AN APPROACH TO SUMMARIZE VIDEO DATA IN COMPRESSED DOMAIN

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Electrical and Electronics Engineering

by
Gökhan ŞİMŞEK

May 2007
İZMİR

We approve the thesis of **Gökhan ŞİMŞEK**

Date of Signature

. . . . . **10 April 2007**

Assist. Prof. Dr. Şevket GÜMÜŞTEKİN

Supervisor

Department of Electrical and Electronics Engineering

Izmir Institute of Technology

. . . . . **10 April 2007**

Prof. Dr. F. Acar SAVACI

Department of Electrical and Electronics Engineering

Izmir Institute of Technology

. . . . . **10 April 2007**

Assist. Prof. Dr. Mustafa ALTINKAYA

Department of Electrical and Electronics Engineering

Izmir Institute of Technology

. . . . . **10 April 2007**

Assist. Prof. Dr. Aylin KANTARCI

Department of Computer Science

Ege University, İZMİR

. . . . . **10 April 2007**

Assist. Prof. Dr. Zekeriya TÜFEKÇİ

Department of Electrical and Electronics Engineering

Izmir Institute of Technology

. . . . . .

Assist. Prof. Dr. Barış ÖZERDEM

Dean Of The Graduate School

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Assist. Prof. Dr. Şevket Gümüştekin for his continious support througout my study. I also want to thank to the committee for the encouragement to support willingness to research new ideas and techniques whıch form the basis of this study.

I extend my thanks to my friends and colleagues in Beko Electronics R&D group for giving me enthusiasm to learn, supplying valuable ideas to improve my study and feeding me with valuable resources.

Finally, I would like to my family for their support and patience.

Gökhan ŞİMŞEK

ABSTRACT

AN APPROACH TO SUMMARIZE VIDEO DATA IN COMPRESSED DOMAIN

The requirements to represent digital video and images efficiently and feasibly have collected great efforts on research, development and standardization over past 20 years. These efforts targeted a vast area of applications such as video on demand, digital TV/HDTV broadcasting, multimedia video databases, surveillance applications etc. Moreover, the applications demand more efficient collections of algorithms to enable lower bit rate levels, with acceptable quality depending on application requirements. In our time, most of the video content either stored, transmitted is in compressed form. The increase in the amount of video data that is being shared attracted interest of researchers on the interrelated problems of video summarization, indexing and abstraction.

In this study, the scene cut detection in emerging ISO/ITU H264/AVC coded bit-stream is realized by extracting spatio-temporal prediction information directly in the compressed domain. The syntax and semantics, parsing and decoding processes of ISO/ITU H264/AVC bit-stream is analyzed to detect scene information. Various video test data is constructed using Joint Video Team's test model JM encoder, and implementations are made on JM decoder. The output of the study is the scene information to address video summarization, skimming, indexing applications that use the new generation ISO/ITU H264/AVC video.

ÖZET

SIKIŞTIRILMIŞ BÖLGEDE VİDEO VERİLERİNİN ÖZETLENMESİ İÇİN BİR YAKLAŞIM

Geçirdigimiz 20 yıl içerisinde, sayısal video ve görüntülerin verimli ve ucuz bir şekilde ifade ihtiyacı büyük bir araştırma, geliştirme ve standartlaştırma gayreti toplamistir. Bu gayretler sayisal yüksek standart televizyon yayınları, çokluortam video veritabanları, gözetim uygulamaları vb gibi çesitli uygulamaları hedeflemektedir. Bunun yaninda, bu uygulamalar bit hizlarını kabul edilebilir video niteliği içerisinde düşüren algoritmalar koleksiyonunlarına ihtiyaç duymaktadır. Günümüzde iletilen ya da kaydedilmiş video içeriğinin çoğu sıkıştırılmıs biçimde bulunmaktadır. Video içeriği sayısındaki büyüme video dizinleme, özetleme ve soyutlama gibi problem sahalarınına olan araştırma ilgisini arttırmıştır.

Bu çalışmada doğuş sürecinde önemli yol almış olan ISO/ITU H264/AVC standardında kodlanmış bit dizisi içerisinde uzay-zamansal öngörme bilgisi sıkıştırılmış sahadan doğrudan çıkarılarak sahne geçişleri sezimi gerçeklenmiştir. ISO/ITU H264/AVC standardının sözdizim, anlamsal tanımları ve ayrıştırma, kodçözme süreçleri sahne kesimi sezimi için incelenmiştir. Çeşitli test verileri JM Referans Test Modeli kullanılarak oluşturulmuş ve JM Referans Kodçözme modeli üzerinde gerçekleştirmeler yapılmıştır. Bu çalışmanın çıktısı ISO/ITU H264/AVC kullanan video dizinleme, özetleme ve soyutlama gibi uygulamalarında kullanılabilecek sahne kesik bilgileridir.

TABLE OF CONTENTS

LIST OF TABLES viii
LIST OF FIGURES x

CHAPTER 1. INTRODUCTION 1
1.1. Introduction 1
1.2. Structure of the Thesis 5

CHAPTER 2. BASICS OF VIDEO CODING AND STANDARDS 6
2.1. Introduction 6
2.2. Encoder/Decoder Model 6
2.3. Video Coding Principles 7
2.3.1. Probability and Information Theory for Video Coding 7
2.3.2. Entropy and Mutual Information for Discrete Sources 8
2.3.3. Bounds for Lossless and Lossy Coding 8
2.4. The Video CODEC Internals 10
2.4.1. Temporal Model 11
2.4.1.1. Motion Model 11
2.4.1.2. Block Based Motion Estimation and Compensation 12
2.4.2. Spatial Model 14
2.4.2.1. Predictive Image Coding 16
2.4.2.2. Transform Coding 16
2.4.2.3. DCT Transform 18
2.4.2.4. Quantization 20
2.4.2.5. Reordering and Zero Encoding 23
2.4.2.5.1. DCT Coefficient Distribution and Run-Level Encoding 23
2.4.2.5.2. Run-Level Encoding 23
2.4.2.6. Entropy Coding 24
2.4.2.6.1. Predictive Coding 24
2.4.2.6.2. Variable-length Coding 24
2.4.2.6.3. Huffman Coding 24

2.4.2.6.4. Arithmetic Coding 26
2.4.2.6.5. Binary Arithmetic Coding 28
2.4.2.6.6. Context-Based Arithmetic Coding 28
2.5. The Hybrid Block Based Video Codec 29

CHAPTER 3. OVERVIEW OF H264/AVC STANDARD 31
3.1. Introduction 31
3.2. History and Overview of the Video Coding Standards 31
3.3. H264 Video Coding Standard 34
3.3.1. The Scope of the Standard 34
3.3.2. Applications and New Features of H264/AVC 34
3.3.3. Network Abstraction Layer 36
3.3.4. Video Coding Layer Concepts of H.264/AVC 37

CHAPTER 4. PROPOSED WORK, RESULTS AND FUTURE WORK 41
4.1. Working Environment and Test data 41
4.2. Proposed Work and Results 42
4.2.1. Extracting Slices from NAL unit stream 43
4.2.2. Extracting Slice and Macro-block Information From Bit-stream 43
4.2.3. Detecting Scene Cuts 44
4.2.4. Future Work 49
4.2.4.1. Zoom Detection 49
4.2.4.2. Pan Detection 50

CHAPTER 5. CONCLUSIONS 52

REFERENCES 54

APPENDIX A. CODE IMPLEMENTATION 57

LIST OF FIGURES

Figure **Page**

Figure 1.1. Video skimming process 5
Figure 2.1. Encoder-Decoder Model 6
Figure 2.3. Encoder Blocks 11
Figure 2.4. The Difference of Two :Consecutive Frames 12
Figure 2.5. Macro-block Structure, 4:2:0 Sampling Lattice 13
Figure 2.6. Integer, Half-Pel, Quarter-Pel Motion Estimation 14
Figure 2.7. Autocorrelation function of a Still Image 15
Figure 2.8. Autocorrelation Function of a Residual 15
Figure 2.9. The Effect of Transforming 17
Figure 2.10. DCT Basis Functions Images 19
Figure 2.11. DCT Example 19
Figure 2.12. Uniform Threshold Quantiser (UTQ) Threshold Value th and the Step Size q 20
Figure 2.13. Quantization Schemes 21
Figure 2.14. Vector Quantization Technique 22
Figure 2.15. The Distribution of a Residual Macro-blocks (Non zero Coefficients in lighter gray) 23
Figure 2.16. Example of Huffman Code for 7 Symbols 25
Figure 2.17. Process of Arithmetic Coding Interval. Scaled up at each stage for the message eaii! 27
Figure 2.18. Three Neighbors of symbol x 28
Figure 2.19. Dataflow in encoder 29
Figure 2.20. Data Flow in the Decoder 30
Figure 3.1. Typical MPEG 1 Frame Order 33
Figure 3.2. The Scope of the standard 34
Figure 3.3. The H264 Layers 35
Figure 3.4. Sequence of NAL units each combined with NAL Header and RBSP Data 36
Figure 3.5. Three Profiles of the H264/AVC 38

Figure 3.6. Encapsulated video coding elements and Associated Decoding Function 39
Figure 4.1. Test Data Generation Process for the Proposed Study 42
Figure 4.2. NAL Unit Types of A typical H.264/AVC Stream 43
Figure 4.3. I Type Macro-block Counts vs Frame Number for Trevor + Suzie Sequence 45
Figure 4.4. The output of The program for Trevor and Suzie Sequence 46
Figure 4.5. I Type MB's in P type Slices with Scene cuts as peaks 46
Figure 4.6. The Summary of a Video Sequence from Barry Lyndon (1975) Movie 47
Figure 4.7. A Dissolve Example 47
Figure 4.8. Example of a Dissolve, I Type MB's vs Frame Number 48
Figure 4.9. Example Video Summarization Application 49
Figure 4.10. Ideal Zoom Pattern 50
Figure 4.11. Ideal Pan (Left) and Angle of Motion vectors (right) 50
Figure 4.12. Panning When a Moving Object is on Focus 51

LIST OF TABLES

Table **Page**

Table 2.1. Example Table of Prob. For Different Symbols 27

Table 3.1. Slice Types, Descriptions, Profiles 39

Table 3.2. Macro-Block Syntax Information 40

CHAPTER 1

INTRODUCTION

1.1. Introduction

The amount of multimedia content is growing larger to serve the usage of applications such as DVD Video, digital TV, and Video on demand applications, video databases, and digital libraries, so on.

This huge information content requires automatic searching, summarizing applications and indexing operations to save time for a human operator. Traditionally a human operator needs to navigate through the whole content back and forward manually in order to acquire the portion of the video content which is the sequence of interest. This demand can be answered by automatic and efficient video browsing and summarizing system. A fundamental tool to accomplish this is to parse video sequence into a set of "shots". Shots are defined as "what is captured by the camera between recording start and stop". Boundaries between shots are so called "scene changes" and the action of detection of the boundaries between shots is called "scene change detection" or scene cut detection. (Yeo and Lui 1995a)

Scene cuts and scene changes can also be called as key-frames, because they can be used to represent a distinct shot. (Gargi et al. 1995) The scene changes may be in different forms such as direct cuts, fades, dissolves, wipes etc. (Mee-Sook et al. 1998)

Video parsing is a combination of video segmentation and video indexing. Video segmentation is segmenting the video into shots and video indexing is the stage where labeling of the segmented elemental video element is done (Mee-Sook et al. 1998)

Many research activities are carried on to detect scene change scenarios to provide necessary tools for video parsing.

The existing techniques and systems are carefully investigated and evaluated in (Gargi et al. 1995 and Aslandogan et al. 1999)

The scene cut change methods have several different performance metrics. The computational cost, the ability to detect video shots containing camera operations such as zooming, tiling, panning etc. are some of them.

Based on the above metrics, several detection domains are used: Direct Pixel or Histogram Comparison, compressed domain features of video coding standards, text recognition and closed captions for video indexing.

Generally, regardless of the domain used to detect scene cut; difference metrics are calculated to compare a global threshold to detect the case where the scene change of interest has been occurred.

Pixel domain algorithms use color and luminance intensity distributions that differ from frame to frame. The degree of changes can be measured as a difference of color or luminance histograms.

The pixel domain algorithms are discussed and compared in (Tse et al. 1995) (Yeo and Lui 1995a). The pixel domain algorithms use various comparison techniques such as gray level histograms, sum of gray level differences, the difference of color histograms, and x^2 comparisons of color histograms.

(Otsuji et al. 1991) uses luminance data to compute both frame-based histogram and pixel-based difference between consecutive frames. A cut is detected by searching a "seamed point" between frames.

(Hsu and Harashima 1994) uses the collections of discontinuities to detect scene cuts in pixel domain. Characterization of activities is carried on by using statistical features such as Gaussian and mean curvature of spatio-temporal video elements.

(Shahraray 1995) uses the motion information to detect scene changes by employing non-uniform content based temporal filtering approach. The block motion vectors are calculated, and a non-linear digital order statistical filter is used to combine block match values.

In (Gunsel and Tekalp 1998), scene change problem is treated as a two-class classification problem, and automatic threshold is calculated and similar video elements are further characterized in one step. The statistical features used to define automatic threshold are also defined. These metrics are available to be used in both uncompressed and compressed domain.

The above methods use pixel data to detect the scene cuts. This introduces the complexity problem, because the compressed data needs to be fully decompressed into pixel domain prior to the detection step. Therefore, the scene change methods are carried on using statistical behavior of compressed domain feature elements.

Particular scene change type detections are further investigated. For example (Fernando and Canagarajah 2000) examines fade-in and fade-out detection in

uncompressed domain. Again, histograms are used to combine the nature of fading with the distribution of the differences.

For the image retrieval case, (Ruey-Feng et al. 2000) discusses the retrieval scenario with a direct feature extraction for JPEG (Wallace 1992) compressed images. This paper gives clues about using compressed data to detect scene changes.

The algorithms that are used in compressed video domain can be classified according to the calculated statistical data. The data extracted from the compressed domain can be discrete cosine transform coefficients, (DC or AC components of macro-blocks) (Yeo and Liu 1995b) (Lee et al. 2000) (Shen and Delp 1995), DCT coefficients and motion vectors in combination (Mezaris et al. 2004) and, macro-block type information(Pei et al. 1999).

(Gerek and Altunbasak 1997) combines various methods carried on MPEG 2 compressed domain to detect scene cuts, detect zoom and pan, and gradual change detection.

(Yeo and Liu 1995b) shows certain feature extraction schemes for shot boundary detection in MPEG compressed video domain and the comparison tests to achieve shot boundary.

In this thesis, the scene cut detection is realized as an application to summarize H264/AVC (ITU-T REC H264 2003) compressed video data. H264 ISO/IEC 144496-10 standard's syntax and semantics, parsing, decoding process is analyzed. This information is used in order to extract direct information from H264/AVC compressed video sequence.

H264 ISO/IEC 144496-10 is the emerging standard scalable for wide area of applications such as streaming over internet, DVD, High-Definition TV, Video on demand etc. It is employed widely in applications ranging from television broadcast to video for mobile devices.

As a practical example, HD channels are in broadcast in the 1080i or 720p formats in SES Astra and Eurobird 1 satellites at 28.2E and 28.5E, using new DVB standard, DVB-S2. (WEB_1 2006) (WEB_2 2006)

The above discussions are just few examples of the emerging H264/AVC standard's practical usage area. This builds the motivation of selecting the H264/AVC coded video bit-stream under study.

The information extracted from H264/AVC coded the bit-stream is in forms of macro-block type of information. This information provides a statistical nature to

automatically detect various events on video data, such as abrupt scene changes, gradual scene changes.

The encoder's decisions during encoding given video sequence, directly reflects on the compressed bit-stream in terms of syntax elements, and this information is collected to gather quantitative information for the decision process. The partial decoding procedure in compressed domain reduces the computational cost.

The deliverables of this study are tools for video summarization. Examples of video summarization can be found in (Li et al. 2006). The shot information can give the user the overview of the scene for a quick understanding. This summary sequence provides users an overview of the entire video.

The output of this study is targeted for H264/AVC coded video skimming applications. The purpose is to provide low-level features of scene cut information, motion information of the scene. Video skimming is the process of representing the original video in the form of a short video clip. A video skimming system is illustrated in Figure 1.1. A typical video skimming is formed of 3 layers. In the first layer, low level features of shot information are derived. At layer 2, high level semantic information is derived, by means of face detection, audio classification, video text detection, event detection etc. The third layer is responsible for assembling the user controlled clips, and represents final video abstraction. This study can be considered in Layer 1 of above discussion, where the input video is H264/AVC coded bit-stream, and output is scene cut information. Audio, and text recognition is out of the scope of this study.

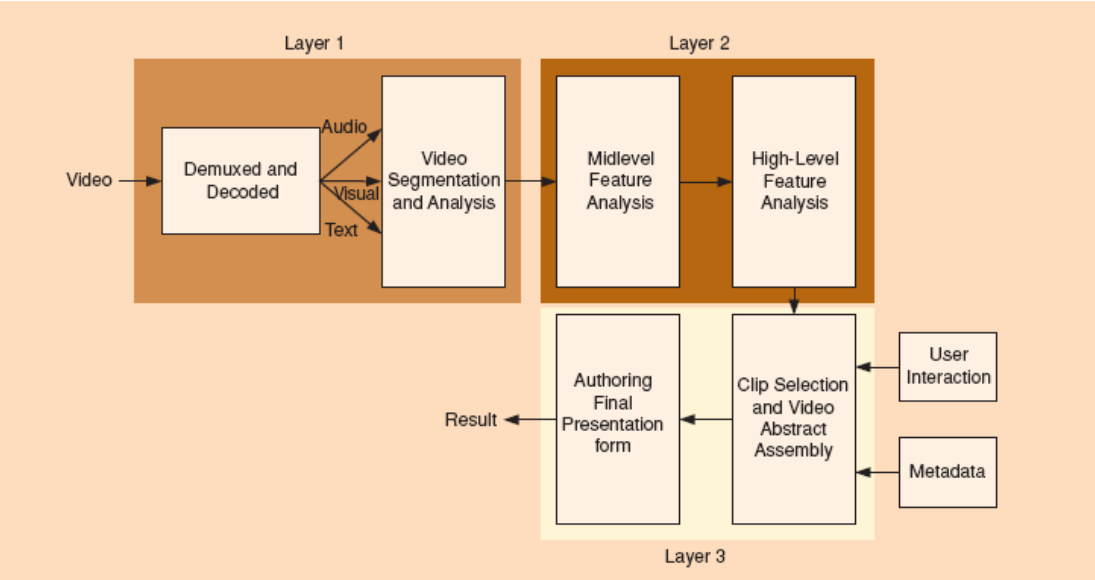

Figure 1.1. Video skimming process
(Source: Li et al. 2006)

1.2. Structure of the Thesis

This study is organized as follows. In chapter 2, the digital representation of video and video compression basics will be described. This will combine the principles and concepts underlying the hybrid DCT based video codecs. After this discussion, the standardization efforts of international organizations will be described. This chapter will form the basis of the video codec H264 ISO/IEC 144496-10 which will follow in chapter 3. The background information given in the first three chapters is followed by Chapter 4 which includes a discussion of compressed domain processing. In Chapter 5 the scene change detection methods will be discussed with an emphasis on video summarization. The proposed work will deeply use the knowledge and researches discussed in these chapters. The proposed work, the tools, algorithms and experimental results with the future work will be the topic of Chapter 6. The properties of spatial and temporal models will be described. The syntax and semantics of the H264 ISO/IEC 144496-10 bit-stream, the encoder internals forms the basis of the scene cut methods.

CHAPTER 2

BASICS OF VIDEO CODING AND STANDARDS

2.1. Introduction

Video coding is the process of compressing a video sequence into smaller number of the bits (Richardson 2003b). Prior to the discussion of fundamentals of video coding principles that will form the background in understanding the hybrid block based video codec internals, and their applied versions defined in the standards, the codec model and the information theoretic definitions are made.. This chapter first discusses temporal and spatial operations that are applied to enable compression. Then the concept of entropy coding is described, and finally the hybrid block based video codec is constructed.

2.2. Encoder/Decoder Model

Encoder works as a compressor to form a bit-stream in the defined syntax prior to storage or transmission, and the decoder decompresses the bit-stream back into the representation domain, typically in pixels. This pair is generally called *CODEC*. (Richardson 2003a)

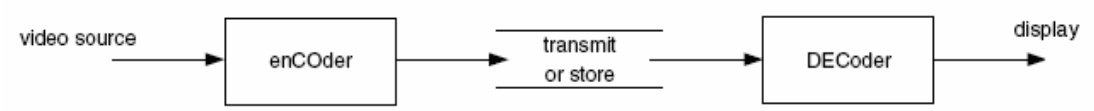

Figure 2.1. Encoder-Decoder Model

2.3. Video Coding Principles

This section will briefly explain the fundamental video coding principles. First, a short review on probability and information theory will be given in the following 3 sub-sections. This discussion will explain the entropy definition and its properties as a background for the bounds provided by the information theory.

In following section 2.4, video codec internals are described. This description is based on spatio-temporal codec model, with an emphasis on entropy coding.

2.3.1. Probability and Information Theory for Video Coding

In video coding, or in any source coding model, the signal is treated as a realization of a random process. Random process and source can be used interchangeably.

The source is a random sequence. Let $\mathbf{F} = \{F_n\}$ be sequence of random variables F_n of n-th sample. f_n is the actual value that F_n takes in. F_n can be multi-dimensional vector or scalar. If F_n takes only symbols from a finite alphabet, $A = \{a_1, a_2, ..., a_n\}$, then F_n is called a discrete random variable (RV). If F_n takes values in continuous range the source is a continuous RV.

For the case of motion video, where sequences are in three dimensions, spatial domain i.e., image points x, y and temporal time domain t; F_n is a $3D$ random vector, and if the corresponding bit-depth is 8-bits, digital video sequence is a discrete source with an alphabet of 256^3. For the case of analog video, F_n is a continuous amplitude random process.

It should be noted that F_n is a stationary process, where F_n does not depend on the index variable n, and joint distribution of a group of N samples is invariant with respect to a common shift in the index.

For the discrete case $p_{F_n}(f)$ represents the probability mass function, and $p_{F_{n+1}, F_{n+2}, ..., F_{n+N}}(f_1, f_2, ..., f_N)$ is used for joint probability mass function (Wang et al. 2002)

2.3.2. Entropy and Mutual Information for Discrete Sources

Entropy and mutual information are two important notions in information theory. They are used in image/video compression to describe the bounds on the minimal bit-rates.

The Entropy of a discrete random variable F with an alphabet A and discrete probability mass function $p_{F_n}(f)$ is defined as follows;

$$H(f) = -\sum_{f \in A} p_F(f) \log_2[p_{F_n}(f)] \tag{1}$$

$$0 < p_{F_n}(f) < 1 \tag{2}$$

Logarithm can be at any base. Since a digital system is under discussion, base-2 case is considered.

Entropy of a random sequence is always non-zero because of (2).

Entropy defines a measure to represent uncertainty for the sequence F_n. It depends directly on the probabilistic model. In the case where F_n can take any value in alphabet with equal probability, it has the maximum entropy. In contrast, if F_n takes a unique symbol with a probability of 1, it has no uncertainty, so it has zero entropy.

Thus this information is the measure to define minimum number of bits to convey random sequence F_n. (Wang et al. 2002)

2.3.3. Bounds for Lossless and Lossy Coding

The theoretical foundations on information theory established bounds on minimal bit-rate required to realize lossless and lossy coding.

Scalar lossless coding refers to assigning binary codeword c_n for each sample f_n to realize sequence $\{f_n\}$ of a discrete source F_n. This requires pre-designed code-book $C = \{c(a_1), c(a_2), ..., c(a_L)\}$. Here $c(a_i)$ is codeword for symbol a_i. Due to this mapping, code word for f_n is $c_n = c(a_i)$. The coded sequence is decodable if this

mapping is one-to-one, i.e; sequence of code words corresponds to one and only one possible sequence of source symbols. Let $l(a_i)$ to be the length of the bits (i.e., the number of bits). Thus the bit rate is defined as;

$$R = \sum_{a_i \in A} p(a_i) l(a_i) \tag{3}$$

The minimum bit-rate required to represent a discrete stationary source F by assigning a codeword to each sample satisfies

$$H_1(F) \le \bar{R}_1(F) \le H_1(F) + 1 \tag{4}$$

The lower bound can be achieved when the pmf of source is a power of two. That is, there exists a set of integers $\{m_1, m_2, ..., m_L\}$ such that $p(a_i) = 2^{-m_i}$. In this case it can be noted that; $l(a_i) = -\log_2 p(a_i) = m_i$.

The above theorem describes the first order entropy of discrete source $H_1(F)$ determines the range of minimum bit-rate for scalar coding source.

In lossy coding case, the N dimensional vector $\{f_1, f_2, ..., f_N\}$ representing the original source F is mapped to the quantized vector $\mathbf{g} = Q(\mathbf{f})$. The vector g must belong to a pre-designed reconstruction code book of a finite size $L, C = \{g_1, g_2, ..., g_L\}$.

Using fixed bit length coding, each quantized vector is represented by $\log_2(L)$ bits. So the bit-rate of the coded sequence is; (Wang et al. 2002).

$$R_N = \frac{1}{N} \log_2(L) \tag{5}$$

2.4. The Video CODEC Internals

The data compression is done using the statistical dependency of the video components in spatial and temporal domains. There are three fundamental redundancy principles: (Richardson 2003a)

Spatial Redundancy: Among the pixels within picture.

Temporal redundancy: Among successive differences

Entropy coding: Redundancy in compressed data symbols, generally using variable length coding techniques. (Ghanbari 2003)

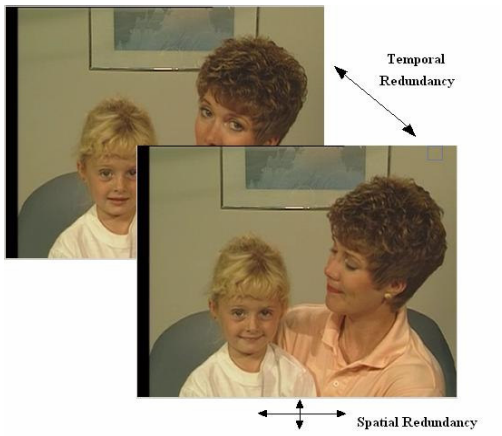

Figure 2.2. Mother Daughter Sequence Showing Spatial and Temporal Redundancies

As it can be seen from the above image, there is a high correlation between the successive frames, or in temporal domain. This correlation is relatively higher if the temporal sampling rate is high. (Richardson 2003a).

Input to encoder is the uncompressed sequence of raw data, as described in video basics section. The video codec implements temporal model for temporal redundancies, and spatial model for spatial redundancies. The core objective of the model is to supply bit-rate efficiency with an acceptable quality.

As described at the beginning of the section, three models exist for full compression: Temporal, spatial and entropy coder. The blocks input output relation is readily seen below.

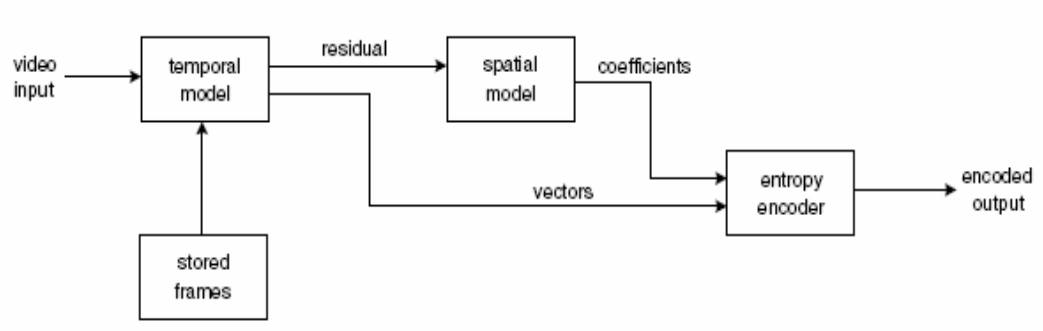

Figure 2.3. Encoder Blocks
(Source: Richardson 2003a)

2.4.1. Temporal Model

The core objective of the temporal model is to reduce temporal redundancy between transmitted frames by forming a predicted frame and making a subtraction from the current frame. The output of this process is residual data represented with the possible amount of energy. The residual is encoded and sent to decoder. Decoder re-creates this prediction frame, adds the decoded residual and constructs the current frame.

It is important to model the motion that is observed through consecutive images to represent the temporal sequence efficiently. Motion estimation and compensation methods are used for this purpose.

2.4.1.1. Motion Model

A fundamental tool for prediction in temporal domain is using the previous frame as a predictor for the current frame. In Figure 2.4, the left frame is frame numbered 1, where the frame in the middle is frame 2. Frame 2 is to be predicted from Frame 1. The residual is formed by direct subtraction. Difference between the pixel values can be absorbed in residual image. It can be noted that, there is considerable amount of energy.

Figure 2.4. The Difference of Two Consecutive Frames

The sources of the changes between the frames in a sequence of picture are object motion in the image, camera motion (zoom, panning, and rotation) and lighting changes. In the first two cases the cause is pixel movements in the image. The overall pixel movements in the image can be globally described by an optical flow field, which is a set of displacement vectors.

Practically, it is costly to predict the movement of each pixel. This problem can be alleviated by considering the movements of the blocks of size 8x8 or 16x16 instead of pixels.

2.4.1.2. Block Based Motion Estimation and Compensation

Practically, instead of predicting each pixel movement, image is partitioned into blocks. To achieve motion estimation and compensation this widely used procedure is used:

1. Search an area in the reference frame (past or future frame, previously coded and transmitted) to find a 'matching' $M \times N$-sample region. This is carried out by comparing the $M \times N$ block in the current frame with some or all of the possible $M \times N$ regions in the search area (usually a small region centred on the current block position) and finding the position that gives the 'best' match. A popular matching criterion is based on minimizing the energy in the residual formed by subtracting the candidate region from the current $M \times N$ block. The candidate position that minimises the residual energy is chosen as the best match. This process of finding the best match is known as *motion estimation*.

2. The chosen block at the candidate position becomes the predictor for the current $M \times N$ block and is subtracted from the current block to form a residual $M \times N$ block i.e., *motion compensation*.

3. The residual block is encoded and transmitted and the offset between the current block and the position of the candidate region (*motion vector*) is also transmitted (Richardson 2003a).

The information derived from the above discussion is used by the decoder to re-create the predicted region, after decoding the residual frame the predicted region is added and the original frame is constructed.

Using block based motion estimation provides an efficient form for practical implementations, and traceability but it introduces blocking artifacts.

The arrangement of a macroblock can be seen below. A macroblock has 16x16 pixel Y-luminance blocks, and 8x8 blocks for chrominance samples, if the sampling scheme is 4:2:0.

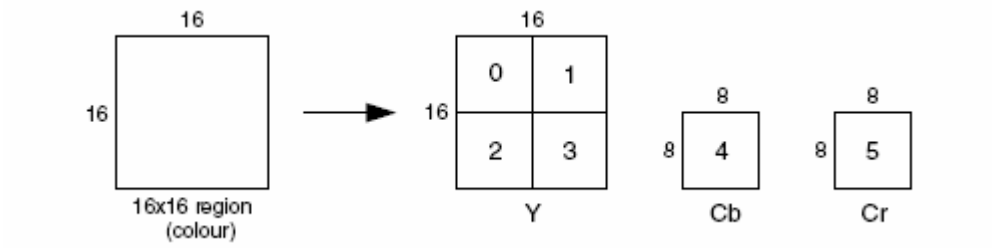

Figure 2.5. Macro-block Structure, 4:2:0 Sampling Lattice

As described earlier, finding the best match of replacement of a macro-block in reference frame is motion-estimation. The reference frame can be previous or next frame, regardless of display order.

The best-match macro-block is subtracted from the reference frame to produce a residual macro-block. This residual macro-block is encoded with an appropriate motion-vector assigned to it. In the encoder, there is a reverse path to decode this residual information with associated motion vector to produce a reference frame. This also ensures the reference pictures are valid for both decoder side, and the encoder side.

An also important trade off in the motion estimation and compensation is between block-size and complexity. Smaller block sizes provide better motion compensation, because it approximates the ideal condition of pixel-wise optical flow in a more precise level.

Another important opportunity to enhance motion estimation is using the sub-pixel positions for mid-level integer values by using interpolation. This provides a more precise best match search operation, and better energy compaction.

The following Figure 2.6 shows the concept of “quarter-pixel” motion estimation is illustrated in Figure 2.6.

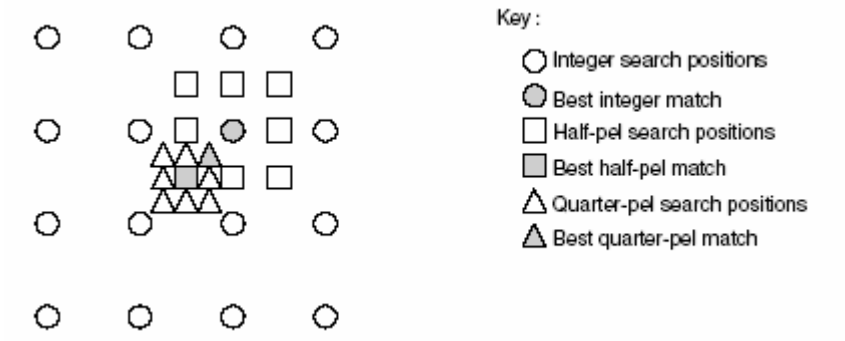

Figure 2.6. Integer, Half-Pel, Quarter-Pel Motion Estimation
(Source: Richardson 2003a)

In the first stage, motion estimation finds the best match on the integer sample grid (circles). The encoder searches the half-sample positions immediately next to this best match (squares) to see whether the match can be improved and if required, the quarter-sample positions next to the best half-sample position (triangles) are then searched. The final match (at an integer, half- or quarter-sample position) is subtracted from the current block or macroblock.

2.4.2. Spatial Model

As discussed earlier, video sequence is formed of images. There is a considerable redundancy in spatial domain. It is not efficient to compress a still image using a simple entropy coding scheme, because of the natural behavior of the still images. If we consider the auto-correlation function of a typical 2D image, we can observe a high spatial correlation. Note that; autocorrelation indicates the similarity between the original image and a spatially-shifted copy of itself. (Richardson 2003a). The autocorrelation function corresponding to a still image can be seen in Figure 2.7.

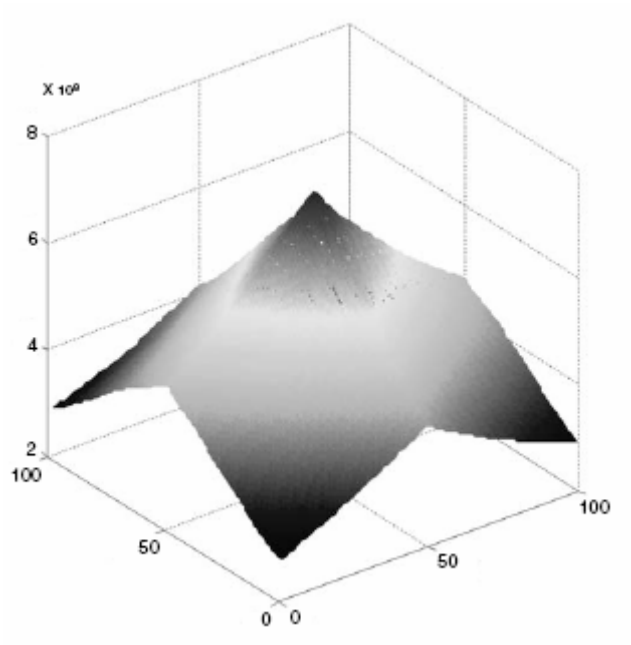

Figure 2.7. Autocorrelation function of a still image
(Source: Richardson 2003a)

As noted earlier, the motion compensated residual compacts the energy of the motion, therefore the correlation between the pixels in residual data is weaker. This effect can be seen in Figure 2.8.

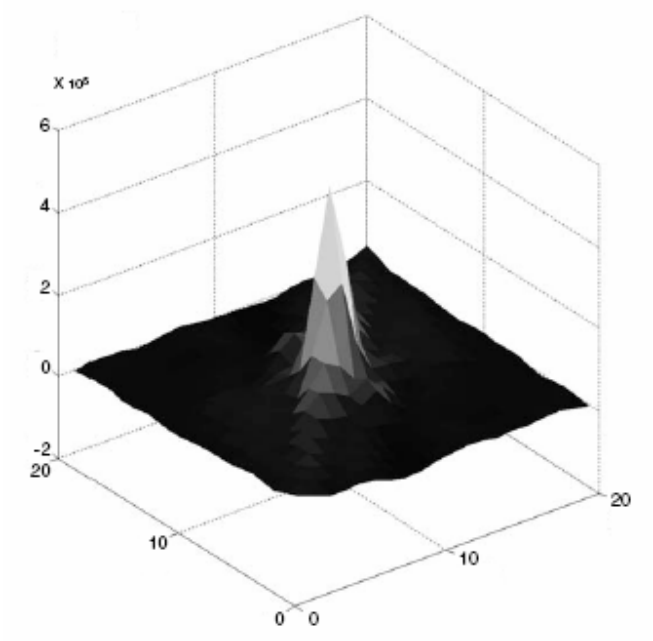

Figure 2.8. Autocorrelation Function of a Residual
(Source: Richardson 2003a)

The function of image model is to de-correlate the residual image further to so that the entropy coder can further compress this data. To achieve such objective, an image model has three blocks; transform coder, quantizer, and re-order mechanism.

2.4.2.1. Predictive Image Coding

Predictive image coding in spatial domain is similar to motion compensation in temporal domain. In predictive image coding, the previously sent samples may be used to predict the current picture element in the same image. This prediction method is generally called Differential Pulse Code Modulation.

2.4.2.2. Transform Coding

The main objective of transform coding stage in a video encoder is to transform image pixel data or residual data to another domain. The main features of the transform should be:

1. Data in the transform domain should be decorrelated (separated into components with minimal inter-dependence) and compact (most of the energy in the transformed data should be concentrated into a small number of values).
2. The transform should be reversible.
3. The transform should be computationally tractable (low memory requirement, achievable using limited-precision arithmetic, low number of arithmetic operations, etc.).

In most of the still scenes, the energy is concentrated on low frequency region. In order to achieve higher compression rates, transform coding transforms the spatial values into transform domain.

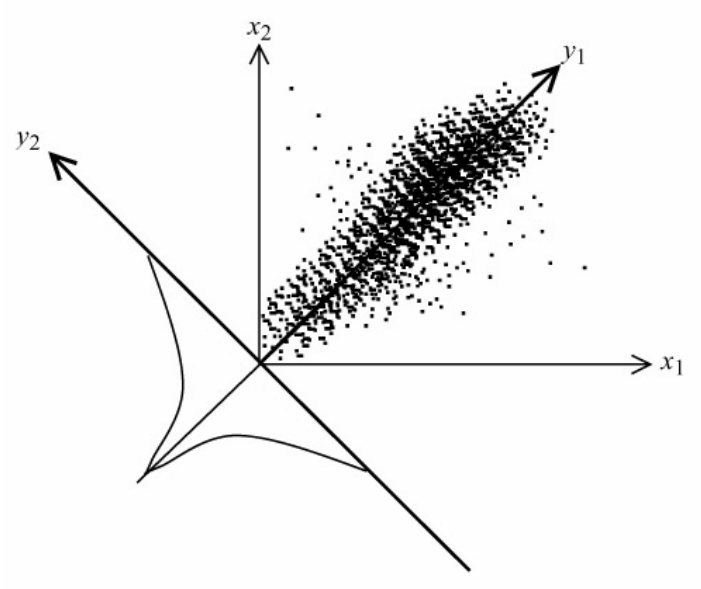

Figure 2.9. The Effect of Transforming
(Source: Ghanbari 2003)

In Figure 2.9, x_1 and x_2 has the pixel number distribution in x_1x_2 Cartesian space. Although pixel values can take values between 0 and 255 when quantized to 8 bits, they are likely to take values in terms of their similarity. In the above figure, their distribution lies on top of the 45 degrees line. If the x_1x_2 axis is rotated by 45 degrees to y_1y_2 axis, the quantity to represent the same data is getting smaller.

The transformation matrix T, is rotates the axis by 45 degrees.

$$T = \begin{bmatrix} \cos 45 & \sin 45 \\ \sin 45 & -\cos 45 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

This makes the resulting axis, transform coefficients y_1, y_2 are:

$$y_1 = \frac{1}{\sqrt{2}}(x_1 + x_2), \text{ and } y_2 = \frac{1}{\sqrt{2}}(x_1 - x_2)$$

According to the Perseval's theorem (Oppenheim and Schafer 1989); the signal energy is preserved due to the transformation. This statement holds for the above

transform pair, where $\frac{1}{\sqrt{2}}$ is the normalization factor. The signal energy in pixel domain is equal to the signal energy in transform domain.

Elements of the transform matrix are called 'basis vectors'. They are used to de-correlate level of transform coefficients, leaving part of coefficients significant, and making the remaining parts small in quantity.

2.4.2.3. DCT Transform

Discrete cosine transform, first designed for the area of digital processing for the purposes of pattern recognition and Wiener filtering (Oppenheim and Schafer 1989).

DCT is defined by:

$$u_{k,n} = \alpha(k)\cos\frac{(2n+1)k\pi}{2N}, n = 0,1,...,N-1 \quad (1)$$

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{N}} & k = 0 \\ \frac{1}{\sqrt{N}} & \text{k=1,2,...,N-1} \end{cases} \quad (2)$$

2D DCT Transform operates on a block of $N \times N$ samples to create a matrix $\mathbf{Y}$. If $\mathbf{X}$ is the matrix with the samples, $\mathbf{Y}$ is the coefficients matrix and $\mathbf{A}$ is $N \times N$ transform matrix, the forward, backward transforms and elements of $\mathbf{A}$ is defined as follows: (Drake et al. 1967)

Forward Transform: $\mathbf{Y = AXA^T}$

Inverse Transform: $\mathbf{X = A^T YA}$

$\mathbf{A}_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N}$ where $C_i = \sqrt{\frac{1}{N}}(i=0)$, $C_i = \sqrt{\frac{2}{N}}(i>0)$

Recall that, basis functions are real, which makes DCT relatively easy to implement.

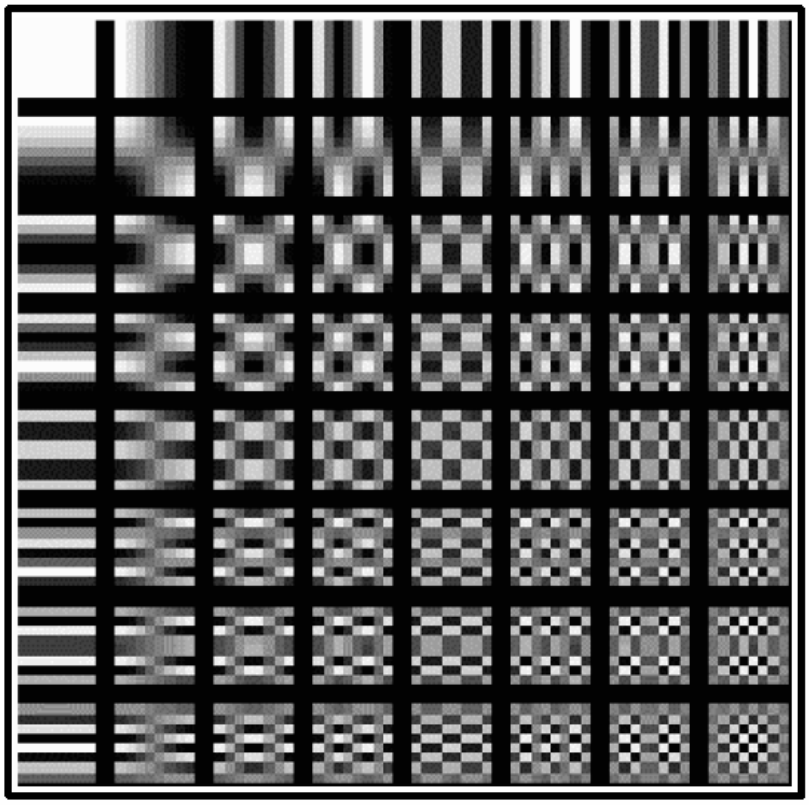

Figure 2.10. DCT Basis Functions Images
(Source: Richardson 2003a)

In the figure below, DCT coefficients of a 4×4 block of an image is shown.

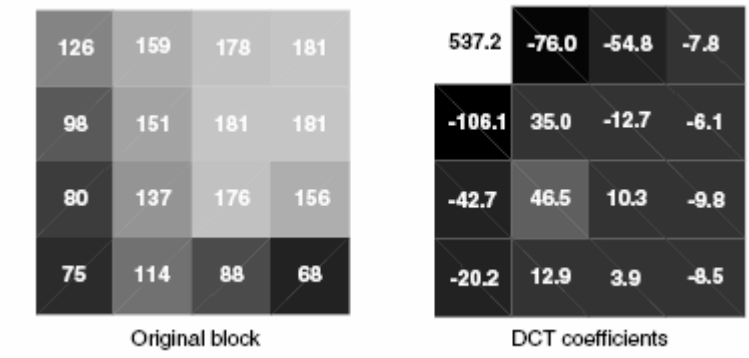

Figure 2.11. DCT Example

2.4.2.4. Quantization

Quantization technique is a mapping from X to a quantized signal with a reduced range of Y. The quantized signal Y has fewer bits to represent X. A *scalar quantiser* maps one sample of the input signal to one quantised output value and a *vector quantiser* maps a group of input samples (a 'vector') to a group of quantised values.

Such mapping is seen on uniform threshold quantiser is seen below. It has equal step sizes with reconstruction values pegged to the centriod of the steps (Ghanbari 2003).

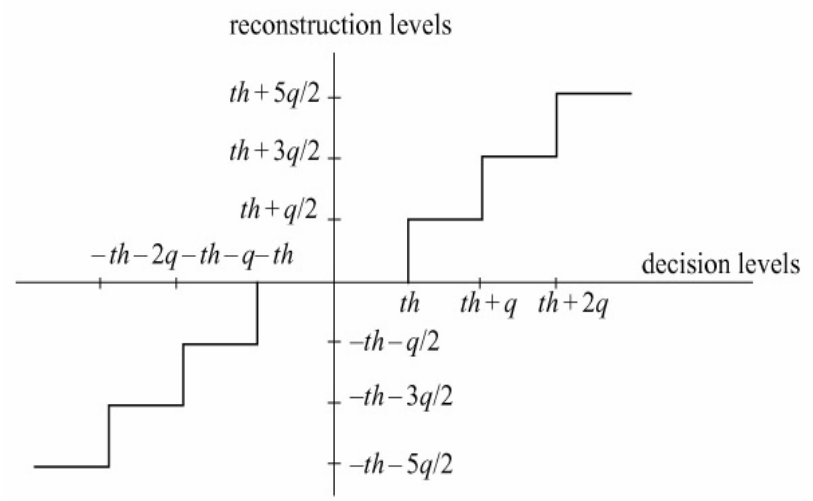

Figure 2.12. Uniform Threshold Quantiser (UTQ) threshold value th and the step size q (Source: Ghanbari 2003)

A further two subclasses of UTQ can be identified within the standard codecs, namely those with and without a dead zone. These are illustrated in Figure 2.13 and will be hereafter abbreviated as UTQ-DZ and UTQ, respectively. The term dead zone commonly refers to the central region of the quantiser, whereby the coefficients are quantised to zero. (Ghanbari 2003)

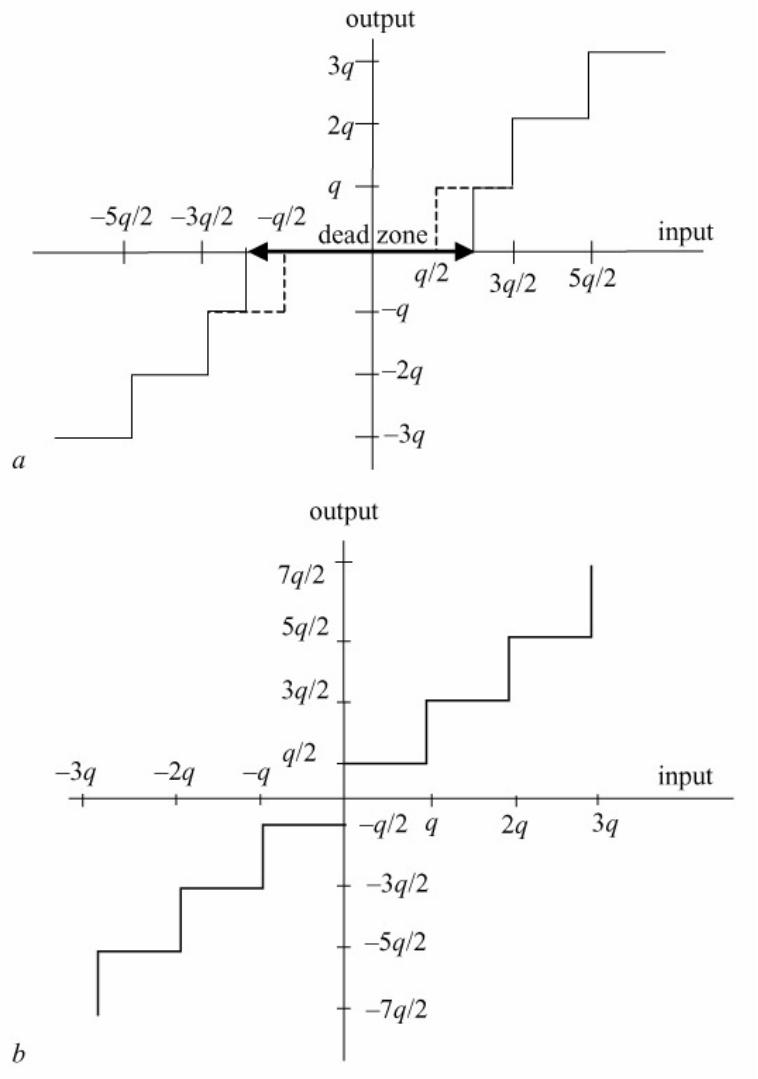

Figure 2.13. Quantization Schemes
(Source: Ghanbari 2003)

The quantization resolution is defined by quantiser parameter QP. If the step size is large, the number of output values if quantiser is small, so a better compression can be achieved. However due to the coarse quantization, loss in the image quality decreases.

The inverse operation of quantization is inverse quantization. Inverse quantization is frequently referred as “scaler” or “rescaler”.

Quantisation may be used to reduce the precision of image data after applying a transform such as the DCT or wavelet transform removing insignificant values such as near-zero DCT or wavelet coefficients. The forward quantiser in an image or video encoder is designed to map insignificant coefficient values to zero whilst retaining a

reduced number of significant, nonzero coefficients. The output of a forward quantiser is typically a 'sparse' array of quantised coefficients, mainly containing zeros.

Another type of quantization is vector quantization, where quantiser maps a vector of inputs to a single value (codeword), at the decoder side, the codeword maps to approximations of original input. The codebook contains the necessary collection of codewords in both decoder and encoder.

Vector quantization works as follows:

1. Partition the original image into regions (e.g. $M \times N$ pixel blocks).

2. Choose a vector from the codebook that matches the current region as closely as possible.

3. Transmit an index that identifies the chosen vector to the decoder.

4. At the decoder, reconstruct an approximate copy of the region using the selected vector. (See following figure)

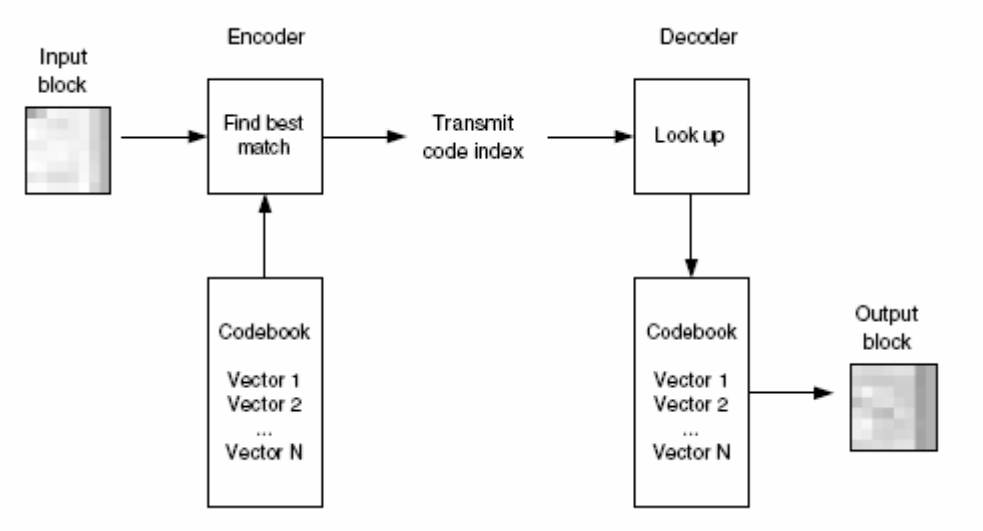

Figure 2.14. Vector Quantization Technique
(Source: Ghanbari 2003)

A critical part of designing a successful scheme is to design codebook for appropriate codeword mapping.

2.4.2.5. Reordering and Zero Encoding

After transform, and quantization step, the coefficients are to be further processed before being transmitted. The output of transform and quantization includes many zeros, and has a sparse structure.

Reordering is the process of preparing the data to entropy coding. The distribution of the transform coding will be described below.

2.4.2.5.1. DCT Coefficient Distribution and Run-Level Encoding

The DC coefficients represent the zero frequency information of the corresponding macro-block and it is located at DC(0,0) of the macro-block. The distribution of the non-zero coefficients is seen in the below figure 14**.**

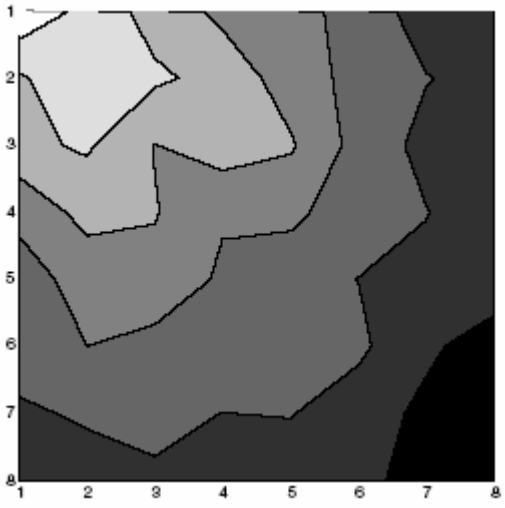

Figure 2.15. The Distribution of a Residual Macro-blocks (Non zero Coefficients in lighter gray) (Source: Richardson 2003a)

2.4.2.5.2. Run-Level Encoding

The output of the reordering process is an array that typically contains one or more clusters of nonzero coefficients near the start, followed by strings of zero coefficients. The large number of zero values may be encoded to represent them more compactly, for example by representing the array as a series of (run, level) pairs where

run indicates the number of zeros preceding a nonzero coefficient and *level* indicates the magnitude of the nonzero coefficient.

2.4.2.6. Entropy Coding

Entropy coder is the latest step in encoder block. Its output is the compressed bit-stream, ready to send or store in an appropriate format, which is not the topic discussed in this section. Inputs can be of type motion vectors, quantized transform coefficients, markers, headers for data in the stream, supplementary information for robustness for streaming etc.

2.4.2.6.1. Predictive Coding

Whether spatial or temporal data is input to the entropy coder, redundancy can still exist a in the bit-stream to be sent. Especially small macro-block replacement vectors are temporally correlated. Compression of the motion vector field may be improved by predicting each motion vector from previously-encoded vectors. (Richardson 2003a)

2.4.2.6.2. Variable-length Coding

A variable-length encoder maps input symbols to a series of codewords (variable length codes or VLCs). Each symbol maps to a codeword and codewords may have varying length but must each contain an integral number of bits. Frequently-occurring symbols are represented with short VLCs whilst less common symbols are represented with long VLCs. Over a sufficiently large number of encoded symbols this leads to compression of the data.

2.4.2.6.3. Huffman Coding

Huffman coding assigns a VLC to each symbol based on the probability of occurrence of different symbols.

The optimal number of bits to be used for each symbol is - $\log_2 p$, where p is the probability of a given symbol.

To generate the Huffman code for symbols with a known probability of occurrence, the following steps are carried out:

- Rank all the symbols in the order of their probability of occurrence
- Successively merge every two symbols with the least probability to form a new composite symbol, and re-rank order them; this will generate a tree, where each node is the probability of all nodes beneath it
- Trace a path to each leaf, noting the direction at each node.

An example can be seen below, where symbols are from A-G, and their probability is in descending order in the third column. The smallest probabilities are coded and combined probability makes the column reorder. The process goes to end of columns to a last probability of 1. Starting from the last column, the bottom probabilities are assigned 1, and top ones are assigned to 0, the corresponding codeword (shown in the first column) is read off by following the sequence from right to left.

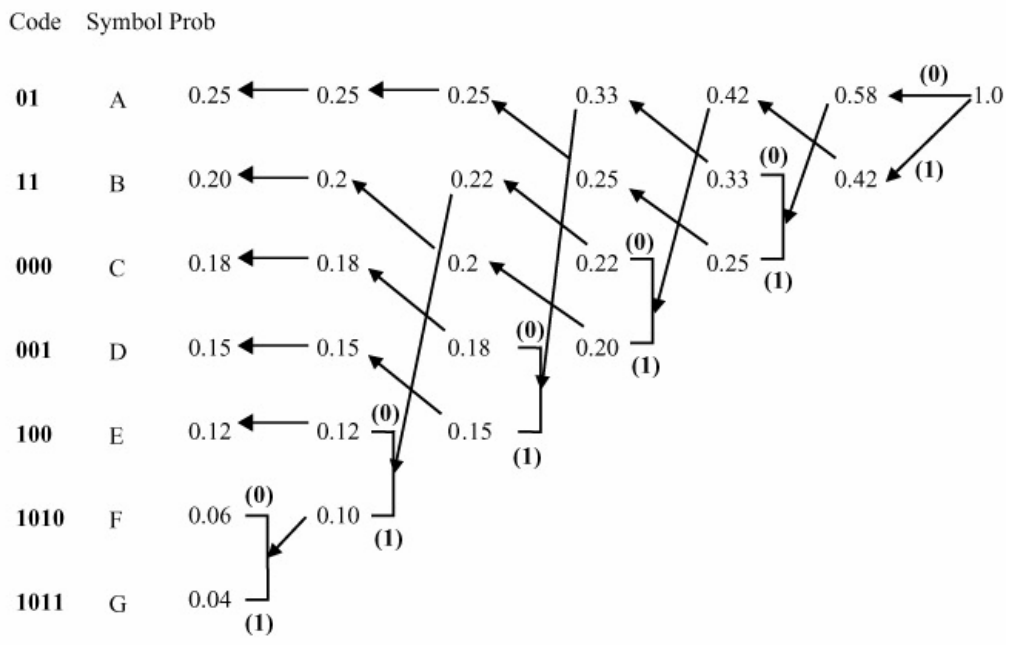

Figure 2.16. Example of Huffman Code for 7 Symbols
(Source: Ghanbari 2003)

2.4.2.6.4. Arithmetic Coding

The variable length coding schemes described in previous section share the fundamental disadvantage that assigning a codeword containing an integral number of bits to each symbol is sub-optimal, since the optimal number of bits for a symbol depends on the information content and is usually a fractional number. Compression efficiency of variable length codes is particularly poor for symbols with probabilities greater than 0.5 as the best that can be achieved is to represent these symbols with a single-bit code. (Richardson 2003a)

In arithmetic coding, a code string is created, such that this string represents a fractional value on the number line in interval [0,1]. It separates the data and encoding information with respect to the model. Each symbol is translated to integral number of bits, thereby improving coding efficiency. It catches the entropy bound effectively. (Ghanbari 2003)

Modeling forms the basis of arithmetic coding. It can be defined as the action of calculating the distribution of probabilities for the next symbol to be coded in any given concept. In models, the probabilities are represented in integer frequencies. There are two types of arithmetic coding; fixed and adaptive.

In the fixed model, both encoder and decoder know the probability assigned to each symbol. These probabilities can be determined by measuring frequencies in representative samples to be coded and the symbol frequencies remain fixed. Fixed models are effective when the characteristics of the data source are close to the model and have little fluctuation.

In the adaptive model, the assigned probabilities may change as each symbol is coded, based on the symbol frequencies seen so far. Each symbol is treated as an individual unit and hence there is no need for a representative sample of text. Initially, all the counts might be the same, but they update, as each symbol is seen, to approximate the observed frequencies. The model updates the inherent distribution so the prediction of the next symbol should be close to the real distribution mean, making the path from the symbol to the root shorter.

For example, suppose the symbol sequence of "**eaii!**" from the symbol elements {a, e, i, o, u, ! } is to be encoded.

In table below, each symbol, its probability of occurrence and range that corresponds to its probability of appearance in the cumulative density function is noted.

Table 2.1. Example Table of Prob. For Different Symbols

Model for alphabet a, e, i, o, u, !		
Symbol	**Probability**	**Range**
a	0.2	[0.0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

Arithmetic coding process starts with the first symbol “e” (in the sequence eaii!). It is in the range [0.2, 0.5). The next coming symbol “a” has a range [0.0, 0.2) but the new sub range is the symbol restricted by the range of “e”. The process can be seen in the below figure. The final range represents the message. Any number x, which falls into the range of $0.23354 \leq x < 0.2336$, represents the sequence “**eaii! “.**

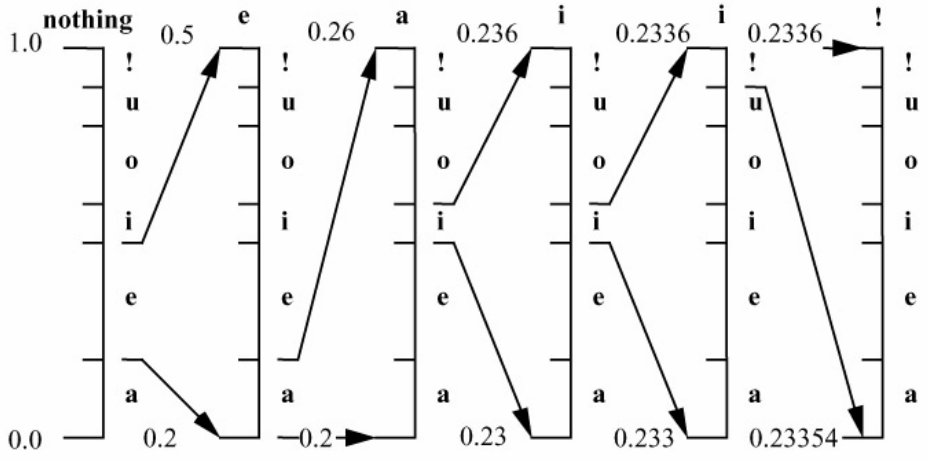

Figure 2.17. Process of Arithmetic Coding Interval. Scaled up at each stage for the message eaii! (Source: Ghanbari 2003)

Generally decoding can be done using $R_{n+1} = \frac{R_n - L_n}{U_n - L_n}$, where R_n is the code in the range defined by lower value L_n, and upper value U_n (Ghanbari 2003).

2.4.2.6.5. Binary Arithmetic Coding

The common pitfall of arithmetic coding described above is observed on the long sequences, because as the length of the sequence increases, the interval where the encoded signal is presented is getting smaller. This leads to a need for technique to define an computer arithmetic friendly upper bound, i.e., the interval [0, 1) is scaled up to [0, *MAX_VAL*] where MAX_VAL is the largest integer that computer can handle.

More complex examples can be seen on (Ghanbari 2003) and (Richardson 2003a)

2.4.2.6.6. Context-Based Arithmetic Coding

A popular method for adaptive arithmetic coding is to adapt the assigned probability to a symbol, according to the context of its neighbors. This is called context-based arithmetic coding.

Assume that binary symbols of *a*, *b* and *c,* which may take values of 0 or 1, are the three immediate neighbors of a binary symbol *x,* as shown in Figure 17.

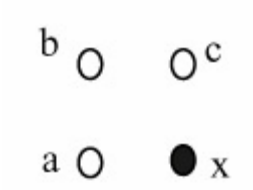

Figure 2.18. Three Neighbors of symbol x

There is a high correlation between the symbols in the image data. If the neighboring symbols "*a, b* and *c"* are mainly 1, then it is logical to assign a high probability for coding symbol *x,* when its value is 1. Conversely, if the neighboring symbols are mainly 0 the assigned probability of $x = 1$ should be reduced. Thus we can define the context for coding a 1 symbol as:

$$context = 2^2 c + 2^1 b + 2^0 a = 4c + 2b + a$$

For the binary values of *a, b* and *c;* the context has a value between 0 and 7. Higher values of the context indicate that a higher probability should be assigned for coding of 1, and a complementary probability, when the value of *x* is 0.

2.5. The Hybrid Block Based Video Codec

This chapter so far described the internal blocks of the hybrid block based video codec. The layout of the encoder which combines these blocks is in Figure 2.19.

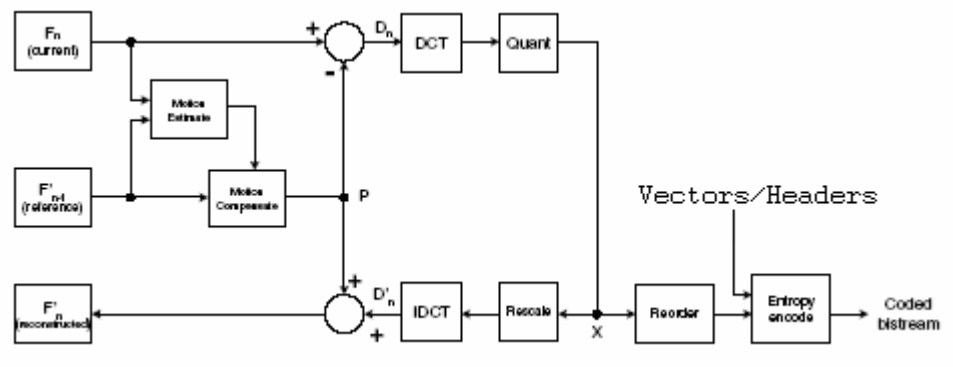

Figure 2.19. Dataflow in encoder
(Richardson 2003a)

The data flow in the encoder can be summarized as below:

1. An input video frame F*n* is presented for encoding and is processed in units of a macroblock (corresponding to a 16 × 16 luma region and associated chroma samples).

2. F_n is compared with a *reference* frame, for example the previous encoded frame (F_{n-1}).

A motion estimation function finds a 16 × 16 region in F_{n-1} (or a sub-sample interpolated version of F_{n-1}) that 'matches' the current macro-block in F*n* (i.e. is similar according to some matching criteria). The offset between the current macroblock position and the chosen reference region is a motion vector MV.

3. Based on the chosen motion vector MV, a motion compensated prediction P is generated (the 16 × 16 region selected by the motion estimator).

4. P is subtracted from the current macroblock to produce a residual or difference macroblock D.

5. D is transformed using the DCT. Typically, D is split into 8 × 8 or 4 × 4 sub-blocks and each sub-block is transformed separately.

6. Each sub-block is quantized.

7. The DCT coefficients of each sub-block are reordered and run-level coded.

8. Finally, the coefficients, motion vector and associated header information for each macroblock are entropy encoded to produce the compressed bitstream (Richardson 2003a).

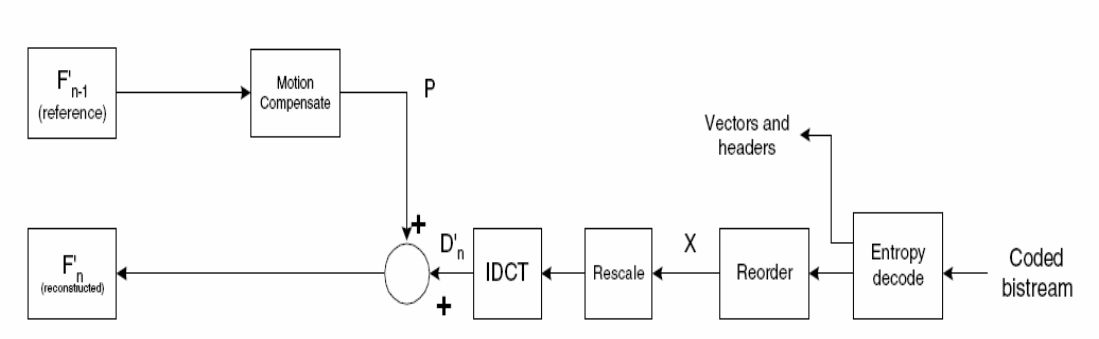

Figure 2.20. Data Flow in the Decoder

The Decoder works in the inverse order of encoder, receiving Coded Bit-stream, as seen in Figure 2.20. The reconstruction process is summarized below.

1. Acompressed bitstream is entropy decoded to extract coefficients, motion vector and header for each macroblock.

2. Run-level coding and reordering are reversed to produce a quantised, transformed macroblock X.

3. X is rescaled and inverse transformed to produce a decoded residual D_n.

4. The decoded motion vector is used to locate a 16 × 16 region in the decoder's copy of the previous (reference) frame F_{n-1}. This region becomes the motion compensated prediction P.

5. P is added to D_n to produce a reconstructed macroblock. The reconstructed macroblocks are saved to produce decoded frame F_n.

CHAPTER 3

OVERVIEW OF H264/AVC STANDARD

3.1. Introduction

H264/AVC standard has been developed to provide two main goals: Compression efficiency and network friendliness. It has been developed by combinational efforts of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). The addressed applications can be grouped to "conversational" applications; such as video conferencing/telephony and "non-conversational" applications; such as storage, broadcast or streaming. The new standard also provides tools to enhance rate-distortion efficiency compared to the existing video coding standards.

This chapter will form the basis of the H264/AVC internals. The discussions here applies directly on next chapters, because the application directly uses the behavior of the encoder's output i.e., predictions that are embedded in the bit-stream.

In Part 2 of this chapter, the purpose and the history of the standards will be briefly mentioned as a background for the new challenges provided by the H264/AVC standard. Then the working model of the standard is described. The design goals of the standard form a motivation for the necessary features of this codec architecture.

Part 2 also gives necessary background on Network Abstraction Layer which defines a format to carry coded video elements. Part 3 illustrates the video coding layer sitting on top of NAL layer. Finally, Part 4 describes the profiles and levels that illustrate the tools of H264/AVC appropriate for application domain.

3.2. History and Overview of the Video Coding Standards

Efficient digital representation of image and video signals was an active research and development area, and resulted many developments in video coding standards. Standards give the opportunity to define interoperability between devices. The applications covered video on demand, digital television, video database systems etc.

Moving Picture Expert Group was formed to develop standards on video coding technology in 1988. The studies collected many efforts on image and video coding algorithms together to meet the needs of interoperability and scalability. Although this was the economical goal, technically the ultimate objective was to lower the bit rates. The first output of the committee was MPEG-1, which was issued in 1992. The standard defined representation of video with associated audio at 1.5 Mbits/s bit rates for storage and retrieval applications. This standard did not cover carrying video information at higher bit rates, e.g. PAL/NTSC resolutions. Also interlaced video handling was another issue not covered in MPEG 1 standard. So in 1994, this demand resulted in MPEG 2 standard, which was available to handle such higher video representation either interlaced or progressive with associated audio. The application areas are extended to packet video networking, digital television over satellite or terrestrial broadcasting techniques, digital tape recorder applications.

MPEG video compression algorithms rely on the collection of techniques in spatial and temporal video domain based on hybrid DCT, as described in Chapter 2. (See Figure 2.19).

MPEG 1, (also MPEG 2) are based on macro-blocks, which is the fundamental unit. Each macro-block consists of four 8x8 luminance blocks and two 8x8 chrominance blocks(1 U and 1 V). Macro-blocks are the units for motion-compensated compression. Blocks are used for DCT compression.

Frames can be encoded in three types: intra-frames (I-frames), forward predicted frames (P-frames), and bi-directional predicted frames (B-frames).

An **I-frame** is encoded as a single image, with no reference to any past or future frames. The encoding scheme used is similar to JPEG compression. A **P-frame** is encoded relative to the past reference frame. A reference frame is a P- or I-frame. The past reference frame is the closest preceding reference frame. Each macro-block in a P-frame can be encoded either as an I-macro-block or as a P-macro-block. An I-macro-block is encoded just like a macro-block in an I-frame. A P-macro-block is encoded as a 16x16 area of the past reference frame. To specify the 16x16 area of the reference frame, a motion vector is included. A **B-frame** is encoded relative to the past reference frame, the future reference frame, or both frames. The future reference frame is the closest following reference frame (I or P). The encoding for B-frames is similar to P-frames, except that motion vectors may refer to areas in the future reference frames. A

typical frame dependency pattern for MPEG 1 can be shown in Figure 3.1. These definitions are also valid for MPEG 2 standard.

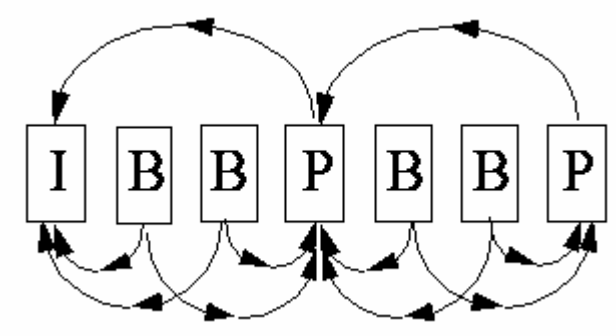

Figure 3.1. Typical MPEG 1 Frame Order
(Source: WEB_7, 2007)

MPEG-1 video sequence is a layered structure. Each video sequence is composed of a series of Groups of Pictures (GOP's). A GOP is composed of a sequence of pictures (frames). A frame is composed of a series of slices. A slice is composed of a series of macro-blocks, and a macro-block is composed of 6 or fewer blocks (4 for luminance and 2 for chrominance) and possibly a motion vector. This structure is also valid for MPEG 2 standard.

MPEG 1 was supporting video sequence (possibly decimated from the original) of about 352 by 240 frames by 30 frames/s. MPEG-1 was optimized for CD-ROM or applications at about 1.5 Mbit/sec. Video non-interlaced (i.e. progressive). MPEG 2 is the superset of MPEG 1. It provides scalability by defining different profiles and levels supporting interlaced video formats and a number of other advanced features, including features to support HDTV. This profile aims to support applications such as compatible terrestrial TV/HDTV, packet-network video systems, backward-compatibility with existing standards (MPEG-1 and H.261), and other applications for which multi-level coding is required. MPEG-2 Video builds on the completed MPEG-1 Video Standard. (WEB_7, 2007) (Tekalp. 1995)

3.3. H264 Video Coding Standard

3.3.1. The Scope of the Standard

The scope of H264/AVC ISO/IEC Part-10 Standard is similar to the purpose of the previous standards. The focus of the standard is on the central decoder part as illustrated in the below figure:

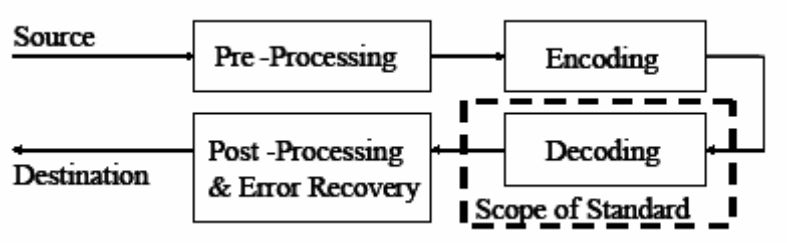

Figure 3.2. The Scope of the standard
(Source: Richardson 2003a)

The standard defines the restrictions on the *bit-stream* and *syntax*, *decoding process, parsing process* of syntax elements as the previous standards are organized. This assures that every decoder that is compliant with the standard will produce similar pixel output from *coded video data*. The purpose of this working model is to bring freedom to those who implements encoder according to the application needs. Note that there is a trade-off in between compression quality, bit-rate, and implementation complexity etc. In short, the responsibility of the encoder is to produce compliant bit-stream output.

3.3.2. Applications and New Features of H264/AVC

H264/AVC standard is designed to meet the requirements of very broad application areas. The most common application field can be listed as the following:

- Broadcast over cable, satellite, cable modem, DSL, terrestrial
- Interactive applications on Storage Media such as optical, magnetic devices etc.
- Conversational services

- Video on demand and multimedia streaming services
- Wireless application scenarios, Multimedia Messaging services, Ethernet.

The standard is capable of supporting network infrastructures, as it is going to be described in the next part. To enable such flexibility, it is necessary to understand the structure of the encoder top level architecture, which is shown on Figure 3.2.

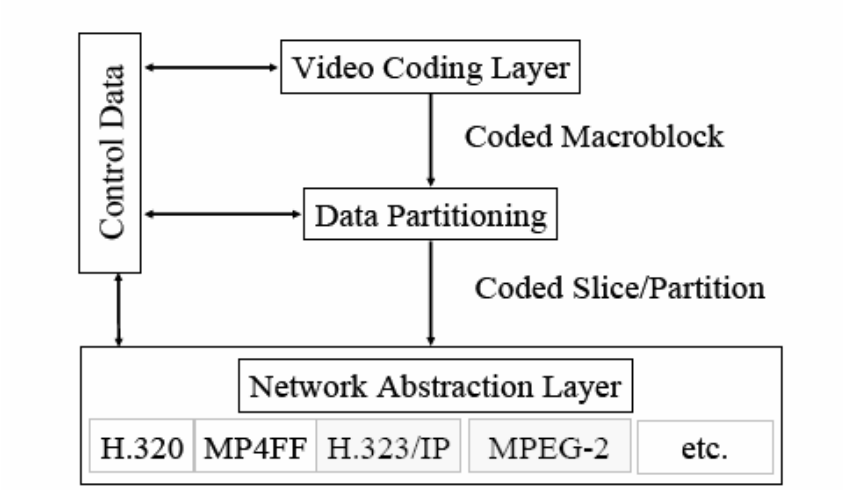

Figure 3.3. The H264 Layers
(Source: Richardson 2003a)

Enhancement in coding efficiency compared to previous standards is accomplished by the prediction algorithms that can be summarized by the following items:

- Variable block-size motion compensation with small sizes
- Motion compensation with quarter sample accuracy
- Motion vectors over picture boundaries.
- Multiple reference picture motion compensation
- Decoupling of reference order from display order
- Decoupling of picture presentation methods from picture referencing
- Weighted prediction
- Improved “skipped” and direct motion reference
- Directional spatial prediction for intra coding.
- In loop de-blocking filter

Other tools improving the coding efficiency increasing methods:

- Small size block transform
- Hierarchical block transform

- Short word length transform
- Exact match inverse transform
- Arithmetic entropy coding: CABAC
- Context adaptive entropy coding

Robustness to data errors/losses plus flexibility opportunities arise by employing following concepts:

- Parameter set structure: gives necessary header information for conveyance. This forms key information for the decoder to detect losses such as sequence header or picture header.

- NAL unit syntax structure: NAL unit forms a structure where the syntax structure of H264/AVC fits in. as the name indicates, it forms an abstraction to the transport layer by forming a packet scheme.

- Flexible slice size
- Flexible Macro-block ordering FMO:
- Arbitrarily slice ordering
- Redundant Pictures
- Data partitioning
- SP/SI Synchronization/Switching Pictures

3.3.3. Network Abstraction Layer

This chapter will discuss the network abstraction layer concept. In order to isolate the underlying packet based transport medium, the standard defines a Network Abstraction Layer (see Figure 3.2). A coded H.264 video sequence consists of a series of NAL units, each containing an RBSP (Raw Byte Sequence Payload). Coded video elements are directly embedded in this sequence. (See Figure 3.4)

Figure 3.4. Sequence of NAL units each combined with NAL Header and RBSP Data

There are different types of NAL units. Each type is identified by an ID and the types are detailed in Table 7.1 of the standard document (ITU-T REC H264 2003). In order to decode H264/AVC video to the video coding layer, NAL unit semantics are to be followed. This is because the encoder's output (see Figure 2.19), which are compressed video elements are directly fed to the NAL stage of H264 encoder. This layer can be thought as the virtual transport medium of H264/AVC elements, because it abstracts the underlying transport environment.

3.3.4. Video Coding Layer Concepts of H.264/AVC

H.264 Standard follows a hybrid block based video coding scheme. The building blocks of this scheme are macro-blocks, which are formed of corresponding luminance and chrominance samples. The encoder uses temporal statistical redundancies between consecutive frames using inter prediction in temporal domain and transform based coding in spatial domain. This forms a hybrid block based structure to code video.

To improve compression efficiency, H.264/AVC uses a collection of methods. There is not a single method to be able to compress video most efficiently.

The H.264/AVC defines three sets of profiles each combines a set of coding tools. Profile is the specified set of syntax. Performance limits are controlled by the level information; these are the constraints for the syntax elements. The *Baseline Profile* supports intra and inter-coding (using I-slices and P-slices) and entropy coding with context-adaptive variable-length codes (CAVLC). The *Main Profile* includes support for interlaced video, inter-coding using B-slices, inter coding using weighted prediction and entropy coding using context-based arithmetic coding (CABAC). The *Extended Profile* does not support interlaced video or CABAC but adds modes to enable efficient switching between coded bitstreams (SP- and SI-slices) and improved error resilience (Data Partitioning). Potential applications of the Baseline Profile include videotelephony, videoconferencing and wireless communications; potential applications of the Main Profile include television broadcasting and video storage; and the Extended Profile may be particularly useful for streaming media applications. Figure 3.5 briefly shows the relations of profiles. (Richardson 2003a).

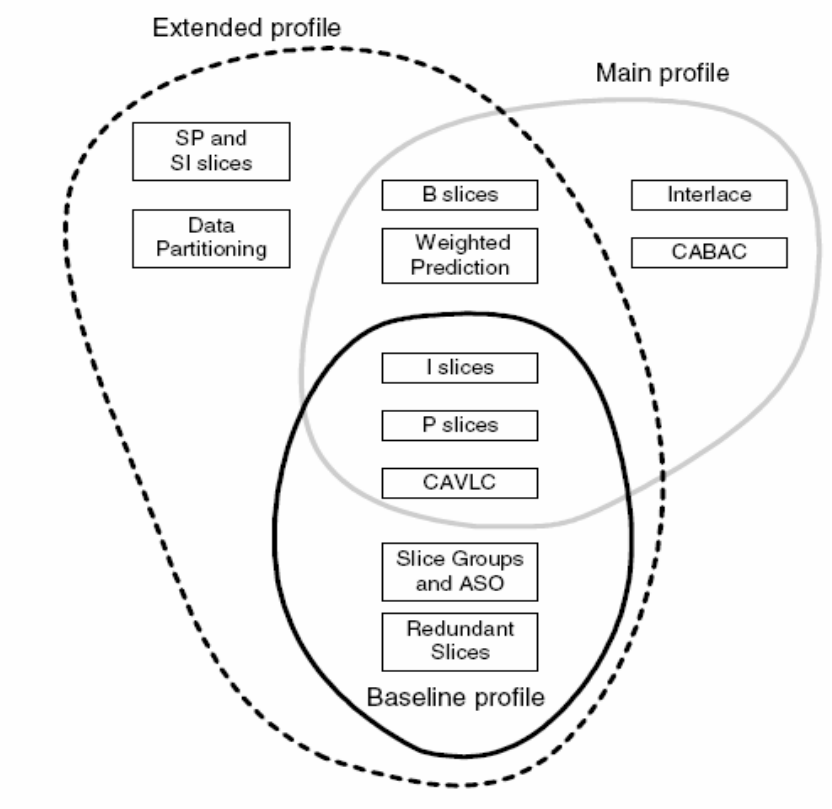

Figure 3.5. Three Profiles of the H264/AVC
(Richardson 2003a)

The video coding layer is again highly hierarchical as the previous video coding standards. Group of pictures are composed of sequential frames formed of slices and slices contain macro-blocks in pre-defined scan order, macro-blocks contain either transform domain information, or prediction information. This hierarchical encapsulated video ordering can be seen on Figure 3.6. Associated decoding layer function can be seen next to the layer of the video element.

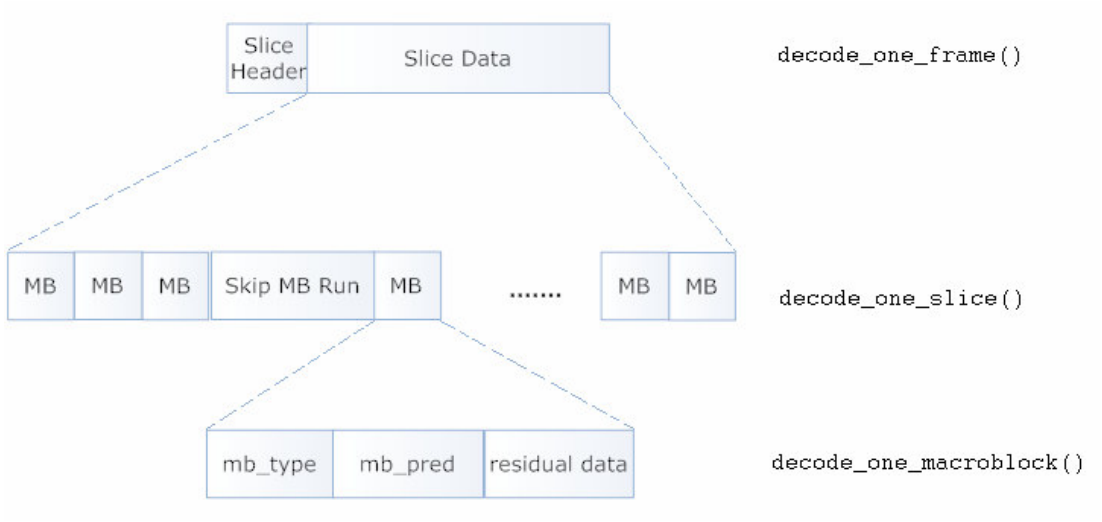

Figure 3.6. Encapsulated video coding elements and Associated Decoding Function

A video picture is coded as one or more slices, each containing an integral number of macro-blocks from 1 (1 MB per slice) to the total number of macro-blocks in a picture (1 slice per picture). The inter and intra prediction modes are directly results different kinds of slices and macro-block types. Slice types, descriptions and associating profile is shown in Table 3.1.

Table 3.1. Slice Types, Descriptions, Profiles
(Source: Richardson 2003a)

Slice type	Description	Profile(s)
I (Intra)	Contains only I macroblocks (each block or macroblock is predicted from previously coded data within the same slice).	All
P (Predicted)	Contains P macroblocks (each macroblock or macroblock partition is predicted from one list 0 reference picture) and/or I macroblocks.	All
B (Bi-predictive)	Contains B macroblocks (each macroblock or macroblock partition is predicted from a list 0 and/or a list 1 reference picture) and/or I macroblocks.	Extended and Main
SP (Switching P)	Facilitates switching between coded streams; contains P and/or I macroblocks.	Extended
SI (Switching I)	Facilitates switching between coded streams; contains SI macroblocks (a special type of intra coded macroblock).	Extended

Slice type information is directly collected by the information stored in sequence information of video data. This procedure is described in Chapter 4 with example video sequence.

A macroblock contains coded data corresponding to a 16 × 16 sample region of the video frame (16 × 16 luma samples, 8 × 8 Cb and 8 × 8 Cr samples) and contains the syntax elements described in Table 3.2. Macroblocks are numbered (addressed) in raster scan order within a frame (Richardson 2003a).

Macro-block syntax elements are shown in Table 3.2.

Table 3.2. Macro-Block Syntax Information
(Source: Richardson 2003a)

mb_type	Determines whether the macroblock is coded in intra or inter (P or B) mode; determines macroblock partition size
mb_pred	Determines intra prediction modes (intra macroblocks); determines list 0 and/or list 1 references and differentially coded motion. vectors for each macroblock partition (inter macroblocks, except for inter MBs with 8 × 8 macroblock partition size).
sub_mb_pred	(Inter MBs with 8 × 8 macroblock partition size only) Determines sub-macroblock partition size for each sub-macroblock; list 0 and/or list 1 references for each macroblock partition; differentially coded motion vectors for each macroblock sub-partition.
coded_block_pattern	Identifies which 8 × 8 blocks (luma and chroma) contain coded transform coefficients.
mb_qp_delta	Changes the quantiser parameter
residual	Coded transform coefficients corresponding to the residual image samples after prediction

This section discussed the video coding layer concepts. It should be noted that the whole coding elements of H.264/AVC has a large collection of algorithms in spatial, temporal and entropy coding perspectives, readers are referred to (Richardson 2003a and Richardson 2003b) and (ITU-T REC H264 2003) for a detailed discussion of new coding techniques listed in Section 3.3.2.

CHAPTER 4

PROPOSED WORK, RESULTS AND FUTURE WORK

4.1. Working Environment and Test Data

In order to experiment on different test scenarios, some existing tools are used, and constructed. This section describes the generation of H264 coded test data by using these tools.

Digital video data in various formats is converted to uncompressed YUV data. The uncompressed YUV output is then used as a source for the encoder to generate H264/AVC Coded bit-stream. The proposed test data generation scheme is illustrated in Figure 4.1.

MPEG 2 data contained in Program Stream format or Transport Stream format is converted to yuv4mpeg (WEB_3 2006) using “mplayer“(WEB_4 2006) which utilizes a collection of various codecs. Yuv4mpeg is a planar uncompressed video sequence data file format, with header information and planar 4:2:0 sampled YUV color spaced video. An application is also provided to convert yuv4mpeg data to YUV data without any header information.

Various test streams are available in [38]. In order to use these well-known test sequences, an application called “myuv2yuv” is constructed. This utility program is built for such purposes from scratch. The program converts uncompressed Yuv4mpeg with header information to raw yuv YV12 format. An application which appends two YUV file is also used to generate raw, uncompressed data. This tool is used to construct appended test streams, therefore a scene cut.

Uncompressed data structure is YV12 format. (WEB_5 2006)

The selected images are at QCIF, CIF, SIF, and PAL resolution type.

In order to generate H264/AVC Coded bit-streams, JM Reference Codec is used. (WEB_6 2006 and Suhring 2005)

The output of the encoding process is raw H264 coded bit-stream. This setup lets to exercise on various H264 bit-streams derived from various types of video sources.

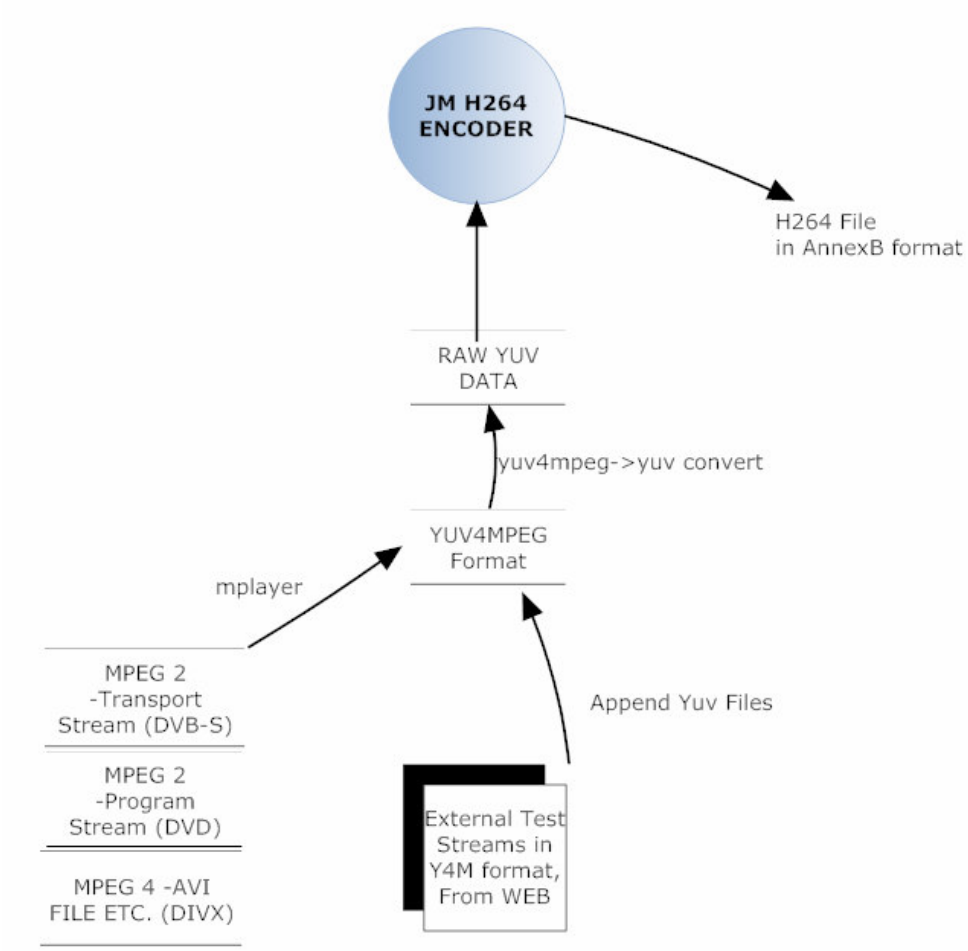

Figure 4.1. Test Data Generation Process for the Proposed Study

4.2. Proposed Work and Results

The test streams are constructed according to the Figure 4.1. The output is at main profile, using CABAC entropy coding and the sequence is in IPBPBPBP... form sequence. The number of B frames between P frames is decided by the encoder.

As discussed in Chapter 1, Scene cut detection is an important tool for video summarization proposes. The scene cuts are transitions between two consecutive shots. In order to detect abrupt scene changes, the macro-block type information in P slices are examined. The occurrence of a scene change coexists with the case where the encoder can not predict the current image by a reference to previous or forward slices, so the encoder is more likely to use intra prediction. This means a high number of Intra coded macro-blocks are inserted in P type slice.

In order to analyze the macroblocks, the decoding process of H264/AVC coded bitstream is done as follows:

4.2.1. Extracting Slices from NAL Unit Stream

The H264 bit-stream is formed of series of Network Abstraction Layer (NAL) units. Each NAL unit contains variable size of Raw Byte Sequence Payload. RBSP data may contain coded Video Coding Layer data, sequence or picture information header.

NAL unit types are detailed in Table 7.1 of (ITU-T REC H264 2003).

An example NAL unit sequence is seen in Figure 4.2. The decoder collects information about the incoming stream by sequence parameter set, and then picture parameter set. The interest is contained in coded Non-IDR picture, because the RBSP data contains the slice header and slice data. The hierarchical view of slice data, and macro-blocks is shown on Figure 3.6.

Figure 4.2. NAL Unit Types of A typical H.264/AVC Stream

4.2.2. Extracting Slice and Macro-block Information From Bit-stream

In order to collect statistics of macro-block distribution, the below decoding procedure is applied.

Once the slice data is extracted from NAL layer, the slice header gives information about the slice type. (See Figure 3.6) Slice types can be one of the following:

Intra (I) Type Slice: Contains only I type of macro-blocks.

Predicted (P) Type Slice: Contains P macro-blocks and/or I macro-blocks

Bi-prediction (B) Type Slice: Contains B type macro-blocks.

Switching P (SP) Type Slice and Switching I (SI) are used in extended profile. (Defined in previous Chapter)

The slice data contains the macro-blocks, each containing mb_type data, mb_prediction data, and the coded residual data. If FMO, is not used, each slice contains (img_height (in pixels) x img_width (in pixels)) / $(16)^2$ macro-blocks. So if the image is QCIF, i.e., 176x144 pixels, there are 99 total macroblocks in each slice.

Slice data syntax and macro-block layer syntax is given on pp 38 and 39 in (ITU-T REC H264 2003), respectively. In order to extract mb_type from the stream, Ex-golomb coded macro-block header information is read by ue(v) call. (sub-clause 9.1 (ITU-T REC H264 2003))

The above decoding process is given for describing the activity to collect information about the macro-block type distribution for defined slice types.

4.2.3. Detecting Scene Cuts

As discussed earlier, macro-block type's distribution is used to detect abrupt scene changes. For experimental purposes, two YUV test sequences are combined, and encoded for scene change study. The sequence "Suzie" at QCIF resolution is appended to "Trevor" sequence at QCIF resolution.

The algorithm counts the Intra coded macro-blocks in P type slices. A fixed threshold is used to detect the frames where scene cut is observed. Because the motion estimation does not help in encoding, the P type slice contains I type macro-blocks.

The program result can be seen below. Originally, Trevor sequence is 150 frames, and the Susie sequence starts at frame 151. Trevor sequence has a scene cut at frame number 59.

The I macro-block count vs frame number can be seen in Figure 4.3.

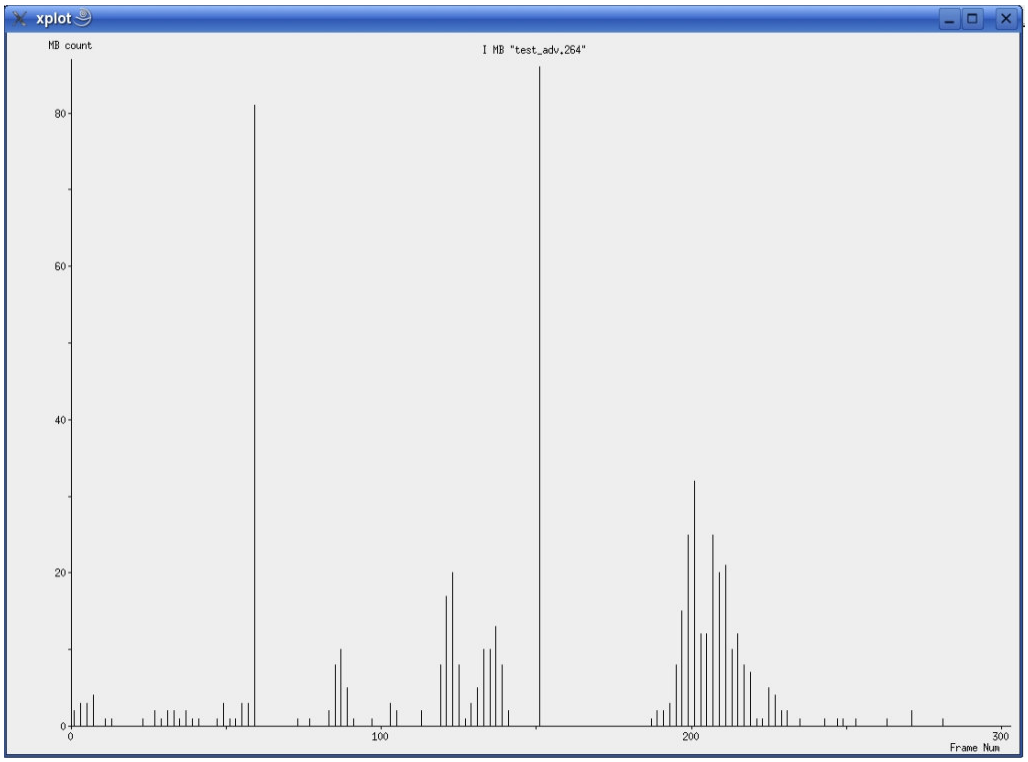

Figure 4.3. I Type Macro-block counts vs Frame Number forTrevor + Suzie Sequence

Applying a global thresholding, the frame numbers where the scene cuts has occurred can be easily detected from Figure 4.3. The frames where the mb_count exceeds the threshold, are possible candidates for the scene cuts. In this work, we specify the threshold δ_p as $MB_{TOTAL}/2$, where MB_{TOTAL} is the total number of macro-blocks in one slice.

A further look in Figure 4.3 can give the idea of motion's effect in detecting scene cut. The distribution of I macro-blocks exists on the neighbourhood of frame 200, but not detected as scene cut. This underlines the importance of selecting the threshold.

In order to detect scene cuts automatically, a patch, which is responsible to collect mb_type information located in P slice, is added to JM reference decoder. The output of the program is the scene cut summary in a single jpeg file, and scene cut frame numbers. A snapshot of the program output is as follows:

```
Possible Scene Cut @ 0 V= 99 THR= 13
Possible Scene Cut @ 60 V= 81 THR= 49
Possible Scene Cut @ 152 V= 86 THR= 49
```

The initial frame contains I type macro-blocks, so noted. Frame numbers 60 and 152 are where the scene cuts are occurred, as illustrated in Figure 4.5.

The program also outputs the scene summary in a single jpeg file as follows, the frames 60 and 152 are marked by the program, and the marked frames are extracted from the decoded stream.

Figure 4.4. The output of The program for Trevor and Suzie Sequence

The second test stream is constructed from the movie Barry Lyndon, 1975. The stream contains a duel shot, with a long zoom-in, and sharp scene cuts between the actor close-ups. The I type macro-block distribution along the P type slices can be seen in Figure 4.5.; the program output is listed below the figure.

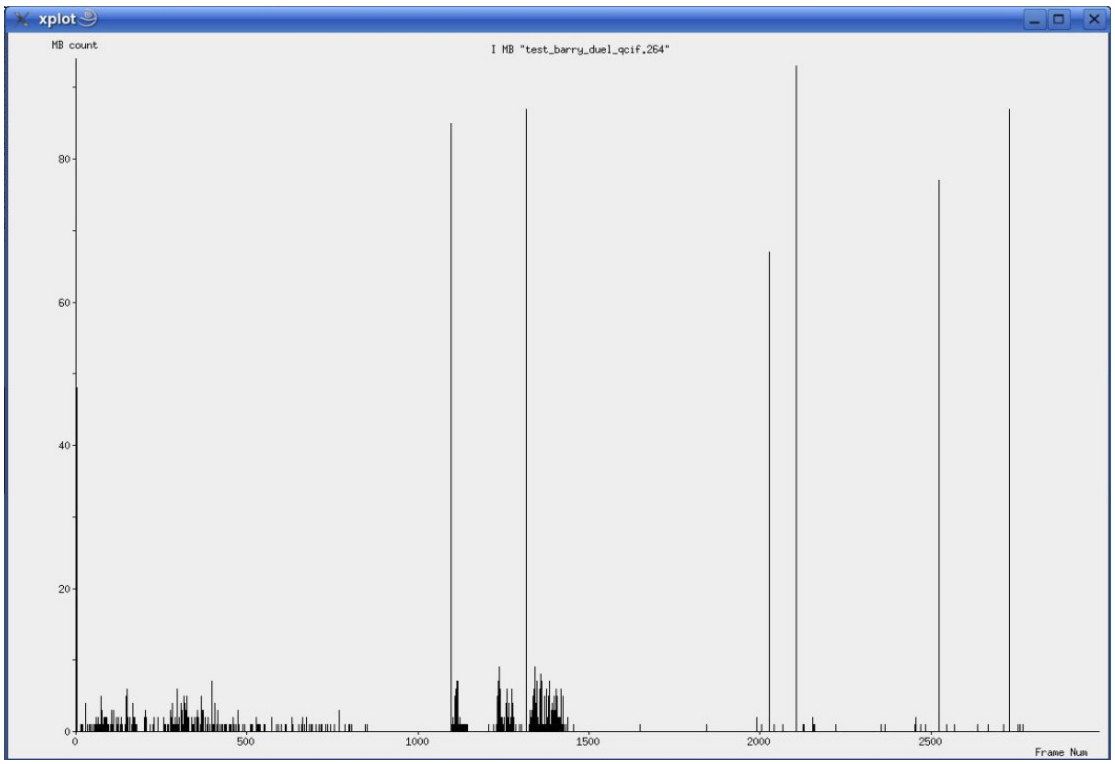

Figure 4.5. I Type MB's in P type Slices with Scene cuts as peaks

```
Possible Scene Cut @ 0 V= 99 THR= 13
Possible Scene Cut @ 2 V= 80 THR= 49
Possible Scene Cut @ 1098 V= 85 THR= 49
Possible Scene Cut @ 1318 V= 87 THR= 49
Possible Scene Cut @ 2028 V= 67 THR= 49
Possible Scene Cut @ 2108 V= 93 THR= 49
Possible Scene Cut @ 2524 V= 77 THR= 49
Possible Scene Cut @ 2730 V= 87 THR= 49
```

The program successfully finds the frame cuts in beginning, 1098, 1318, 2028, 2108, 2524, and 2730. The summary of the movie sequence is seen in Figure 4.6.

Figure 4.6. The Summary of a Video Sequence from Barry Lyndon (1975) Movie

The two examples discussed above contain direct, sharp scene cuts. These are called abrupt scene cuts. But many movies contain gradual scene cuts where a new scene begins with a fade in, or the previous scene fades out, in other words dissolves.

A dissolving sequence and its analysis is seen in Figure 4.7. The sequence is recorded from a live TV discussion programme.

Figure 4.7. A Dissolve Example

The scene change slowly occurs. When the macro-blocks for this change is analyzed, it can be noted that the number of I macro blocks increases as the frame number increases, reaches a top and decreases again. In this case, we need to detect the beginning and end frames of the dissolving segment. Figure 4.8 illustrates this effect.

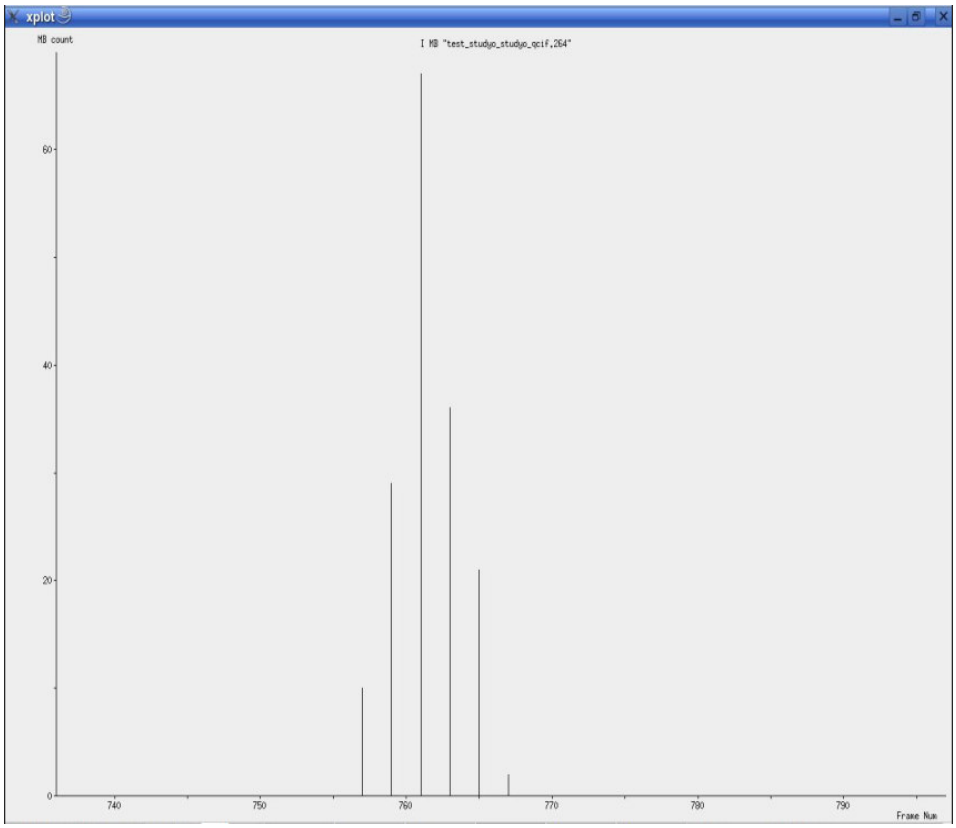

Figure 4.8. Example of a Dissolve, I Type MB's vs Frame Number

In order to detect gradual scene changes, the following ramp is detected by the following algorithm, which is shown in Figure 4.6.

The algorithm buffers the number of potential I type macro-block activity in P slices. Potential I type macro-block activity are points where gradual scene cut, dissolves, fade in or outs are observed, as shown in Figure 4.8.

A graphical user interface is developed, which summarizes the scene information as shown in Figure 4.7. This program enables the user to click on the scene of interest to play the movie starting from that scene using an external YUV sequence viewer. This program can give useful summary of the scenes under investigation. A screen shot is available in Figure 4.9.

Figure 4.9. Example Video Summarization Application

4.2.4. Future Work

Based on the same idea of collecting direct information from compressed media more information can be collected by observing the analysis being taken on the syntax elements of the coded video. This section briefly discusses ideas that might be adapted to the study.

4.2.4.1. Zoom Detection

Ideal zoom pattern is depicted in Figure 4.10. The zoom detection method can be adapted to our case is proposed by (Gerek and Altunbasak 1997). The variance of number of motion vectors versus motion vectors 'angle histogram is calculated, and the flatness of the angle is investigated. More flat variance means a zoom. However, it is also necessary to investigate the distribution of the I type macro-blocks in P slices. It can be noted that from Figure 4.7, there is a I macro-block activity during a zoom-out in the original sequence.

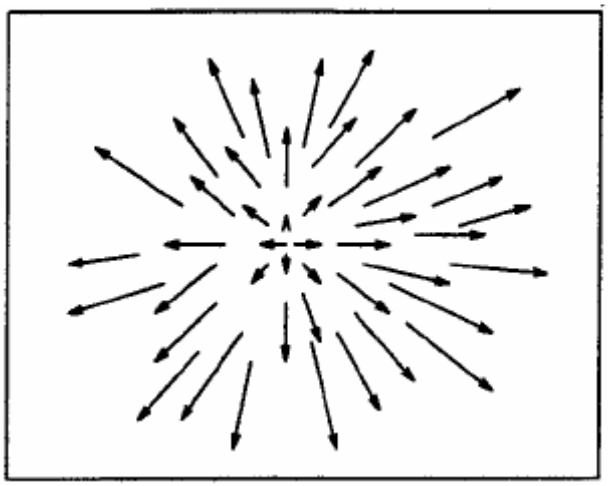

Figure 4.10. Ideal Zoom Pattern
(Gerek and Altunbasak 1997)

The zoom information between scene shots can be informed to the user in the application depicted in Figure 4.10.

4.2.4.2. Pan Detection

Like the zoom case, a statistical pattern is also extractable form the direction distribution of motion vectors. For an ideal pan, the directions of motion vectors are in the same relatively in the same direction.

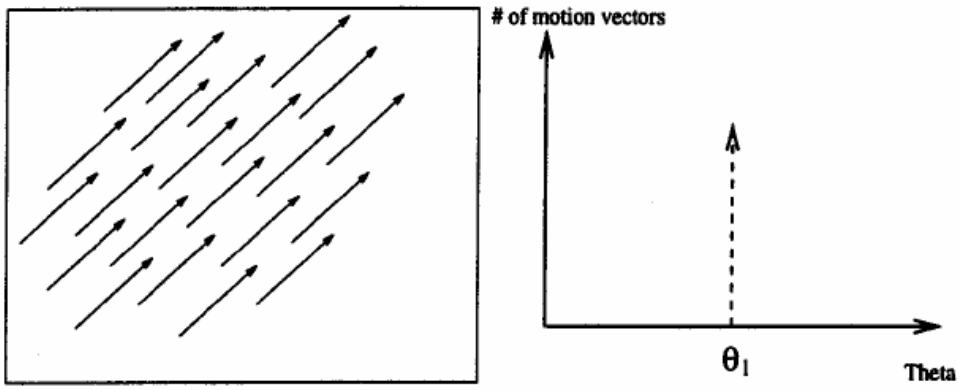

Figure 4.11. Ideal Pan (Left) and Angle of Motion vectors (right)
(Source: Gerek and Altunbasak 1997)

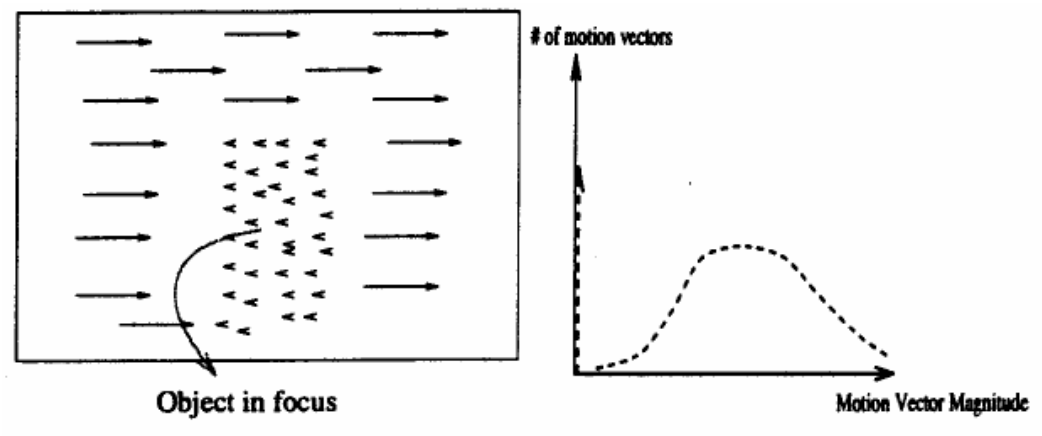

Figure 4.12. Panning When a Moving Object is on Focus
(Source: Gerek and Altunbasak 1997)

If panning is done when a moving object is in focus; the distribution of motion vectors is seen like seen in Figure 4.12. The object seems motionless so the motion vector magnitude is distributed as follows.

An ideal pan is detected, if the variance of number of motion vectors versus angle histogram is impulsive as seen on the right side of Figure 4.11.

The above discussions panning can be added to the program illustrated in Figure 4.11 for a more compact representation of the sequence to be summarized.

CHAPTER 5

CONCLUSIONS

In this study, H264/AVC coded bit-stream is analyzed in terms of CODEC internals, to provide an approach to summarize the video into meaningful scenes.

The application gives top level summary information about the scenes. This gives the user the opportunity to visualize the highlighted excerpts from a H264/AVC coded video. As a codec, H264/AVC is selected because of the increasing popularity of the codec as a result of the accomplishments of the codec in reducing bit rates, scalability, and supportability in various environments from packet video networks, to wireless networks, from high definition video to video on demand.

The video analysis is heavily dependent to the parsing, decoding processes of the video codec. The syntax and semantics descriptions in the standard are the crucial definitions used by the study, and getting more familiar with such well standardized data set, applications may take shorter time to be developed.

An important part of the study is that the video analysis is directly taken on compressed video data. In terms of computational costs, this makes the application computation efficient. This logic may be carried to network layer 3 devices, who are responsible to manage packet network traffic management, by understanding the video data distribution by directly looking at the bit-stream itself without fully decompressing the data. The study also gave the chance to get more familiar with compressed domain video understanding techniques. The study may also be combined with advanced pattern recognition and learning algorithms and techniques to yield more robust and extended studies, since the data extracted in different forms of syntax elements forms a statistical nature which can be processed in a decision making or classification.

The performance of the study depends on the selected metric of threshold. The motion in the video introduces I type coded macro-blocks in P type slices, so the threshold should be selected such that these kind of I type macro-block activity is eliminated to give correct results of scene cuts in the scene.

The idea lying under this study can be implemented to hybrid block based video codec scheme; whereas H264 AVC standard provides P type slices formed of I type macro-blocks, this flexibility which is provided by the standard is used in this study.

The application can be enhanced by adding, zoom, panning, fading in/out detection functionalities. By working on more test data, provided by an automatic testing environment will make the application more robust. More notes can be added to the user interface of the output of the project. The interactivity, portability and user friendliness of the application is open to develop. This study can serve as a starting point for video analysis, indexing or skimming system, noting that there will be a great need for video skimming applications since the amount of the content of digital video increases dramatically in the following years.

REFERENCES

Ahmed, N., Natarajan, T. , and Rao, K. R., 1974. "Discrete Cosine Transform"**,** *IEEE Transactions on Computers,* Vol. C-23 pp. 90 – 93.

Aslandogan, Y. A. and Yu, Clement, T., 1999. "Techniques and Systems for Image and Video Retrieval", *IEEE Transactions on Knowledge and Data Engineering* Vol. 11, No. 1.

Drake, A. W. 1967 *Fundamentals of Applied Probability Theory,* (McGraw-Hill), pp 44.

Fernando, W. A. C., Canagarajah, C. N. and Bull, D. R. 2000. "Fade-in and Fade-out Detection in Video Sequences using Histograms", IEEE International Symposium on Circuits and Systems, Geneva, Switzerland, vol.4 , pp. 709 – 712.

Gargi, U., Rangachar, K. and Strayer, Susan, H. 1995. "Performance Characterization of Video-Shot-Change Detection Methods", I*EEE Transactions on Circuits and Systems for Video Technology*. Vol.10, No 1.

Gerek, Ö. N. and Altunbasak, Y. 1997. "Key frame detection from MPEG video data", *IS&T/SPIE Visual Comm. Image Proc*., VCIP'97, San Jose, CA, vol. 3024, part. 2, pp. 920-925.

Ghanbari, M. 2003. *Standard Codecs: Image Compression to Advanced Video Coding* , (The Institution of Electrical Engineers, United Kingdom).

Gunsel, B. and Tekalp, A.M., 1998. "Content Based Video Abstraction**",** ICIP 98 International Conference on Image Processing, pp. 128 - 132 vol.3.

Hsu, P.R. and Harashima, H., 1994. **"**Detecting scene changes and activities in video databases", IEEE International Conference on Acoustics, Speech and Signal Processing Volume v, pp. V/33 - V/36 vol.5.

"ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC Draft", ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, 2003.

Lee, S. W., Kim, Y. M. and Choi, S. W., 2000. "Fast scene change detection using direct feature extraction from MPEG compressed videos"**,** *IEEE Transactions on Multimedia*, Volume 2, Issue 4, pp. 240 – 254.

Li, Y., Lee, S. H., and Yeh, C. H.; and Kuo, C. C. J. 2006. "Techniques for movie content analysis and skimming: tutorial and overview on video abstraction techniques"**,** *IEEE Signal Processing Magazine*, Volume 23, Issue 2, pp. 79 – 89.

Mee-Sook, L. and Bon-Woo, H.; Sanghoon, S. and Seong-Whan, L., 1998. "Automatic video parsing using shot boundary detection and camera operation analysis", Fourteenth International Conference on Pattern Recognition, Vol. 2, pp. 1481 – 1483.

Mezaris, V., Kompatsiaris, I., Boulgouris, N.V. and Strintzis, M.G. 2004. "Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing and retrieval", IEEE Transactions on Circuits and Systems for Video Technology, Volume 14, Issue 5, pp. 606 – 621.

Otsuji, K. and Tonamura, Y. and Ohba, Y. 1991. "Video Browsing using brightness data", *Visual Comm. And Image Processing*, Vol. SPIE-1606 pp 980-989.

Oppenheim, A. V., and Schafer, R.W. 1989. *Discrete Time Signal Processing*, (Prentice Hall, New Jersey) pp 4-6.

Pei, S. C, and Chou, Y. Z. 1999. "Efficient MPEG Compressed Video Analysis Using Macroblock Type Information", *IEEE Transactions on Multimedia*, Volume 1, Issue 4, pp. 321 – 333..

Richardson, I. E. G., 2003. *H.264 and MPEG4 Video Compression- Video Coding for Next Generation Multimedia*, (Wiley & Sons, England) pp. 29.

Richardson, Iain E. G., 2003. *Video Codec Design – Developing Image and Video Compression,* (Wiley & Sons U.K.), pp 27-45.

Ruey-Feng, C., Kuo, W. J. and Tsai, H. 2000. "Image Retrieval on Uncompressed and Compressed Domains" International Conference on Image Processing, Volume 2, 10-13 pp.546 - 549 vol.2..

Sadka, A. 2002. *Compressed Video Communications*, (John Wiley & Sons) pp. 40-47.

Shahraray, B. 1995. "Scene Change Detection and Content Based Sampling of Video Sequences" Digital Video Compression, Algorithms and Technologies, Vol. SPIE-2419, pp 2-13.

Sikora, T., 1997. "MPEG Digital Video-coding Standards", IEEE Signal Processing Magazine, Vol. 14, Issue 5, Sept. Page(s):82 – 100.

Sühring, K., 2005. *H.264/MPEG-4 AVC Reference Software Manual,* (Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG), pp. 4.1.

Shen, K. and Delp, E.J. 1995. "A Fast Algorithm For Video Parsing Using MPEG Compressed Sequences", International Conference on Image Processing Proceedings, Volume 2, 23-26 pp. 252 – 255.

Tekalp, M. A. 1995 *Digital Video Processing,* Prentice Hall pp 432.

Tse, K., Wei, J., and Panchanathan, S. 1995. "A Scene Change Detection Algorithm for MPEG Compressed Video Sequences". Canadian conference on Electrical and Computer Engineering, Vol. 2, pp. 827 – 830.

Wallace, G. K. 1992. "The JPEG Still Picture Compression Standard"**,** *IEEE Transactions on Consumer Electronics*, Volume 38, Issue 1 pp. xviii – xxxiv.

Wang, Y, Ostermann, J, and Zhang, Y. 2002. V*ideo Processing and Communications*, by (Prentice-Hall), pp. 80.

WEB_1, 2006. DVB Consortium's Web Site, DVB-S2 White Paper, 25/12/2006, http://www.dvb.org/documents/white-papers/wp06.DVB-S2.final.pdf

WEB_2, 2006, European Broadcasting Union Web Site, 22/12/2006, http://www.ebu.ch/en/technical/trev/trev_300-morello.pdf

WEB_3, 2006. MJPEG Tools Manuals Web Site, 11/12/2006, http://www.penguin-soft.com/penguin/man/5/yuv4mpeg.html

WEB_4, 2006. Mplayer- The Movie Player Home Page, 10/10/2006, http://www.mplayerhq.hu/

WEB_5, 2006. FourCC Home Page, 10/10/2006, www.fourcc.org/yuv.php

WEB_6, 2005. JM Reference Software Home Site, 10/10/2006, http://iphome.hhi.de/suehring/tml/download/

WEB_7, 2007 Berkeley Multimedia Research Center MPEG FAQ, 10/01/2007 http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_overview.html

WEB_8,2007 ImageMagick Tools http://www.imagemagick.org/

Wiegand, T., and Sullivan, G. J., Bjøntegaard, G., Luthra, A., 2003. "Overview of the H.264/AVC Video Coding Standard"*, IEEE Trans. on Circuits and Systems for Video Technology*, VOL. 13, NO. 7.

Yeo, B. L., and Lui, B.,1995a. "Rapid Scene Analysis on Compressed Video", *IEEE Transactions on Circuits and Systems for Video Technology*. Vol 5, No 6.

Yeo,B. L. and; Liu, B. 1995b; "A unified approach to temporal segmentation of motion JPEG and MPEG compressed video**",** IEEE Proceedings of the International Conference on Multimedia Computing and Systems, pp. 81 – 88, 15-18.

APPENDIX A

CODE IMPLEMENTATION

The study requires an implementation on JM Reference Software decoder. More information about this software is included in WEB_6, 2005.

The selected platform is Pentium® 4 CPU Running at 2.4 GHz. The Operating System selected is Suse Linux 10.1 (i586 Compliant, Kernel version is Linux 2.6.16.13-4. The implementation language is C Programming Language.

Additionally, the program uses the external executables to provide a jpg output which contains listed below, so in order to compile the source code without error, the following executables must be in correct path :

Extractyuv: Executable that extracts a single frame from an uncompressed yuv sequence, this code is written from scratch. *Extractyuv* application works as follows:

extractyuv [-n] { | {[-i bitstream.yuv] [-o output.yuv]

Parameters

Options
-n : Number of frame to extract
-i : Input file name.
-r : Input file name.
-o : Output file name. If not specified default output is set as output.yuv

Supported video file formats
Input : .yuv -> YUV bitstream sequence .
Input : integer -> The number of Frame in the sequence
Input : res -> resolution of the image
Output: .yuv -> RAW file. Single Frame.

Examples of usage:
extractyuv
extractyuv -n
extractyuv -i bitstream.y4m -n framenum -r CIF -o output.yuv

Convert: An application provided by (WEB_8, 2007) to convert a single yuv file to jpeg file.

Montage: An application provided by (WEB_8, 2007) to montage several jpg files to a single jpeg file.

Mplayer : A general open-source multimedia player running on Linux Plays uncompressed.

The main part of the code is implemented in *ldecod.c, image.c, macroblock.c* files.

In the bottom level, *void decode_one_slice(struct img_par *img,struct inp_par *inp)* function in *image.c* file is where the macroblocks are processed so this function is where Scene Cuts are processed via call to following *ProcessSceneCuts* function, listed below. Threshold variable is derived from Frame Size. When the I type activity excesses threshold, the index is written to text file "scene_cuts.txt", via call *fprintf(img->fp_scene_write,"%d \n",index);*

```
int ProcessSceneCuts(int sample,int index)
{

if(sample > Threshold)
{
        printf("Possible Scene Cut @ %d V= %d THR= %d\n",index,sample,Threshold);
        fprintf(img->fp_scene_write,"%d \n",index);//,(img->FrameSizeInMbs  / 2));
}
```

When whole decoding is done to the end of stream, marked with EOS, "*scene_cuts.txt"* text file is used to extract frames by using a system call made to *extractyuv* application as follows.

```
        strcpy(jpgfiles," ");
        img->fp_scene_read = fopen("scene_cuts.txt","r");

        while(read_return != EOF)
        {
        read_return = fscanf(img-fp_scene_read,"%d",&possible_scene_cut);
        sprintf(extractyuvbuf,"./extractyuv -i test_dec.yuv -o scene_%d.yuv -r QCIF -n %d \n",scene_count,possible_scene_cut);

        system(extractyuvbuf);
        sprintf(convertbuf,"convert -size 176x144 scene_%d.yuv scene_%d.jpg\n",scene_count,scene_count);
        system(convertbuf);
        printf(">>>>Possible scene cut at %d\n",possible_scene_cut);
        scene_count++;
        }
```

sprintf*(jpgfiles,"montage *.jpg scene_summary.jpg\n");*
system*(jpgfiles);*

sprintf*(mplayerbuf,"mplayer test_dec.yuv -demuxer rawvideo -rawvideo w=176:h=144:fps=25\n");//, scene_w,scene_h);*

system*(mplayerbuf);*

With the last system call, the program outputs are constructed in the location of the executable, and the ***system****(mplayerbuf);* system call plays the clip.