5.2, that IUM requires 0.3%-2.5% of the

number of operations required by HIA. In addition, for index matching, HIA involves arithmetic operations (bytewise operations which includes additions and multiplications) while IUM involves only bitwise operations. Hence, the proposed technique integrates image compression and image indexing at a significantly reduced cost for computing, storing and comparing the indices making possible real time implementation.



(a)



(b)

Figure 5.6: (a) Query image. (b) Retrieval results.

Image size	HIA	IUM
256 × 256	1.7×10^5	4096
352 × 288	2.7×10^5	4096
512 × 512	6.9×10^5	4096
720 × 576	1.1×10^6	4096

Table 5.2: The number of operations/index

Image size	HIA	IUM
255 × 255	10752	4096
352 × 288	11236	4096
512 × 512	12288	4096
720 × 576	12796	4096

Table 5.3: Number of bits/index.

5.5. Summary

In this chapter, we have presented a technique which combines image compression and indexing using adaptive vector quantization. The images are first decomposed using wavelet transform followed by VQ of the transform coefficients. We note that similar images map to similar labels. Hence, the labels corresponding to an image constitute a feature vector which is used as an index to store and retrieve the image.

Simulation results have shown that the proposed technique performs comparably to the hierarchical indexing approach in terms of retrieval rates. Comparing the proposed technique to the histogram of the code words weighted by the frequency of labels (H-CL) and the histogram of labels (H-LB), it can be seen that the proposed technique integrates the index within the compressed bit stream and employs a similarity metric which only involves bitwise operations. Hence the proposed indexing technique has following advantages: (i) indices are generated at compression time, (ii) does not require additional memory for storing the indices, and (iii) provides fast access to the images stored in the database.

6

Spatio-Temporal Video Indexing

6.1 Introduction

In this chapter, we present an indexing technique for compressed video using vector quantization. Here, a video sequence is first compressed using VQ. Each frame is represented by an usage map, a set of VQ labels and a set of motion vectors. The video sequence is partitioned into shots and the various camera operations and motion within each shot are then determined by processing the VQ label maps. Each shot is indexed using a *spatio-temporal* index. The spatial index refers to the spatial content of the representative frame of a shot, while the temporal index represents the temporal content of the shot. The spatial index is based on the codewords used to compress the representative frame, while the temporal index is based on motion and camera operations within the shot. The proposed indexing technique is executed entirely in the compressed domain. This entails significant savings in computational and storage costs resulting in faster execution.

This chapter is organized as follows. First, the requirement of the proposed spatio-temporal indexing are detailed in section 6.2. The proposed algorithm for video compression

using VQ is then presented in section 6.3. The scene change detection algorithm and the generation of the spatial index are presented in sections 6.4 and 6.5, respectively. This follows in section 6.6. with the details of the proposed algorithm for the generation of the temporal index (object motion and camera operations). Finally, a summary is presented in section 6.7.

6.2 Spatio-temporal Video Indexing

Video indexing has numerous applications each with different goals. For example, in a movie industry, a film editor essentially looks for footage of a particular type of scene and/or specific camera operations from a database populated with similar shots. On the other hand, a producer in a television station interested in profiling a celebrity may require access to all the significant video clips relating to that specific celebrity. In a distance learning application, a student is interested in a specific lecture video or associated material, while in a telemedicine application a medical practitioner may be interested in the past investigations of a patient or particular examples of a disease. This points to the need for a video indexing scheme which is flexible and adaptable to the wide variety of queries in different application domains. In other words, a good video index must integrate both temporal and spatial structure of the video sequence.

We present an indexing technique for compressed video using vector quantization. Here, a video sequence is first compressed using VQ. Each frame is represented by a usage map (which indicates the subset of codewords in the codebook that were used in compressing the frame), a set of labels and a set of motion vectors as shown in Figure 6.1. The video

sequence is then partitioned into shots using a metric based on the histogram of the label. Each shot is indexed using a *spatio-temporal* index as shown in Figure 6.2. The spatial index represents the spatial content of the representative frame of a shot. We propose to employ the usage map corresponding to a representative frame of a shot as the spatial index. The temporal activity within a shot is represented by the temporal index which is essentially the motion information and camera operations within the shot as shown in Figure 6.2. In this chapter, the motion activity is detected by tracking the trajectories of the motion vectors of the labels, while camera operations are detected by analyzing the directionality of the spatio-temporal patterns of the label maps. We note that the proposed spatio-temporal index is generated entirely in the VQ compressed domain. This entails significant savings in computational and storage costs for decompression and recompression resulting in faster execution.

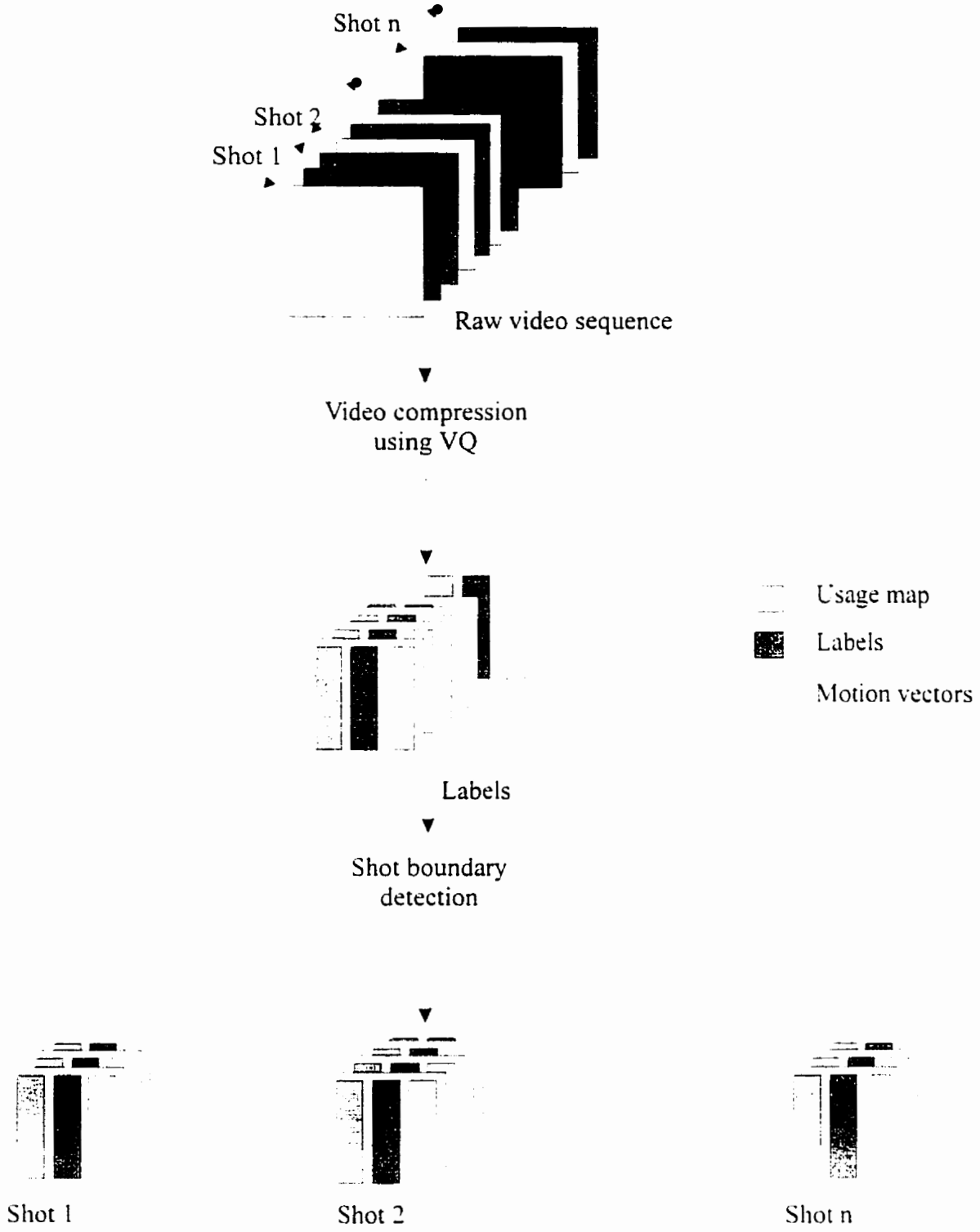


Figure 6.1: Block diagram of the proposed video indexing technique

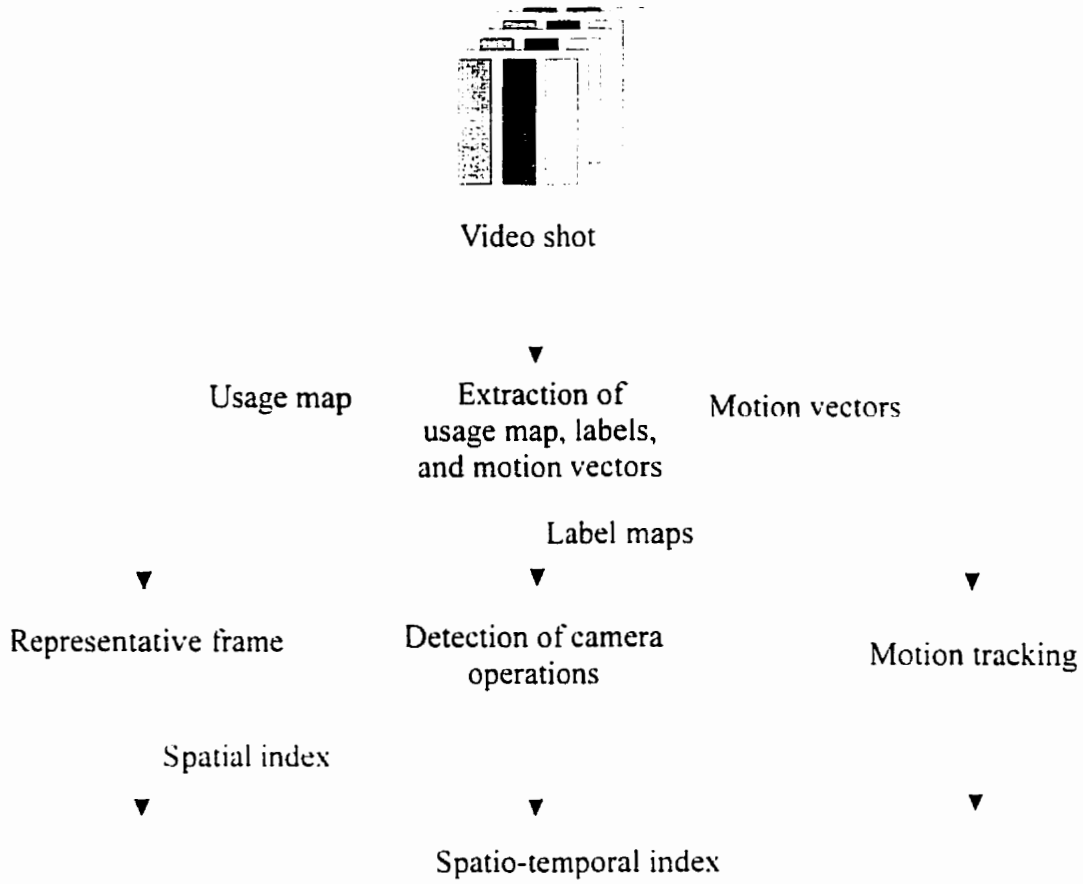


Figure 6.2: Block diagram of the proposed video indexing technique

6.3 Video Compression using VQ

In our approach, we extend the image compression algorithm, which was described in section 2.3, to video compression by exploiting the temporal redundancy in the labels of successive frames. To start with, the frame f_m to be compressed is decomposed into L-dimensional vectors. For each vector, v_i in the current frame f_m the label of the nearest codeword w_j is first determined using the universal codebook. The usage map $\{u(f_m,j);\ 0 \le j \le K\text{-}1\}$ where $u(f_m,j) \in \{0,1\}$ is updated to indicate that codeword w_j is used. The label w_j corresponding to v_i

in the current frame f_m is compared with the label at the same spatial location in the previous frame f_{m-1}. If they match, a flag S_f is transmitted to the receiver. Otherwise, a match is sought within a search area in f_{m-1}. If a match within the search area is obtained, a flag S_m followed by the displacement (motion) vector of w_j are transmitted. However, if a match is not obtained even within the search area, the label corresponding to w_j in the reduced codebook is used to encode the input vector . The algorithm can be expressed in pseudo code as follows:

Begin

Decompose the input frame f_m into L-dimensional vectors:

For each vector Do

w_j = the label of the nearest vector in the universal codebook:

update the usage map:

w_m = the label corresponding to the vector at the same spatial location of the input vector:

if (w_m matches w_j) then send a flag S_f

else

w_n = the nearest label within a search area:

if (w_j matches w_n) then send S_m followed by the motion vector:

else

send a flag S_l followed by the label corresponding to w_j in the reduced codebook:

End for

End.

We note that the codebook is arranged in the ascending order of the average and standard deviation of the codewords. By ordering the codebook, similar vectors map to neighboring labels and hence the label map of an image produces a scaled version of the image. For example, the label map of the Lena image using non-ordered and ordered codebooks are shown in Figure 6.3. The ordering of the codebook has two advantages: (i) the labels can be coded to further reduce the bit rate and, (ii) the label maps of a video sequence is used to extract the camera operations as will be discussed in section 6.3.

Computer simulations were carried out on the Miss America sequence with a frame size of 288×360 pixels and 8 bits/pixel. The sequence is obtained form "ftp://ftp.ipl.rpi.edu/image/sequence/". The coding performance of the proposed algorithm is evaluated using rate distortion criterion. For an image of size $N_1 \times N_2$ and a maximum pixel value of 255, the Peak Signal to Noise Ratio (PSNR) of the reconstructed image is calculated by:

$$PSNR = 10\log\left(\frac{255^2}{\frac{1}{N_1 N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}(X_{ij}-Y_{ij})^2}\right) \qquad \textbf{(6.1)}$$

where X_{ij} and Y_{ij} are the intensity of the pixel (i,j) in the original and the reconstructed image, respectively. The total bit rate is calculated by:

$$\frac{K + (\log_2 K + 1) \times N_l + 2 \times N_f + 5 \times N_m}{N_1 \times N_2} \qquad \textbf{(6.2)}$$

where K, N_l, N_f and N_m are the codebook size, the number of labels, the number exact matches (S_f flags), and the number of motion vectors, respectively. In our experiments, the values of the flags S_l, S_f, and S_m are={0}, {10}, {11}.

(a) (b)

Figure 6.3: Label map of the Lena image. (a) Non-ordered codebook, (b) Ordered codebook.

The bit rate and the PSNR per frame which result from applying the compression technique on every other *4th* frame of the Miss America sequence are shown in Figure 6.4 and Figure 6.5, respectively. Note that the intermediate frames are skipped to allow for larger changes between successive frames. The average bit rate and PSNR are 0.3 bpp and 33.3 dB, respectively. It can be seen from Figure 6.4, that when there are significant changes between the frames (e.g., frames 79 to 91) the bit rate increases. It can be seen from Figure 6.5, that the proposed compression technique maintains a relatively constant quality throughout the sequence. The overhead for storing the usage map is 0.0025 bit per pixel. It can be seen from Figure 6.4, that the overhead is ranges from 0.7% to 1% of the total bit rate. The overhead of the usage map as a function of codebook size for some typical images is shown in Figure 6.6.

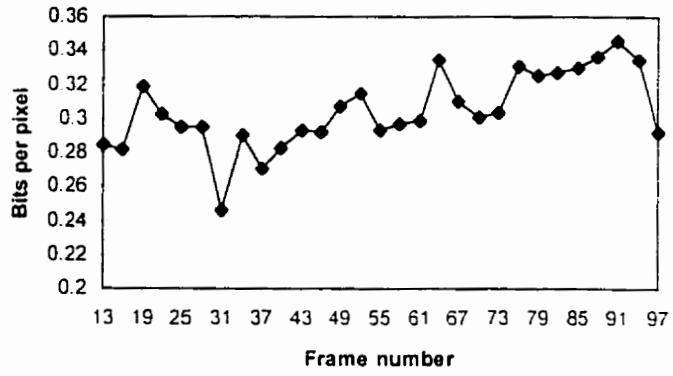


Figure 6.4: The total bit rate as a function of frame number.

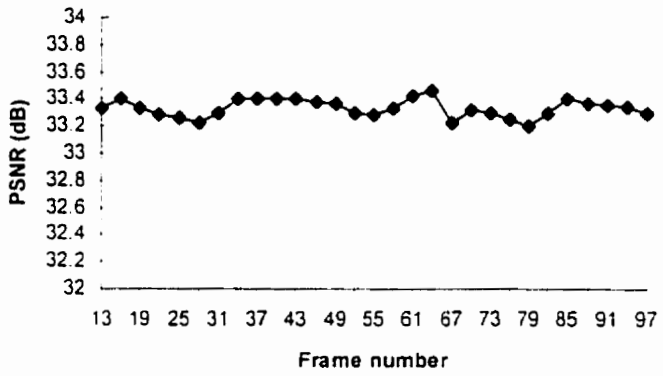


Figure 6.5: The PSNR as a function of frame number

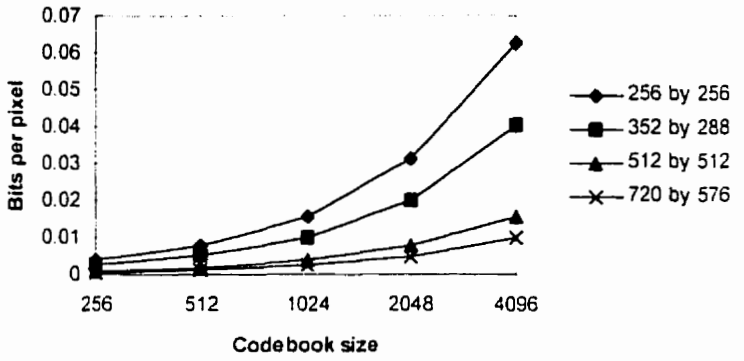


Figure 6.6: Usage map overhead in bits per pixel as a function of codebook size form some typical image sizes.

6.4 Video Segmentation

The structure within a video stems from the fact that video streams are formed by editing different video segments known as shots. A shot is a sequence of frames generated during a continuous operation and it represents continuous action in time and space [127]. Shots can be joined together in either an abrupt transition mode, in which two shots are simply concatenated, or through gradual transitions, in which additional frames may be introduced using editing operations such as dissolve, fade-in, fade-out and wipe. Furthermore, each shot might contain several clips where each clip is defined by a homogenous camera operation (e.g., zoom, tilt, ..., etc.).

The purpose of the segmentation process is to partition a video stream into a set of meaningful and manageable clips as shown in Figure 6.7, which then serve as the basic units for indexing. The segmentation process can be performed in two stages as shown in Figure

6.8. We now present the algorithm for scene change detection. The algorithm for the detection of camera operations is detailed in the next section.

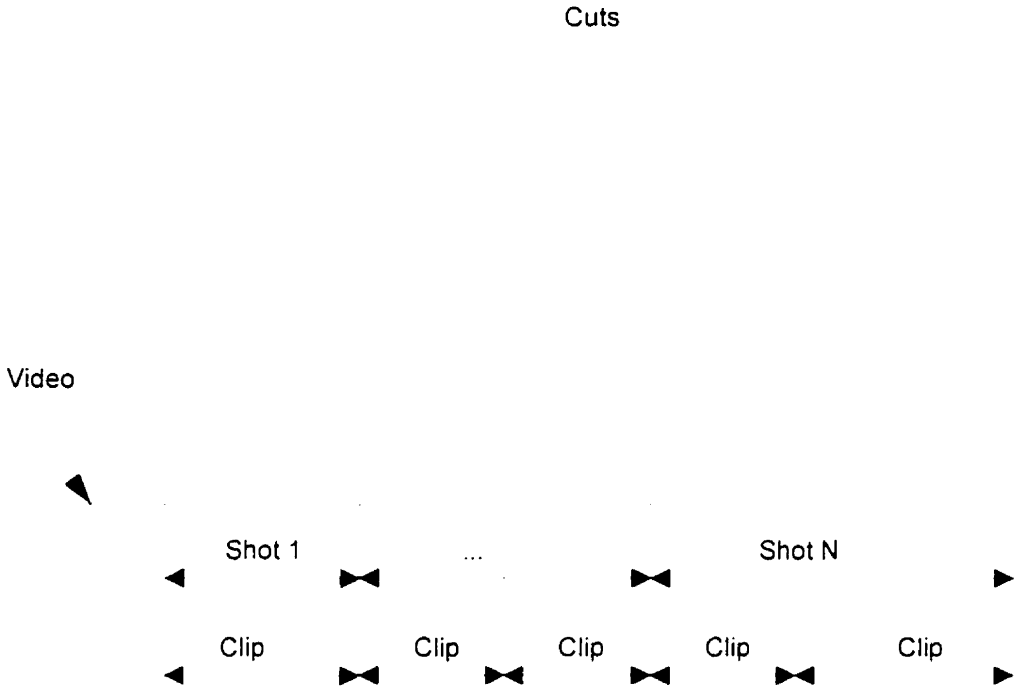


Figure 6.7: Video segment in terms of shots and clips.

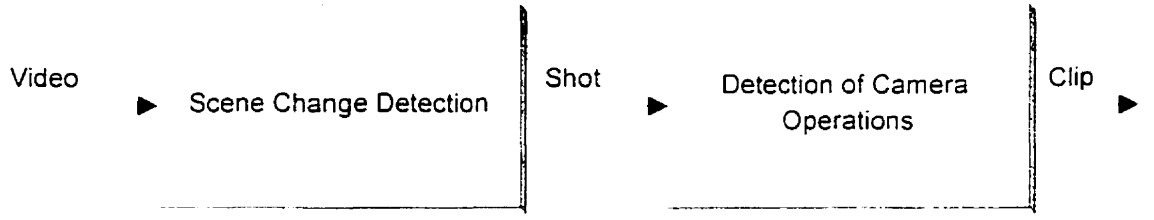


Figure 6.8: Block diagram for video segmentation.

6.4.1 Scene Change Detection

Video has both spatial and temporal dimensions, and hence a good video index should capture the spatio-temporal content of the scene. We recall that, in order to achieve this, a video is first segmented into elemental scenes called shots. Shots can be joined together in either an abrupt transition mode in which two shots are simply concatenated, or through gradual transitions, in which additional frames may be introduced using editing operations such as dissolve fade-in, fade-out and wipe. In general, video segmentation is achieved by employing a difference metric to measure the changes between two frames. A scene change is declared if the difference between the two frames exceeds certain threshold.

The histogram of the labels of a frame f_m is the K-dimensional vector $[H(f_m,i) : i=1, 2, \ldots, K]$, where $H(f_m,i)$ is the number of labels i in the compressed frame and K is the number of codewords in the codebook. The difference between two frames f_m and f_n is measured using the χ^2-metric :

$$d(f_m, f_n) = \sum_{i=1}^{K} \frac{\left(H(f_m,i) - H(f_n,i)\right)^2}{\left(H(f_m,i) + H(f_n,i)\right)} \qquad \textbf{(6.3)}$$

A large value of $d(f_m,f_n)$ indicates that f_m and f_n belong to different scenes. An abrupt scene change is declared if the difference between two successive frames exceeds a threshold. A gradual transition is detected if the difference between the current frame and the first frame of the present shot is larger than a threshold.

Simulations were executed using three music video sequences. We refer to the video sequences as "S1", "S2" and "S3". Each sequence has a frame size of 120 × 160 pixels. The first sequence, S1, has 201 frames and contains 7 abrupt scene changes. The second sequence

S2 has 201 frames and contains 21 cuts and has many special effects. The third sequence, S3 has 500 frames and contains a total of 13 gradual scene changes with very smooth transitions.

In the first experiment, the video sequences were compressed as described in section 2, using a codebook of 256 codewords and 16-D vectors corresponding to a compression ratio of 16:1. Segmentation results are tabulated in Table 6.1. We note that N_d, N_m and N_f are the number of detected, missed and false cuts, respectively. It can be seen from Table 6.1 that one cut is missed and there are some expected false alarms. We note that the largest number of false alarms is for the sequence S2. This due to the fact that S2 is characterized by a large number of special effects and camera operations.

Sequence	N_d	N_m	N_f
S1	7	0	1
S2	21	0	4
S3	12	1	2

Table 6.1: Scene change detection results using VQ at a compression ratio of 16:1.

In the second experiment, the video sequences were compressed using VQ (section 3) at a compression ratio of 64:1 using a codebook of size 256 and 64-D vectors. Detection results are tabulated in Table 6.2. Comparing Table 6.1 and Table 6.2, it can be seen that at a compression ratio of 64:1, there are few misses and the number of false alarms increases.

Sequence	N_d	N_m	N_f
S1	7	0	2
S2	19	2	6
S3	11	2	5

Table 6.2: Scene change detection results using VQ at a compression ratio of 64:1.

However, false cuts do not cause problems as the frames within such segments satisfy the requirements of a shot. Hence, the proposed algorithm has an excellent performance at both low and high compression ratios.

In the third experiment, the sequences were compressed using motion JPEG at a compression ratios of approximately 16:1 and 27:1. The χ^2-metric applied to the histogram of the DC coefficients is used for scene change detection. The detection results at a compression ratio of 16:1 and 27:1 are tabulated in Table 6.3 and Table 6.4, respectively. The number of missed cuts increases from 10% at a compression ratio of 16:1 to 24% at a compression ratio of 27:1, while the number of false cuts increases from 36% to 61%.

It can be seen from Table 6.1 and Table 6.3, that scene changes were detected at a rate of 90% in sequences compressed using VQ at a high compression ratio of 64:1, while cuts were detected only at rate of 75% in sequences compressed using motion JPEG at a lower compression ratio of 27:1. Hence, segmentation of compressed video sequences using VQ is efficient at both high and low compression ratios. However, the performance of the segmentation algorithm degrades using motion JPEG at low bit rates.

Sequence	N_d	N_m	N_f
S1	7	0	1
S2	19	2	8
S3	11	2	6

Table 6.3: Detection results using the DC coefficients at a compression ratio of 16:1.

Sequence	N_d	N_m	N_f
S1	6	1	3
S2	16	5	12
S3	9	4	10

Table 6.4: Detection results using the DC coefficients at a compression ratio of 27:1.

6.5 Spatial Index

We recall from section 6.2, that after a video sequence has been partitioned into shots, a representative frame for each shot is selected and image indexing techniques are then applied to the reference frame. VQ is a mapping from a vector in L-dimensional space into a finite set (codebook) of reproduction vectors (codewords). We note that the information conveyed by a set of quantized vectors, is also encoded in the set of codeword labels. Hence, similar images map to similar codewords. We propose to employ the usage map as feature vector to index the representative frame of a shot. We recall from section 3 that during the compression of a

frame, a usage map is generated to indicate the used codewords. We recall from chapter 5, that the reduced codebook corresponding to an image reflects the contents of the image and. The usage map corresponding to an image constitutes a feature vector which is used as an index to store and retrieve the image.

The similarity between two images f_m and f_n is measured using the following equation:

$$S(f_m, f_n) = \sum_{i=0}^{N-1} \left(u(f_m, i) \oplus u(f_n, i) \right)$$

where $\oplus$ is the XOR operation. Using this metric, the comparison of two indices requires $O(N)$ bit operations. The number of bits required to store an index is $O(N)$ bits. This technique doesn't require any additional operations to calculate the indices as the usage map is generated during the compression stage. Hence, the proposed technique provides fast access to the compressed images in the database and has lower storage requirements

Simulation were performed using approximately 500 representative images each of size 256×256 pixels. The images are compressed to form 2 databases namely: B2 and B3. The images in B2 and B3 are compressed using adaptive VQ (section 6.3) at compression ratios of 15.9:1 and 62:1, respectively.

Retrieval results are shown in Table 6.5. We note that B0 and B1 refer to the databases in which the images are indexed using the histogram of the pixels (uncompressed domain). The images in B0 are indexed using the histogram of pixels with 256 bins, while in B1 each histogram is quantized to 64 bins. Using the usage map as a spatial index at compression ratios of 15.9:1 and 62:1, the retrieval rates were 91.3% and 90.2%.

respectively. Using the histogram of the pixels as an index the images were retrieved at a rate of 88.6%.

Database	Retrieval rate
B0	88.6
B1	89.4
B2	91.3
B3	90.2

Table 6.5: Retrieval rates using histogram of pixels, and usage map on the databases B0, B1, B2 and B3, respectively.

Image size	No. of operations	No. of bits
256 × 256	1.3×10^5	4096
352 × 288	2.0×10^5	4258
512 × 512	5.3×10^5	4608
720 × 576	8.3×10^5	4777

Table 6.6: Computational and storage requirements.

The number of operations and the number of bits required to calculate and store the histogram of pixels for some typical image sizes are shown in Table 6.6. For a codebook of

size 256 with 16-dimensional vectors, the number of operations required to calculate a single index in each database is 256. It can be seen that the proposed technique has very low computational and storage requirements. It is important to note that indexing using the histogram of pixels requires arithmetic operations (additions and multiplications), while indexing using the usage map involves only bitwise operations.

6.6 Temporal Index

We recall from section 6.2, that the temporal index consists of two parts: the first represents motion activity while the second represents camera operations within a shot. In this section, the generation of the temporal index is detailed.

6.6.1 Motion

We recall from section 6.3, that during the compression of a video sequence motion estimation on each label has been performed. This information can be exploited to describe the motion within each shot. To start with, the motion vectors are used to track each label. Each track can be thought of as an n-tuple of motion vectors. The tracking operation is performed as follows: Given (x_1,y_1), the coordinate of a label in frame f_1, and the motion vector $(\Delta x, \Delta y)$ between f_1 and f_2, the coordinate of the label in frame f_2 is $(x_1+\Delta x,y_1+\Delta y)$. If during the tracking procedure the initial label moves out of its position, then we have to generate a new track for the new label whose position coincides with the coordinates of the initial label. The track of a label is represented by

$$\{(x_1, y_1),(x_2, y_2), \ldots ,(x_n, y_n), f_s, f_e\} \quad (4)$$

where $\{(x_1,y_1),(x_2,y_2), \ldots, (x_n,y_n)\}$, f_s and f_e are a set of points representing the absolute frame coordinates, number of the first frame in which the track started, number of the last frame in which the track ended.

6.6.2 Detection of Camera Operations

We recall from section 2.5, that the basic camera operations are: fixed, panning (horizontal rotation), tracking (horizontal transverse movement), tilting (vertical rotation), booming (vertical transverse movement), zooming (varying the focusing distance) and dollying (horizontal lateral movement) as shown in Figure 2.6. Camera operations include the basic operations and all the different possible combinations. Several techniques for camera motion estimation have been reported in the literature. However, we note that these techniques are generally affected by noise and have high computational complexity.

For indexing purposes it is not essential to know exactly how much pan or zoom occurred (quantitative camera parameters); the important requirement is to recognize which camera operations have occurred in a given shot. Therefore, the purpose of the proposed technique is to extract qualitative camera operations. In this section, we present a technique for the detection of camera operations. The proposed technique is less sensitive to noise and has a lower computational complexity. We start by presenting a camera model for the basic camera operations.

6.6.2.1 Camera model

A video camera projects the 3-D space onto the 2-D image plane. Using the notation in Figure 6.9, the point at coordinates (X,Y,Z) in the 3-D space is mapped onto (x,y) in the image plane. The coordinates (X,Y,Z) and (x,y) are related by the perspective transformation

$$x = \frac{FX}{Z}, \quad y = \frac{FY}{Z} \tag{6.4}$$

where F is the focal length.

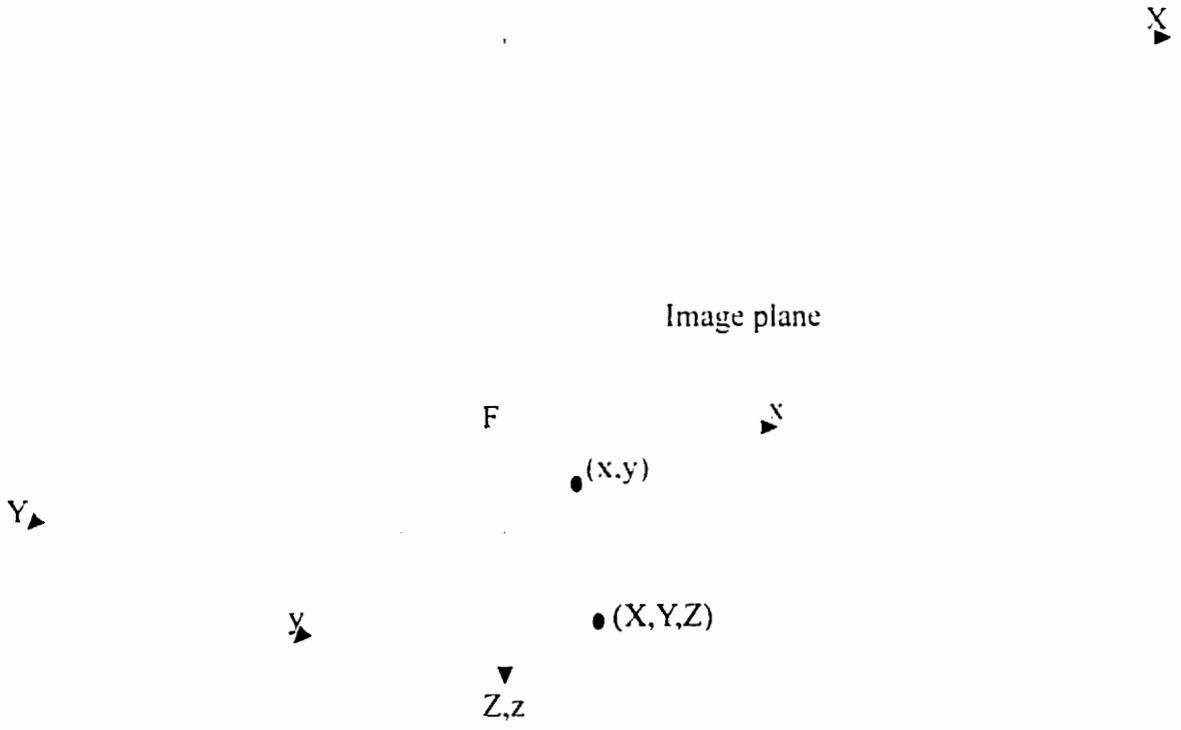


Figure 6.9: Calculation of the projected point (x,y) from the object point (X,Y,Z).

6.6.2.1.1 Zoom:

A zoom is the change of the camera focal length and results in a change which manifests itself as a radial motion towards (zoom out) or away (zoom in) from the center of the image. Let (x_1,y_1) denote the image plane coordinates of the point (X,Y,Z) with a focal length of F_1, and (x_2,y_2) is the image plane coordinates of the same point with a focal length F_2 (after the zoom). Using equation 6.4, the following relation is derived:

$$x_2 = \frac{F_2}{F_1} x_1 \qquad y_2 = \frac{F_2}{F_1} y_1 \tag{6.5}$$

We note (x_2,y_2) is independent of the distance between the camera and the object (the depth Z).

6.6.2.1.2 Pan

A pan is a rotation of the camera around the Y-axis by an angle β. Let (x_1,y_1) and (x_2,y_2) be the image plane coordinates of a point (X,Y,Z) before panning and after panning, respectively. The relation between (x_1,y_1) and (x_2,y_2) can be expressed as follows:

$$x_2 = \frac{x_1 \cos\beta + F \sin\beta}{-\frac{x_1}{F}\sin\beta + \cos\beta} \qquad y_2 = \frac{y_1}{-\frac{x_1}{F}\sin\beta + \cos\beta} \tag{6.6}$$

Assuming the value of β is small and $x_1/F << 1$, equation 6.6 reduces to:

$$x_2 = x_1 + F\beta \qquad y_2 = y_1 \tag{6.7}$$

Hence a pan operation results in a shift by a constant amount.

6.6.2.1.3 Tracking

A tracking operation along the horizontal axis results in a horizontal shift of the image. The shift depends on the distance of the objects from the camera (depth). However, for the purpose of indexing, the tracking effect can be approximately considered to be identical to that of the pan operation.

6.6.2.1.4 Tilt

A tilt is a rotation of the camera around the X-axis by an angle α. Let (x_1,y_1) and (x_2,y_2) be the image plane coordinates of a point (X,Y,Z) before tilting and after tilting, respectively. The relation between (x_1,y_1) and (x_2,y_2) can be expressed as follows:

$$x_2 = \frac{x_1}{-\frac{y_1}{F}\sin\alpha + \cos\alpha} \quad y_2 = \frac{y_1\cos\alpha + F\sin\alpha}{-\frac{y_1}{F}\sin\alpha + \cos\alpha} \tag{6.8}$$

Assuming a small value of α and $y_1/F << 1$, equation 6.8 reduces to:

$$x_2 = x_1 \quad y_2 = y_1 + F\alpha \tag{6.9}$$

Hence a tilt operation results in a vertical shift by $F\alpha$.

6.6.2.1.5 Booming

A booming operation along the vertical axis results in a vertical shift of the image. The shift depends on the distance of the objects from the camera (depth). However, to obtain qualitative information for indexing the booming effect can be approximately considered to be identical to that of the tilt operation.

Based on the previous camera model, the qualitative camera operations in a shot can be determined by analyzing the spatio-temporal patterns which is discussed in the following section.

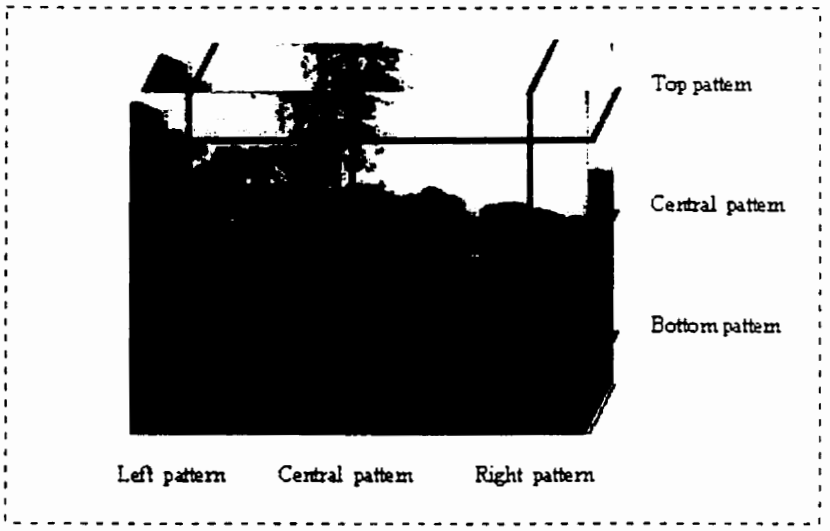


Figure 6.10: Image block.

6.6.2.2 Generation of Spatio-temporal Patterns

The proposed technique for the extraction of camera operations is based on analyzing the direction of spatio-temporal patterns. Here, a set of frames are stacked in time one after another to form a an image block as shown in Figure 6.10. A spatio-temporal pattern is a slice of the image block as shown Figure 6.10. Two types of spatio-temporal patterns are distinguished: vertical and horizontal. A vertical spatio-temporal pattern of size $x \times y$ pixels at location (m,n) is generated by first selecting from each label map the subimage of size $x \times 1$ at

(m,n). The subimages are then placed next to each other from left to right as shown in Figure 6.11. This can be expressed as follows:

$$VST(i,j) = l_j(m+i,n) \tag{6.10}$$

where *VST(i,j)* is the *(i,j)* pixel of a vertical spatio-temporal pattern and l_j is the jth label map.

A horizontal spatio-temporal pattern of size $y \times x$ pixels at location *(m,n)* is generated by first selecting from each label map the subimage of size $1 \times x$ at *(m,n)*. The subimages are then placed next to each other from top to bottom as shown in Figure 6.12. This can be expressed as follows:

$$VST(i,j) = l_j(m+i,n) \tag{6.11}$$

where *HST(i,j)* is the *(i,j)* pixel of a horizontal spatio-temporal pattern.

6.6.3 Analysis of Spatio-temporal patterns

Camera operations within a sequence can be detected by analyzing the directionality of a set of spatio-temporal patterns. Three vertical spatio-temporal patterns are selected. The first at the left side (left pattern), the second at the center (center pattern) and the third is at the right side (right pattern). Similarly, three horizontal spatio-temporal patterns (top pattern, central pattern and bottom pattern) are selected as shown in Figure 6.13. A spatio-temporal pattern can be viewed as a texture image and hence camera operations can be estimated by estimating the directionality of a set of spatio-temporal images. The directionality of a spatio-temporal pattern, can be estimated from the power spectrum of the pattern. If the directionality of a pattern is θ, then the energy is concentrated in the direction perpendicular to θ. Figure 6.14 shows a spatio-temporal pattern and its power spectrum.

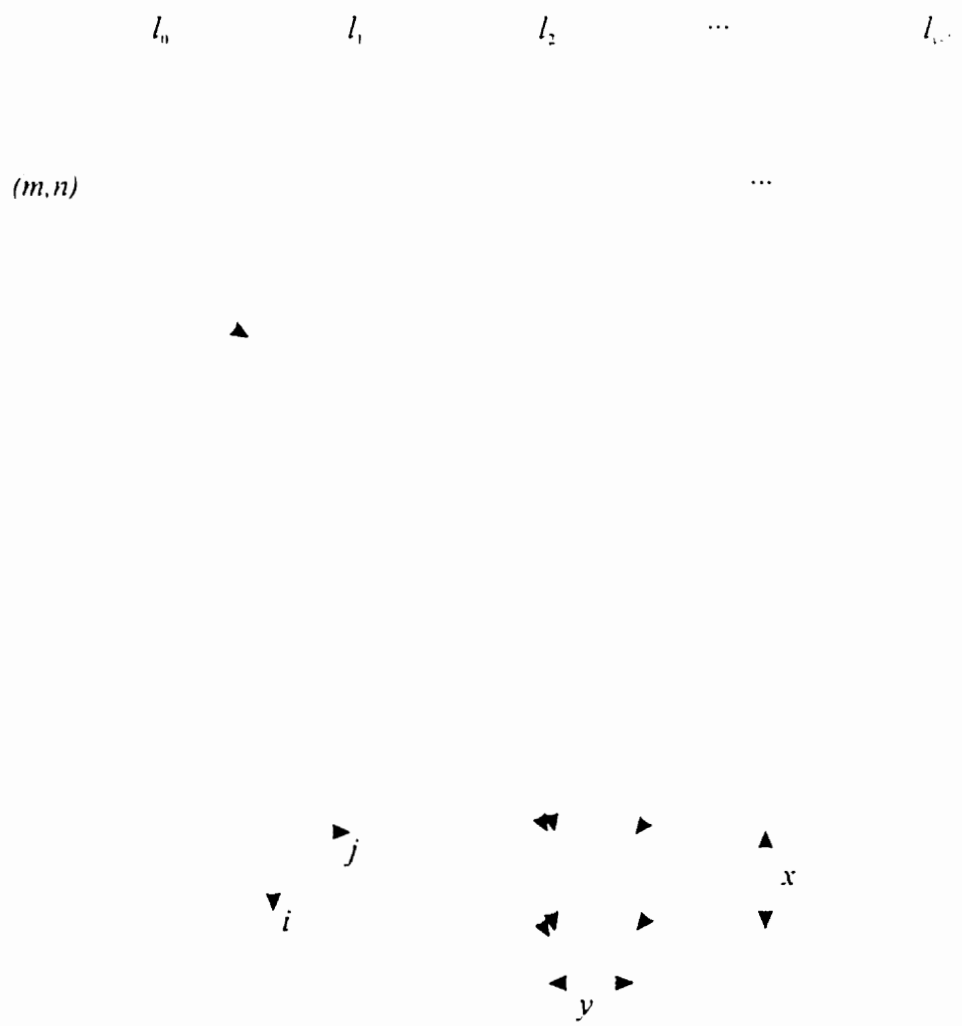


Figure 6.11: Generation of a vertical spatio-temporal pattern.

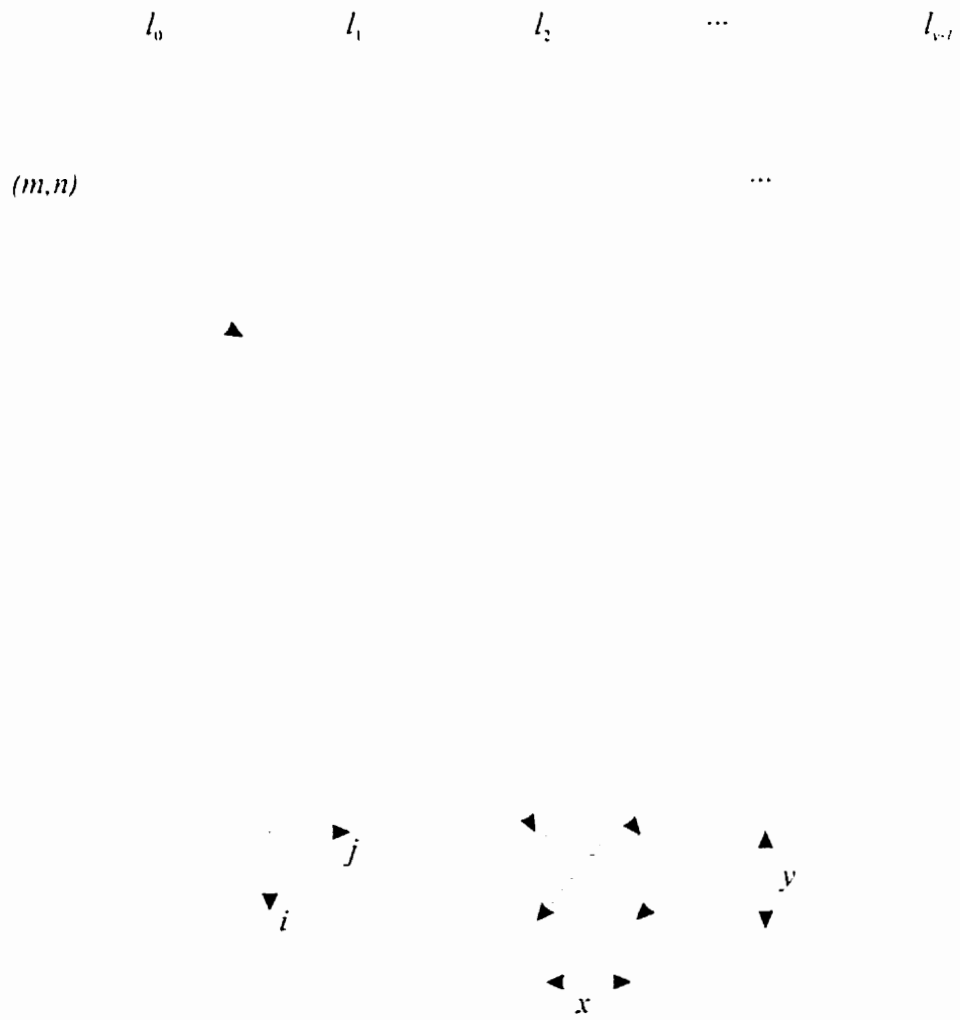


Figure 6.12: Generation of a horizontal spatio-temporal pattern.

Top pattern

Central pattern

Bottom pattern

left pattern | central pattern | right pattern

Figure 6.13: The location of horizontal and vertical patterns.

(a) (b)

Figure 6.14: Fourier spectrum of a spatio-temporal pattern. (a) pattern; (b) spectrum.

Let θ_1, θ_2, and θ_3 be the directionality of the top, central and bottom horizontal spatio-temporal patterns, respectively. Let γ_1, γ_2, and γ_3 be the directionality of the left, central and

right vertical spatio-temporal patterns, respectively. From our simulations, fixed, pan, tilt and zoom camera operations can be detected from the directionalities of the spatio-temporal patterns as shown in Tables 6.7 and 6.8. We now illustrate how pan and tilt operations are determined.

Camera operation	θ_1	θ_2	θ_3
Fixed	$\cong 90°$	$\cong 90°$	$\cong 90°$
Pan right	$\theta_1 < 90$	$\theta_1 < 90$	$\theta_1 < 90$
Pan left	$\theta_1 > 90°$.	$\theta_1 > 90°$.	$\theta_1 > 90°$.
Tilt up	Not defined	Not defined	Not defined
Tilt down	Not defined	Not defined	Not defined
Zoom in	$\theta_1 > 90°$	$\cong 90°$	$\theta_1 < 90°$
Zoom out	$\theta_1 < 90°$	$\cong 90°$	$\theta_1 > 90°$

Table 6.7: Detection of fixed, pan and zoom camera operations.

Consider the sequence which involves a pan to the right operation as shown in Figure 6.15. The sequence is compressed as described in section 6.2. Three spatio-temporal patterns of the label maps are shown in Figure 6.16. The top, central and bottom horizontal spatio-temporal patterns are generated at (6,1), (18,1) and (30,1), respectively. It can be seen from Figure 6.16. that the directionality of the three patterns are approximately the same, i.e. $\theta_1 \cong \theta_2 \cong \theta_3$ and θ_1 is less than 90°. In case of a pan to the left we note that $\theta_1 \cong \theta_2 \cong \theta_3$ and θ_1 is larger than 90°.

Camera operation	γ_1	γ_2	γ_3
Fixed	$\cong 0°$	$\cong 0°$	$\cong 0°$
Pan right	Not defined	Not defined	Not defined
Pan left	Not defined	Not defined	Not defined
Tilt up	$> 90°$.	$> 90°$.	$> 90°$.
Tilt down	$< 90°$.	$< 90°$.	$< 90°$.
Zoom in	$\theta_1 > 90$	$\cong 90°$	$\theta_1 < 90$
Zoom out	$\theta_1 < 90$	$\cong 90°$	$\theta_1 > 90$

Table 6.8: Detection of fixed, tilt and zoom camera operations.

Figure 6.15: Frames 1, 7 and 41 of the pan sequence.

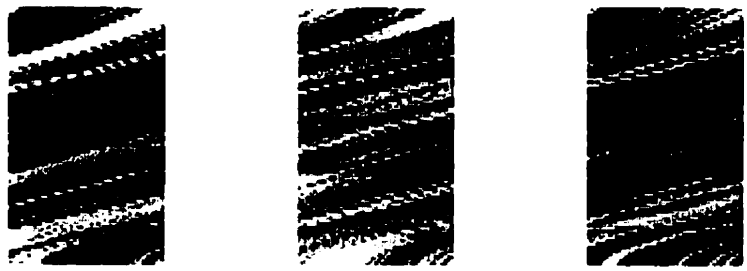


Figure 6.16: :Left, central and right horizontal spatio-temporal patterns of the sequence shown in Figure 6.15

Consider the sequence shown in Figure 6.17 which involves a tilt up camera operation. The left, center and right vertical spatio-temporal patterns are shown in Figure 6.18. It can be seen from Figure 6.18 that the directionality of the three patterns are approximately the same, i.e. $\gamma_1 \cong \gamma_2 \cong \gamma_3$ and γ_1 is larger than 90°. In case of a tilt down we note that $\gamma_1 \cong \gamma_2 \cong \gamma_3$ and γ_1 is less than 90°.

Figure 6.17: Frames 5,10 and 15 of a sequence which involves a tilt up camera operations.

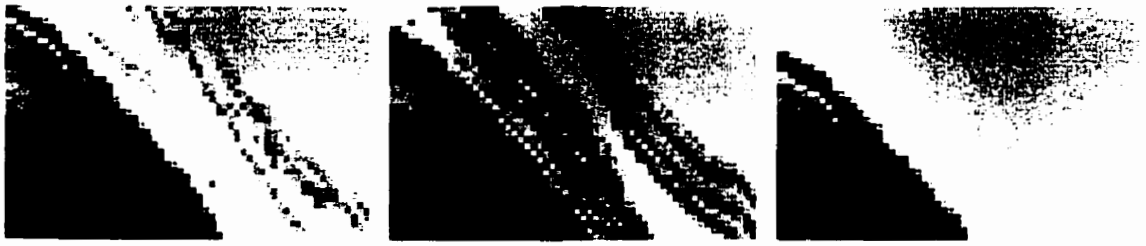


Figure 6.18: The left, central and right vertical spatio-temporal patterns corresponding to the sequence shown in Figure 6.17.

To illustrate how a zoom camera operation is detected, consider the zoom in sequence shown in Figure 6.17. The corresponding horizontal spatio-temporal patterns are shown in

Figure 6.18. It can be seen from Figure 6.18 that the first half of each pattern has a directionality less than 90°, while the second half has a directionality greater than 90°. In case of a zoom out camera operation the first half of each pattern has a directionality greater than 90°, while the second half has a directionality less than 90°.

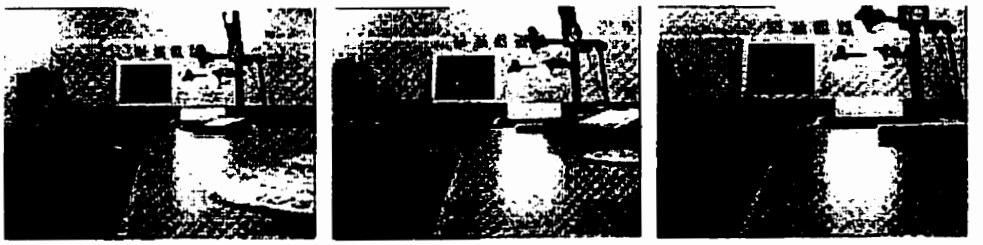


Figure 6.19: Frames 5, 30 and 60 of a zoom in sequence.

Figure 6.20: Top, central and bottom horizontal spatio-temporal patterns corresponding to the sequence shown in Figure 6.19.

The directionality of a subpattern is determined is estimated from the 2-D power spectrum.as follows. Let a spatio-temporal pattern and its DFT be represented by $f(x,y)$ and $F(u,v)$. The power spectrum of the transform. which gives the energy of the frequency (u,v), is defined as:$P(u,v)=.|F(u,v)|^2$ The amount of energy in direction α is given by $\Sigma_r P(r,\alpha)$, where $P(r,\alpha)$. is $P(u,v)$ expressed in polar coordinates. If the of $\Sigma_r P(r,\alpha)$ has a peak at γ, then spatio-temporal pattern has a directionality perpendicular to γ.

Simulations performed on four video sequences $S_{camera1}$, $S_{camera2}$, $S_{camera3}$ and $S_{camera4}$. The sequences $S_{camera1}$, $S_{camera2}$, $S_{camera3}$ and $S_{camera4}$ composed of 10, 16, 20 and 22 camera operations, respectively. In addition, $S_{camera1}$, $S_{camera2}$, and $S_{camera3}$ do not have large moving objects that dominate the scene. The sequences are compressed using a codebook of size 256 codewords and a 16-dimensional vectors.

Detection results are shown in Table 6.9. It can be seen from Table 6.9, that for the sequences $S_{camera1}$, $S_{camera2}$, and $S_{camera3}$ the all camera operations in the sequences are detected, while for $S_{camera4}$ only 2 operations are missed. It can be also seen from Table 6.9 that number of false detections ranges from 12.5%-60%. We note that is high in sequences which contain moving objects. Hence, the proposed algorithm has an excellent performance. We recall that here camera operations are detected from label maps which results in low computational complexity. In addition, the proposed technique is less sensitive to camera vibration and flash noise, since the detection process is based on a set of frames rather than individual frames.

Sequence	N_d	N_m	N_f
$S_{camera1}$	10	0	3
$S_{camera2}$	16	0	2
$S_{camera3}$	20	0	12
$S_{camera4}$	22	2	5

Table 6.9: Number of detected (N_d), missed, (N_m) and false detected (N_f) camera operations.

6.7 Summary

In this chapter, we have presented an indexing technique for compressed video using vector quantization. The video sequence is partitioned into shots using a metric based on the histogram of the label maps. Each shot is indexed using a *spatio-temporal* index. The spatial index is the usage map corresponding to a representative frame of the shot. The temporal activity within a shot is essentially the motion information and camera operations within the shot.. The motion activity is detected by tracking the trajectories of the motion vectors of the labels, while camera operations are detected by analyzing the directionality of the spatio-temporal patterns of the label maps. The spatio-temporal index provide an efficient representation of the content of a video shot. In addition, it is generated entirely in the VQ compressed domain which results in significant savings in computational and storage costs.

7

Summary and Future Research Directions

1. Summary

Visual media indexing is crucial in several applications for efficient retrieval of image and video information. With the progress of multimedia technology, large amounts of visual data will be widely accessible and thus will become one of the primary sources of information, much as text is today. Whether the application is distance learning, digital libraries, interactive television, multimedia news or banking, large volumes of video data will be required to be accessed precisely and efficiently.

One of the key features for efficient, economic storage and retrieval required in a database system is efficient indexing to enable fast access to the stored data. While indexing techniques for textual data are well established to the extent that a considerable number of database systems are commercially available, there is an impending need to develop content-based indexing techniques to facilitate retrieval from a visual database.

We recall from chapter 1, that research in image and video indexing take one of two directions. The first direction is to develop indexing techniques for compressed images and video. The second direction is to develop compression algorithms that are optimized not only in coding performance (bit rate vs. quality) but also in terms of retrieval efficiency (joint compression and indexing). In this thesis, we have addressed the problem of image and video indexing using vector quantization (VQ).

In chapters 2 and 3,we have presented a comprehensive review of image and video indexing techniques in the uncompressed and compressed domains, respectively. In chapter 4, we have presented two techniques for indexing of vector quantized images. In the first technique, for each codeword in the codebook, a histogram is generated and stored along with the codeword. The summation of the histograms of the codewords weighted by the number of labels is used as an index to store and retrieve the image. In the second technique, the histogram of the labels of an image is used as an index to access the image. We have shown that the proposed techniques provide fast access to the images in a database, have lower storage requirements and combines image compression with image indexing.

In chapter 5, we have proposed a new technique for the storage and retrieval of compressed images. The proposed algorithm is applied in the wavelet transform domain. In this technique, the images are first decomposed using wavelet transform followed by adaptive vector quantization of the transform coefficients. The usage map of the codebook is used as an index for image retrieval. We have shown that the proposed technique provide fast access to the stored images and has a lower cost for computing and storing the indices compared to other techniques reported in the literature.

In chapter 6, we have presented an indexing technique for VQ compressed video. Here, the video sequence is partitioned into shots using a metric based on the histogram of the label maps. Each shot is indexed using a *spatio-temporal* index. The spatial index is the usage map corresponding to a representative frame of the shot. The temporal activity within a shot is essentially the motion information and camera operations within the shot. The motion activity is detected by tracking the trajectories of the motion vectors of the labels, while camera operations are detected by analyzing the directionality of the spatio-temporal patterns of the label maps. The spatial and temporal indices are generated entirely in the VQ compressed domain, which entails significant savings in computational and storage costs for decompression, resulting in faster execution.

2. Future Research Directions

Future research work in the area of image and video indexing using VQ can be carried out along the following directions:

- A natural level for representing visual content would be the object level (e.g., a horse, or a racing car). This will provide a hierarchical representation where image/video objects can be indexed at different levels. An important research issue is to develop techniques for object segmentation and tracking in the VQ compressed domain. We recall from chapters 4 and 6 that the label map of an image can be viewed as a scaled version of the original image. Hence, object extraction algorithms can be applied directly to the label maps, while object tracking can be implemented using motion parameters computed from label maps. We note that object segmentation and tracking are executed in the VQ domain, which eliminate the need for decompression.

- In a generic visual database system, it is impossible to foresee all possible queries *a priori*. For example, a news producer in a television station interested in profiling the leader of a country may require access to all video clips relating to that specific person. On the other hand, a film editor essentially looks for shots of a particular type of scene. It is impractical to have an attribute for each possible query (e.g., color, camera motion, etc.). Hence, he development of a generic index structure is of great interest. A generic index enables us to derive a dominant feature to perform the search operation based on the specific input query.
- The simulations for the proposed algorithm for the detection of camera operations using spatio-temporal patterns of label maps, have demonstrated that the technique is feasible and useful. We note that it is not possible to distinguish tracking from panning, and booming from tilting. Optical flow analysis of the label maps can be used in order to distinguish tracking from panning, and booming from tilting. This technique is based on the idea that if the components of the optical flow due to camera rotation and zoom are subtracted from the optical flow, the residual flow will be parallel [103].
- In the proposed techniques, we have used full-search VQ. For K input vectors, the encoding complexity of a full search VQ is $O(KLN)$ where L and N are the vector dimension and codebook size, respectively. We recall from chapter 3, that vector quantization algorithms which reduces the computational complexity have been reported. We note that the extension of the proposed techniques to other VQ algorithms such as tree-structured VQ, multi-stage VQ and classified VQ is useful and efficient. Here, it is possible to generate an index which provide a mechanism for hierarchical image and video retrieval.

References

[1] Y. Takao, S. Itoh and J. Isaka, "An Image Oriented Database System", *Database Techniques for Pictorial Applications*, Springer-Verlag, 1980, pp. 527-538.

[2] D. M.Mckeown, Jr. and D. R. Reddy, "A Hierarchical Symbolic Representation For An Image Database", *IEEE Workshop on Picture Data Description and Management*, 1977, pp. 40-44.

[3] N. S. Jayant and P. Noll, "Digital Coding of Waveforms: Principles and Applications to Speech and Video", Prentice Hall, 1984.

[4] A. K. Jain, "*Fundamentals of Digital Image Processing*", Prentice Hall, Englwood Cliffs, New Jersey, 1989.

[5] R. Gray, "Source Coding Theory", Kluwer Academic Publisher, 1990.

[6] K. R. Rao and P. Yip, "Discrete Cosine Transform Algorithms, Advantages and Applications", Academic Press, Second Edition, 1990.

[7] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1991.

[8] Y. Linde, A. Buzo and R. Gray, "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communications*, Vol. COM-28, pp. 84-95, January 1980.

[9] J. Shanbehzadeh and P. O. Ogunbona, "On the Computational Complexity of the LBG and PNN Algorithms", *IEEE Transactions on Image Processing*, Vol. 6, No. 4, April 1997, pp. 614-616.

[10] C. F. Barnes, S. A., Rizvi, and N. M. Nasrabadi, "Advances in Residual Vector Quantization: A review", *IEEE Transactions on Image Processing*, Vol. 5, No. 2, February 1996, pp. -226-262.

[11] P. Yu and A. N. Venetsanopoulos, "Hierarchical Multirate Vector Quantization for Image Coding", *Signal Processing: Image Communication*, Vol. 4, No. 6, November 1992, pp. 497-505.

[12] P. Yu and A. N. Venetsanopoulos, "Hierarchical Finite-State Vector Quantization for Image Coding, *IEEE Transactions on Communications*, Vol. 42, No.11, November 1994, pp. 3020-3026.

[13] M. Goldberg, P. R. Boucher and S. Shlien, "Adaptive Image Compression Using Vector Quantization", *IEEE Transactions on Communications*, Vol. COM-34, No. 8, pp. 180-187, February 1986.

[14] A. Gersho and M. Yano, "Adaptive Vector Quantization by Progressive Codevector Replacement", *Proceedings International Conference on Acoustics, Speech and Signal Processing*, pp. 133-136, March 1985.

[15] M. Barlaud, P. A. Chou, N. M. Nasrabadi, D. Neuhoff, M. J. T Smith, and J. W. Woods, "Guest Editorial: Introduction to the Special Issue on Vector Quantization", *IEEE Transactions on Image Processing*, Vol. 5, No. 2, February 1996, pp. 197-199.

[16] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using Vector Quantization for Image Processing", *Proceedings of the IEEE*, Vol. 81, No. 9, September 1993.

[17] K. Aizawa and T. S. Huang, "Model-Based Image Coding: Advanced Video Coding Techniques for Very Low Bit-Rate Applications", *Proceedings of the IEEE*, Vol. 83, No. 2, February 1995, pp. 259-271.

[18] D. Saupe and R. Hamzaoui, "A review of the Fractal Image Compression Literature", *Computer Graphics*, Vol. 28, No. 4, November 1994, pp. 268-276.

[19] M. F. Barnsley and L. P. Hurd, *Fractal Image Compression*, Wellesley, MA: AK Peters Ltd., 1993.

[20] X. Wang, E. Chan, M. Mandal and S. Panchanathan, "Wavelet Based Image Coding using Non-Linear Interpolative Vector Quantization", *IEEE Transactions on Image Processing*, Vol. 5, No. 3, March 1996, pp. 518-522.

[21] M. Ohta, M. Yano and T. Nishitani, "Entropy Coding for Wavelet Transform of Image and its Application for Motion Picture Coding", *Proceedings of Visual Communications and Image Processing '91: Visual Communications*, Vol. 1605, November 1991, pp. 456-466.

[22] J. A. Saghri, A. G. Tescher and J. T. Reagan, "Practical transform coding of multispectral imagery," *IEEE Signal Processing Magazine*, Vol. 12, No. 1, pp. 33-43, Jan 1995.

[23] S. G. Mallat, "A theory for multiresolution signal representation: the wavelet decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674-693, July 1989.

[24] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image Coding Using Wavelet Transform", *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 205-220, April 1992.

[25] G. K. Wallace, "JPEG Still Picture Compression Standard", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 31 - 45.

[26] D. L. Gall, "MPEG: A Video compression Standard for Multimedia Applications", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 59-63.

[27] M. Liou, "Overview of the p×64 kbits/s Video coding Standard", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 46-58.

[28] D. Anastassiou, "Current Status of the MPEG-4 Standardization Effort", *SPIE Proceedings: Visual Communications and Image Processing*, Vol. 2308, pp. 16-24, 1994.

[29] I. Corset, S. Jeannin and L. Bouchard, "MPEG-4: Very Low Bit Rate Coding for Multimedia Applications", *Proceedings: Visual Communications and Image Processing*, Vol. 2308, pp. 1065-1073, 1994.

[30] http://www.mpeg.org.

[31] T. H. Meng, "Low-Power Wireless Video Systems", *IEEE Communications Magazine*, Vol. 36, No. 6, June 1998, pp. 130-136.

[32] G. Salton, M.J. McGill, "*An Introduction to Modern Information Retrieval*", McGraw-Hill, New York, 1983.

[33] J.J. Fan, K.Y. Su, "An Efficient Algorithm for Matching Multiple Patterns", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 2, April 1993, pp. 339-351.

[34] H. Tamura and N. Nokoya, "Image Database Systems: A Survey", *Pattern Recognition*, Vol. 17, No. 1, 1984, pp. 29-43.

[35] Y. Niu, M. T. Ozsu and X. Li, "A Study of Image Indexing Techniques for Multimedia Database Systems", *Department of Computing Science, University of Alberta, Technical Report TR 95-19*, July 1995.

[36] G. Ahanger and T. D. C. Little, "A Survey of Technologies for Parsing and Indexing Digital Video, *Journal of Visual Communication and Image Representation*, Vol. 7, No. 1, March 1996, pp. 28-43.

[37] S. W. Smoliar and H.-J. Zhang, "Content-Based Video Indexing and Retrieval", *IEEE Multimedia*, Vol. 1, No. 2, Summer 1994, pp. 62-72.

[38] P. Aigrain, H.-J. Zhang and D. Petkovic, "Content-Based Representation and Retrieval of Visual Media: A state-of-the-Art-Review", *Multimedia Tools and Applications*, Vol. 3, No. 3, November 1996, pp. 179-202.

[39] V. N. Gudivada and V. V. Raghvan, "Content-Based Image Retrieval Systems", *IEEE Computer*, Vol. 28, No. 9, September 1995, pp. 18-22.

[40] M. J. Swain and D. H. Ballard, "Color Indexing", *International Journal of Computer Vision*", Vol. 7, No. 1, 1991, pp. 11-32.

[41] M. J. Swain, "Interactive Indexing into Image Databases", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, Vol. 1908, February 1993, pp. 95-103.

[42] Y. Gong, H. Zhang, H. Chuant and M. Sakauuchi, "An Image Database System with Content Capturing and Fast Image Indexing Abilities", *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994, pp. 121 - 130.

[43] W. Niblack, R. Barber, W. Equitz, M. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, "The QBIC Project: Querying Images By Content Using Color, Texture and Shape", *Storage and Retrieval for Image and Video Databases*, Vol. 1908, February 1993, pp. 173-187.

[44] M. Stricker and M. Orengo, "Similarity of Color Images", *Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 381-392.

[45] M. Stricker, "Bounds for the Discrimination Power of Color Indexing Techniques", *Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 15-24.

[46] A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearance", *IFIP: Visual Database Systems II*, 1992, pp. 113-127.

[47] T.-S. Chua, S.-K. Lim and H.-K. Pung, "Content-Based Retrieval of Segmented Images", *ACM Multimedia 94*, 1994, pp. 211-218.

[48] A. Vellaikal and C.-C. J. Kuo, "Content-Based Image Retrieval Using Multiresolution Histogram Representation", *Digital Image Storage and Archiving Systems*, Vol. 2602, October 1995, pp. 312-323.

[49] M. Stricker and A. Dimai, "Color Indexing with Weak Spatial Constraints", *Storage and Retrieval for Still Image and Video Databases IV*, Vol. 2670, February 1996, pp. 29-39.

[50] A. Vellaikal and C.-C. J. Kuo, "Content-Based Retrieval of Color and Multispectral Images Using Joint Spatial-Spectral Indexing", *Digital Image Storage and Archiving Systems*, Vol. 2602, October 1995, pp. 232-243.

[51] X. Wan and C.-C. J. Kuo, "Color Analysis and Quantization for Image Retrieval", *Storage and Retrieval for Still Image and Video Databases IV*, Vol. 2670, February 1996, pp. 8-16.

[52] R. Rickman and J. Stonham, "Content-Based Image Retrieval Using Color Tuple Histograms", *Storage and Retrieval for Still Image and Video Databases IV*, Vol. 2670, February 1996, pp. 2-7.

[53] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Function", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 7, July 1995, pp. 729-736.

[54] M. Tuceryan and A. K. Jain, "Texture analysis", *Handbook of Pattern Recognition and Computer Vision*, Editors: C. H. Chen, L. F. Pau and P. S. P. Wang, World Scientific, Singapore, 1993, pp. 235-276.

[55] I. M. Elfadel and R. W. Picard, "Gibbs Random Fields, Co-occurences, and Texture Modeling", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 1, January 1994, pp. 118-125.

[56] M. Unser, "Sum and Difference Histogram for Texture Classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, 1986, pp. 118-125.

[57] L. J. Guibas, B. Rogoff and C. Tomasi, "Fixed-Window Image Descriptors for Image Retrieval", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 352-362.

[58] R. W. Picard and T. Kabir, "Finding Similar Patterns in Large Image Databases", *International Conference on Acoustics, Speech and Signal Processing*, Vol. V, April 1993, pp. 161-164.

[59] R. W. Picard and F. Liu, "A New Wold Ordering for Image Similarity", *International Conference on Acoustics, Speech and Signal Processing*, Vol. V, April 1994, pp. 129-132.

[60] A. Petland, R. W. Picard and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 34-47.

[61] H. Tamura, S. Mori and T. Yamawaki, "Texture Features Corresponding to Visual Perception", *IEEE Transactions on Systems Man and Cybernetics*, Vol. SMC-8, No. 6, June 1978, pp. 460-473.

[62] P. Brodatz, "Textures: A Photographic Album for Artists and Designers", Dover, New York, 1966.

[63] W. Y. Ma and B. S. Manjunath, "Image Indexing Using a Texture Dictionary", *Digital Image Storage and Archiving Systems*, Vol. 2606, October 1995, pp. 288-298.

[64] A. D. Alexandrov, W. Y. Ma, A. El Abbadi, and B. S. Manjunath, "Adaptive Filtering and Indexing for Image Databases, *Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 12-23.

[65] B. S. Manjunath and W. Y. Ma, "Texture Features for browsing and Retrieval of Image Data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, August 1996, pp. 837-842.

[66] H. Zhang and D. Zhong, "A Scheme for Visual Feature Based Image Indexing", *Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 36-46.

[67] J.-L. Chen and A. Kundu, "Rotation and Gray Scale Invariant Texture Identification Using Wavelet Decomposition and Hidden Markov Model", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, February 1994, pp. 208-214.

[68] A. R. Rao, N. Bhushan, and G. L. Lohse, "The relationship between texture terms and texture images: A study in human texture perception, *Storage and Retrieval for Still Image and Video Databases IV*, Vol. 2670, February 1996, pp. 206-214.

[69] G. Taubin and David B. Coper, "Recognition and Positioning of Rigid objects using Algebraic Moment Invariants", *Proceedings of SPIE: Geometric Methods in Computer Vision*, Vol. 1570, 1991, pp. 175-186.

[70] E. Persoon and K. S. Fu, "Shape Discrimination using Fourier Descriptors", *IEEE Transactions on Systems Man and Cybernetics*, Vol. SMC-8, 1977, pp. 170-179.

[71] S. R. Dubois and F. H. Glanz, "An Autoregressive Model Approach to Dimensional Shape Classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, 1986, pp. 55-66.

[72] H. H. Chen and J. S. Su, "A Syntactic Approach to Shape Recognition", *Proceedings of the International Computer Symposium*, 1986, pp. 103-122.

[73] W. I. Grosky and Z. Jiang, "A Hierarchical Approach to Feature Indexing", *Proceedings of SPIE: Image Storage and Retrieval Systems*, Vol. 1662, February 1992, pp. 9-20.

[74] B. Scassellati, S. Alexopoulos and M. Flickner, "Retrieving Images by 2D Shape: A Comparison of Computation Methods with Human Perceptual Judgments", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 2-14.

[75] J. E. Gary and R. Mehrotra, "Shape Retrieval in Image Database Systems", *Proceedings of SPIE: Image Storage and Retrieval Systems*, Vol. 1661, February 1992, pp. 2-8.

[76] R. Mehrotra and J. E. Gary, "Similar Shape Retrieval in Shape Data Management", *IEEE Computer*, Vol. 28, No. 9, September 1995, pp. 57-62.

[77] D. Tegolo, "Shape Analysis for Image Retrieval", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 59-69.

[78] J. P. Eakins, K. Shields, and J. Boardman, "ARTISAN - a shape retrieval system based on boundary family indexing", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases IV*, Vol. 2670, February 1996, pp. 17-28.

[79] Y. H. Ang, Z. Li, and S. H.Ong, "Image Retrieval based on Multidimensional Feature Properties", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 47-57.

[80] T. Kato, T. Kurita, N. Otsu and K. Hirata, "A Sketch Retrieval Method For Full Color Image Database", *International Conference on Pattern Recognition*, September 1992, pp. 530-533.

[81] S.-K. Chang, Q.-Y. Shi and C.-W. Yan, "Iconic Indexing by 2-D Strings", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 3, May 1987, pp. 413-428.

[82] E. Jungert, "Extended Symbolic Projection as a Knowledge Structure for Image Database Systems", *4th BPRA Conference on Pattern Recognition*, Springer Verlag, March 1988, pp. 343-351.

[83] S. Chang, E. Jungert and Y. Li, "Representation an Retrieval of Symbolic Pictures using Generalized 2-D Strings", *Proceedings of SPIE: Visual Communications and Image Processing IV*", Vol. 1199, 1989, pp. 1360-1372.

[84] S. K. Chang, C. M. Lee and C. R. Dow, "A 2-D String Matching Algorithm for Conceptual Pictorial Queries", *Proceedings of SPIE: Image Storage and Retrieval Systems*, Vol. 1662, February 1992, pp. 47-58.

[85] S.-Y. Lee and F.-J. Hsu, "Spatial Reasoning and Similarity Retrieval of Images using 2-D C-String Knowledge Representation", *Pattern Recognition*, Vol. 25, No. 3, 1992, pp. 305-318.

[86] S.-Y. Lee, M.-C. Yang and J.-W. Chen, "Signature Files as a Spatial Filter for Iconic Image Database", *Journal of Visual Languages and Computing*, Vol. 3, No. 4, December 1992, pp. 373-397.

[87] T.-Y. Hou, A. Hsu and M.-Y. Chiu, "A Content Based Indexing Technique Using Relative Geometry Features", *Image Storage and Retrieval Systems*, Vol. 1662, February 1992, pp. 59-68.

[88] S.-F. Chang, "Compressed Domain Techniques for Image/Video Indexing and Manipulation", *IEEE International Conference on Image Processing*, Part 1, October 1995, pp. 314-317.

[89] M. Shneier and M. Abdel-Mottaleb, "Exploiting the JPEG Compression Scheme for Image Retrieval", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, August 1996, pp. 849-853.

[90] H. Sakamoto, H. Suzuki and A. Uemori, "Flexible Montage Retrieval for Image Data", *Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 25-33.

[91] R. C. Jain, S. N. J. Murthy and P. L.-J. Chen, "Similarity Measures for Image Databases", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases III*, Vol. 2420, February 1995, pp. 58-65.

[92] L. A. Rowe, J. S. Boreczkyand C. A. Eads, "Indexes for User Access to Large Video Databases", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 150-161.

[93] B. Holt and L. Hartwick, "Visual Image Retrieval for Applications in Art and History", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 70-81.

[94] R. Kasturi and R. Jain, "Dynamic Vision", *Computer Vision: Principles*, IEEE Computer Society Press, 1990, pp. 469-480.

[95] H. J. Zhang, A. Kankanhalli and S. W. Smoliar and S. Y. Tan, "Automatic Partitioning of Full Motion Video", *ACM Multimedia Systems*, Vol. 1, No. 1, 1993, pp. 10-28.

[96] H. J. Zhang, Y. Gong and S. W. Smoliar and S. Y. Tan, "Automatic Parsing of News Video", *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994, pp. 45-54.

[97] Y. Tonomura, "Video Handling Based on Structured Information for hypermedia Systems", *ACM Proceedings: International Conference on Multimedia Information Systems '91*, 1991, pp. 333-344.

[98] K. Otsuji, Y. Tonomura and Y. Ohba, "Video Browsing Using Brightness Data", *Visual Communications and Image Processing '91*, Vol. 1606, November 1991, pp. 980-989.

[99] K. Otsuji and Y. Tonomura, "Projection Detecting Filter for Video Cut Detection", *ACM Multimedia 93*, 1993, pp. 251-257.

[100] D. Swanberg, C.-F. Shu and R Jain, "Knowledge Guided Parsing in Video Databases", *Proceedings of SPIE:* , Vol. 1908, 1993, pp. 13-24.

[101] S. Shahraray, "Scene Change Detection and Content-Based Sampling of Video Sequences", *Digital Video Compression: Algorithms and Technologies*, Vol. 2419, February 1995, pp. 2-13.

[102] F. Arman, A. Hsu and M.-Y. Chiu, "Image Processing on Compressed Data for Large Video Databases", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, 1993, pp. 267-272.

[103] F. Arman, A. Hsu and M.-Y. Chiu, "Feature Management for Large Video Databases", *ACM Multimedia 93*, Vol. 1908, February1993, pp. 2-12.

[104] H. J. Zhang, C. Y. Low, Y. Gong and S. W. Smoliar and S. Y. Tan, "Video Parsing Compressed Data", *Proceedings of SPIE: Image and Video Processing II*, Vol. 2182, 1994, pp. 142-149.

[105] H. J. Zhang, C. Y. Low and S. W. Smoliar and S. Y. Tan, "Video Parsing and Browsing Using Compressed Data", *Multimedia Tools and Applications*, No. 1, 1995, pp. 89-111.

[106] B.-L. Yeo and B. Liu, "A Unified Approach to Temporal Segmentation of Motion JPEG and MPEG Compressed Videos", *Proceedings of the International Conference on Multimedia Computing and* Systems, May 1995, pp. 81-88.

[107] S.-F. Chang and D. G. Messerschmitt, "Manipulation and composition of MC-DCT compressed Video", *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 1, January 1995, pp. 1-11.

[108] H.-C. H. Liu and G. L. Zick, "Scene Decomposition of MPEG Compressed Video", *Digital Video Compression: Algorithms and Technologies*, Vol. 2419, February 1995, pp. 26-37.

[109] J. Meng, Y. Juan and S.-F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence", *Digital Video Compression: Algorithms and Technologies*, Vol. 2419, February 1995, pp. 14-25.

[110] J. Lee and B. W. Dickinson, "Multiresolution Video Indexing for Subband Coded Video Databases", *Proceedings of SPIE: Image and Video Processing II*, Vol. 2185, 1994, pp. 162-173.

[111] F. Arman, R. Depommier, A. Hsu and M.-Y. Chiu, "Content-Based Browsing of Video Sequences", *ACM Multimedia 94*, 1994, pp. 97-103.

[112] H. J. Zhang and S. W. Smoliar, "Developing Power Tools for Video and Retrieval", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 140-149.

[113] P. Aigrain and P. Joly, "The Automatic Real-Time Analysis of Film Editing and Transition Effects and its Applications", *Computers & Graphics*, Vol. 18, No. 1, 1994, pp. 93-103.

[114] A. Hampapur, R. Jain and T. e. Weymouth, "Production Model Based Digital Video Segmentation", *Multimedia Tools and Applications*, No. 1, 1995, pp. 9-46.

[115] A. Akutsu, Y. Tonomura, H. Hashimoto and Y Ohba, "Video Indexing Using Motion Vectors", *Proceedings of SPIE: Visual Communications and Image Processing '92*, Vol. 1818, 1992, pp. 1522-1530.

[116] A. Akutsu and Y. Tonomura, "Video Tomography: An Efficient Method for Camerawork Extraction and Motion Analysis", *ACM Multimedia 94*, pp. 349-356, 1994.

[117] M. V. Srinivasan, S. Venkatesh, and R. Hosie, "Qualitative Estimation of Camera Motion Parameters from Video Sequences".

[118] Y. Wu and D. Suter, "A Comparison of Methods for Scene Change Detection in Noisy Image Sequence, *Proceedings of Visual '96: The First International Conference on Visual Information Systems*, February 1996, pp. 459-468.

[119] A. Dailianas, R. B. Allen and P. England, "Comparison of Automatic Video Segmentation Algorithms", *Proceedings of SPIE: Integration Issues in Large Commercial Media Delivery Systems*, Vol. 2615, 1995, pp. 2-16.

[120] M. Ioka and M. Kuroka, "A Method for retrieving Sequences of Images on the basis of Motion Analysis", *Proceedings of SPIE: Storage and Retrieval Systems*, Vol. 1662, February 1992, pp. 35-46.

[121] S. Y. Lee and H. M. Kao, "Video Indexing - An Approach based on Moving Object and Track", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, February 1993, pp. 25-36.

[122] N. Dimitrova and F. Golshani, "$\mathcal{R}_\chi$ for Semantic Video Database Retrieval", *ACM Multimedia 94*, 1994, pp. 219-226.

[123] H. S. Sawhney, "Motion Video Analysis using Planar Parallax", *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases II*, Vol. 2185, February 1994, pp. 231-242.

[124] R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems", Redwood City, California, The Benjamin/Cummings Publishing Company, 1989.

[125] S.-K. Chang and A. Hsu, "Image Information Systems: Where Do Ws Go From Here?", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, October 1992, No. 5, pp. 431-441.

[126] A. Gupta, T. Weymouth and R. Jain, "Semantic Queries With Pictures: The VIMSYS model", *Proceedings VLDB'91*, 1991, pp. 69-79.

[127] G. Davenport, T. A. Smith and N. Pincever, "Cinematic Primitives for Multimedia", *IEEE Computer Graphics & Applications*, July 1991, pp. 67-74.

[128] T. Sellis, N. Roussopoulos and C. Faloutsos, "The R^+ Tree: A Dynamic Index for Multidimensional objects" *Proceedings of the 13th International Conference on Very Large Databases*, 1987, pp. 507-518.

[129] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, "The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles", *Proceedings ACM SIGMOD the International Conference on the Management of Data*, May 1990, pp. 322-331.

[130] I. Gargantini, "An Effective Way to Represent Quadtrees", *Communications of the ACM*, Vol. 25, No. 12, 1982, pp. 905-910.

[131] J. Nievergelt, H. Hinterberger and K. C. Sevcik, "The Grid File: An Adaptable Symetric Multikey File Structure", *ACM Transactions on Database Systems*, Vol. 9, No. 1, March 1984, pp. 38-71.

[132] IEEE Spectrum, Special Issue on Interactive Multimedia, March 1993.

[133] T. H. Meng, B. M. Gordon, E. K. Tsern and A. C. Hung, "Portable Video-on-Demand in Wireless Communication", *Proceedings of the IEEE*, Vol. 83, No. 4, April 1995, pp. 659-680.

[134] F. Idris, "An algorithm and Architecture for Video Compression", University of Ottawa, 1993, pp. 6-35.

[135] H. S. Stone, C. S. Li, "Image matching by means of intensity and texture matching in the Fourier domain," *Proc. of SPIE*, Vol. 2670, pp. 337-349, 1996.

[136] M. Augusteijn, L. E. Clemens and K. A. Shaw, "Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 33, No. 3, pp. 616-626, May 1995.

[137] A. Celentano and V. D. Lecce, "A FFT based technique for image signature generation," *Proc. of SPIE: Storage and Retrieval for Image and Video Databases V*, Vol. 3022, pp.457-466, Feb 1997.

[138] D. L. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, pp. 831-836, Aug 1996.

[139] T. Chang and C. C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Transactions on Image Processing*, Vol. 2, No. 4, Oct 1993.

[140] J. L. Chen and A. Kundu, "Rotation and gray scale invariant texture identification using wavelet decomposition and hidden Markov model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, pp. 208-214, Feb 1994.

[141] M. K. Mandal, T. Aboulnasr and S. Panchanathan, "Image indexing using moments and wavelets," *IEEE Transactions on Consumer Electronics*, Vol. 42, No. 3, pp. 557-565, Aug 1996.

[142] K. A. Birney and T. R. Fischer, "On the modeling of DCT and subband image data for compression," *IEEE Transactions on Image Processing*, Vol. 4, No. 2, Feb. 1995.

[143] M. K. Mandal, S. Panchanathan, and T. Aboulnasr, "Image indexing using translation and scale-invariant moments and wavelets", *Proc. of SPIE: Storage and Retrieval for Image and Video Databases V*, Vol. 3022, pp. 380-389, Feb 1997.

[144] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, pp. 837-841, Aug 1996.

[145] C. E. Jacobs, A. Finkelstein and D. H. Salesin, "Fast multiresolution image querying," *Proc. of SIGGRAPH*, Los Angeles, Aug 1995.

[146] J. Z. Wang, G. Wiederhold, O. Firschein and S. X. Wei, "Wavelet-based image indexing techniques with partial sketch retrieval capability," *Proc. of the Fourth Forum on Research and Technology: Advances in Digital Libraries (ADL'97)*, Washington D.C., May 1997.

[147] P. Rashkovskiy and L. Sadovnik, "Scale, rotation and shift invariant wavelet transform," *Proc. of SPIE : Optical Pattern Recognition V*, Vol. 2237, pp. 390-401, 1994.

[148] J. S. Barbas and S. I. Wolk, "Efficient organization of large ship radar databases using wavelets and structured vector quantization," *Proc. of Asilomer Conference on Signals, Systems and Computers*, Vol. 1, pp. 491-498, 1993.

[149] A. Vellaikal, C. C. J. Kuo and S. Dao, "Content-based retrieval of remote-sensed images using vector quantization," *Proc. of SPIE*, Vol. 2488, pp. 178-189, 1995.

[150] E. L. Hall, "*Computer Image Processing and Recognition*", Academic Press, New York, 1979.

[151] F. Idris and S. Panchanathan, "Review of Image and Video Indexing Techniques", Journal of Visual Communication and Image Representation-Special Issue on Indexing, Storage and Retrieval of Images and Video, vol. 8, no. 2, June 1997, pp. 146-166.

[152] M. Mandal, F. Idris, and S. Panchanathan, "Image and Video Indexing in Compressed Domain - A Critical Review, the Journal of Image and Vision Computing - special issue on Content-based Image Indexing, March 1998.

[153] F. Idris and S. Panchanathan, "Storage and Retrieval of Compressed Images", *IEEE Transactions on Consumer Electronics*, Vol. 41, August 1995, pp. 937-941.

[154] F. Idris and S. Panchanathan, "Image and Video Indexing using Vector Quantization", Journal of Machine Vision and Applications, vol. 10, July 1997, pp. 43-50.

[155] F. Idris and S. Panchanathan, "Storage and Retrieval of Compressed Images using Wavelet Vector Quantization", Journal of Visual Languages and Computing, vol. 8, August 1997, pp. 289 - 301.

[156] F. Idris and S. Panchanathan, "Detection of Camera Operations in Compressed Video", Proc. Electronic Imaging Symposium - Storage and Retrieval for Image and Video Databases V, San Jose, California, Vol. 3022, February 1997, pp. 493-505.

[157] F. Idris and S. Panchanathan, "Spatio-Temporal Indexing of Vector Quantized Video Sequences", IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 5, October 1997, pp. 728-740.