CONTEXT-BASED COMPLEXITY REDUCTION APPLIED TO H.264 VIDEO COMPRESSION

by

Laleh Sahafi
B.Sc., Sharif University of Technology, 2002.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
Spring 2005

APPROVAL

Name: Laleh Sahafi

Degree: Master of Applied Science

Title of thesis: Context-based complexity reduction applied to H.264 video compression

Examining Committee: Dr. Albert M. Leung
Chair

Dr. Rodney Vaughan, Professor, Engineering Science
Simon Fraser University
Senior Supervisor

Dr. Tejinder S. Randhawa, Adjunct Professor, Engineering Science
Simon Fraser University
Supervisor

Dr. R.H. Stephen Hardy, Professor, Engineering science
Simon Fraser University
SFU Examiner

Date Approved: March 14, 2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

Abstract

The Achilles' heel of video over wireless services continues to be the limited bandwidth of wireless connections and the short battery life of end-user handheld devices such as cell-phones and PDAs. Efficient coding and compression techniques are required to meet the quality of service requirements of such services while effectively managing the bandwidth and power. In this thesis some strategies are proposed in order to reduce the complexity of the encoder, and are applied to H.264, the latest video compression standard. Using the knowledge of the context scenes in the video sequences, less important regions in the frames are isolated and the processing is reduced. Experimental results are presented to demonstrate the viability of the proposed strategies for minimizing the processing time of H.264 while maintaining desired quality and low bitrate. The results indicate about 50% reduction in the computational complexity of H.264.

To my dear husband, Soroush,
my parents and sisters
for their support at every moment of my life

Acknowledgments

I appreciate Dr. Rodney Vaughan, my senior supervisor, and Dr. Tejinder Randhawa, my co-supervisor, for their guidance, suggestions, assistance and encouragement during my course of research. Many thanks to Dr. Rodney Vaughan, for reading and correcting this thesis patiently.

My deep regards to Dr. Jacques Vaisey, my deceased supervisor, who helped me come to SFU, settle down in Canada and learn first stages of video compression.

Thanks to Dr. Stephen Hardy and Dr. Albert Leung for reviewing this thesis.

My special thanks to Roozbeh Ghaffari and Sara Khodadad, skilled computer engineers and my good friends, who always solved my software problems.

I am grateful to Dr. Alexis (Alexandros) Michael Tourapis, an expert in H.264 standard. He answered all my questions about this standard.

I owe thank to Wayne Huang and Susan Chiu, who implemented face recognition software and let me employ it.

I also would like to thank Dr. Jie Liang for his suggestions and help.

Finally, I want to take this chance to express my gratitude to all my supportive and caring friends in Iran, Europe and North America, who are there for me whenever I need them.

Contents

List of Tables

List of Figures

Preface

Rapid development of signal processors, telecommunication systems and video digitization has fostered new applications, including video games, movies, video conferencing, video telephony, video on demand and several others. Video transmission and storage are not practical without compression techniques because of the huge amount of information in digital video. Therefore, video coding standards have been developed to compress this data. Generally, the high compression gain and good reconstructed video quality are achieved at the expense of complex CODEC algorithm, which costs more in both time and power consumption. Available power resources are restricted in handheld devices and processors. The bandwidth of transmission channels is also restricted, which in turn limits the bitrate of compressed data. Owing to these limitations, it is necessary to employ a technique which meets the quality of service requirements while effectively managing the bandwidth and power resources. Here, we present a new approach in order to reduce the power consumption based on the context of the video sequence, and maintain the desired quality and bitrate at the same time.

In this thesis, video coding techniques and standards are introduced, and then, the proposed method and its results are discussed. The introductory sections draw heavily on [1], and this is denoted as a reference at the end of each section.

The topics covered here are organized in three chapters: video coding concepts, video coding standards and context-based complexity reduction.

- Chapter 1: this chapter comprises introductory material, required for dealing with video signals and compression. The video and color formats, objective and subjective video quality are explored and compared. The goals and advantages of motion-estimated and predictive coding are clarified. Transform coding, especially Discrete Cosine Transform, used in almost all video compression standards, is introduced.

Then, quantization and statistical compression processing (entropy coding) is explained. This chapter concludes with a description of the generic DPCM/DCT CODEC.

- Chapter 2: this chapter focuses on H.264, which is the platform of this thesis. H.264 is the latest video compression standard, and was developed as a joint project of the Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG) in 2003. Chapter 2 includes the video coding layer, the network abstraction layer, inter and intra coding, the integer transform used in this standard, special features offered by H.264 and its various profiles. Also, the older standards such as H.261, H.263, MPEG-1, MPEG-2, and MPEG-4 are introduced.

- Chapter 3: a new complexity reduction method is proposed and applied to the H264 coding system that results in decreased power consumption and processing time of the CODEC. This is achieved by treating various areas of each frame differently according to their importance for the viewer. A segmentation algorithm is exploited to classify regularly the video sequence to foreground and background areas. Some experiments are performed to support the proposed method. In these tests, H.264 software is employed and the required modifications are added to the core of this software.

There were some reasons that made this work complicated. When the H.264 standard was prepared, its software underwent rapid development. Many known and unknown bugs were in the software and the software did not support all the features of the standard. Since then, many new versions have been released with some bugs fixed and new features added. This made previous software versions obsolete and keeping up with the changes has been challenging. Also, the software documentation was not clear, which required continuous communication with the original developers.

Glossary

AVC	Advanced Video Coding
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CODEC	COder/DECoder
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
FB	Foreground/Background
FMO	Flexible Macroblock Order
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ITU	International Telecommunication Union
JVT	Joint Video Team
MB	Macroblock
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
MV	Motion Vector
NAL	Network Abstraction Layer
PSNR	Peak Signal to Noise Ratio
PSNRB	Peak Signal to Noise Ratio of Background
PSNRF	Peak Signal to Noise Ratio of Foreground
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
SI	Switching Intra
SP	Switching Prediction
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VLC	Variable Length Code

Chapter 1

Video Coding Concepts

Digital video has taken the place of analogue video in a wide range of applications. It has a number of key advantages over traditional analogue video and television. Digital form makes the video like the other data, so the same techniques and systems can be used for storage, processing, and transmission. Digitized video signal requires a very high data rate. For efficient storage, processing and transmission of digital image and video, it has been necessary to develop techniques for compressing the video data. Current video coding techniques enable video data to be compressed by between 20 and 50 times.

Generally speaking there is a large amount of statistical and subjective redundancy in digital video sequences. Video compression or video source coding is the process by which redundant information from a source is removed resulting in a saving in terms of space and bandwidth. The information can be recreated by the inverse process known as decoding or decompressing. There are two methods of image and video compression: lossless and lossy coding. If the reconstructed image or video matches exactly with the original information, the method is lossless. The lossy method compresses the data with loss of information so that the reconstructed data is not the same as the original one. Lossy methods are widely used in digital image and video application due to the high compression ratio provided.

The range of applications for digital video continues to grow and includes the following: video conferencing and video telephony; home entertainment digital video; broadcast digital television; video databases; video on demand; medical applications; and several others.[2][3]

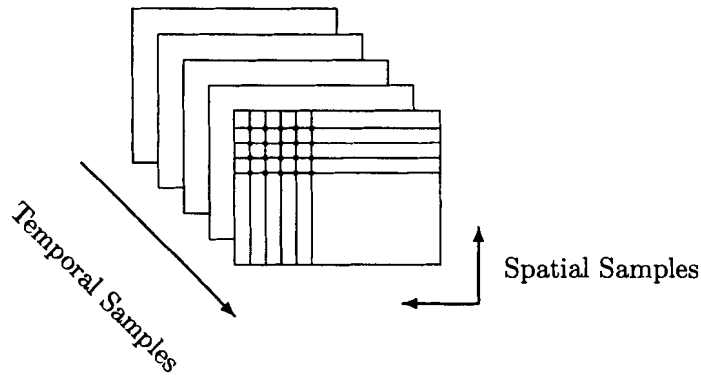

Figure 1.1: Temporal and Spatial samples

1.1 Digital Video

A natural visual scene is spatially and temporally continuous. Representing a visual scene in digital form involves sampling the real scene spatially, which is usually on a rectangular grid in the video image plane and temporally, which is sampling as a sequence of still images at regular intervals in time (Figure 1.1). Digital video is the representation of a sampled video scene in digital form. Each sample, picture element or pixel, is represented as a number or set of numbers that describes the brightness and color of the sample.[1]

1.1.1 Frames and Fields

A video signal may be sampled as a sequence of complete frames or as a series of interlaced fields. In an interlaced video sequence, there are two fields instead of each frame. A field consists of either the odd-numbered or even numbered lines within a complete video frame (Figure 1.2). The advantage of this sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving the appearance of smoother motion.[1]

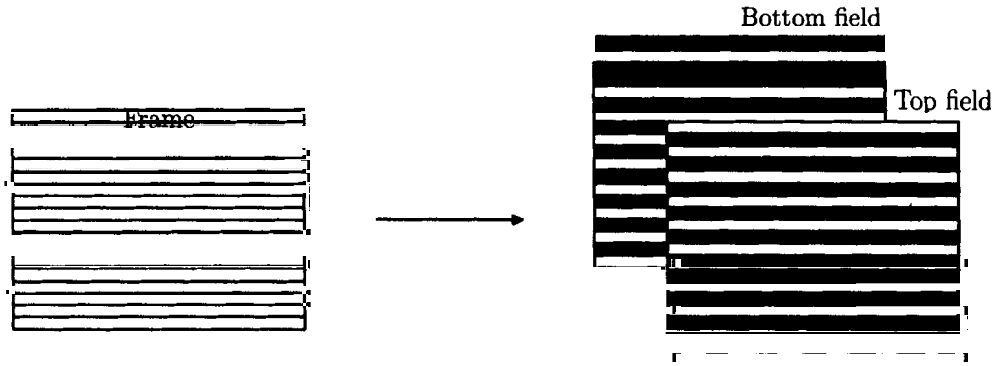

Figure 1.2: Interlaced video frame

1.1.2 Color Formats

The standards use a variety of color formats. The basic color format is RGB color space. In this color space, each pixel is represented by three numbers indicating the relative ratio of red, green and blue. These are the three primary colors of light which can combine with different proportions to produce any other color. The RGB color space is not very efficient for representation of images because the human visual system is more sensitive to brightness than the colors; however in the RGB system the color components are equally important in producing the other colors so each component has the same resolution, and brightness (luminance) is present in all three components. That is why many image and video coding standards use luminance and color-difference signals, for example, YIQ, YUV, YCbCr, and SMPTE 240M color formats. Most image and video compression standards adopt YCbCr color format as an input signal. This color space was developed as part of ITU-R BT.601 in the creation of digital video component standards [4]. Y is the luminance component, a weighted average of R, G and B. Cr and Cb are the difference between the luminance and the red and blue color. $Cr = R - Y, Cb = B - Y$. In the YCbCr color space, the Cb and Cr components can be represented by lower resolution than Y because of the less sensitivity of human visual system to color. This reduces the data required to represent chrominance component without having an obvious effect on visual quality. Figure 1.3 shows the three popular patterns for sampling Cr and Cb. 4:4:4 means that the three components have the same resolution and hence a sample of each component exists at every pixel position. In 4:2:2 sampling, the chrominance components have the same vertical resolution but half the horizontal resolution. 4:2:0 means that Cr and Cb each have half the horizontal and vertical resolution of Y.[5][6]

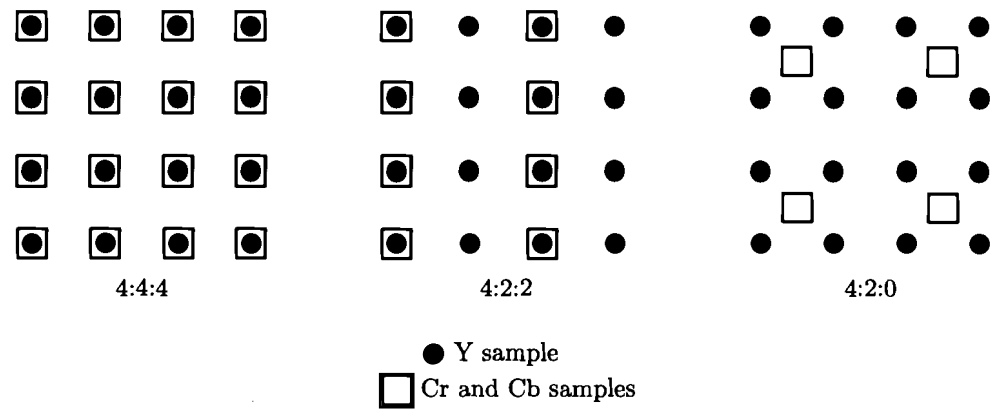

Figure 1.3: Sampling

1.1.3 Standards for Representing Digital Video

The video coding standards can compress different video formats but in practice, usually the video signal is transformed to one of a number of intermediate formats before the compression and transmission. The Common Intermediate Format(CIF) is the basis for a popular set of formats listed in Table 1.1. The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is suitable for standard-definition television and DVD-video; CIF and QCIF are used for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bitrate are limited.[1]

Format	Luminance resolution (Horiz.× Vert.)
CIF	352 × 288
QCIF	176 × 144
4CIF	704 × 576
Sub-QCIF	128 × 96

Table 1.1: Intermediate formats

1.1.4 Video Quality

An important design object for a digital video system is that the viewer is satisfied with the quality of the produced video. Determination of visual quality is necessary for assessment and comparison of video coding and communication systems. Visual quality is naturally subjective and therefore, it is affected by many subjective factors that make it difficult to

get a precise measure of quality. Objective methods of measurement give the accurate result but there are differences between this result and the subjective experience of a human viewer watching a video display.

Subjective quality measurement

A complex interaction between the components of the human visual system, the eye and the brain make our opinion of a visual scene. The clarity of various parts of images and the smoothness of the motion of the frames are factors in the perception of visual quality. However, a viewer's opinion of quality is also affected by other factors such as the presentation atmosphere, the observer's state of mind and the level to which the observer interacts with the visual scene. Also, recently viewed video streams have more effect on the perception of the quality [7][8]. All of these factors make it very complicated to measure visual quality accurately.

Various test methods for subjective quality measurement are described in ITU-R recommendation BT.500-11 [9]. One of the most common quality assessment procedures is the Double Stimulus Continuous Quality Scale (DSCQS) method in which a pair of images or short video sequences A and B is consecutively displayed to an evaluator, and the evaluator gives A and B a score by marking on a continuous line with five intervals. The DSCQS test is generally accepted as a realistic measure of subjective visual quality but the results depend on the evaluator, the video sequence test and the environment. These factors make it expensive and time-consuming to perform the DSCQS tests thoroughly.

Objective Quality measurement

Because of the problems of subjective measurement, objective measures of visual quality are used a lot. The most commonly used objective measure is peak signal to noise ratio, PSNR, which is shown in equation(1.1) [5]. MSE (Mean Square Error) is the mean square of the difference between an original and an impaired image or video frame. $(2^n-1)^2$ is the square of the highest possible signal value in the image, where n is the number of bits per image sample.

$$PSNR_{dB} = 10\log_{10}\frac{(2^n-1)^2}{MSE} \tag{1.1}$$

Figure 1.4: Original image

Figure 1.5: Impaired images, Right image: PSNR=49.95dB, Left image: PSNR=40.73dB

The PSNR measure has some limitations. PSNR requires an original image for comparison but this may not be available in every case and it may not be easy to verify that an original image has perfect fidelity. A more important problem is that PSNR does not correlate well with subjective video quality measures. This problem is obvious in figure 1.5. This figure shows two impaired versions of the original image in figure 1.4. The PSNR of the left image is 40.73 dB, whereas the PSNR of the right image is 49.95 dB. By definition of the PSNR, the quality of the second image is better than the first one, while most viewers would rate the second as significantly poorer than the first, because the face in the right image, is not as clear as the face in the left image. This example shows that PSNR ratings do not necessarily correlate with true subjective quality.

Development of a method for objective measurement that closely approach subjective results has been one of the problems and none of the proposed methods [10][11][12] could

be the clear substitute to subjective tests. The PSNR is widely used as a rough objective measure for visual quality and so we will use this parameter for quality comparison in this thesis; however, it is important to remember the restrictions of PSNR when comparing different systems and techniques.

1.2 Predictive Coding

One of the simplest image compression techniques is differential pulse code modulation (DPCM). In a DPCM system, a prediction of each pixel value is produced based on the neighboring pixels. The prediction error between the predicted pixel value and the actual value is quantized and transmitted instead of the pixel value itself. This system exploits the high correlation between neighboring pixels which causes spatial redundancy. Because of this correlation, the prediction error is small so the transmitted data is less than the situation when the whole image is transmitted without using predictive coding. The decoder must use the same prediction method as the encoder. In the decoder, predicted pixel is produced based on previous reconstructed pixels. The predicted pixel is added to the transmitted error value in order to reconstruct the current pixel.[2]

1.3 Video CODEC

A video encoder compresses a video input signal and the decoder reconstructs a copy or approximation of the original signal from the compressed data. The concept of differential prediction can be extended to enable efficient encoding of moving video sequences. A video encoder takes advantage of both temporal and spatial redundancy to achieve compression. Consecutive frames in a video sequence usually have a high similarity. This is correlation in temporal domain, which is usually due to movement in the scene. Also there is usually high correlation between neighboring pixels, which is correlation in the spatial domain.

Temporal coding, spatial coding and entropy coding are three main functions of a video encoder. Temporal coder tries to decrease the temporal redundancy by exploiting the correlation between successive frames. The input to the temporal coder is an uncompressed video sequence. The temporal coder usually constructs a prediction of the current video frame based on the reference frames and computes a residual frame by subtracting the prediction frame from the original one. The prediction method is usually called motion estimation.

The output of the temporal coder is the residual frame and a set of parameters, typically a set of motion vectors explaining how the motion was compensated. A spatial coder makes use of similarities between neighboring samples to reduce spatial redundancy. This is attained by applying a transform to the samples and quantizing the results. The transform converts the samples into a new domain. In this domain the energy of the samples has been compacted into some coefficients. All of the transform coefficients are quantized to remove unimportant values (close-to-null coefficients) and only the significant values which have the most of the energy of the signal remain. The output of the spatial coder is a set of quantized transform coefficients. The parameters of the temporal and the spatial coder are compressed by an entropy encoder (section 1.3.4). The entropy encoder produces a compressed sequence consists of coded motion vector parameters, coded residual coefficients and also header information for transmission or storage.

The video decoder reconstructs a video frame from the compressed bit stream. The coefficients and motion vectors are decoded by an entropy decoder after which the spatial model is decoded to reconstruct a version of the residual frame. The decoder uses the motion vector parameters and reference frames to create a prediction of the current frame, then this predicted frame is added to the residual frame to recreate the current frame. This frame is not usually the same as the original frame, which is compressed by the encoder, because most compressions are lossy. One of the goals of CODECs is to reduce this difference as much as possible while using as few bits as possible for the compressed signal. This is a tradeoff between compression and quality.[1]

1.3.1 Motion Estimation and Compensation

As mentioned before, for reduction of temporal redundancy, a predicted version of current frame is subtracted from it to give the residual frame. For finding the prediction frame, motion estimation technique is used. Motion estimation, in general, can improve the prediction accuracy. The block-matching algorithm (BMA) has been verified to be very efficient in terms of quality, bit rate and the complexity. For this reason it has been adopted by many standard CODECs.

In the BMA, a single frame is divided into non-overlapping $M \times N$ blocks. Each block in the current frame is compared with some or all of the possible $M \times N$ blocks in the reference frame (usually previously coded frame) to find the best match. The best match is the one that minimizes the energy of the difference between the current and the matching area.

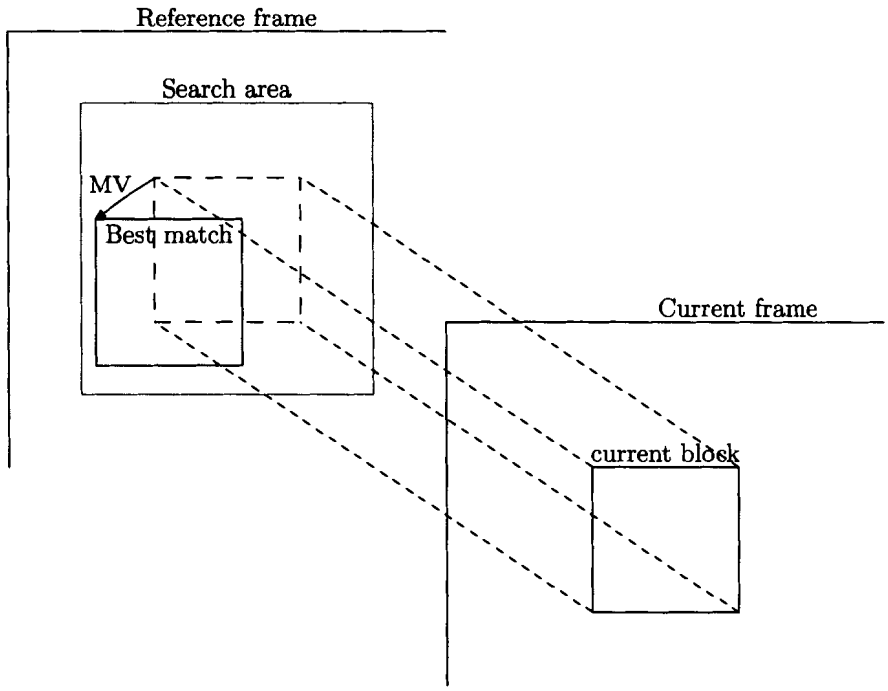

Figure 1.6: Motion estimation

The vector pointing from the original block to the best match is chosen as motion vector (MV). The process of finding the best match is known as motion estimation (Figure 1.6). The residual is computed in motion compensation process by subtracting the best match area from the current region. Then the residual and the motion vectors are coded and transmitted.

The decoder uses the received motion vector to recreate the predictor based on the reference frame and decodes the residual block, adds it to the predictor and reconstructs a version of the original block. The reference frame is usually the previous reconstructed frame. Sometimes there is a significant difference between the reference and current frame, for example when the scene changes. In these cases, using the motion estimation is not efficient. So based on this fact, an encoder may choose intra mode, which is encoding without motion compensation or inter mode, which is encoding with motion compensation for each frame.[6][1]

Sub-pixel Motion Compensation

In many cases, the best match is not located exactly in a position with integer pixel offset in the search area and it may be between pixel positions. In sub-pixel motion estimation, the values of sub-sample positions are created by interpolating the surrounding integer pixels; then the search is performed on both integer-sample and sub-sample positions to find the best match. In general, finer interpolation provides better motion compensation performance at the expense of increased complexity as extra interpolation points have to be computed. In order to decrease the computation, sub-pixel search is usually carried out only around the best integer-sample match.

Choice of References

The most apparent option of reference frame for current frame is the previous frame, since it is expected that these two frames are highly correlated and also the previous frame is available in the encoder and decoder. Forward prediction involves using an older frame as reference for the current frame but sometimes forward prediction has not good performance in certain cases. In these cases the prediction efficiency can be enhanced by using a future frame as reference. Using future frames is known as backward prediction. It needs the encoder to buffer coded frames and encode them out of temporal order, in order that the future frame is encoded before the current frame. In some cases, bidirectional prediction is used. In this method the prediction reference is produced by merging forward and backward references.[5]

Region-based motion compensation

Moving objects have usually irregular shapes and are located at arbitrary positions and are not aligned exactly along block borders. This has led the developers of the video compression standards to seek better performance by motion compensating arbitrary regions of the picture, called object based motion compensation. There are, however, a number of practical complexities like identifying the region boundaries precisely and consistently, signaling (encoding) the contour of the boundary to the decoder and encoding the residual after motion compensation.[1]

1.3.2 Transform Coding

In transform coding, a signal is mapped from one domain to another domain. In image and video coding the image or motion-compensated residual data is the input of transform coding to be converted to another domain where the compression is easier. High similarity between neighboring samples and equal distribution of energy across an image makes it difficult to get rid of spatial redundancy. With an appropriate choice of transform, the data is easier to compress in the transform domain. The transform which is used in compression should have certain properties. It should decorrelate the data so the energy is concentrated into a small number of significant values, be reversible, and be suitable for practical implementation in software and hardware.[5]

The transforms which have been proposed for image and video compression consist of two types: block-based and image-based. The most popular block-based transform is the Discrete Cosine Transform (DCT) [13] and the most common image transform is the Discrete Wavelet Transform (DWT or just Wavelet).

Block-based transforms operate on blocks of $N \times N$ so the image or residual samples are processed in units of a block. In these kinds of transforms, the required memory is low and it is compatible with block-based motion estimation. The problem is the effect of block edges or granularity of the image. Image-based transforms operate on the whole image or frame or a big part of the image known as a tile. These types of transforms have better performance for still image compression but they need higher memory and do not match with block-based motion compensation.[1]

Discrete Cosine Transform

DCT is the most popular transform for image and video coding. There are some reasons for this popularity:

- The energy of the signal is packed in a few coefficients.
- It has fast implementation, forward and inverse.
- It is close to the statistically optimal transform.
- There is minimum residual correlation.
- It can be effectively implemented in software and hardware.

Figure 1.7: Frequency and energy distribution of a DCT block

The energy in the transformed coefficients is concentrated about the low frequency coefficients which are in the top-left corner of the block of the coefficients. Usually the DC coefficient has the highest value and the coefficient values rapidly decrease to the bottom-right of the block, which are the higher-frequency coefficients (Figure 1.7). The DCT coefficients are decorrelated and the energy is compacted in low frequency coefficients, then many small coefficients (high frequency coefficients) can be skipped without considerably influencing image quality.[5][6]

The general equation of $N \times N$ two dimensional DCT and inverse of DCT (IDCT) is defined by the following equations based on [1].

$\mathbf{X}$ is a matrix of samples, $\mathbf{Y}$ is a matrix of coefficients and $\mathbf{A}$ is an $N \times N$ transform matrix. The forward DCT is given by:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathbf{T}} \tag{1.2}$$

and the inverse DCT by:

$$\mathbf{X} = \mathbf{A}^{\mathbf{T}}\mathbf{Y}\mathbf{A} \tag{1.3}$$

The elements of A are:

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N} \tag{1.4}$$

where $C_i = \sqrt{\frac{1}{N}}$ for $i = 0$ and $C_i = \sqrt{\frac{2}{N}}$ for $i > 0$.

Equation 1.2 and equation 1.3 may be written in summation form:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \tag{1.5}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \tag{1.6}$$

DCT is reversible which means that applying the transform followed by its inverse to the data results in the original image data. Therefore the DCT or any other reversible transform does not reduce the redundancy. It is just different representation form of the data (i.e., its spectrum). In order to compress the data, quantization process usually comes after the transform process.[5]

1.3.3 Quantization

The output of a quantizer is a value from a predetermined finite set of permitted numerical values, which is the closest approximation to the input [14]. Quantizers can be classified as uniform and nonuniform. The difference between these two groups is the step-size. In a uniform quantizer , the step-size is constant but in a nonuniform quantizer the step-size is variable. Figure 1.8 shows examples of uniform and nonuniform quantizers.

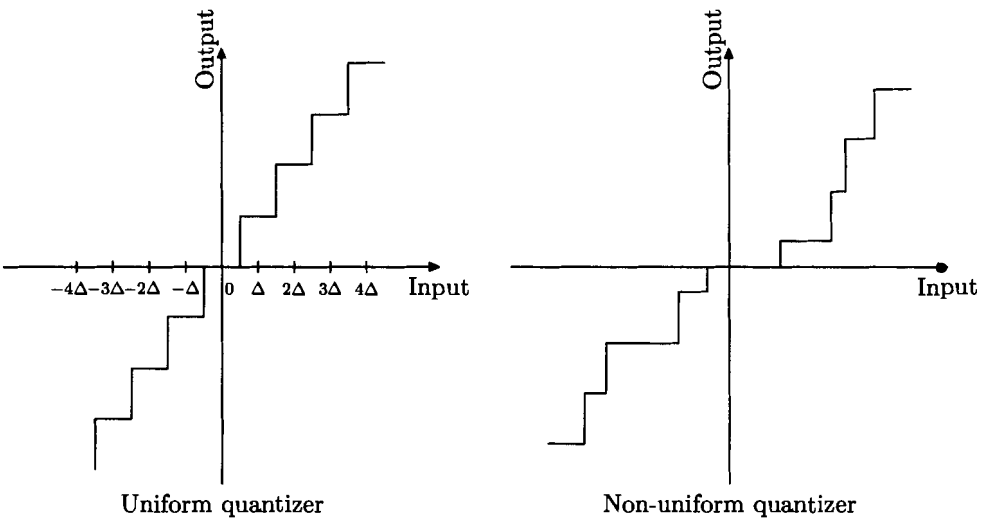

Figure 1.8: Uniform and Non-uniform Quantizers

Using the quantization process after DCT makes possible to compress the data. There are a lot of near-zero coefficients in the transformed values. Quantization discards these coefficients so only significant DCT coefficients are left after quantization.

The number of skipped coefficients can be controlled by quantization step-size (Q). Large step-size causes more zeros in the output of the quantization and in contrast, when a small step-size is used, more coefficients are kept. The level of Q governs the number of zero coefficients, video quality and final compression rate.[5]

1.3.4 Entropy Coding

In video and image coding, the entropy coder converts a series of symbols representing elements of the video sequence into compressed bit stream appropriate for transmission or storage. The compression techniques which are used in entropy coding are general-purpose statistical methods and are not specified for video or image signals.

Quantized transform coefficients, motion vectors and side information consists of headers, synchronization markers, etc., are the data to be encoded by an entropy coder. The method of coding side information depends on the standard. Compression of the motion vectors may be enhanced by predictive coding. Transform coefficients can be represented efficiently with run-level coding. The entropy encoder maps input symbols to a compressed data stream. The compression method is that the entropy coder assigns a small number of bits to symbols which occur frequently and a large number of bits to symbols which occur rarely. The two mostly used entropy coding methods in video coding standards are modified Huffman variable length coding and Arithmetic coding. In Huffman coding each input symbol is represented by a variable length codeword which has integer number of bits. In Arithmetic coding the number of bits can be fractional. Huffman is simpler to implement but Arithmetic has a better compression ratio.[5]

Run-Level Coding

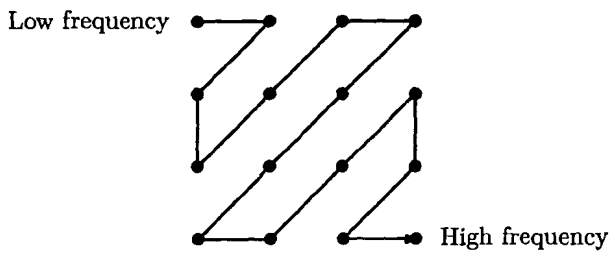

Figure 1.9: Zigzag scan of the quantized coefficients of DCT block

After quantization, there are only a few non-zero coefficients left in low frequencies. The quantized coefficients are reordered into one dimensional array by zigzag scanning (Figure 1.9). The DC coefficient is at the first position of the array, followed by the low frequency coefficients and then high frequency ones. So this zigzag order splits the zero and non-zero values because most of the high frequency coefficients tend to be zero. The one dimensional array is coded as a series of run-level pairs, run is the number of consecutive zeros before the next non-zero value and level is the sign and magnitude of the non-zero coefficient. The run-level pairs are further compressed by entropy coding.[1]

1.3.5 Generic DPCM/DCT CODEC

The main video coding standards since the early 1990s have been based on the same general design of a video codec that exploits motion estimation and compensation described, as predictive coding, transform and entropy coding. The model is often known as a hybrid DPCM/DCT CODEC. Figure 1.10 and figure 1.11 show a block diagram of a generic DPCM/DCT encoder and decoder.

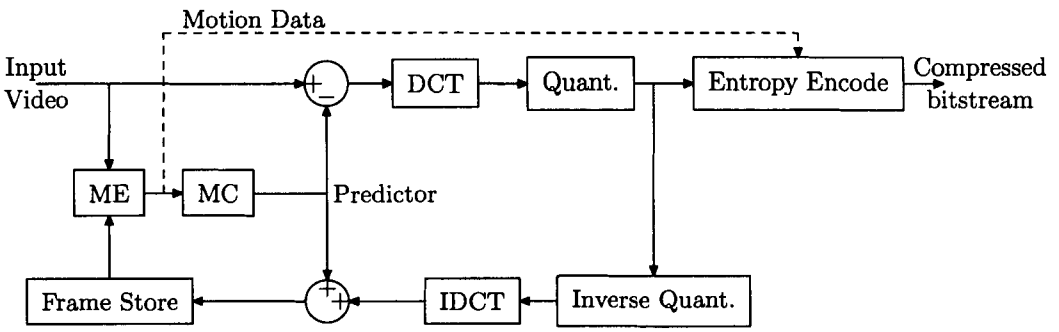

Figure 1.10: DCT/DPCM Encoder

In the encoder, each unit block of current frame is motion estimated and the best match is found in the reference frame, then motion vector is computed. The predictor (best match) is subtracted from the current block to produce the difference, which is transformed, quantized, reordered and run-level coded. Motion vectors found in the motion estimation process, run-level coded coefficients and side information for each unit block are entropy coded to produce the compressed bit stream. Meanwhile the quantized transform coefficients are inverse quantized and inverse transformed and combined with the motion compensated prediction to produce a reconstructed block. The result is not exactly the same as original

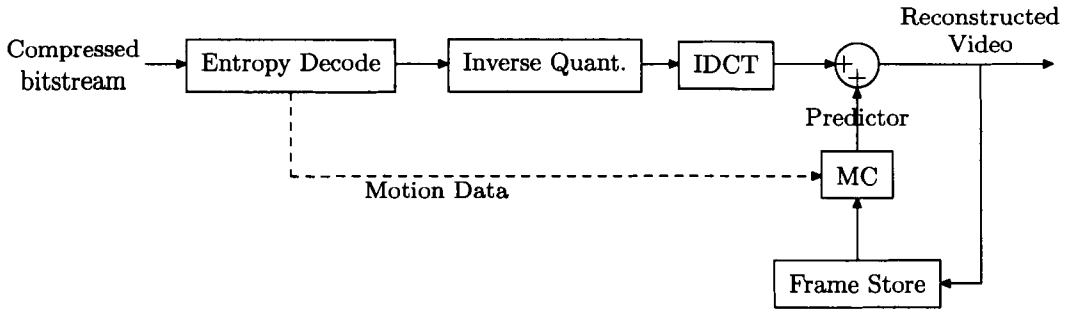

Figure 1.11: DCT/DPCM Decoder

input frame due to the data loss during the quantization; however it is a model of the decoded frame at the decoder, and is saved. This decoded frame is used as reference frame for the next encoding process. It is necessary to use this reconstructed block instead of the original block as a reference in the encoder to make sure that encoder and decoder use an equal reference frame for motion estimation. Hence, it avoids the accumulated error caused by different reference frames at encoder and decoder.

The decoder performs inverse operations, receiving compressed bit stream and entropy decoding it to extract coefficients, motion vectors and side information. Run-level coding, reordering, quantization and DCT are reversed to produce a decoded residual. The decoded motion vector and reference frame are used to find the motion compensated prediction which is added to the decoded residual to reconstruct the input block. As mentioned before, this is not exactly matched with the initial block. The decoded frame is saved for use in motion estimation of the next frame.[2]

Chapter 2

Video Coding Standards

The growing interest in digital video applications has led academics and industry to work together to standardize compression techniques in order to meet the requirements of various applications. ITU-T[1] Video Coding Experts Group (VCEG) and ISO/IEC[2] Motion Picture Experts Group (MPEG) are two organizations that develop video coding standards. MPEG developed MPEG-1 and MPEG-2 standards for coding video and audio, now widely used for communication and storage of digital video. MPEG-4 is the latest standard for audio-visual coding. MPEG-7 [15] and MPEG-21[16] are also the standards which are developed by this group but they are not compression methods. These two standards are considered for multimedia content representation and a generic multimedia framework respectively. VCEG was responsible for the first widely-used video telephony standard, H.261, and its successor, H.263, and started the early development of the H.26L proposal and converted it into an international standard (H.264/MPEG4 Part 10) published by both ISO/IEC and ITU-T.

2.1 MPEG-1

The first MPEG standard was MPEG-1, developed for video storage and playback on CDs. Video, Audio and Systems are three parts of this standard which are developed to support the video , audio compression and creation of a multiplexed bitstream respectively. MPEG-1 Video uses block-based motion compensation, DCT and quantization and is optimized for

[1]International Telecommunication Union - Telecom Standardization

[2]International Standard Organization /International Electrotechnical Commission

a compressed video at bitrates of 1.1 to 1.5 Mbit/s. The video quality of MPEG-1 is not sufficiently better than VHS tapes to encourage consumers to switch to the new technology but it is widely used for PC and web-based storage of compressed video files.[17]

2.2 MPEG-2

In order to improve the quality of MPEG-1, MPEG-2 was developed to support a large potential market, digital broadcasting of compressed television. It is based on MPEG-1 but with quite a few important variations to support the features like efficient coding of interlaced video, a more flexible syntax, some improvements to coding efficiency and a significantly more flexible and powerful systems part of the standard. The final MPEG-2 is basically a fully generic system for audiovisual interactive services. MPEG-2 algorithm can be applied for a wide range of applications, from low bit rate to high bitrate, from low-resolution to high-resolution, and from low picture quality to high picture quality. The scalability of MPEG-2 in terms of bitrates, resolutions, quality levels, and services allow it to be used in broadcast television, cable TV, electronic cinema, DVD-video, video communications, and computer graphic.[18]

2.3 H.261

The ITU-T H.261 coding standard was defined to support video telephony and video conferencing over ISDN circuit-switched networks. These networks operate at multiple of $64Kbit/s$ and the standard supports rates of $p \times 64Kbit/s$, where p varies from 1 to 30, therefore it works between 64 $Kbit/s$ to $2Mbit/s$.The standard uses the hybrid DPCM/DCT model with integer pixel motion compensation.[19]

2.4 H.263

In an effort to enhance the compression performance of H.261, H.263 was developed for PSTN (Public switched telephone network) video telephony aimed at bitrates of less than $64Kbit/s$. Many improvements were performed to the H.261 that resulted in H.263. Some of these include, using different arithmetic and variable length coding, advanced prediction modes, half-pixel motion compensation and so on. The original version of H.263 has four

optional coding modes and each of them has been explained in an Annex to the standard. H.263+ and H.263++ are the other versions which added further modes to the original version to support features such as improved compression efficiency and robust transmission over lossy networks.[20]

2.5 MPEG-4

MPEG-4 standard was developed to increase the capabilities of the earlier standards. MPEG-4 Part 2 (Visual) [21] is more efficient and flexible in comparison with MPEG-2 standard so it enables a much wider range of applications. Part 10 of MPEG-4 is the latest video coding standard developed by VCEG and MPEG together. This new standard is entitled Advanced Video Coding(AVC) and is published jointly as MPEG-4 Part 10 and ITU-T Recommendation H.264. The other parts of MPEG-4 are Systems, Audio, File format etc.

The higher efficiency and flexibility of MPEG-4 stems from exploiting advanced compression algorithms and the provision of a wide set of tools for coding and controlling digital media. MPEG-4 Visual consists of a core video CODEC model together with a number of additional coding tools. The video coding algorithms that form the very low bit rate video core of MPEG-4 Visual are almost identical to the baseline H.263 video coding standard. MPEG-4 Part 2 can support many different applications like digital TV broadcasting, video-conferencing, video storage, streaming video over Internet and mobile channels, high-quality video editing and distribution for the studio production environment, computer generated graphics and animated human face and bodies.

2.5.1 Visual Coding Tools

The most important shift in MPEG-4 is object-based or content-based coding. In this method a video scene is divided to set of foreground and background objects instead of just a sequence of rectangular frames. An MPEG-4 visual scene, which is a sequence of video frames, is made up of a number of Video Objects (VO). The equivalent of a frame in video object term is a video object plane (VOP) which is the snapshot of VO at a single instant in time. A video object can occupy the entire frame size or rectangular area or even arbitrary shaped object, which is coded independently by using motion compensation, shape coding and texture coding.

Shape Coding

Shape coding tools are used to characterize the borders of arbitrarily shaped objects by using extra information, explained in a map of the same resolution as luma[3] signals. There are two types of shape information: binary and grey-scale. In binary shape coding the pixels that are internal to the VOP, described as opaque and the pixels that are out of the VOP are transparent; therefore, two states are possible for each pixel. Gray-scale information is more complex and requires more bits to code. In this method the transparency of each pixel is identified by a 8-bit number, therefore semi-trasnparent VOPs and overlapped objects are possible.[22]

The binary shape values of each pixel are predictive coded and the texture information of the opaque pixels is coded as described later. Block-based DCT and motion compensation are used to compress the values of gray scale shape information.

Motion Compensation

For macroblocks that lie fully within the current and reference VOP, block-based motion compensation is used. For macroblocks along the border of the VOP this process is modified. In the reference VOP, pixels in the search area are padded based on the pixels along the edge of the VOP. The macroblock in the current VOP is matched with the search area using block matching; however, the difference value is only computed for those pixels that are within the VOP.

Texture Coding

The coding of pixels or motion compensated residual values is called texture coding, which includes DCT, quantization and variable-length coding, in a similar way to H.263. An alternative transform which is used to code static texture efficiently is the wavelet transform. The static texture is texture data which does not change fast.

For a boundary macroblock that has both opaque and transparent pixels, the transparent pixels are filled with some values, and then the boundary macroblock is coded in the same way as the other macroblocks. Filling the transparent pixels is different in inter coding and intra coding. In inter coding, motion compensated values are the input of the texture

[3]Luminance

coding stage and the transparent positions are filled with zeros. In intra coding, the input is original pixel data and the transparent positions are filled by extrapolating the pixel values along the boundary of the VOP.

Error Resilience

Several techniques may be used to enhance coding robustness. First of all, unique markers are inserted into the bitstream so that the decoder can stop decoding process until the next marker, when an error is found. This is called resynchronization. In MPEG-4, texture data and motion data may be decoded separately by data partitioning, therefore the different kinds of data are independent. Header extension is the other way to protect the data from errors. In this technique redundant copies of header information are added at intervals in the bitstream to be used for data recovery in case of losing an important header due to an error. Finally, reversible VLCs restrict the propagation of error by allowing forward and backward decoding of the bitstream.

Scalability

Scalability allows the decoder to decode selectively only parts of the compressed bitstream. The coded stream includes different layers: one base layer and two or more enhancement layers. Decoding just the base layer gives basic quality sequence, whereas decoding all layers gives high quality sequence.

Synthetic Visual Scene Coding

The concept of hybrid synthetic and natural video objects for visual communication is introduced by MPEG-4. Based on this idea, a combination of tools from the video coding methods designed for coding of real world or natural video material, and tools from the 2D and 3D animation methods designed for rendering synthetic or computer-generated visual scenes, can be used depending on the application. 2D and 3D mesh coding and Face and body model coding are tools which offer the potential for fundamental improvements in video coding performance and flexibility; however their application is currently limited due to the high processing resources needed.[23]

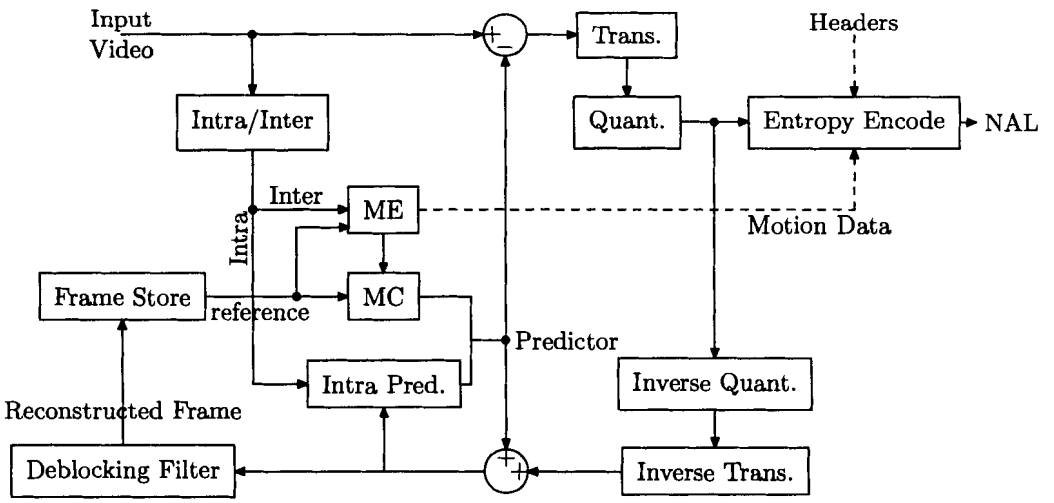

Figure 2.1: H.264 Encoder

2.6 H.264/MPEG-4 Part 10

H.264 is the latest video coding standard which is developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). In this standard, compression and rate-distortion efficiency has been improved significantly, and a network-friendly video representation is provided which is useful in conversational (video telephony) and non-conversational (storage, broadcast or streaming) applications. The H.264/AVC design includes a Video Coding Layer (VCL), which processes the original video stream and generates compressed information by removing temporal and spatial redundancy, and a Network Abstraction Layer (NAL), which formats the VCL representation of the video and provides header information in a suitable way for transmitting or storage.

2.6.1 Video Coding Layer

The video coding layer is based on the generic block-based motion-compensated hybrid video CODEC. The block diagram of H.264 encoder is shown in figure 2.1. Each input video frame is split into macroblocks and each macroblock based on its type is coded in inter or intra mode. Inter or Intra coding discard some parts of redundancy and produces the prediction block which is subtracted from the original macroblock. Then, the transform coding, scaling and entropy coding are applied consecutively to the resulted difference macroblock. Other

information such as motion data, quantization parameters and side information are also entropy coded and sent to the NAL. In the decoder, the quantized samples are rescaled and after inverse transform are added to the motion-compensated or intra prediction block. Then the deblocking filter is applied on each macroblock to produce the reconsructed macroblocks which are stored for future prediction. The functions of the decoder exist in the encoder in order to recreate the reconstructed picture and use it as the reference. Thus, we can make sure that the reference pictures are the same in the encoder and decoder.

Picture Structure

Each picture of a video, which can either be a frame or a field, is divided into macroblocks that contain a 16 × 16 pixels luma component and 8 × 8 pixels of each of two chroma[4] components utilizing 4:2:0 sampling. This partitioning into macroblocks has been adopted into all previous ITU-T and ISO/IEC video coding standards since H.261.

The macroblocks are arranged in slices, which generally represent subsets of a given picture that can be decoded independently. H.264 supports five different kinds of slices.

- *I* slice which contains only *I* macroblocks. These macroblocks are coded without prediction from other pictures within the video sequence. *I* macroblocks are predicted from decoded samples in the current slice using intra coding.
- *P* slice may contain *I* and *P* macroblocks. *P* macroblocks are predicted from prior-coded picture(s) using inter coding.
- *B* slice may contain *B* and *I* macroblocks. *B* macroblocks are like *P* macroblocks, but the difference is in the number of motion compensated prediction signals per prediction block. *B* macroblocks can be coded using inter prediction with two motion-compensated prediction signals per prediction block but this number is one for *P* macroblocks.

The above three coding types are very similar to those in previous standards with the exception of the use of reference pictures which is described later. The following two kinds of slices are specific in H.264.

SP (switching P) and *SI* (switching I) are specified for switching between streams which are coded at different bitrates. These two kinds will be explained in section 2.6.7.[24]

[4]Chrominance

2.6.2 Intra and Inter Coding

Intra Mode

In intra mode, the prediction is based on previously encoded and reconstructed neighbor blocks in the current slice. Intra-4×4 and Intra-16×16 together with chroma prediction, and I-PCM prediction modes are different kinds of intra coding. Intra-4 × 4 is suitable for coding of areas with significant details. There are nine optional prediction modes for each 4 × 4 luma block in this method. Four modes are available for Intra-16 × 16. Intra-16 × 16 is more appropriate for smooth image areas without too much detail. Each 8 × 8 chroma block has four types of prediction modes which are very similar to the 16 × 16 luma prediction modes. Both chroma components always use the same prediction mode. The encoder chooses the prediction mode for each block that minimizes the difference between prediction and the original block. A further intra coding mode, I-PCM, let the encoder bypass the prediction and transform coding processes and send the samples directly. In some special cases like inconsistent and irregular image content and very low quantizer parameters, this mode may be more efficient than the usual process of intra prediction, transformation, quantization and entropy coding. In contrast to some previous video coding standards such as H.263+ and MPEG-4 Visual, where intra prediction has been conducted in the transform domain, in the H.264 intra prediction is always conducted in the spatial domain.[25]

Inter Mode

In inter mode, the prediction block is created from the reference pictures by motion estimation.

Reference Pictures

The encoder and the decoder, each maintain one or two lists of reference pictures, containing pictures that have been encoded and decoded (occurring before or after the current picture in display order). Inter coded macroblocks in P slices are predicted from pictures in a single list so there is one reference picture for each macroblock. Inter coded macroblocks in a B slice may be predicted from two lists. Thus, there are two reference pictures for each macroblock.

Variable Block-Size Motion Compensation

H.264 supports more flexibility in the selection of motion compensation block sizes and shapes than any previous standards, with a minimum luma motion compensation block size as small as 4×4. The luminance component of each macroblock may be divided in four ways and motion compensated either as one 16×16 macroblock partition, two 16×8 partitions, two 8×16 partitions or four 8×8 partitions. If the 8×8 mode is selected, each of the four 8×8 sub-macroblocks may be motion compensated itself or split up in two 4×8 partitions, two 8×4 partitions or four 4×4 sub-macroblock partitions (Figure 2.2).

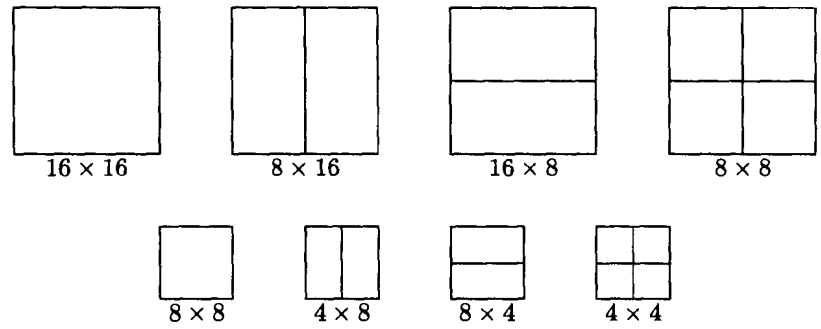

Figure 2.2: Macroblock and Sub-macroblock partitions

A separate motion vector is needed for each partition so a maximum of sixteen motion vectors may be transmitted for one P slice. Each motion vector and also the choice of partitions should be encoded and transmitted. Selecting a large partition size means that a small number of bits are required to represent the choice of partition mode and motion vectors, but the motion compensated residual contains a significant amount of energy in images with high detail relative to choosing small partition sizes.
Each chroma component in a macroblock has half the horizontal and vertical resolution of the luma component and is divided in the same way as the luma block. The horizontal and vertical components of each motion vector of chroma block are half of the components of the motion vector of the luma block.[1]

Multiple Reference Picture Motion Compensation

H.264 supports multi-picture motion-compensated prediction. It means that more than one previously coded picture can be used as a reference for motion estimation. H.263++ used this enhanced reference picture selection technique to facilitate efficient coding by allowing

an encoder to choose the reference among a larger number of pictures that have been decoded and stored. Multi-frame motion-compensated prediction requires both encoder and decoder to store the reference pictures used for inter prediction in a multi-picture buffer. The choice of reference picture is transmitted to the decoder by an index number.[26][27]

Motion Vector Prediction

Encoding a motion vector for each partition can cost a large number of bits, particularly if small sizes are selected. For this reason, the predictive coding is used for coding motion vectors. This method can be used since motion vectors for neighboring partitions are often highly correlated so each motion vector can be predicted from vectors of nearby, previously coded partitions. A predicted motion vector (MVp) is formed based on already calculated motion vectors. The difference between the current vector (MVD) and the predicted vector, is encoded and transmitted. The prediction method is based on the partition size and availability of nearby vectors. At the decoder, the predicted vector is generated by the same method and added to the decoded vector difference to produce the motion vector which is used in reconstruction.[1]

Quarter-sample-accurate Motion Compensation

H.264 employs sub-pixel motion compensation with the accuracy of a quarter luma sample. H.263 enables half-pixel motion vector accuracy. In cases where the motion vector points to an integer-pixel position, the prediction samples are the matching samples of the reference picture; if not, they are found by using interpolation at the sub-sample positions. The prediction values at half-sample positions are obtained by applying a one-dimensional 6-tap FIR filter. Prediction values at quarter-sample positions are produced by averaging samples at the integer and half sample positions. For the chroma components, the prediction values are generated by bi-linear interpolation.[28]

Inter coding in B slices

The concept of B slices is generalized in H.264. The significant difference between B and P slice is that weighted average of two distinct motion-compensated prediction values can be used for generating the prediction block in B slices. B slices employ two lists of references, list0 and list 1. Four different kinds of prediction for inter coding are supported in B slices:

list 0, list 1, bi-predictive and direct mode. Different prediction modes may be selected for each partition; if the 8×8 partition size is used, the chosen mode for each 8×8 partition is applied to all sub-partitions within that partition. In the bi-predictive mode, each sample of the prediction block is formed by averaging (or weighted averaging) of the list 0 and list 1 prediction samples. In the direct prediction mode, motion vector is not transmitted and the decoder calculates list 0 and list 1 vectors based on previously coded vectors and uses these to perform bi-predictive motion compensation of the decoded residual samples.[29] B slices are further investigated in [30].

SKIP mode

A P-slice or B-slice macroblock can also be coded in SKIP mode. For this mode, residual block, motion vector and reference index are not transmitted. A skipped macroblock in a B slice is reconstructed at the decoder using direct prediction and in a P slice the decoder calculates a vector for the skipped macroblock and reconstructs the macroblock using motion-compensated prediction from the first reference picture in list 0. In the case of a skipped macroblocks, there is no decoded vector difference and a motion-compensated macroblock is produced using MVp as the motion vector.

2.6.3 In-loop Deblocking Filter

One particular characteristic of block-based coding is visible block structures. Block borders are usually reconstructed with less precision than inner samples and blocking is considered to be one of the most obvious artifacts. For this reason, H.264 uses an adaptive in-loop deblocking filter to decrease the granularity without much influencing the sharpness of the content. Therefore, the subjective quality is significantly enhanced. The filtered image is used for motion-compensated prediction of future frames and this can improve compression performance because the filtered image is often more similar to the original frame than a blocky, unfiltered image.[31]

2.6.4 Transform and Quantization

H.264 employs transform coding like the previous video coding standards. Instead of the 8×8 DCT transform which is used in previous standards, in H.264 the transformation is

applied to 4×4 blocks and instead of discrete cosine transform (DCT), a separable integer transform with similar properties is used. The transform matrix is given as:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

This transformation is an approximation to the 4×4 DCT but its result is not identical to the 4×4 DCT. Since the inverse transform is defined by exact integer operations, inverse-transform mismatches are avoided, unlike unmodified DCT. In the context of the H.264 CODEC, the approximate transform has almost identical compression performance to the DCT.

Another transform which is used in H.264 is the Hadamard transform. This transform is applied to 4×4 block of luma DC coefficients in intra macroblock predicted in 16×16 mode and to 2×2 chroma DC coefficients in any macroblock. This additional stage of transform is useful for obtaining more accurate result in smooth areas since in that circumstance the reconstruction precision is proportional to the inverse of the one dimensional size of the transform. Therefore, for a very smooth area, the reconstruction error with a transform covering the complete 8×8 block is halved compared to using only 4×4 transform. However, there are several reasons for using a small size transform. The residual signal has less spatial correlation because of an improved prediction process, hence the 4×4 transform is as efficient as a larger transform in removing statistical correlation . Also the smaller transform has visual advantages resulting in less noise around edges. Finally, the smaller transform needs less computations and processing power.[32][33]

H.264 uses a scalar quantizer. Avoiding division and floating point arithmetic makes the mechanisms of the forward and inverse quantizer complicated. A total of 52 values of quantizer step-size are supported, indexed by a Quantization Parameter (QP). When QP is incremented by six, the quantization step-size is doubled. The wide range of quantizer step-sizes enables the encoder to manage the tradeoff between bit rate and the quality. The values of QP can be different for luma and chroma samples.[1]

2.6.5 Entropy Coding

H.264 supports two techniques of entropy coding. The default mode employs context-adaptive variable length coding (CAVLC) method for coding the residual data and Exp-Golomb codes for all other syntax elements such as headers, macroblock type, quantizer parameter, reference frame index and motion vector. Exp-Golomb codes (Exponential Golomb codes) are variable length codes with very simple and regular construction and decoding process. Instead of designing a different VLC table for each element, only the mapping to the single codeword table is customized depending on the data statistics [34]. CAVLC is more sophisticated and designed to take advantage of several characteristics of quantized 4×4 blocks. In this method, VLC tables for different syntax elements are switched, according to prior-transmitted syntax elements. Since the VLC tables are well designed to match the corresponding conditioned statistics, the entropy coding performance is enhanced relative to method using just a single VLC table.[35]

The other option for entropy coding which is available in H.264 to improve the efficiency of entropy coding is Context-Adaptive Binary Arithmetic Coding (CABAC). CABAC achieves good compression performance through context modeling, adaptive coding and arithmetic doing. In this method, probability models are selected for each syntax element based on the statistics of recently-coded data symbols. The selected context model is updated according to the actual coded value. At last, the arithmetic coding allows the assigning of a non-integer number of bits to each symbol. In H.264, the arithmetic coding core engine and its related probability estimation are specified as multiplication-free low-complexity methods, using only shifts and table look-ups. In comparison with CAVLC, CABAC typically presents a reduction in bitrate of between 10-15%, in coding TV signals at the same quality.[36]

2.6.6 Special Features

Adaptive frame/field Coding

Pictures in H.264 can be coded in frame or field mode. Also the choice of field or frame coding may be specified at the macroblock level which is macroblock-adaptive frame/field coding mode. This technique is useful when each image has some moving and non-moving regions since it is typically more efficient to code the non-moving regions in frame mode and the moving regions in the field mode. The choice between the coding options can be made

adaptively for each frame in a video stream.[25]

SP/SI switching pictures

SP and SI slices are specially-coded slices that make switching between video sequences and random access for video decoders efficient. SP-slices are more appropriate to support switching between similar coded sequences, for instance, the same source sequences at different bitrates. The SI-slices are usually used to switch from one sequence to a completely different sequence. In this case SP-slices are not efficient since there is no correlation between the two streams.[37]

Flexible macroblock ordering

H.264 gives this ability to divide the picture into areas called slice groups. A slice group consists of the macroblocks and may contain one or more slices and is independently decodable. Macroblock to slice group map represents to which slice group each macroblock belongs. By using FMO, a picture can be split into many macroblock scanning patterns. Figure 2.3 shows two examples for subdivision of a frame into slice groups. When FMO is not being used we can say the whole picture consists of a single slice group. Efficient using of FMO can improve robustness to data losses by managing the spatial relationship between the slice groups. FMO can also be used for other purposes as well.[38]

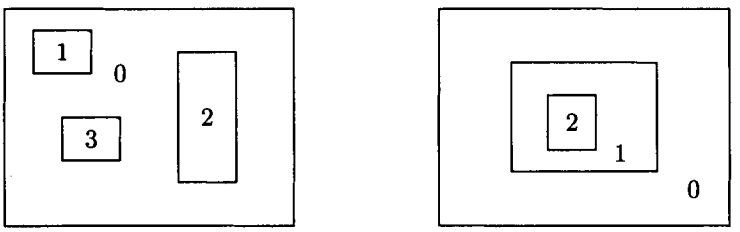

Figure 2.3: Subdivision of a frame into slice groups

Data Partitioning

The coded data of one slice is separated to three different partitions, each containing a subset of the coded slice. The header information are placed in Partition A, compressed residual data for I and SI macroblocks are placed in Partition B and coded residual data for

inter coded macroblocks are formed in Partition C. Each partition can be transmitted and decoded independently. Partition A is very sensitive to transmission error because it is too difficult or even impossible to decode the bitstream if the data of partition A is lost.

2.6.7 Network Abstraction Layer

The Network Abstraction Layer (NAL) is designed to supply network friendliness to allow different systems to fit VCL to their specifications. A short description of some key concepts of the NAL is given below whereas a more detailed description is provided in [38] and [39].

The video data is formed into NAL units, each of which is in fact a packet that includes an integer number of bytes. The header byte, which is the first byte of each NAL unit, indicates the type of data in the NAL unit, and the left over bytes contain payload data of the type designated by the header. The NAL unit arrangement denotes a general format for use in both packet-oriented and bitstream-oriented transmission systems. The NAL units can be VCL or non-VCL units. The VCL NAL units includes the coded video stream, and the non-VCL NAL units contain any extra information such as parameter sets and supplemental enhancement information (timing information and other supplemental data that may increase usability of the decoded video signal, but are not needed for decoding the values of the samples in the video pictures.). A parameter set contains information which is not often changed and offers the decoding of a large number of VCL NAL units. Parameter sets which apply to a sequence of successive coded pictures are called sequence parameter sets and the parameter sets which apply to the decoding of one or more individual pictures within a coded video sequence are called picture parameter sets. Parameter sets can be sent well ahead of the VCL NAL units that they apply to, and can be repeated to provide robustness against data loss. A sequence of NAL units which can represent one picture after decoding is called an access unit and a series of access units which uses only one sequence parameter set is referred as a coded video sequence.

2.6.8 Profiles and Levels

Three different profiles are defined in H.264 and each of these profiles supports special set of functions. The Baseline Profile supports I and P slices, flexible macroblock ordering and entropy coding with CAVLC. Potential application of this profile consists of video telephony, videoconferencing and wireless communication. The Main Profile supports all

available functions of the baseline profile and also interlaced video, B-slices, inter coding using weighted prediction and entropy coding using CABAC. Applications of this profile are television broadcasting and video storage. The Extended profile is similar to Main profile with some differences. It does not support interlace video and CABAC, but adds modes to enable efficient switching between streams (SP and SI slices) and improved error resilience (Data partitioning). This profile may be mainly useful for streaming media applications. However, each profile has enough flexibility to support a wide range of applications and these examples of applications should not be considered definitive.[1]

Chapter 3

Context-Based Complexity Reduction

The most important performance constraints of CODEC algorithms are the bitrate of the coded video bitstream, the video quality and the complexity of algorithms used in the CODEC. Changing coding parameters, adding optional coding modes, and choosing various coding algorithms affect the output of video encoders, resulting in different levels of visual quality, bitrate and computation. The exact relationship between bitrate, visual quality and computational complexity varies depending on the characteristics of the video sequence. In general, good quality video requires a complicated coding scheme and/or high bitrate.

In many applications such as wireless communication, the bitrate of a coded video sequence is limited by the characteristics of transmission channels. Rate-distortion optimization attempts to maximize image quality subject to transmission bitrate constraints. The best optimization performance comes at the expense of very high computational complexity. For portable wireless terminals, the battery consumption is an important factor for the algorithm complexity. In many application scenarios such as real-time and power-limited applications, video quality is restricted by the available computational resources as well as the available bitrate. In these cases, there is a need to reduce the complexity to meet the constraints of coding time and processing power.

The complexity reduction, however, is achieved at the expense of lower compression gain and/or lower visual quality. In order to minimize its effect on the bitrate and end-users' perception of picture quality, the complexity reduction should be applied only to the

less significant areas of each frame in the video sequence. To achieve this, we need to use context-based coding.

3.1 Context-based Coding

Context-based coding is a video coding scheme that treats the areas of interest with higher priority and codes these areas at a higher quality than the less significant background scene. Therefore, recognizing the viewer's region of interest from the less important background is the first step for context-based coding. Each frame of the source sequence is segmented into two non-overlapping areas of varying significance. The most significant regions are segmented out and classified to be the foreground, leaving the remaining areas as the background. The foreground and background regions are then coded using the same encoder but with different coding parameters. Transmission of the information of region classification depends on the source coding method and the syntax of the coded video. Sometimes the decoder needs to know the clear location of foreground and background regions to decode such a video stream. In our method, this knowledge is not required at the decoder.

In deciding the relevance and significance of each part of the frame, prior knowledge of the context is taken into consideration. For example, in applications such as videophone or video conferencing, each frame typically consists of a head-and-shoulders view of a speaker in front of a simple or complex background scene. Hence, in such case, the face of the speaker is usually more important to the viewer than the rest. Therefore, the face is considered as the foreground region of the input image [40].

The basic concepts of foreground/background coding schemes and related methods are addressed in the literature as follows: In [41] FB coding[1] has been applied to H.263 and in [42] and [43] the implementation of FB coding scheme on the H.261 framework has been discussed. In these three papers foreground and background are encoded with different quantization step-sizes to improve the subjective image quality of videophone sequence. [44] and [45] have proposed rate control schemes that adjust the quantization level based on the content classification.

[1]Foreground/Background Coding

3.2 CODEC Complexity Reduction

In video communication systems, the total power consumed in the mobile device is dominated by the transmission power and processing power. The processing power includes the source coding and channel coding. The transmission power depends on the transmission bit energy and total bitrate [46]. Therefore, reducing the bitrate decreases the transmission power. However, this reduction of bitrate results in inferior visual quality. If we don't want to degrade the picture quality, we must use more complex algorithms for compression. The more complex the CODEC is, the more time- and power-consuming it will be.

CODEC complexity reduction is desired for applications in which time and power are important. It is obvious that most of these techniques have an impact on bitrate or/and visual quality. There is a trade-off between complexity of algorithm, visual quality of reconstructed video sequence and the bitrate of the coded bitstream.

Various strategies that may help reduce the computational complexity of CODEC are as follows:

- Motion estimation search window: the computational complexity of a motion estimation algorithm typically depends on the search size and/or number of comparison steps. Consequently, the complexity can be adjusted by increasing or decreasing the search area size.

- Frame skipping: skipping frames is a simple way to reduce the processing utilization; however, it may lead to variable frame rate as the available resources change and this can be distracting to the viewer. Also when the frame rate is low because of frame skipping, the difference between two consecutive frames is large and more data needs to be coded.

- Number of references: decreasing the number of reference pictures will decrease the complexity. This is because the search process over the reference pictures will be shorter.

- Pruned DCT: the DCT process generates coefficients in frequency domain. In a typical image block, low-frequency coefficients are non-zero and high-frequency coefficients are zero or very small. Hence, it is possible to discard some of the higher frequency components without losing too much picture quality. The smaller the residual block,

the more zero coefficients result from the DCT. Thus, only some of the coefficients needs to be calculated and the other coefficients are set to zero. A pruned DCT algorithm only computes a subset of the 8×8 DCT coefficients in order to reduce the computational overhead of the DCT. For example, the full 8×8 may be reduced to a 4×4 or 2×2 DCT. However, applying a pruned DCT to all blocks means that the small (but significant) number of high-frequency coefficients is lost and this can have a very visible impact on image quality.[47]

- Zero testing in IDCT: each row or column of eight DCT coefficients is tested for zeros. If the seven highest coefficients are all zero, then the row or column will contain a uniform value which is the DC coefficient after the inverse DCT. In this case, the inverse DCT may be skipped and all samples set to the DC value. This may be exploited to reduce the complexity of the inverse DCT which must be computed in both the encoder and the decoder.[5]

Some of these techniques have been used in our experiments.

3.3 Implementation and Results

Two QCIF (176×144 pixels) video streams, named 'Foreman' and 'Claire', were selected for tests [48]. About 100 frames of each were encoded for experimentation and benchmarking. In 'Claire', the background is almost constant and the person's change is slow whereas the background in 'Foreman' has details and varies with time and the person moves relatively fast . The region of interest in each frame of the sequence is identified and subsequently the frames are segmented into background and foreground regions.

In our experiments the complexity is measured in terms of the computational time of the processor. Processing time is a factor which is related to the complexity of the algorithm. In this thesis we have used time consumption as a sign for complexity.

3.3.1 Segmentation

There are some methods for face tracking in the literature. In this thesis, these different methods are not investigated. Wayne Huang and Susan Chiu have implemented a method

for face segmentation [2]. Their implementation, but with necessary changes in input and output format (to be compatible with H.264 software), has been used in this thesis. The algorithm employed here for face recognition is based on the human skin color model to distinguish facial areas from the background. The color distinction is based on the intensity values of luminance and chrominance of human skins. Research and experiments have shown that the intensity value of human skin tends to lie in a relatively small range for each of the Y, Cb, and Cr component [49]. This range has been used as the upper and lower threshold value for each component in this project to initially identify an object as a face or a non-face region. Thus, Y, Cb and Cr components of an image are read individually to compare with its corresponding upper and lower thresholds, which essentially are three independent threshold-filtering processes. Then the results from each of the three components are combined to obtain the final areas of interest. This algorithm is simple and yet effective.

Segmentation is applied to each frame of the input video stream and divides each frame into two regions. Segmentation is performed at the macroblock level and each macroblock (16×16 block) is defined as foreground or background. The result is an array whose elements are zero for background and one for foreground. Video streams with QCIF formats (176×144 for luminance component) have 99 macroblocks in luminance component of one frame, so the result for each frame is an array with 99 elements.

Figures 3.1 and 3.2 show one frame of 'Foreman' and 'Claire' video streams. In these figures the original frame and the distinguished region of interest in each frame are shown.

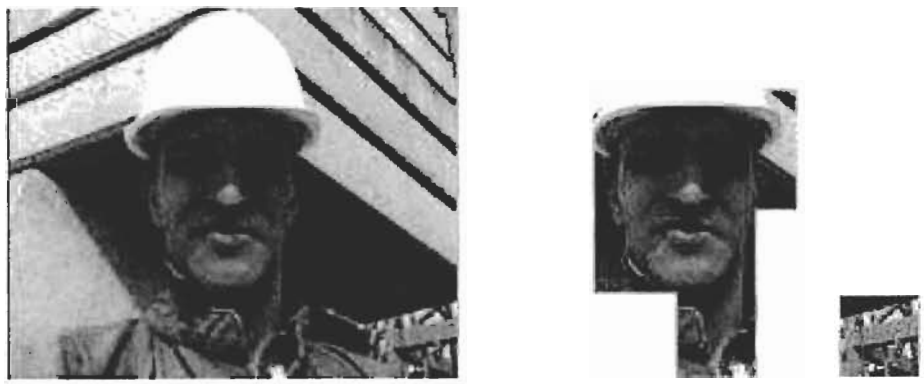

Figure 3.1: Original and segmented versions of one frame of 'Foreman' sequence

[2]Two undergraduate students at SFU who did this project for their ENSC 494 course, 2004-2.

Figure 3.2: Original and segmented versions of one frame of 'Claire' sequence

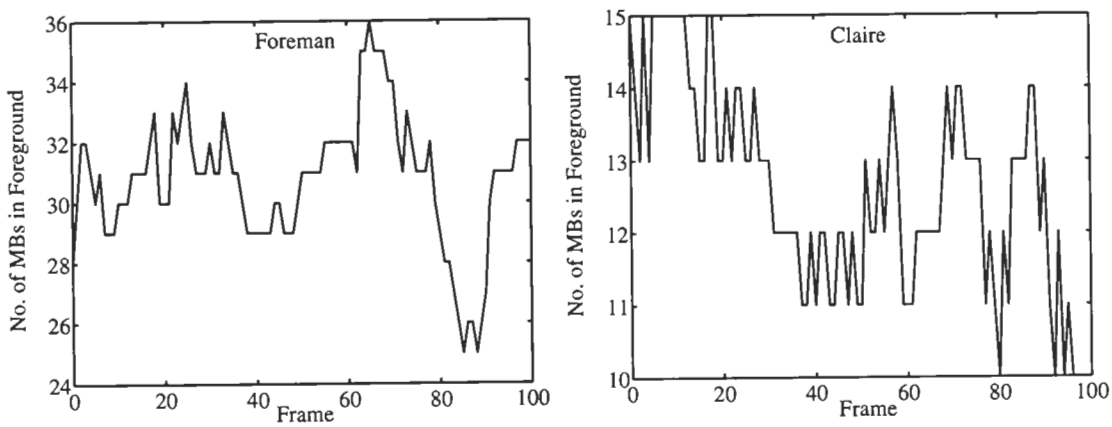

Figure 3.3: Number of macroblocks in the region of interest

The region of interest in figure 3.1 is not just the head and shoulder of the person in the picture, but includes an area at the lower right, as shown. This lower right area contains not just complex features, but it is initially selected because it contains skin-colors. This demonstrates a limitation of the threshold filtering process.

In a video sequence, the region of interest can be changed for different frames so the number of macroblocks in the foreground is changed from one frame to another frame. This variation may be caused by motion of the face, dissimilar distances from the camera, or wrong detection of the region of interest. Figure 3.3 shows this number versus frame number for the two video sequences. In this figure, 100 frames of each video sequence are divided into the foreground and the background. In the 'Claire' sequence, the person's face is much smaller than the person's face in the 'Foreman', so the number of foreground

macroblocks in the first case (about 13) is less than this number (about 31) in the second case.

3.3.2 Processor Utilization

Video coding algorithms consist of different processes. Some of these are more complex and time-consuming than the others. For reducing the complexity of the CODEC, it is reasonable to focus our efforts on more complex processes. Figure 3.4 shows the approximate processor utilization of each operational block of the encoder of H.264. This data is from C simulation of the H.264 software with 'Foreman' and 'Claire'. The processor utilization did not change much with other video contents. The operations related to motion estimation and rate-distortion optimization are the most complex followed by forward and inverse transform and quantization functions.

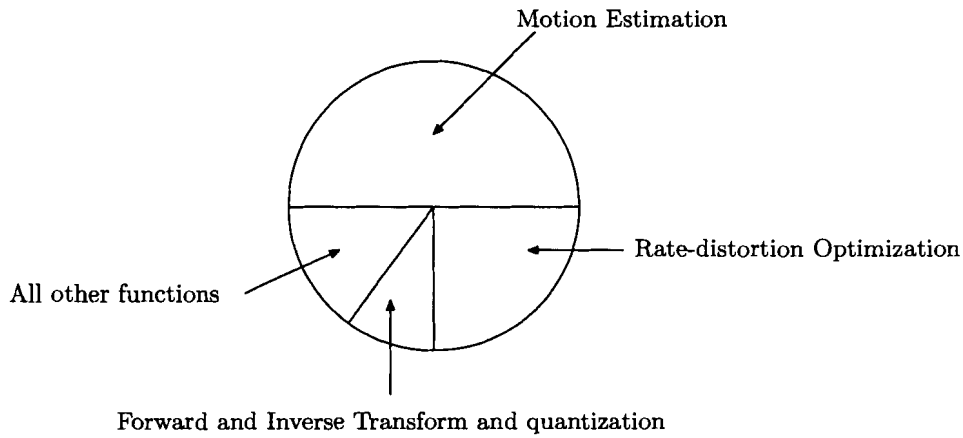

Figure 3.4: Processor Utilization

3.3.3 Reduction of Processing Time

For reducing time consumption the motion vectors of the background macroblocks are set to zero to bypass the search for best match (motion estimation process). Thus, the residual has more energy and needs more bits to be coded. To prevent this latter effect, which causes higher bitrate, the residual is set to zero and motion compensation is avoided for background. Avoiding motion estimation and compensation for background macroblocks makes them constant and decreases the PSNR of the background. To increase the quality

of background and follow its general variations, the background is refreshed by intra coding. If the PSNR of the background goes below a certain threshold, the whole frame is coded in intra mode (I-frame). Using trial and error this threshold is chosen to be 20dB for 'Foreman' and 30dB for 'Claire'. The background in 'Claire' is quite constant; therefore, the average PSNR of the background in this case, is higher than that of the sequences with varying background so the threshold should be higher. Selecting the threshold for one video depends on desired bitrate and quality of the background. If the threshold decreases, the bitrate and the PSNR of the background will increase. We could have programmed the encoder to produce periodically intra frames as is the norm, but our proposed scheme results in a lower average bitrate for the same picture quality since sometimes the pictures change more rapidly and more intra frames are needed.

To control the bitrate, conventional methods update the quantization step-size frame by frame in order to maintain the desired bitrate. This results in equal distortion for both foreground and background. As an improvement, a constant quantizer step-size is employed for all the frames in the video stream. The quantization parameter is decided at the macroblock level. A constant, small step-size is chosen for the foreground and a constant large step-size is used for the background and in this way the PSNR of the region of interest will not be affected by large step-sizes in regular methods.

The number of reference frames and size of the search area are reduced in our experiments, for the entire frame (foreground and background). Rate-distortion optimization is applied for the foreground but not for the background in order to reduce the high computational complexity of this process. While implementing Pruned DCT we noted that in H.264 the transformation is applied to 4×4 blocks. Therefore, each 16×16 block has sixteen 4×4 blocks for transform. Forward transform is skipped on all sixteen sub-blocks and just four of the low-frequency blocks are picked for this process, instead of sixteen.

We have enhanced the implementation of JM8.1a version of H.264 encoder [50] to accommodate the aforementioned schemes. Our experiments use the Baseline profile of H.264. The implementation has been benchmarked on an AMD Athlon, 2500 MHz processor.

The results are shown in tables 3.1 and 3.2 and figures 3.5 to 3.14. The tables show the bitrate[3], average PSNR for the background and foreground of luminance component[4], and

[3]Total number of bits for 100 frames multiplied by frame rate (30 frames/sec) is defined as bitrate expressed as Kbits/sec.

[4]The luminance component is more important because of human visual system.

Characteristics	Bit rate (Kbps)	PSNRB (dB)	PSNRF (dB)	Time (Sec.)
Regular coding with rate control	109.08	35.24	35.02	203
FB Coding without intra frame QP = 28	84.54	15.42	35.32	93
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=1, Search range=4	109.81	23.15	35.54	93
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=5, Search range=4	108.7	23.15	35.66	119
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=1, Search range=30	109.52	23.15	35.54	136
Transform complexity reduction in the background	109.13	34.13	34.94	195

Table 3.1: Results for 'Foreman' sequence

Characteristics	Bit rate (Kbps)	PSNRB (dB)	PSNRF (dB)	Time (Sec.)
Regular coding with rate control	34.29	41.39	35.09	150
FB Coding without intra frame QP = 28	29.67	31.40	34.68	52
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=1, Search range=4	34.82	31.76	34.82	52
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=5, Search range=4	34.97	31.75	34.82	67
FB Coding with intra frame, QPF=28 QPB=42, Ref. No.=1, Search range=30	34.85	31.76	34.80	74
Transform complexity reduction in the background	33.19	39.65	35.06	143

Table 3.2: Results for 'Claire' sequence

total processing time for the 100 frames of each video stream. Regular coding is a method without the concept of context-based coding. Rate-distortion optimization and rate control are effective in this case. FB coding is compression with using the concept of context-based[5]coding. In FB coding, motion estimation and compensation and rate-distortion optimization are not used for the background region. The quantization parameter[6] in the foreground (QPF) is set to 28, and in the background (QPB), to 42.

The results are shown for different values of the number of reference pictures (Ref. No.)

[5]Foreground/background coding

[6]Quantization parameter (QP) can be in the range of 0 to 51. Step-size doubles for every increment of 6 in QP. For QP equal to zero, the step-size is 0.625.

and search window. When the search window is equal to four, the processing time is about 30% less than when this parameter is equal to thirty, while all the other parameters are held the same. This is expected because the search area is smaller in the first case which causes the motion estimation to be carried out more rapidly. The number of reference frames has similar impact. Sometimes increasing the search area size and number of reference frames leads to better prediction for the current block and results in lower bitrate. In our examples there is no obvious difference between the quality and bitrate for a larger search range and a larger number of reference pictures.

It is apparent from the tables that significant reduction in complexity is not achieved by reducing the complexity of transforms. The reason is that in H.264, the transformation is applied to a 4×4 block, instead of an 8×8 in conventional video compression standards, and instead of discrete cosine transform, a separable integer transform with similar properties is used. Figure 3.4 shows that forward and inverse transform and quantization functions consume about 10% of the processing power. For H.263, on the other hand, this value is about 35%. Therefore, our results imply that these strategies are perhaps more appropriate for H.263 than H.264.

In the second and third cases (rows), in tables 3.1 and 3.2, the coding time is minimum for FB coding with intra and without intra frame. The first frame is always coded in intra mode because there are no reference frames to be used for inter coding. As a result, FB coding with and without intra frame do not concern the first frame. Here, some figures are shown to compare the regular coding and context-based coding.

Figure 3.5 demonstrates the PSNR of the foreground per frame in the 'Foreman' sequence for regular and FB coding with intra frames. Figure 3.6 displays this parameter for the background. The quasi-periodic peak values of the PSNR in FB coding with intra in both pictures, indicate the intra mode coding. The first frame is coded in intra as explained above. Apart from the first frame, eleven more frames are coded in intra mode which result in peaks of PSNR for the background and foreground, and make the new background for the next frame.

The difference between the PSNRF of regular and FB coding is that sometimes rate control results in low quality because the quantization parameter is raised to follow the desired bitrate. This effect is clearer in frames 66 and 23 in figure 3.5. While in FB coding the constant quantization parameter for the foreground makes its PSNR stable and the bitrate is then controlled by the quantization parameter of the background.

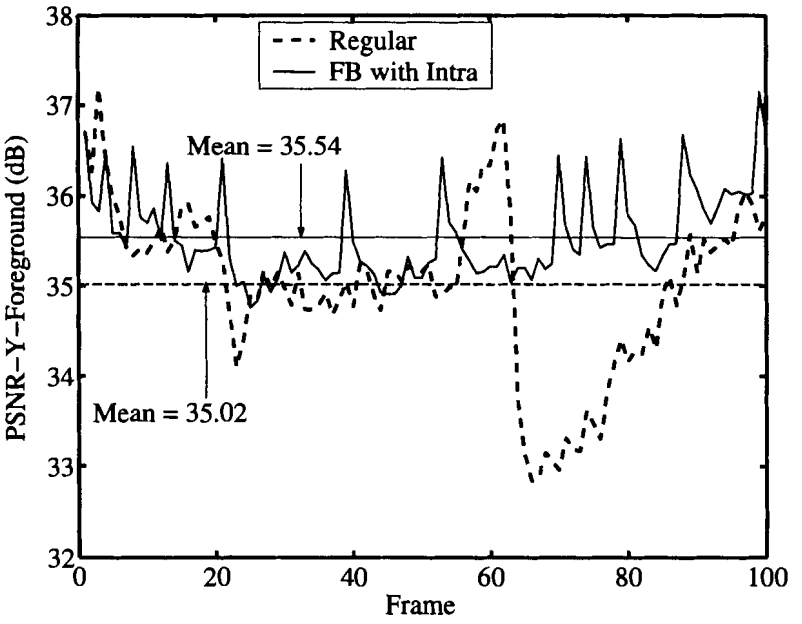

Figure 3.5: PSNRF for regular and FB coding with I-frames ('Foreman')

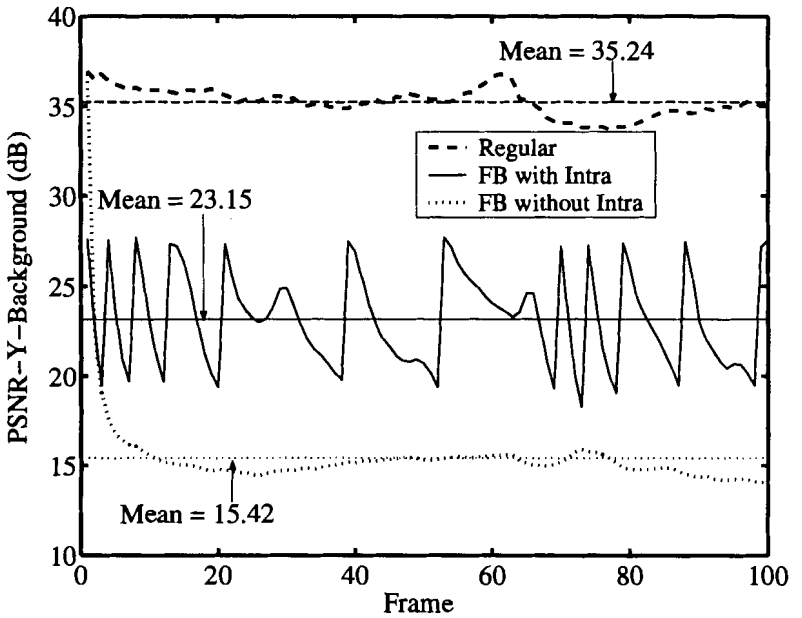

Figure 3.6: PSNRB for regular and FB coding with and without I-frames ('Foreman')

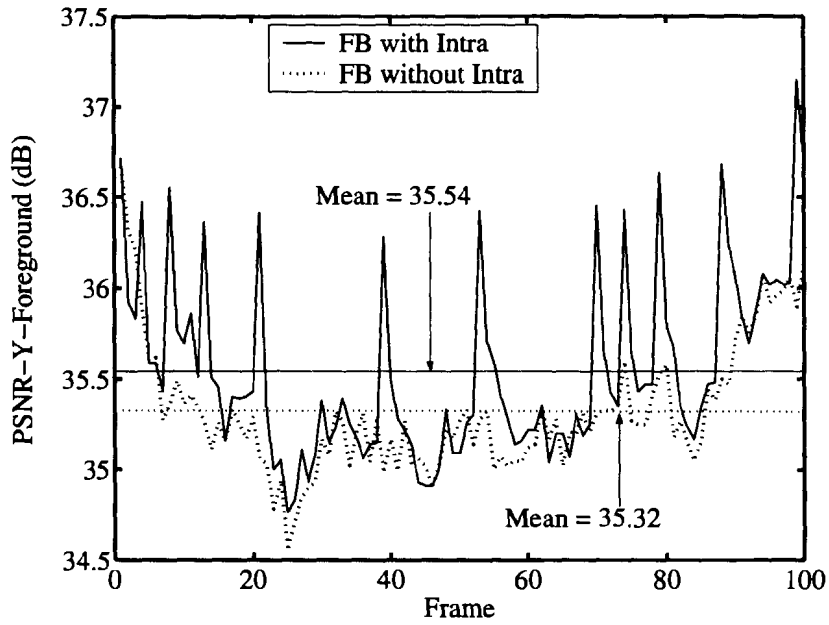

Figure 3.7: PSNRF for FB coding with and without I-frames ('Foreman')

The significant distinction between FB with and without intra is obvious in figures 3.6 and 3.7. The PSNR of the foreground is almost the same in both cases, but the PSNR of background is very low for the case of FB coding without intra frame. The enhancement in PSNR is achieved at the expense of a higher bitrate.

Figures 3.8, 3.9 and 3.10 show the same parameters for 'Claire' video sequence. The impact of rate control can be seen especially in frames 23, and the number of 90 and above. FB coding with intra, displays four intra frames except the first one. Intra frames are less than in the case of 'Foreman', because of the characteristics of the video sequence. If the PSNR of the background in FB coding, with and without intra frames are compared, there is no major distinction between the average values of PSNRB, but the PSNRB in FB coding without intra is degrading and the effect will be clearer, as the number of frames grows.

Based on our results for the two sequences, it can be concluded that the PSNRF and the bitrate of FB coding with intra and regular coding, are roughly the same. The total processing time is now considered. The coding time per frame for 'Foreman' and 'Claire' is displayed in Figures 3.11 and 3.12. It is obvious that coding time in FB coding is less than in regular coding. The improvement is more than 50%. The enhancement factor

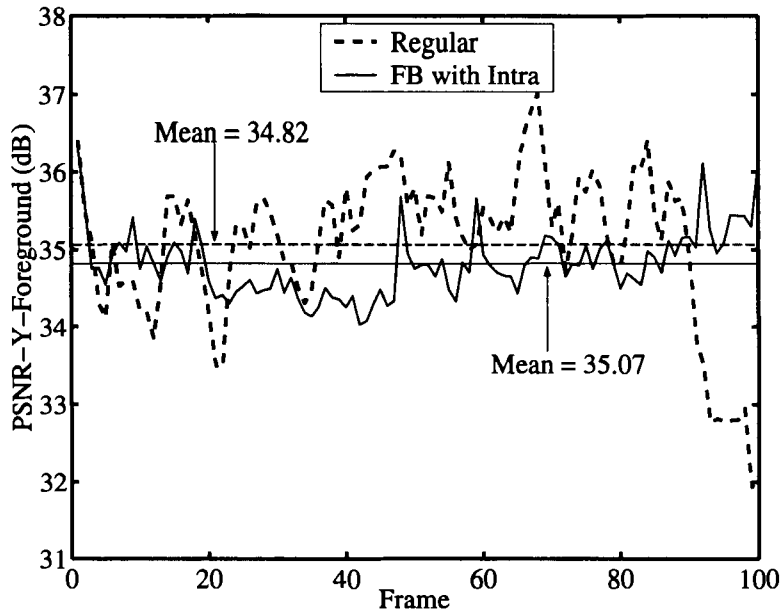

Figure 3.8: PSNRF for regular and FB coding with I-frames ('Claire')

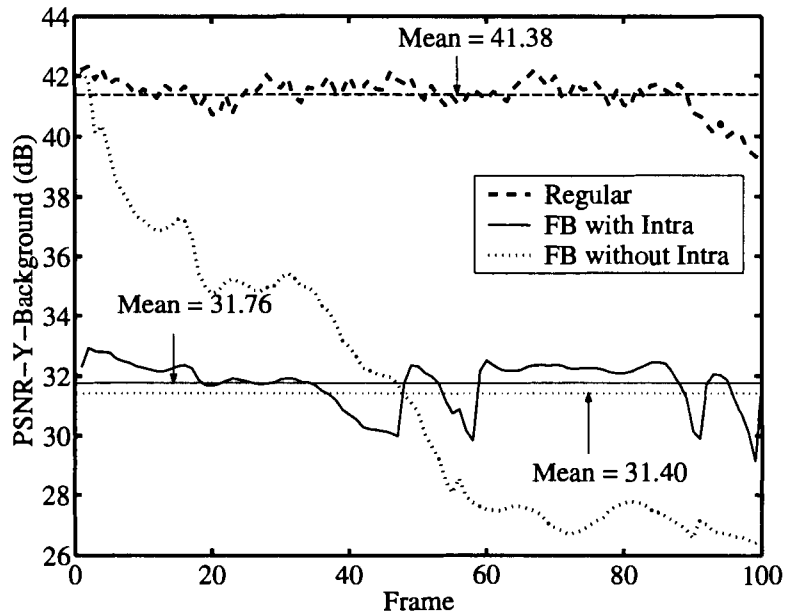

Figure 3.9: PSNRB for regular and FB coding with and without I-frames ('Claire')

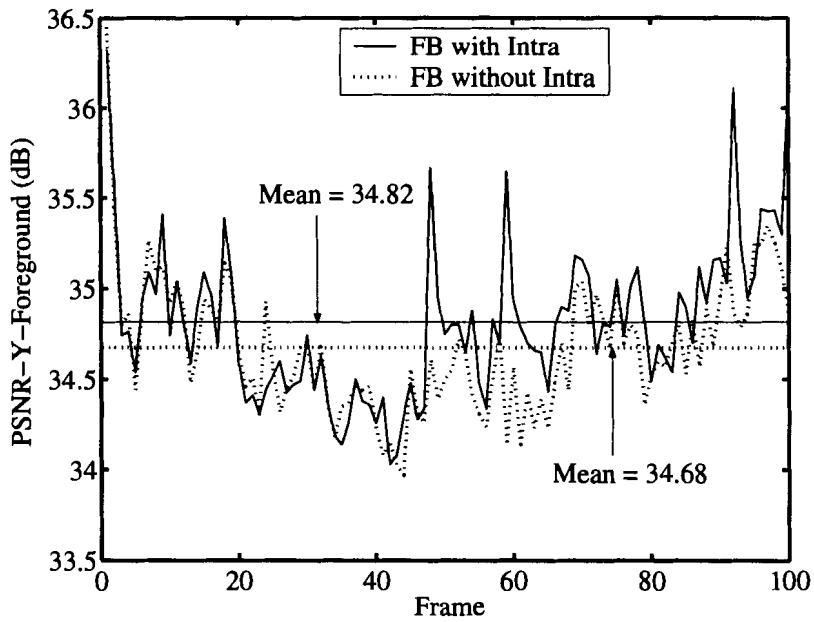

Figure 3.10: PSNRF for FB coding with and without I-frames ('Claire')

depends on the number of foreground macroblocks (i.e. the size of the foreground). A large number of foreground macroblocks results in high complexity and high processing time. The average number of foreground macroblocks in 'Claire' and 'Foreman' are about 13 and 31 respectively; therefore, the improvement in 'Claire' and 'Foreman' are about 65% and 54%. Figures 3.13 and 3.14 show the coding time and the number of foreground macroblocks in one graph for each video stream. The patterns of these two parameters are similar, which implies that in each frame of a video stream a larger number of macroblocks in the region of interest leads to a high processing time.

In these tests, the foreground is defined adaptively frame by frame. The initial experiments were done with a constant region of interest. Specific macroblocks were selected as the region of interest and this foreground was not changed for the whole test. In this method, a too-large foreground causes unnecessary processing time, and a too-small one leads to poor quality in the region of interest. The reason is that, when the foreground is constant, it must be large enough to cover the locations of the moving object (for example moving face) in the whole video sequence (in our case 100 frames), otherwise the constant foreground can not contain the region of interest in all frames. This affects those video sequences in which the

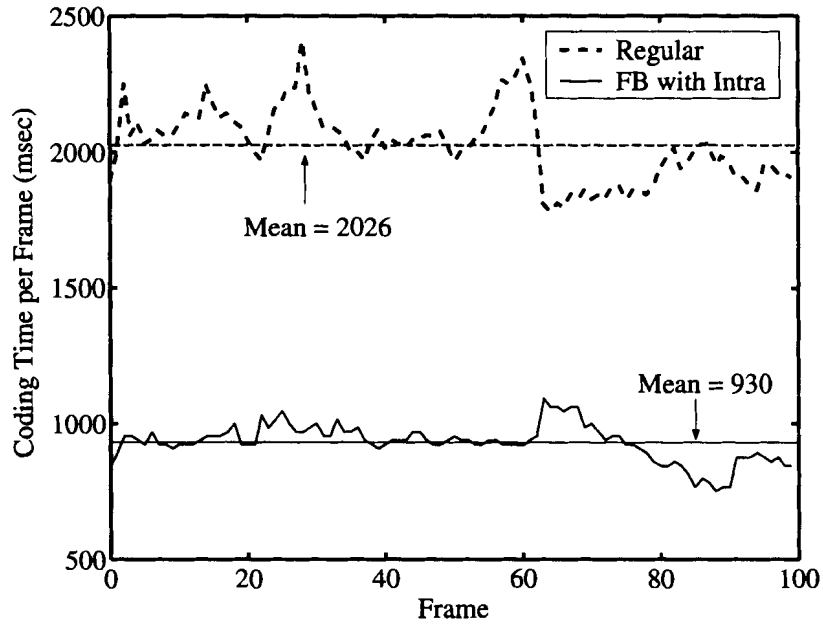

Figure 3.11: Processing time per frame for regular and FB coding with I-frame ('Foreman')

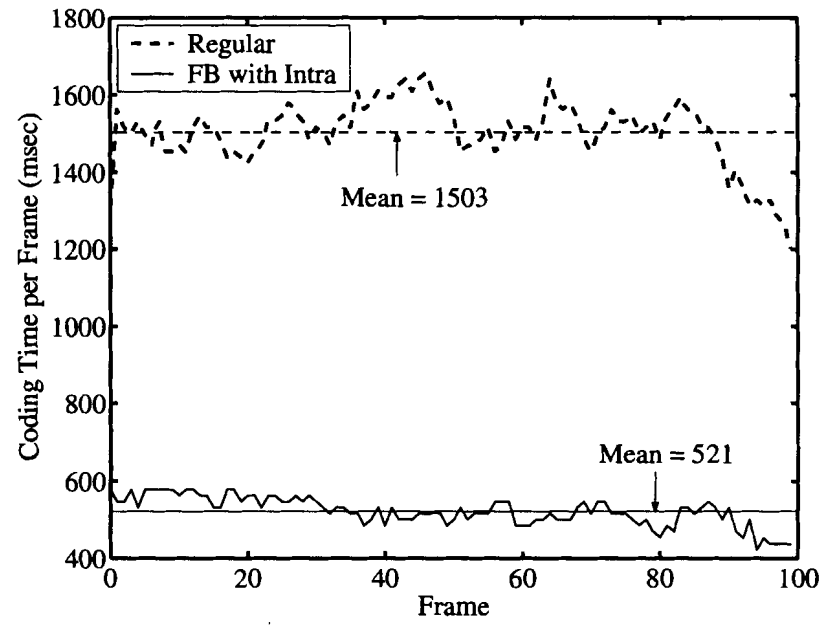

Figure 3.12: Processing time per frame for regular and FB coding with I-frame ('Claire')

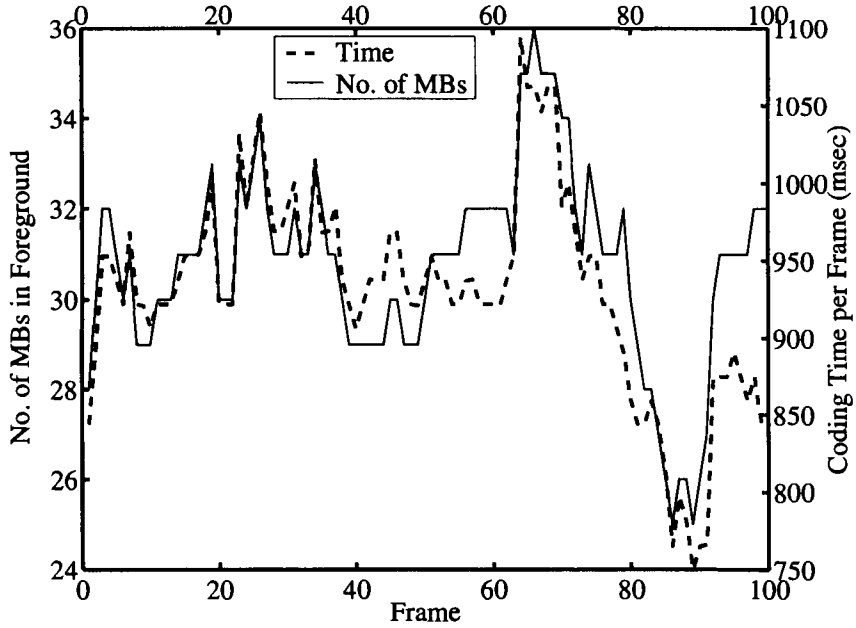

Figure 3.13: Comparison between No. of foreground MBs and coding time ('Foreman')

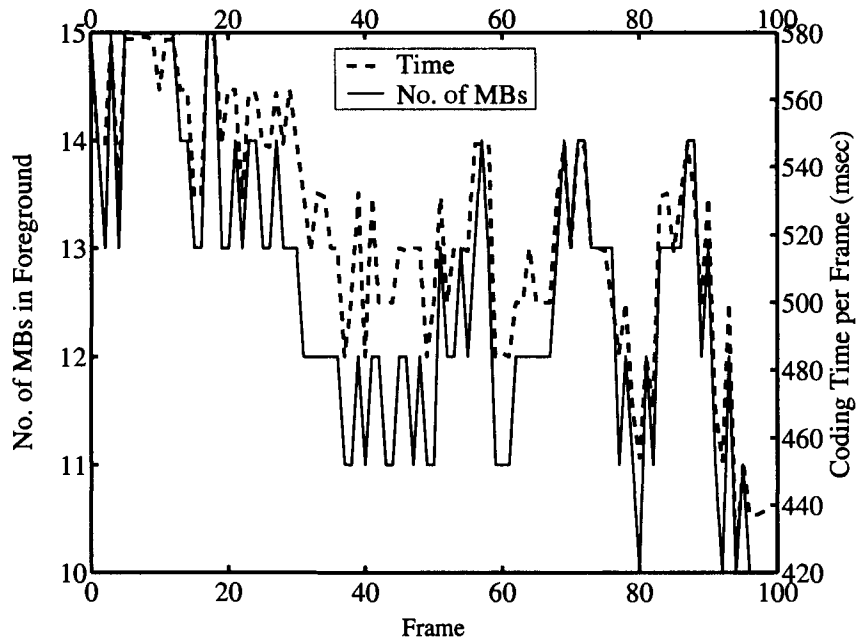

Figure 3.14: Comparison between No. of foreground MBs and coding time ('Claire')

motion range is larger, more than the other sequences. Results with adaptive segmentation show an improvement over constant segmentation; however, the constant segmentation also reduces the complexity and processing time by about 40% in comparison with regular coding [51]. FB coding with intra frames using the constant segmentation takes about 120 seconds for coding 100 frames of 'Foreman' video [51] while the required time for this activity with adaptive segmentation is about 93 seconds. Another reason for this improvement is the relatively simple algorithm of segmentation . Employing this segmentation method on each frame, extends the processing time by just about 6% which is not a significant complexity increase.

Chapter 4

Conclusions

This thesis proposes a method to reduce the complexity of the H.264 encoder. The method differentiates a picture's segments and compresses them based on their priority levels. The bitrate of coded video data and the quality of the reconstructed video in the region of interest, are kept the same as for regular coding. Each frame of the video sequence is separated into two areas: background (less significant region) and foreground (region of interest). Segmentation is based on the video context and the end-user's perception of quality.

Profiling results show that motion estimation and rate-distortion optimization are the two most complex processes in the coding algorithm. Therefore, these two processes are bypassed for the background, and the background motion vectors are set to zero. This makes the background constant in all frames. To maintain some dynamcis of the background, some frames are coded in intra mode. In intra mode, the current block is predicted from its neighboring blocks in the current frame. (c.f., inter mode, where the current block is predicted using previous frames.) The frame is coded in intra mode if the PSNR of the background goes below a certain threshold. The threshold depends on the characteristics of the video and the desired bitrate. The bitrate is tuned by a rate control system, which manages the bitrate by assigning a small quantization step-size to the foreground and a large step-size to the background. The goal of the employed strategies is to adjust the complexity and the bitrate with minimal impact on the quality of the region of interest. The proposed method is useful for the applications which have time, battery and bandwidth constraints such as real-time and wireless video communication, and for video streams in which some areas are more important than the other parts, for instance head of a person in front of a

background. The processing time is used for comparing and evaluating complexity of the different methods in this experiment.

The strengths of the proposed method in this project are:

- The experiments are executed on H.264 software, the latest standard. In the literature, most of the proposed techniques related to video coding are implemented on previous coding systems such as H.261 and especially H.263.
- In the proposed method, there is no need to transmit extra side information or adjust the decoder to be able to decompress the bit stream. The changes are just applied on the encoder; therefore, the compressed data is compatible with the standard of H.264.
- The encoder processing time is improved more than 50%, while the bitrate of the compressed data and the quality of reconstructed video in the region of interest, are maintained at a desired level.
- A simple segmentation algorithm is applied to each frame and the region of interest is detected in every frame. Using this algorithm does not increase the processing time significantly. Furthermore, distinguishing the region of interest in each frame is less time-consuming than using constant foreground in all frames.

In addition, the experiments in the thesis indicate:

- It is worthwhile reducing the search range in motion estimation and using a smaller number of reference frames, because this leads to less processing time and complexity.
- The complexity of the encoder depends on the size of the region of interest. Whenever the number of foreground macroblocks increases, the coding time grows.
- The complexity reduction for the transform used in this thesis, does not improve the results significantly. The reason is that the transform in H.264 is a modified version of the discrete cosine transform. In fact, the complexity is already decreased; therefore, the transform in H.264 is not as complex as the DCT in the other video compression techniques.
- Typically, conventional rate control systems adjust the bitrate by changing the quantization step-size frame by frame. These systems result in inferior quality in all regions including foreground, when the quantization step-size is raised. By using the proposed

method, the quantization step-size is not changed in the foreground so the foreground's quality is not affected.

4.1 Future work

The techniques presented in this work can be extended to improve the CODEC performance. Here is the list of possible topics for further development.

- An option for reducing the motion estimation complexity in the background is the prediction of motion vectors based on motion vectors in the foreground, instead of setting them to zero in our method. This may increase the PSNR of the background and as a result, the required intra frames may decrease, which leads to decrease in the bitrate.
- The segmentation method employed in this thesis, is based on skin color and it does not work very well in some cases such as too bright or too dark video sequences. Some areas with colors similar to skin color are detected inadvertently. The segmentation could be enhanced to distinguish the region of interest more precisely.
- In this thesis, each frame is separated into two levels, the less significant background and the more important foreground. For future research, it is recommended that each video frame be divided into more than two levels. Every level may utilize various parameters for encoding, based on its importance.
- The FMO (Flexible macroblock ordering) feature of H.264, explained in section 2.6.7, can be used. Further experiments are required to check if it can improve the performance.

Bibliography

[1] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression, Video Coding for Next-generation Multimedia*, Chichester, England: John Wiley and Sons, 2003.

[2] M. J. Riley and I. E. G. Richardson, *Digital Video Communications*, Chichester, England: Artech House, 1997.

[3] K. N. Ngan, C. W. Yap, and K. T. Tan, *Video Coding for Wireless Communication Systems*, New York: Marcel Dekker, 2001.

[4] "ITU-T Recommendation 601, Encoding parameters of digital television for studios", 1982. http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=R-REC-BT.601.

[5] I. E. Richardson, *Video Codec Design*, Chichester, England: John Wiley and sons, 2002.

[6] K. Rao and J. Hwang, *Techniques & Standards for Image, video and audio coding*, New Jersey: Prentice Hall, 1996.

[7] N. Wade and M. Swanston, *Visual Perception: An Introduction*, Psychology Press, 2nd ed., 2001.

[8] R. Aldridge, J. Davidoff, D. Hands, M. Ghanbari, and D. Pearson, "Recency effect in the subjective assessment of digitally coded television pictures", *Proceedings of Fifth International Conference on Image Processing and its Applications*, pp. 336–339, July 1995.

[9] "ITU-T Recommendation BT.500-11, Methodology for the subjective assessment of the quality of television pictures", 2002. http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=R-REC-BT.500.

[10] C. J. van den Branden Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio-temporal model of the human visual system, digital video compression algorithms and technologies", *Proceeding SPIE (International society for optical engineering)*, vol. 2668, pp. 450 – 461, February 1996.

[11] S. W. H. Wu, Z. Yu and T. Chen, "Impairment metrics for MC/DPCM/DCT encoded digital video", *The 22nd Picture Coding Symposium*, pp. 129–131, April 2001.

[12] K. Tan and M. Ghanbari, "A multi-metric objective picture quality measurement model for MPEG video", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 10, pp. 1208 – 1213, October 2000.

[13] K. Rao and P. Yip, *Discrete Cosine Transform*, Academic Press, 1990.

[14] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Massachusetts: Kluwer Academic Publishers, 4th ed., 1995.

[15] "ISO/IEC 15938, Information technology - Multimedia content description interface ", 2002. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER =34232&scopelist=.

[16] "ISO/IEC 21000, Information technology - Multimedia framework (MPEG-21)", 2003. http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=35366 &ICS1=35.

[17] "ISO/IEC 11172, Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s (MPEG-1)", 1993. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22412.

[18] "ISO/IEC 13818, Information technology - Generic coding of moving pictures and associated audio information (MPEG-2)", 1995. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=25434.

[19] "ITU-T Recommendation H.261, Video CODEC for audiovisual services at px64 kbit/s", 1993. www.itu.int/itudoc/itu-t/rec/h/h261.html.

[20] "ITU-T Recommendation H.263, Video coding for low bit rate communication,version 2", 1998. www.itu.int/itudoc/itu-t/rec/h/h263.html.

[21] "ISO/IEC 14496-2, Amendment1, Information technology-coding of audiovisual objects-part2:visual", 2001. http://www.iso.org/iso/en/CatalogueDetailPage.Catalogue Detail?CSNUMBER=39259&scopelist=.

[22] N. Brady, "MPEG-4 standardized method for the compression of arbitrarily shaped video objects", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 9, pp. 1170–1189, December 1999.

[23] P.Eisert, T. Wiegand, and B. Girod, "Model-aided coding: a new approach to incorporate facial animation into motion-compensated video coding", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 10, pp. 344–358, April 2000.

[24] T. W. Ralf Schafer and H. Schwarz, "The emerging H.264/AVC standard", *Heinrich Hertz Institute, Berlin, Germany*, January 2003.

[25] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.

[26] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 9, pp. 70–84, February 1999.

[27] T. Wiegand and B. Girod, *Multi-frame Motion-Compensated Prediction for Video Transmission*, Kluwer Academic Publishers, 2001.

[28] T. Wedi, "Motion compensation in H.264/AVC", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 577 – 586, July 2003.

[29] M. Flierl, T. Wiegand, and B. Girod, "A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction", *Data Compression Conference*, March 1998.

[30] M. Flierl and B. Girod, "Generalized b pictures and the draft JVT/H.264 video compression standard", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 587–597, July 2003.

[31] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 614–619, July 2003.

[32] H. Marpe, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.

[33] M. Wien, "Variable block-size transforms for H.264/AVC", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 604–613, July 2003.

[34] S. W. Golomb, "Run-length encoding", *IEEE Transaction on Information Theory*, vol. 12, 1966.

[35] G. Bjontegaard and K. Lillevold, "Context-adaptive VLC coding of coefficients", *Joint Video Team document JVT-C028*, May 2002. ftp://standards.polycom.com/2002_05_Fairfax/.

[36] D. Marpe, H. Schwarz, and T. Wiegand, "Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 620–636, July 2003.

[37] M. Karczewicz and R. Kurceren, "The SP and SI frames design for H.264/AVC", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 637 – 644, July 2003.

[38] S. Wenger, "H.264/AVC over IP", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 645–656, July 2003.

[39] T. Stockhamer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wirelss environments", *IEEE Transaction Circuits and Systems for Video Technology*, vol. 13, pp. 657 - 673, July 2003.

[40] K. N. Ngan, T. Meier, and D. Chai, *Advanced Video Coding: Principles and Techniques*, Amsterdam, The Netherlands: Elsevier Science B. V., 1999.

[41] D. Chai and K. N. Ngan, "Foreground/background video coding scheme", *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 1448–1451, June 1997.

[42] D. Chai and K. N. Ngan, "Coding area of interest with better quality", *IEEE International Workshop on Intelligent Signal Processing and Communication Systems*, pp. S20.3.1–S20.3.10, November 1997.

[43] D. Chai and K. N. Ngan, "Foreground/background video coding using H.261", *SPIE (International society for optical engineering) Visual Communications and Image Processing*, pp. 434–445, January 1998.

[44] C.-H. Lin and J.-L. Wu, "Content-based rate control scheme for very low bitrate video coding", *IEEE Transactions on Consumer Electronics*, vol. 43, pp. 123–133, May 1997.

[45] A. Eleftheriadis and A. Jacquin, "Model-assisted coding of video teleconferencing sequences at low bitrates", *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 177–180, June 1994.

[46] Z. Ji, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end power-optimized video communication over wireless channels", *IEEE International Workshop on Multimedia Signal Processing*, pp. 447–452, October 2001.

[47] J. C. C. Christopoulos, J. Bormans and A. N. Skodras, "The vector-radix fast cosine transform: pruning and complexity analysis", *Signal Processing*, vol. 43, 1995.

[48] http://trace.eas.asu.edu/yuv/qcif.html.

[49] C. A. B. A. Albiol, L. Torres and E. Delp, "A simple and efficient face detection algorithm for video database applications", *IEEE International Conference - Image Processing*, vol. 2, pp. 239–242, 2000.

[50] http://bs.hhi.de/suehring/tml/download/old-jm/.

[51] L. Sahafi, T. S. Randhawa, and R. H. S. Hardy, "Context-based complexity reduction of H.264 in video over wireless applications", *IEEE International Workshop on Multimedia Signal Processing*, September 2004.