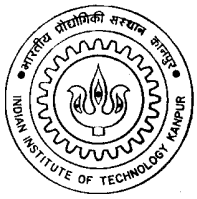
Implementation of an MPEG-2 Video Decoder

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Subhendu Bhadra



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

May, 1999

Certificate

This is to certify that the work contained in the thesis entitled "*Implementation of an MPEG-2 Video Decoder*", by *Subhendu Bhadra*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

May, 1999

(Dr. Rajat Moona)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Dedicated to my parents

Acknowledgements

I express my sincere thanks and heart-felt gratitude to Dr. Rajat Moona for his guidance and valuable suggestions throughout this work.

I would like to thank Neomagic Corporation, California, US , especially its Vice-president, Engineering Dr. Clement Leung, for extending financial and technological support to my thesis and choosing me "Neomagic Fellow".

I am indebted to Dr. Leonardo Chaiariglione, convener, MPEG and Dr. Thomas Sikora, chairman, MPEG video group, for providing me various information I needed at different times of my thesis through emails. I am indebted to all my class-mates, seniors and juniors for making my stay at IITK memorable. I thank all my friends in Hall-V for giving me moral encouragement and mental satisfaction throughout my stay at IITK. I would also like to remember the IITK-Kanpur sports team members who have accompanied me during the Inter-IIT sports meets in IIT-Kharagpur (Dec'97) and IIT-Bombay (Dec'98) for being part of my most favorite entertainment.

This auspicious moment reminds me of my parents, my brothers and all my near and dear ones for their love and emotional support throughout my life.

Abstract

Moving Picture Experts Group (MPEG) has developed two generic audio-video standards till date, namely MPEG-1 and MPEG-2. While MPEG-1 was developed for storage and retrieval of moving pictures and audio on storage media, MPEG-2 targeted at higher quality and bandwidth applications like digital television. With the number of applications of MPEG-2 standard increasing everyday, there arises the necessity of a stable and efficient decoder for MPEG-2 compatible bit-streams that can overcome the bottleneck of speed in real-time applications.

In the current work, we have developed a real-time video decoder for MPEG-2 bit-streams with an emphasis on *stability*, *efficiency* and *picture quality*. We have experimented with several approaches to compute the inverse discrete cosine transform, and chosen the most efficient one which takes $O(N^2 \log_2 N)$ time for a block size of $N \times N$. As the decoder should be able to display frames at the coded frame-rate, it may intermittently discard some of the frames when otherwise the required frame-rate can not be met.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The techniques of capturing images by cameras date long back. Traditional cinematography use techniques by which individual pictures called frames are displayed at a rate that human being cannot distinguish separately due to persistence of vision, thus giving an illusion of continuous motion picture. Therefore, the motion pictures are basically a collection of frames displayed typically at a rate exceeding about 20 frames per second. With the advent of digital computers, it became inevitable to evolve techniques to store images and more importantly video in a non-volatile media.

A digital image is essentially an array of pixel values represented by a finite number of bits. The term digital image processing generally refers to processing of a digital image by a digital computer. Digital Signal Processing (DSP) techniques are the most commonly used techniques for digital image processing. A digital video is a collection of digital images stored in an efficient format so that it can be processed in real-time and requires small space for storage. With increasing interest in multimedia applications, audio is also added alongwith the video to get the flavor of real movies in digital computers.

Several efforts have been undertaken in order to standardize the digital audio-and-video coding technology. The Moving Picture Experts Group (MPEG) [11] has developed two important standards, namely MPEG-1 and MPEG-2, which are the

most widely accepted ones in this direction. MPEG-2 is a logical extension of MPEG-1 standard, incorporating further refinements and more functionalities. The standard has certain inherent capabilities that have made it an automatic choice of the industry. In this thesis we have developed a real-time video decoder for MPEG-2 coded motion pictures.

1.2 Digital Compression

Compression is a technique to represent digital information in a compact form in order to save precious storage space. For images and digital audio applications, the compression also yields the minimization of the bit-rate. The compression in such applications can even be lossy under the constraints of signal quality, implementation complexity and compression/decompression speed. A considerable amount of statistical redundancy exists in such data due to spatial and temporal correlation among the different samples and therefore a large compression can be achieved.

1.2.1 Importance of compression

Usually an enormous amount of data is associated with audio-visual information, the storage for which is not always affordable. Although the capacities of many digital storage media are substantial, their access speeds usually vary inversely in relation to their capacity. Therefore storage and transmission of image data require huge capacity and bandwidth, which is often very expensive. The image compression techniques have been of great significance by which the number of bits required to store or transmit images can be reduced by orders of magnitude without any perceivable loss of information.

A reasonable quality motion picture has spatial resolution of approximately 512× 512 pixels per frame. At 8 bits per color component (i.e., 24 bits per pixel value) and 30 frames per second, this translates into a rate of nearly 23 MBytes/sec of uncompressed data. Therefore, an half hour (1800 seconds) movie will require a storage space of about 40 GB. Further enhancements in the picture quality result in even higher data rates.

1.2.2 Lossless versus Lossy compression

Compression methods may be either lossy or lossless. In lossless compression, the reconstructed data is identical to the original data. But a majority of applications in image processing do not require the reconstructed data to be identical to the original data as long as the picture quality is within the acceptable limit. MPEG-2 video standard recommends the evaluation of the Discrete Cosine Transform (DCT) for the compression [11]. Though inherently, the DCT itself is a lossless transform, the reconstructed video is not necessarily identical to the original data due to the quantization of the resultant values.

1.3 Discrete Cosine Transform

Discrete Cosine Transform (DCT) is a fundamental computation in many digital signal processing applications, such as image data compression. For image data it results in a compression in the range of about 10 to 100.

1.3.1 Definition of DCT

The $N \times N$ cosine transform matrix $C = c(k, n)$, called *discrete cosine transform*, is defined as

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0,\ 0 \le n \le N-1 \\ \frac{2}{\sqrt{N}} cos\frac{(2n+1)k}{2N}\pi, & 1 \le k \le N-1,\ 0 \le n \le N-1 \end{cases}$$

For a given 2-d data sequence $\{x_{ij} : 0 \le i, j \le N-1\}$, the 2-d DCT sequence $\{Y_{mn} : 0 \le m, n \le N-1\}$ is given by the following [1].

$$Y_{mn} = \frac{4}{N^2} u(m)u(n) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij}\, cos\frac{(2i+1)m}{2N}\pi\, cos\frac{(2j+1)n}{2N}\pi \tag{1.1}$$

and the inverse DCT (IDCT) is given by

$$x_{ij} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m)u(n)Y_{mn}\, cos\frac{(2i+1)m}{2N}\pi\, cos\frac{(2j+1)n}{2N}\pi \tag{1.2}$$

where

$$u(m) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if m} = 0 \\ 0, & \text{otherwise} \end{cases}$$

In order to use DCT, the image is subdivided into 8×8 block of samples. Each of these 8×8 blocks of the original image is then mapped to the frequency domain, i.e., it is represented as a compositions of DCT *basis functions* with 64 appropriately chosen coefficients, representing different horizontal and vertical intensities. The figure 1.1 illustrates the corresponding spatial frequency[1] patterns. Each of these spatial frequency patterns has a corresponding *coefficient*, the amplitude needed to represent the contribution of that spatial frequency in the block of data being analyzed.

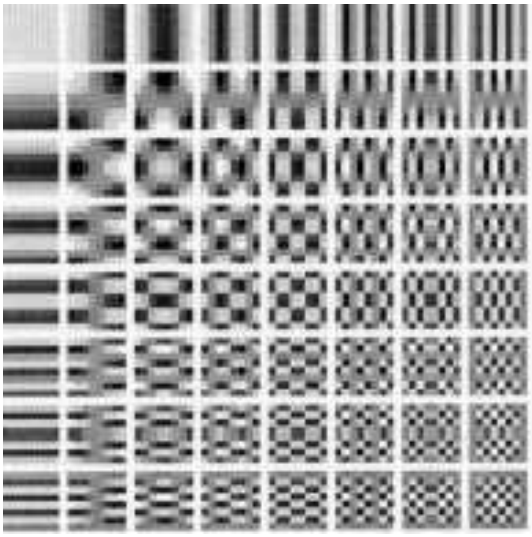


Figure 1.1: Basis function patterns of an 8X8 cosine transforms

Several fast implementations of DCT and IDCT - both software and hardware - have been proposed in the literature [2, 4, 6, 18]. We have used the row-column approach proposed by [1] to transform the 2-d IDCT computation into 16 1-d IDCTs and then used the algorithm proposed by B.G.Lee [8] to compute 1-d IDCT in $O(N\log_2 N)$ time. The overall complexity of the 2-d IDCT computation is then $O(N^2\log_2 N)$.

[1] luminance changes w.r.t. spatial coordinates

1.3.2 Merits of DCT

DCT is so useful because of its following properties :

- The cosine transform requires only real number arithmetic and sans handling of complex numbers.
- The cosine transform is an orthogonal transform, i.e.,

$$C^{-1} = C^T$$

- The computation of cosine transform is fast. The cosine transform of a vector of N elements can be calculated using $O(N\log_2 N)$ operations.
- The cosine transform has excellent energy compaction for highly correlated data, i.e., if there is considerable amount of correlation between different data-points, the most of the information in the transformed data will be stored in small number of data-points and so we can represent the original data with fewer data-points.
- The cosine transform is very close to the statistically optimal Kerhunen-Loève transform (KLT) for highly correlated images [1]. This property of the cosine transform together with the fact that it is a fast transform based on real operations has made it a useful substitute for the KL transform for highly correlated image data.

1.4 MPEG Standards for Motion Pictures

MPEG is a working group of ISO[2]/IEC[3] consisting individuals operating in research, academia and industry. The group is responsible for the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio and their combination. Established in January 1988, MPEG has already developed two evolutionary standards MPEG-1 (Nov. 92) (formally, "Information technology - Coding of moving pictures and associated audio for digital

[2]International Standard Organization
[3]International Electrotechnical Commission

storage media at up to about 1,5 Mbits/s"), and MPEG-2 (Nov. 94) (formally, "Information technology - Generic coding of moving pictures and associated audio information"). Two more standards, namely MPEG-4, the standard for multimedia applications and MPEG-7, the content representation standard for information search are currently being developed.

Worldwide, the MPEG-1 and MPEG-2 audio/video coding standards have attracted much attention over the past few years, with an increasing number of VLSI and software implementations of these standards becoming commercially available. While MPEG-1 has made a remarkable impact in audio/visual CD-ROM applications, with various implementations in both software and hardware, MPEG-2 standard has changed the concepts of digital television with applications like DVD (Digital Versatile Disk), HDTV (High-Definition Television) etc.

1.4.1 MPEG-2 Standard

MPEG-2 standard was developed with many video-coding algorithms integrated into a single syntax to meet diverse applications requirements. As a rule, every MPEG-2 decoder should be able to decode a valid MPEG-1 bit-stream. New coding features were added in MPEG-2 to achieve sufficient functionality and quality. However, implementation of the whole syntax is not required for every application. Therefore, MPEG-2 has defined subsets of the standard to address specific classes of applications with similar functional requirements. This is accomplished by introducing the concept of *Profiles*. A profile is thus a defined subset of the entire syntax of MPEG-2. MPEG-2 defines five different profiles, namely Simple profile (SP), Main profile (MP), SNR-scalable profile (SNR), spatially-scalable profile (Spt) and High profile (HP) with the relation SP $\subset$ MP $\subset$ SNR $\subset$ Spt $\subset$ HP. In general, each profile defines a new set of algorithms added as a superset of the profile below.
Even with profiles to define specific subsets of the entire syntax and functionality, the parameter ranges are quite large to achieve compliance over the whole range. Therefore, *Levels* are introduced to put constraints on some of the parameters. A Level specifies the ranges of parameters like frame-size, frame-rate, and bit-rate that are to be supported by a particular application to conform to a specific profile. Four levels are defined in MPEG-2 : low (LL), main (ML), high-1440 (H-14), and high (HL). It is expected that most MPEG-2 implementations will at least conform to the

Main Profile at Main Level (MP@ML).

MPEG-2 Systems

MPEG-2 standard evolved from MPEG-1 standard. MPEG-1 was designed to work with digital storage media that have minimum errors. On the other hand, MPEG-2 is designed to work through the communication networks also and therefore it needed improved error resilience. MPEG-2 systems solved these diverse requirements by defining two different data stream types : the program stream (PS) , optimized for storage devices and personal computer implementations, resulting in long variable length packets [4] and the transport stream (TS), which uses short, fixed length packets (188 bytes) in order to facilitate transmission over noisy channels. Both data-stream types use the same packet structure called packetized elementary stream (PES) structure. Each PES contains exactly one video or audio elementary stream. A model for MPEG-2 systems is shown in the figure 1.2.

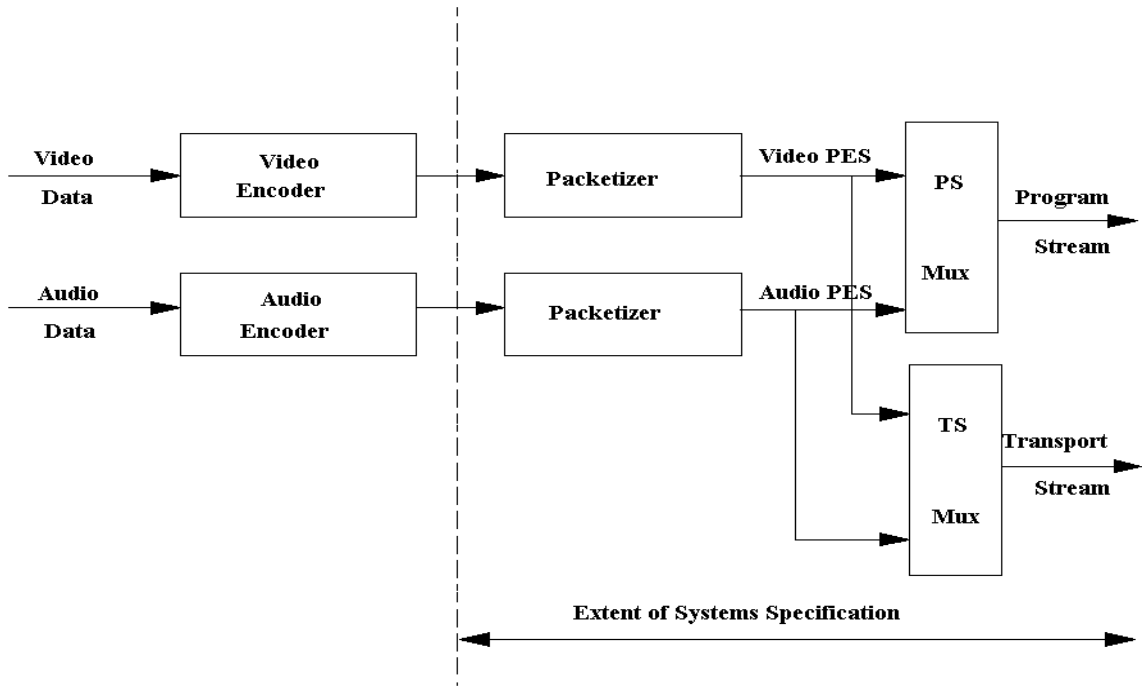


Figure 1.2: Model for MPEG-2 Systems encoding

[4] long packets can be supported because of a presumed low media error rate. Resultant low overheads allow processing of packet headers in software by a general purpose processor

The data streams are synchronized between the encoder and the decoder. For this, MPEG standards provide a mechanism of timestamping each PES with respect to a common reference point. The common reference point is first communicated between the encoder and the decoder.

MPEG-2 Video

MPEG-2 video standard was originally intended for coding interlaced video at television resolution in the bit-rate range of 4-9 Mbits/s. However, MPEG-2 video also supports higher resolution such as those needed for HDTV, at higher bit-rates. In MPEG-2, image frames are encoded into one of the four frame types I-frame (intra-coded), P-frame (predictive-coded), B-frame (bi-directionally predictive-coded) and D-frames (DC-coded). I-frames are are coded independently, i.e., without any reference to the other pictures. P- and B-frames are compressed by coding the differences between the current frame and the reference[5] I-frame or P-frame, thereby exploiting the temporal correlation between successive frames. P-frames calculate predictions from a previous I-frame or P-frame, whereas B-frames obtain predictions from the nearest and upcoming I-frame or P-frames. D-frames are allowed in the MPEG-1 compatibility mode only. They are coded using only the DC coefficients of the DCT block.

MPEG-2 Audio

The MPEG audio compression standard defines a family of algorithms appropriate for a wide range of audio material, like speech, music, and the range of special effects that might be expected on a movie soundtrack. MPEG-2 audio extends the two-channel stereo capability of MPEG-1 audio (ISO/IEC 11172-3) to five-channel surround sound, with the option of a sixth low-frequency enhancement (LFE) channel which can be used to produce loud sound effects.

[5] a preceding and upcoming anchor (I-frame and P frame) frame w.r.t. which the difference are calculated

1.4.2 Emergence of MPEG-4 for multimedia applications

Anticipating the rapid convergence of telecommunications, computer, and television, the MPEG group has officially initiated a new standardization effort in 1994 with an eye to standardize algorithms and tools for coding and to provide a flexible representation of audio-visual information to meet the challenges of future multimedia applications. This effort is formally known as MPEG-4.

- *MPEG-4 Video*

  1. MPEG-4 video standard is intended to support various formats, bit-rates and resolution.
  2. MPEG-4 video is expected to achieve compression ratio higher than the existing standards.
  3. MPEG-4 video is expected to provide content-based functionalities like random-access scalability of texture, images and video.

- *MPEG-4 Audio*

The new features that will be supported by MPEG-4 audio are features like synthesized speech, trick-mode functionalities like pause, resume, forward and backward.

1.5 Related work

Before the official release of MPEG-1 and MPEG-2, MPEG Software Simulation Group (MSSG) developed decoders for them and tested the standards thoroughly to check for any hidden flaws. After the actual release of the standard, many hardware and software vendors and research groups have implemented these standards. The Berkeley Plateau Multimedia Research [13] developed the first widely-distributed software decoder for MPEG-1 video in November 1992. Neil Gray of University of Wollongong, Australia, has implemented an MPEG-1 motion picture decoder in C++ [14]. Researchers in the Vision and Neural Networks Lab. of Wayne State University has developed MPEG Developing Classes (MDC) to help people implement

their own MPEG related softwares without actually going into minute details of MPEG [15].

1.6 Current Work

In the current work, we have implemented an MPEG-2 video decoder. The video decoder decodes video frames and then displays them at a speed reasonable enough for real-time video. However, if it cannot meet the frame-rate requirement for a particular bit-stream, the decoder discards some of the frames to adaptively catch up with the required speed. In this current work, the audio data is not decoded. Instead the audio information is read and sent to an MP3 [16] audio decoder in a form understandable by it. Thus audio and video are played by two different processes. However, if no audio player is available it discards the audio and displays the video only. In the decoder, the image decompression algorithms are developed in such a way that we get the best quality of video at a reasonable speed. The decoder can also play from a VideoCD [19] instead of a file on the hard disk. The decoder can also display the video frames at any size, including the Full-Screen size. Trick-mode functionalities like Rewind, Fast/Slow Reverse, Fast/Slow Forward and Pause/Resume are provided with the video player. A user-friendly graphical user-interface is also provided with the video decoder.

1.7 Organization of this report

The rest of this report is organized as follows. In chapter 2, we describe the design of the video decoder and discuss the related issues. In chapter 3, we present the implementation details of the decoder and provide the outlines of the algorithms. In chapter 4, we conclude this thesis and provide test setup, results and conclusions.

Chapter 2

Design of the Video Decoder

2.1 Overview

The design of decoder has been kept as simple as possible. We have followed the data flow as described in the MPEG-2 standard. Any valid MPEG-2 bit-stream contains at least one PES that consists of either a video or an audio stream. Since an MPEG-2 decoder should be able to decode an MPEG-1 bit-stream also, the video bit-stream structure shown in the figure 2.1 is valid for MPEG-2 standard also.

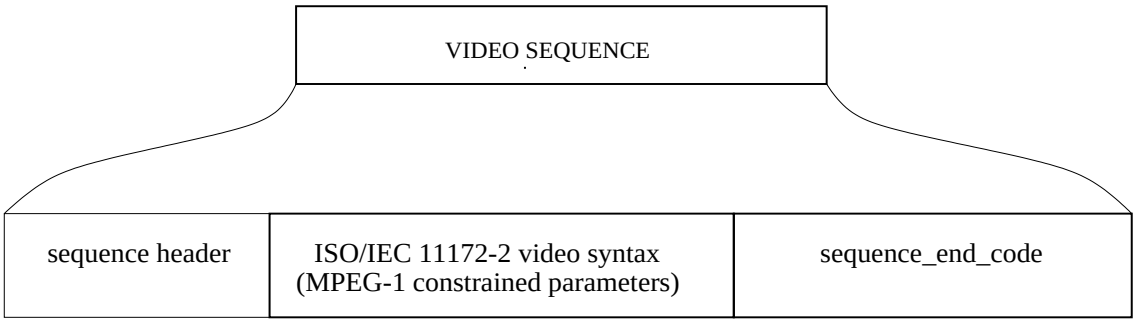


Figure 2.1: MPEG-1 video syntax

MPEG-2 syntax for a typical coded video sequence is defined in a hierarchical representation with six layers as shown in the figure 2.2.

Every valid MPEG-2 video bit-stream starts with a sequence header. A sequence header may optionally be followed by *sequence extension* For true MPEG-2 video bit-streams the sequence extension is necessary part of the sequence layer. For MPEG-1

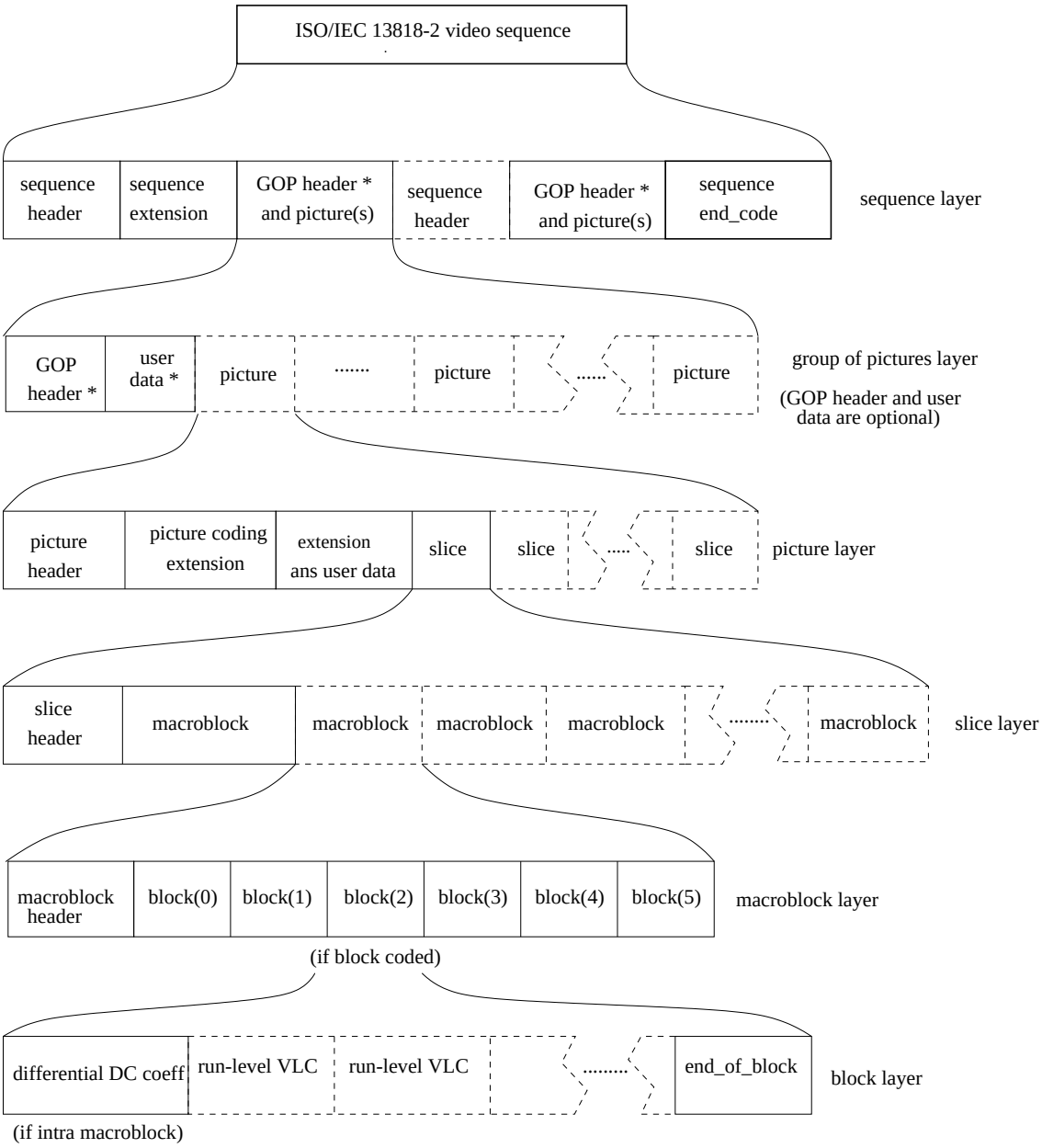


Figure 2.2: MPEG-2 video syntax

video bit-streams, it is never needed. Thus the decoder can identify the type of bit stream (MPEG-1 or MPEG-2) and decompress it accordingly. In essence, the possible routes in the MPEG-2 video bit-stream syntax can be one of the two shown in the figure 2.3.

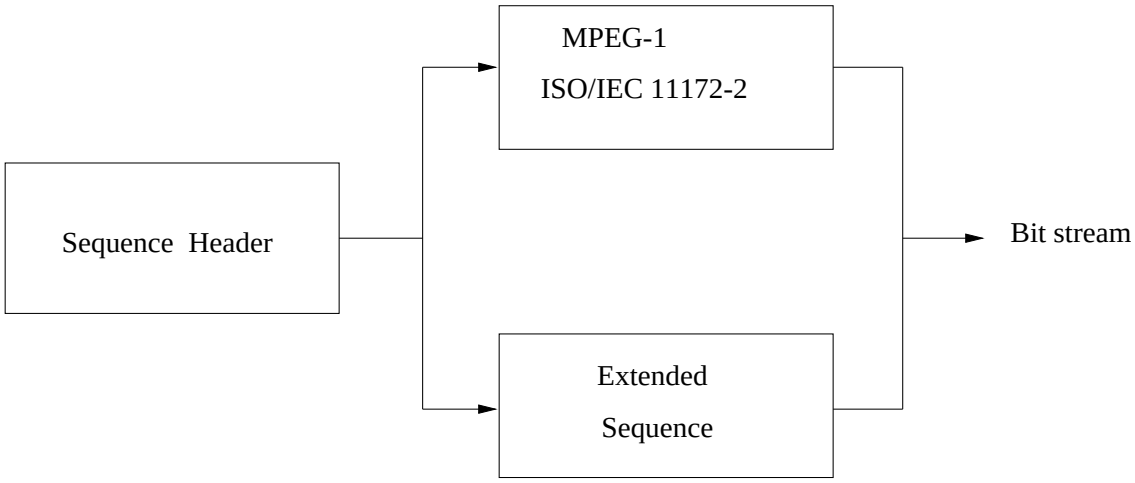


Figure 2.3: Possible routes in MPEG-2 video syntax

Since the syntax of the MPEG-1 video standard already supported very large size pictures and wide range of bit-rates, it became necessary to define a minimum set of universal parameters, which could be decoded by any decoder. Those parameters are called "constrained parameters" and the bit-stream satisfying them are called "constrained bit-streams". The MPEG-1 constrained parameter set is shown in the table 2.1.

Parameter	Upper Bound
Horizontal resolution	720 pixels/line
Vertical resolution	576 lines/frame
No. of macroblocks per frame	396
No. of macroblocks per second	396×25
Frame rate	30 Hz
Motion vector range	± 64
Bit rate	1.856 Mbps

Table 2.1: MPEG-1 Constrained Parameter Bounds

Though MPEG-2 video decoder should be able to decode any MPEG-1 video bit-stream, the standard expects that it should be able to decode at least the constrained bit-streams.

2.2 Different Syntax Layers

The layered structure of the MPEG-2 data stream facilitates flexibility and efficiency in the decoder. Coding process can be logically distinct and the layers can be decoded systematically. Each layer supports a specific function as described in the table 2.2.

Layers	Functions
Sequence layer	One or more groups of pictures
Group of pictures layer	Random access into the sequence
Picture layer	Primary coding unit
Slice layer	Resynchronization unit
Macroblock layer	Motion compensation unit
Block layer	DCT unit

Table 2.2: Functional comparison of six layers

The top coding layer is called *Sequence Layer* because the video bit-stream is referred to as a sequence in MPEG-2 terminology. It consists of a sequence header, one or more groups of pictures (GOP), and an end-of-sequence code. Any valid video sequence must end with the sequence-end-code.

The *GOP Layer* is a set of pictures in the contiguous display order. A GOP contains at least one I-frame. In MPEG-2, GOP headers are optional. The figure 2.4 shows the structure of a typical GOP, along with the prediction dependencies between different types of frames.

The *Picture Layer* is a primary coding unit that consists of the luminance and chrominance components of a frame. It defines the coding information for each picture. Picture header contains a temporal reference number that defines the display order of the picture. In addition to that, it contains information about the picture type (I,P,B or P). Each picture is divided into slices. A slice is a string of consecutive

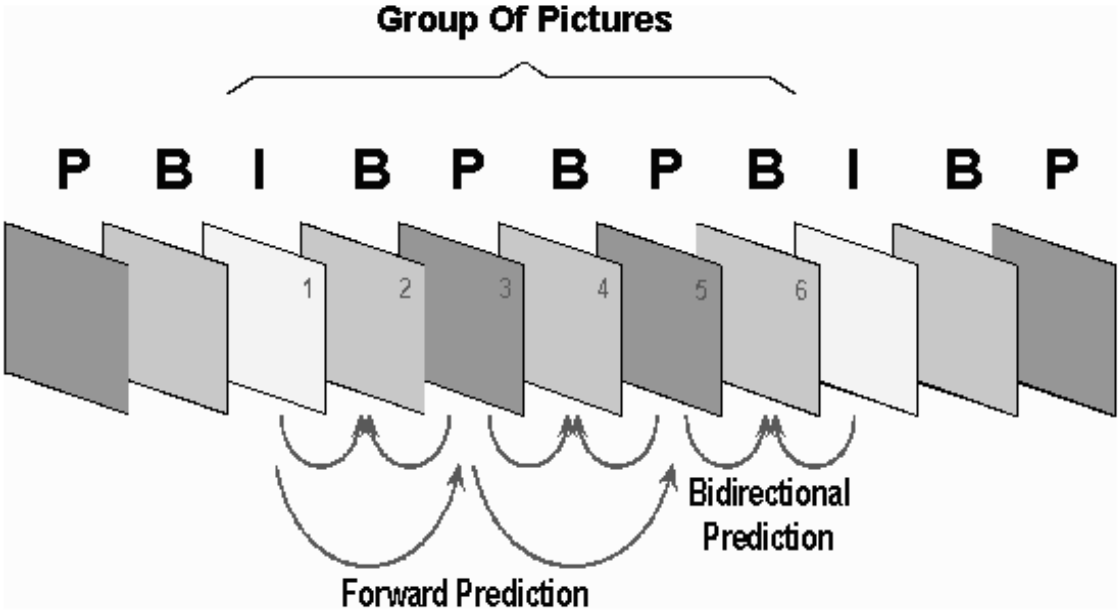


Figure 2.4: MPEG-2 GOP structure

macroblocks of arbitrary length from left to right across the whole picture. MPEG-2 requires that the slices begin and end in the same macroblock row. The slice header contains information about its position in the picture and the quantizer scale factor.

The *Slice Layer* is important in handling errors. The decoder can skip the corrupted slices and go to the start of the next slice, if the bit-stream is corrupted somehow. It contains a number (≥ 1) of macroblocks (MB). A macroblock is a $16pixel \times 16pixel$ motion compensation unit. Each macroblock begins with a Header which defines the macroblock type, motion vector type, quantizer scale code and a coded block pattern (CBP) indicating which blocks in the macroblock are actually coded. Macroblock coding is a complex operation, but all the overhead is in the encoder. It is the responsibility of the encoder to decide where a picture should be coded as I-frame, P-frame or B-frame. In fact even within an individual I-frame, P-frame or B-frame, macroblocks can be coded differently. The job of the decoder is to follow the coding of the encoder and process the macroblocks accordingly.

The *Macroblock Layer* defines a number of blocks depending on the chroma format (maximum is 6 for 4:2:0, 8 for 4:2:2 and 12 for 4:4:4) used. Some of the blocks may not be coded as indicated in CBP.

The *Block Layer* contains the actual data for the quantized DCT coefficients of an 8×8 block in the macroblock. The blocks are coded using variable length code (VLC) tables. The overhead of choosing which blocks are to be coded in bestowed upon the encoder. The decoder only needs to decode the CBP and use the VLC tables to find out the DCT data and then perform inverse quantization and 8×8 IDCT operation respectively on it.

2.3 Inverse DCT

The IDCT coefficients are not always the actual pixel[1] values as many of the MPEG-2 frames (P-frames and B-frames) are inter-coded to take the advantages of temporal redundancies between successive frames in a normal video sequence. However, if the frame is intra-coded, then the IDCT values are directly sent for processing required for display.

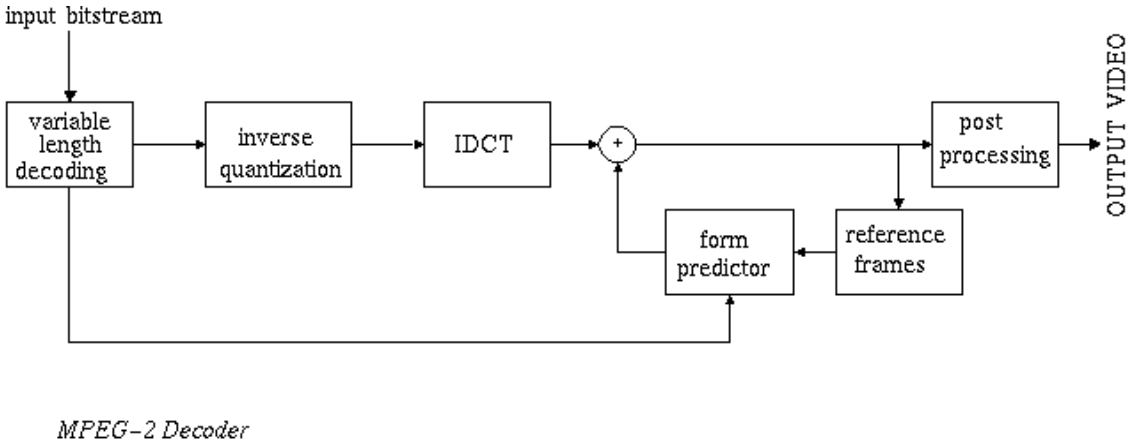


Figure 2.5: A typical MPEG-2 decoder structure

As shown in the figure 2.5, the IDCT coefficients for inter-coded frames are then added to the predictions generated by the form predictor to generate the actual frame if the frame is inter-coded. The form predictor takes the the motion-vectors coded in the bit-stream and reference frames to generate the predictions for the next frame in the coding order. If this decoded frame is not a B-frame, then it is stored in the buffer because it will be the next reference frame.

[1] *pel* in MPEG terminology

The frame is then displayed on the VDU if it is the next frame in the display order, i.e. a frame rearrangement is done prior to the final display. Some additional processing is required for the frame in case the user wants to see it in *zoom mode*.

2.4 Design Issues

The design decisions of the video decoder were largely affected by the following issues.

- Speed - The speed (rather frame-rate) is a big issue for any real-time application. We tried to meet the deadline of the coded frame-rate. We have used integer-arithmetic wherever possible because it is much faster compared to the floating-point arithmetic. For a slow processor or system, the decoder may discard some of the frames and meet the frame-rate requirement of the particular video-sequence in order to achieve the coded frame-rate. The current implementation discards only the B-frames. If the number of skipped frames become very large, there may be some kind of jerkiness in the display.

- Picture quality - Due care has been taken to preserve the required picture quality. The overall quality of the reconstructed frames should be as similar as possible to the original ones. Many implementations do not take into account all the IDCT basis functions during the IDCT computations in order to achieve greater frame-rate. But that may affect the overall video quality. In the current implementation, we take all the IDCT basis functions for computations.

- Precision - In MPEG-2, pixel values of one picture are often used in the reconstruction of the subsequent pictures. So careful attention should be paid to the accuracy of the all arithmetic computations, such that the noise due to arithmetic precision does not influence the overall picture quality.

- Compatibility - The decoder should be forward (a new generation decoder's capability to decode bit-streams created by an existing encoder) and upward (a higher-resolution decoder's capability to decode bit-streams created by a low-resolution encoder) compatible with the exiting MPEG-1 encoders, meaning it should be able to decode the bit-streams created by the MPEG-1 encoders.

- Error resilience - The decoder should be able to understand and if possible, recover any kind of eventual errors.

Chapter 3

Implementation Details

In the current implementation, the MPEG-2 decoder decodes the video data by itself and sends the audio data to an audio decoder in its expected format for playing. It is however the responsibility of the video decoder to ensure real-time operation of the player. In this chapter the outlines of the various algorithms have been presented.

3.1 Inverse Discrete Cosine Transform

Inverse Discrete Cosine Transform computation is one of the most important part of MPEG-2 decoding. As IDCT operation on the whole image at a time is highly compute-intensive, MPEG-2 recommends IDCT on the basis of 8×8 block size. Thus the pixel values are computed by :

$$x_{ij} = \frac{1}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} u(m)u(n)Y_{mn}\, cos\frac{(2i+1)m}{16}\pi\, cos\frac{(2j+1)n}{16}\pi \tag{3.1}$$

where $0 \leq i, j \leq 7$.

It may be noted that for most of the blocks, the DCT coefficients Y_{mn} exhibit high energy compaction, i.e., the most significant values are contained in just a few coefficients and the rest of them are insignificant as shown in the figure 3.1.

As shown in the figure 3.1, on an average only a small number of DCT coefficients need to be transmitted to the receiver to obtain a valuable approximate reconstruction

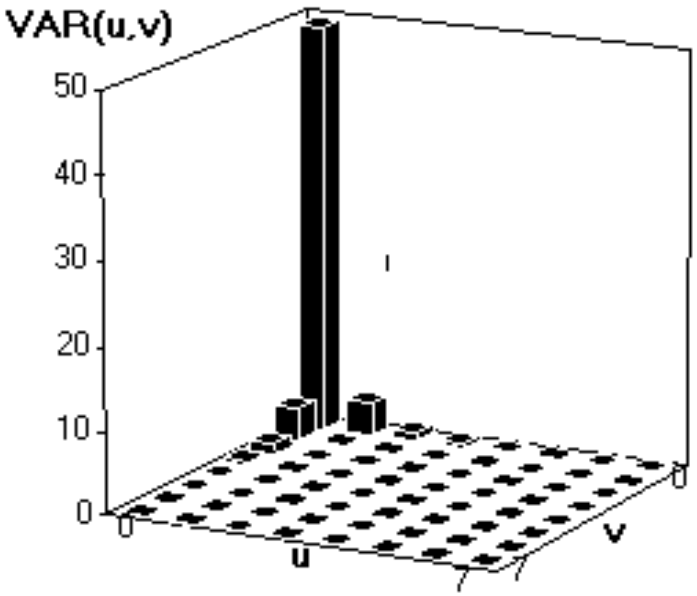


Figure 3.1: Typical energy distribution of DCT

of the image blocks. Moreover, the most significant DCT coefficients are concentrated around the upper left corner (low DCT coefficients) and the significance of the coefficients decays with increased distance. This implies that higher DCT coefficients are less important for reconstruction than lower coefficients. Keeping this in mind, it was possible to optimize the IDCT computations significantly.

We have observed that in many cases, the entire 8×8 block contains only one non-zero DCT coefficient Y_{00}. Therefore, the pixel values of all the 64 elements in the block is same ($= \frac{1}{8}Y_{00}$). However if the 8×8 block contains more than one non-zero coefficients then we need to transform the 2-d IDCT computation into into 16 1-D IDCTs using row-column approach. For 1-d IDCT we used the algorithm proposed by B.G.Lee [8] whose complexity is $O(N\log_2 N)$ for one N-point IDCT. And so the time complexity of 2-d IDCT computation for a single block in our implementation is $O(N^2\log_2 N)$ which is fairly good as compared to the $O(N^4)$ complexity of the original IDCT formula given by the equation 3.1. Therefore the overall time complexity of evaluating one frame comes out to be $O(M^2\log_2 N)$ where the frame size is $M \times M$ and block size is $N \times N$.

3.1.1 Lee's Algorithm to evaluate 1-d IDCT

If we denote the DCT of the data sequence $x(k), k = 0, 1, ..., N-1$, by $X(n), n = 0, 1, \ldots, N-1$, then we have [1]

$$x(k) = \sum_{n=0}^{N-1} e(n)X(n)\, cos\frac{(2k+1)n}{2N}\pi, \tag{3.2}$$
$$k = 0, 1, 2, \ldots, N-1$$

where

$$e(n) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if n = 0} \\ 0, & \text{otherwise} \end{cases}$$

We consider the equation 3.2, which is the inverse DCT (IDCT), and define C such that

$$C_{2N}^{(2k+1)n} = cos\frac{(2k+1)n}{2N}\pi.$$

Then N-point IDCT becomes

$$x(k) = \sum_{n=0}^{N-1} \hat{X}(n)\, C_{2N}^{(2k+1)n}, \tag{3.3}$$
$$k = 0, 1, 2, \ldots, N-1$$

where $\hat{X}(n) = e(n)X(n)$

Now if we define,

$$g(k) = \sum_{n=0}^{N/2-1} \hat{X}(2n)\, C_{2(N/2)}^{(2k+1)n}, \tag{3.4}$$

and

$$h(k) = \sum_{n=0}^{N/2-1} [\hat{X}(2n-1) + \hat{X}(2n+1)]\, C_{2(N/2)}^{(2k+1)n}, \tag{3.5}$$
$$k = 0, 1, 2, \ldots, N-1$$

then it has been shown [8] that the N-point IDCT in the equation 3.3 can be rewritten as

$$x(k) = g(k) + (1/(2C_{2N}^{2k+1}))h(k)$$
$$x(N-1-k) = g(k) - (1/(2C_{2N}^{2k+1}))h(k),$$
$$k = 0, 1, 2, \ldots, N-1$$

Therefore, the N-point IDCT in the equation 3.3 can be decomposed into sum of two $N/2$-point IDCT's in the equations 3.4 and 3.5. By repeating this process, we can decompose the IDCT further.

The figure 3.2 shows the flow graph for an eight-point IDCT using Lee's algorithm.

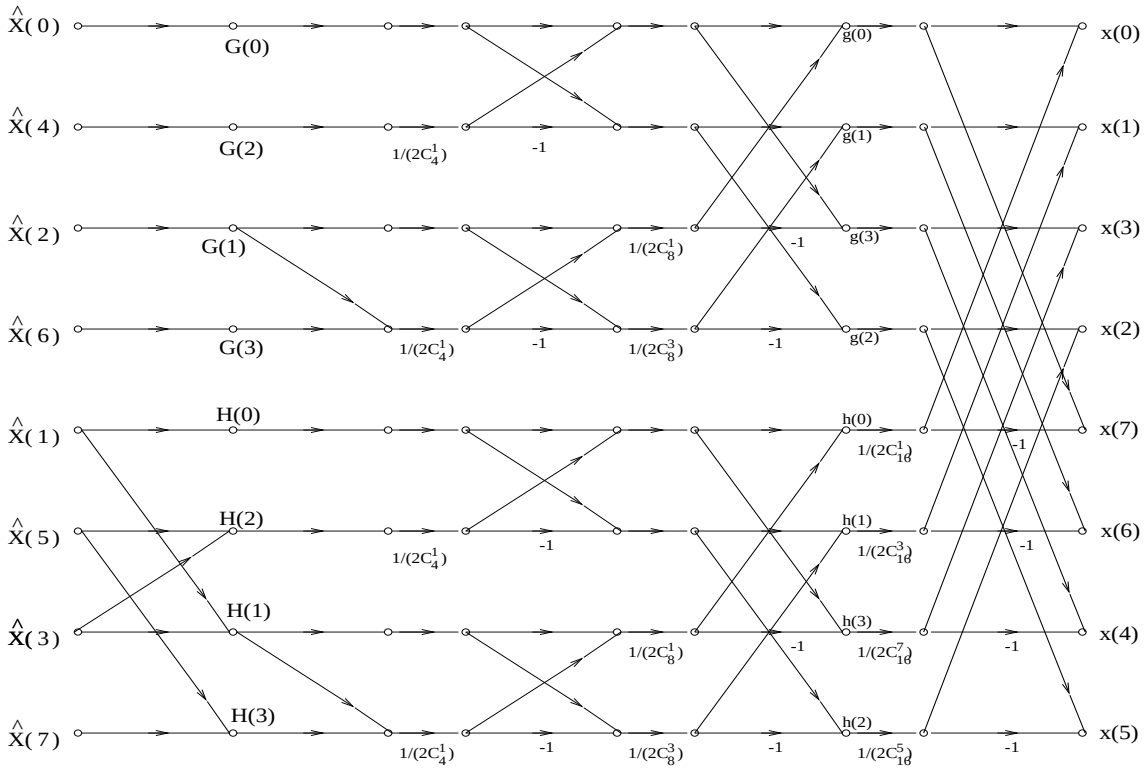


Figure 3.2: Flow graph for 8-point IDCT

It is evident that the structure shown is recursive and simple. In the current implementation, table-lookup approach is used wherever possible to avoid CPU-intensive cosine computations and multiplications.

The number of real multiplication is $(N/2)\log_2(N)$ for an N-point IDCT where $N = 2^m$, which is about half the number required by existing efficient algorithms. The number of additions, however, is $(3N/2)\log_2(N) - N + 1$.

Since only a discrete number of cosine function values are needed, these are precomputed and kept in a table. Later this table is used instead of re-computing the cosine functions. This save enormous amount of computation.

3.1.2 Pseudocode to calculate IDCT

The pseudocode given below calculates IDCT for an 8×8 block. The DCT coefficients are stored in an one-dimensional array *block* and the output IDCT coefficients are also returned in *block*.

```
idct(block)
{
    if(eob_pos == 0) /* if only DC coefficient */
    {
        for(ii=0; ii<64; ii++)
            block[ii] = block[0]/8;
    }
    else
    {
        for(row=0; row<8; row++)
            idctrow(row);
        for(col=0; col<8; col++)
            idctcol(col);
    }
}
```

3.2 Bit-stream decoding

3.2.1 Start codes

In MPEG-2 data stream, the system and video layers contain unique byte-aligned 32-bit patterns called start codes. There are 256 start codes provided in MPEG-2, some of which are not used. Some of these start codes are given in the table 3.1. The video start codes (0x00000100 through 0x000001B8) are found only in the video syntax layers while the system start codes (0x000001B9 through 0x000001FF) are found only in the system syntax.

In the MPEG-2 bit-stream, start codes are aligned at the byte boundaries by

Start code	Value (in Hexadecimal representation)
video start codes	
PICTURE_START_CODE	0x00000100
SLICE_START_CODE 1	0x00000101
...	
SLICE_START_CODE 175	0x000001AF
RESERVED	0x000001B0
RESERVED	0x000001B1
USER_DATA_START_CODE	0x000001B2
SEQUENCE_HEADER_CODE	0x000001B3
SEQUENCE_ERROR_CODE	0x000001B4
EXTENSION_START_CODE	0x000001B5
RESERVED	0x000001B6
SEQUENCE_END_CODE	0x000001B7
GROUP_START_CODE	0x000001B8
system start codes	
ISO_END_CODE	0x000001B9
PACK_START_CODE	0x000001BA
SYSTEM_HDR_START_CODE	0x000001BB
packet start codes	
RESERVED	0x000001BC
PRIVATE_STREAM 1	0x000001BD
PADDING_STREAM	0x000001BE
PRIVATE_STREAM 2	0x000001BF
AUDIO_STREAM 0	0x000001C0
AUDIO_STREAM 31	0x000001DF
VIDEO_STREAM 0	0x000001E0
VIDEO_STREAM 15	0x000001EF
RESERVED 0	0x000001F0
RESERVED 05	0x000001FF

Table 3.1: Start Codes used in MPEG-2

inserting necessary number of bits as 0. The decoder therefore requires to align at byte boundaries. The next_start_code() procedure shown below positions the bit-stream pointer at the start of the next start code.

```
next_start_code()
{
     while(!bytealigned())
          zero_bit();
     while(Nextbits(24) != 0x000001)
          zero_byte();
}
```

The bytealigned() returns whether the bit-stream pointer is at a byte boundary or not.

```
bytealigned()
{
     if(bit-stream pointer % 8 == 0)
          return 1;
     else
          return 0;
}
```

The functions Nextbits looks ahead in the bit-stream and returns the next n bits of the bit-stream without altering the current position of the bit-stream buffer.

3.2.2 System Layer

Though a video bit-stream is fully decodable, but by itself, is an incomplete specification. The MPEG system layer contains the control information that enables parsing and precise control of the playback of the bit-stream. It is used to multiplex one or more video and audio streams into a single bit-stream. From system perspective, an MPEG bit-stream is made up of a system layer and compression layers.

The highest level in systems layer consists of a sequence of packs followed by a four-byte ISO_END_CODE (0x1B9). Each pack consists of a unique 32-bit byte-aligned PACK_START_CODE (0x1BA) and a header. Following pack header is a variable-length system header. A pack is further subdivided into a number of packets. A packet starts with a 32-bit PACKET_START_CODE (0x1BC-0x1FF), followed by a packet header. Part of the packet header is packet-length which is the length of the rest of the packet. The video, audio, padding or private streams follow the packet header as packet-data-bytes. All of the streams in a given packet are of same type, as specified by the stream-id (final byte of packet_start_code).

The figure 3.3 shows the layered structure of MPEG-2 system.

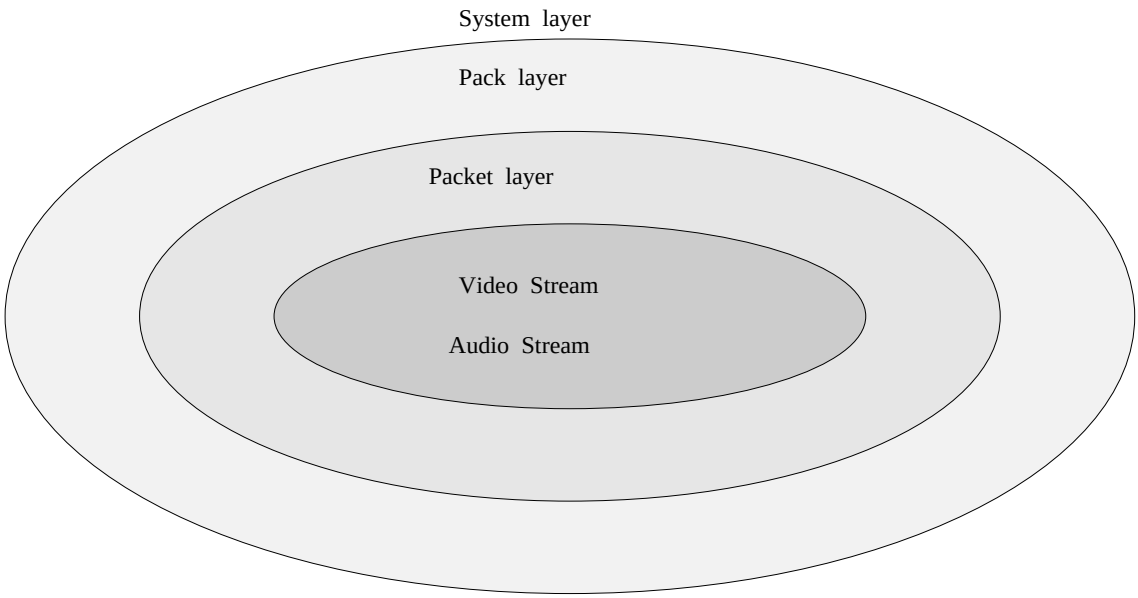


Figure 3.3: Layered structure of MPEG-2 system

In the current implementation, the video decoder parses the bit-stream and decodes the data by itself when it gets video data, but in case of audio data, it sends the

data to an audio decoder which then plays the audio. The system layer information is taken into account and used to demultiplex the video and the audio data and to achieve synchronized play-back of the movie clips. The overall operation of the video decoder is depicted in the following pseudocode.

```
for(;;)
{
    next_pack();
    nextval = Nextbits(32);
    switch(nextval)
    {
        case PACK_START_CODE :
            pack-header();
            break;
        case PADDING_STREAM :
        case PRIVATE_STREAM :
            discard_data();
            break;
        case SYSTEM_HDR_START_CODE :
            system_header();
            break;
        case VIDEO_STREAM :
            video_data();
            break;
        case AUDIO_STREAM :
            audio_data();
            sendto_audioplayer();
            break;
        case ISO_END_CODE :
            show_statistics();
            break;
    }
}
```

3.2.3 Video Stream

MPEG-2 follows a layered structure as described in the section 2.2. The figure 3.4 shows the inter-connection between different structures. A group of pictures comprises of several frames (or pictures). A picture contains several slices of macroblocks. Each macroblock comprises of four blocks of size 8 pixel by 8 pixel.

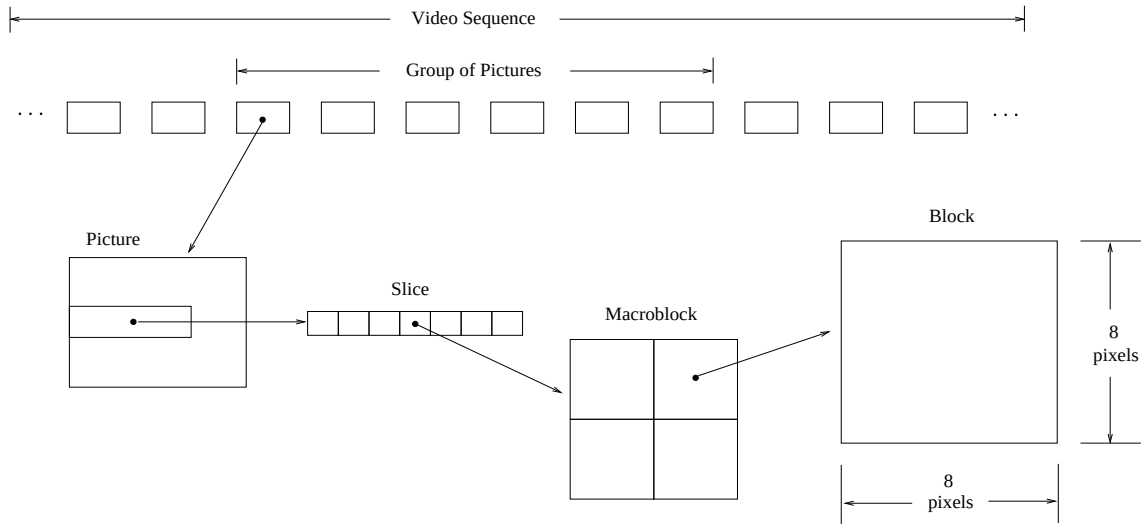


Figure 3.4: MPEG-2 video structure

The pseudocode below defines the syntax for an MPEG-2 video sequence.

```
video_sequence()
{
     next_start_code();
     sequence_header();
     if(Nextbits(32) == EXTENSION_START_CODE)
     {
          sequence_extension();
          do
          {
               extension_and_user_data_V();
               do
               {
```

```
                    if(Nextbits(32) == GROUP_START_CODE)
                    {
                         group_of_pictures_header();
                         extension_and_user_data_G();
                    }
                    picture_header();
                    picture_coding_extension();
                    extension_and_user_data_P();
                    picture_data();

               } while((Nextbits(32) == PICTURE_START_CODE) ||
                         (Nextbits(32) == GROUP_START_CODE))

               if(Nextbits(32) != SEQUENCE_END_CODE)
               {
                    sequence_header();
                    sequence_extension();
               }

          } while(Nextbits(32) != SEQUENCE_END_CODE)
     }
     else
     {
          /* MPEG-1 bit-stream*/
     }
     sequence_end_code();
}
```

As it can be observed in the pseudocode, every valid MPEG-2 bit-stream must start with *sequence header* and end with *sequence_end_code*. If the first sequence header in the bit-stream is followed immediately by the extension_start_code, then every sequence header must be followed by the sequence extension. Optional extension and user data often follow the sequence extension.

The video decoder extracts the basic 8×8 block data (DCT coefficients) from the video-stream. The 8×8 DCT coefficients are arranged in a special 1-d sequence called scanning order. An order called zigzag order (figure 3.5) is used in both MPEG-1 and MPEG-2. In this order, the DCT coefficients are stored in the order of increasing spatial frequency.

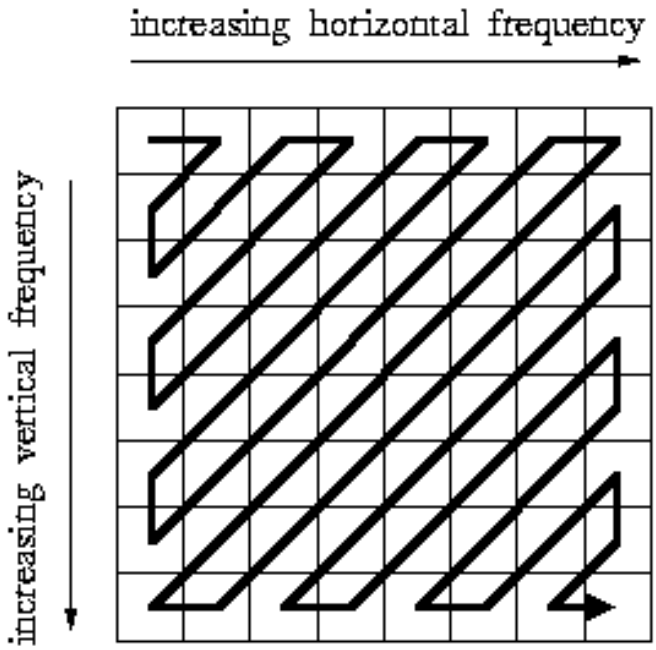


Figure 3.5: Zigzag scan order

MPEG-2 also defines another scan order, known as *alternate or vertical scan* order that may be specified by the encoder on a picture to picture basis. The alternate scanning order is shown in the figure 3.6.

A typical DCT array in the scan order will have significant values only in the beginning and zeros at the end. Since typically there are many zeros in a DCT block, these need not be stored. For this feature, a special symbol called End-of-block (EOB) is used in MPEG-2 as well as in MPEG-1. When this symbol is used, it represents that all remaining DCT coefficients for the block are zero. However, when the entire block of DCT data contains only zero, the bit corresponding to the block is reset to zero in the coded block pattern (CBP) and the entire block is not stored. Sometimes, even a whole macroblock is skipped if all the blocks of the macroblock are zero. This is accomplished by simply setting the macroblock_address_increment to a value greater

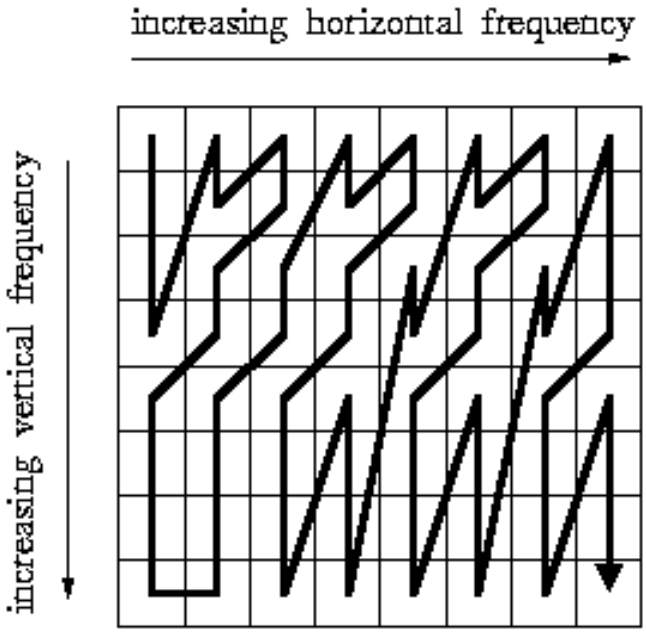


Figure 3.6: Alternate scan order

than one (the number of macroblocks skipped minus one). However, in an I-frame, macroblock skipping is not permitted.

3.2.4 Audio Stream

The MPEG audio coders, aimed for generic audio coding, are perceptual audio coders, rather than 'waveform coders'. In a perceptual audio coder, the codec does not retain the input signal exactly after encoding, rather it tries to ensure that the output signal sounds the same as the original to a human listener. The primary psychoacoustic effect that the perceptual audio coder uses is called 'auditory masking', where some parts of the input signal are not audible due to the inherent nature of the human auditory system. The parts of the signal that are masked are commonly called 'irrelevant', as opposed to the parts of the signal that are treated as 'redundant' and so removed by a source coder. Three layers have been defined for MPEG audio, namely Layer I, Layer II and Layer III. These layers represent a set of coding algorithms, with encoder/decoder complexity increases from Layer I to Layer III. In all the layers, the audio data is a sequence of audio-frames. Each frame starts with a 32-bit

frame-header, first 12 bits (all 1s) of which is called syncword. The other bits of the frame-header contains information like layer-code, bit-rate, sampling frequency, etc. Following the frame-header, a 16-bit Cyclic Redundancy Code can be optionally present. Next the actual audio-data is coded, the size of which can be calculated from the header information itself.

In the current implementation the audio-decoding is not handled by the video decoder. Instead the audio data is extracted from the bit-stream, formatted and arranged in a sequence of audio-frames as expected by an audio-only decoder [16] and then sent to an audio decoder for playing.

3.3 VideoCD Support

As the cost of CD-ROMs are decreasing with the invention of better technologies for mass-production, a significant number of customers are now switching to CD-ROMs as general purpose storage and so Philips has developed a new standard for storage of MPEG data on CD-ROM called *White Book*, commonly known as video-CD or VCD [19]. A Linux kernel that can read VCDs is required to play from VCD. In the current implementation, the xreadvcd [20] tool is used to read from VideoCD. The tool is developed by Ales Makarov who is the author of the kernel patch to read VideoCD which is included in Linux version 2.2.

3.4 Graphical User Interface

The current implementation provides a graphical user interface for the easy use of the decoder. The GUI was developed using *XForms* [21], a GUI toolkit based on Xlib for X Window Systems. The form was designed by *fdesign*, an interactive GUI builder bundled with the XForms Library. The interface screen is shown in the figure 3.7. It was generated by the tool *fd2ps* of the package.

A brief overview of the menus and the buttons are given below.

- File - The File menu has three options, namely "Play File", "Play VCD" and "Exit". If the user selects "Play File", a file selector window comes up and the user can choose an MPEG file to view. If "Play VCD" option is chosen, it plays

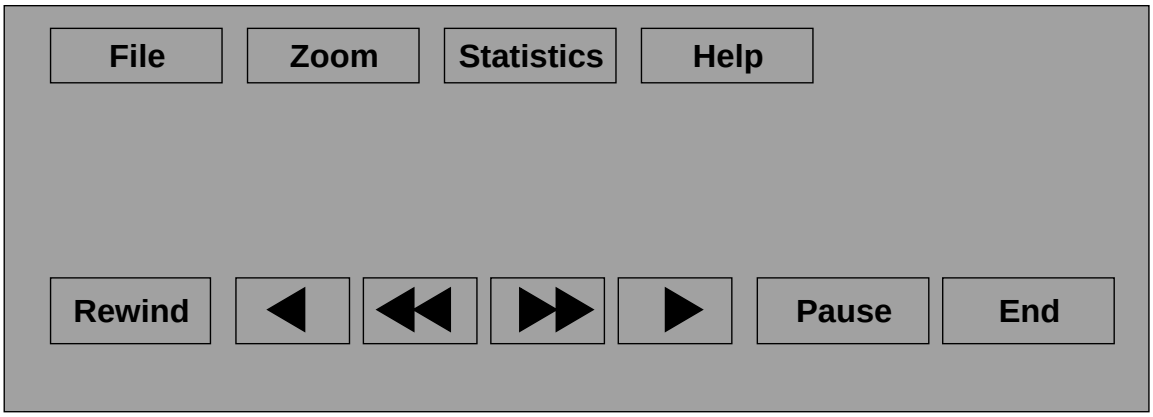


Figure 3.7: Graphical User Interface

the MPEG data on the VideoCD. On choosing "Exit", the user can quit the application.

- Zoom - The Zoom menu has three options, namely "Normal", "Double Size" and "Full Screen". The video will be displayed at the size as chosen by the user.
- Statistics - The user can click on this button, if he wants to see the statistics like the number of frames played, time elapsed, frame-rate, etc. anytime during the current run.
- Help - The user can click on this button, if he wants to get some help about the video decoder.
- Rewind - The user can click on this button, if he wants to view the last played MPEG file again from the beginning again.
- Reverse - Two buttons have been provided for this purpose. One is for normal reverse and another is for fast reverse. The user can click on these buttons, if he wants to view the current movie again from a previous frame.
- Forward - Two buttons have been provided for this purpose. One is for normal forward and another is for fast forward. The user can click on these button, if he wants to skip some part of the current movie and view it from a future frame.
- Pause - The user can click on this button, if he wants to stop the current run for a while. Once pressed, the label of the button is changed to "Resume".

- Resume - The user can click on this button, if he wants to resume the current run again. Once pressed, the label of the button is changed to "Pause".
- End - The user can click on this button, if he wants to end the current run of the decoder. If this button is pressed, the Image window is removed, but the MPEG file can be viewed again by clicking on "Rewind" button.

Chapter 4

Results and Conclusions

4.1 Test Setup

The current implementation of the MPEG-2 video decoder was developed on Linux 2.2.5. The decoder was thoroughly tested to check its efficiency and robustness. The table 4.1 shows the configuration of the system.

Resource	Configuration
Processor	Pentium-II (i686)
CPU Clock Speed	350 MHz
Memory	64 MB RAM
Video RAM	4 MB

Table 4.1: Test System configuration

The decoder also needs 8-bit display depth which is supported by all available graphics-cards. It supports 16-bit display mode also.

The audio player used to play the audio data is famous mpg123 [16] written by Michael Hipp and Oliver Fromme. It is a real time MPEG audio player for Unix.

In order to achieve the coded frame-rate, the current implementation of the video decoder discards some frames when otherwise real-time deadline cannot be met. So if the system is slow, many frames will be discarded causing somewhat jerkiness in the video.

4.2 Results

The decoder was run with many MPEG files as input for several times. The table 4.2 and 4.3 show the average results taken after 10 different runs for each of the sample MPEG files.

We list in the table 4.2 the frame-size, the total number of frames and the percentage of the frames skipped by the decoder in order to achieve the coded frame-rate in Normal mode, Double Size Zoom and Full Screen Zoom for each sample. It is evident that the computational overhead for zoom mode display causes more frames to be skipped.

Sample no.	Frame Size	Total no. of frames	% of frames skipped		
			Normal	Double Size Zoom	Full Screen Zoom
1.	352 × 288	1150	10	15	20
2.	352 × 240	1004	5	14	18
3.	352 × 240	60	11	20	25
4.	320 × 240	278	0	0	0
5.	512 × 320	215	19	25	30

Table 4.2: Results : Percentages of frames skipped at different zoom levels

We list in the table 4.3 the coded frame-rate (number of frames per second) and the achieved frame-rate for each sample in Normal mode, Double Size Zoom and FullScreen Zoom.

Sample no.	Coded frame-rate	Achieved frame-rates		
		Normal	Double Size Zoom	Full Screen Zoom
1.	25.000	25.003	24.989	25.012
2.	30.000	30.267	30.078	30.257
3.	30.000	30.142	30.642	30.045
4.	22.977	22.399	22.399	22.399
5.	25.000	25.087	25.128	25.141

Table 4.3: Results : Achieved frame-rates at different zoom levels

4.3 Comparison with existing players

In the table 4.4 a comparitative study with two existing MPEG players for linux, viz. xanim and mpeg-tv is presented. Though they differ fundamentally with our work in that they do not support MPEG-2 standard, some other features have been compared here.

Important Features	Different MPEG Players		
	mplay	xanim	mpeg-tv
video quality[1]	smooth	jerky	smooth
picture quality [1]	good	not good	moderate
audio support	supported	unsupported	supported
frame-types supported	all	only I-frame	all
frame-rate	nominal (coded)	non-uniform	less in zoom mode
frames skipped	less	n/a	not documented
VCD support	supported	unsupported	supported
GUI support	good	few features	good
zoom levels supported	any (dynamic)	any (static)	Double (2x2) only
Full Screen zoom	supported	not truly	supported

Table 4.4: Comparison with existing players

4.4 Conclusions

In spite of considerable effort to make the decoder very efficient, for many cases it was not possible to achieve the coded frame-rate without dropping some of the frames. Yet we think the current implementation is reasonable taking into account the overall video quality and speed. The IDCT computations are also extremely accurate.

[1] comparisons are qualitative only

Appendix A

A.1 List of Acronyms

AAC	Advanced Audio Coding
CBP	Coded Block Pattern
CD-ROM	Compact Disk - Read Only Memory
CPB	Constrained Parameter Bit-stream
DCT	Discrete Cosine Transform
DSM	Digital Storage Media
DSP	Digital Signal Processing
DVD	Digital Versatile Disk
EOB	End Of Block
FDCT	Forward Discrete Cosine Transform
FFT	Fast Fourier Transform
FPS	Frame Per Second
GOP	Group Of Pictures
GUI	Graphical User Interface
HDTV	High Definition Television
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
IQ	Inverse Quantization
IS	International Standard
ISO	International Organization for Standardization
ITU	International Telecommunications Union

JPEG	Joint Photographic Experts Group
JTC	Joint Technical Committee
KLT	Kerhunen-Loève transform
LFE	Low-Frequency Enhancement
MB	Macroblock
MBA	Macroblock Address
MC	Motion Compensation
MPEG	Moving Picture Experts Group
MSSG	MPEG Software Simulation Group
MV	Motion Vector
PES	Packetsized Elementary Stream
PS	Program Stream
RLC	Run-Length Coding
SCR	System Clock Reference
SIF	Source Input Format
SM	Simulation Model
SNR	Signal-to-Noise Ratio
STC	System Time Clock
TM	Test Model
TS	Transport Stream
VBR	Variable Bit Rate
VBV	Video Buffering Verifier
VCD	Video CD
VDU	Video Display Unit
VLC(D)	Veriable Length Coding (Decoding)
VLSI	Very Large Scale Integration
WG	Working Group

Bibliography

[1] N.Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", IEEE Trans Commun., vol. COM-23, pp 90-93, Jan. 1974

[2] Nam Ik Cho and Sang Uk Lee, "Fast Algorithm and Implementation of 2-D DCT", IEEE Trans. on CAS, vol 38, pp 297-305, Mar. 1991

[3] W.H. Chen, C.H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform,", IEEE Trans. Comm., vol. COM-25, pp. 1004-1009, Sep. 1977.

[4] M.A.Haque, "A two-dimensional cosine transform", IEEE Trans. on Accoust., Speech and Signal Processing, vol ASSP-33, pp. 1532-1539, Dec. 1985

[5] B. G. Haskell, A. Puri and A. N. Netravali, "Digital Video: An Introduction to MPEG-2," Chapman & Hall, 1997.

[6] F.A. Kamangar and K.R. Rao, "Fast algorithms for the 2-D discrete cosine transform", IEEE Trans. Comput., vol. C-31, pp. 899-906, Sep 1982.

[7] K.R. Rao and J.J. Hwang, "Techniques and Standards for Image, Video, and Audio Coding", Prentice Hall, 1996.

[8] B.G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform", IEEE Trans. on Accoust., Speech and Signal Processing, vol ASSP-32, pp. 1243-1245, Dec. 1984

[9] Leonardo Chairiglione, "The Development of an Integrated Audiovisual Standard : MPEG", Proceedings of the IEEE, Vol 83, No. 2, Feb. 1995.

[10] J.L.Mitchell, W.B.Pennebaker, C.E.Fogg and D.J.LeGall, "MPEG Video Compression Standard", in Digital Multimedia Standards Series,Chapman & Hall, New York, 1997.

[11] MPEG Homepage
http://www.cselt.it/mepg/

[12] MPEG Video SubGroup Homepage
http://bs.hhi.de/mpeg-video/

[13] MPEG Research at UC Berkeley
http://bmrc.berkeley.edu/frame/research/mpeg/

[14] MPEG-1 implementation in C++
http://www.itacs.uow.edu.au/people/nabg/MPEG/MPEG0.html

[15] MPEG Developing Classes
http://vision.cs.wayne.edu/mpeg/

[16] Homepage of mpg123 - an MPEG audio player for Unix
http://dorifer.heim3.tu-clausthal.de/~olli/mpg123/

[17] MPEG Audio SubGroup Homepage
http://www.tnt.uni-hannover.de/project/mpeg/audio/

[18] K.R. Rao and P.Yip, "Discrete Cosine Transform : Algorithms, Advantages, and applications", Academic Press Inc., 1990.

[19] VideoCD
http://www.c-cube.com/technology/videocd.html

[20] Homepage of xreadvcd - a tool to read VideoCD in Linux
ftp://mca.sh.cvut.cz/pub/readvcd

[21] XForms Library
http://bragg.phys.uwm.edu/xforms

Index