LOW COMPLEXITY ENCODER USING MACHINE LEARNING

by

THEJASWINI PURUSHOTHAM

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2010

ACKNOWLEDGEMENTS

Multitudes of pixels come together to form a lovely portrait. Similarly, the fruition of a thesis happens because of the encouragement and guidance of numerous people. Thus, I would like to take this opportunity to thank everyone who invested their precious time on me in the last two years.

In the fall of 2008, I walked into the room of Dr.K.R. Rao with the hopes of learning from the master of video coding. Though we were total strangers, he immediately put me at ease by creating a very positive working atmosphere which entails my sincere appreciation. His mentoring has undoubtedly had a profound impact on me. I am greatly indebted to him.

I am deeply grateful to Dr. Dongil Han for always being available in the lab and providing me with continued financial support and technical advice. I would like to thank the other members of my advisory committee Dr. W. Alan Davis and Dr. Jonathan Bredow for reviewing this thesis document and offering insightful comments. My sincere thanks to Pragnesh and Suchethan. The love, affection and encouragement of Bhumika, Bhavana, Gunpreet, Thara, Srikanth and all my friends who kept me going through the trying times of my Masters.

Finally, my sincere gratitude and love goes out to my mom Ms. N. Pushpalatha and my dad Mr. M. Purushotham. They have been my role models and have shaped me to be the positive and independent person that I am today. My brother Arvind has been very loving and supportive throughout and this thesis is dedicated to my family.

September 8, 2010

ABSTRACT

LOW COMPLEXITY H.264 ENCODER USING MACHINE LEARNING

Thejaswini Purushotham, M.S

The University of Texas at Arlington, 2010

Supervising Professor: K.R. Rao

H.264 is currently one of the most widely accepted video coding standards in the industry. Several software and hardware solutions for the H.264 video encoder exist in the market at present. H.264 is used in such applications as Blu-ray Disc, videos on the internet, digital video broadcast, direct-broadcast satellite television service, cable television services, and real-time videoconferencing. This thesis uses the WEKA (Waikato Environment for Knowledge Analysis) tool to generate the classification rule. WEKA is detailed in Chapter 3. The input attributes to the WEKA have been calculated from the video sequence to be encoded. The procedure has been elaborated in Chapter 4.

For real time applications like videoconferencing it is essential that the encoding time taken by the video codec be as low as possible. In the H.264 video codec, the macroblock mode decision in inter frames is computationally the most expensive process since it uses such features as variable block size, motion estimation and quarter pixel motion compensation in H.264 encoder. Hence, the goal of this thesis is to reduce the encoding time while conserving the quality and compression ratio.

Machine learning has been used to decide the mode decisions and hence reduce the motion estimation time. The proposed machine learning method on an average decreases the encoding time by 42.86405% for mode decisions in H.264 encoder with a loss of only .01070% decrease in structural similarity index metric (SSIM).

Motion Estimation is the most time consuming part of the encoder. An average of 60 -70 % of the total encoding time is taken for motion estimation. The time consuming sum of absolute differences(SAD) method adopted in the H.264 encoder in JM 16.2 software has been replaced with a classification rule. Assuming FS (Full Search) and P block types, Q reference frames and a search range of MxN, MxNxPxQ computations are needed.

The classification rule has been implemented as a series of if-else statements. The time taken to execute the if-else statements is lesser than the time taken to execute the SAD. Hence this thesis describes a reduction in the H.264 encoder execution time.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

3G/4G	Third or Fourth Generation
3GPP	3rd Generation Partnership Project
ASO	Arbitrary slice ordering
AVC	Advanced Video Coding
AVI	Audio Video Interleave
B MB	Bi-predicted MB
CIF	Common Intermediate Format
DCT	Discrete Cosine Transform
DP	Data Partition
DSP	Digital Signal Processing
DVD	Digital Versatile Disc
FMO	Flexible macroblock ordering
FRExt	Fidelity Range Extensions
GOP	Group Of Pictures
IDR	Instantaneous Decoder Refresh
I MB	Intra Predicted MB
IEC	International Electro Technical Commission
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Transmission sector
JM	Joint Model

JVT	Joint Video Team
P MB	Inter Predicted MB
IDCT	Inverse Discrete Cosine Transform
IQ	Inverse Quantizer
MB	Macroblock
MBAFF	Macroblock level Adaptive Frame/Field
PicAFF	Picture level Adaptive Frame/Field
ME	Motion Estimation
MC	Motion Compensation
MV	Motion Vector
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
NAL	Network Abstraction Layer
PSNR	Peak –to – peak Signal to Noise Ratio
Q	Quantizer
QCIF	Quarter Common Intermediate Format
R-D	Rate – Distortion
SP/SI	Switched P / Switched I
SMPTE	Society of Motion Picture and Television Engineers
SSIM	Structural Similarity Index Measure
SVC	Scalable Video Coding
VCEG	Video Coding Experts Group
VLC	Variable Length Coding
VLD	Variable Length Decoder

YUV	Y- Luminance and UV- Chrominance
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

1.1 Significance

In the era of fast paced digital world, the distribution and creation of digital video is widespread. Technologies that can accommodate the needs of a broad array of customers from the video expert or an industry professional to the casual at-home user are necessary.

Regardless of the user's level of expertise, it is important to select an encoding solution that offers extensive format support plus the highest quality and fastest video encoding speed. For many users, the ability to repurpose content for distribution to a growing number of new media channels, such as web, DVD and mobile phones, is also very important. There are various stages in the digital video workflow process, however, one of the most critical and challenging is encoding, since this step has a direct impact on the ultimate quality of the viewing experience. With the many complexities that are involved in video encoding, this step can make or break the end product if the "right" solution is not selected.

Many of the challenges experienced during the encoding process involve file format compatibility, optimizing picture quality, and/or processing speed. Video encoding can be viewed as a complex process due to the many variables at play, including an ever-growing number of video codecs, frame sizes and frame rates required for viewing on everything from HDTV to mobile devices. With this in mind, it is important to find an encoding solution that addresses all of these variables and simplifies the process while offering maximum flexibility to distribute the video to more viewers at the highest quality, regardless of how it is being viewed. The best solutions in the market not only ensure the highest compression quality, but also include the ability to apply video processing filters to ensure the highest visual quality of the final content [1].

Video encoding requires high processing and hence has traditionally been implemented on special-purpose hardware. Software solutions are feasible for applications requiring low encoding rates, typically being the non real-time applications. Software solutions need a general-purpose computing system .The usage of general-purpose computing system has various advantages. Software solutions are easily available and flexible and allow experimenting with and hence improving the various components of the encoder. For a real-time software solution to be feasible it is necessary that the encoding time offered by the solution be very low. Several prominent companies/organizations that provide software implementations for H.264/AVC are as follows [1]: Adobe Systmes, AVS Video Editor, Apple Inc, BT Group, Sorenson, Intel, Main Concept, On2 Technologies, LGPL, CoreAVC, NeroLogic, Blu-Code, Sonic Cinevision, Siway, SBMC Media Center.

Figure 1.1 shows the usage of H.264/AVC video codec in the market. The H.26/AVC[1][8][9][10] standard provides more than 100 times the number of coding modes of MPEG-2, and H.264/AVC achieves a compression ratio more than two times that of MPEG-2 by selecting the best coding mode from among them. As a result, the encoding complexity of the encoder processing is about ten times or more than that of MPEG-2 when H.264/AVC JM reference software[29] distributed by JVT is used. In addition, the memory bandwidth required for processing increases in proportionencoding complexity. To achieve a practical implementation of H.264/AVC, a significant reduction in encoding complexity must be achieved while maintaining coding efficiency.

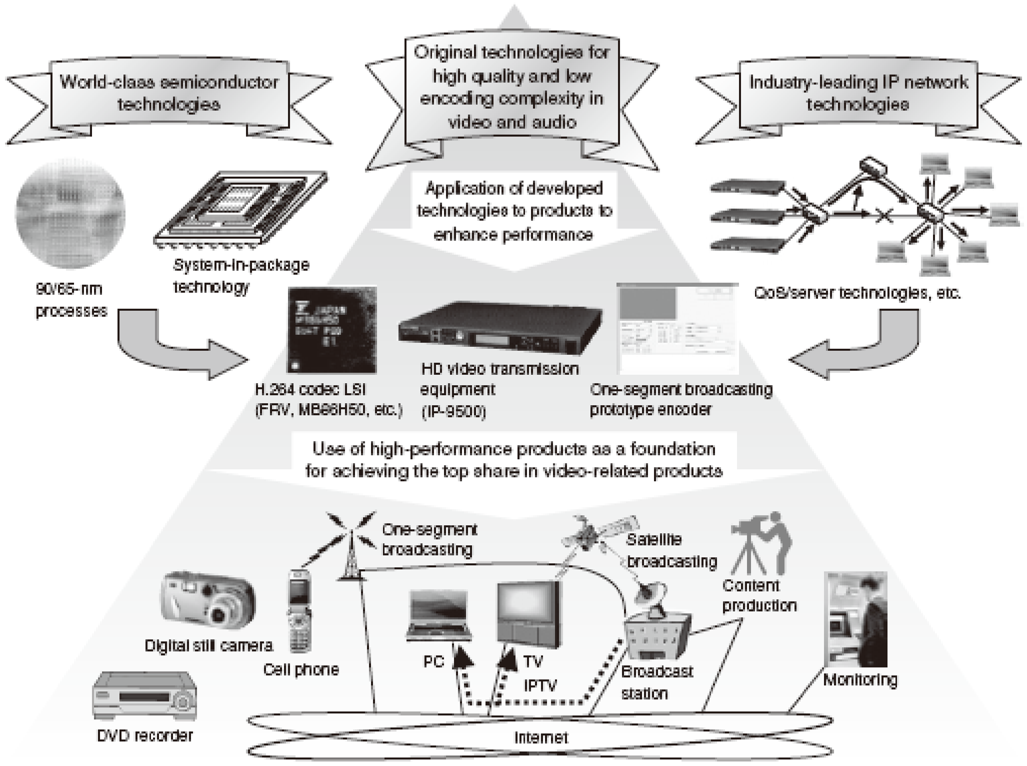

World-class semiconductor technologies
Original technologies for high quality and low encoding complexity in video and audio
Industry-leading IP network technologies
Application of developed technologies to products to enhance performance
System-in-package technology
90/65-nm processes
QoS/server technologies, etc.
H.264 codec LSI (FRV, MB86H50, etc.)
HD video transmission equipment (IP-9500)
One-segment broadcasting prototype encoder
Use of high-performance products as a foundation for achieving the top share in video-related products
One-segment broadcasting
Satellite broadcasting
Content production
Digital still camera
Cell phone
PC
TV
IPTV
Broadcast station
Monitoring
DVD recorder
Internet

Figure 1.1 H.264/AVC products to video-related markets [51]

H.264 video coding standard is the latest block-oriented motion-compensation-based codec standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG). H.264 can achieve considerably higher coding efficiency than previous standards. Unfortunately, this comes at a cost in considerably increased complexity at the encoder mainly due to motion estimation [1], in-loop deblocking filter [1], sub-pel interpolation [15] and mode decision [1]. The high-computational complexities of H.264 and realtime requirements of video systems represent the main challenge to overcome in the development of efficient encoder solutions. Different techniques to reduce complexity in H.264 have been proposed in [41], [42], [44] and [45].But few approaches have been done using machine learning [43]. This thesis focuses on reducing the complexity of the H.264 JM 16.2 [29] encoder using machine learning techniques. The idea behind using machine learning is to exploit structural similarities in video in order to make optimal prediction modes throu the use of fast if-else statements instead of the usual cumbersome SAD (sum of absolute differences) and cost evaluations.

1.2 Summary

This thesis contributes in the study and implementation of machine learning and data mining techniques applied to video compression. A new technique is implemented for reducing complexity in H.264 using as reference software optimized encoder. Results show reduction in complexity in terms of encoding time for different video formats and video context. Chapter 2 is an introduction to H.264/AVC standard. It discusses the characteristic features of H.264.

CHAPTER 2

INTRODUCTION

2.1 Introduction

H.264/MPEG4-Part 10 advanced video coding (AVC) introduced in 2003 became one of the latest and most efficient video coding standards [4]. The H.264 standard was developed by the Joint Video Team (JVT), consisting of VCEG (Video Coding Experts Group) of ITU-T (International Telecommunication Union – Telecommunication standardization sector), and MPEG (Moving Picture Experts Group) of ISO/IEC [4].

H.264 can support various interactive (video telephony) and non-interactive applications (broadcast, streaming, storage, video on demand) as it facilitates a network friendly video representation. It leverages on the previous coding standards such as MPEG-1, MPEG-2, MPEG-4 part 2, H.261, H.262 and H.263 [8] and adds many other coding tools and techniques which give it superior quality and higher compression efficiency.

The standardization of the first version of H.264/AVC was completed in May 2003. The JVT then developed extensions to the original standard that are known as the fidelity range extensions (FRExt) [11]. These extensions enable higher quality video coding by supporting increased sample bit depth precision and higher-resolution color information, including sampling structures known as YUV 4:2:2 and YUV 4:4:4. Several other features are also included in the fidelity range extensions, such as adaptive switching between 4×4 and 8×8 integer transforms, encoder-specified perceptual-based quantization weighting matrices, efficient inter-picture lossless coding, and support of additional color spaces. The design work on the fidelity range extensions was completed in July 2004, and the drafting work on them was completed in September 2004.

Scalable video coding (SVC) [48] as specified in Annex G of H.264/AVC allows the construction of bitstreams that contain sub-bitstreams that conform to H.264/AVC. For temporal bitstream scalability, i.e., the presence of a sub-bitstream with a smaller temporal sampling rate than the bitstream, complete access units are removed from the bitstream when deriving the sub-bitstream. In this case, high-level syntax and inter prediction reference pictures in the bitstream are constructed accordingly. For spatial and quality bitstream scalabilities, i.e. the presence of a sub-bitstream with lower spatial resolution or quality than the bitstream, network abstraction layer (NAL) units are removed from the bitstream when deriving the sub-bitstream. In this case, inter-layer prediction, i.e., the prediction of the higher spatial resolution or quality signal by data of the lower spatial resolution or quality signal, is typically used for efficient coding. The scalable video coding extension was completed in November 2007.

The next major feature added to the standard was Multiview Video Coding (MVC) [49]. Specified in Annex H of H.264/AVC, MVC enables the construction of bit streams that represent more than one view of a video scene. An important example of this functionality is stereoscopic 3D video coding. Two profiles were developed in the MVC work: one supporting an arbitrary number of views and designed specifically for two-view stereoscopic video. The Multiview Video Coding extensions were completed in November 2009 [5]. Like any other previous motion-based codecs, H.264 uses the following basic principles of video compression [4]:[4]

- Transform for reduction of spatial correlation
- Quantization for control of bit rate
- Motion compensated prediction for reduction of temporal correlation
- Entropy coding for reduction in statistical correlation.

The improved coding efficiency of H.264 can be attributed to the additional coding tools and the new features. Listed below are some of the new and improved techniques used in H.264 for the first time [9]:

- Adaptive intra-picture prediction

- Small block size transform with integer precision
- Multiple reference pictures and generalized B-frames
- Variable block sizes for ME and MC.
- Quarter pixel precision for motion compensation
- Content adaptive in-loop deblocking filter and
- Improved entropy coding by introduction of CABAC (context adaptive binary arithmetic coding) and CAVLC (context adaptive variable length coding)

The increase in the coding efficiency and increase in the compression ratio result in a greater complexity of the encoder and the decoder algorithms of H.264, as compared to previous coding standards. In order to develop error resilience for transmission of information over the network, H.264 supports the following techniques [9]:

- Flexible macroblock ordering (FMO)
- Switched slice
- Arbitrary slice order (ASO)
- Redundant slice (RS)
- Data partitioning (DP)
- Parameter setting

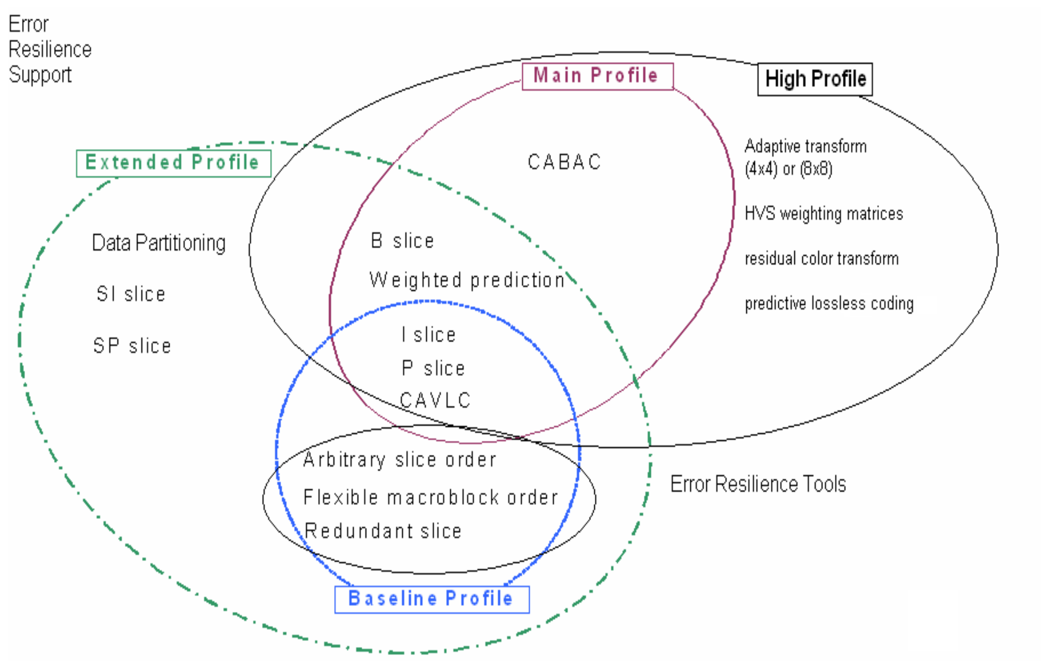


Figure 2.1 Different profiles in H.264 with distribution of various coding tools among the profiles [8]

2.2 Profiles and Levels of H.264

2.2.1. Baseline Profile

The list of tools included in the baseline profile are I (intra coded) and P (predictive coded) slice coding, enhanced error resilience tools of flexible macroblock ordering, arbitrary slices and redundant slices. It also supports CAVLC (context-based adaptive variable length coding). The baseline profile is intended to be used in low delay applications, applications demanding low processing power, and in high packet loss environments. This profile has the least coding efficiency among all the three profiles.

2.2.2. Main Profile

The coding tools included in the main profile are I, P, and B (bi directionally predictive coded) slices, interlaced coding, CAVLC and CABAC (context-based adaptive binary arithmetic coding). The tools not supported by main profile are error resilience tools, data partitioning and

SI (switched intra coded) and SP (switched predictive coded) slices. This profile is aimed to achieve highest possible coding efficiency.

2.2.3. High Profile

The coding tools included in the main profile are I, P, and B (bi directionally predictive coded) slices, interlaced coding, CAVLC and CABAC (context-based adaptive binary arithmetic coding). The tools not supported by main profile are error resilience tools, data partitioning and SI (switched intra coded) and SP (switched predictive coded) slices. This profile is aimed to achieve highest possible coding efficiency.

2.2.4. Extended Profile

This profile has all the tools included in the baseline profile. As illustrated in the Figure 2 1 this profile also includes B, SP and SI slices, data partitioning, interlaced frame and field coding, picture adaptive frame/field coding and MB adaptive frame/field coding. This profile provides better coding efficiency than baseline profile. The additional tools result in increased complexity.

2.2.5. High Profiles defined in the FRExts amendment

In September 2004 the first amendment of H.264/MPEG-4 AVC video coding standard was released [11]. A new set of coding tools were introduced as a part of this amendment. These are termed as “Fidelity Range Extensions” (FRExts). The aim of releasing FRExts is to be able to achieve significant improvement in coding efficiency for higher fidelity material. It also has lossless representation capability: I PCM raw sample value macroblocks and entropy coded transform by-pass lossless macroblocks (FRExts only). The application areas for the FRExts tools are professional film production, video production and high-definition TV/DVD.

The FRExts amendment defines four new profiles (Figure 2 2) [12]:

- High (HP)
- High 10 (Hi10P)

- High 4:2:2 (Hi422P)

- High 4:4:4 (Hi444P)

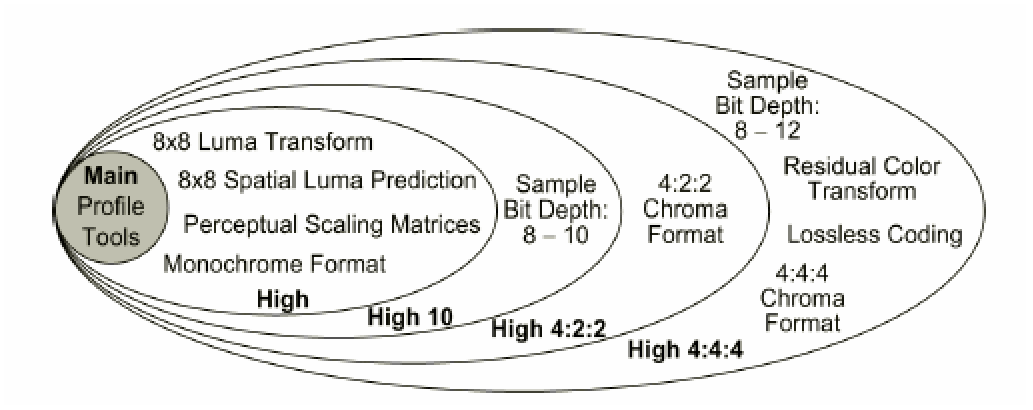


Figure 2.2 Tools introduced in FRExts and their classification under the new high profiles [11]

All four of these profiles build further upon the design of the prior Main profile. The Table 2.1 provides a comparison of the high profiles introduced in FRExts with a list of different coding tools and which of them are applied to which profile. All of the high profiles include the following three enhancements to coding efficiency performance [13].

- Adaptive macroblock-level switching between 8x8 and 4x4 transform blocks.

The main aim behind introducing the 8x8 transform in FRExts is to fulfill high fidelity video demands preservation of fine details and textures. To achieve this, larger basis functions are required. However a smaller transform like the 4x4 reduces ringing artifacts and reduces computational complexity. The encoder adaptively chooses between the 4x4 and 8x8 transforms.

The transform selection process is limited by the following conditions

If an inter-coded MB has sub-partition smaller than 8x8 (i.e. 4x8, 8x4, 4x4), then 4x4 transform is used.

If an intra-coded MB is predicted using 8x8 luma spatial prediction, only 8x8 transform is used.

- Encoder-specified perceptual-based quantization scaling matrices

The encoder can specify a matrix for scaling factor according to the specific frequency associated with the transform coefficient for use in inverse quantization scaling by the decoder. This allows optimization of the subjective quality according to the sensitivity of the human visual system (HVS), less sensitive to the coded error in high frequency transform coefficients [8].

- Encoder-specified separate control of the quantization parameter for each chroma component

Table 2.1 Comparison of the high profiles and corresponding coding tools introduced in the FRExts [11]

Coding tools	High	High 10	High 4:2:2	High 4:4:4
Main Profile tools	x	x	X	x
4:2:0 Chroma format	x	x	X	x
8 bit sample bit depth	x	x	X	x
	x	x	X	x

Table 2.1 -*Continued*

8x8 vs. 4x4 transform adaptivity				
Quantization scaling matrices	x	x	X	x
Separate Cb and Cr Quantization parameter (QP) control	x	x	X	x
Monochrome video format	x	x	X	x
9 and 10 bit sample depth		x	X	x
4:2:2 Chroma format			X	x
11 and 12 sample bit depth				x
4:4:4 Chroma format				x
Residual color transform				x
Predictive lossless coding				x

2.2.6. Overview of Scalable video codec

A video bit stream is called scalable when parts of the stream can be removed in a way that the resulting sub-stream forms another valid bit stream for some target decoder, and the sub-stream represents the source content with a reconstruction quality that is less than that of the complete original bit stream. The modes of scalability are temporal, spatial, and quality [14] . Spatial and temporal scalabilities describe cases in which subsets of the bit stream represent the source content with a reduced picture size (spatial resolution) or frame rate (temporal resolution), respectively. With quality scalability, the sub-stream provides the same spatial-temporal resolution as the complete bit stream, but with a lower signal-to-noise ratio (SNR).

Quality scalability is also commonly referred to as fidelity or SNR scalability. Figure 2.3 shows the basic principle and applications of scalable coding.

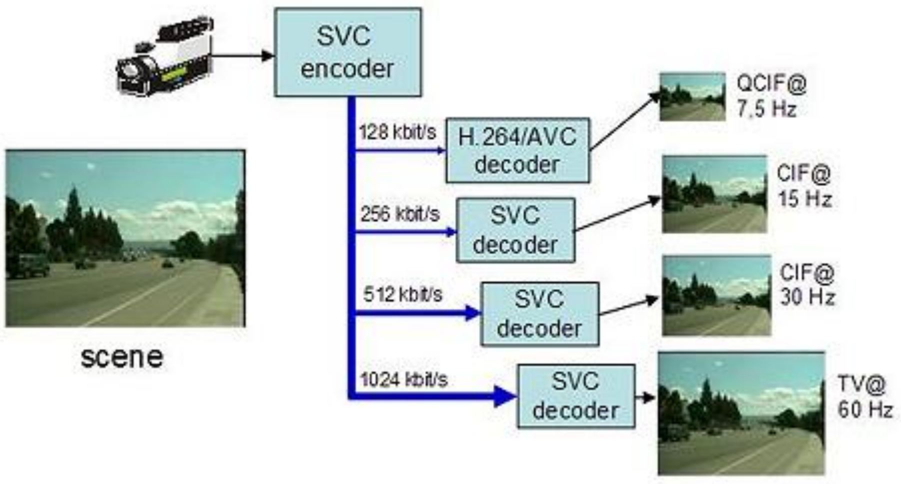


Figure 2.3 Scalable video coding [50]

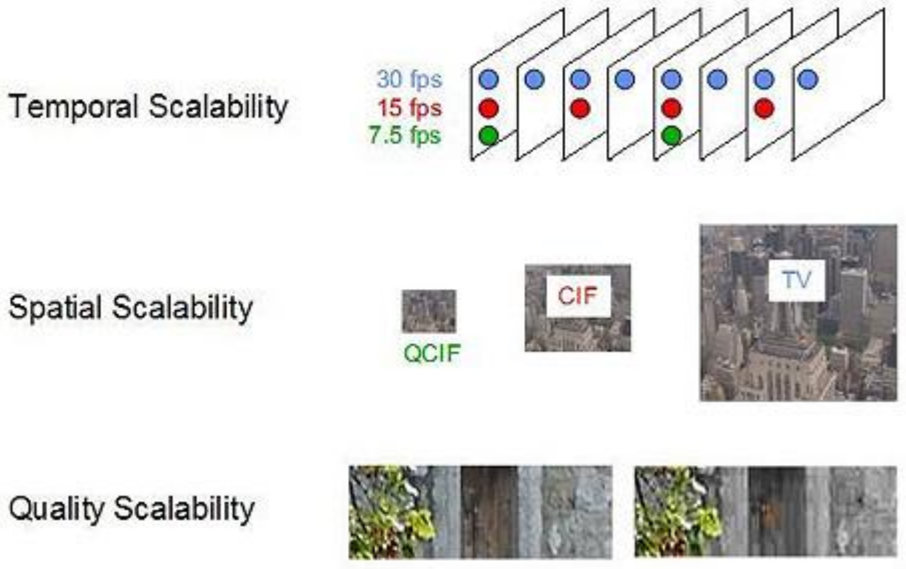


Figure 2.4 The basic types of scalability in video coding [50]

2.2.6.1 Spatial Scalability

For supporting spatial scalable coding, SVC follows the conventional approach of multi-layer coding. In each spatial layer, motion-compensated prediction and intra prediction are employed as for single-layer coding. In addition to these basic coding tools of H.264/AVC, SVC provides inter-layer prediction methods, which allow an exploitation of the statistical dependencies between different layers for improving the coding efficiency of enhancement layers. The supported inter-layer prediction methods employ the reconstructed samples of the lower layer signal. The prediction signal is either formed by motion-compensated prediction inside the enhancement layer, by up-sampling the reconstructed lower layer signal, or by averaging such an up-sampled signal with a temporal prediction signal. Apart from these, two additional inter-layer prediction concepts have been added in SVC: prediction of macroblock modes and associated motion parameters and prediction of the residual signal. All inter-layer prediction tools can be chosen on a macroblock or sub-macroblock basis allowing an encoder to select the coding mode that gives the highest coding efficiency [14].

2.2.6.2 Inter-layer intra prediction

For SVC enhancement layers, an additional macroblock coding mode (signaled by the syntax element base_mode_flag equal to 1) is provided, in which the macroblock prediction signal is completely inferred from co-located blocks in the reference layer without transmitting any additional side information. When the co-located reference layer blocks are intra-coded, the prediction signal is built by the up-sampled reconstructed intra signal of the reference layer – a prediction method also referred to as inter-layer intra prediction.

2.2.6.3 Inter-layer macroblock mode and motion prediction

When the base_mode_flag is equal to 1 and at least one of the co-located reference layer blocks is not intra-coded, the enhancement layer macroblock is inter-picture predicted as in single-layer H.264/AVC coding, but the macroblock partitioning – specifying the decomposition into smaller block with different motion parameters – and the associated motion parameters are completely derived from the co-located blocks in the reference layer. This concept is also referred to as the inter-layer motion prediction. For the conventional inter-coded macroblock types of H.264/AVC, the scaled motion vector of the reference layer blocks can also be used as replacement for usual spatial motion vector predictor.

2.2.6.4 Inter-layer residual prediction

A further inter-layer prediction tool, referred to as inter-layer residual prediction, targets a reduction of the bit rate required for transmitting the residual signal of inter-coded macroblocks. With the usage of residual prediction (signaled by the syntax element residual_prediction_flag equal to 1), the up-sampled residual of the co-located reference layer blocks is subtracted from the enhancement layer residual (difference between the original and the inter-picture prediction signal) and only the resulting difference, which often has a smaller energy then the original residual signal, is encoded using transform coding as specified in H.264/AVC [14] as illustrated in Figure 2.5.

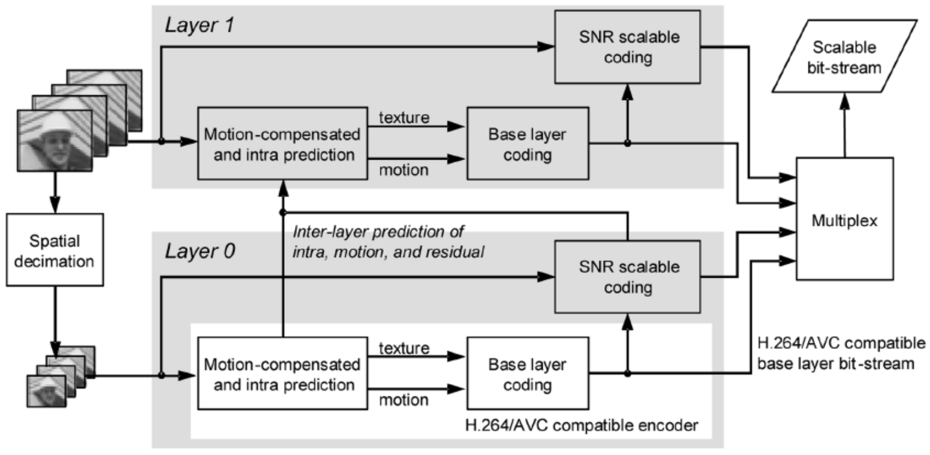

Layer 1
SNR scalable coding
Scalable bit-stream
texture
Motion-compensated and intra prediction
Base layer coding
motion
Multiplex
Spatial decimation
Inter-layer prediction of intra, motion, and residual
Layer 0
SNR scalable coding
texture
Motion-compensated and intra prediction
Base layer coding
motion
H.264/AVC compatible base layer bit-stream
H.264/AVC compatible encoder

Figure 2.5 Coding structure example with two spatial layers. [47]

2.2.6.5 SNR Scalability

Different techniques exist in the Joint Scalable Video Model (JSVM) for providing SNR scalability [14]

2.2.6.7 Fine-Grain Scalability

Fine-Grain Scalability (FGS) uses an advanced form of bit-plane coding for encoding successive refinements of transform coefficients. The FGS slices have the property that they can be truncated at any byte-aligned position to achieve SNR scalability. FGS SNR scalability has the advantage that it provides a larger degree of flexibility, allowing a quasi-continuous spectrum of achievable bitrates, while CGS is limited to a number of pre-determined bitrates, i.e., one extraction point per layer. Due to its high computational complexity, however, the FGS concept was not included in one of the SVC profiles. As a consequence, it was removed from the joint draft. After further study and complexity reduction, FGS might be included in a future amendment to the current SVC specification.

2.2.6.8 Medium-Grain Scalability

As an alternative to FGS, Medium-Grain Scalability (MGS) was introduced. MGS tackles a number of problems that are encountered for CGS, such as the limited number of rate points, and the lack of flexibility for bitstream adaptation. MGS increases the number of achievable rate points by allowing different quality levels within one dependency layer. The flexibility is improved by allowing the removal of these quality levels at any point in the bitstream. Switching between the number of dependency layers (as is required for CGS), is only allowed at certain pre-defined points. Presently, 16 quality refinement levels are allowed for every dependency layer. In conjunction with CGS, this means that 128 quality extraction points are now achievable for SVC bit-streams.

2.2.6.9 Temporal Scalability

A bit stream provides temporal scalability when the set of corresponding access units can be partitioned into a temporal base layer and one or more temporal enhancement layers with the following property - Let the temporal layers be identified by a temporal layer identifier T, which starts from 0 for the base layer and is increased by 1 from one temporal layer to the next. Then for each natural number k, the bit stream that is obtained by removing all access units of all temporal layers with a temporal layer identifier T greater than k forms another valid bit stream for the given decoder [14]. For hybrid video codecs, temporal scalability can generally be enabled by restricting motion-compensated prediction to reference pictures with a temporal layer identifier that is less than or equal to the temporal layer identifier of the picture to be predicted. H.264/AVC provides a significantly increased flexibility for temporal scalability because of its reference picture memory control. It allows the coding of picture sequences with arbitrary temporal dependencies, which are only restricted by the maximum usable decoded picture buffer (DPB) size. Hence, for supporting temporal scalability with a reasonable number of temporal layers, no changes to the design of H.264/AVC are required. The only related change in SVC refers to the signaling of temporal layers.

2.2.7. Levels in H.264

In H.264 /AVC, 16 levels are specified. Each level defines upper bounds for the bit stream or lower bounds for the decoder capabilities. A profile and level can be combined to define the conformance points. These points signify the point of interoperability for applications with similar functional requirements. The levels defined in H.264 are listed in Table 2.2. The level '1b' was added in the FRExts amendment.

Table 2.2 Levels defined in H.264 (P represents Progressive scanning and I represents interlaced scanning). [38]

Level number	Typical picture size	Typical frame rate	Maximum compression bit rate (for VLC) in Non-FRExt profiles	Maximum number of reference frames for typical picture size
1	QCIF	15	64 kbps	4
1b	QCIF	15	128 kbps	4
1b	QCIF	15	128kbps	4

Table 2.2 - *Continued*

1.1	CIF or QCIF	7.5 (CIF) / 30 (QCIF)	192 kbps	2 (CIF) / 9 (QCIF)
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR	30 / 25	4 Mbps	6
2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD formats	60p / 30i	20 Mbps	4
4.1	HD formats	60p / 30i	50 Mbps	4
4.2	1920x1080p	60p	50 Mbps	4
5	2k x 1k	72	135 Mbps	5
5.1	2k x 1k or 4k x 2k	120 / 30	240 Mbps	5

2.3 H.264 Encoder

Figure 2.6 illustrates the schematic of the H.264 encoder. H.264 encoder works on macroblocks and motion-compensation like most other previous generation codecs. Video is formed by a series of picture frames. Each picture frame is an image which is split down into blocks. The block sizes can vary in H.264. The encoder may perform intra-coding or inter-coding for the macroblocks of a given picture. Intra coded frames are encoded and decoded independently. They do not need any reference frames. Hence they provide access points to the coded sequence where decoding can start. H.264 uses nine spatial prediction modes in intra-coding to reduce spatial redundancy in the source signal of the picture. These prediction modes are explained in section 2.7. Inter-coding uses inter-prediction of a given block from some previously decoded pictures. The aim to use inter-coding is to reduce the temporal redundancy by making use of motion vectors. Motion vectors give the direction of motion of a particular block from the current frame to the next frame. The prediction residuals are obtained which then undergo transformation to remove spatial correlation in the block. The transformed coefficients, thus obtained, undergo quantization. The motion vectors (obtained from inter-prediction) or intra-prediction modes are combined with the quantized transform coefficient information. They are then encoded using entropy code such as context-based adaptive variable length coding (CAVLC) or context-based adaptive binary arithmetic coding (CABAC) [8].

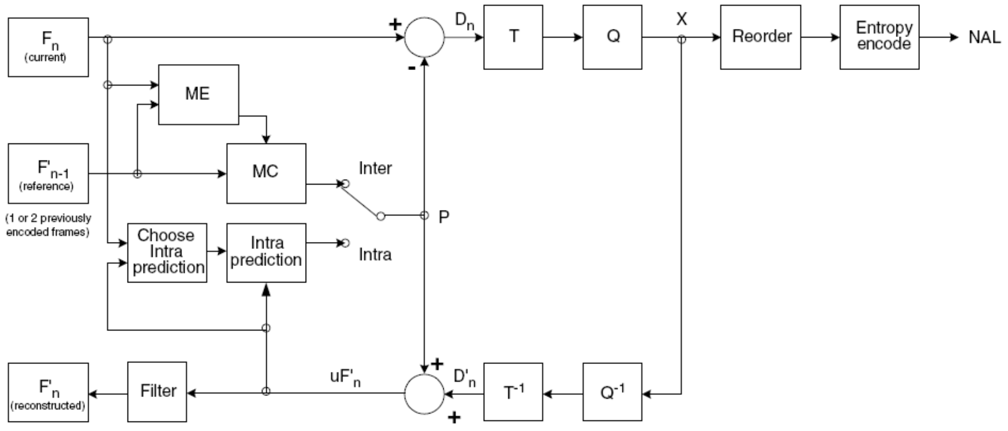


Figure 2.6 H.264 Encoder block diagram [1]

There is a local decoder within the H.264 encoder. This local decoder performs the operations of inverse quantization and inverse transform to obtain the residual signal in the spatial domain. The prediction signal is added to the residual signal to reconstruct the input frame. This input frame is fed in the deblocking filter to remove blocking artifacts at the block boundaries. The output of the deblocking filter is then fed to inter/intra prediction blocks to generate prediction signals.

The various coding tools used in the H.264 encoder are explained in the sections 2.7 through 2.12.

2.3.1. Intra-Prediction

Intra-prediction uses the macroblocks from the same image for prediction. Two types of prediction schemes are used for the luminance component. These two schemes can be referred as INTRA_4x4 and INTRA_16x16 [16]. In INTRA_4x4, a macroblock of size 16x16 pixels are divided into 16 4x4 sub blocks. Intra prediction scheme is applied individually to these 4x4 sub blocks. There are nine different prediction modes supported as shown in Figure 2.7. In the FRExts profiles alone, there is also 8x8 luma spatial prediction (similar to 4x4 spatial prediction) and with low-pass filtering of the prediction to improve prediction performance.

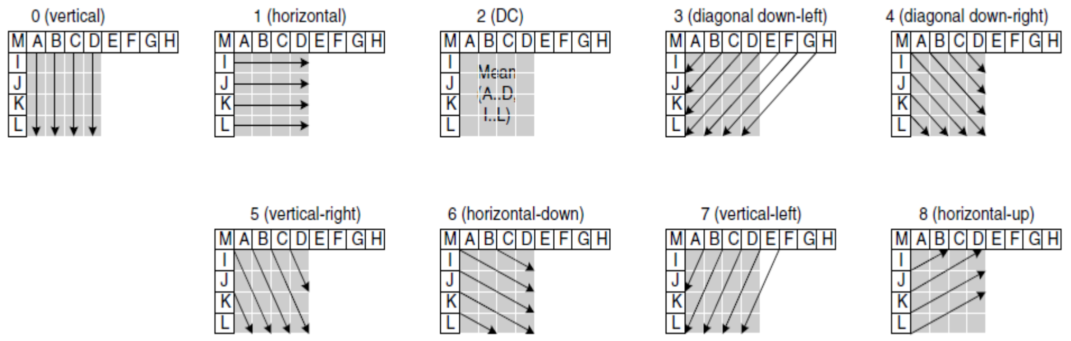


Figure 2.7 7 4x4 Luma prediction (intra-prediction) modes in H.264 [1]

In mode 0, the samples of the macroblock are predicted from the neighboring samples on the top. In mode 1, the samples of the macroblock are predicted from the neighboring samples from the left. In mode 2, the mean of all the neighboring samples is used for prediction. Mode 3 is in diagonally down-left direction. Mode 4 is in diagonal down-right direction. Mode 5 is in vertical-right direction. Mode 6 is in horizontal-down direction. Mode 7 is in vertical-left direction. Mode 8 is in horizontal up direction. The predicted samples are calculated from a weighted average of the previously predicted samples A to M.

For prediction of 16x16 intra prediction of luminance components, four modes are used. The three modes of mode 0 (vertical), mode 1 (horizontal) and mode 2 (DC) are similar to the prediction modes for 4x4 block. In the fourth mode, the linear plane function is fitted in the neighboring samples.

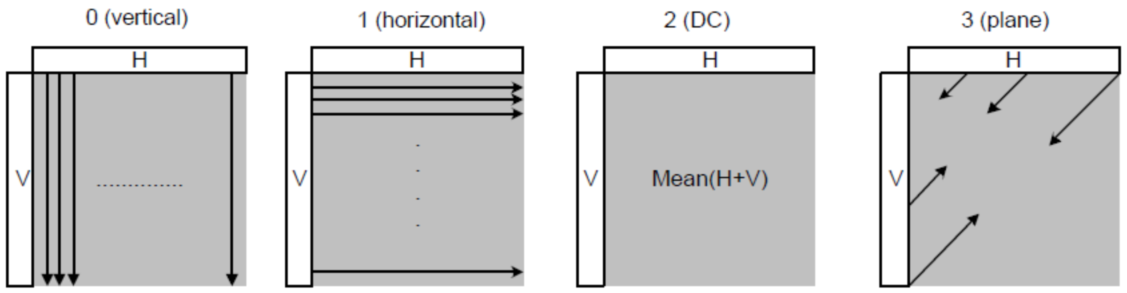


Figure 2.8 4x4 Luma prediction (intra-prediction) modes in H.264 [1]

Figure 2.8 The chroma macroblock is predicted from neighboring chroma samples. The four prediction modes used for the chroma blocks are similar to 16x16 luma prediction modes. The number in which the prediction modes are ordered is different for the chroma macroblock: mode 0 is DC, mode 1 is horizontal, mode 2 is vertical and mode 3 is plane. The block sizes for the chroma prediction depend on the sampling format. For 4:2:0 format, the 8x8 size of the chroma block is selected. For the 4:2:2 format, the 8x16 size of the chroma block is selected. For the 4:4:4 formats; the 16x16 size of chroma block is selected [1].

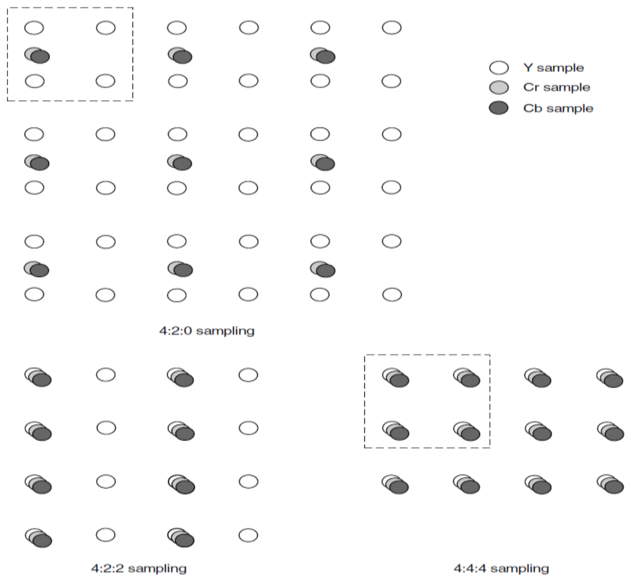
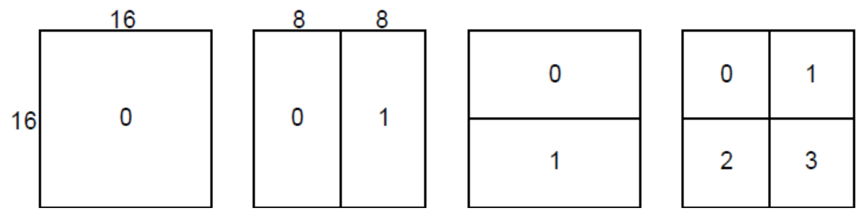


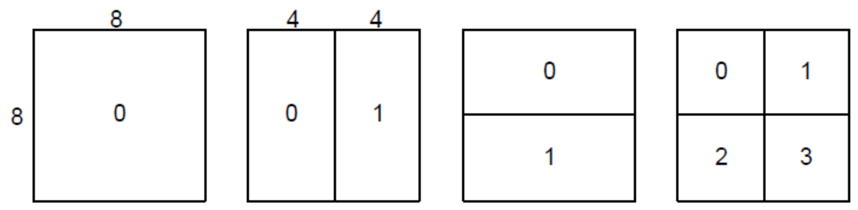
Figure 2.9 Chroma sub sampling [1]

2.3.2. Intra-Prediction

Inter-prediction is used to capitalize on the temporal redundancy in a video sequence. The temporal correlation is reduced by inter prediction through the use of motion estimation and compensation algorithms [1]. An image is divided into macroblocks; each 16x16 macroblock is further partitioned into 16x16, 16x8, 8x16, 8x8 sized blocks. A 8x8 sub-macroblock can be further partitioned into 8x4, 4x8, 4x4 sized blocks. Figure 2.9 illustrates the partitioning of a macroblock and a sub-macroblock [1]. The input video characteristics govern the block size. A smaller block size ensures less residual data; however smaller block sizes also mean more motion vectors and hence more number of bits required to encode these motion vectors [1].



(a)



(b)

Figure 2.10 Macroblock portioning in H.264 for inter prediction [1] (a) (L-R) 16x16, 8x16, 16x8, 8x8 blocks; (b) (L-R) 8x8, 4x8, 8x4, 4x4 blocks

Each partition or sub-macroblock partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has quarter-sample resolution for the luma component and one-eighth-sample resolution for the chroma components. The luma and chroma samples at sub-sample positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby coded samples. Figure 2.10 and Figure 2.11 illustrate half and quarter pixel interpolation used in luma pixel interpolation respectively. Six-tap filter with weights (1, −5, 20, 20, −5, 1)/32 is used to derive half-pel luma sample predictions, for sharper sub pixel motion-compensation. Quarter-pixel motion is derived by linear interpolation of the half-pel values, to save processing power.

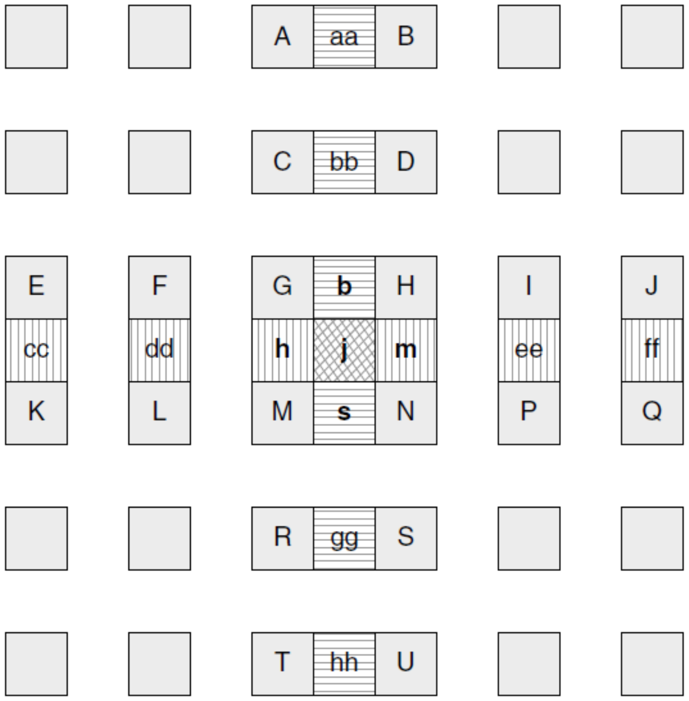


Figure 2.11 Interpolation of luma half-pel positions [1]

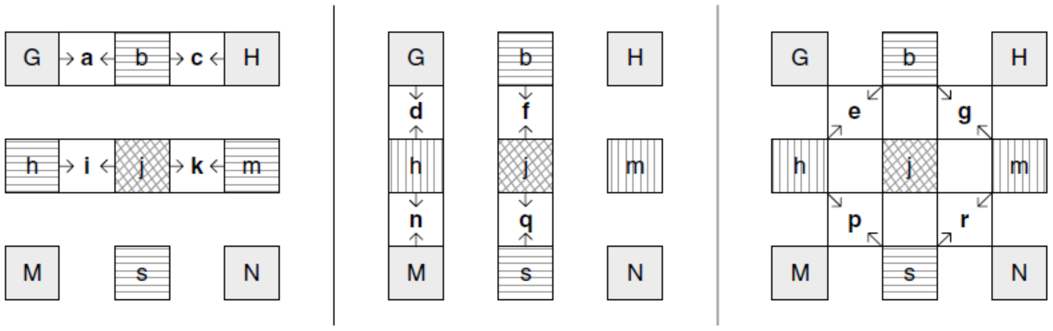


Figure 2.12 Interpolation of luma quarter-pel positions [1]

The reference pictures used for the inter prediction are previously decoded frames and are stored in the picture buffer. H.264 supports the use of multiple frames as reference frames. This is implemented by the use of an additional picture reference parameter which is transmitted along with the motion vector. Figure 2.12 illustrates an example with 4 reference pictures.

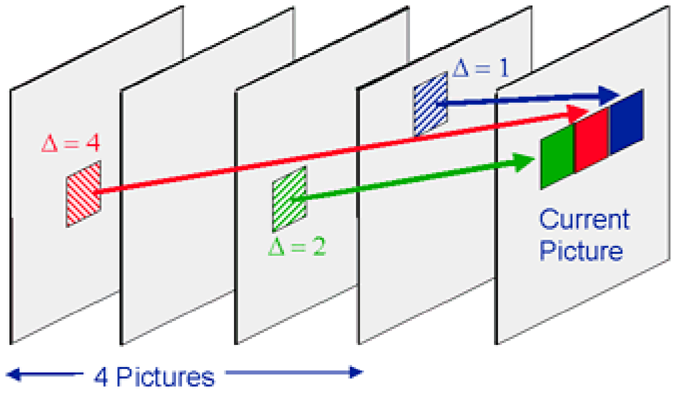


Figure 2.13 Motion compensated prediction with multiple reference frames [1]

2.3.3. Transform Coding

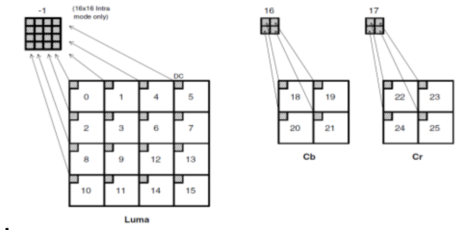
There is high spatial redundancy among the prediction error signals. The H.264 implements a block-based transform to reduce this spatial redundancy [1]. The former standards of MPEG-1 and MPEG-2 employed a two dimensional discrete cosine transform (DCT) [36] for the purpose of transform coding of the size 8x8 [1]. H.264 uses integer transforms instead of the DCT. The size of these transforms is 4x4 [1]. The advantages of using a smaller block size in H.264 are stated as follows:

- The reduction in the transform size enables the encoder to better adapt the prediction error coding to the boundaries of the moving objects and to match the transform block size with the smallest block size of motion compensation.

- The smaller block size of the transform leads to a significant reduction in the ringing artifacts.

- The 4x4 integer transform has the benefit for removing the need for multiplications.

H.264 employs a hierarchical transform structure, in which the DC coefficients of neighboring 4x4 transforms for luma and chroma signals are grouped into 4x4 blocks (blocks -1, 16 and 17) and transformed again by the Hadamard transform as shown in Figure 2.14 (a). As shown in Figure 2.14(b) the first transform (matrix H1 in Figure 2.14(e)) is applied to all samples of all prediction error blocks of the luminance component (Y) and for all blocks of chrominance components (Cb and Cr). For blocks with mostly flat pixel values, there are 25 significant correlation among transform DC coefficients of neighboring blocks. Hence, the standard specifies the 4x4 Hadamard transform (matrix H2 in Figure 2.14 (e)) for luma DC coefficients (Figure 2.14 (c)) for 16x16 intra-mode only, and 2x2 Hadamard transform as shown in Figure 2.14 (d) (matrix H3 in Figure 2.14 (e)) for chroma DC coefficients



(a)

$$\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}\begin{bmatrix} & & & \\ & \mathbf{X} & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix}1 & 2 & 1 & 1\\ 1 & 1 & -1 & -2\\ 1 & -1 & -1 & 2\\ 1 & -2 & 1 & -1\end{bmatrix}\right)$$

(b)

$$\mathbf{Y}_D = \left(\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix} & & & \\ & \mathbf{W}_D & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\right)/2$$

(c)

$$\mathbf{W}_{QD} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\mathbf{W}_D\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

(d)

$$H_1 = \begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix} \quad H_2 = \begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix} \quad H_3 = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

(e)

Figure 2.14 H.264 Transformation (a)DC coefficients of 16 4x4 luma blocks for 4 4x4 Cb and Cr blocks [1] (b) Matrix H1 is applied to 4x4 block of luma/chroma coefficients X (c) [8](c) Matrix H2 (4x4 Hadamard transform) applied to luma DC coefficients WD [8] (d) Matrix H3 (2x2 Hadamard transform) applied to chroma DC coefficients WD [8] e) Matrices H1, H2 and H3 of the three transforms used in H.264 [8]

2.3.4. Deblocking Filter

The deblocking filter is used to remove the blocking artifacts due to the block based encoding pattern. The transform applied after intra-prediction or inter-prediction is on blocks; the transform coefficients then undergo quantization. These block based operations are responsible for blocking artifacts which are removed by the in-loop deblocking filter. It reduces the artifacts at the block boundaries and prevents the propagation of accumulated noise. The presence of the filter however adds to the complexity of the system [1]. Figure 2.15 illustrates a macroblock with sixteen 4x4 sub blocks along with their boundaries.

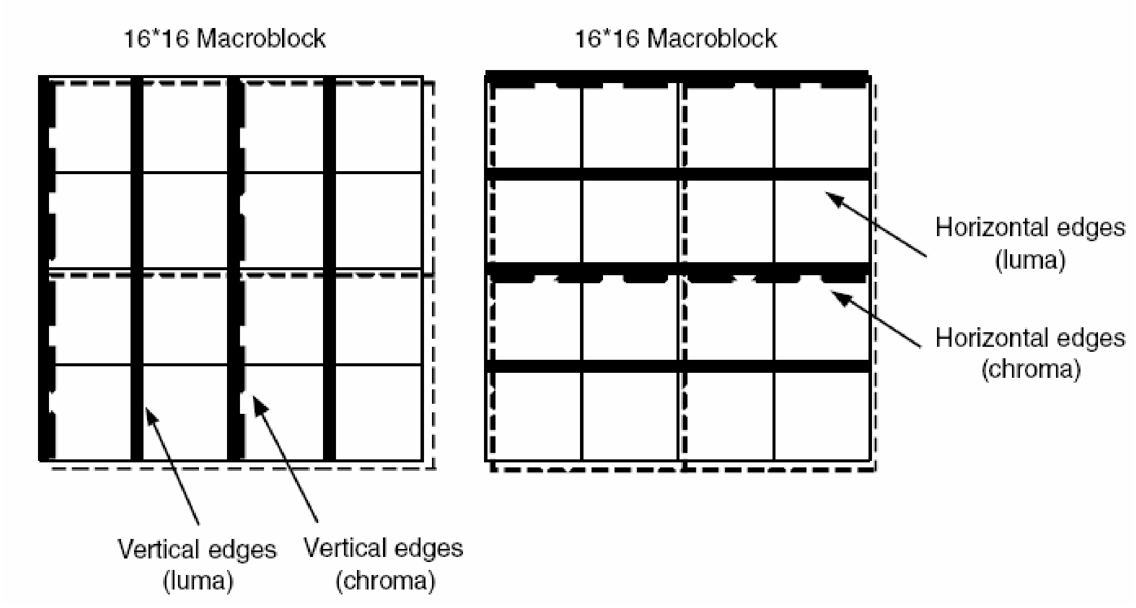


Figure 2.15 Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines and chroma boundaries shown with dotted lines)

As shown in the Figure 2.15, the luma deblocking filter process is performed on the 16 sample edges – shown by solid lines. The chroma deblocking filter process is performed on 8 sample edges – shown in dotted lines.

H.264 employs deblocking process adaptively at the following three levels:

- At slice level – global filtering strength is adjusted to the individual characteristics of the video sequence

- At block-edge level – deblocking filter decision is based on inter or intra prediction of the block, motion differences and presence of coded residuals in the two participating blocks.

- At sample level – it is important to distinguish between the blocking artifact and the true edges of the image. True edges should not be de blocked. Hence decision for deblocking at a sample level becomes important.

2.3.5. Entropy Coding

H.264 uses variable length coding to match a symbol to a code based on the context characteristics. All the syntax elements except for the residual data are encoded by the Exp-Golomb codes [1]. The residual data is encoded using the Context adaptive Variable Length Coding (CAVLC). The main and the high profiles of H.264 use CABAC.

- Context-based adaptive variable length coding (CAVLC):

  After undergoing transform and quantization the probability that the level of coefficients is zero or +1 is very high [1]. CAVLC handles these values differently. It codes the number of zeroes and +1. For other values, their values are coded.

- Context-based adaptive binary arithmetic coding (CABAC):

  This technique uses the arithmetic encoding to achieve good compression. The schematic for CABAC is shown in Figure 2.16.

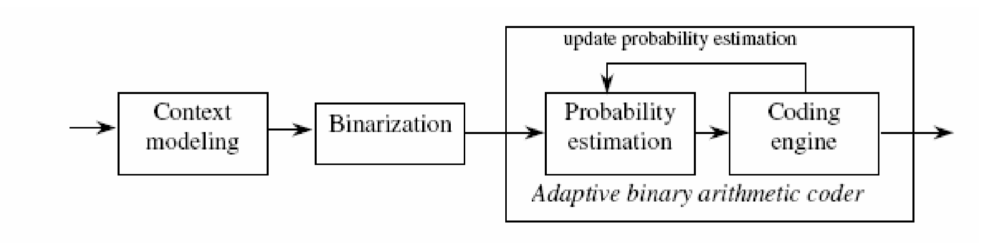


Figure 2.16 Schematic block diagram of CABAC [1]

CABAC consists of three steps:

Step 1: Binarization: A non-binary value is uniquely mapped to a binary sequence

Step 2: Context modeling: A context model is a probability model for one or more elements of binarized symbol. The probability model is selected such that the corresponding choice depends on previously encoded syntax elements.

Step 3: Binary arithmetic coding: An arithmetic encoder encodes each element according to the selected probability model.

2.3.6. B-slices and adaptive weighted prediction

Bi-directional prediction which uses both past and future frames for reference can be very useful in improving the temporal prediction. Bi-directional prediction in H.264 uses multiple reference frames. Figure 2.17 show bidirectional prediction from multiple reference frames. The standards, before H.264, with B pictures use the bidirectional mode, with limitation that it allows the combination of a previous and subsequent prediction signals. In the previous standards, one prediction signal is derived from subsequent inter-picture, another from a previous picture, the other from a linear averaged signal of two motion compensated prediction signals.

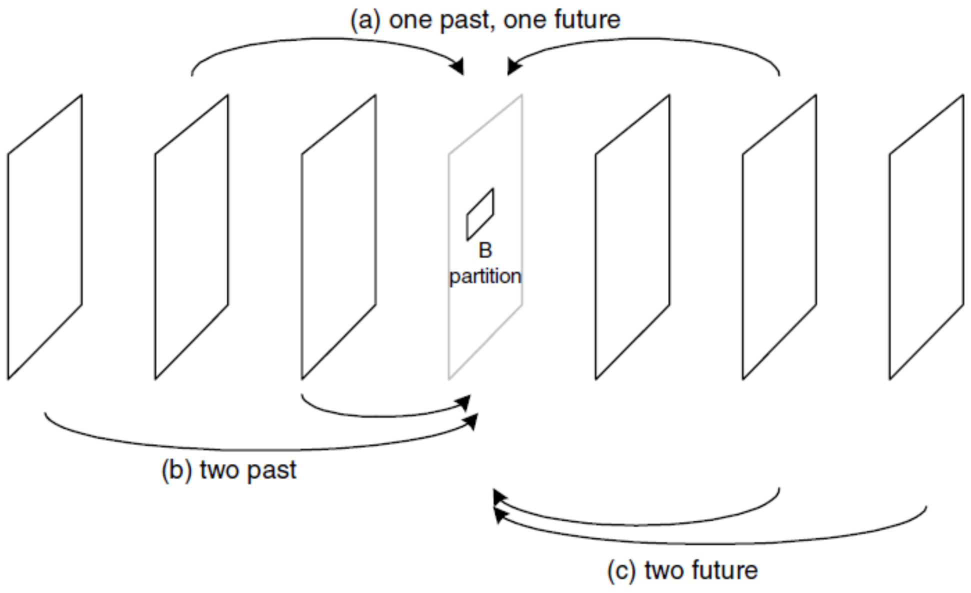


Figure 2.17 Partition prediction examples in a B macroblock type: (a) past/future, (b) past, (c) future [1]

H.264 supports the forward/backward prediction pair and also supports forward/forward and backward/backward prediction pair [1]. Figures 2.17 (a) and 2.17 (b) describe the scenario where bidirectional prediction and multiple reference frames respectively are applied and a macroblock is thereby predicted as a linear combination of multiple reference signals using weights as described in (2.1). Considering two forward references for prediction is beneficial for motion compensated prediction of a region just before scene change. Considering two backward reference frames is beneficial for frames just after scene change. H.264 also allows bi-directionally predictive-coded slice which may also be used as references for inter-coding of other pictures. Except for the H.264 codec, all the existing standards consider equal weights for reference pictures. Equal weights of reference signals are averaged and the prediction signal is obtained. H.264 also uses weighted prediction [1]. It can be used for a macroblock of P slice or B slice. Different weights can be assigned to two different reference signals and the prediction signal is calculated as follows:

$$p = w1 * r1 + w2 * r2 \quad (1)$$

In(1), p is the prediction signal, r1 and r2 are the reference signals and w1 and w2 are the prediction weights.

2.3.6. H.264 Decoder

The H.264 decoder works similar in operation to the local decoder of H.264 encoder. An encoded bit stream is the input to the decoder. Entropy decoding (CABAC or CAVLC) takes place on the bit stream to obtain the transform coefficients. These coefficients are then inverse scanned and inverse quantized. This gives residual block data in the transform domain. Inverse transform is performed to obtain the data in the pixel domain. The resulting output is a 4x4 blocks of residual signal. Depending on inter predicted or intra-predicted, an appropriate prediction signal is added to the residual signal. For an inter-coded block, a prediction block is constructed depending on the motion vectors, reference frames and previously decoded pictures. This prediction block is added to the residual block to reconstruct the video frames. These reconstructed frames then undergo deblocking before they are stored for future use for prediction or being displayed. Figure 2.18 illustrates the decoder.

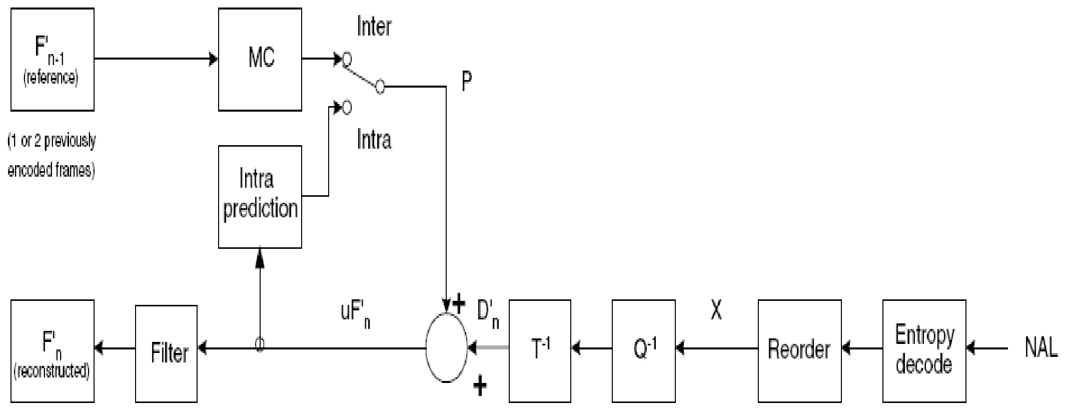


Figure 2.18 H.264 Decoder block diagram [1]

2.4 Summary

This chapter outlines the coding tools of H.264 codec. The next chapter describes the coding tools used in Machine learning.

CHAPTER 3

INTRODUCTION

Machine learning shows how computers are used to simulate human learning activities, and to study self-improvement methods of computers Figure 3. Compared with human learning, machine learning is a faster process. The accumulation of knowledge (knowledge here refers to the information about the system behavior in response to various inputs to the system.) is to facilitate faster and easier learning. So, any progress in the field of machine learning will enhance the capability of computers and thus have an impact on human society. In the process of machine learning, the quality of information that external environment provides to the system is the primary factor [51].

The external environment is referred to the information set that delivers itself to the learning process in some form. It represents sources of outside information; Learning is the process that processes the outside information to knowledge, first it obtains information from the outside environment and then processes the information to provide knowledge, and puts this knowledge into a repository; the repository stores many general principles that guide the implementation action. The environment provides all kinds of information for the learning system, the quality of information impacts directly on the learning realization. The repository is the second factor that impacts the design of a learning system. The expression of knowledge can come in the form of eigenvector, logic statements of the first order, production rules, semantic networks, frameworks and so on; these fashions of expression have strong and weak points.

The important aspects that need to be considered are the robustness of expression, simplicity of the expression, ease in the repository manipulation and the ease with which the knowledge can be expanded. The implementation is the process that uses the knowledge of repository to complete a certain task and to feed back the information which is obtained in the process of completing the task to the process of learning, and to guide further study.

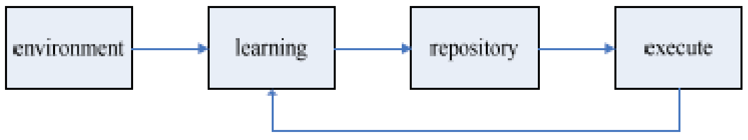


Figure 3.1 Basic model of machine learning [51]

.

3.1 Machine Learning Methods

3.1.1. Rote learning [51]

Rote learning is a learning process where memory is involved; the process stores the new knowledge and calls for it when necessary. A calculation and reasoning a learning system need not remember any of the knowledge. In the rote learning system, knowledge is called in a direct way. The system does not require much processing. The implementation part of learning system can be considered abstractly as a function. The learning system receives the input variables and calculates the output values of the functions. The input and output values are stored. On arrival of the next input variable, the rote learning system searches the memory rather than re-calculating the knowledge. Memory is the critical parameter in Rote Learning rather than the learning process.

3.1.2. Inductive Learning [52]

Inductive consequence applies inductive methods to summarize general knowledge from sufficient specific examples, and to distill a general law of things. This process extrapolates form the individual to general. Learning can be based on the guidance of pre-determined rules or not. Inductive learning can be divided into learning by examples and learning by observation.

The former is learning with the rules; the latter is learning without the rules. The model of a learning system is shown in Figure 3.2.

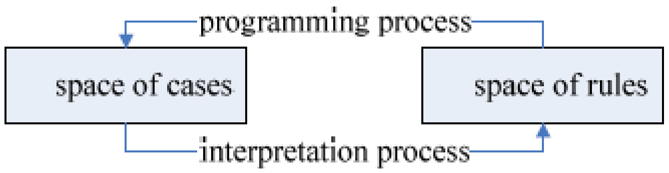


Figure 3.2 The model of learning system [51]

The learning by examples method concludes the learning methods by examples. In this method, the general concepts are learnt through the examples related with a concept in the environment. Take a group of animals as an example, and tell the learning system which animal is "Ma", and which is not. When the sample is enough, the learning system can conclude the conceptual model about “Ma”, so it can identify “Ma”, and can distinguish “Ma” from other animals. The learning by observation is also known as the descriptive summary, has as its goal to identify a general description of a law or a theory. It depicts an observation set and specifies the nature of certain objects.

3.1.3. Analogy Learning [51]

Analogy describes the similarity between objects clearly and concisely. Analogy learning carries out learning by comparing similar things. For example, when a professor wants to teach a new concept which is more difficult to understand for the students, he or she always use some of the examples that students have mastered that are similar to the new concept to be learned.

3.1.4. Explained Learning

The learning based on interpretation is called explained learning. This form of learning analyzes and classifies the current instances by the knowledge already known while it continues to learn. This type of learning generally yields a classification tree; which classifies any new instance. It learns new knowledge by considering the relationship between the various factors affecting the knowledge. The generalized process based on interpretation is shown in Figure 3.3.

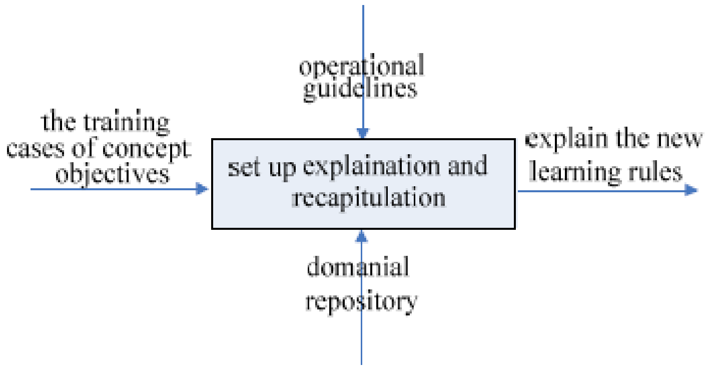

operational guidelines
the training cases of concept objectives
set up explaination and recapitulation
explain the new learning rules
domanial repository

Figure 3.3 The generalized process based on interpretation [51]

3.1.5. Learning Based on Neural Network

The nature of a neural network depends on three main factors: the topology structure of network, right values and work rules of network. Combination of the three can form the main characters of a network. The learning problem of a neural network is the problem of adjustment of network values. There are two ways to determine the value of a neural network: one is determined through the design calculations; another is determined by the study of the network through certain rules. Most neural networks use the second method to determine the value of its network. The well-known network model and learning algorithm contain the back propagation arithmetic, The Hopfield network and so on [51] Figure 3.4 describes the approach.

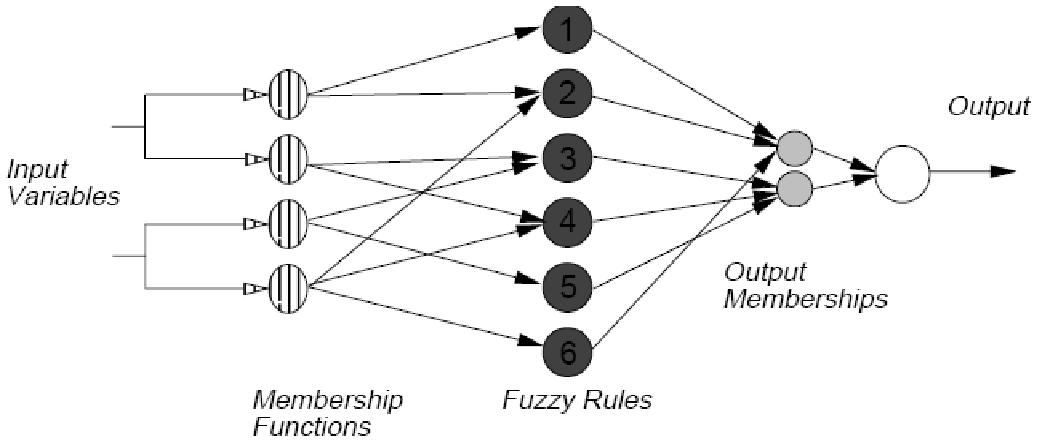


Figure 3.4 Network constructed from fuzzy rules [55]

In Figure 3.4, the input variables are fed into the first layer, which contains nodes which act as membership functions, responding only in a certain region of the input variable domain, with a result between zero and one. The second layer is formed by the rules themselves. The nodes in this layer perform the fuzzy ‘AND’ operation, computing a minimum of their inputs. The third layer contains a node for each output fuzzy. Finally, the last layer contains a single node which performs a centroid defuzzification of the outputs. This node normalizes the weights from the previous layer and weights them with the centers of the output membership functions to calculate the actual output. [55]

3.1.6. Knowledge Discovery [51]

Knowledge discovery of repository is a senior managed process to identify effective, novel, potential and a useful model from large amounts of data. The discovery process of knowledge is shown in Figure 3.5

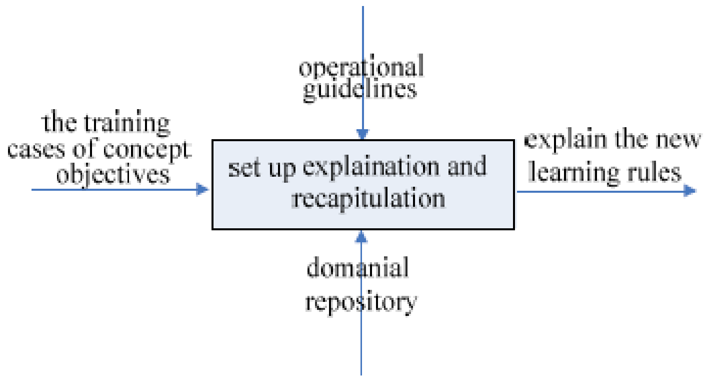

operational guidelines
the training cases of concept objectives
set up explaination and recapitulation
explain the new learning rules
domanial repository

Figure 3.5 The discovery process of knowledge [51]

Data selection extracts relevant data from the database based on the needs of users. Pre-processing checks data's integrity and consistency, processes the noisy data, fills up the loss of data by statistical methods, and forms explored database. Data transformation chooses data from explored database, the methods of transformation mainly use discrimination analysis and clustering analysis. Data mining identifies the goal of what type of knowledge is to discover based on the user's requirements. It then extracts the knowledge which users need from the database. Knowledge evaluation mainly evaluates the acquired rules, and then determines if the rules have to be added to the basic knowledge database. Knowledge discovery process can be generalized in three steps. That is, pretreatment of data mining, data mining and post treatments of data mining. Knowledge discovery has been successfully applied in the banking, insurance, retail, healthcare, engineering and manufacturing, scientific research, satellite observation, entertainment industries and so on. It has provided a great help in people's scientific decision-making.

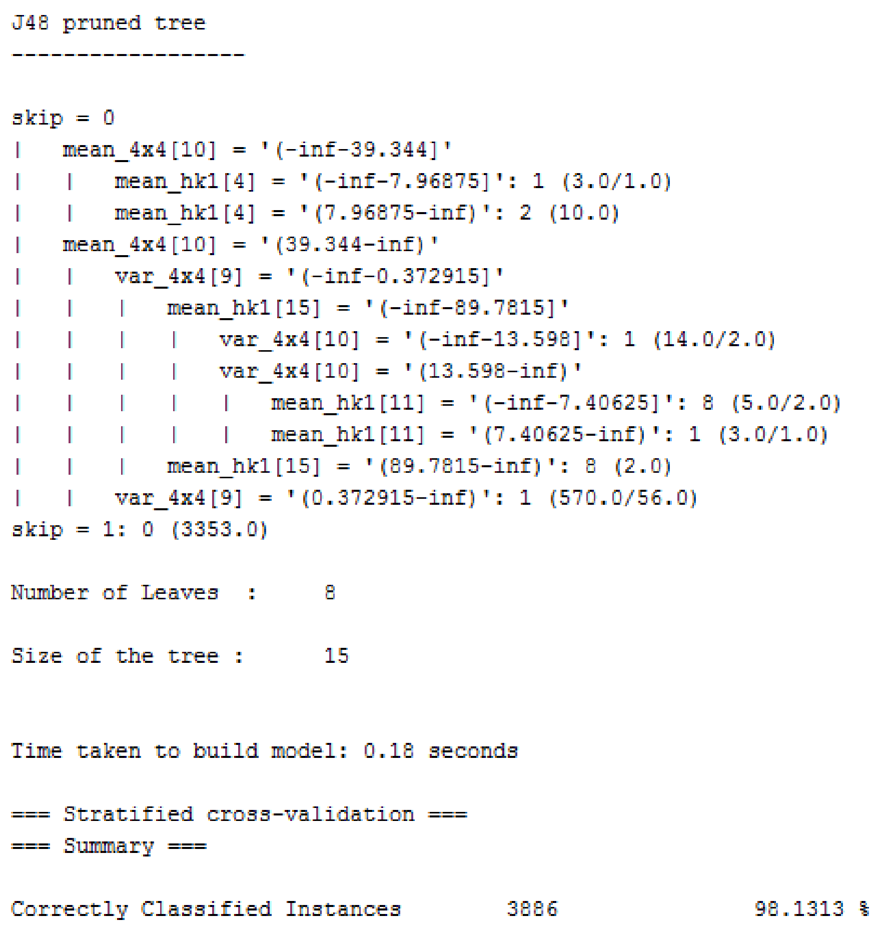
3.2 Application of Machine Learning [51]

Research shows that machine learning technology has been widely used in marketing, finance, telecommunications and network analysis. [51] In the field of marketing, machine learning technology is more widely used in the area of tasks classification and related fields; in the field of finance, machine learning technology is more widely used in tasks of forecasts; in the field of telecommunications, machine learning technology is widely used in the tasks of classification and prediction. In addition, machine learning is also applied to the field of data mining combination with other applications. The typical methods are based on the neural network initialization, the application of evolutionary computation in machine learning research and the study of level classification of machine learning.

3.3 Weka

Machine learning has the potential to become one of the key components of intelligent information systems, enabling compact generalizations, inferred from large databases of recorded information, to be applied as knowledge in various practical ways; such as being embedded in automatic processes like expert systems, or used directly for communicating with human experts and for educational purposes. Presently, however, the field is not well placed to do this. Most research effort is directed towards the invention of new algorithms for learning, rather than towards gaining experience in applying existing techniques to real problems. The WEKA project (Waikato Environment for Knowledge Analysis) [52] is redressing the balance by applying standard machine learning techniques to a variety of agricultural and horticultural problems. The goal of WEKA is to discover and characterize what is required for successful applications of machine learning to real-world data. To support this effort, a workbench has been developed to provide an integrated environment which not only gives easy access to a variety of machine learning techniques through an interactive interface, but also incorporates those pre- and post-processing tools that is found to be essential when working with real-world data sets. Other systems for machine learning experimentation exist, but these are libraries of routines that are intended for use by a researcher who is extending and comparing algorithms. One exception although still a library of modules is Consultant, an expert system that allows domain experts to choose a learning algorithm suited to their needs. Consultant assumes that a machine learning algorithm exists that can be applied directly to solve the problem at hand. Although a suitable algorithm may well exist, it is unlikely that its direct application on the domain expert’s data will produce a meaningful result. Domain experts need an environment in which they can easily manipulate data and run experiments themselves. [52]

The philosophy behind WEKA is to move away from supporting a computer science or machine learning researcher, and towards supporting the end user of machine learning. The end user is someone typically, an agricultural scientist-with an understanding of the data and sufficient knowledge of the capabilities of machine learning to select and investigate the application of different techniques. In order to maintain this philosophy, WEKA concentrates on ensuring that the implementation details of the machine learning algorithms and the input formats they require are hidden from the user.



```
J48 pruned tree
------------------

skip = 0
|   mean_4x4[10] = '(-inf-39.344]'
|   |   mean_hk1[4] = '(-inf-7.96875]': 1 (3.0/1.0)
|   |   mean_hk1[4] = '(7.96875-inf)': 2 (10.0)
|   mean_4x4[10] = '(39.344-inf)'
|   |   var_4x4[9] = '(-inf-0.372915]'
|   |   |   mean_hk1[15] = '(-inf-89.7815]'
|   |   |   |   var_4x4[10] = '(-inf-13.598]': 1 (14.0/2.0)
|   |   |   |   var_4x4[10] = '(13.598-inf)'
|   |   |   |   |   mean_hk1[11] = '(-inf-7.40625]': 8 (5.0/2.0)
|   |   |   |   |   mean_hk1[11] = '(7.40625-inf)': 1 (3.0/1.0)
|   |   |   mean_hk1[15] = '(89.7815-inf)': 8 (2.0)
|   |   var_4x4[9] = '(0.372915-inf)': 1 (570.0/56.0)
skip = 1: 0 (3353.0)

Number of Leaves  :     8

Size of the tree :      15


Time taken to build model: 0.18 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3886               98.1313 %
```

Figure 3.6 Classification tree for Container sequence from Weka tool.

3.4 C4.5 Algorithm

3.4.1. Algorithm

The C4.5 [54] algorithm creates a model based on a tree structure. Nodes in the tree represent features, with branches representing possible values connecting features. A leaf representing the class terminates a series of nodes and branches. Determining the class of 'n' number of instance is a matter of tracing the path of nodes and branches to the terminating leaf. C4.5 uses the 'divide and conquer' method to construct a tree from a set S of training instances. If all instances in S belong to the same class, the decision tree is a leaf labeled with that class. Otherwise the algorithm uses a test to divide S into several non-trivial partitions. Each of the partitions becomes a child node of the current node and the tests separating S are assigned to the branches. C4.5 uses two types of tests each involving only a single attribute A. For discrete attributes the test is to test the value of A with one outcome for each value of A. For numeric attributes the test is $A \leq \theta$ where θ is a constant threshold. Possible threshold values are found by sorting the distinct values of A that appear in S and then identifying a threshold between each pair of adjacent values. For each attribute a test set is generated. To find the optimal partitions of S C4.5 relies on greedy search and in each step selects the test set that maximizes the entropy based gain ratio splitting criterion.

The 'divide and conquer' approach partitions until every leaf contains instances from only one class or further partition is not possible for example, two instances have the same features but different class. If there are no conflicting cases the tree will correctly classify all training instances. However, this over-fitting decreases the prediction accuracy on unseen instances. C4.5 attempts to avoid over-fitting by removing some structure from the tree after it has been built. Pruning is based on estimated true error rates. After building a classifier the ratio of misclassified instances and total instances can be viewed as the real error. However this error is minimized as the classifier was constructed specifically for the training instances. Instead of using the real error the C4.5 pruning algorithm uses a more conservative estimate, which is the upper limit of a confidence interval constructed around the real error probability. With a given confidence CF the real error will be below the upper limit with 1-CF. C4.5 uses sub tree replacement or subtree rising to prune the tree as long as the estimated error can be decreased. In our test the confidence level is 0.25 and the minimum number of instances per leaf is set to two. We use subtree replacement and subtree rising when pruning.

3.4.2. Flow and Feature Definitions [57]

The tool can calculate the attributes value of the network traffic. The tool forms packs which have the same source IP and source port, destination IP, destination port and protocols as a flow. In order to facilitate understanding, based on the definition of IP Packet header, the following features were found and became the base feature set for the experiments:

- Flow duration
- Flow size in bytes and packets
- Packet length (minimum, mean, maximum and standard deviation)
- Inter-arrival time between packets (minimum, mean, maximum and standard deviation)

Packet lengths are based on the IP length excluding link layer overhead. Inter-arrival times have at least microsecond precision and accuracy. As the traces contained both directions of the flows, features were calculated in both directions (except protocol and flow duration).This produces a total of 37 flow features, which we refer to as the full feature set. Our features are simple and well understood within the network community. They represent a reasonable benchmark feature set. More complex features might be added in the future.

3.4.3. Feature Selection Algorithms

In order to reduce the amount of computation, as well as other resources (such as memory) consumption, and raise the classification rate. The use of CFS (correlation feature selection) and genetic search method to select some of the most representative attributes from the dozens of attributes of the flow to form a subset. The CFS algorithm uses an evaluation heuristic that examines the usefulness of individual features along with the level of inter correlation among the features. High scores are assigned to subsets containing attributes that are highly correlated with the class and have low inter-correlation with each other. Conditional entropy is used to provide a measure of the correlation between features and class and between features. If H(X) is the entropy of a feature X and H(X|Y) the entropy of a feature X given the occurrence of feature Y the correlation between two features X and Y can then be calculated using the symmetrical uncertainty:

$$C\left(X/Y\right) = \frac{H(X) - H\left(X/Y\right)}{H(Y)} \quad (2)$$

The class of an instance is considered to be a feature. The goodness of a subset is then determined as:

$$G_{subset} = \frac{k\overline{r_1}}{\sqrt{k + k(k-1)\overline{r_2}}} \quad (3)$$

where k is the number of features in a subset, $\overline{r_1}$ is the mean feature correlation with the class and $\overline{r_2}$ is the mean feature correlation.

When $\overline{r_1}$ is larger and the $\overline{r_2}$ is smaller, the classification results of subset is better.

3.4.4. The C4.5 Tree-Construction Algorithm [53]

The algorithm constructs a decision tree starting from a training set S, which is a set of cases or tuples in the database terminology. Each case specifies values for a collection of attributes and for a class. Each attribute may have either discrete or continuous values. Moreover, the special value unknown is allowed to denote unspecified values. The class may have only discrete values. We denote with C_1,……,C_{NClass} the values of the class.

Decision Trees: A decision tree is a tree data structure consisting of decision nodes and leaves. A leaf specifies a class value. A decision node specifies a test over one of the attributes, which is called the attribute selected at the node. For each possible outcome of the test, a child node is present. In particular, the test on a discrete attribute A has h possible outcomes $A = d_i$ where $i \in 1, n$ and d_i are the known values for attribute A. The test on a continuous attribute has two possible outcomes, $A \leq th$ and $A > th$, where th is a value determined at the node and called the threshold.

A decision tree is used to classify a case, i.e., to assign a class value to a case depending on the values of the attributes of the case. In fact, a path form the root to a leaf of the decision tree can be followed based on the attribute values of the case. The class specified at the leaf is the class predicted by the decision tree. A performance measure of a decision tree over a set of cases is called classification error. It is defined as the percentage of misclassified cases, i.e., of cases whose predicted classes differ from the actual classes.

The C4.5 algorithm constructs the decision tree with a divide and conquers strategy. In C4.5, each node in a tree is associated with a set of cases. Also, cases are assigned weights to take into account unknown attribute values. At the beginning, only the root is present and associated with the whole training set ΓS and with all case weights equal to 1.0 .At each node, the following divide and conquer algorithm is executed , trying to exploit the locally best choice, with no backtracking allowed.

Program 1: Pseudo code of the C4.5 Tree-Construction Algorithm

```
(1)Compute ClassFrequency(T);
(2)if 'OneClass' or 'FewCases'
   Return a leaf;
Create a decision node N;
(3)For Each Attribute A
   ComputeGain(A);
(4)N.test=AttributeWithBestGain;
(5)if N.test is continuous;
```

```
  Find threshold;
(6)For Each T' in the splitting of T
(7)if T' is Empty
  Child of N is a leaf
Else
(8) Child of N =FormTree(T');
(9) Compute Errors of N;
   Return N
```

Let T be the set of cases associated at the node. The weighted frequency freq (C_i, T) is computed (Step (1))) of cases in T whose class is C_i for $i \in 1, Nclass$.

In all cases (Step (2)) in T belong to the same class as C_j (or the number of cases in T is less than a certain value), then the node is a leaf, with associated class C_j (respectively, the most frequent class). The classification error of the leaf is the weighted sum of cases in T whose class is not C_j(respectively, the most frequent class).If T contains cases belonging to two or more classes (Step (3)), then the information gain of each attribute is calculated. For discrete attributes, the information gain is relative to the splitting of cases in T into sets with distinct attribute values. For continuous attributes, the information gain is relative to the splitting of T into two subsets, namely, cases with an attribute value not greater than a local threshold and cases with an attribute value greater than a certain local threshold, which is determined during information gain calculation. The attribute with the highest information gain (Step (4)) is selected for the test at the node. Moreover, in case a continuous attribute is selected, the threshold is computed (Step (5)) as the greatest value of the whole training set that is below the local threshold.

A decision node has s children if T_i (where i[1,s]) are the sets of the splitting produced by the test on the selected attribute (Step(6)). Obviously, s=2 when the selected attribute is continuous, and s=h for discrete attributes with h known values. For i=[1,s], if T_i is empty, (Step(7)) the child node is directly set to be a leaf, with associated class the most frequent class at the parent node and classification error 0.

If T_i is not empty, the divide and conquer approach consists of recursively applying the same operations (Step (8)) on the set consisting of T_i plus those cases in T with an unknown value of the selected attribute. Note that cases with an unknown value of the selected attribute are replicated in each child with their weights proportional to the proportion of cases in T_i over cases in T with a known value of the selected attribute. Finally, the classification error (Step (9)) of the node is calculated as the sum of the error of classifying all cases in T as belonging to the most frequent class in T, then the node is set to be a leaf and all sub trees are removed.

The information gain of an attribute a for a set of cases T is calculated as follows: if a is discrete, and T_i (where i [1,s]) are the subsets for T consisting of cases with distinct known value for attribute a, then:

$$\text{gain} = \text{info}\ T \ - \sum_{i=1}^{s} \frac{|T_i|}{|T|} \times \text{info}\ T_i$$

Where

$$\text{info}(T) = -\sum_{j=1}^{NClass} \frac{freq\ C_j, T}{|T|} \times \log_2 \left(\frac{freq\ C_j, T}{|T|} \right)$$

is the entropy function .The function $freq\ C_j, T$,is the function calculating frequency of classifying case T as C_j .While having an option to select information gain, by default, however, C4.5 considers the information gain ratio of splitting T_i (where $i \in 1, s$), which is the ratio of information gain to its split information:

$$Split\ \ T = -\sum_{i=1}^{s} \frac{|T_i|}{|T|} \times \log_2 \left(P \frac{|T_i|}{|T|} \right)$$

where P represents the probability function.

It is easy to see that if a discrete attribute has been selected at an ancestor node, then its gain and gain ratio are zero. Thus, C4.5 does not even compute the information gain of those attributes. If a is a continuous attribute, cases in T with a known attribute value are first ordered using a Quicksort ordering algorithm[53]. Assume that the ordered values are v_i (where $i \in 1, m$).Consider for $i \in 1, m-1$ the value $v = \frac{v_i + v_{i+1}}{2}$ and the splitting:

$$T_1^v = \ v_j / v_j \le v \ \ ; T_2^v = \ v_j / v_j > v$$

For each value v, the information gain $gain_v$ is computed by considering the splitting above, The value v' for which gain is $gain_v$. By default, again, C4.5 calculates the information gain ratio of the splitting sets T_1^{st}, T_2^{nd}. Finally, note that, in case the attribute is selected at the node, the threshold is calculated (Step(5)) by means a linear search in the whole training set ΓS of the attribute value that best approximates the local threshold v' from below(i.e., which is not greater than v'). Such a value is a set to be the threshold at the node.

Since constructed decision trees may be large and unreadable and may suffer from the over fitting problem. The C4.5 system offers a simplified tree obtained by cutting paths according to a given confidence level. Both the decision tree and its simplified version are evaluated by computing the percentage of cases misclassified by the trees. Also, such an evaluation can be performed on a test set, a set of unseen cases during the tree construction. [53].

3.5 Summary

Chapter 3 explains the basic concepts of machine learning and the C4.5 algorithm that the thesis used for the research. Chapter 4 summarizes the experimental results.

CHAPTER 4

PROPOSED ENCODER

4.1 Introduction

Motion estimation accounts for about 70% of the total encoding time in H.264 [36]. This thesis aims at applying machine learning to the motion estimation block to reduce the encoding. The implementation has been done on JM 16.2 [29]. Machine learning is used to exploit the spatial and temporal redundancies in video in order to make optimal mode decisions by replacing the sum of absolute differences (SAD) and other cost evaluations by if – else statements in the motion estimation block. The flow chart shown in Figure 6 sums up the approach incorporated in this thesis. J4.8 analysis is used to reduce the complexity in determining the mode decisions. The statistics for each 16x16 macroblock of the first four frames of the video sequence is calculated. The statistics being the mean, variance, variance of means for all the sub macroblock sizes in the macroblock, mean of the adjacent macroblocks, variance of the adjacent macroblocks and variance of means for all the submacroblock sizes in the adjacent blocks. The modes for the same first four frames from the video sequences are determined from the H.264 encoder in the JM 16.2 software. These modes and the determined statistics are collectively given as attributes for training in the WEKA [40] tool. This is an offline process. The improvement in the encoding varies with the number of frames and the statistical data of the macroblocks in those frames. The improvements in the encoding time for 4 frames and 100 frames have been listed in this thesis report.

4.2 Approach

C4.5 (J48) of the WEKA [40] is used as the classifier algorithm to determine the mode decision tree. This thesis has tried to determine a universal tree that can give relatively accurate mode decisions to any video sequence. To demonstrate this, this thesis has used different combinations of video sequences for training the mode decision trees and later testing the mode decision trees. Table 4.1 summarizes the results. The attributes most commonly considered for mode decision in all the entries in the table are considered to determine the mode decision for the universal mode decision tree. This tree is implemented in the form of if – else statements in the motion estimation block of JM16.2. Hence, the motion estimation process is reduced to if-else statements.

The idea of applying the concept of machine learning to improve the encoding time in H.264 encoder is taken from [18] as shown in figure 4.1. But the approach that this thesis has implemented is different from that in [18]. The idea in [18] is that the attributes of a sequence along with its mode decisions are given to the WEKA tool. The classifier result is obtained. The motion estimation code is then replaced by this tree. A unique tree is first built for every sequence. The same sequence is used for both training and testing. Hence, the efficiency of the tree (with respect to the motion estimation time) is greater than the approach of this thesis.

4.3 Experimental Results

Table 4.1 shows the WEKA tool results for various combinations of video sequences as training and test sequences. The if-else statements derived from the mode decision trees were used to replace the mode decision block in the JM encoder. The assessment metrics like PSNR, MSE, SSIM and file compression ratios for the H.264 video encoder as in JM 16.2 and the encoder based on machine learning are tabulated in Tables 4.2 through 4.13.

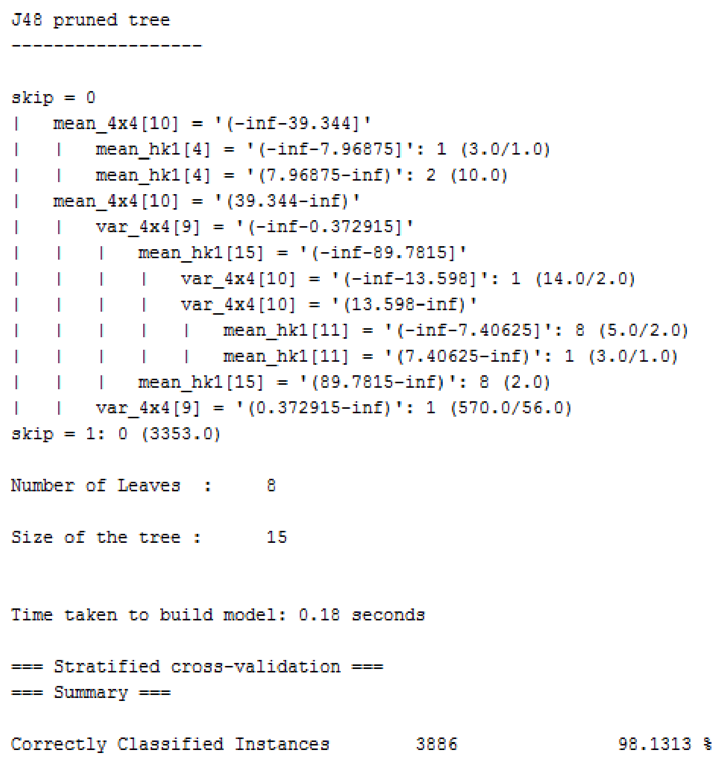
Table 4.1 Weka Results

Training Seq 1	% Accuracy * for Training seq 1	Training seq 2	% Accuracy * for Training seq 2	Test seq	% Accuracy*
Bus_cif	70.6861	Foreman_cif	80.7645	Mobile_cif	77.188
Stefan_cif	81.8182	Tempete_cif	82.8897	Container_cif	85.207
Container_cif	98.9268	------	----	Waterfall_cif	93.358
Waterfall_cif	90.5636	---------	-------	Stefan_cif	85.9583
Bus_cif	70.6861	--------	------	Container_cif	86.529
Bus_cif	75.4665	Foreman_cif	94.9495	Mobile_cif	82.0896
Stefen_cif	88.3838	Tempete_cif	85.0444	Container_cif	90.1812
Container_cif	98.131	----------	-----	Waterfall_cif	95.00442
Waterfall_cif	92.1086	--------	-------------	Stefan_cif	88.952
Bus_cif	70.6861	---------	-------------	Bus_cif	74.8865

Table 4.1 - *Continued*

Waterfall_cif	90.5636	---------	------	Bus_cif	83.0469

%Accuracy * refers to the accuracy in determining the mode decision using machine learning in comparison to the mode decision in JM 16.2 encoder .



```
J48 pruned tree
------------------

skip = 0
|   mean_4x4[10] = '(-inf-39.344]'
|   |   mean_hk1[4] = '(-inf-7.96875]': 1 (3.0/1.0)
|   |   mean_hk1[4] = '(7.96875-inf)': 2 (10.0)
|   mean_4x4[10] = '(39.344-inf)'
|   |   var_4x4[9] = '(-inf-0.372915]'
|   |   |   mean_hk1[15] = '(-inf-89.7815]'
|   |   |   |   var_4x4[10] = '(-inf-13.598]': 1 (14.0/2.0)
|   |   |   |   var_4x4[10] = '(13.598-inf)'
|   |   |   |   |   mean_hk1[11] = '(-inf-7.40625]': 8 (5.0/2.0)
|   |   |   |   |   mean_hk1[11] = '(7.40625-inf)': 1 (3.0/1.0)
|   |   |   mean_hk1[15] = '(89.7815-inf)': 8 (2.0)
|   |   var_4x4[9] = '(0.372915-inf)': 1 (570.0/56.0)
skip = 1: 0 (3353.0)

Number of Leaves  :     8

Size of the tree :      15


Time taken to build model: 0.18 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3886               98.1313 %
```

Figure 4.1 Classification tree for container sequence from Weka tool

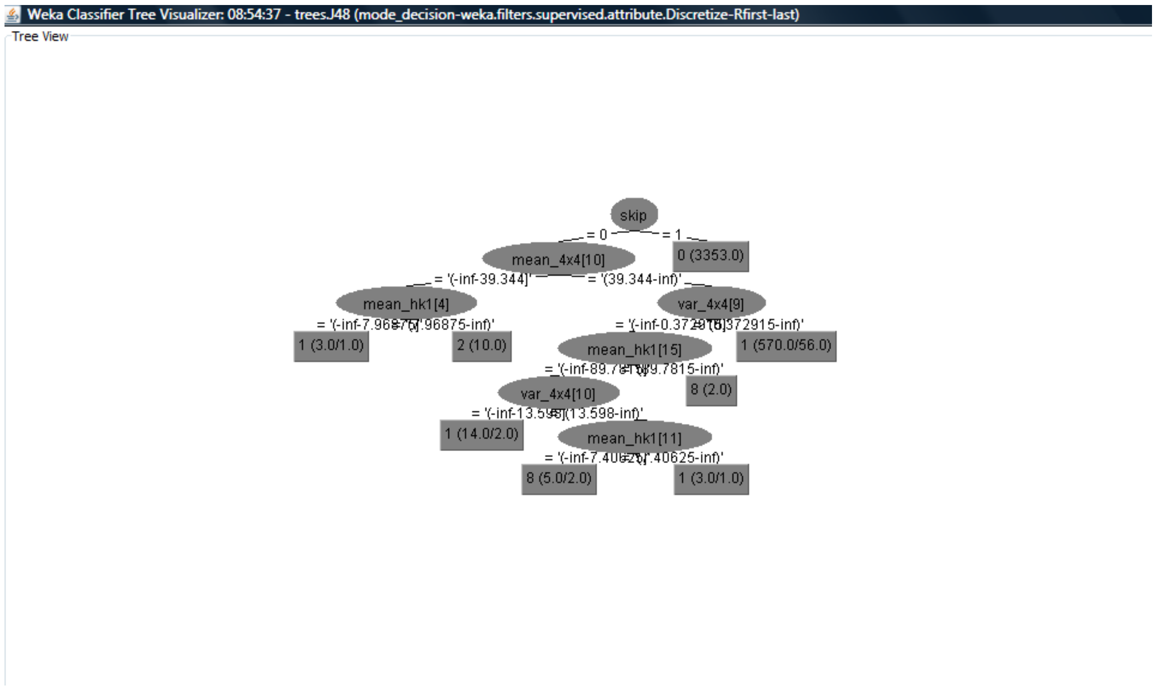
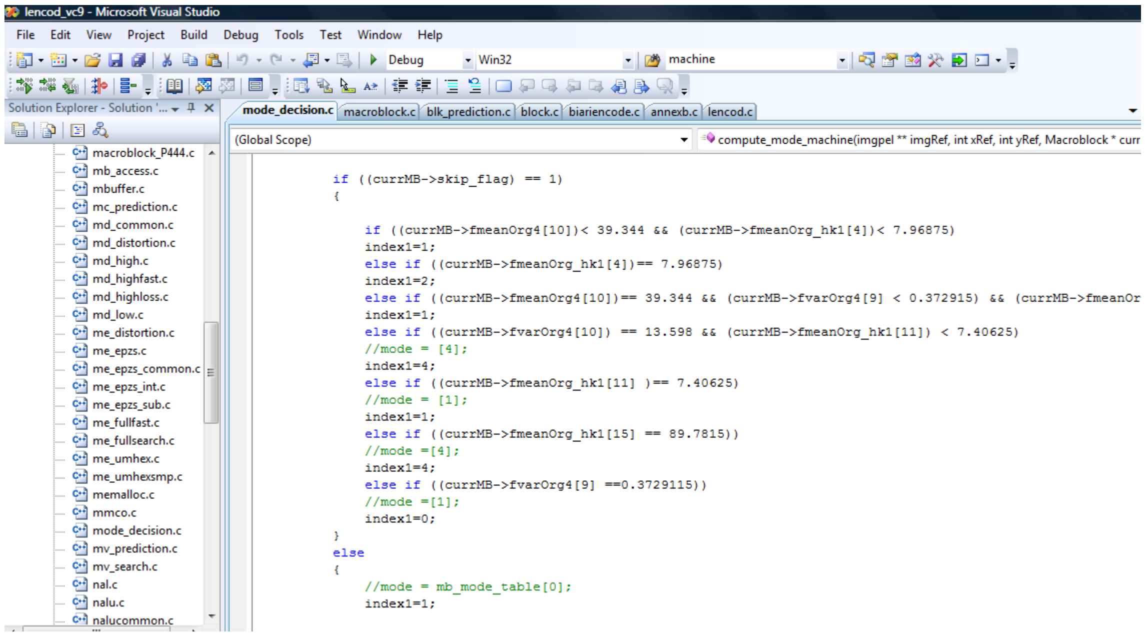


Figure 4.2 Tree structure for container sequence from Weka tool



```
        if ((currMB->skip_flag) == 1)
        {

            if ((currMB->fmeanOrg4[10])< 39.344 && (currMB->fmeanOrg_hk1[4])< 7.96875)
            index1=1;
            else if ((currMB->fmeanOrg_hk1[4])== 7.96875)
            index1=2;
            else if ((currMB->fmeanOrg4[10])== 39.344 && (currMB->fvarOrg4[9] < 0.372915) && (currMB->fmeanOr
            index1=1;
            else if ((currMB->fvarOrg4[10]) == 13.598 && (currMB->fmeanOrg_hk1[11]) < 7.40625)
            //mode = [4];
            index1=4;
            else if ((currMB->fmeanOrg_hk1[11] )== 7.40625)
            //mode = [1];
            index1=1;
            else if ((currMB->fmeanOrg_hk1[15] == 89.7815))
            //mode =[4];
            index1=4;
            else if ((currMB->fvarOrg4[9] ==0.3729115))
            //mode =[1];
            index1=0;
        }
        else
        {
            //mode = mb_mode_table[0];
            index1=1;
```

Figure 4.3 Snap shot of the implementation of the tree structure in JM 16.2 encoder

The simulation results (4 frames)obtained using JM 16.2 and the JM modified using machine learning .The results are as shown in Table 4.2.

Table 4.2 Results obtained using JM 16.2 and JM using machine learning for 4 frames

Seq No	Sequence	Encoding time (seconds) for JM 16.2 without machine learning.	Encoding time (seconds) using machine learning.	ME time (seconds) for JM 16.2 without machine learning.	ME time (seconds) using machine learning.
1	Foreman_ qcif	346.720	270.037	247.147	151.595
2	Coast_qcif	361.714	279.803	242.531	144.371
3	Car phone_qcif	347.85	269.674	249.081	152.576
4	Silent_qcif	368.155	253.006	254.297	139.053
5	Suzie_qcif	343.983	342.583	263.777	260.981
6	Miss-america_q cif	368.694	198.909	310.542	141.584
7	Bus_cif	1608.934	1346.542	1010.012	617.088
8	Container_	1542.106	1241.772	1109.672	686.165

Table 4.2 – *Continued*

9	Foreman_ cif	1689.383	889.833	1316.543	537.128
10	Mobile_cif	2031.07	1695.243	1066.867	627.440
11	Tempete_ cif	1808.560	1361.954	1078.435	590.689
12	Stefan_cif	1750.255	1267.813	1136.800	617.822
13	Waterfall_ cif	1497.525	994.996	1017.974	529.557
14	Mother- daughter_ qcif	422.332	360.371	322.011	276.212

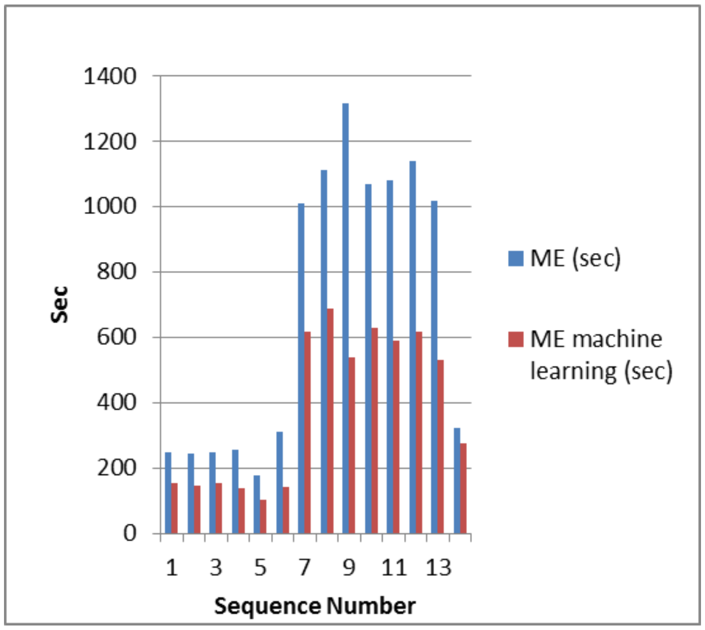


Figure 4.4 Motion estimation time for 4 frames for sequences in Table 4.2

The speed up in the encoding time and motion estimation time (4 frames)by using machine learning .The results are tabulated in Table 4.3.

Table 4.3 Speed up in encoding time and motion estimation time for 4 frames using machine learning compared to JM 16.2 encoder

Sequence Number	Sequence	Speed up in Encoding time	Speed up in ME time
1	Foreman_qcif	22.11 %	38.66 %
2	Coast_qcif	22.64 %	40.47 %
3	Car phone_qcif	22.47 %	38.74 %
4	Miss-america_qcif	15.772%	28.86%

Table 4.3 - *Continued*

5	Bus_cif	16.30 %	38.90 %
6	Container_cif	19.47 %	38.16 %
7	Foreman_cif	47.32 %	59.20 %
8	Mobile_cif	47.47%	62.99%
9	Tempete_cif	40.370%	56.629%
10	Stefan_cif	35.04%	51.268%
11	Waterfall_cif	32.022%	46.778%
12	Silent_qcif	30.9266%	45.039%
13	Suzie_qcif	23.36779%	23.819%
14	Mother-daughter_qcif	23.75%	23.353%

The compressed file sizes (4 frames) obtained using normal JM 16.2 and JM using machine learning. It is important that the H.264/AVC encoder with machine learning maintains the compression ratio. Hence it is important to compare the compressed file sizes for the encoder with machine learning and the encoder in JM 16.2 .The results are tabulated in Table 4.4.

Table 4.4 Comparison of compressed file sizes for four frames for sequences in Table 4.2

Sequence Number	Sequence	Compressed file size (KB) in JM 16.2 encoder.	Compressed file size (KB) using machine learning.	% Increase in encoded file size using machine learning
1	Foreman_qcif	4.34	4.34	0
2	Coast_qcif	5.68	5.67	+ 0.0017
3	Silent_qcif	4.0	4.0	0
4	Suzie_qcif	3.0	3.0	0
5	Car phone_qcif	4.52	4.54	0.0044
6	Bus (cif)	31.9	32.2	0.0093
7	Container (cif)	12.0	12.0	0
8	Foreman (cif)	12.4	12.7	0.1903
9	Mobile(cif)	50.4	51.0	0.0119

Table 4.4 - *Continued*

10	Stefan(cif)	32.5	34.0	0.0462
11	Waterfall(cif)	18.7	19.0	0.0160
12	Tempete(cif)	36.7	37.0	0.0082
13	Miss-america_qcif	2.0	2.0	0
14	Mother-daughter_qcif	2.279	2.279	0.0

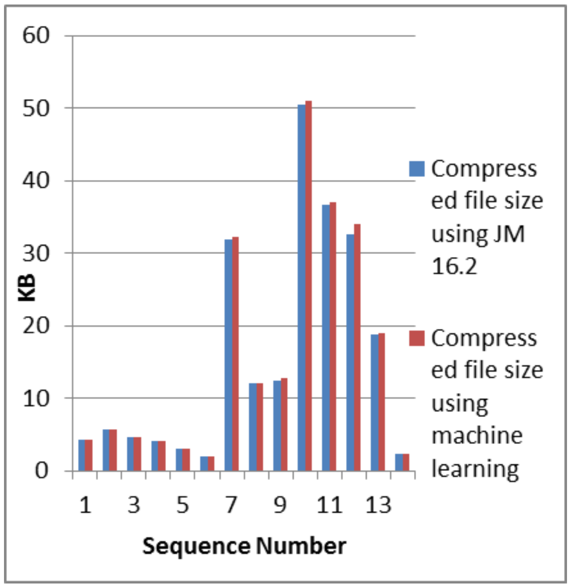


Figure 4.5 Compressed file sizes using machine learning for four frames for sequences in Table 4.4

It is important that the H.264/AVC encoder with machine learning maintains the fidelity of the video sequences. Hence it is important to compare the fidelity measures like PSNR and MSE for the encoder with machine learning and the encoder in JM 16.2. The results are tabulated in Table 4.5.

Table 4.5 Comparison of PSNR and MSE for four frames

Sequence Number	Sequence	PSNR(dB) using JM 16.2 encoder	PSNR (dB) using machine learning	MSE using JM 16.2 encoder.	MSE using machine learning.
1	Foreman_qcif	37.389	37.324	11.881	12.068
2	Coast_qcif	35.24	35.21	19.539	19.681
3	Car ph_qcif	37.937	37.879	10.472	10.619
4	Miss-america_qcif	40.949	40.881	5.22970	5.31475
5	Bus_cif	35.961	35.932	16.518	16.633
6	Container_cif	37.162	37.153	12.517	12.544

Table 4.5 – *Continued*

7	Foreman_cif	37.833	37.85	10.371	10.684
8	Mobile_cif	35.541	35.512	18.2873	18.419
9	Tempete_cif	35.962	35.93	16.594	16.705
10	Stefan_cif	37.011	36.985	13.00572	13.08644
11	Waterfall_cif	35.912	35.906	16.6923	16.716
12	Mother-daughter_qcif	38.363	38.363	9.481	9.481
13	Silent_qcif	36.784	36.775	13.63795	13.6775
14	Suzie_qcif	37.749	37.741	10.938	10.381

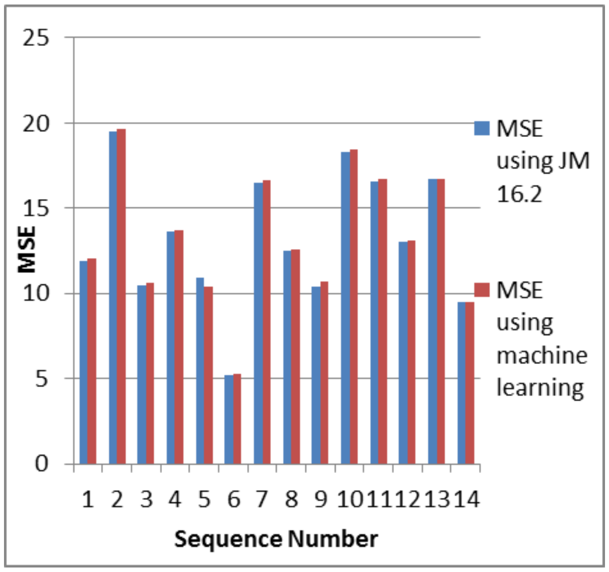


Figure 4.6 MSE in JM 16.2 and machine learning for four frames for sequences in Table 4.5

The simulation results obtained (4 frames) with respect to SSIM .

It is important that the H.264/AVC encoder with machine learning maintains the fidelity of the video sequences. Hence it is important to compare the perceptual quality measure SSIM for the encoder with machine learning and the encoder in JM 16.2 the results are tabulated in Table 4.6

Table 4.6 SSIM comparison for four frames

Sequence Number	Sequence	SSIM for JM 16.2	SSIM using machine learning.	% decrease **
1	Foreman_qcif	0.95944	0.95910	0.035
2	Coast_qcif	0.91793	0.91763	0.032
3	Car phone_qcif	0.96670	0.96641	0.029
4	Suzie_qcif	0.9555	0.9557	0.0002
5	Bus_cif	0.94973	0.94941	0.033
6	Container_cif	0.92827	0.92823	0.0043
7	Foreman_cif	0.94302	0.94306	+ 0.0042
8	Mobile_cif	0.9758	0.9755	.00003
9	Tempete_cif	0.9711	0.9709	0.02
10	Stefan_cif	0.9807	0.9806	0.0001
11	Waterfall_cif	0.9420	0.9420	0.00
12	Silent_qcif	0.9600	0.9600	0.00

Table 4.6 - *Continued*

13	Miss-america_qcif	0.9707	0.9706	0.001
14	Mother-daughter_qcif	0.9663	.9663	0.00

%decrease** refers to the percentage decrease in SSIM using machine learning in comparison to the SSIM as obtained in JM 16.2 encoder

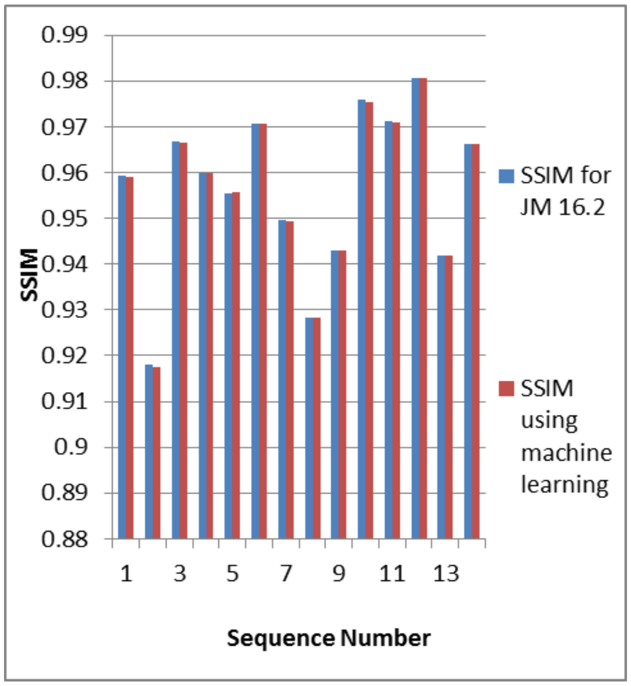


Figure 4.7 SSIM comparison for four frames for sequences in Table 4.6

The simulation results (100 frames) obtained using JM 16.2 and the JM modified using machine learning .The results are as shown in Table 4.7.

Table 4.7 Encoding time and ME time for 100 frames using JM 16.2 and JM using machine learning

Seq No	Sequence	Encoding time (seconds) for JM 16.2 without machine learning.	Encoding time (seconds) using machine learning.	ME time (seconds) for JM 16.2 without machine learning.	ME time (seconds) using machine learning.
1	Foreman_qcif	12061.84	12135.46	12309.39	11334.7
2	Coast_qcif	12466.84	11593.49	12050.3	11094.56
3	Car phone_qcif	11988.99	11121.75	12385.94	11908.72
4	Suzie_qcif	11088.88	10539.32	11050.17	10502.91
5	Miss-america_qcif	13062.02	11595.37	13020.76	11559.16
6	Bus_cif	51582.83	44868.16	50387.4	44700.1
7	Container_cif	49234.23	40012.79	48001.23	38785.19
8	Foreman_cif	58235.76	50674.88	56342.9	49463.13

Table 4.7 – *Continued*

9	Mobile_cif	100018.6	82372.02	9981.046	8156.502
10	Tempete_cif	37896.78	36143.12	35963.65	34172.36
11	Stefan_cif	64173.79	61685.47	63243.57	58976.46
12	Waterfall_cif	56789.43	53845.07	49876.13	47876.32
13	Mother-daughter_qcif	10137.79	9248.341	9103.293	8078.816

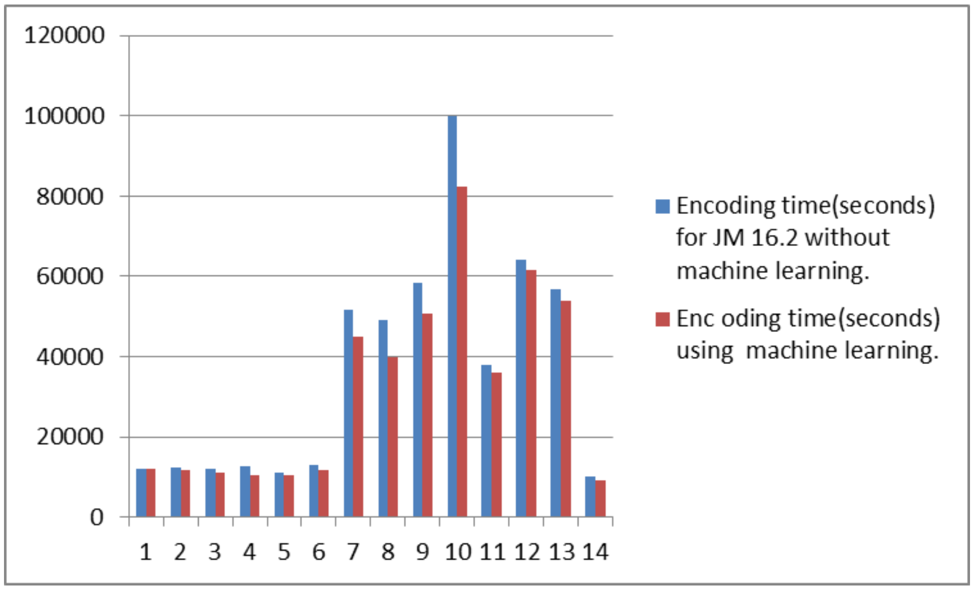


Figure 4.8 Estimation time for 100 frames for sequences in Table 4.8

The speed up in the encoding time and motion estimation time (100 frames) by using machine learning. The results are tabulated in Table 4.8.

Table 4.8 Speed up in encoding time and motion estimation time for 100 frames using machine learning compared to JM 16.2 encoder

Sequence Number	Sequence	Speed up in Encoding time	Speed up in ME time
1	Foreman_qcif	12.11%	28.66%
2	Coast_qcif	12.64%	30.47%
3	Car phone_qcif	12.47%	28.74%
4	Miss-america_qcif	21.27%	35.31%
5	Bus_cif	12.43%	31.25%
6	Container_cif	36.05%	44.40%
7	Foreman_cif	15.30%	28.90%
8	Mobile_cif	19.47%	28.16%
9	Tempete_cif	37.32%	49.20%
10	Stefan_cif	16.53%	31.18%
11	Waterfall_cif	14.69%	35.22%
12	Silent_qcif	17.56%	35.65%
13	Suzie_qcif	23.55%	37.97%

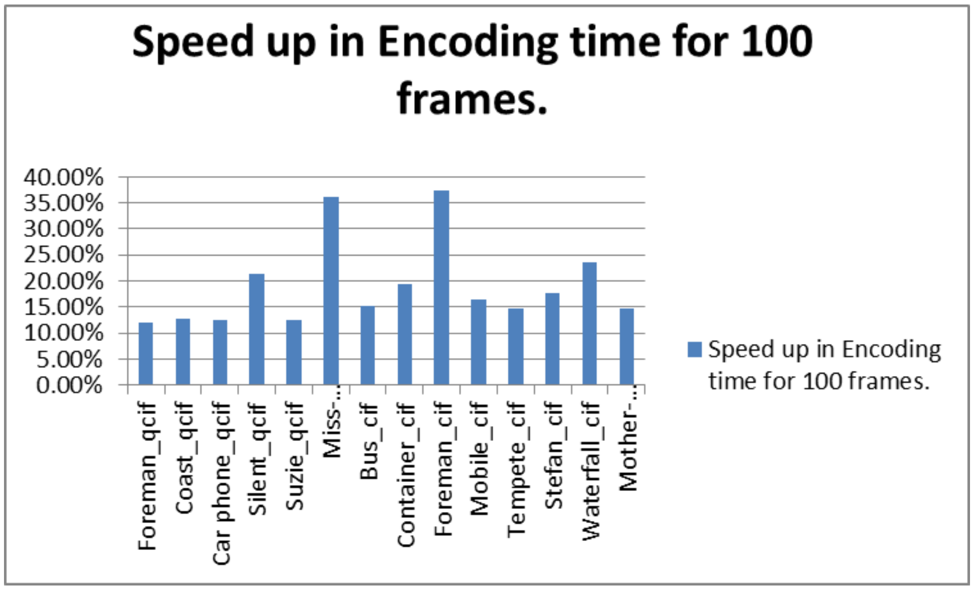


Figure 4.9 Speed up in total encoding time for 100 frames for sequences in Table 4.8

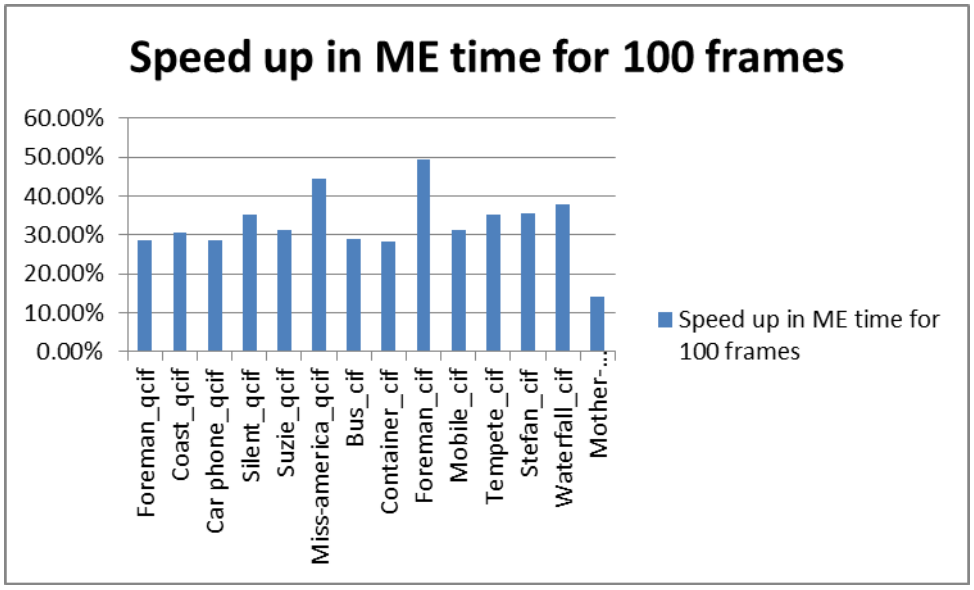


Figure 4.10 Speed up in Motion estimation time for 100 frames for sequences in Table 4.8

The compressed file sizes (100 frames) obtained using normal JM 16.2 and JM using machine learning. The results are tabulated in Table 4.9.

Table 4.9 Comparison of compressed file sizes for 100 frames

Sequence Number	Sequence	Compressed file size (KB) in JM 16.2 encoder.	Compressed file size (KB) using machine learning.	% Increase in encoded file size using machine learning
1	Foreman_qcif	4.34	4.34	0
2	Coast_qcif	5.68	5.67	+ 0.0017
3	Silent_qcif	4.0	4.0	0
4	Suzie_qcif	3.0	3.0	0
5	Car phone_qcif	4.52	4.54	0.0044
6	Bus (cif)	31.9	32.2	0.0093
7	Container (cif)	12.0	12.0	0
8	Foreman (cif)	12.4	12.7	0.1903
9	Mobile(cif)	50.4	51.0	0.0119
10	Stefan(cif)	32.5	34.0	0.0462
11	Waterfall(cif)	18.7	19.0	0.0160
12	Tempete(cif)	36.7	37.0	0.0082
13	Miss-america_qcif	2.0	2.0	0

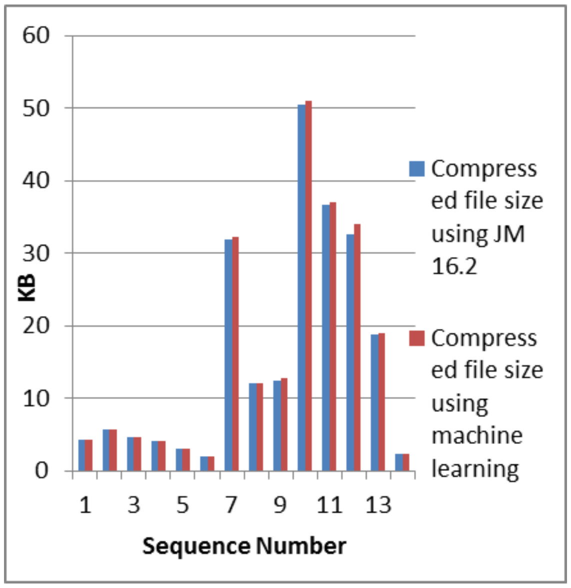


Figure 4.11 Compressed file sizes for 100 frames using machine learning for sequences in Table 4.4

The simulation results obtained (100 frames) with respect to PSNR and MSE are tabulated in Table 4.10.It is important that the H.264/AVC encoder with machine learning maintains the fidelity of the video sequences. Hence it is important to compare the fidelity measures like PSNR and MSE for the encoder with machine learning and the encoder in JM 16.2. Table 4.10 tabulates the comparisons.

Table 4.10 Comparison of PSNR and MSE for 100 frames

Sequence Number	Sequence	PSNR(dB) using JM 16.2 encoder	PSNR (dB) using machine learning	MSE using JM 16.2 encoder.	MSE using machine learning.
1	Foreman_qcif	37.389	37.324	11.881	12.068

Table 4.10 – *Continued*

2	Coast_qcif	35.24	35.21	19.539	19.681
3	Car ph_qcif	37.937	37.879	10.472	10.619
4	Miss-america_qcif	40.949	40.881	5.22970	5.31475
5	Bus_cif	35.961	35.932	16.518	16.633
6	Container_cif	37.162	37.153	12.517	12.544
7	Foreman_cif	37.833	37.85	10.371	10.684
8	Mobile_cif	35.541	35.512	18.2873	18.419
9	Tempete_cif	35.962	35.93	16.594	16.705
10	Stefan_cif	37.011	36.985	13.00572	13.08644
11	Waterfall_cif	35.912	35.906	16.6923	16.716
12	Mother-daughter_qcif	38.363	38.363	9.481	9.481
13	Silent_qcif	36.784	36.775	13.63795	13.6775
14	Suzie_qcif	37.749	37.741	10.938	10.381

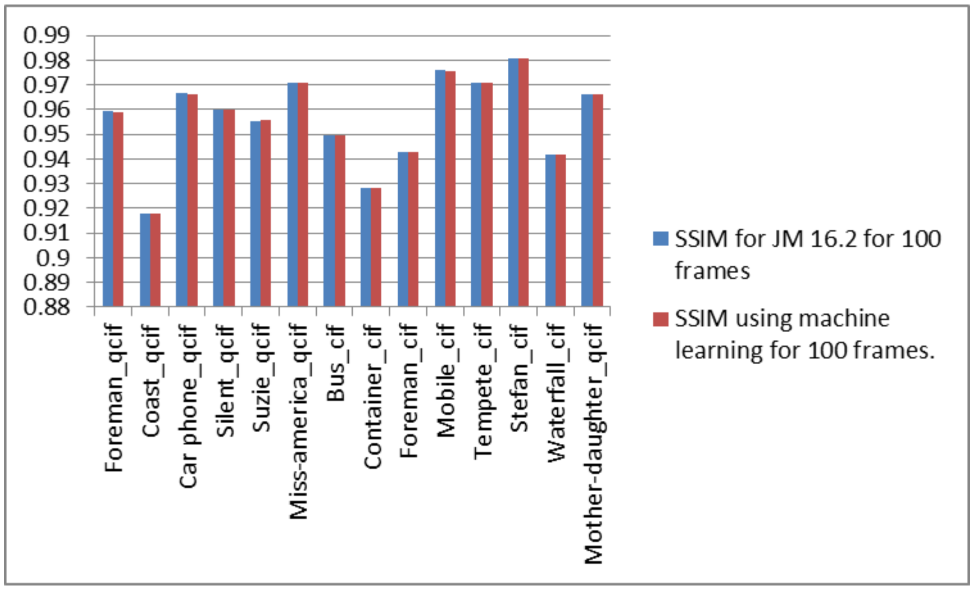


Figure 4.12 SSIM in JM 16.2 and machine learning for 100 frames for sequences in Table 4.10

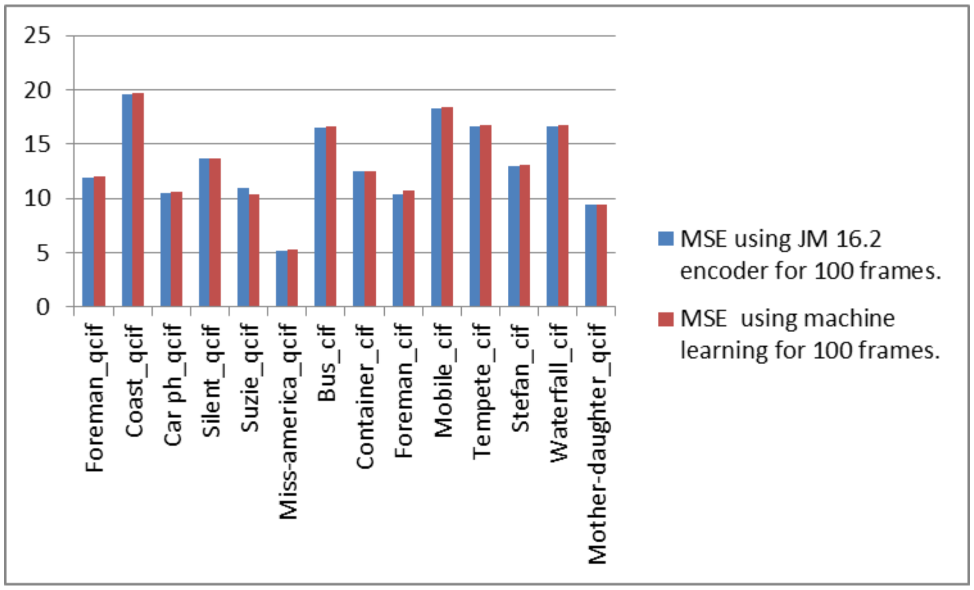


Figure 4.13 MSE in JM 16.2 and machine learning for 100 frames for sequences in Table 4.10

The simulation results obtained (100 frames) with respect to SSIM . It is important that the H.264/AVC encoder with machine learning maintains the fidelity of the video sequences.

Hence it is important to compare the perceptual video quality SSIM for the encoder with machine learning and the encoder in JM 16.2The results are tabulated in Table 4.11.

Table 4.11 SSIM comparison for 100 frames

Sequence Number	Sequence	SSIM for JM 16.2	SSIM using machine learning.	% decrease **
1	Foreman_qcif	0.95944	0.95910	0.035
2	Coast_qcif	0.91793	0.91763	0.032
3	Car phone_qcif	0.96670	0.96641	0.029
4	Suzie_qcif	0.9555	0.9557	0.0002
5	Bus_cif	0.94973	0.94941	0.033
6	Container_cif	0.92827	0.92823	0.0043
7	Foreman_cif	0.94302	0.94306	+ 0.0042
8	Mobile_cif	0.9758	0.9755	.00003
9	Tempete_cif	0.9711	0.9709	0.02
10	Stefan_cif	0.9807	0.9806	0.0001
11	Waterfall_cif	0.9420	0.9420	0.00
12	Silent_qcif	0.9600	0.9600	0.00
13	Miss-america_qcif	0.9707	0.9706	0.001

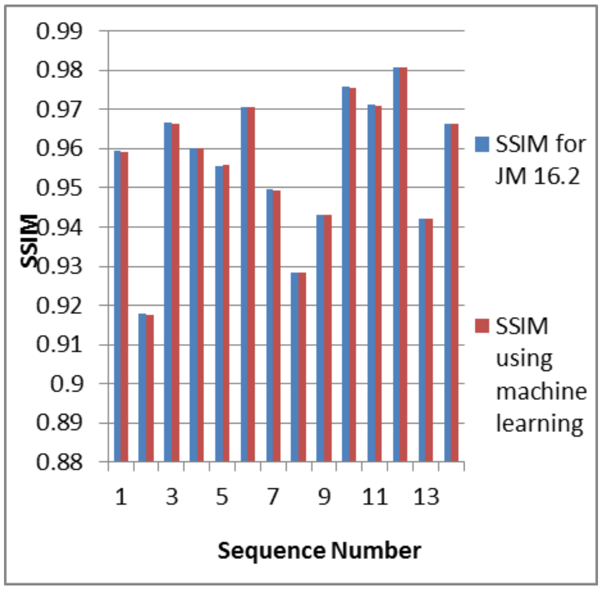


Figure 4.14 SSIM comparison for 100 frames for sequences in Table 4.11

4.4 Observations

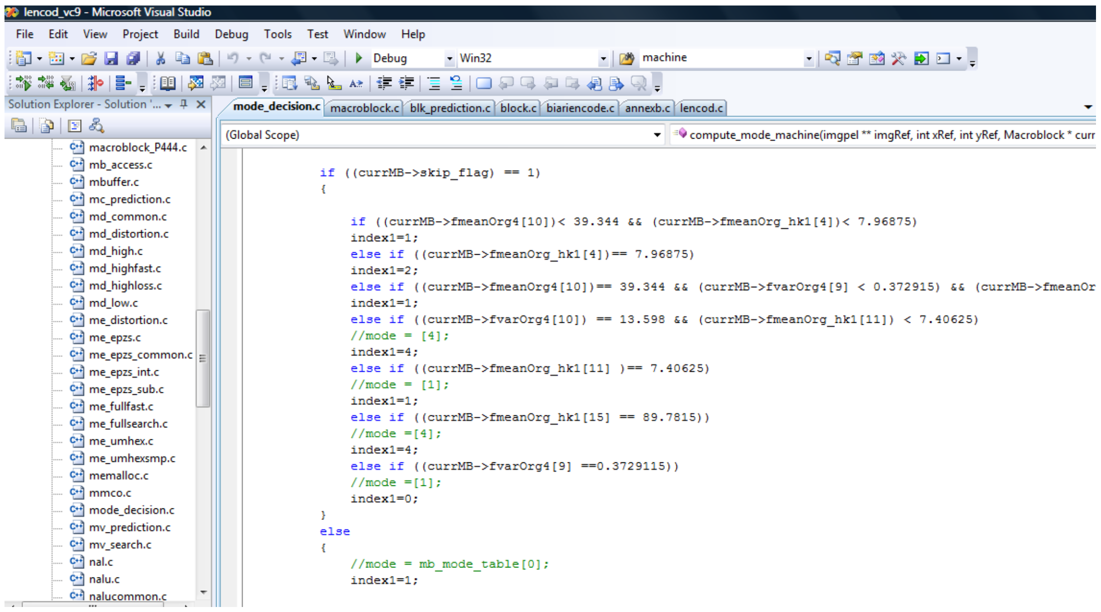
Tables 4.2 through Tables 4.6 tabulate the simulation results for encoding 4 frames of the video sequences and Tables 4.7 through Tables 4.11 tabulate the simulation results for encoding 100 frames of the video sequences. The observation made is that the average improvement in encoding time for the first 4 frames is 40.28% and the average improvement in encoding time for the first 100 frames is 18.3%. This variation can be attributed to the fact that the training for the classification rule has been done using the first four frames. Training on 100 frames is not done because the amount of data in 100 frames would lead to the over-fitting problem and hence a general classification rule cannot be built. Hence the performance of the classification rule is optimum for the first four frames only.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

It was observed that a single universal mode decision tree failed to produce good fidelity of the video when all the modes for ME/MC were used in the machine learning algorithm. So this thesis considered only the sub macro block modes, i.e. 8x8, 8x4, 4x8 and 4x4 modes (Figure 2.6) for the machine learning. The function called 'submacroblock_mode_decision' in the JM 16.2 software [29] has been replaced by the if-else statements as shown in Figure 5.1. The results for four frames are tabulated in the Tables 4.3 through 4.7. From Table 4.3, it is clear that the average speed up in the encoding time is 28.5%. The average speed up in the motion estimation time is 42.84605%. From table 4.5, the average percentage decrease in compressed file size is 0.36%. From Table 4.7, it is evident that the average decrease in SSIM is less than 0.0107%. The results for hundred frames are tabulated in the Tables 4.8 through 4.11. From Table 4.8, it is clear that the average speed up in the encoding time is 8.5%. The average speed up in the motion estimation time is 18.34600%. From Table 4.10, it is evident that the average decrease in SSIM is less than 0.0109%. This can be attributed to the over-fitting problem of the classifier. Since the training sequence includes only the first four frames, the efficiency of the classifier is less when the testing sequence includes one hundred frames.



```
if ((currMB->skip_flag) == 1)
{

    if ((currMB->fmeanOrg4[10])< 39.344 && (currMB->fmeanOrg_hk1[4])< 7.96875)
    index1=1;
    else if ((currMB->fmeanOrg_hk1[4])== 7.96875)
    index1=2;
    else if ((currMB->fmeanOrg4[10])== 39.344 && (currMB->fvarOrg4[9] < 0.372915) && (currMB->fmeanOr
    index1=1;
    else if ((currMB->fvarOrg4[10]) == 13.598 && (currMB->fmeanOrg_hk1[11]) < 7.40625)
    //mode = [4];
    index1=4;
    else if ((currMB->fmeanOrg_hk1[11] )== 7.40625)
    //mode = [1];
    index1=1;
    else if ((currMB->fmeanOrg_hk1[15] == 89.7815))
    //mode =[4];
    index1=4;
    else if ((currMB->fvarOrg4[9] ==0.3729115))
    //mode =[1];
    index1=0;
}
else
{
    //mode = mb_mode_table[0];
    index1=1;
```

Figure 5.1 Snap shot of the implementation of the Tree structure in JM 16.2 encoder

5.2 Future Work

As explained in section 5.1, the sub-macroblock 8x8, 8x4, 4x8 and 4x4 modes have been used for classification in this thesis. A classification rule which considers all the modes for mode decision can be developed. This thesis used data from four frames for training the classification rule. A scheme which would consider more number of frames for training while considering the over-fitting problem can be pursued for future work.

APPENDIX A

STRUCTURAL SIMILARITY INDEX METRIC (SSIM)

SSIM is a new objective method that can measure image quality between a distorted image and a reference image. The SSIM measures the degradation of structural information based on the assumption that human visual system characteristics are adopted for extracting structural information from an image scene. SSIM has better consistency with perceived image quality than pixel error model based on different performance evaluations. The system diagram of structural similarity measurement system is shown in Figure A.1.

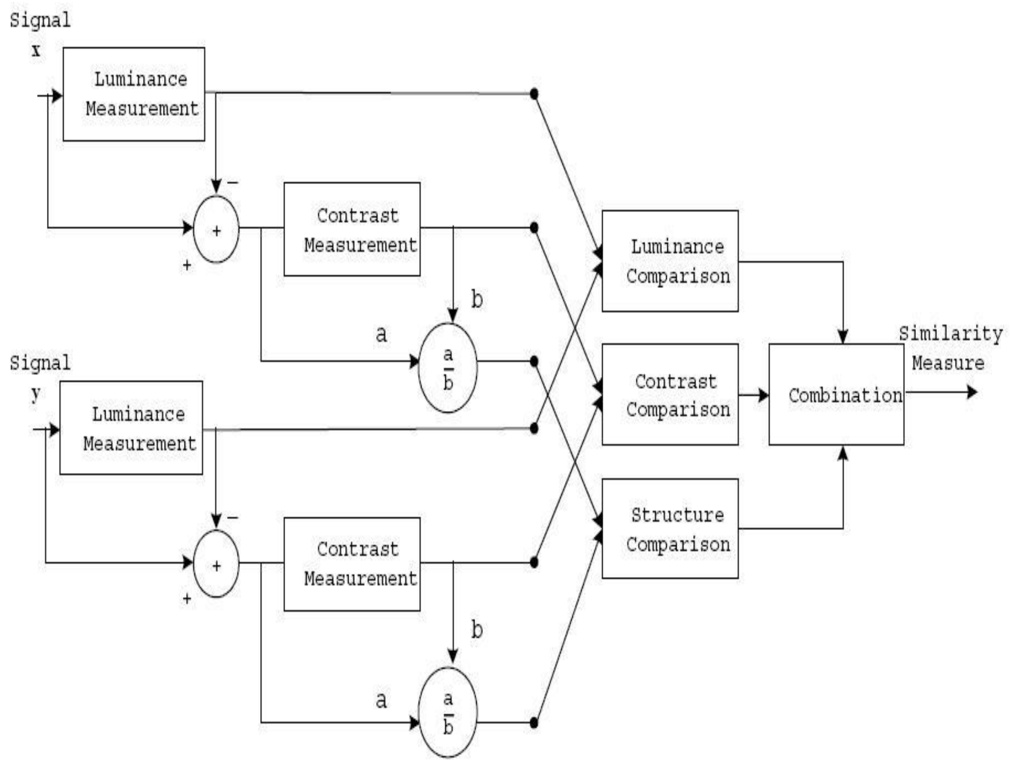


Figure A.1 Structural similarity index measurement system [56]

Suppose **x** = { x_i |i = 1, 2, · , N} and **y** = { y_i |i = 1, 2, · , N} are two finite-length image signals, which have been aligned with each other, SSIM is defined as the product of three local quantities: luminance comparison (function of mean), contrast comparison (function of variance), and structure comparison (function of correlation coefficient and variance).

$$SSIM(x,y) = [l(x,y)]^{\alpha}\ [c(x,y)]^{\beta}[s(x,y)]^{\gamma}$$

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y{}^2 + C_2}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where μ_x and μ_y are the means of x and y respectively.

σ_x and σ_y are the standard deviations of the images X and Y respectively.

σ_{xy} is the covariance of X and Y .

C_1 , C_2 and C_3 are small constants such that $C_1 = L^2K_1^2$

$C_2 = (K_2L^2)$ and $C_3 = \frac{C_2}{2}$

where K_1,K_2 and L are constants

SSIM can have a maximum value of 1.

The MSE and its derivative PSNR are conventional metrics to compare any two images. MSE measures the difference between the original and distorted pixels. PSNR is a logarithmic representation of the inverse of this measure. Compared to other objective measures, PSNR is easy to compute and well understood by most researchers. However both MSE and PSNR do not correlate well with the subjective quality of the reconstructed images. The subtle differences between degradations of different intensities are not properly reflected using PSNR. The SSIM metric has proven to be a metric that is closest to the human perception of the received video sequence. This method utilizes structural distortion as an estimate of perceived visual distortion, whereas most other proposed approaches are error sensitivity based methods [25].

APPENDIX B

VIDEO SEQUENCES CONSIDERED IN THIS THESIS

foreman_qcif

coast_qcif

carphone_qcif

silent_qcif

Miss-america_qcif

Mother-daughter_qcif

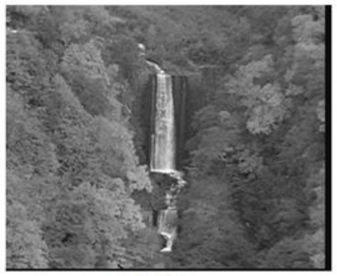
Bus_cif

Conatiner_cif

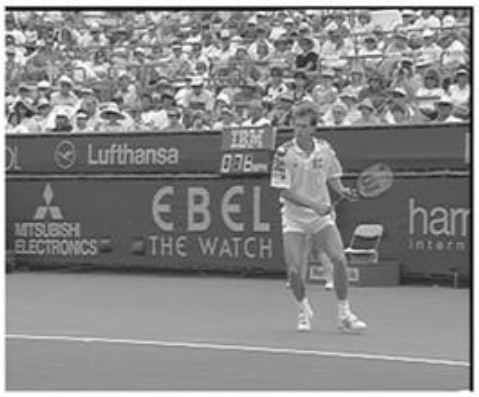
Foreman_cif

mobile_cif

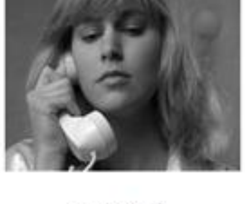
tempete_cif



waterfall_cif



stefan_cif



suzie_qcif

Figure B.1 Test Sequences mentioned in table 4.1

REFERENCES

[1] I. E.G. Richardson, “H.264 and MPEG-4 video compression: video coding for next-generation multimedia”, Wiley, 2003.

[2] K. Sayood, "Introduction to Data Compression,” 3rd Edition, Morgan Kaufmann Publisher Inc., 2006.

[3] R. Schafer and T. Sikora, "Digital video coding standards and their role in video communications," Proceedings of the IEEE, Vol. 83, pp. 907-923, Jan. 1995.

[4] Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 / ISO / IEC 14496-10, Nov. 2009.

[5] Open source article, “H.264/MPEG-4 AVC,” Wikipedia Foundation, http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[6] G.A Davidson et al, “ATSC video and audio coding”, Proceedings of the IEEE, vol. 94, pp. 60-76, Jan. 2006 (www.atsc.org).

[7] G.F.Escribano et al, “ An MPEG-2 to H.264 video transcoder in the baseline profile”, IEEE Trans. CSVT, vol. 20, pp. 763-768, May 2010.

[8] S. Kwon, A. Tamhankar and K.R. Rao, ”Overview of H.264 / MPEG-4 Part 10”, J. Visual Communication and Image Representation, vol. 17, pp.183-216, April 2006.

[9] T. Wiegand and G. J. Sullivan, “The H.264 video coding standard”, IEEE Signal Processing Magazine, vol. 24, pp. 148-153, March 2007.

[10] A. Puri et al, “Video coding using the H.264/ MPEG-4 AVC compression standard”, Signal Processing: Image Communication, vol. 19, pp: 793 – 849, Oct. 2004.

[11] D. Marpe, T. Wiegand and S. Gordon, “H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas”, Proceedings of the IEEE International Conference on Image Processing 2005, vol. 1, pp. 593 - 596, Sept. 2005.

[12] G. Sullivan, P. Topiwala and A. Luthra, “The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions”, SPIE conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, Aug. 2004.

[13] K. R. Rao and P. C. Yip, “The transform and data compression handbook”, Boca Raton,FL: CRC press, 2001.

[14] T. Wiegand et al**, “**Introduction to the Special Issue on Scalable Video Coding—Standardization and Beyond” IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, pp. 1034, Sept. 2007.

[15] HHI presentation of the Scalable Extension of H.264/AVC, http://ip.hhi.de/imagecom_G1/savce/index.htm

[16] Y. Huh, K. Panusopone, K.R. Rao, “Variable block size coding of images with hybrid quantization”, IEEE Trans. Circuits And Systems for Video Technology vol. 6, pp 679–685, Dec. 1996.

[17] J. Ribas-Corbera and D.L. Neuhoff, “Optimizing Block Size in Motion Compensation”, Journal of Electronic Imaging, vol. 7, pp.155-165, Jan. 1998

[18] T.D.Tran, J.Liang and C. Tu, “Lapped Transform via Time-Domain Pre- and Post-Filtering”, IEEE Trans on Signal Processing, vol.51, no.6, pp. 1557-1571, Jun. 2003.

[19] W.B.Pennebaker and J.L.Mitchell, JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, 1993.

[20] M. Wien, “Variable block size transforms for H.264/AVC”, IEEE Trans. For Circuits and Systems for Video Technology, vol. 13, pp. 604–613, July 2003.

[21] Nokia, Proposal to support MPEG-4 AVC / H.264/ AVC in Rel-6, 3GPP SA4 Meeting #27, document S4-030478. (available from http://www.3gpp.org/ftp/tsg sa/WG4 CODEC/TSGS4 27/Docs/)

[22] M. Ravassi, M. Mattavelli and C. Clerc, “JVT/H.26L decoder complexity analysis”, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, doc. JVT-D153, Klagenfurt, Austria, 22–26 July, 2002 (available via anonymous ftp from ftp://ftp.imtc-files.org/jvt-experts/).

[23] M.-T. Sun, T.-D. Wu and J.-N. Hwang, “Dynamic bit allocation in video combining for multipoint conferencing,” IEEE Trans. Circuits and Syst. II, vol. 45, no. 5, pp. 644-648, May 1998.

[24] O. Werner, “Re-quantization for transcoding of MPEG-2 intra frames,” IEEE Trans. Image Processing, vol. 8, no. 2, pp. 179-191, Feb. 1999.

[25] T. Shanabelah and M. Ghanbari, “Heterogeneous video transcoding to low spatial temporal resolutions and different encoding formats,” IEEE Trans. Multimedia, vol. 2, no. 2, pp. 101-110, Jun. 2000.

[26] K.-H. Tan and M. Ghanbari, “Layered image coding using the DCT pyramid,“ IEEE Trans. Image Processing, vol. 4, no. 4, pp. 512-516, Apr. 1995.

[27] J.-N.Hwang and T.-D. Wu, “Motion vector re-estimation and dynamic frame-skipping for video transcoding,” Conf Rec. 32nd Asilomar Conf. Signals, Systems and Computers vol. 2, pp .1606-1610, 1998.

[28] I. E. Richardson, “The H.264 Advanced Video Compression Standard”, Second Edition, Wiley, 2010.

[29] JM reference software http://iphome.hhi.de/suehring/tml/

[30] A. Luthra, G. Sullivan and T. Wiegand, “Introduction to the special issue on the H.264/AVC video coding standard”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, issue 7, pp. 557-559, July 2003.

[31] Ten things to consider when considering video for distribution for multiple platforms http://www.internetvideomag.com/articles_2010/010610_EncodingVideo.html.

[32] A. Nakagawa, “Fujitsu’s Approach to H.264/AVC and Application Trends”, Sci.Tech. Journal, vol.44, pp.343-350, 2008.

[33] G .Holmes, A. Donkin and I .H. Witten, “Weka: a machine learning workbench, ”IEEE conference on IIS , pp 357-361, 1994.

[34] W.Hua, M.Cuiqin and Z.Lijuan, “A Brief Review of Machine Learning and Its Application”, IEEE conference on ICIECS , pp.1-4, 2009.

[35] Video test sequences (YUV 4:2:0): http://trace.eas.asu.edu/yuv.

[36] M. Chelemal and K.R.Rao, “Fast Computuation for the Discrete Cosine Transform”, Nineteenth Asilomar Conf. on circuits, systems and computers, pp. 273-277,Nov 1985.

[37] http://www.apple.com/quicktime/technologies for H.264 codec reference.

[38] Z. Wang and A. C. Bovik, Modern Image Quality Assessment. Synthesis Lectures on Image, Video and Multimedia Processing. Morgan and Claypool, 2006.

[39] J.Lee,et.al, “SIMD optimization of the H.264/SVC decoder with efficient data structure”, IEEE International Conf. on Multimedia and expo, pp. 69-72, June 2003.

[40] http://www.cs.waikato.ac.nz/ml/weka/ for WEKA tool download.

[41] Z. He and S. K. Mitra, “Optimum bit allocation and accurate rate control for video coding via p-domain source modeling”, IEEE Trans. Circuits Syst. Video Tech, vol.12, pp. 840-849, Oct. 2002.

[42] H.Kim and Y.Altunhasak, “Low-Complexity macroblock mode selection for H.264-AVC encoders”, International Conference on Image Processing, vol.2, pp .765-768, Oct 2004.

[43] P.Carrillo, H. Kalva and T.Pin, “Low complexity H.264 video encoding”, Applications of Digital Image Processing Proc of SPIE, vol. 7443, 74430A, Sept 2009.

[44] M.Enriquez. et.al, “A Fast Motion-Cost Based Algorithm for H.264/AVC Inter Mode Decision”, ICIP 2007, pp 325-328, 2007.

[45] J.M.Moon and J.H. Kim, “A new Low-Complexity Integer Distortion Estimation Method for H.264/AVC Encoder”, IEEE Trans. on Circuits and Systems for video Technology, 2010, vol.20, pp 207-212, Feb.2010.

[46] Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 / ISO / IEC 14496-10, Nov. 2009.

[47] M.Wien, H.Schwarz and T.Oelabum, “Performance Analysis of SVC”, IEEE Trans on Circuits and systems for video Technology, vol.17, pp.1194-1203, Sept.2007.

[48] T. Schierl, T. Stockhammer and T. Wiegand, "Mobile Video Transmission using Scalable Video Coding (SVC)," IEEE Trans. On Circuits and Systems for Video Technology, vol.17,pp.1204-1217,Sept .2007.

[49] E.Martinian ,”Extensions of H.264/AVC for Multimedia Video Compression”, IEEE Proc. of ICIP “,pp 2981-2984,2006.

[50] HHI presentation of the Scalable Extension of H.264/AVC, http://ip.hhi.de/imagecom_G1/savce/index.htm

[51] W.Hua, M.Cuiqin and Z.Lijuan, “A Brief Review of Machine Learning and Its Application.”, International conference on ICIECS, pp 1-4, 2009.

[52]] G.Holmes, A.Donkin and I.H.Witten, “WEKA: a machine learning workbench”, Proceedings of the 1994 second Australian and New Zealand Conference on Intelligent Information Systems, pp .357-361,1994.

[53] S.Ruggieri, “Efficient C4.5”, IEEE Trans on Knowledge and Data engineering, Vol.14, pp. 438-444, Aug.2002.

[54] Y.Ma, et.al, “Study of information network traffic identification based on C4.5 algorithm”, 4Th international conference on Wireless Communications, Networking and Mobile computing, pp.1-5, 2008.

[55] C.M. Higgins and R.M. Goodman, “Learning fuzzy rule-based neural networks for function approximation”, International Joint Conference on Neural Networks, vol.1, pp.251-256, 1992.

[56] Zhi-Yi Mai, et al “A new-rate distortion optimization using structural information in H.264
I-frame encoder” ACIVS 2005, LNCS 3708, pp. 435–441, 2005.

[57] N.Williams, S.Zander and G.Armitage, “A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification” ACM SIGCOMM computer communication review, vol.36, issue 5, pp 5-16, Oct 2006.

BIOGRAPHICAL INFORMATION

Thejaswini Purushotham was born in Bangalore, Karnataka, India. She received her bachelor's degree in Electronics and Communication from P.E.S.I.T, Bangalore in May 2008. She was a member of the Multimedia Processing Lab, headed by Dr. K.R. Rao. She worked as a Graduate Research Assistant for Dr. K. R. Rao. She did her internship with Damaka Inc. Her research interests are in the field of video compression, networking and wireless communications and she plans to join the video development group at Damaka Inc after graduation.