Digital Video Compression For Software-based Real-time Applications

by

Tilaye Terrefe

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science, Graduate Department of Electrical and Computer Engineering, in the University of Toronto

© Copyright by Tilaye Terrefe 1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Digital Video Compression for Software-based Real-time Applications

Master of Applied Science 1997
Tilaye Terrefe
Department of Electrical and Computer Engineering
University of Toronto

Abstract

In this thesis, vector quantization (VQ) is investigated as a technique for source encoding of digital video data for applications where the encoding is done once but the decoding process is repeated many times. Emphasis is placed on minimizing the complexity of the decoder. A VQ based video codec is implemented in software and used to encode and decode video sequences.

Interframe encoding is demonstrated to yield additional compression with some loss in quality. The encoding of color video is also investigated. Techniques for generating colormaps to facilitate fast pixel-to-color mapping are proposed and implemented for displays with limited frame-buffers.
Acknowledgement

I would like to express my sincere gratitude for the advice, guidance and patience of my supervisor Professor Frank R. Kschischang during my graduate study.

A special thanks is due to my family, in particular my mother, whose support is an invaluable source of strength. Many friends have made my graduate studies a memorable time.

I would also like to acknowledge the financial support provided to me through an Information Technology Research Center Graduate Scholarship and the University of Toronto Open Master’s Fellowship.
## Contents

1 Introduction .......................... 1
   1.1 Background .......................... 2
   1.2 Video Compression ................. 3
       1.2.1 Lossless Techniques ............ 4
       1.2.2 Lossy Techniques ............... 5
   1.3 Motivation .......................... 7
   1.4 Organization of Thesis ............... 7

2 Video Data Compression Schemes ....... 9
   2.1 Assessment Criteria of Codecs ........ 10
   2.2 Transform Domain Techniques .......... 13
       2.2.1 Discrete Cosine Transform ....... 14
       2.2.2 Subband Coding ................. 15
   2.3 Spatial Domain Techniques .......... 15
       2.3.1 Differential Pulse Coded Modulation ........ 15
       2.3.2 Vector Quantization ............. 16
   2.4 Hybrid Coding Techniques .......... 17
   2.5 Second Generation Techniques ........ 17
   2.6 Evaluation of Techniques .......... 18
   2.7 Vector Quantization Revisited ....... 19
       2.7.1 Structure of Vector Quantizers .... 21
       2.7.2 Advantages over Scalar Quantization .... 21
       2.7.3 Optimality Conditions ............ 23
5.2.3 Skewed Lattice Colormaps ............................................. 66
5.2.4 Very Limited Search Colormaps ............................................. 67
5.3 Problems Colormaps ............................................................. 68
  5.3.1 Static Colormaps ............................................................. 68
  5.3.2 Colormap Flashing ............................................................. 70

6 Conclusion ........................................................................... 71
  6.1 Summary of Research ........................................................... 71
  6.2 Suggestion for Further Research ............................................. 72
List of Figures

1.1 A typical rate-distortion function ........................................ 6
2.1 Classification of coding algorithms ...................................... 10
2.2 A simple Vector Quantizer .................................................. 20
2.3 Voronoi Cells ............................................................... 22
2.4 scalar vs. vector quantization in 2-D .................................. 22
3.1 A Modular Software Unit .................................................... 27
3.2 An Abstract Encoder ....................................................... 28
3.3 A spatio-temporal cube ..................................................... 29
3.4 Compressed File Format .................................................... 30
3.5 Video Header ................................................................. 31
3.6 Encoder ........................................................................ 32
3.7 The frame data structure .................................................... 33
3.8 Differential Encoding ...................................................... 33
3.9 The Binary Splitting method ............................................. 36
3.10 Codebook Initialization Module ....................................... 37
3.11 Decoder ....................................................................... 41
3.12 Colormap Implementation .............................................. 44
4.1 Comparison with JPEG ..................................................... 48
4.2 Rate-Distortion Curves (block size 4x4) ............................. 49
4.3 Compression Using Differential Encoding ............................ 51
4.4 initial codebook generation and convergence .................... 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Effect of threshold $\epsilon$</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Decoder Speed</td>
<td>56</td>
</tr>
<tr>
<td>4.7</td>
<td>Effect of block shapes on decoder speed</td>
<td>58</td>
</tr>
<tr>
<td>5.1</td>
<td>Gray-scale values in the RGB color-space</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>An example of a lattice colormap</td>
<td>66</td>
</tr>
<tr>
<td>5.3</td>
<td>Colormap Performance</td>
<td>69</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Most digital video applications involve the transmission or storage of a large amount of data. To reduce the requirements in terms of bandwidth and storage, some form of data compression is often used to transmit or store digital video. A variety of data compression techniques have been developed over the last few years for various applications [1, 2]. In this thesis, digital video compression is studied for use in areas where a low-complexity decoder is desired. These arise in applications involving software-based real-time decoding of video where the encoding is typically done once while the decoding is repeated many times. Specifically, the goals of the thesis are:

- to identify a suitable encoding technique for software-based implementation.
- to provide a software implementation of the codec\(^1\).
- to measure the performance of the codec and carry out optimizations and comparisons.
- to be able to handle displays that make use of colormaps.

Applications where such a low-complexity software decoder can be used include playback of video clips from CD-ROM or other information storage and retrieval devices for entertainment, advertising and educational products.

\(^1\)a codec stands for an Coder-Decoder pair
Such a decoder will also be useful in platform independent multi-media applications such as Java. Here, applications are written by a developer for a virtual machine and are compiled into byte-codes. These codes are then transported over a network to an end-user and interpreted locally. A low-complexity software video decoder would be a useful tool as byte-codes for the virtual machine execute at a much slower speed than if they had been written for an underlying platform.

1.1 Background

Digital video typically consists of a sequence of images (called frames) that are themselves comprised of a rectangular array of numbers called pixels. These frames are formed by sampling and quantizing analog video. The most common technique used when digitizing images is raster scanning. Raster scanning converts a two dimensional image into a one dimensional waveform. Basically the image is scanned in a series of horizontal sweeps along horizontal segments called lines; at each sweep the light reflected by the image is converted to an electrical signal, the instantaneous amplitude of which represents the amount of reflected light [3]. Still pictures are usually scanned from left to right along a line and progressively line by line. This type of scanning is called sequential scanning or non-interlaced scanning. However, for pictures viewed on Cathode Ray Tube (CRT) displays where the picture flickers if the scan rate is too low, other scanning patterns have been considered to reduce the required scan rate (in pixels/sec) and video transmission bandwidth [4]. In broadcast television interlaced scanning is used to transmit the odd numbered lines during a half-frame period and then even numbered lines are transmitted during the other half-frame period.

Pixels may be gray-level pixels, which are typically quantized to 8 bits, or they may be color pixels. Color pixels consist of three values each signifying the relative intensities of the primary colors red, green and blue in the RGB color-space. These may be converted to another color-space such as the YIQ\textsuperscript{2} color space, where the luminance ($Y$) and chromaticity ($I$ and $Q$) will give an equivalent specification of the

\textsuperscript{2}used in the National Television Systems Committee(NTSC) standard
1.2 Video Compression

The growing popularity of digital video is bringing about a number of changes in the way image and video communications take place. In broadcasting, the delivery medium is separated from program content in such a way that broadcasting is evolving into a type of electronic publishing competing for subscribers. Much more flexible delivery mechanisms and a wide variety of applications can be envisioned with digital video. These include digital laser-disk, electronic camera, video-phone and video conferencing services, images and interactive video on PCs and workstations, program delivery on cables and satellites and High Definition Television (HDTV) [5].

Despite of these advantages, digital video is has not replaced analog video. The principal challenge to the switch to digital video is that the data rates involved are so high that unless a reduction in the bit rates is achieved, many of the envisaged applications can not be realized. Unlike audio whose bandwidth is about 20KHz and where a data rate of about 1.4Mbps is sufficient for a very high quality stereo sound, the data rate for video ranges from about 10Mbps for broadcast quality to hundreds of Megabits per second for HDTV\(^3\) [5].

The problem associated with such high data rates is two-fold. The first is that high capacity channels will be required for transmission. Bandwidth is a scarce resource and data rates in the order of 100Mbps or more would require a bandwidth of 20 - 50 MHz at coding efficiencies of 2 - 5 b/s/Hz [6]. Information retrieval devices have limited data transfer rates. For instance CD-ROM drives had data rates ranging from 150KBytes/s to the 600KByte/s in 1994[7]. Although advances have now pushed these rates to 1800 KBytes/s they are still insufficient for high quality raw video data retrieval.

The second major problem is that of storage. A single colour image with a resolution of 1000 \(\times\) 1000 at 24bpp will require 3MByte of storage; it will not fit on a

\(^3\)At 30 frames/s and 1440 \(\times\) 960 pixels/frame and 12bpp the data rate is about 500Mbps
high density floppy diskette.

The primary solution to both problems is data compression. Consequently, a flurry of research activities in the past decade and a half have resulted in many different image and video compression techniques. In addition to compression, fast data processing capabilities are essential at both the encoder and decoder end, especially for applications requiring real-time video.

In digital communication systems, an encoder typically consists of a source encoder and a channel encoder. Source coding can be considered separately from channel coding. This is justified by the data transmission theorem which states that a system for data compression (source coding) and error control (channel coding) can be designed separately and cascaded to to obtain a good overall system [8].

In source coding the aim is to design efficient codes assuming a noiseless channel. A code is optimal if it achieves the best possible fidelity subject to the constraints imposed on it by the channel. In theory the only constraint is the rate, however in practice, complexity may be as important as the rate. If the source alphabet and the coder's output alphabet are the same then, under the assumption a noiseless channel, the decoder can accurately recover the encoded source data; this is referred to as lossless compression. If however the source data cannot be recovered exactly, the compression scheme is known as lossy.

1.2.1 Lossless Techniques

Lossless compression techniques are characterized by the their ability to compress the source sequence in such a way that it can be decoded exactly. This is achieved by using the statistical properties of the source to remove redundancy. In information theory, which has its origins in Shannon's 1948 paper [9], a quantity called entropy for a discrete memoryless source is defined as

\[ H(X) = \sum_{x \in X} p(x) \log_2 \left( \frac{1}{p(x)} \right) \text{ bits} \]  

(1.1)
where \( \mathcal{X} \) is the alphabet and \( x \) is the symbol and \( p(x) \) is the probability (or relative frequency) of the letter \( x \). For sources with memory, the symbols are not independent of each other. Thus joint entropy is used instead. For a pair of discrete sources \( X \) and \( Y \), the joint entropy is defined as

\[
H(X, Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 \left( \frac{1}{p(x, y)} \right) \text{ bits.} \tag{1.2}
\]

A very useful interpretation of entropy is as the \textit{minimum} number of bits needed to code the source for transmission or storage. It is with the aim of achieving this theoretical benchmark, without too much complexity, that lossless compression algorithms are designed. Several such algorithms exist including Huffman coding, arithmetic coding, comma coding, and run-length coding. Huffman coding is optimal in the sense that it achieves on average, the lowest number bit per source symbol of any prefix code [10].

Despite the excellent quality achieved using lossless methods, the compression ratio rarely exceeds 3:1 for video coding [5]. This drawback limits their use to sensitive applications such as medical imaging systems.

### 1.2.2 Lossy Techniques

Lossy methods are overwhelmingly favoured for commercial, industrial and consumer applications involving images and video. Lossy compression schemes offer substantial savings on storage and bandwidth. In addition lossy schemes generally exploit the \textit{human visual system (HVS)}. For example, the human eye is more receptive of detail in luminance than in the chromaticity. Thus the chromaticity signals are often sampled at lower rates and assigned fewer bits during quantization than the luminance signal.

Schemes explicitly designed to take advantage of the HVS are sometimes referred to as \textit{second generation} compression methods. This is in contrast to the \textit{first generation techniques} that apply information theoretic approaches to source coding. The ability of the second generation techniques to achieve compressions of up to 100:1 with good quality is explained by their consideration of the features of the HVS.
Much of the image and video coding literature focuses on the encoding of the luminance signal as the color information can be encoded with little extra cost. The color information is sub-sampled in both horizontal and vertical directions. Moreover, the co-ordinate transformation from an RGB color-space to YIQ or YUV color-space results in an energy distribution where about 90% of the energy is in the luminance signal [2]. Thus the luminance signal is quantized with more bits than the chromaticity signals.

For video compression schemes, which are almost always lossy in practice, rate-distortion theory provides the general description of the trade-off involved between bit-rate and subjective quality. Rate-distortion theory evaluates the minimum rate necessary at which information needs to be transmitted in order to ensure that the average distortion does not exceed a certain value. The function specifying the relationship between the rate and the distortion is known as the rate-distortion function $R(D)$ as shown in Figure 1.1. The theory, however, is deficient as it suggests neither a practical way of compressing the data nor does it provide a subjectively meaningful evaluation of the distortion $D$.

![Figure 1.1: A typical rate-distortion function.](image-url)
1.3 Motivation

As stated earlier, the goal of this thesis is to design a codec with a very low complexity at the decoder. Such a decoder can then be implemented in software for use on desktop computers for real-time playback of compressed video.

Due to the increased interest in multi-media applications, and the growing use of digital video for education and entertainment, it is desirable to find ways of delivering real-time video on desktop computers with inexpensive decoders.

As image compression is now a somewhat mature subject, several techniques have been developed that can also be used for video compression. The requirements for a high quality image and video are being met with emergence of coding standards such as the Joint Photographic Experts Group (JPEG) and Motion Picture Experts Group (MPEG). However, the quality in these schemes is achieved at the cost of a relatively complex decoder. At present, this often makes the use of dedicated hardware necessary to deliver real-time performance at reasonable window sizes. It should be noted that the MPEG video standard aims at achieving the highest possible quality of decoded video at a given rate [11].

There is a potential advantage in decoder speed that can be gained by using other compression techniques for applications where quality or compression ratio can be traded off for the decreased complexity of the decoder.

1.4 Organization of Thesis

The rest of the thesis is organized as follows. In the next chapter, common video compression techniques and approaches are identified. The evaluation criteria for codecs are described and some common techniques are evaluated for suitability in the design of a low complexity codec.

In Chapter 3, the architecture of a codec is given. The basic encoding and decoding modules are outlined and some of the data structures and file format used for the compressed video stream are described.
In Chapter 4, a characterization of the video codec is given in terms of its compression, decoder speed and fidelity. The effect of parameters used in the encoding algorithm is investigated. Comparisons are carried out with currently used image and video coding standards.

In Chapter 5, colormap design is discussed for limited frame-buffer displays. Some colormap strategies are discussed and comparisons are given in terms of complexity of implementation and fidelity.

Chapter 6 summarizes this research and suggestions are given for improvements on the basic scheme investigated.
Chapter 2

Video Data Compression Schemes

A wide variety of techniques have been developed for the compression of digital images and video. It is difficult to classify image and video coding algorithms into distinct classes since there is a significant amount of overlap between the various techniques that exist. It has already been noted earlier that two very broad classifications exist; lossless and lossy compression techniques. Interest has focused almost exclusively on lossy techniques due to the compression ratios that are achievable with these methods. Lossy methods may also be classified into what are called first generation (also called classical) techniques where information theoretic approaches are applied to data compression; and second generation techniques where the emphasis is placed on trying to incorporate the human visual system into the encoding and decoding chain [12]. Further classification coding techniques uses the domain in which the image processing takes place. Thus there are spatial domain techniques, transform domain techniques and hybrid methods.

Video codecs are designed using one or more of these techniques to achieve bit rate reduction. Due to the diversity of these techniques, their suitability for implementation depends on a given application.

There are several features of video compressors which enable comparisons between schemes for particular applications. As is very common in most areas where several methods of achieving a given task exist, the desirability of one compression scheme over another depends on the particular application at hand. An application that
Lossless Methods | Lossy Methods
---|---
| “Classical” Methods | 2nd Generation Methods

<table>
<thead>
<tr>
<th></th>
<th>Spatial</th>
<th>Transform</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huffman</td>
<td>DPCM</td>
<td>DCT</td>
<td>BTC/VQ</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>VQ</td>
<td>SBC</td>
<td>SBC/DPCM</td>
</tr>
<tr>
<td>Run-Length</td>
<td>Morphological</td>
<td>KLT</td>
<td>SBC/VQ</td>
</tr>
<tr>
<td>Lempel-Ziv</td>
<td>Interframe VQ</td>
<td>BTC</td>
<td>Fractal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Contour/Texture</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Model based</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VPIC</td>
</tr>
</tbody>
</table>

Figure 2.1: Classification of coding algorithms.

requires two-way real time video communication would place an emphasis on high compression while another for playing real-time video from a CD-ROM may stress fidelity.

### 2.1 Assessment Criteria of Codecs

The following list introduces several assessment criteria for video codecs. Criteria that are important to our goal of achieving high frame-rates at the decoder are identified and used to evaluate available compression techniques. Ideally a video compression scheme, or any compression scheme for that matter, should have high compression ratio, high fidelity, low complexity of implementation and should be very flexible.

**Compression** Compression is among the most important parameters of performance.

The amount of compression achieved can be measured in several ways. Some of the conventions used are the compression ratio (ratio of uncompressed bits to compressed bits) and the average number of bits per pixel (bpp). Color image compression is often higher per color pixel than gray-scale image coding if accomplished in the YIQ space since the color information is subsampled. Therefore comparisons in compression ratio should be made for the same type of images.

In video coding additional compression may result by exploiting the frame-to-frame redundancy that is inherent in video signals. If this redundancy is
exploited, higher compression ratios are achievable for video sequences.

**Fidelity** Fidelity refers to the quality of the decompressed video stream. Although *subjective quality* is what ultimately matters, objective measures such as the peak signal to noise ratio (PSNR) are often used for comparison purposes. Due to the lack of good distortion measures these objective measures are not always reflective of the subjective quality. Often the major motivation in the application of second generation techniques is that they have been found to be promising in achieving very high compression ratio while maintaining good quality.

Apart from the ability to reconstruct frames as close to their original appearance as possible, the frame rates and dimension of each frame are important factors in the fidelity criteria. Hardware assisted codecs are capable of delivering good quality video although at a higher cost than their software based counter-parts.

**Scalability** Scalability, in the context of video compression, is a feature of the encoding and decoding algorithm. It is often misunderstood to include the capabilities of scalable hardware and frame dropping. Scalability signifies that the video coding algorithm compresses the video in multiple frame resolutions, multiple sets of frequency bands and multiple pixel color depth and that this is transmitted or stored in a hierarchical format [13]. Thus the compressed video streams can be decoded and displayed at different frame rates, resolutions and frame sizes and delay depending on the resources available to the decoder or encoder. It is a very desirable feature in any codec.

**Complexity** Complexity refers to the relative ease with which the compression scheme is implemented at both the encoder and the decoder. Depending on application, it may be required to have a low complexity decoder, a low complexity encoder or both. The terms *ease of encode/decode* and *cost* are often used to refer to the complexity of the codecs.
Codecs may be implemented in hardware or software. A lot of research and commercial activities are geared towards hardware and software implementations of codecs, stimulated in part by standardization efforts.

**Hardware Implementation** Hardware codecs are high quality codecs that are used for digital video encoding and decoding as a result of the inability of software codecs to achieve real time playback at reasonable quality and speed. Hardware solutions to real time compression and decode already in the market include; INMOS-A21, C-Cube CCL550 and the SGS-Thomson STV-3208 [6].

Hardware based codecs are generally more costly than software based methods and are thus, generally considered complex codecs. Programmability (scalability) of codecs is a desirable feature that allow designers to configure their own computer architectures at the system level. An example of a hardware codec that exhibits programmability is Intel's Intel750 which is based on the Digital Video Interactive (DVI) architecture and uses a VQ scheme for real time compression and decompression. The high cost of hardware codecs is another disadvantage [5].

**Software Implementation** Software codecs are not yet capable of the high quality performance delivered by their hardware counterparts. Compression may have come a long way to tackle the storage problem, but there is yet a considerable way to go in terms of the computation needed at the decoder if real-time playback at reasonable frame sizes and frame rates of about 30Hz is desired. Codecs implemented in software are considered low complexity codecs.

In applications that require real-time video on desktop computers in software, it is essential to have a very low complexity decoder. The complexity of the decoder is primarily dependent on the number of operations that the CPU must perform. However, comparisons on the complexity of decoders must be
careful not to neglect the storage device, data bus, operating system and video hardware which can all affect the performance.

**Asymmetry** Asymmetry refers to relative complexity of the encoder to that of the decoder. Typically an increase in complexity in the encoder results in increased complexity at the decoder. An important exception is vector quantization where the decoder is often much simpler than the encoder and an increase in encoder complexity need not result in an increase in decoder complexity.

**Delay** This is an important characteristic of video codecs that must be considered in their evaluation. Delay typically results from the computation of table lookups during playback for tasks such as colour conversion. A delay of more than half a second at the beginning of playback is considered unacceptable [13].

**Memory usage** Real-time decoding of video in software must operate with in the bounds of the Random Access Memory (RAM) available. Interframe coding techniques require that some frames be stored in memory. Lookup tables and codebooks must also reside in memory while the decompression and display process is carried out.

In the following sections, some of the common compression techniques that are used in video coding are discussed and evaluated for suitability according to some of the criteria above. A scheme with a low complexity decoder is desired for real-time playback performance.

### 2.2 Transform Domain Techniques

In transform coding the image is transformed from the spatial domain to another domain (e.g. spatial frequency domain) where the transform domain coefficients are coded and transmitted. Examples of transform techniques include the popular Discrete Cosine Transform (DCT), Walsh-Hadamard Transform (WHT), The Fourier Transform, the lapped orthogonal transform (LOT) and the optimum Karhunen-Loeve Transform (KLT).
2.2.1 Discrete Cosine Transform

DCT is typically performed on $8 \times 8$ or $16 \times 16$ blocks. The general form of the DCT for an $N \times N$ block is

$$Y_{mn} = \frac{4}{N^2} E_m E_n \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} X_{jk} \cos((2j+1)m\pi/2N) \cos((2k+1)/2N)$$

where

- $X_{jk}$ = are pixels
- $Y_{mn}$ = are the DCT coefficients.
- $E_m, E_n = \begin{cases} 1/\sqrt{2} & (m,n =0) \\ 1 & \text{else} \end{cases}$

The inverse DCT (IDCT) is given by a similar function

$$X_{mn} = E_m E_n \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} Y_{mn} \cos((2j+1)m\pi/2N) \cos((2k+1)/2N).$$

The DCT reduces the pixel block data to only a few non-zero coefficients. To maintain the desired entropy sometimes the frequency domain coefficients may be discarded. This is done in one of two ways. In zonal coding the higher frequency samples that are of little visual significance are discarded. However, in threshold coding coefficients below a certain threshold value are discarded.

The DCT generates good quality pictures at entropies of about 0.5bpp - 2bpp relative to an 8bpp original [6]. Combining DCT with other techniques such as temporal DPCM achieves even greater compression. The performance of the DCT has made it the compression algorithm of choice for the MPEG and JPEG standards for video compression and image compression, respectively, as defined by the ITU-T.

The drawback with such approaches for the purposes we have in mind is that the compute-intensive inverse transform operations must be carried out at the decoder.

\footnote{International Telecommunication Union - Telecommunication Standardization Sector (formerly CCITT).}
2.2.2 Subband Coding

Subband Coding (SBC) is another popular technique that is applied to video/image compression. In SBC the image/frame is divided into several frequency bands by filtering. In each band the image is sub-sampled (decimated) and and coded for transmission or storage. At the decoder, subbands are interpolated by zero padding and then filtered (synthesis filtering). The filtered outputs are then recombined to reconstruct the image.

Due to the non-ideal characteristic of filters, aliasing is a problem in all bands. Synthesis filters must be chosen so that they cancel aliasing from the decimation. This is often accomplished by using Quadrature Mirror Filters (QMF).

Subband coding has the desirable property that uniform quality is maintained in the images or frames that are coded unlike spatial domain techniques where blocking artifacts are often a problem. Subband coding can provide good quality pictures at entropies of about 1bpp [6].

Subband coding has also been applied successfully to encode video in a hierarchical fashion, exhibiting scalability [14].

2.3 Spatial Domain Techniques

2.3.1 Differential Pulse Coded Modulation

Differential Pulse Coded Modulation (DPCM) is a popular technique among a class of coding schemes called predictive coding schemes. Although the name DPCM implies modulation for transmission, this term is also used when the scheme is applied for storage purposes. DPCM is used in image/video coding by applying intra-frame pixel prediction using a weighted sum of the previous few pixels (linear prediction). The general form of the predictor is

\[ \hat{X}_n = \sum_{i=0}^{N-1} a_i X_i \]
where \( X_i \) are pixels, \( a_i \) are weights, and \( \hat{X}_n \) is the predicted value of \( X_n \). The error \( e_n = \hat{X}_n - X_n \) is then encoded and transmitted or stored. Since \( e_n \) is significantly less than \( X_n \), bit reduction can be achieved through variable length coding such as Huffman coding.

To reduce the accumulation of errors, pixels are periodically transmitted in PCM form. Bit rates of 6bpp/component can be achieved for an original colour image of 8bpp/component [6].

### 2.3.2 Vector Quantization

Vector Quantization (VQ) is a compression scheme that has an elegant mathematical description and is useful for the coding of speech, images and video. In nearly all coding schemes, the quantization is performed at individual real valued samples of the waveform or pixels. In transform techniques, for instance, the transformed coefficients are scalar quantized prior to transmission. However, according to Shannon’s rate-distortion theory, better performance is always achievable by coding vectors instead of scalars even when the source is memoryless [15].

A vector quantizer can be defined as a mapping \( Q \) of a k-dimensional Euclidean space onto a finite set of k-tuples \( Y \).

\[
Q : \mathbb{R}^k \rightarrow Y
\]

VQ can also be seen as an encoder/decoder pair that perform the functions \( Q \) and \( Q^{-1} \) respectively. For any vector \( x \) the output \( Q(x) \) is the index in the codebook \( Y \) of the vector \( \hat{x} \) that is closest \(^2\) to \( x \) in some sense. The elements of \( Y \) are called reproduction vectors or codewords. \( Y \) is also called the reproduction alphabet.

A vector quantizer is in practice applied to finite set \( S \), which is a subset of \( \mathbb{R}^k \). When a VQ is used over the set \( S \), it results in the partition of the set \( S \) into several disjoint subsets \( S_i \) called Voronoi regions. An optimal VQ is then one that minimizes

\(^2\text{A suitable distortion measure } d(x, Q(x)) \text{ is assumed.}\)
the expected distortion.

The encoder needs to transmit only the index of the closest vector in the codebook. The decoder, which has an identical codebook, then uses the index to lookup the appropriate vector. The key elements of a good vector quantization scheme are the codebook design and the codebook search. A popular vector quantizer design that is used to optimize codebooks is the Generalized Lloyd Algorithm (GLA) also known as the LBG algorithm [16].

It is important to note that the decoding process, which could be as simple as table lookup, makes vector quantization extremely attractive for low complexity decoder applications.

2.4 Hybrid Coding Techniques

Hybrid coding techniques combine spatial domain techniques with transform methods where pure transform domain techniques are too complex to carry out.

These include methods that combine DCT with predictive coding in the temporal domain, Walsh-Hadamard transform with vector quantization (WHT/VQ) and subband coding with vector quantization (SBC/VQ).

Methods that make use of transform coding along with motion compensation also fall in this category. In order to achieve the maximum benefit from the frame to frame redundancy of video it is useful to take advantage of the displacement of identical pixel blocks from one frame to the next. If such a motion can be detected then the motion vector can be transmitted instead of the whole block. The extension of hybrid transform methods to three dimensions has become a somewhat standard approach to image sequence coding at various rates [2].

2.5 Second Generation Techniques

Second generation techniques are techniques that propose more efficient representation of images than the conventional canonical form. As such, the incorporation of
HVS into the encoding and decoding process is fundamental to these schemes [12].

Visual Pattern Image Sequence Coding (VPISC) is among the second generation of coding techniques that are capable of high compression with good fidelity. VPISC is based on Visual Pattern Image Coding (VPIC) which is designed for still images in accordance with the HVS characteristics. In particular, the non-linear sensitivity to the presence of edges and variation in contrast [17] are incorporated. Segmentation based coding, contour coding and fractal coding techniques fall in this category.

Although asymmetric codecs have been designed with relatively low decoder complexity, second generation schemes do not place particular emphasis on low complexity decoding. For instance, the complexity of fractal based coders is typically higher than transform coders and vector quantizers [12].

### 2.6 Evaluation of Techniques

Basic techniques for digital video compression and the desirable features of codecs have been outlined in the previous sections. Based on this discussions, the choice of techniques for implementation can be made for our goal. It has been seen that lossless methods, while they retain the original image quality, are inadequate in terms of the compression ratios they can achieve. Lossy methods, which are more commonly used on the other hand, can deliver the required compression with some cost in terms of distortion.

The evaluation of a compression technique has to be considered within the context of a particular application. For real-time two-way communication the complexity of the encoder as well as the decoder are very important considerations. Codecs for such applications would tend to be symmetric, that is, both encoder complexity and decoder complexity are equally important. However, for applications where real-time encoding and decoding are not necessary, it may be more important to insist on a higher quality at the cost of increased complexity.

The coding techniques described earlier have been applied in various coders designed for video compression. Each has its advantages and disadvantages and part of
the reason for the availability of such a large array of options for encoding of images and video is that the particular strengths of the techniques described are application-dependent.

Spatial domain methods generally have a lower ease-of-decoding compared with transform based methods or hybrid methods. This is largely because spatial domain techniques avoid the evaluation of the inverse-transform that has to be carried out at the decoder when transform domain processing is applied. Simple DPCM does not achieve low bit rate coding performance (greater than 10:1). On the other hand, second generation techniques are excellent for low bit rate coding have relatively complex encoders and decoders [12].

Vector quantization, has the lowest complexity at the decoder and is easily amenable to interframe coding methods. At its simplest implementation, the decoder of a vector quantizer, simply needs to perform a table lookup from a decoded index. The existence of good algorithms for its design and its ability do deliver high compression ratios make it an excellent choice for implementation. VQ’s unusual property that the encoder can be made more complex at no computational cost to the decoder is an added advantage.

### 2.7 Vector Quantization Revisited

Vector quantization is a generalization of the scalar quantization to vector valued inputs. While scalar quantization is often used for analog-to-digital conversion, vector quantization is commonly used for compression of data that are already in digital form.

**Definition: 1** A Vector Quantizer (VQ) of size \( N \) and dimension \( k \) is a mapping from a vector in a \( k \)-dimensional Euclidean space to a finite set \( C \) of output points in the same Euclidean space, i.e., a map

\[
Q : \mathbb{R}^k \rightarrow C = \{c_1, c_2, \ldots, c_N\} \subset \mathbb{R}^k.
\]
The *code rate*, which measures the number of bits required to describe each vector component, is defined as

\[ r = \frac{1}{k} \left\lfloor \log_2(N) \right\rfloor. \]  

(2.1)

A vector quantizer forms a partition of \( \mathbb{R}^k \) so that

\[ \bigcup_i R_i = \mathbb{R}^k \quad \text{and} \quad R_i \cap R_j = \emptyset \quad \text{for} \ i \neq j. \]

The regions \( R_i \) are the Voronoi cells that will be represented by the code-vector \( c_i \).

For any coding scheme that maps a signal vector onto one of a set of \( N \) binary words, and reconstructs an approximate vector, there is an equivalent vector quantizer of size \( N \) that gives the same reproduction.

It is interesting to note that the encoder's function is deciding which partition a given input vector lies in; the mapping from the region to its associated codeword is then a trivial matter. Figure 2.2 illustrates the encoder/decoder model a simple vector quantizer.

Figure 2.2: A simple Vector Quantizer
2.7.1 Structure of Vector Quantizers

For purposes of analysis, a VQ may undergo a structural decomposition. As a primary decomposition, a vector quantizer may be represented as

\[ Q(x) = \sum_{i=0}^{N} c_i S_i(x) \]

where the selector function or the cell assignment function \( S_i \) is defined as

\[ S_i(x) = \begin{cases} 1 & \text{if } x \in R_i, \\ 0 & \text{otherwise.} \end{cases} \]

A further decomposition is not possible for the operation of the selector function without restrictions on the vector quantizer's partition geometry [18]. If some restrictions are applied to the partition geometry, further reductions are possible.

An important class of vector quantizers that are often taken to be synonymous with vector quantizers are Voronoi vector quantizers. This class of quantizers are completely determined by the distortion measure and the codebook. The advantage of these quantizers is that the partition geometry need not be described explicitly for implementation. The selector function's output can be found easily using the nearest-neighbour condition for vectors.

\[ R_i = \{ x : d(x, c_i) \leq d(x, c_j) \} \text{ for all } j \neq i. \]

2.7.2 Advantages over Scalar Quantization

The obvious advantage of VQ lies in the fact that it is able to exploit correlations between vectors that are coded, in ways that can not be achieved with scalar quantization methods.

Consider the case illustrated in Fig. 2.4. The first picture is a sketch of the joint pdf of two random variables. The two inputs are clearly dependent on each other.
However, the marginal pdfs are uniform for both inputs. Assuming the squared error distortion measure the scalar quantizer will choose the four values shown in the second sketch, as its code-vectors. The scalar quantizer thus will have a worst case distortion of $\frac{\sqrt{2}}{2}z \approx 1.12z$ where $z$ is shown Figure 2.4.

However, the vector quantizer, shown in the last sketch in Figure 2.4 will have a worst case distortion of $\frac{z}{\sqrt{2}} \approx 0.71z$.

Its advantage is clear for the case where, the input values are dependent on each other. More surprising is the fact that even when the components are independent of each other, the VQ out-performs the scalar quantizer.

This result can be seen from Shannon’s source coding theorem. Shannon argued
that for a sufficiently long sequence, one only needs to encode typical sequences of symbols to achieve an arbitrarily small distortion even when the source symbols are independent [9].

A geometrical explanation for the above result is that scalar quantization, when viewed as a special case of vector quantization (product VQ), forces the regions in the input to be rectangular, while in VQ arbitrary shapes are allowed. For example, it is known that hexagonal packing is better than rectangular packing in two dimensions when boundary effects are negligible [19].

A further advantage is that it allows us fractional bit-rates per dimension of the vectors while the scalar quantization needs a minimum of one bit per quantized value. The compression ratio achieved with VQ is independent of the vector dimension \( k \) for a fixed codebook size \( N \). This is an enormous advantage in applications that require substantial compression, such as image communication.

2.7.3 Optimality Conditions

Given a distortion measure and a statistical description of the input vectors, optimal vector quantizers and their design can be investigated. Beginning with the following definitions, we can proceed to state the conditions for optimality of vector quantizers.

**Definition: 2** A vector quantizer is locally optimal if every small perturbation of the code vectors does not lead to a decrease in the average distortion.

**Definition: 3** A vector quantizer is globally optimal if no other codebook can give a smaller value of average distortion.

The nearest-neighbour condition and the generalized centroid condition apply to the vector quantization. Thus the necessary conditions of optimality are:

1. Given a codebook, the optimal encoder assigns index \( i \) of the code-vector \( c_i \) to each input \( x \) if no other code-vector \( c_j \) is closer to \( x \) than \( c_i \) for all \( j \neq i \).

2. For a given partition, the code-vector \( c_i \) associated with \( R_i \) minimizes the conditional average distortion of representing a vector in \( R_i \).
3. Boundary points occur with zero probability.

The last condition is important for discrete input vectors where it is indeed the case that boundary points do occur. The region they belong to is often decided by arbitrary conventions, and if different conventions are used better quantizers can result.

2.7.4 Application to Images and Video

The theoretical treatment of VQ deals with the vectors once they were formed. The application of VQ focuses more attention on the way in which the vectors are formed and the distortion measure to use. The problem with the failure of quantitative distortion measures to reflect actual distortions of interest as well as the choice of various approaches that can be taken to form the vectors have led to a number of ways in which VQ can be applied. VQ has been successfully applied to the encoding of digitized speech for transmission over a communications channel. However, its application to the image (and video) coding problem was not reported until 1980 [20].

Implementation complexity of the encoder is a serious impediment to the application of VQ to high fidelity video coding. While it is clear that VQ can be made as good as any coding system that maps a signal to binary words, the cost involved in the design and implementation of VQ encoder for a given fidelity criterion may be too high.

In all practical applications, the vector quantizer cannot have an arbitrarily large number of codewords. The codewords have to be stored and storage medium is a scarce quantity. Moreover, the encoder has to look through the codebook and find the best match for a given input. This places limits on the vector dimension as well. The complexity of the encoding process grows exponentially with the vector dimension. The expected number of operations necessary to encode a vector (which serves as a measure of complexity), will be proportional to $kN$. For a given rate $r$,
the complexity is thus proportional to

\[ kN = k2^r. \]

This makes the complexity of real time encoding of good quality video beyond the capacity of most processors if brute force methods are attempted, when \( k \) is moderate to large.

The choice of the distortion measure is another factor to be considered when designing a VQ. The problems with the lack of suitable measures is common problem to most image coding techniques. It suffices to note that the ease of computation of the distortion measure has significant implications for the complexity of the encoder.

A straight forward application of VQ to images often leads to poor perceptual quality when a small codebook is used. And although VQ can achieve significant reductions in bit rates, this requires that large block sizes be used, which in turn implies higher dimensional vectors and may lead to segmentation artifacts in the decoded image [21].

A spatial domain application of vector quantization along with predictive coding offers very low decoding complexity. In the subsequent chapters, a VQ based video codec architecture is introduced and its performance on desktop computers is investigated. Implementation issues that arise in video coding will also be addressed.
Chapter 3

Codec Architecture and Design

We have identified in Chapter 1 that for a software implementation of digital video, a low complexity decoder is essential. Codec asymmetry and scalability are desirable features of any low-complexity codec. It is clear from the discussion in the Chapter 2 that vector quantization offers a low complexity decoder and an asymmetric implementation of the encoding/decoding process. We have also seen that there are algorithms for designing good vector quantizers. We will now give a description of the architecture, used to implement our digital video codec in software.

Software design involves several steps including the functional decomposition of the project, interface definition, and data definition [22]. Software architecture is the design of high-level partitioning of large projects into smaller sub-systems and their specifications. Software projects should be designed in accordance with basic principles of software architecture in mind. These include the principles of modularity, portability, malleable design, intellectual control and conceptual integrity [23].

Modularity involves the division of complex task into relatively independent smaller tasks sometimes referred to as modules. Modularity is the most important of the attributes of a software architecture from the point of view of making the task of software projects manageable.

Portability refers to the re-usability of the software design in different environments; both in part and as a whole.
Malleable design refers to the ease of adaptation of the project to new requirements not known at the original specification.

Intellectual control is said to be exercised over a design project if, despite its complexity, the process is well understood by those who are responsible for its correctness.

Conceptual integrity refers to the harmony, symmetry and predictability of the project. The task of using and maintaining the design should be executed with relative ease.

In a modular software unit, which we may call Specification/Design Unit, consists of a public specification of interface and behaviour, and a private (hidden) design and implementation.

![Figure 3.1: A Modular Software Unit](image)

This distinction is an important one as it allows changes in implementation detail with out affecting the rest of the modules. It is also the basis of the object-oriented design philosophy [23].

In the design of a codec the two largest modules of a codec are the encoder and the decoder. These are two separate tasks that can be carried out independently. Both the decoder and the encoder will have to perform their tasks correctly if the overall codec is to work as expected. To separate the two tasks, it is necessary to specify the interface between these two modules.
3.1 Encoder Specification

As the encoder is to perform all the compute-intensive parts of the codec, it is designed to allow a lot of flexibility to encode the video stream. The specification/design model is very convenient here since the encoder’s hidden design can be changed at any time without affecting the decoder. Thus a specification of the output data is a sufficient description of the encoder to the decoder. At the most abstract level of the encoder design, we have an a raw video stream input to the encoder and at the output results a compressed video bit stream.

![Diagram](image)

Figure 3.2: An Abstract Encoder

The input to this encoder, must be chosen among the many digital image storage formats that are available. Among the popular image formats are Joint Photographic Experts Group (JPEG), Graphical Interchange Format (GIF), Sun Raster, TIFF, and Portable Pix Map (PPM).

PGM and PPM files are organized in a simple and straightforward fashion and require minimal efforts to read. Refer to on-line reference manual pages for PGM and PPM file format specifications. The input is specified to be a concatenated sequence of PGM raw format for gray-scale video, and PPM format for color video streams. The encoder must also know the number of frames to be encoded. This allows the encoding of a subset of a large input file.

3.1.1 Vector Representation

For images, the vectors are formed by segmenting the image into blocks of adjacent pixels. Typical dimensions of the blocks are $2 \times 2$, $4 \times 4$, $2 \times 4$, and $4 \times 2$ pixels. The resulting $m \times n$ block is then stored as a vector of dimension $mn$. Such a segmentation works well for images because neighbouring pixels are usually highly correlated.
For color images that are coded in the RGB domain, the vectors can be formed in each color plane and quantized separately. It is common to transform color images in the RGB domain to the YIQ or YUV domain and encode the different planes separately. Since the luminance component (Y) in the transformed color space is more visually important (as it carries most of the energy in the signal), it is usually coded with a higher resolution than the chromaticity signals.

In video coding, the temporal information can be incorporated to the vector which is formed. A spatio-temporal cube can be formed by treating each frame as an image and grouping together blocks of the same relative location and size a few successive frames. Refer to Fig. 3.3.

![Spatio-temporal cube](image)

Figure 3.3: A spatio-temporal cube

The output from the encoder is a compressed video. Since our goal is not to combat transmission errors but instead to reduce the storage requirement within the constraint of a very low complexity decoder implementation, the data exchange format needs to be strictly adhered to by the encoder.

The output of the encoder is illustrated below in Figure 3.4. It consists of a video header as shown in Figure 3.5 and a series of frames. For notational convenience, we will use tele-type fonts when referring to data structures used in our codec throughout this chapter.

A frame consists of a frame header, an optional codebook, and a sequence of indices to a codebook. The frame header stores information about the frame to
Figure 3.4: Compressed File Format
be decoded. It specifies whether a frame is a base-frame or a differentially encoded frame with respect to another frame. It also specifies whether the frame has its own codebook or uses a codebook from a previous frame. The codebook is stored as a series of vectors that are accessed by referring to their index.

![Video Header](image)

Figure 3.5: Video Header

The encoder's task is to generate an output stream as specified in Figure 3.4 from a given input. The encoder's main tasks are

1. Read in a frame in the given format
2. Compress the frame using Vector Quantization
   (a) Generate or retrieve a codebook
   (b) Encode the frame
   (c) Compress the indices (bit-packing)
3. Write out compressed frame in appropriate format
4. If frames are left, then goto 1
The diagram of the encoder in Figure 3.6 is an expansion of the abstract encoder diagram that was shown in Figure 3.2. The main procedures that are necessary for the encoder to perform on the input frame are shown.

![Diagram](image)

**Figure 3.6: Encoder**

### 3.1.2 Input Module

As can be seen in the diagram for the encoder, an input frame is first taken from the input stream and an image is formed from the data. An image is a data structure that contains the height, the width and the pixel values of a particular still frame in a raster format.

This data is then segmented into sub-images (blocks, vectors) and the data is reorganized to show this segmentation. Here the image is stored as a frame. A frame is a data structure that has fields describing its type, dimension of vectors and the total number of vectors in the frame. It also contains pointers to a list of vectors in the frame and to a list of codeword indices associated with each vector.
A frame is classified as either a base frame or a difference frame. Difference frames are those formed from the error pixels resulting from the difference encoding of the input frame with respect to a base frame. When encoding the difference between frames, the encoder can be viewed as predictive vector quantizer where the predictor predicts that the next frame will be the same as the stored base frame. This technique is used to exploit the high degree of temporal redundancy that exists in video sequences.

Figure 3.8: Differential Encoding

The output from this unit is a frame with all fields filled in and all the vectors...
present. This frame is now ready for encoding.

3.1.3 Encoding Module

The encoding is accomplished using a vector quantizer designed by the well known LBG algorithm. The frame with all the vectors formed as described above, is the input to this module. There are a number of steps that are taken here.

- Determine which codebook to use.
  
  If a new codebook is needed then
  
  - select the codebook initialization method
  - initialize the codebook
  - run the LBG algorithm
  - save the new codebook

- Fill in the frame header information

- Encode the frame

- Save the indices and the codebook

3.1.4 An Algorithm for Vector Quantizer Design

An iterative technique by Lloyd [24] for scalar quantization was extended to the design of good vector quantizers by Linde, Buzo and Gray [16]. This algorithm is known as *The Generalized Lloyd Algorithm (GLA)* and is also referred to as the *LBG Algorithm* after Linde et al. The algorithm is as follows for the case of an unknown distribution which often arises in practice.
The Generalized Lloyd Algorithm (LBG Algorithm)

1. Initialization: Given the codebook size $N$, a distortion threshold $\epsilon \geq 0$, a distortion measure, an initial codebook $C_0$ and a training sequence $\{x_i\}$, set an iteration index $m = 1$ and an initial distortion $D_0 = \infty$

2. Partition the training sequence into regions $R_i, 1 \leq i \leq N$, using the nearest-neighbour condition.

3. Compute the average distortion $D_m$.
   If $\frac{D_{m-1} - D_m}{D_m} \leq \epsilon$ then
   $C_m$ is the final codebook
   Stop.
   else Continue

4. Find $C_{m+1} = \{\hat{c}_i\}$ where $\hat{c}_i = Cent(R_i)$ is the centroid of $R_i$
   $m \leftarrow m + 1$
   Goto 2

An important question remains as to the design of the original codebook with which the iterative algorithm starts. Since different initial codebooks lead to different performance, the method by which initial codebooks are designed is not insignificant. There are some techniques that exist for the design of the initial codebook.

Linde et al. suggested a technique called binary splitting. The technique consists of first finding the generalized centroid of all the training vectors. This centroid is then perturbed slightly to find a second vector and the two vectors are then stored in a codebook. The training set is then coded with the new codebook, effectively partitioning the training set into new regions. The code words are then replaced by the centroids of the new regions. The whole process is then applied recursively, on each partition to form more vectors until the desired codebook size is reached. Figure 3.9 illustrates the process in two dimensions.
Figure 3.9: The Binary Splitting method
Alternately, one can also take the first $N$ vectors in the training set as an initial codebook and continue with the algorithm. Random initialization and product codes have also been reported as possible solutions to the design of the initial codebook [25].

The codebook generation will start with a specification the initialization method for the initial codebook. A separate sub-module, shown in Figure 3.10 processes the request for the initialization of the codebook. The output is an initial codebook which is then used by the LBG module.

![Create Initial Codebook Diagram](image)

Figure 3.10: Codebook Initialization Module

The generation of the codebook is the most compute-intensive part of the whole codec and its complexity grows exponentially with the vector dimension.

The encoding module's input is a frame which has all its fields filled except the indices corresponding to the closest code-vectors of each vector. The output is a frame with all the necessary information fields completed.
3.1.5 Distortion Measures

A quantitative distortion measure has been assumed, in some of the discussions involving the nearest-neighbour condition and Voronoi quantizers. The most widely used distortion measures for the design of vector quantizers are known as additive distortion measures. Additive distortion measures are those in which the distortion between a vector and its approximation is the sum of the distortion \( d \) of each component and the corresponding approximation, i.e.,

\[
d(X, \hat{X}) = \sum_{i=1}^{k} d(x_i, \hat{x}_i),
\]

where \( X = (x_1, x_2, \ldots, x_k) \) and \( \hat{X} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k) \) and \( k \) is the dimension. Common examples of additive distortion measures are of the form

\[
d(X, \hat{X}) = \sum_{i=1}^{k} |x_i - \hat{x}_i|^\beta.
\]

For \( \beta = 2 \) the above reduces to the squared Euclidean distance distortion.

The selection of a distortion measure can be a difficult and often controversial problem for applications involving images and video [8]. Mathematically a distortion measure \( d(x, \hat{x}) \) is a mapping

\[
d : A \times C \rightarrow [0, \infty),
\]

where \( A \) is the input symbol alphabet and \( C \) is the output alphabet of the coder. Ideally the distortion measure \( d \) should quantify to subjective quality, be amenable to mathematical analysis and be measurable. The first requirement is often at odds with the other two and thus \( d \) is rarely meaningful, tractable and computable at the same time.

As the LBG algorithm works for a general distortion measure, distortion method must be chosen for implementation. The Squared Euclidean Distance distortion is used in the codec. This measure is easy to compute. Its major drawback is that it
does not always quantify to subjective image quality.

3.1.6 Bit Packing

The input here is a file stream, a frame and number of bits needed to represent each code vector. The output is a compressed representation of the indices to the codeword entries in the codeword associated with each vector in the frame pointed to by the pointer.

Storage efficiency is achieved by noting that the number of bits that is needed to represent the each code-vector in a codebook of size 8 is 3, while the smallest storage unit available to us in the computer is a BYTE (8 bits). Thus by packing the bits so as to avoid the transmission (storage) of the unused bits to the decoder, efficient representation can be achieved.

For instance a representation of 8 codeword entries where the codebook size is 8 with an index list 5,6,7,3,0,5,4,3 will look like this ¹:

<table>
<thead>
<tr>
<th>Integer Representation</th>
<th>Packed Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000000101</td>
<td>1100110</td>
</tr>
<tr>
<td>0000000000000110</td>
<td></td>
</tr>
<tr>
<td>0000000000000111</td>
<td>01110001</td>
</tr>
<tr>
<td>0000000000000111</td>
<td>10110101</td>
</tr>
<tr>
<td>0000000000000101</td>
<td></td>
</tr>
<tr>
<td>0000000000000100</td>
<td></td>
</tr>
<tr>
<td>000000000000011</td>
<td></td>
</tr>
</tbody>
</table>

3.1.7 Encoder Output Module

The input to this module are the various data structures that form the output format of Figure 3.4 and the file stream that the output data are to be written to.

¹The number of bytes allocated to an integer varies depending on the machine architecture.
The output is the a compressed video file that adheres to the specified output specification of Figure 3.4. The first piece of data to be written is the video_header. This is followed by the first frame_header and a codebook. Then an index-list is written in a bit-packed representation. Then the next frame information is written with strict adherence to the specification.

The output module checks the number of bytes written is correct.

3.2 Decoder Specification

We will start with a description of the entire decoder as a sub-system in the design/specification model mentioned earlier. The specification and design will be outlined in the next subsections. A similar design/specification exists for each of the modules in the decoder in Figure 3.11, and they will be described briefly.

The input to the decoder is in the format that is specified in Figure 3.4. The decoder's output is a bit-stream that written to raster display device. The decoder will decode the codebook and indices and generate a frame by performing a table-lookup operation. An outline of the decoder's task is given below:

read video_header
initialization and allocate memory for the pixels of a frame
loop
   read frame_header
   read the codebook (if any)
   read in the indices in packed form
   unpack the indices
   construct the frame
   inverse quantize codebook values (if interframe )
   add to reference frame (if interframe)
   display the image at appropriate time interval
until (all frames processed)
3.2.1 Decoder Input Module

The decoder reads several data structures and process the data contained in them. This is performed in the input module. There is a separate procedure to read each type of data structure. Memory allocation is also handled here for the data input buffering.

The first one to be read is the video header. A frame header is then read and depending on the information that this provides, the encoder reads a codebook or retrieves a previously stored codebook. The indices, which must always exist for each frame, are then read.
3.2.2 Decompression Module

Here the bit-packed index list is unpacked to produce a index list. This is used in conjunction with the codebook to construct the frame. The process is just the reverse of what is described in section 3.1.6 above. The frame is constructed with a simple table-lookup operation. Recall that our choice of vector quantization as the compression technique to be used in this codec was motivated by the simplicity the lookup operation from the codebook.

For instance, consider the simple case where we have a codebook size of 2 and vector dimension of 4 where:

\[
\text{code-vector}[0] = [0,0,0,0] \\
\text{code-vector}[1] = [1,1,2,2]
\]

and an index list

0, 0, 1, 1, 0, 1, 0, 1

The reconstructed output image is:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 2 & 2 & 0 & 0 & 2 & 2 \\
\end{bmatrix}
\]

This illustration shows what needs the input and the output of the module. In the intermediate step a frame data structure is created and the reconstructed frame needs to be turned into a image suitable for display. A procedure handles this task by performing the inverse operation of the vector formation (image segmentation) procedure done in the encoder.

If differential encoding was used then this would have to be processed further to produce the actual frame that can be used for display. The difference frame contains quantized values. Thus the difference frame values are first inverse quantized to give actual difference pixel values between the current frame and the reference frame.
These values are then added to the reference frame in memory to reconstruct the output frame for display.

The illustration in Figure 3.11 gives a good conceptual picture of the decoders task.

3.2.3 Decoder Output Module

Routines that access the display hardware are used to display reconstructed image. For the display operations, memory-mapped output is used. Once the video memory is suitably mapped, the video sequence can be displayed by writing to the mapped memory at a controlled rate. This control allows the decoder to decode at different rates.

For displays that make use of a colormap, appropriate color-map loading is essential to generate the desired output colors. Most color display devices use the RGB color model [26]. Each pixel is made up of three phosphors that are sensitive to a separate electron beam. When each phosphor is illuminated, the resulting pixel color is generated by the additive mixing of the three colors.

Since it is expensive to store all colors that are potentially available using a 24-bit RGB color scheme (it would require about 16 million colors to be stored), display hardwares often use a color map with 256 color entries. That means, the pixel value associated with a given pixel location is in fact an index to the color map as shown in Figure 3.12.

The colormap to use is specified here using techniques described in next chapter. The input to this module is an image in memory and the output is the frame on the display device.
Figure 3.12: Colormap Implementation
Chapter 4

Codec Performance and Characterization

In this chapter we characterize the codec by studying its performance and investigating some of its many features that were built into the architecture. The advantage in designing a flexible codec is that it allows various parameters to be varied. As such, it is also more difficult to optimize than a less flexible approach. As the possible combination of parameter values is increased the potential areas of investigation for possible improvements become numerous. Thus it becomes necessary to focus the investigations to areas of interest and carry out the measurements required.

In taking the measurements that pertain to one performance parameter of the codec, the state of the other related parameters need to be known to get a meaningful interpretation of data. Thus, in characterizing the compression ratio, one must take into account the level of fidelity of the decoded image. Similarly, measurements on the complexity of the decoder must be accompanied by an indication of the size of the frames being decoded. For these obvious reasons, the characterization and performance of the encoder and decoder should be considered as a whole, with reference to particular applications in mind.
4.1 Compression Vs Fidelity

Among the most important parameters of an encoder’s performance is the compression ratio. It is so important that performance is sometimes used to refer to the compression ratio. At the decoder on the other hand, it is desirable to have high fidelity (quality of the decoded frames) and high decoder speed (also known as frame rate).

An attempt to characterize the performance of the codec in terms of compression ratio, must include the fidelity of the resulting output of the codec. We must note that any the chosen scheme (in this case VQ) is necessarily lossy for most practical video sequences and thus an indication of the quality degradation is an essential requirement. We will first look at intraframe coding, followed by interframe coding.

The fidelity will be given by the peak signal to noise ratio (PSNR) measure. PSNR has gained wide acceptance in video coding literature although other measures do exist are occasionally used [2]. The PSNR is defined as

\[
PSNR = \frac{x_{max}^2}{\frac{1}{NM} \sum_{i}^{NM} (x_i - \hat{x}_i)^2}
\]

where the image is taken to be of \(N \times M\), \(x_{max}\) is maximum pixel value and \(x_i\) and \(\hat{x}_i\) are the original and reconstructed pixel values, respectively.

4.1.1 Intraframe Encoding

Intraframe coding of video is the application of a coding technique, to each frame of in the sequence independently. Intraframe coding using vector quantization, in particular the design of a vector quantizer, is investigated here to study the effects if any, of the parameters that are used in the design. The design of a vector quantizer using the LBG algorithm has a number of parameters that can vary. The dimension of the vectors, the codebook size, the distortion measure used and the method of generating the initial codebook on which the algorithm operates are not dictated by the algorithm itself.
The codebook size and the vector dimension are closely related factors in the performance of a vector quantizer. They play a significant role in determining the compression ratio that can be achieved by the encoder. The encoder is designed to take an optional argument specifying the codebook size, and the block dimensions \((m \text{ and } n)\). The vector dimension is \(mn\).

For an \(M \times N\) image with \(p\) bits per pixel, subdivided into fixed size blocks of \(m \times n\) and quantized using a codebook size of \(K\), the compression ratio is

\[
\frac{MNp}{\frac{MN}{\log_2(K)} + p(K)mn}
\]

(4.2)

The value of \(p\) is 8 for the input format used. As can be easily seen, for a fixed size block, compression ratio decreases with increasing codebook size. Here additional bit-rate reduction due to entropy coding, and the bit-rate increase due to overhead information are ignored.

The compression ratio achieved for a fidelity level, or equivalently the fidelity for a given compression ratio gives and indication of the performance of the codec, and makes rough comparisons with other codecs possible. Figure 4.1 shows plot of the peak signal to noise ratio (PSNR) for various codebook sizes for the vector quantizer. For a given bit-rate the JPEG fidelity is also given.

For the 260 \(\times\) 260 frame, using block of size 2 \(\times\) 2 and a codebook size of 128, the compression is:

\[
\frac{(260)(260)(8)}{(260)(260)/(2 \times 2)(7) + (8)(128)(2 \times 2)} \approx 4.5 : 1
\]

This compression is achieved at 33 dB, and the reconstructed image is of very good quality. However, the compression is inadequate. On the other hand the compression ratio from the same frame, when using the same block size, but a codebook size of 8 gives 25.2 dB and a compression ratio of about 11:1 is achieved. The second term is the denominator of equation 4.2, \(p(K)mn\) is a fixed term in the video compression, and thus its effect is less pronounced the longer the sequence that is coded using the same codebook.

47
Figure 4.1: Comparison with JPEG
Figure 4.2: Rate-Distortion Curves (block size 4x4)
4.1.2 Interframe Encoding

Interframe coding involves encoding the difference vectors (blocks) that result from taking the difference between the pixel values of the current frame with respect to a reference frame. Interframe encoding using vector quantization extends the well-known Differential Pulse Coded Modulation (DPCM) of scalar values to the case where the input is vector valued.

Interframe coding is particularly well suited for video, since additional compression can be achieved by exploiting the temporal redundancy in the sequence. In general, if there are no scene changes or rapid motions, successive frames in a video sequence are similar. Thus, the investigation into possibilities of additional compression using interframe coding are natural in digital video applications.

Equations of the form (4.2) are not readily available for interframe compression since there are many more factors that govern how much compression is achieved. Among the questions that must be answered are:

- which frames to encode as difference frames with respect to a reference frame
- resolution at which to encode the differences which generally have a larger dynamic range
- codebook strategy
  - use the same codebook for all frames
  - use a separate codebook for all difference frames
  - use a new codebook for each difference frame

The encoder can encode in the interframe mode. Sequences are grouped together and a base reference frame is intraframe encoded. Then the rest of the frames in the group are differentially encoded with respect to a reference frame. The period in Figure 4.3 refers to the number of frames in a such group.

To achieve bit-rate reduction, the difference frame pixels are quantized using fewer number of bits than the reference frame pixels. Difference frames can also be encoded using fewer code-vectors than reference frames thus allowing more compression.

50
It was found that the vector inputs to the vector quantizer needed their own new codebooks. The use of the codebook generated for the base frame, to encode the difference vectors was found to give very poor results at the decoder. The main reason for this is that the difference vectors contain components which may be negative while the reference (intraframe coded) frames do not. Furthermore reuse of even the difference codebooks was found to give poor results. The difference vectors are appear to be well de-correlated.

![Interframe Coding](image)

**Figure 4.3: Compression Using Differential Encoding**

Although, advantages can be anticipated in the applying differential encoding of video sequence. The resulting decoded video, is expected to suffer some degradation in quality. Here, the trade-off is between increased compression and fidelity. Compression is increased at the cost of some quality degradation.

The differential encoding results are given for the football sequence in Figure 4.3.
It shows that, as expected, difference encoding results in additional compression. The period indicates the frequency reference frames, thus a period of $p$ means that for every $p$ frames encoded one is a reference frame and the rest $p - 1$ are difference frames. The decrease in performance in going from $p = 1$ to $p = 2$ in Figure 4.3 is due to the overhead caused by the codebooks for difference frames. However, for larger number of difference frames ($p > 2$) it can be seen that a performance improvement can be achieved.

### 4.1.3 Initial Codebook Generation

The method used to generate the initial codebook and the distortion measure used to encode the vectors appear to have no effect on the compression ratio of Equation 4.2 above. However, since the compression ratios have to be compared for a fixed quality of the decoded image, these factors will have important implications.

Recall that the LBG algorithm used to design the vector quantizers is not guaranteed to give a globally optimum solution [16], but rather a locally optimum codebook. Thus, an investigation of which training methods result in better initial (and therefore final) codebooks is carried out. These codebook generation methods are compared. The methods are:

- **Binary Splitting**: this is described in Chapter 3, and is due to Linde, Buzo, and Gray [16].
- The first few input vectors can be used as training vectors.
- Uniform gray ramp, which is same value for each component in a vector and increases linearly can be used as an initial codebook (gray-scale sequences).

For a gray-level image tested, the three initialization methods, binary splitting, first-few vectors, and gray-ramp methods, the distortions calculated showed that the same performance results. This is due to the fact that the resulting codebooks of the LBG iterations are the same, illustrating that indeed the algorithm converges to a local optimum.
Figure 4.4: initial codebook generation and convergence
4.1.4 Threshold Effect

The choice of the threshold value \( c \) as used in the LBG algorithm is another parameter that can be varied. Intuitively, it is obvious that this figure plays a role in the rate of convergence for the LBG algorithm. The other factors that play a part in the rate of convergence are the codebook size, the vector dimension and number of training vectors.

As it has been shown that the LBG algorithm is guaranteed to give a codebook that is no worse (for the MSE distortion measure) than the codebook generated by any previous iteration \([16]\); the threshold may be significant in determining quality. Thus given a codebook, a smaller value of threshold (which usually implies larger number of iterations) is expected to produce a better codebook (in MSE sense).

The trade-off here is is between encoder complexity (more operations and thus longer time taken for convergence) and decoder quality. For highly asymmetric applications, one would tolerate increased encoder complexity for better quality at the decoder end.

The simulations shown in Figure 4.5 were carried out using the same training sequence and initial codebook and a block size of \( 4 \times 4 \). It can be seen that the choice of threshold is indeed an important factor in the quality of the decoded picture and thus care should be taken not to choose too large a value. As expected smaller codebooks require fewer iterations to converge but suffer from larger distortion; this is consistent with the rate-distortion theory.

For a vector dimension of \( N \), the computation of the squared Euclidean distance involves \( N \) subtractions, \( N \) multiplications, and \( N - 1 \) additions per vector.

4.2 Decoder Speed

The decoder speed is the parameter of interest in quantifying the complexity of the decoder of the codec. The decision has been made to use a software decoder, at the architectural level. The potential for fast decode-time was one of the most important consideration that was key to the decision to investigate the VQ as video coding.
Figure 4.5: Effect of threshold $\epsilon$
technique. As noted earlier, VQ in its simple form, involves only a table-lookup operation, which makes it an ideal candidate where fast decode speed is considered important.

To characterize the decode speed we need to measure the decode time for a given size of frame. Interframe require that additional processing be done on a decoded frame with respect to the reference frame before display. A plot of the decode speeds is shown in Figure 4.6.

![Decoder Speed Plot]

**Figure 4.6: Decoder Speed**
4.2.1 Effect of Aspect Ratio of Blocks

The method used to write a collection of pixel values (which are index values to a colormap for limited frame buffer displays) is similar to the raster scanning technique used in image scanning; that is the pixels are written horizontally. Thus the arbitrariness of aspect ratio of rectangular blocks comes into question. Intuitively, it seems that raster output is somewhat faster when displaying a sequence of pixels along a given row, than along a given column of a given image.

This potential, while it may seem unlikely to be very useful in image compression applications, is worth investigating in video compression applications where any gain in speed is significant. An experiment conducted to determine the gain if any, that the shapes of the blocks had on the decoder’s speed shows that if the rectangular shapes are chosen so that the width is greater than the height, decoder speed can be improved for the same compression ratio. The results are as shown in Figure 4.7.

It should be noted that larger block sizes result in a faster decoding performance.

4.2.2 Comparison With MPEG

Comparison is also carried out with the highly optimized MPEG decoder from the University of California at Berkeley. Results shown in the table below illustrate that indeed faster display times can be achieved with a vector quantization based approach. Comparing the compression achieved using the 2x2 blocks where the compressed file sizes are comparable (385598 bytes for MPEG and 396259 bytes using VQ) and the quality is not degraded, the gain in decoder speed is not significant. Using 4x4 blocks (99643 bytes) both speed and compression gains over MPEG are made, but this is achieved at the cost of quality (26 dB for the VQ based method).

The table below illustrates the performance of the VQ based method with an MPEG decoder. The experiments were carried out on a Sun SPARC-5 workstation with a clock speed of 70 MHz and 28MB of available memory.
<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Frame Rate</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VQ (2x2)</td>
<td>VQ (4x4)</td>
<td>MPEG</td>
</tr>
<tr>
<td>352x240</td>
<td>7.66 ± 0.06</td>
<td>17.7 ± 0.3</td>
<td>4.58 ± 0.72</td>
</tr>
<tr>
<td>256x176</td>
<td>14.8 ± 0.45</td>
<td>36.2 ± 0.83</td>
<td>9.32 ± 0.68</td>
</tr>
</tbody>
</table>

![Decoding Speed (football)](image1)

![Decoding Speed: (cheer)](image2)

![Relative Decoding Speed (4x2 : 2x4)](image3)

![Relative Decoding Speed (4x2 : 2x4)](image4)

Figure 4.7: Effect of block shapes on decoder speed
Chapter 5

Colormap Design for Faster Display

Color frames consist of three planes. When color frames are encoded in the RGB domain the red, blue and green color planes are separately processed. The processing at the encoder can also take place in other domains where the planes are de-correlated. This often involves a coordinate transformation of the RGB color space into a new color space. The YIQ and HSI\(^1\) are examples.

Most Color display monitors display frames by modulating the intensity of the primary colors (Red, Green and Blue) at each pixel. With the standard color images, each primary color component takes up 8 bits requiring 24 bits to specify a full-color. The cost of high-speed memory required to support \(2^{24} = 16,777,216\) colors makes this approach impractical. Thus a color lookup table is used in most display hardware, to select a subset of the possible colors and pixel values are used as indices to them. Thus, at the decoder, each color-pixel value (three values corresponding to each color) must be decoded to approximation of the original color frame in the RGB domain for display.

For displays with only 8 bits per pixel frame-buffers, once the colors are decoded in the RGB domain, the pixels need to be mapped to the appropriate color in a

\(^1\)Hue, Saturation and Intensity
color look-up table (CLUT) also known as a colormap. The efficient use of colormaps involves two major goals:

1. the colormap should consist of colors such that the distortion that results from the quantization is acceptable

2. the mapping from pixel-colors to colormap indices should be done quickly

The first criterion of finding an adequate representation of the colors in the sequence for storage is a color quantization problem. The second requirement, that of mapping the pixels quickly to the appropriate color in the colormap, arises naturally in video decoding applications especially when low decoder complexity is desired. The following sections will deal with the generation and structures of colormaps and pixel mapping techniques for efficient frame-buffer display.

5.1 Gray-scale Video

For gray-scale sequences, the solution to the problem of mapping the indices to the appropriate color is relatively simple. Suppose all the colors in the input are quantized to 8 bits per color pixel as is often the case. The set of all colors in the RGB space consist of $256^3$ colors. Gray-scale values lie along a diagonal line in the the color cube connecting the color black at $(0,0,0)$ and the color white at $(255,255,255)$ as shown in figure 5.1.

The transformation matrix, used to change RGB values into YIQ values is given below.

\[
\begin{bmatrix}
    Y \\
    I \\
    Q
\end{bmatrix} =
\begin{bmatrix}
    0.299 & 0.587 & 0.114 \\
    0.596 & -0.275 & -0.321 \\
    0.212 & 0.523 & 0.311
\end{bmatrix}
\begin{bmatrix}
    R \\
    G \\
    B
\end{bmatrix}
\]

Thus the luminance value is given by

\[
Y = 0.299R + 0.587G + 0.114B
\] (5.1)
Thus if all values in the RGB domain are equal, say $R = G = B = \psi$ then

$$Y = 0.299R + 0.587G + 0.114B$$
$$= 0.299\psi + 0.587\psi + 0.114\psi$$
$$= \psi(0.299 + 0.587 + 0.114)$$
$$= \psi$$

Figure 5.1: Gray-scale values in the RGB color-space

Let the colormap be constructed such that the $i^{th}$ color in the colormap is $<i,i,i>$ in the RGB domain. Once the colormap is set in such a manner, any pixel can be mapped to its appropriate index in the colormap by the identity mapping $I$

$$I : \mathbb{R} \rightarrow \mathbb{R}$$

that maps each input gray-scale value $x$ to an index in the colormap $I(x)$ such that

$$I(x) = x$$

Here the pixel value associated with the typical colormap size of 256 colors is adequate to display any gray level that is quantized with a resolution of 8 bits or less.

Alternatively, as the set of all possible inputs in this case is 256, one can also compute an inverse quantization table of size no more that 256 and use it for pixel to
colormap-index mapping. Thus the restriction on the colormap structure discussed above need not be used if colormap flashing is a problem. Flashing will be discussed later.

5.2 Color Quantization

Colormaps were implemented as a result of the need to display color in the RGB space in an inexpensive manner. Indeed it can be observed from the discussion above that gray-scale video need not manipulate colormaps in any sophisticated fashion.

For color image and sequence display, the problem of color quantization has been investigated in recent years [27, 28]. Previously the problem was treated as a one dimensional quantization problem in a color-spaces such as the YIQ space. However, recently the problem has been formulated as a multi-dimensional quantization problem in the RGB color-space and some algorithms have been suggested. Among the algorithms suggested for the quantization of colors to construct good colormaps are the popularity algorithm, where a histogram of frequencies is constructed and colors with the highest frequencies are used to construct the colormap, and the median cut algorithm [27] where the emphasis is placed on choosing colors which are used by approximately equal number of pixels.

The efficient mapping of input colors to indices in the colormap has two competing criteria that it should meet. First, the color chosen from the colormap should be a reasonable approximation to the input. Ideally the chosen color would be best approximation to the input in the colormap. Second, the mapping should be done as quickly as possible.

As is often the case in engineering problems, these two criteria for the efficient mapping of pixels to indices in the colormap are in conflict with one another. To achieve the best representation of a given color from among the candidates in a colormap, an exhaustive search of the colormap would be required. The distortions between the input color and each color in the colormap have to be evaluated to de-

\footnote{except in cases where the structure of the colormap makes this unnecessary}
termine the best color in the colormap for a given input. This approach is generally compute intensive and thus at odds with the requirement of quick mapping. On the other hand, the mapping operation can be done quickly if the number of colors in the colormap that are to be searched is smaller. However, this would generally degrade the quality of the decoded frames as the reduced effective colormap size, would mean that optimum mapping is not guaranteed.

5.2.1 Exhaustive Search

An exhaustive search involves the evaluation of the distortion between a given input and all colors in the colormap to determine the one with the smallest distortion. The index of this color can then be written to the frame buffer. The disadvantage of this approach is that the method is very slow. A lot of computations are carried out on colors which are too distant and should, if possible, be eliminated from consideration. The advantage, on the other hand, is that the colormap can be made to contain colors that give good performance in quality.

The construction of an inverse quantization table, similar to the one suggested for gray-scale video, is prohibitive in terms of storage and the complexity of generating the table. This is due to the fact that the number of possible inputs is now $256^3$ as opposed to merely 256 for gray-scale sequences.

An exhaustive search approach with the MSE distortion measure was used to decode a video sequence and, as expected, it was found to be too computationally complex for the decoder to decode in real-time. Since a full search of the colormap is not a solution to the problem of finding an efficient mapping, different alternatives were investigated to find an acceptable solution. In particular, structured colormaps can be used so that the search for a suitable color in the colormap is reduced a small subset of the colors available.
5.2.2 Lattice-type Colormaps

The idea of introducing some structure to the colormap so that the color pixel to colormap index mapping can be done quickly, was discussed with respect to gray-scale video decoding. This idea can also be useful in displaying colors.

Lattices exhibit very regular structures and are investigated as possible solutions to the problem of pixel to colormap index mapping.

A lattice $\Lambda$ is a discrete set of $N$ tuples in Euclidean $N$-space that forms a group under addition. A sub-lattice $\Lambda'$ of a lattice $\Lambda$ is a subset of the lattice $\Lambda$ that is itself a lattice.

Two $N$-tuples $x_1$ and $x_2$ are said to be equivalent modulo $\Lambda'$ if $x_1 - x_2 \in \Lambda'$. Such an equivalent relation forms a partition of $\Lambda$ into a family of disjoint equivalence classes. The different equivalence classes modulo $\Lambda'$ are called the cosets of $\Lambda'$ in $\Lambda$ and are denoted $\Lambda'/\Lambda$.

Since the cosets of $\Lambda'$ in $\Lambda$ form a partition, the lattice $\Lambda$ can be specified in terms of its coset decomposition.

$$\Lambda = \Lambda' + [\Lambda/\Lambda'],$$

where $[\Lambda/\Lambda']$ is a set of arbitrary elements chosen from each of the coset of $\Lambda'$ in $\Lambda$ called cosets representatives.

A lattice $\Lambda$ can be specified by its generator matrix $M$. The matrix consists of vectors $m_i = (m_{i1}, m_{i2}, \ldots, m_{iN})$

$$M = \begin{bmatrix}
m_{11} & m_{12} & \cdots & m_{1N} \\
m_{21} & m_{22} & \cdots & m_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
m_{N1} & m_{N2} & \cdots & m_{NN}
\end{bmatrix}$$

Then any vector in the lattice can be specified by $\xi M$ where $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$ is an arbitrary vector with integer components.
Lattices, by definition, consist of an infinite number of points and their practical applications require that only a subset of the points that lie within a given region be considered. An example is a uniform $6 \times 6 \times 6$ colormap formed from the intersection of a set of all points in a cubic volume in the $RGB$ space $CC$, with the sub-lattice $42\mathbb{Z}^3$

$$42\mathbb{Z}^3 \bigcap CC$$

where

$$CC = CC(r, g, b) = \begin{cases} (r, g, b) \text{ such that } & 0 \leq r \leq 255, \\ 0 \leq g \leq 255, \\ 0 \leq b \leq 255 \end{cases}$$

and $r \in \mathbb{Z}, g \in \mathbb{Z}$ and $b \in \mathbb{Z}$

The sub-lattice used above could have been replaced with any of the cosets of $42\mathbb{Z}^3$ in $\mathbb{Z}^3$.

Given the uniform colormap above, the mapping problem may be interpreted as deciding to which of the cubic cells (which partition $CC$) formed by the lattice a given color belongs. Imposing further structure on the colormaps allows the colormap index to be generated easily.

As an example let the colors in the uniform colormap be lexicographically ordered, i.e., the colormap be as shown in figure 5.2. Then, an integer division operation carried out on the individual color components and the results can be combined to yield the correct index in the colormap.

The process is here is the very familiar vector quantization. Ironically once the colors values are decoded using the inverse vector quantization process, VQ encoding is used to map them to the desired index in the colormap.

The main advantage behind the use of the lattice-codebook is the fact that since the points (colors) have a regular arrangement, the computational effort required to find the closest color in the colormap for any given valid color can be greatly reduced. This happens because the multidimensional quantization problem, which is
generally difficult, is reduced to a few one dimensional quantization operations which are relatively easy to carry out.

Limitations of this method are apparent. The non-uniform distribution of colors is not exploited. Moreover, the uniform colormap uses $N$ levels across each axis (r,g and b). The total number in our codebook (colormap) is fixed to the typical size of $C_{size}$. Thus to obtain an $N \times N \times N$ colormap with the most colors we use

$$N = C_{size}^{\frac{1}{3}}$$

where $C_{size}$ is the number of colors in the colormap. Thus, for $C_{size} = 256$ we get $N = 6$. This means that for a 256 color codebook, we can use a 216 of them and the rest $256 - 216 = 40$ are unused.

### 5.2.3 Skewed Lattice Colormaps

To utilize more of these unused colormap entries, it is enough to realize that by adding an extra level in one of the axes, additional $N^2$ colors can be added to the set of output colors. This does not destroy the lattice structure of the codebook; it merely uses a skewed underlying lattice. However, we can have $7 \times 6 \times 6 = 252$ colors. This arrangement can be thought of as weighting the relative importance of
the individual primary components of each color as one would be better preserved than the other two.

The question still remains as to which one of the primaries should get the extra level assigned to it.

The use of an $8 \times 8 \times 4$ colormap that favours the red and blue components at the expense of the blue which is quantized to only four levels has been reported for image-independent color quantization in [28].

An indication is given by the color conversion matrix that transforms the $RGB$ color space into the $YIQ$ color space. The luminance component is the most important signal in the $YIQ$ color space as it nearly all the energy in the signal.

From equation 5.1 it can easily be observed that the $G$ component contributes the most to the luminance signal.

The sub-lattice that will be the basis of this colormap is then specified by the generator matrix

$$M_{green} = \begin{bmatrix} \frac{256}{6} & 0 & 0 \\ 0 & \frac{256}{7} & 0 \\ 0 & 0 & \frac{256}{6} \end{bmatrix}$$

The red or blue components can be favoured by changing the coefficients of this generator matrix accordingly.

### 5.2.4 Very Limited Search Colormaps

Another approach to tackle the mapping problem is to use a limited search of candidate colors once the color value has been assigned to a given cell.

Limited search techniques where a color cube is subdivided into smaller cubes each of them containing a sorted list of representative colors has been reported to give good results for color quantization of images [27]. A limited form of search based on the use of larger cubes is attempted. These searches allow the use of colormaps with larger set of points in the cube. The use of 5 levels to quantize the each of the axes in color-cube results in $5^3 = 125$ cubes. Using two representative per each cube will result in 250 colors; this better utilizes the available colormap entries.
An even better utilization of the allowed number of colors can be envisaged with the use of 4 levels along each axis and the use of 4 points in each cube. This results in a total of $4 \times 4^3 = 256$. All entries will be used up in this approach which may not necessarily be desired since it may cause flashing problems.

In these schemes, once a given $(r, g, b)$ color has been decoded into one of the cubic cells within the color cube, then comparison has to be made to find out which of the points that are in the cell is the best representation. Thus, a limited search is carried out which is much smaller than an exhaustive search. For the 125 cubic cells resulting from a 5 level quantization the search is limited to two. For the 4 level quantization, the search is limited to 4.

Figure 5.2.4 shows the positions of 5 different color quantization schemes. The time axis serves as a measure of complexity while the vertical axis shows the mean squared error achieved. Good schemes would lie in the bottom-left corner of the map.

The $4 \times 4 \times 4$ and $5 \times 5 \times 5$ codebooks are limited search colormaps with 4 and 2 points per cell respectively. Figure 5.2.4 shows that they are to be the most complex to implement as they involve comparison operations.

The other colormaps $6 \times 6 \times 6$ and $8 \times 8 \times 4$ and $6 \times 7 \times 6$ are skewed-lattice colormaps where the color components are quantized with different resolutions. These have the lower complexity than the limited-search colormaps. The fidelity performance depends on the choice of the colors in each cube for the limited-search colormaps and is sequence dependent. The encoder can determine the codebook(s) to be used for a given sequence and transmit this information along with the bitstream.

5.3 Problems Colormaps

5.3.1 Static Colormaps

The above discussion assumes that the colormap is static, i.e., it will be used for the entire video sequence. However, in video coding, a static colormap, even when it may be globally optimum for the entire sequence, would almost certainly not be locally
Figure 5.3: Colormap Performance
If the lattice-type codebooks are to be used, the amount of localization that can be achieved is limited. However, if colormaps are transmitted along a greater degree of localization is possible. A colormap that is known to perform better locally can be periodically sent to achieve better quantization colors. This approach has some drawbacks.

- There will be a slight penalty in compression ratio
- Slower decoder speed

### 5.3.2 Colormap Flashing

In video coding experiments, the emphasis is placed on the investigation of parameters of interest which often means bypassing overheads that can be associated with an integrated environment. One disadvantage of this is that user interfaces and portability are more difficult to achieve.

A frequent problem in color intensive applications that use the same display monitor is that when the colors in the colormap are insufficient for all the applications, only the active window's colormap is used. This causes the remaining applications to appear different as colors are changed dramatically when new colormaps are installed. This creates a very noticeable effect which is referred to as *colormap flashing*.

The utilization of all or most of the entries in the colormap by a video codec will cause such problems if it is used concurrently with applications requiring a different set of colors. One way to avoid this is to leave a few colormap entrees for use along with the rest of applications and use a portion of the colormaps for private colors.
Chapter 6

Conclusion

6.1 Summary of Research

Some of the popular image and video compression techniques were reviewed. The advantages of digital transmission and storage of video and the difficulties associated with it have been discussed.

The work done, showed the feasibility of a software based codec for asymmetric applications of video on desktop computers. The crucial requirement that the computational complexity of a software only decoder be low was the primary motivation for the investigation of vector quantization as a compression technique for digital video. Vector quantization was shown to be a powerful technique capable of high compression as well as having a very low complexity decoder. Interframe encoding techniques, which are suited for video compression applications, have been applied and additional compression was achieved. This was accomplished by exploiting the temporal redundancy inherent in video sequences.

Techniques for displaying video on hardware with limited frame-buffers that make use of colormaps have been suggested. Simple color quantization based on lattices have been suggested and variations of these were implemented.

Video coding has entered the stage where its widespread use is imminent. As progress in microprocessor technologies continues and prices fall, and the capacity of storage devices increases, software based codecs will have the ability to deliver good
quality video. The increasing interest in multi-media applications to display software video using computers, was a motivating factor for this work.

### 6.2 Suggestion for Further Research

Several areas of improvement and further research can be mentioned.

- The first area involves scalable video compression. A compression scheme that organizes data hierarchically for decoder dependent decompression can be incorporated into the encoding process. While work in this area has been done [29], focus has not been placed on software based decoders.

- Encoding of color-video together with a priori knowledge of colormaps to be used can be investigated. In particular, pre-quantization of the source input into values that can be better recovered at the decoder may yield good results.

- Incorporation of the HVS model into the compression scheme may help improve the quality of a decoded sequence. Many similarities exist between some HVS-based methods such as VPIC and VQ.

- Address-predictive vector quantization (APVQ), where the encoder exploits the fact that only the address information (index) is available to the decoder and uses correlations between indices to get additional compression [30], can be investigated.
References


