LOW COMPLEXITY AVS-M USING MACHINE LEARNING ALGORITHM C4.5

by

PRAGNESH R. RAMOLIA

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

MAY 2011

ACKNOWLEDGEMENTS

I would like to take this opportunity to convey my sincere gratitude to Dr. Rao, playing a role model to me. It was he who gave me the courage to undertake research work in a field which was completely strange to me. With his immense experience in the field and patience to undertake any challenge even at this age, he inspired me to do something different. I want to thank him for always being there for me, whenever I needed.

I also want to express my sincere thanks to Dr. Dongil Han for coming up with an industrial project to our team of multimedia processing lab. I was not sure to go for research in this field, until he came up with a project which sounded very interesting to me. I knew this was a part of an industrial project, which motivated me to undertake that task.

I am also thankful to Dr. Jonathan Bredow and Dr. Saibun Tjuatja for taking time out of their busy schedule to be the members of my defense committee. I am also thankful to all my friends at UTA for all their love, support and encouragement for me.

Finally, I would like to thank my family for always believing in me. Without their support I would not have come a long way here, I dedicate this thesis to my family.

April 13, 2011

ABSTRACT

"LOW COMPLEXITY AVS-M CODEC USING MACHINE LEARNING ALGORITHM C4.5"

PRAGNESH R. RAMOLIA, MS

UNIVERSITY OF TEXAS AT ARLINGTON, 2011

Supervising Professor: Dr. K.R. RAO

AVS China is the video coding standard developed by the AVS work group of China employing the latest video coding tools for achieving high compression while preserving the quality of the video [15]. AVS China video standard has been proven superior to the existing video standards such as MPEG-2 and MPEG-4 part 2, in terms of reduced complexity and coding efficiency. AVS is a set of integrated standard system which contains systems, video, audio and media copyright management. AVS-M, the 7th part of the system is video codec targeting mobile devices. AVS-M uses high efficiency tools like macroblock partitioning, entropy coding and motion estimation and motion compensation to exploit the temporal and spatial redundancies present inside the video sequence. This helps to save the number of bits used to encode the video sequence, in turn saving bandwidth of video transmission. But all these tools come at the cost of computational complexity. Motion estimation alone, consumes 75%-80% of the encoding time. With all these tools, AVS-M, simulated on a general purpose processor, can process upto 2 frames per second. Live streaming on the other hand will demand at least 15 frames per second. The huge gap between the power demanded and available power can be bridged by embedding special purpose hardware like FPGA (field programmable gate array) or ASICS (application specific integrated circuits). But these solutions are very expensive and also create problems like consuming lot of power and overheating problems in mobile devices. One possible way is to reduce the complexity of the encoder.

Various studies have been undertaken, to implement fast motion estimation algorithms in place of regular motion estimation block. Machine learning algorithm, is implemented to predict the macroblock mode decision for encoding. In the first step to achieve the desired result, many attributes of frames are extracted, and then fed to an .arff (attribute relation file format) file. .arff file is then used as input to a Weka tool. Weka is embodiment of various data mining algorithms. Using C4.5 of Weka, a decision tree in terms of if-else statements is obtained. The decision tree obtained here, traces back to maximum of 7 if-else statement executions, to decide the macroblock mode, which compared to the RDO (rate distortion optimization) process, is very less. Hence large amounts of time saving are expected, with the proposed encoder. As expected the proposed encoder achieved on an average 75%- 80% reduction in encoding time, compared to the original encoder AVS-M.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

LIST OF ILLUSTRATIONS viii

LIST OF TABLES x

LIST OF ACRONYMS AND ABBREVIATIONS xi

Chapter Page

1. INTRODUCTION 1

2. AVS PART – 7 4

2.1 AVS Introduction 4

2.2 Introduction to AVS- 5

2.2.1 Data structure of AVS-M 7

2.2.2 Embodiment of AVS-M 12

2.2.3 Various Levels in Jiben profile 13

2.3 Block mode prediction modes 17

2.3.1 Intra prediction 17

2.3.2 Inter prediction 20

2.3.3 Skip mode prediction 22

2.3.4 RD Optimization 22

2.4 Transform, Quantization and Entropy coding 23

2.4.1 Transform 23

2.4.2 Quantization 23

2.4.3 Entropy coding 24

2.5 The deblocking filter 25

2.6 Summary 26

3. DATA MINING, DATA MINING ALGORITHMS AND WEKA 27

3.1 Introduction to data mining 27

3.1.1 Steps for data mining 27

3.2 Classification of data mining methods 29

3.2.1 Classification methods in data mining 29

3.2.2 Supervised and unsupervised learning 32

3.3 Machine learning algorithm C4.5 33

3.3.1 Initial stage of C4.5 34

3.3.2 Tree construction algorithm 34

3.4 Weka 37

3.5 Summary 39

4. PROPOSED BLOCK DECISION METHOD 40

4.1 Introduction 40

4.2 Approach. 41

4.3 Experimental results 42

4.4 Conclusions and future work 56

APPENDIX

A. SNAPSHOTS OF THE VIDEO SEQUENCES USED 57

B. ATTRIBUTES EXTRACTED FROM THE VIDEO FRAMES 60

C. CONFIGUREATION FILE-ENCODER.CFG 70

REFERENCES 75

BIOGRAPHICAL INFORMATION 79

LIST OF FIGURES

Figure Page

1.1 Use of Multimedia data in different environments [26] 2

2.1 History of video coding standards [11] 3

2.2 Standard structure of AVS profiles [2] 4

2.3 Various Profiles of AVS in terms of key features and coding tools 5

2.4 Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame [45] 8

2.5 Slices [45] 9

2.6 MB structure for 4:2:0 format [45] 10

2.7 Scanning order in a MB for 4:2:0 format [45] 11

2.8 Data structure of AVS-M 11

2.9 Block diagram of AVS-M encoder [10] 13

2.10 Block diagram of AVS-M decoder [10] 13

2.11 Intra prediction modes [25] 18

2.12 Intra chroma prediction modes in AVS-M [25] 19

2.13 The positions of integer, half and quarter pixel samples [16] 21

2.14 Inter prediction block sizes [9] 24

2.15 Zigzag scanning pattern used for quantized transform coefficients [24] 23

2.16 Block edges of a macroblock to be filtered [45] 26

3.1 The process of knowledge discovery in data bases [41] 29

3.2 Typical neural networks [41] 30

3.3 Typical Bayesian network [41] 30

3.4 Typical tree [41] 31

3.5 (a) Supervised learning process block diagram (b) Unsupervised Learning process [39] 33

3.6 Classifier tree in terms of statements obtained by Weka.. 38

3.7 Tree structure obtained by Weka.. 39

4.1 Block diagram of the proposed encoder 40

4.2 Decision tree obtained from Weka tool.. 43

4.3 The classifier tree in terms of statements 44

4.4 Implementing the tree in AVS-M 45

4.5 Bar chart to compare encoding time of AVS-M and the proposed encoder 46

4.6 Bar chart to compare PSNR of luma component when encoded by AVS-M and the proposed encoder 48

4.7 Bar chart to compare PSNR of U component when encoded by AVS-M and the proposed encoder 49

4.8: Bar chart to compare PSNR of V component when encoded by AVS-M and the proposed encoder 51

4.9: Bar chart to compare PSNR of YUV component when encoded by AVS-M and the proposed encoder 52

4.10 Bar chart to compare number of bits used AVS-M and the proposed encoder 54

LIST OF TABLES

Table Page

2.1 Profiles of AVS with their targeted applications [37] 5

2.2 different parts of AVS [10] 6

2.3 Limits on the values of some syntax elements for level 1.x [45] 14

2.4 Limits on the syntax elements of Jiben profile for level 2.x [45] 15

2.5 Limits on the syntax elements of Jiben profile for level 3.x [45] 16

2.6 Most probable mode table [25] 19

2.7 K^{th} order exponential Golomb coding [6] 25

4.1 %accuracy in mode prediction by C4.5 embedded in Weka 42

4.2 Comparison of the encoding time between AVS-M and the proposed encoder 45

4.3 Comparison of the Y component when encoded by AVS-M
and the proposed encoder 47

4.4 Comparison of the U component when encoded by AVS-M
and the proposed encoder 48

4.5 Comparison of the V component when encoded by AVS-M
and the proposed encoder 50

4.6 Comparison of YUV component when encoded by AVS-M
and the proposed encoder 51

4.7 Comparison of number of bits used by AVS-M
and the proposed encoder 53

4.8 Comparison of encoding performances by AVS-M and the proposed encoder 53

LIST OF ACRONYMS AND ABBREVIATIONS

2D-VLC	2 dimensional variable length coding
3-G	3rd generation
4-G	4th generation
ARFF	attribute relational file format
AVS	audio video coding standard
AVS-M	audio video coding standard- mobile video
C_b	blue difference
C_r	red difference
DC	direct current (here refers to the average of coefficients)
DIP	direct intra prediction
DM	data mining
GOP	group of pictures
HD	high definition
HDTV	high definition television
IBL	instance based learning
ICT	integer cosine transform
IDR	instantaneous decode refresh
IMS	IP multimedia subsystem
IP	internet protocol
I-picture	intracoded picture
ITU-T	international telecommunication union, telecommunication sector
KDD	knowledge discovery in data bases
LTE	long term evolution
MB	macroblock

MPEG	moving picture experts group
MPM	most probable mode
NAL	network abstraction layer
P- Picture	predictive picture
PBP	padding before prediction
PSNR	peak signal to noise ratio
QP	quantization parameter
RDO	rate distortion optimization
SAD	sum of absolute differences
SCI	simplified chroma intraprediction
SD	standard definition
SEI	supplemental enhancement information
SVM	support vector machines
Th	threshold
TSFT	two step four taps
VLC	variable length coding
Weka	Waikato environment for knowledge analysis
Y	luma

CHAPTER 1

INTRODUCTION

The recent deployment of 3-G (3rd generation) network and 4-G (4th generation) / LTE (long term evolution) has lead to exponential increase in multimedia data in mobile devices. Additionally recent developments in multimedia processing have changed the face of multimedia communication over the internet and mobile devices. Multimedia traffic, and more specifically digital video traffic, has grown exponentially in recent years. Everybody in this era of digital world, wants the best possible, be it the communication speed or quality of multimedia content they are getting. This demand puts pressure on entire fraternity in research area to develop networks as fast as possible and with the least possible transmission errors, computers and laptop devices to process the data as fast as possible and mobile devices with maximum number of features possible. One of the most demanding situations is created by the live-streaming of videos over the internet. In this world of 3-G and 4-G networks people expect to view HD (high definition) videos over mobile devices and at speed of 30 frames per second. Videoconferences, streaming video and digital television are some of the currently available video services. Furthermore, HDTV (high definition television) systems are a contemporary revolution in the world of domestic multimedia applications, offering the best video qualities available in the broadcast systems. This demand has forced the network providers to improve their networks and calibrate networks to the current load on a regular basis. Figure 1.1 shows the use of multimedia contents in different environments and communication of that data across different types of networks, on different types of devices. Now transmitting a video sequence in SD (standard definition) and now even in HD, requires a large amount of bandwidth, and bandwidth is a very expensive resource, network providers cannot blindly allocate more and more bandwidth to meet the demands of the users. It is at this point where the video compression standards come into picture. Reducing considerably the amount of data and therefore reducing the resources required for their transmission and storage, while guaranteeing a high quality image, compression schemes enable the storage and transmission of video information in a more compact form.AVS (audio video coding standard) China is the video coding standard developed by the AVS work group of China employing the latest video coding tools for achieving high compression while preserving the quality of the video [15]. AVS China video standard has been proven superior to the existing video standards such as MPEG-2 and MPEG-4 part 2 in terms of reduced complexity and coding efficiency. In other words, this codec can provide high quality video at the same bandwidth or the same quality video at a reduced bandwidth. Due to these robust features, AVS China standard has been adopted for video compression and transmission in this thesis. AVS-M, (AVS mobile) is the video codec targeting mobile devices. If a live streaming of a video is to take place from a device, it should be capable of processing at least 15 frames (to create the perception of moving video) and transmit it. Now if AVS-M is to be implemented in software, it cannot process more than 2 frames per second, to narrow this huge gap between the required processing power and the power available with general processing hardware, AVS-M is to be implemented in special hardware (digital signal processors) to empower the device. But these specialized hardware solutions then consume a lot of power and create over-heating device problems. To overcome this problem, an effort is made here to implement a machine learning algorithm C4.5 [35] [43] inside the AVS-M, and reduce the complexity present inside the codec.

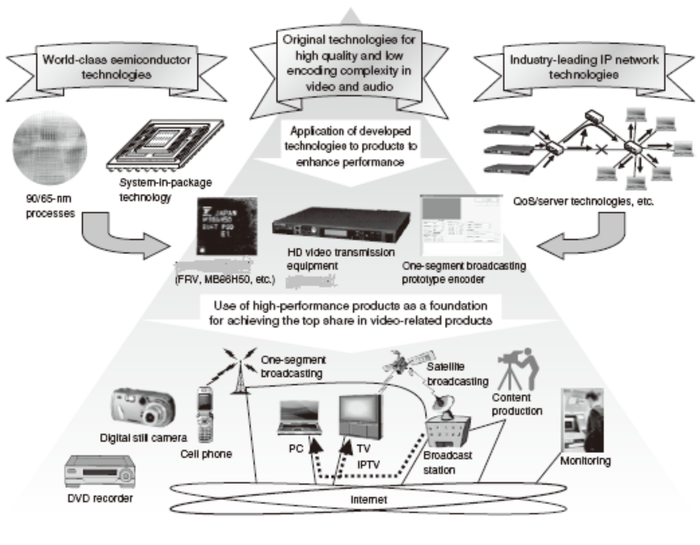

World-class semiconductor technologies
Original technologies for high quality and low encoding complexity in video and audio
Industry-leading IP network technologies
Application of developed technologies to products to enhance performance
90/65-nm processes
System-in-package technology
QoS/server technologies, etc.
HD video transmission equipment
(FRV, MB86H50, etc.)
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 Use of multimedia data in different environments [26].

CHAPTER 2

AVS PART-7

2.1 AVS Introduction

AVS China was developed by the AVS workgroup, and is currently owned by China. This audio and video standard was initiated by the Chinese government in order to counter the monopoly of the moving picture experts group (MPEG) standards [24]

AVS is a set of integrated standard system which contains systems, video, audio and media copyright management. Different parts of audio video coding standard (AVS) and their applications are shown in Table 2.2 [10]. The development of various video coding standards over the years is shown in Figure 2.1. Also Figure 2.2 shows various AVS profiles and their applications.

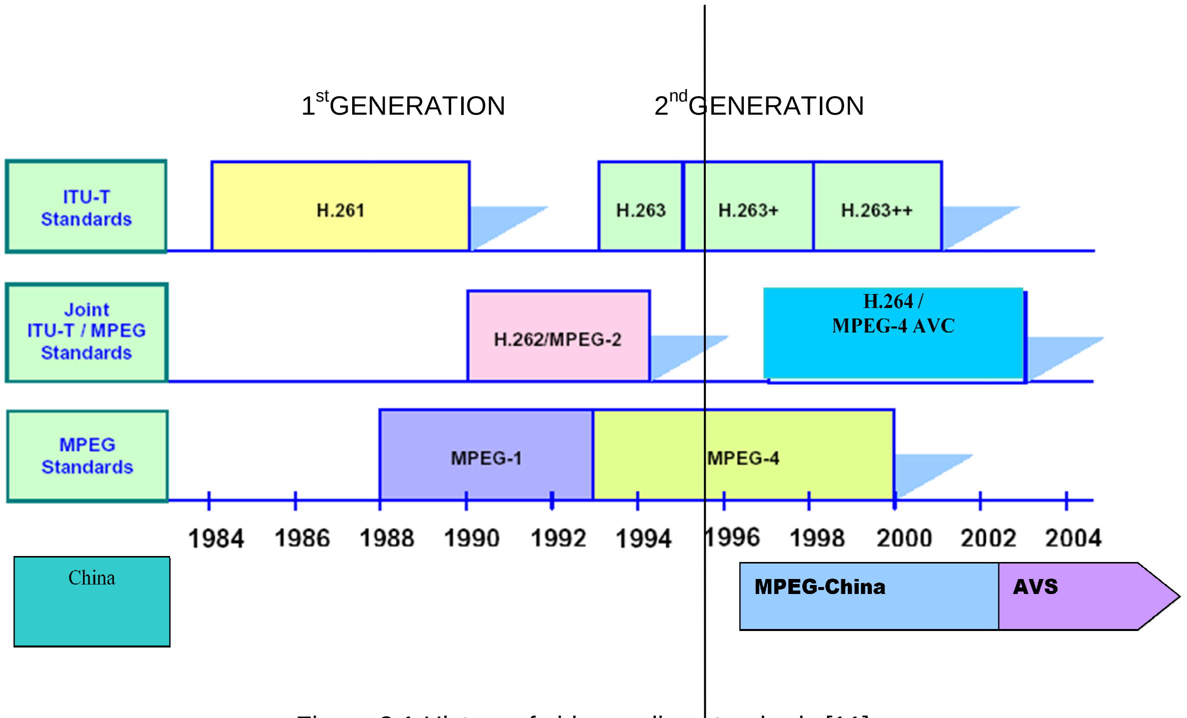


Figure 2.1 History of video coding standards [11]

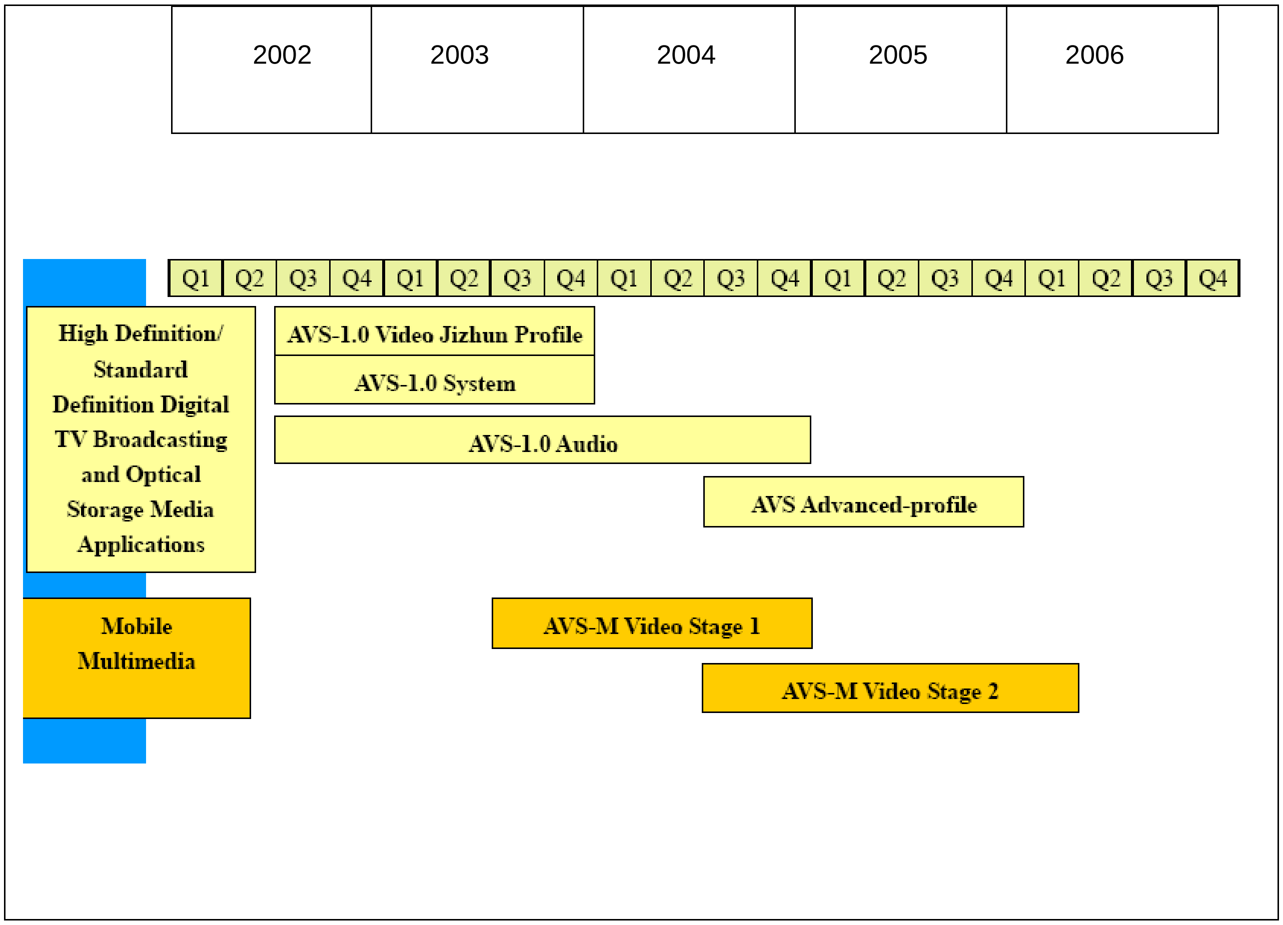

2002
2003
2004
2005
2006
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
High Definition/ Standard Definition Digital TV Broadcasting and Optical Storage Media Applications
AVS-1.0 Video Jizhun Profile
AVS-1.0 System
AVS-1.0 Audio
AVS Advanced-profile
Mobile Multimedia
AVS-M Video Stage 1
AVS-M Video Stage 2

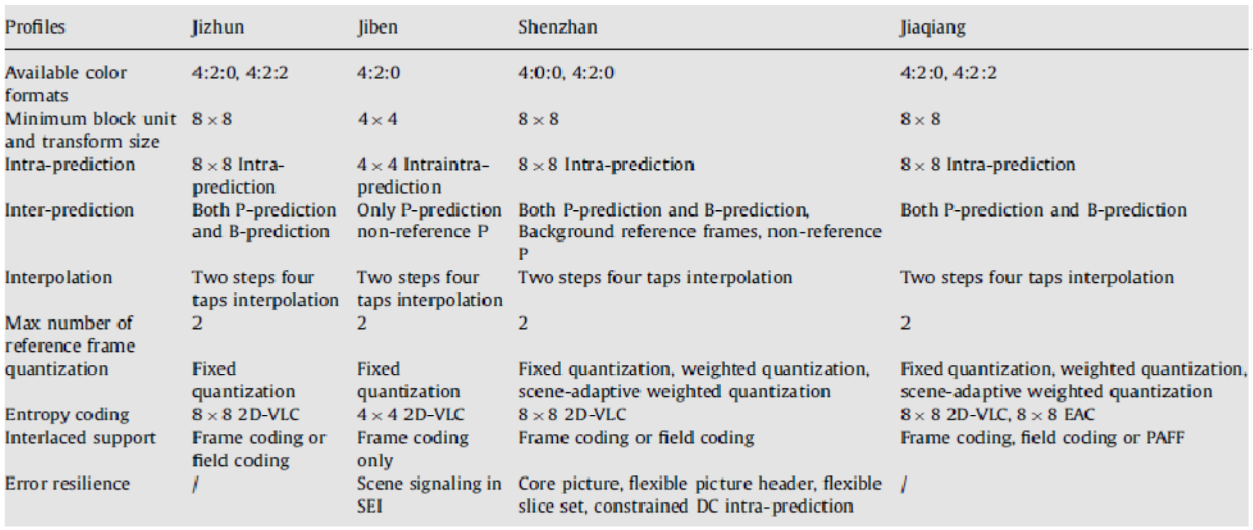
Figure 2.2 Standard structure of AVS profiles [2]

This standard can serve a wide range of bit rates, resolutions and qualities. Considering the capability of interoperation, a limited number of subsets of the syntax are stipulated by means of ‘profiles’ and ‘levels’. A ‘profile’ is a subset of the syntax elements, semantics, and algorithmic features of this standard. A ‘level’ is a specified set of limits on the syntax elements and the values that may be taken by the syntax elements of a certain profile. The purpose of defining profiles and levels is to target different applications for different profiles and levels. Table 2.1 shows various profiles with their targeted applications.

Table 2.1 Profiles of AVS with their targeted applications [37].

Profile	Key applications:
Jizhun profile	Television broadcasting, HDTV (High Definition Television)etc.
Jiben profile	Mobility applications
Shenzhan pfofile	Video surveillance systems
Jiaqiang profile	Multi-media applications, entertainment.

Figure 2.3 shows various profiles of AVS with their key features available for coding and some high efficiency coding tools used to achieve the requirements for the target application.



Profiles	Jizhun	Jiben	Shenzhan	Jiaqiang
Available color formats	4:2:0, 4:2:2	4:2:0	4:0:0, 4:2:0	4:2:0, 4:2:2
Minimum block unit and transform size	8 × 8	4 × 4	8 × 8	8 × 8
Intra-prediction	8 × 8 Intra-prediction	4 × 4 Intraintra-prediction	8 × 8 Intra-prediction	8 × 8 Intra-prediction
Inter-prediction	Both P-prediction and B-prediction	Only P-prediction non-reference P	Both P-prediction and B-prediction, Background reference frames, non-reference P	Both P-prediction and B-prediction
Interpolation	Two steps four taps interpolation	Two steps four taps interpolation	Two steps four taps interpolation	Two steps four taps interpolation
Max number of reference frame	2	2	2	2
quantization	Fixed quantization	Fixed quantization	Fixed quantization, weighted quantization, scene-adaptive weighted quantization	Fixed quantization, weighted quantization, scene-adaptive weighted quantization
Entropy coding	8 × 8 2D-VLC	4 × 4 2D-VLC	8 × 8 2D-VLC	8 × 8 2D-VLC, 8 × 8 EAC
Interlaced support	Frame coding or field coding	Frame coding only	Frame coding or field coding	Frame coding, field coding or PAFF
Error resilience	/	Scene signaling in SEI	Core picture, flexible picture header, flexible slice set, constrained DC intra-prediction	/

Figure 2.3 various profiles of AVS in terms of some key features and coding tools. [45]

2.2 Introduction to AVS-M

AVS-M is the seventh part of the standards developed by the audio video coding standard (AVS) [11] workgroup of China with emphasis on video coding for mobile systems and devices with limited computation capability and power consumption. AVS-M standard can cover a broad range of applications which include mobile multimedia broadcasting, (internet protocol) IP multimedia subsystem (IMS), multimedia mailing, and multimedia services over packet networks, videoconferencing, video phone, and video surveillance.

Table 2.2 Different parts of AVS [10]

Part	Name
1	System
2	Video
3	Audio
4	Conformance test
5	Reference Software
6	Digital media rights management
7	Mobile video

Table 2.2 - *continued*

8	Transmit AVS via IP network
9	AVS file format
10	Mobile speech and audio coding

In this standard, number of techniques may be used to achieve highly efficient compression, including interprediction, intraprediction, transform, quantization, entropy coding, etc. Intercoding uses motion vectors for block-based interprediction to exploit temporal statistical dependencies among adjacent pictures. Intracoding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantized, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are multiplexed with the quantization coefficients and encoded using entropy coding.

2.2.1 Data structure of AVS-M [45]

AVS-M has two bit stream formats: the network abstraction layer (NAL) and unit stream format or the byte stream format. The NAL unit stream consists of a sequence of syntax structures called NAL units. The byte stream format can be constructed from the NAL unit streams, which include a sequence of NAL units ordered in decoding order. Each NAL unit is prefixed with a start code.

The video sequence is encoded in terms of layers: coded video sequence, picture, slice, macro block and block where video sequence is the highest layer and block is the lowest layer. Higher level layer encloses the lower level layers along with their headers.

Coded Video Sequence [45]

This is the highest structure present in the bit stream. The bit stream of coded video sequence starts with an IDR (instantaneous decode refresh) picture, followed by zero or more non-IDR pictures up to but not including the next IDR picture or the end of the bit stream. Decoded pictures are ordered consecutively in the bit stream, whose order is the same as decoding order.

Pictures [45]

Picture represents one frame of a sequence. Coded bit stream for a picture consists of a picture header and three matrices as data: Y (luma), C_b (blue difference) and C_r (red difference) and relation between Y, C_b, C_r and primary analog (Red, Green and Blue) components. A decoder outputs a series of frames. Two successive frames have an interval time, called as one frame duration. Picture decoding process includes parsing (of header and matrices information) process and decoding process. This standard supports the 4:2:0 formats. In this format the C_b and C_r matrices shall be half the sizes of Y-matrix in both horizontal and vertical dimensions shown in Figure 2.4. "O" denotes the locations of luma samples Y and "X" denote the locations of chroma samples C_b and C_r.

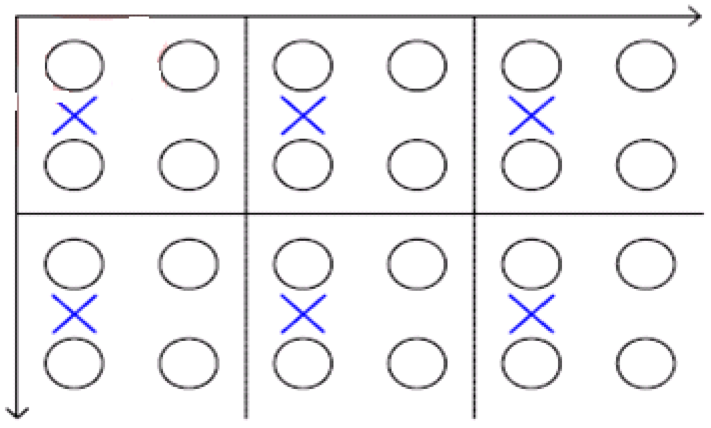


Figure 2.4 Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame [45]

This standard specifies two types of decoded pictures. An intra decoded picture (I-picture). A forward inter decoded picture (P-picture). P picture can have a maximum of two reference frames for forward prediction. A motion vector can exceed the boundaries of the reference picture. In this case, the nearest pixel within the frame shall be used to extend the boundary. A reference pixel block shall not exceed the picture boundary beyond 16 pixels for luma samples and 8 pixels for chroma samples, in both horizontal director and vertical directions.

Slice

A slice is a sequence of macro blocks ordered consecutively in the raster scan. MBs (macroblocks) within a slice never overlap with each other and neither does the slice overlap with other slice. The slices cover the whole area of frame, and except in loop filtering process, the decoding process for MBs within a slice shall not use the data from other slices of the picture. Figure 2.5 shows an example of the division of picture into slices.

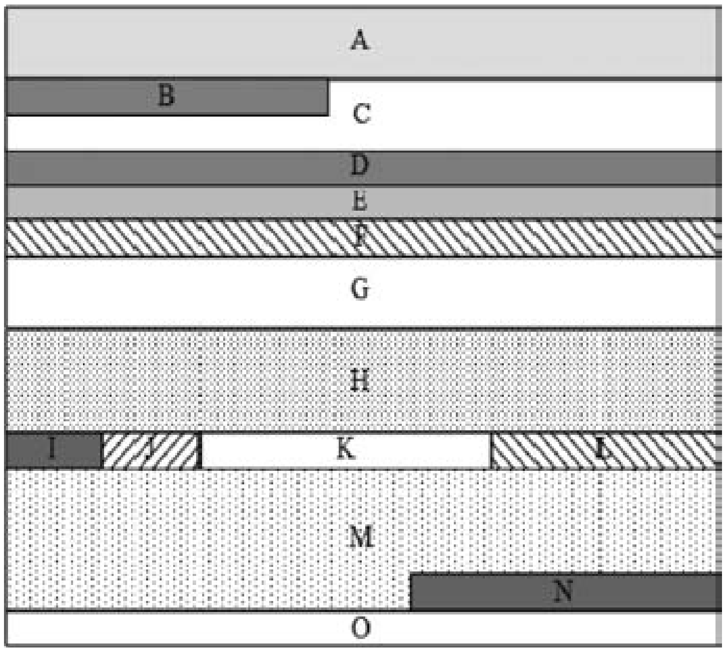


Figure 2.5 Slices [45]

Macroblock

Slice is divided into MBs. A MB is pixel domain block of size 16x16. The upper-left sample of each macro block shall not exceed picture boundary. The partitioning is used for motion compensation. The number in each rectangle specifies the order of appearance of motion vectors and reference indices in a bit stream. The partition of image into slice is done so that there is no overlapping of macro blocks of two slice and no overlapping of two slice for a macro block, this is ensured to make sure that all the macro blocks in a slice can be decode using only the macro blocks in the same slice as their neighbors.

A MB includes four 8×8 luma blocks (Y) and two 8×8 chroma blocks (one C_b and one C_r) as shown in Figure 2.6. The number in each rectangle specifies the coding order of each 8×8 block of a macro-block in a bit-stream.

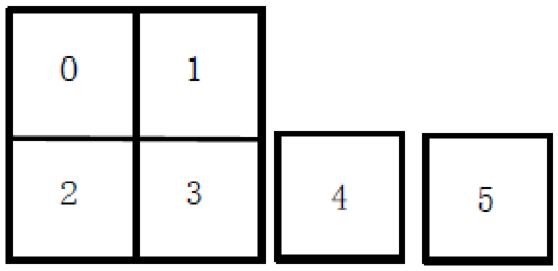


Figure 2.6 MB structure for 4:2:0 format [45]

A macro-block if coded with a sub-block type, each 8x8 block is portioned in sub-blocks of size ranging from 8x8 to 4x4. The Figure 2.7 shows the partition of a Macro-block in to 16 4x4 luma blocks and 2 chroma blocks portioned into 4 4x4 blocks each. The numbers in the boxes represent the order of scanning while encoding a macro block.

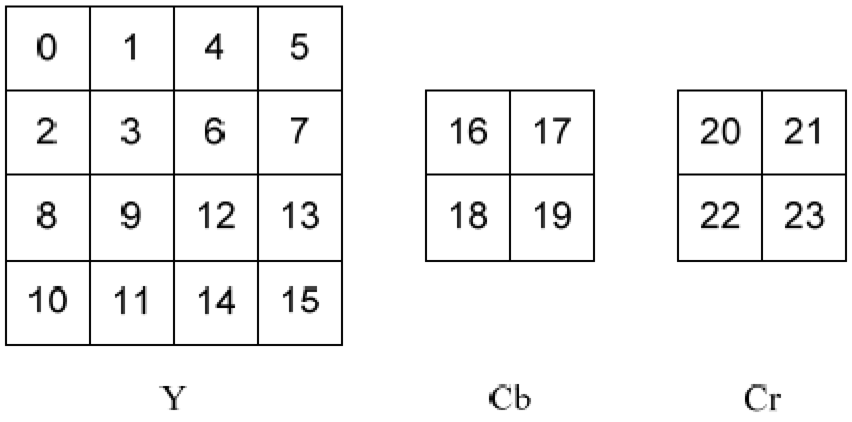


Figure 2.7 Scanning order in a MB for 4:2:0 format [45]

Overall layered data structure present in the encoded video sequence bit-stream is as shown in Figure2.8**.**

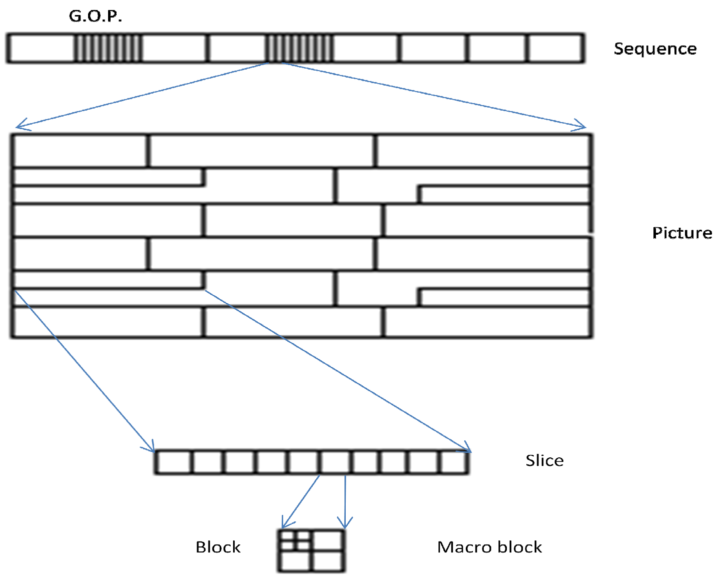


Figure 2.8 Data structre of AVS-M.

2.2.2 Embodiment of AVS-M

The block diagrams of AVS-M encoder and decoder [10] are depicted in Figure-2.9 and Figure-2.10 respectively. MB needs to be predicted (intra predicted or inter predicted). In an AVS-M encoder, the S_0 is used to select the right prediction method for current MB whereas in the decoder, the S_0 is controlled by the MB type of current MB. The predicted MB is then subtracted from the original MB to obtain the prediction residue. The residue is then transformed by ICT and then quantized. The quantized coefficients along with the motion vectors (if the MB was inter-predicted) are entropy coded with 2-D VLC and transmitted to decoder side. The encoder also maintains the local decoder, on its side, to get the exact reconstructed frame, as will be obtained on the decoder side. This is done so that the exact frame, the encoder uses for prediction, is used by the decoder for the reconstruction. So both encoder and decoder work in synchronization. If this is not maintained then the quantization error goes on accumulating.

Once the decoder obtained the AVS-M bitstream, it entropy decodes the stream, separates the MVs if a macroblock is interpredicted, and then adds the inverse transformed, quantized coefficients to the predicted macroblock (inter or intra predicted), and then applies the deblocking filter on the edges, hence video frame is ready to played by any player. The video frame reconstructed is also stored in a frame buffer for future reference, for inter prediction of future frames and intra prediction of future MBs of the same frame. The block diagram of decoder is shown in Figure 2.10.

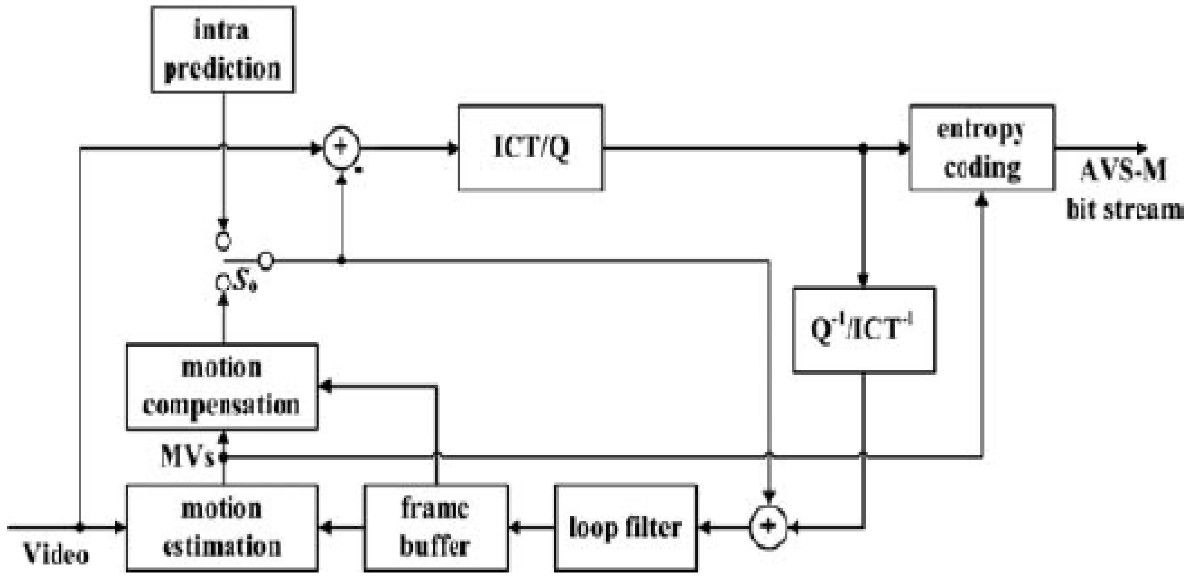


Figure 2.9 Block diagram of AVS-M encoder [10]

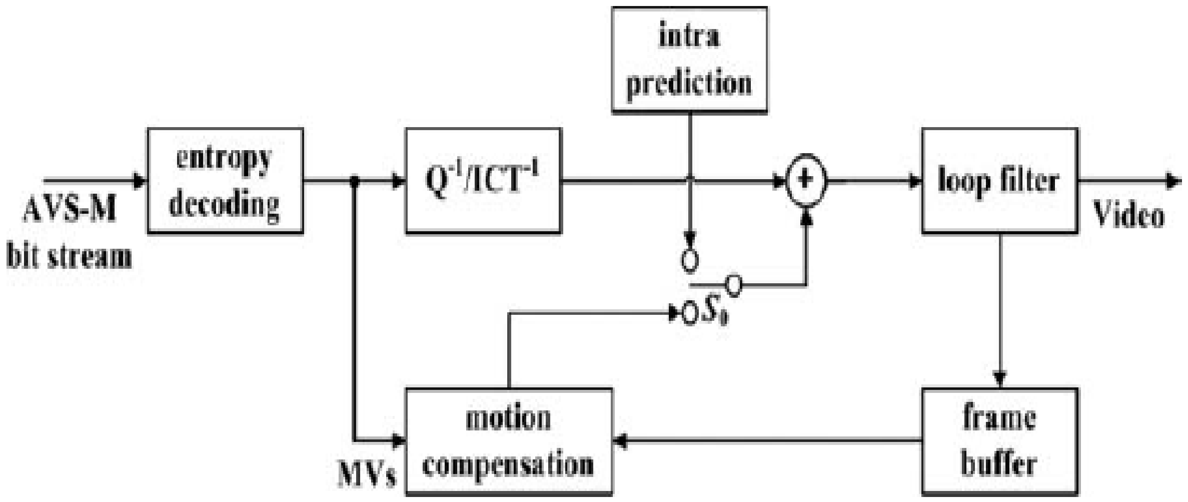


Figure 2.10 Block diagram of AVS-M decoder [10].

2.2.3 Various levels in Jiben profile

The part of AVS considered here is AVS-M, also known as Jiben profile. AVS-M defines Jiben Profile wit h 9 different levels: Tables 2.3 through 2.5 specify limits on certain syntax elements for a particular level.

Table 2.3 Limits on the values of some syntax element for level-1.X [45]

Profile	Level			
	1.0	1.1	1.2	1.3
Maximum number of MBs per second	1485	1485	6000	11880
Maximum number of MBs per frame	99	99	396	396
Maximum MBs coding bit rate (bits/second)	4096	4096	4096	4096
Maximum bit rate (bits/second)	64000	128000	384000	768000
Maximum decoded picture buffer size (bytes)	114048	114048	456192	456192
Maximum coded picture buffer size (bits)	175000	350000	1000000	2000000
Maximum range of horizontal motion vector	[-2048,2047.75]	[-2048, 2047.75]	[-2048, 2047.75]	[-2048, 2047.75]

Table 2.3 - *continued*

Maximum range of vertical motion vector	[-32,31.75]	[-32,31.75]	[-32,31.75]	[-32,31.75]
Max SubMbRectSize	572	572	572	572
Minimum compression ratio	2	2	2	2

Table 2.4 Limits on the syntax elements of Jiben profile for level 2.X [45]

Parameter	Level		
	2.0	2.1	2.2
Maximum number of MBs per second	11880	19800	20250
Maximum number of MBs per frame	396	792	1620
Maximum MB coding bit rate (bits/second)	4096	4096	4096
Maximum bit rate (bits/second)	2000000	4000000	4000000
Maximum decoded picture buffer size (bytes)	456192	912384	15552000
Maximum coded picture buffer size (bits)	2000000	4000000	4000000

Table 2.4 - *continued*

Maximum range of horizontal motion vector	[-2048,2047.75]	[-2048, 2047.75]	[-2048, 2047.75]
Maximum range of vertical motion vector	[-32,31.75]	[-256,255.75]	[-256,255.75]
Max SubMbRectSize (bytes)	572	572	572
Minimum compression ratio	2	2	2

Table 2.5 limits on the syntax elements of Jiben profile for Level 3.X [45]

Parameter	Level	
	3.0	3.1
Maximum number of MBs per second	36000	40500
Maximum number of MBs per frame	1620	1620
Maximum MB coding bit rate (bits/second)	4096	4096
Maximum bit rate (bits/second)	6000000	8000000
Maximum decoded picture buffer size (bytes)	1555200	1555200
Maximum coded picture buffer size (bits)	6000000	8000000
Maximum range of horizontal motion vector	[-2048,2047.75]	[-2048,2047.75]

Table 2.5 - *continued*

Maximum range of vertical motion vector	[-256,255.75]	[-256,255,75]
Max SubMbRectSize (bytes)	572	572
Minimum compression ratio	2	2

2.3 Block mode prediction modes

2.3.1 Intra prediction

The intra predictions are derived from the neighboring pixels in left and top blocks. There are 9 intra_4*4 modes as shown in Figure 2.11. The unit size of intra prediction is 4×4 because of the 4×4 integer cosine transform (ICT) used by AVS-M. Some specific techniques are working together with 4_4 intraprediction, such as direct intra prediction (DIP), padding before prediction (PBP) and simplified chrominance intra-prediction (SCI) [16]. Prediction of most probable mode from neighboring blocks is also used [16]. Table 2.6 [25] shows the most probable mode for the current block based on the left (L) and upper (U) blocks. Mode '-1' is assigned to L or R whenever there is no left or upper block or that block is inter coded. One flag at block level indicates if the particular block is encoded with "most probable mode" and one flag at macro block level indicates the use of DIP [17][18]. If one macro block is marked as DIP-mode, it infers that each of the 16 luminance 4*4sub-blocks in this macro block takes the most probable mode as its intra-prediction mode, even though the intra-prediction mode for each 4*4 sub-block might be different no more mode information is transmitted in bit-stream, saving a large number of bits. The probability of occurrence of most probable mode is very high [22] thus saving a lot of bits in entire process. PBP is applied for both luminance and chrominance components, during which the reference pixels "Ds" and "Hs" are padded from "D" and "H" respectively, so as to skip conditional test of availability of up-right and down-left reference pixels. SCI means that only DC, vertical and horizontal modes are available for chrominance components [19].

Adaptive intra-prediction enables 4*4_intra-prediction to be applied along with 8*8_intra-prediction, using an indicator at macro block header in AVS-2[20]. Besides, mapping is needed between the modes used in 4*4_intra-prediction and those in 8*8_intra-prediction before the prediction of most probable mode, if the current block and its neighboring blocks are using different block-sized intra-predictions.

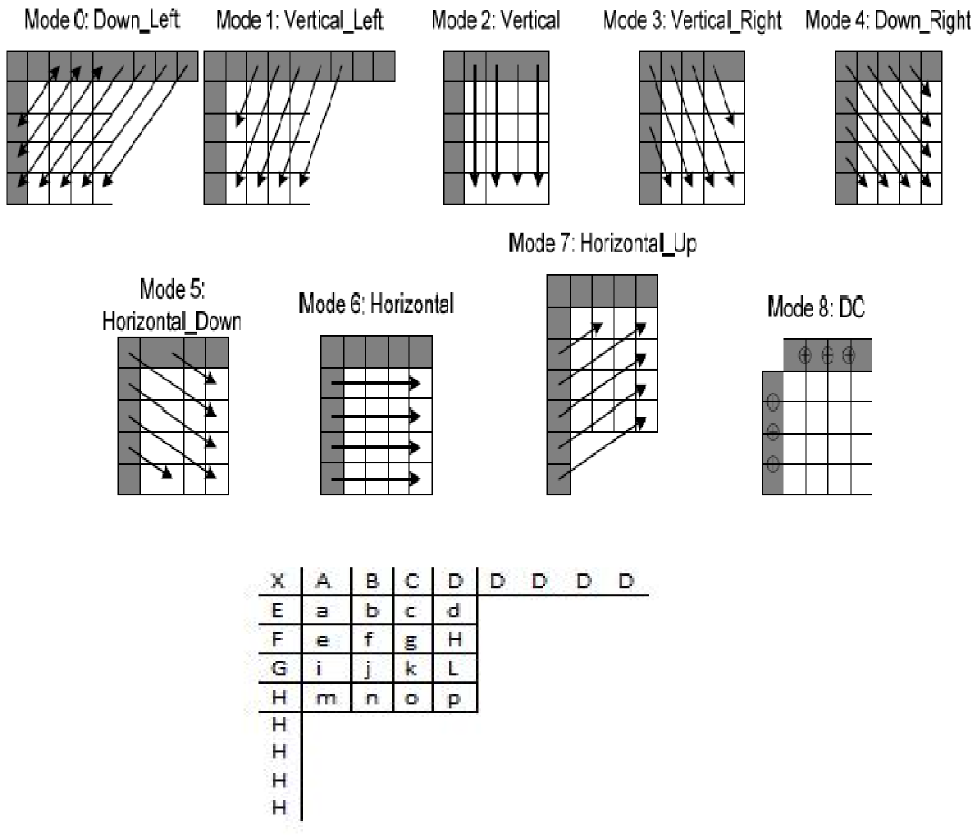


Figure 2.11 Intra prediction modes [25]

If current block is a chroma block, the prediction mode Intra_Chroma_Pred_Mode is equal to intra_Chroma_pred_mode. Every 4×4 chroma block of the macro-block uses the same prediction mode. The 4×4 chroma intra prediction modes are illustrated Figure 2.12

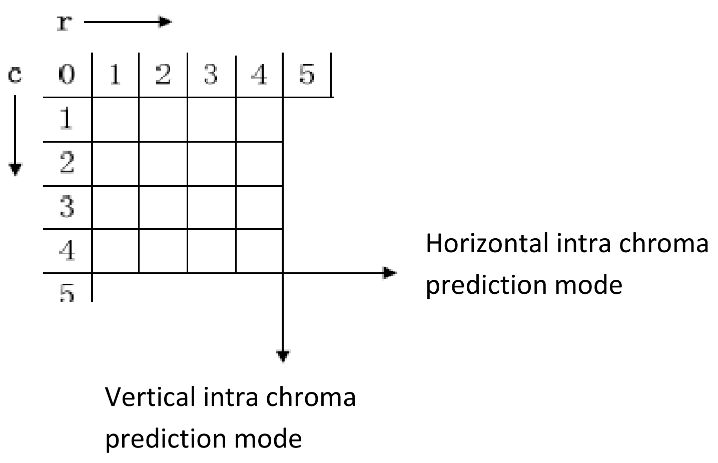

r
c
0
1
2
3
4
5
1
2
3
4
5
Horizontal intra chroma prediction mode
Vertical intra chroma prediction mode

Figure 2.12 Intra chroma prediction modes in AVS-M[25]

Table 2.6 Most probable mode table [25]

U→ ↓ L	-1	0	1	2	3	4	5	6	7	8
-1	8	8	8	8	8	8	8	8	8	8
0	8	0	0	2	0	0	0	2	0	2
1	8	2	1	2	2	2	2	2	2	2
2	8	2	2	2	2	2	2	2	2	2

Table 2.6 - *continued*

3	8	2	1	2	3	4	5	2	7	2
4	8	4	4	2	5	5	5	6	5	5
5	8	5	5	2	5	5	5	6	5	5
6	8	6	6	6	6	6	6	6	6	6
7	8	7	7	2	7	7	7	6	7	7
8	8	0	1	2	3	4	5	6	7	8

(L indicates the intra mode for left macroblock and U indicates the intra mode for right macroblock)

2.3.2 Inter prediction

To remove temporal redundancy in video sequence, inter-frame prediction (inter-prediction) predicts from previously decoded frames/fields. A number of techniques jointly contribute to the coding efficiency of inter prediction in AVS-video. There are two types of inter prediction: P-prediction and Bi-prediction. AVS-M supports only P-prediction, which uses only past decoded frames as its reference frames.

The inter predictions are derived from the decoded frames. Seven types of block sizes, 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 are supported in AVS-M as shown in Figure-2.14[9]. The precision of a motion vectors in inter prediction is up to 1/4 pel (pixel) accurate, since the motion-compensated prediction in AVS-video allows motion vector accuracy up to one quarter pel, corresponding reference pixel values of fractional motion vectors are obtained by sub pel interpolation as shown in Figure 2.13.

Default sub-pel interpolation in AVS-video is called as two steps four taps (TSFT) interpolation [21] and three kinds of filters are applied, respectively, onto sub-pels of different positions. The two steps four taps interpolation applies a filter with (-1, 5, 5, -1) coefficients to get the half-pel reference pixel values as the first step. A filter with (1, 7, 7, 1) coefficients is applied for quarter-pel reference pixel values either horizontally or vertically as the second step. The exception of the second step is that for quarter pel reference pel values of e, g, p, r, where a diagonal bilinear filter is used.

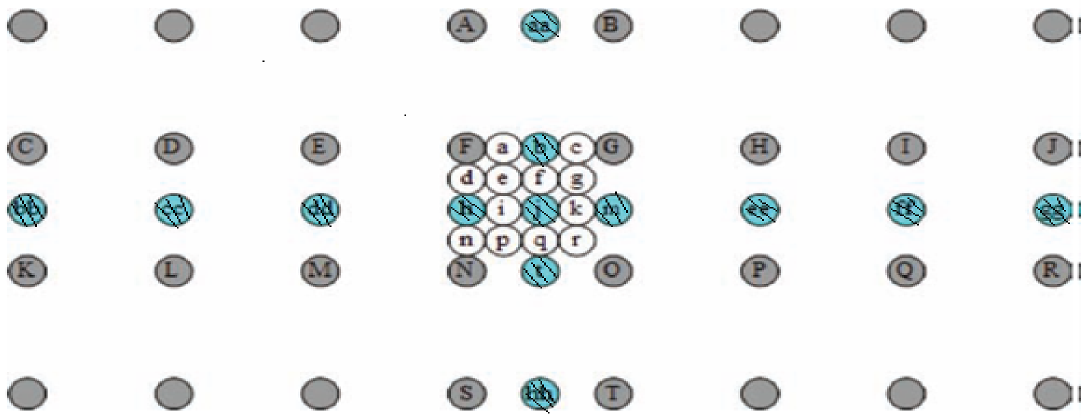


Figure 2.13 The position of integer, half and quarter pixel samples [16]

(Hatched lines show half pixels; empty circles are quarter pixels; capital letter shows full pixel)

Small block sizes perform better than large ones for lower image resolution. 4x4 is the unit of transform [10, 23], intra prediction and smallest block size motion compensation in AVS Part 7.

2.3.3 Skip mode prediction

When a macroblock is decided to be encoded with skip mode, no information for that MB is sent except the flag that it is encoded with skip mode. Now when decoder sees that the MB is encoded with skip mode, decoder uses the default reference marked zero in the buffer and motion vectors are calculated based on the motion vectors of neighboring MBs, left macroblock, up macroblock MVs.[45]

2.3.4 RD Optimization

For I-Frame an RD cost is calculated for each of the intra-block mode, by (2.1) to select the best mode out of available 9 intra modes.

$$\text{RD-Cost (mode)} = D\text{ (mode)} + \lambda * R\text{ (mode)} \qquad (2.1)$$

where λ is a lagarangian multiplier, which is derived based on the rate-cost plot optimization. RD-Cost (mode) is the rate-distortion cost for a particular mode for a block, and D (mode) represents the distortion if the block is coded with that mode, and R (mode) is the bit-rate produced if the block is coded with that particular mode. So to decide a block mode for one block, all the 9 mode costs are calculated. For calculating each cost the encoder needs to transform, quantize and entropy code a block with all the modes, to calculate R (mode) because R (mode) is the amount of bits used to encode a block with that mode. Also encoder has to perform entropy decoding, dequantization and inverse transform to reconstruct the image on the encoder side itself, to calculate the D (mode) because D (mode) is the difference between the original image and the image after reconstruction. After calculating the best RD-Cost for all 16 blocks in the MB, the encoder calculates the RD-Cost of the MB if all the blocks are coded with MPM (most probable mode), and if it happens to be less than sum of the best RD-Costs of all 16 blocks, DIP (direct intra prediction) is used to encode the MB.

For P-frames, encoder calculates the cost for all inter-modes. The best mode is calculated for intra prediction. Then best intermode is selected based on R-D optimization and then R-D cost for skip-macro block is also calculated. The encoding mode with the least cost is selected and MB is encoded with that mode.

2.4 Transform, quantization and entropy coding

2.4.1 Transform

The motion compensated residuals are transformed with 4×4 ICT (integer cosine transform). The 4x4 transform used in AVS is [45]. ICT are integer separable and integer precision. It is designed to minimize decoder implementation complexity.

$$T_4 = \begin{bmatrix} 2 & 3 & 2 & 1 \\ 2 & 1 & -2 & -3 \\ 2 & -1 & -2 & 3 \\ 2 & -3 & 2 & -1 \end{bmatrix}$$

2.4.2 Quantization

An adaptive uniform quantizer is used to perform the quantization process on the 4×4 transform coefficients matrix [10] [24] [25].The step size of the quantizer can be varied to provide rate control. In constant bit rate operation, this mechanism is used to prevent buffer overflow. The transmitted step size quantization parameter (QP) is used directly for luminance coefficients and for the chrominance coefficients it is modified on the upper end of its range. The quantization parameter may optionally be fixed for an entire picture or slice. If it is not fixed, it may be updated differentially at every macroblock. The quantization parameter varies from 0 to 63 in steps of one. The uniform quantization process is modified to work together with the transform in order to provide low complexity decoder implementation. Quantized coefficients are scanned using the zigzag pattern shown in Figure-2.15 [24].

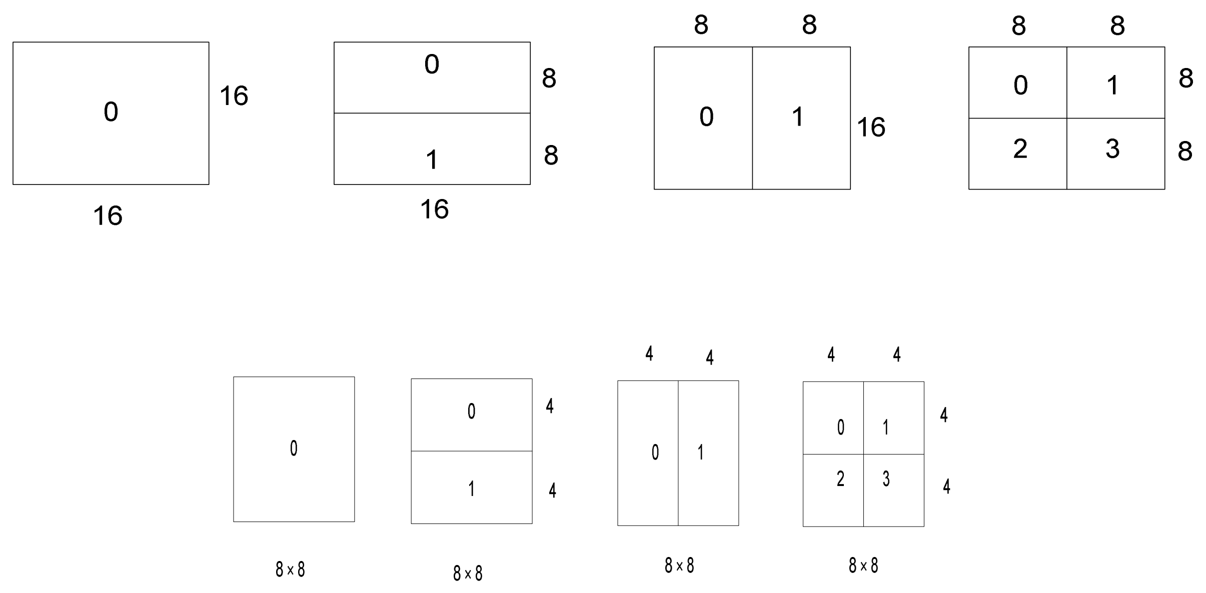
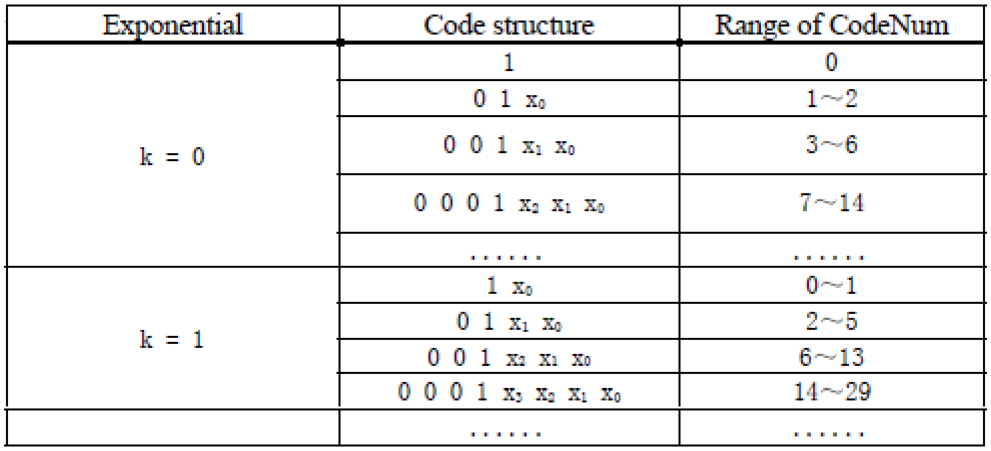


Figure 2.14 Inter-prediction block sizes [9]

2.4.3 Entropy coding

AVS-M uses exp-Golomb code[6], for entropy coding, as shown in Table-2.7 to encode syntax elements such as quantized coefficients, macro block coding type, and motion vectors. Eighteen tables are used for encoding quantized coefficients. The encoder uses the run and the absolute value of the current coefficient to select Table 2.7. AVS-M employs an adaptive variable length coding (VLC) coding technique [10].

Table 2.7 Kth order exponential Golomb coding [6]



Exponential	Code structure	Range of CodeNum
k = 0	1	0
	0 1 x_0	1~2
	0 0 1 x_1 x_0	3~6
	0 0 0 1 x_2 x_1 x_0	7~14
	......	......
k = 1	1 x_0	0~1
	0 1 x_1 x_0	2~5
	0 0 1 x_2 x_1 x_0	6~13
	0 0 0 1 x_3 x_2 x_1 x_0	14~29
	......	......

The reconstructed image is the sum of prediction and current reconstructed error image. In Figures 2.9 and 2.10, encoder adds the inverse quantized and inverse transformed coefficients to the predicted frame (either intra-predicted or inter-predicted selected through S0 switch) to get the reconstructed frame.

2.5 The deblocking filter

AVS-M uses the de-blocking filter on the reconstructed frame, and then stores them in the buffer for future reference. The de-blocking process directly acts on the reconstructed reference first across vertical edges and then across horizontal edges. Since, different image regions and different bit rates need different smoothes; the de-blocking filter is adjusted accordingly in AVS-M depending on activities of blocks and quantization parameters. Except edges at image boundary and slice edges with disable_loop_filter_slice_flag equal to 1, all 4×4 block edges of a macroblock are filtered. Loop filtering regards a macroblock as a unit. Every macroblock is processed as follows:

Luma and chroma are filtered separately, as shown in Figure 2.16. First, vertical edges are filtered from Left to right, and then horizontal edges are filtered from top to bottom. Sample values that have not been modified by the loop filtering process are used as input to the filtering process on the current macroblock. Sample values may be modified during the filtering process of current macro block. The input values of horizontal edges filtering process are the sample values that have been modified during [45] vertical edge filtering process of current macro block.

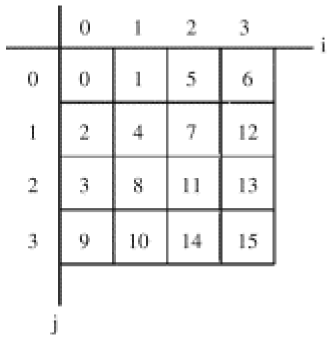


Figure 2.15 Zigzag scanning pattern used for quantized transform coefficients [24]

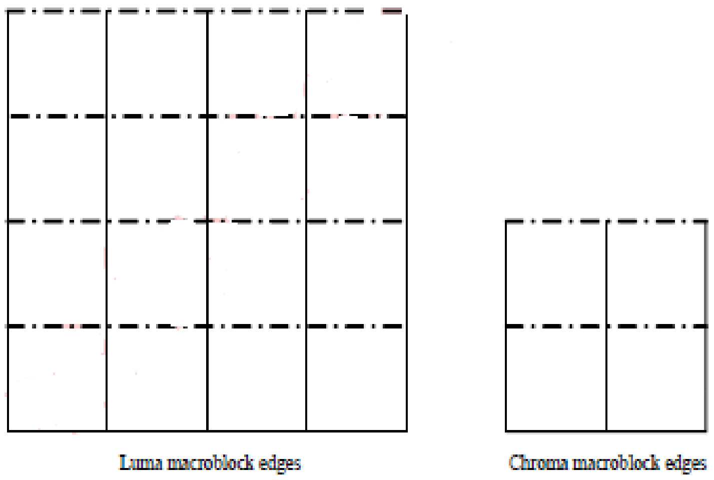


Figure 2.16 Block edges of a macroblock to be filtered [45].

2.6 Summary

This chapter explains various high efficiency tools used in AVS-M, for encoding and decoding of a video sequence. Also it explains the working of AVS-M encoder and decoder using block diagrams, which will help to understand the flow of the entire process in the further chapters.

CHAPTER 3

DATA MINING, DATA MINING ALGORITHMS AND WEKA

3.1 Introduction to data mining [41]

Data mining is the association of an observation to past experience or knowledge in predicting the event without undergoing the complex process of calculating the actual probability. KDD (knowledge discovery in data bases) is an automatic, exploratory analysis and modeling of large data repositories. It is an organized process of identifying valid novel useful and understandable patterns from large and complex data sets. DM (data mining) is the core of KDD process, involving the inferring of algorithms that explore the data, develop the model and discover previously unknown patterns. This is then used for analysis and prediction.

3.1.1 Steps for data mining

The following are the 9 steps of KDD process; they are also shown in Figure 3.1.

1. Developing an understanding of the application domain

   In this step, people should understand and define the goals of the end user and environment in which the knowledge discovery process will take place.

2. Selecting data set

   After understanding the application, important data from the application should be identified, which will be used as input for further steps. Larger the amount of dataset considered, better are the chances of getting good classifier. But collecting large amounts of data in repositories is expensive and the tradeoff between the chances for better classifier and the amount of data to be extracted be known before initializing the process.

3. Preprocessing and cleansing

In this stage, data reliability is enhanced. This stage includes data clearing such as handling missing values and removal of noise or outliers.

4. Data transformation

In this stage, generation of better data for data mining is prepared and developed. Methods here include dimension reduction (such as feature selection and extraction and record sampling) and attribute transformation (such as discretization of numerical attributes and functional transformation). This stage can be crucial for the success of the entire KDD project and it is usually very project-specific.

5. Choosing the appropriate data mining task

This stage decides which type of data mining process to use, e.g. classification, regression or clustering.

6. Choosing the DM algorithm

Having the strategy, this stage decides on the tactics. This stage includes selecting the specific method to be used for searching patterns.

7. Employing the data mining algorithm:

Finally the implementation of the data mining algorithm is reached. This stage needs to employ the algorithms several times until a satisfied result is obtained, for instance by tuning the algorithm's control parameters.

8. Evaluation

This stage evaluates and interprets the mined patterns (rule, reliability etc.) with respect to the goals desired in the first step.

9. Using the discovered knowledge

Now the rules obtained so far can be incorporated in the targeted application system. Now note down the changes in the system and make sure the effect it has are desired ones.

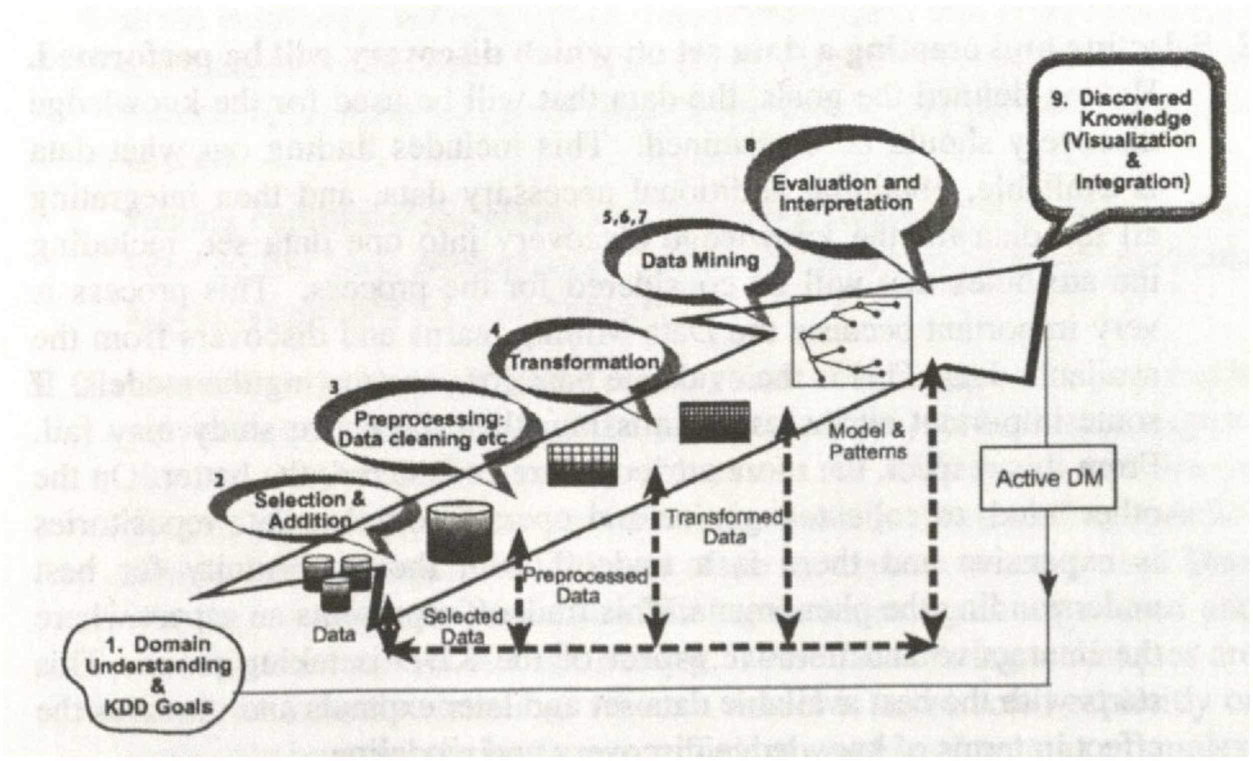

9. Discovered Knowledge (Visualization & Integration)
8
Evaluation and Interpretation
5,6,7
Data Mining
4
Transformation
3
Preprocessing: Data cleaning etc.
2
Selection & Addition
1. Domain Understanding & KDD Goals
Data
Selected Data
Preprocessed Data
Transformed Data
Model & Patterns
Active DM

Figure 3.1 The process of knowledge discovery in data bases. [41]

3.2 Classification of Data mining methods

3.2.1 Classification methods in data mining

Following are the classification methods in data mining:

1. Neural network [41]

Neural networks are computing models for information processing and are particularly useful for indentifying the fundamental relationship among a set of variables or patterns in the data. They grew from research in artificial intelligence; specifically, attempts to mimic thee learning of biological neural networks especially those in human brain which may contain more than billions of highly interconnected neurons. They are very popular due to their powerful modeling capabilities for pattern recognition. Figure 3.2 shows an example of neural network.

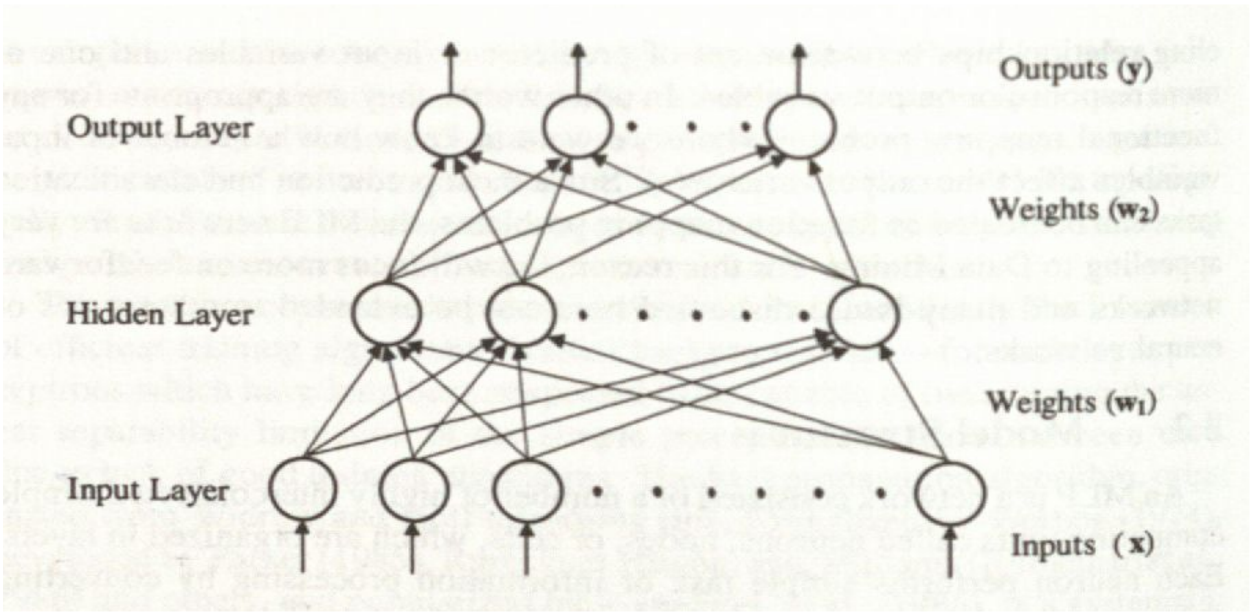
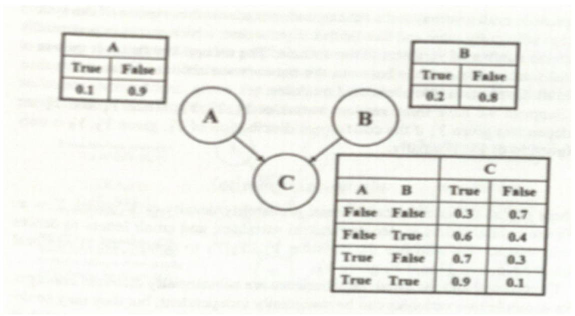


Figure 3.2 Typical neural networks. [41]

2. Bayesian network [41]

They belong to a more general class of models called probabilistic graphical models that arise from the combination of graph theory and probability theory. Their success rests on their ability to handle complex probabilistic models by decomposing them into smaller amenable components.



A	
True	False
0.1	0.9

B	
True	False
0.2	0.8

		C	
A	B	True	False
False	False	0.3	0.7
False	True	0.6	0.4
True	False	0.7	0.3
True	True	0.9	0.1

Figure 3.3 Typical Bayesian network. [41]

3. Support vector machines (SVM)[41]

They are a set of related methods for supervised learning, applicable to both classification and regression problems. The SVM classifiers creates a maximum-margin hyper-plane that lies in a transformed input space and splits the example classes, while maximizing the distance to the nearest cleanly split examples. The parameters of the solution hyper-plane are derived from a quadratic programming optimization problem.

4. Decision trees[41]

A decision tree is a classifier expressed as a recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called “root” that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edge is known as internal node. All others are called leaves. Each internal node splits the instance space into two or more sub-spaces, and finally each leaf is assigned a class.

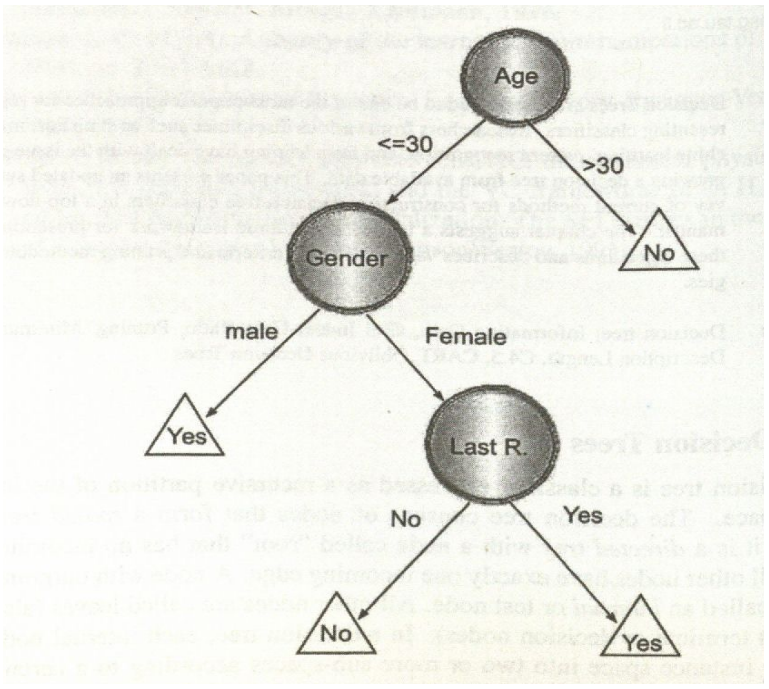


Figure 3.4 Typical tree [41]

5. Instance based learning [36]

   IBL algorithms are derived from the nearest neighbor pattern classifier [33]. They are highly similar to edited nearest neighbor algorithms [30][32], which also save and use only selected instances to generate classification predictions. IBL algorithms differ from most other supervised learning methods: they do not construct explicit abstractions such as decision trees or rules. Most learning algorithms derive generalizations from instances when they are presented and use simple matching procedures to classify subsequently presented instances. This incorporates the purpose of the generalizations at presentation time.

3.2.2 Supervised and unsupervised learning [39]

There are two types of learning methods in data mining:

1. Supervised learning

In this learning method, the pattern recognition system will first undergo a training process. During training process the system will first undergo a training of prototypes, that is, with a set of input patterns along with the category to which each particular pattern belongs. Whenever an error occurs in the system output, an adjustment on the system parameters will follow. Because the system output is fed back to the training algorithm, it is called supervised learning. Figure 3.5 (a) shows the block diagram of supervised learning process.

2. Unsupervised learning

Learning process when there is no prior knowledge of the categories into which the patterns are to be classified, unsupervised learning will play an important role in pattern classification. In unsupervised learning patterns are associated by themselves into clusters based on some properties in common. Figure 3.5(b) shows the block diagram of unsupervised learning process.

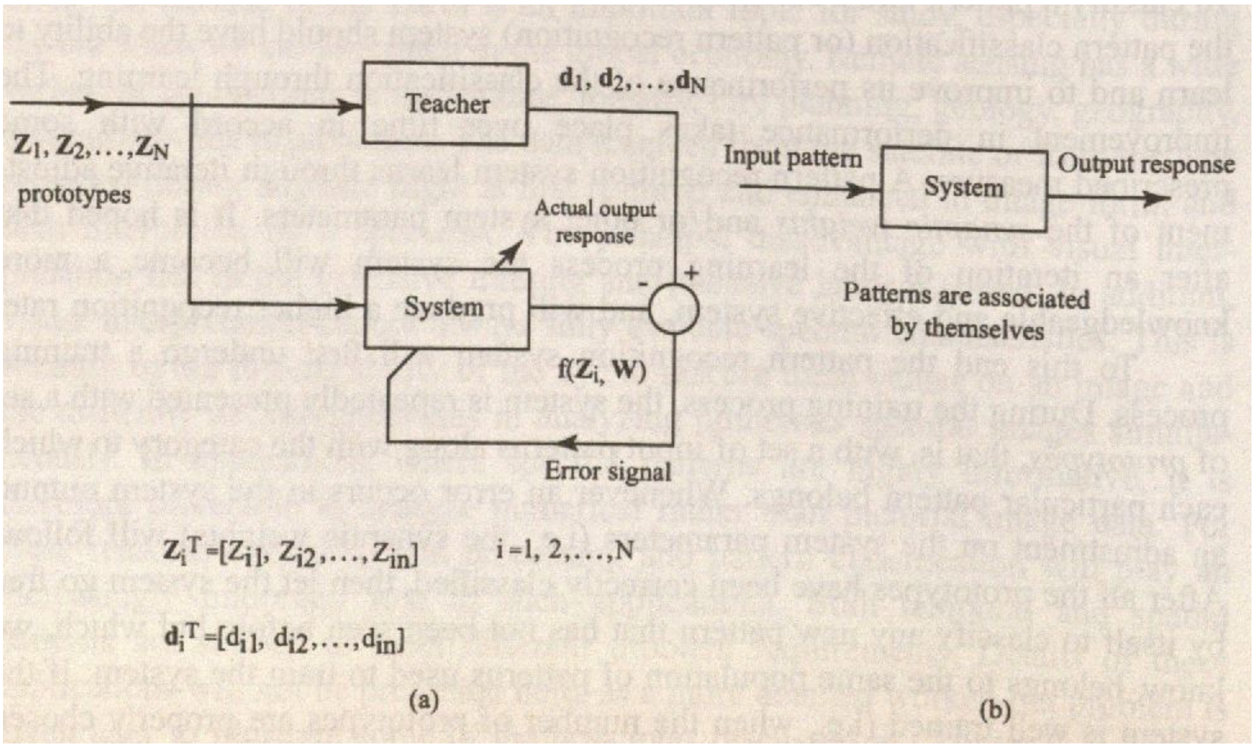


Figure 3.5 (a) Supervised learning process block diagram (b) Unsupervised learning process. [39]

3.3 Machine learning algorithm C4.5 [26]

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.

C4.5 tree induction algorithm shows good classification accuracy and is the fastest among the compared to man-memory algorithms for machine leaning and data mining. [35][28] C4.5 generates classifiers expressed as decision trees, but it can also construct classifiers in a more comprehensible rule set form. 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 has some attributes and a class, attribute being discrete or continuous value, but class being only discrete ranging from $C_1 ... C_N$.

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. The test on a continuous attribute has two possible outcomes, $A <= Th$ and $A > Th$, where Th is a value determined at the node and is call the threshold. Decision tree assigns each case with a class depending on the values of the attributes of the case; the class specified at the leaf is a predicted class. It is compared with the actual class of the case to measure classification error.

3.3.1 Initial stage of C4.5 [35]

Given a set S of cases, C4.5 first grows an initial tree using the divide-and-conquer algorithm as follows: If all the cases in S belong to the same class or S is small, the tree is a leaf labeled with the most frequent class in S otherwise, choose a test based on a single attribute with two or more outcomes. Make this test the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets *S*1, *S*2*, . . .* according to the outcome for each case, and apply the same procedure recursively to each subset.

3.3.2 Tree construction algorithm [28]

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 entire training set S and with all case weights equal to 1. At each node, the following divide and conquer algorithm is executed, trying to exploit the locally best choice, with no back tracking allowed.

Pseudocode of the C4.5 tree-construction algorithm [28]

```
FormTree (T)
    1. ComputeClassFrequency(T);
    2. If OneClass or FewCases return a leaf, create a decision node N;
    3. ForEach Attribute A, ComputeGain(A);
    4. N.test = AttributeWithBestGain;
```

```
5. If N.test is continuous find Threshold;
6. ForEach 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. ComputeErrors 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 (- [1, N_{Class}];

If all cases (Step 2) in T belong to a same class 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 the cases in T whose class in 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. The attribute with highest information gain (step 4) is selected for the test at 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 below threshold. A decision node has 's' children if T_1,……….T_s are the sets of the splitting produced by the test on the selected attribute (step 6). s = 2, if 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 values 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 case 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 errors of the child nodes. If the result is greater than 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.

Information Gain

The information gain of an attribute 'a' for a set of cases T is calculated as follows: if 'a' is discrete, and $T_1, \ldots\ldots T_s$ are the sub-sets of T consisting of cases with distinct known value for attribute 'a', then:

$$\text{Gain} = \text{info (T)} - \sum_{i=1}^{s} \left(\frac{|\text{Ti}|}{|T|} \; X \; info\ (T\text{i}) \right) \qquad (3.1)\ [28]$$

where

$$\text{Info (T)} = - \sum_{j=1}^{NClass} \left(\frac{freq(Cj,T)}{|T|} \; X \; log_2 \quad \frac{\text{freq(Cj,T)}}{|T|} \right) \qquad (3.2)\ [28]$$

is the entropy function. While having an option to select information gain, by default, however C4.5 considers the information gain ratio of splitting $T_1 \ldots\ldots T_s$, which is the ratio of information gain to its split information.

$$\text{Split (T)} = - \sum_{i=1}^{s} \left(\frac{|Ti|}{|T|} \; X \; log_2 \; (\text{Px} \frac{|Ti|}{|T|} \right) \qquad (3.3)\ [28]$$

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 Quick sort ordering algorithm [34]. Assume that the ordered values are $v_1, \ldots\ldots\ldots, v_m$. Consider for *i (- [1,m-1]* the value $v = (v_i+v_{i+1})/2$ and splitting:

$T_1^{v} = \{V_j | V_j <= V\}$ $\quad T_2^{v} = \{V_j | V_j > V\}$

For each value *V*, the information gain $gain_v$ is computed by considering the splitting above, the value v' for which $gain_{v'}$ is maximum is set to be the local threshold and the information gain for the attribute *a* is defined as $gain_{v'}$.

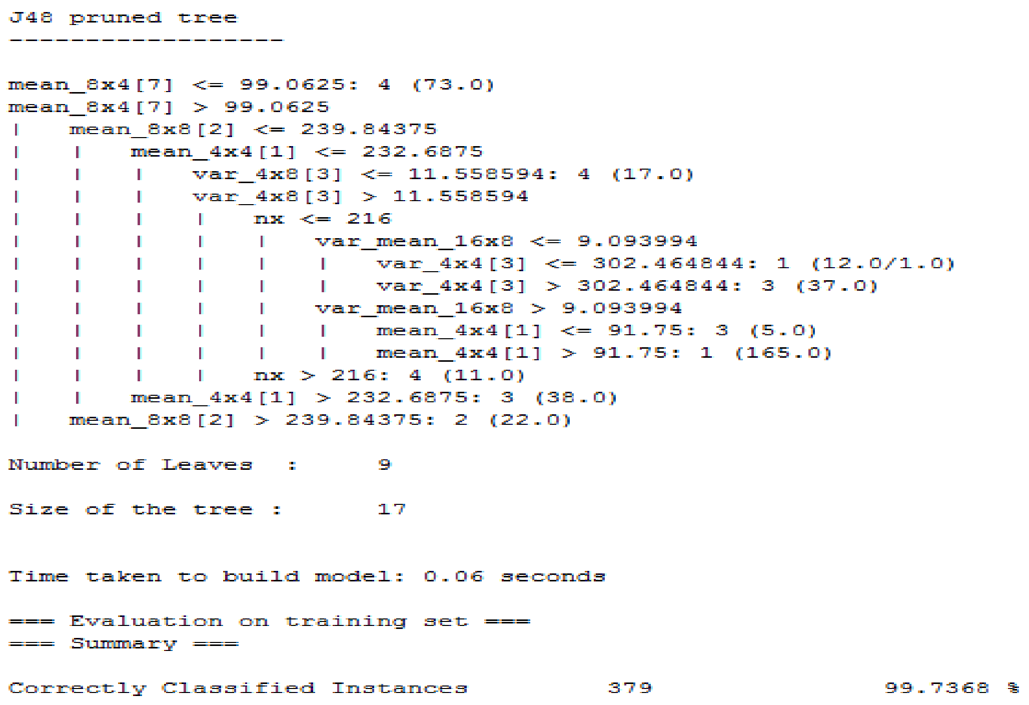
C4.5 considers [42] the information gain ratio[1] (ratio of the information gains by equation 3.1) of the splitting $T^{v}_{1}, T^{v'}_{2}$. Finally, note that, in case the attribute is selected at the node, the threshold is calculated (step -5) by means of a linear search in the whole training set S 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 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 [43], 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 case misclassified by the trees. Also, such an evaluation can be performed on a test set, a set of unseen cases during the tree construction.

3.4 Weka [27]

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from user's Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU general public license. The types of files used by the WEKA data mining program are known as ARFF files. An ARFF file is written in ASCII text and shows the relationship between a set of attributes. These files contain the datasets to be classified. An ARFF file has two different sections: the first section is the header with the information about the name of the relation, the attributes that are used and their types; the second section contains the data with comma division. In the header section, there are attributes declarations. For each MB different metrics are calculated and then stored in the ARFF file.

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 are 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. 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 [27]. 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
------------------

mean_8x4[7] <= 99.0625: 4 (73.0)
mean_8x4[7] > 99.0625
|   mean_8x8[2] <= 239.84375
|   |   mean_4x4[1] <= 232.6875
|   |   |   var_4x8[3] <= 11.558594: 4 (17.0)
|   |   |   var_4x8[3] > 11.558594
|   |   |   |   nx <= 216
|   |   |   |   |   var_mean_16x8 <= 9.093994
|   |   |   |   |   |   var_4x4[3] <= 302.464844: 1 (12.0/1.0)
|   |   |   |   |   |   var_4x4[3] > 302.464844: 3 (37.0)
|   |   |   |   |   var_mean_16x8 > 9.093994
|   |   |   |   |   |   mean_4x4[1] <= 91.75: 3 (5.0)
|   |   |   |   |   |   mean_4x4[1] > 91.75: 1 (165.0)
|   |   |   |   nx > 216: 4 (11.0)
|   |   mean_4x4[1] > 232.6875: 3 (38.0)
|   mean_8x8[2] > 239.84375: 2 (22.0)

Number of Leaves  :     9

Size of the tree :      17


Time taken to build model: 0.06 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         379               99.7368 %
```

Figure 3.6 Classifier tree in terms of statements obtained by Weka.

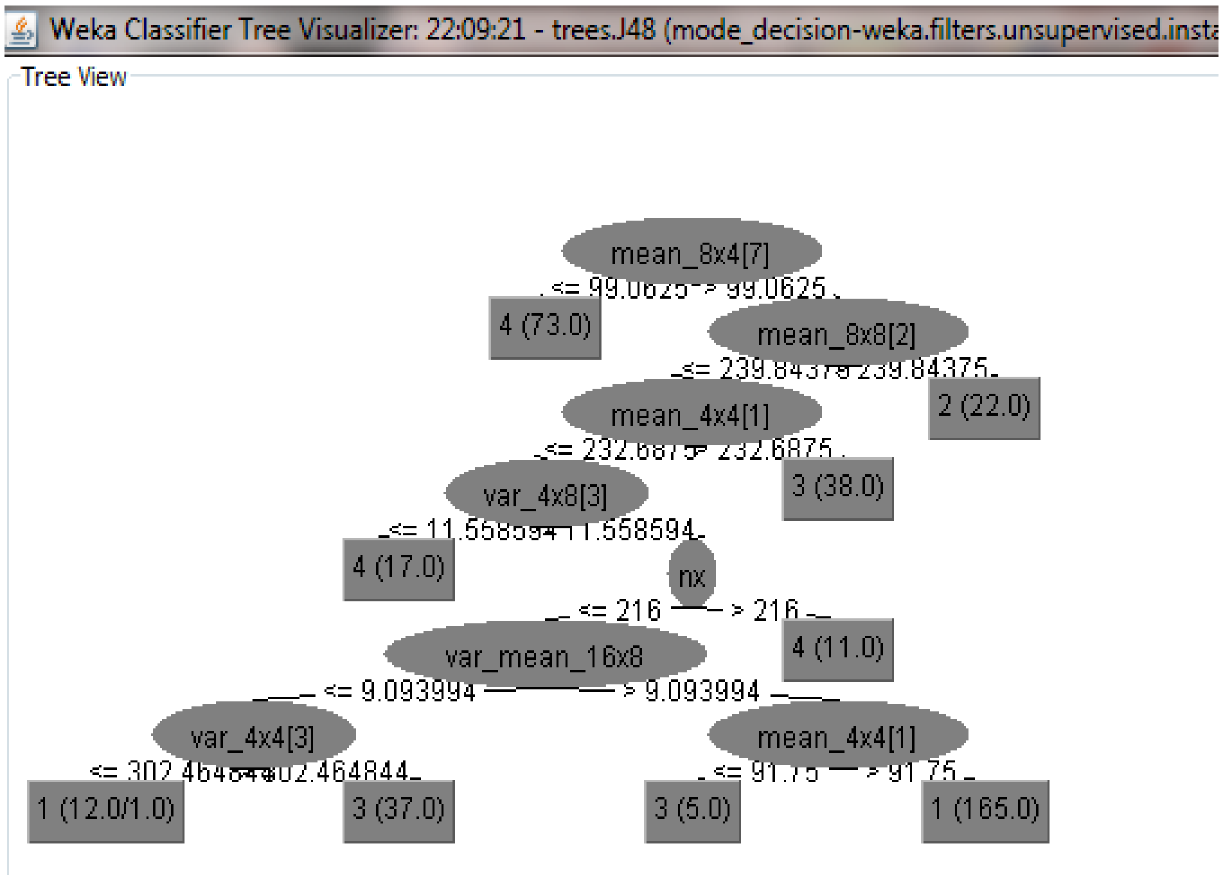


Figure3.7 Tree structure obtained by Weka

3.5 Summary

This chapter explains some concepts of data mining and classification of data mining based on the approach, detailed working (pseudo code) of C4.5 and finally explains Weka tool. This stage of the thesis is the intermediate and final stage of the whole thesis. Attributes used in figure 3.6 are obtained along with many other attributes [appendix B] from the video frames [appendix A], and are written to .arff file. .arff file is then given as input to Weka, which selects key attributes and gives a decision tree (Figure 3.6). The decision tree is then implemented inside the AVS-M encoder.

CHAPTER 4

PROPOSED BLOCK DECISION METHOD

4.1 Introduction

Motion estimation along with block decision accounts for 75% - 80 % of the total encoding time in AVS-M. This thesis aims to replace block decision mode with classifier from data mining algorithm C4.5 [43][28]. The reference software used for AVS-M is obtained from avs.org [8].

Actual AVS-M exploits temporal and spatial redundancies to reduce the number bits required to code the video sequence, but the highly efficient coding tools like block prediction and motion estimation come with lot of encoding complexity. This makes it almost impossible to broadcast a video sequence in real time without taking help of specialized hardware solutions for the application. This thesis replaces block motion estimation mode calculated by sum of absolute differences (SAD) or rate distortion optimization (RDO) present in actual AVS-M with if-else statements obtained from machine learning algorithm C4.5. Figure 4.1 explains block diagram of the proposed encoder.

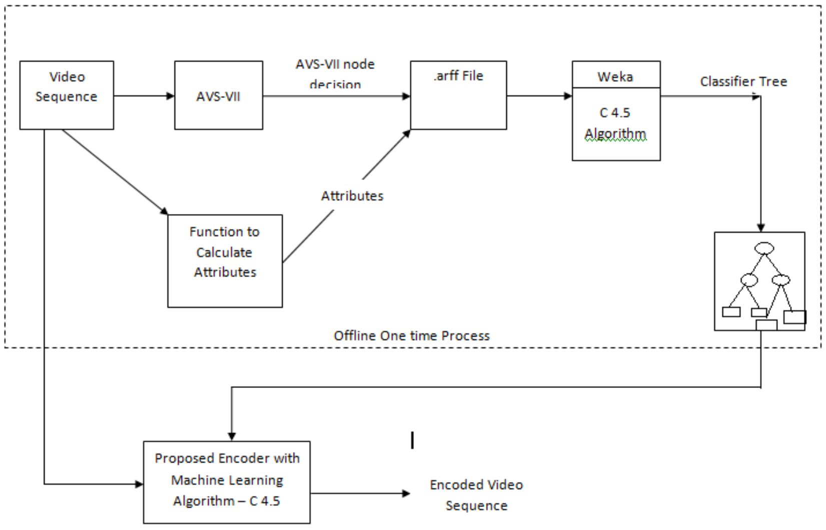


Figure 4.1 Block diagram of the proposed encoder.

4.2 Approach

Various attributes such as mean, variance, mean of variance, variance of mean, edge of all possible block sizes (4x4,4x8,8x4,8x8,8x16,16x8,16x16) are extracted for each macro-block of 4 frames of all sequences along with actual mode decision taken by AVS-M are written to a .arff (Attribute relation file format) file.

C4.5 implemented in Weka [27] is used as the classifier algorithm to get the decision tree.

An effort is made initially to use various sequences as trainer to obtain the decision tree and then use the same tree on different test sequence for testing, results of which are tabulated in Table 4.1. Finally a tree is developed from the frames of different sequences which can be embedded in the encoder efficiently for all sequences. Though there are 7 block modes used in AVS-M, this thesis exploited only the classifiers with macro block mode types – 16x16, 16x8, 8x16, 8x8 leaving sub-macro block mode in their regular processing because of fitting problem. Different video sequences have different numbers of macroblocks in particular mode type depending on the amount of information present in each macro block and its redundancy with regards to spatial and temporal neighbors. Because of this vast variations Weka does not give reliable tree for all the 7 block types. So what is done here is that, in initial stage macroblock type ranging from 16x16 to 8x8 are decided with if-else statement from their attributes and is coded, but if a macroblock is decided to be coded with 8x8 block type, then it further undergoes whole RDO process to decide with which of the sub-block type ranging from 8x8 to 4x4 and skip-mode, it will be coded. The entire process is shown in the block diagram of the proposed encoder, Figure 4.1.

The idea of applying machine learning algorithm C4.5 was taken from [2]. But in that thesis, a unique tree is developed for each and every sequence to be encoded, and that tree is embedded only for that sequence. This approach cannot be applied in real world, because then every encoder will need, embodiment of the C4.5 and the mode decision part will have to be changed for encoding each and every sequence. Though it will give highly efficient results in terms for PSNR and bit savings, it is not viable. Though this thesis has tried to establish the fact that mode decisions can be correctly predicted with accuracy as good as 95% (Table 4.1), with the C4.5 trees, but then they are good for that sequence only and some time even not viable to embody them in to the encoder because of their size and though gives reasonable accuracy of 80%, for some other sequences depicted as test sequence in (Table 4.1). Here a tree with moderate accuracy of 60%-70% is implemented, because it is moderate for all the sequences tested and also, is viable to implement because of its size.

4.3 Experimental results

Table 4.1 shows the % accuracy in prediction the block mode decision by Weka too. The accuracy is measured with respect to the actual decision taken by AVS-M while running the whole RDO process. The if-else statements obtained from this tool are then embodied in the AVS-M replacing the block mode decision process. The metrics such as PSNR, encoding time and number of bits used to encode the file are compared for both encoders as shown in Tables 4.2 through 4.7.

Table 4.1 % accuracy in mode prediction by C4.5 embodied in Weka.

Sequence no:	Training sequence	% accuracy for training sequence	Test sequence	% accuracy for test sequence *
1.	Foreman_qcif	95.78	Highway_qcif	94.04
2.	Coastguard_qcif	95.074	Carphone_qcif	84.172
3.	Akiyo_qcif	90.625	Container	83.186
4.	Highway_qcif	96.90	Bridge-close_qcif	87.234
5.	News_qcif	98.05	Akiyo_qcif	77.718
6.	Miss-america_qcif	95.693	News_qcif	88.429
7.	Container_qcif	96.95	Miss-america_qcif	87.288
8.	Carphone_qcif	93.99	Coastguard_qcif	80.108
9.	Bridge-clsose_qcif	95.69	Foreman_qcif	84.211

(* - % accuracy is obtained by calculating the number of modes predicted same as that predicted by AVS-M and converting it into %)

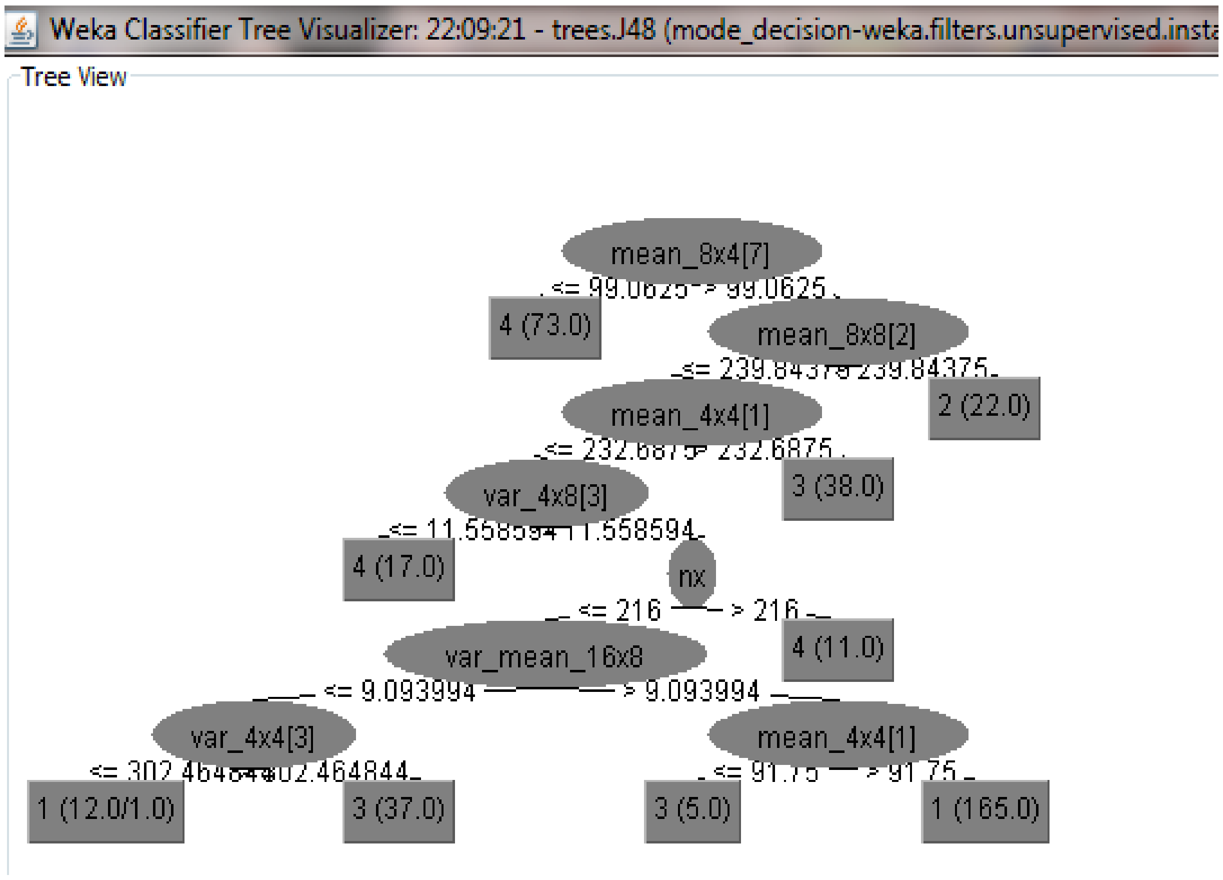
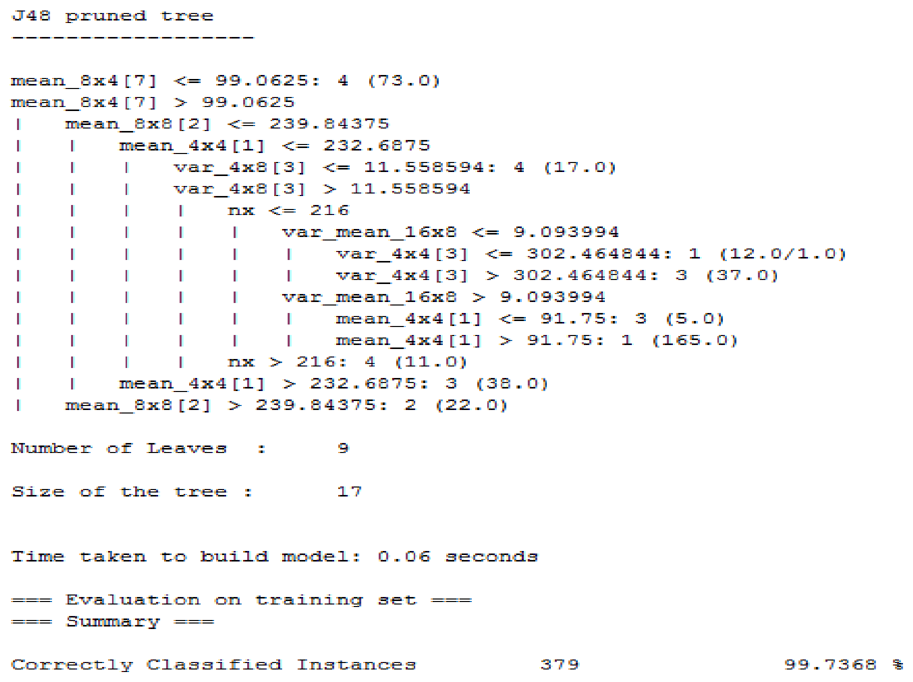


Figure 4.2 Decision tree obtained from Weka tool.

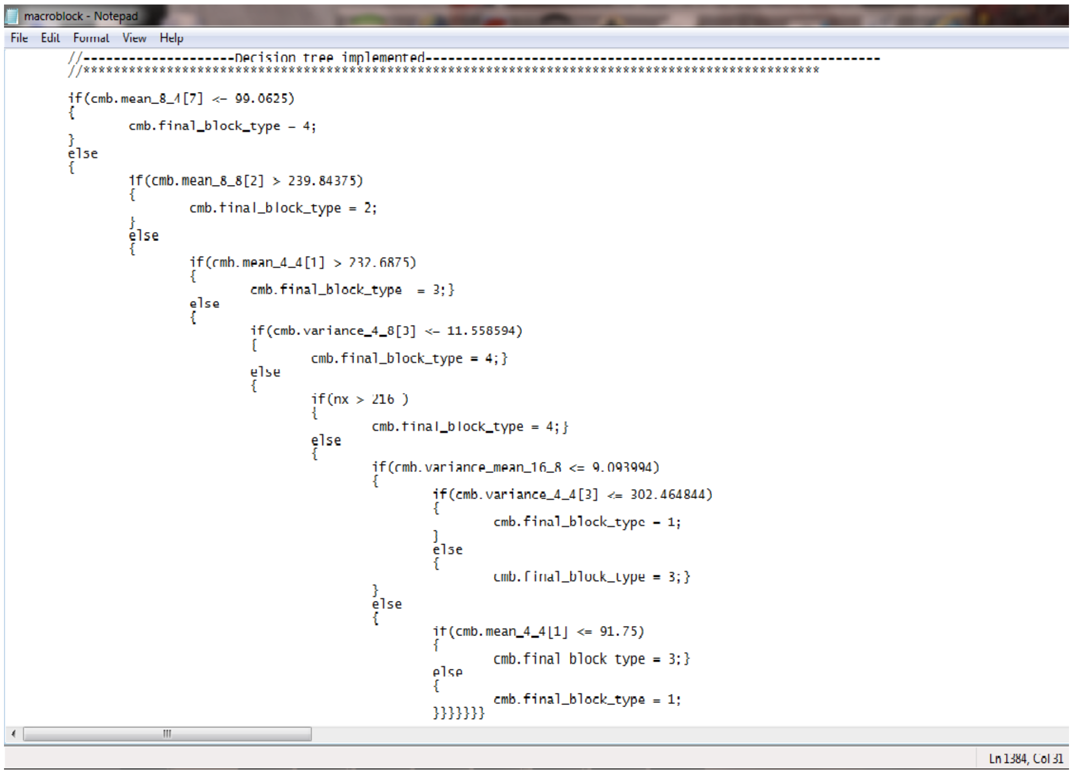
The tree structure in Figure 4.2 can be implemented in terms of if-else statement in the block-estimation section of the AVS-M video encoder. Each ovular structure is called node of the tree: it represents the attribute value extracted from the current macro-block of the frame. Each line joining the nodes is called the branch: the value on the branch tells the decision maker, depending on the current value of that attribute, which direction the decision process will progress. Finally the rectangular blocks are called the leaves: they are the final mode decision taken by the classifier after passing through their corresponding root, branches and nodes. The Weka gives two different views of the same tree, one is shown as tree structure as in Figure 4.2 and the other one is in terms of statement and its prediction accuracy as shown in Figure 4.3



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

mean_8x4[7] <= 99.0625: 4 (73.0)
mean_8x4[7] > 99.0625
|   mean_8x8[2] <= 239.84375
|   |   mean_4x4[1] <= 232.6875
|   |   |   var_4x8[3] <= 11.558594: 4 (17.0)
|   |   |   var_4x8[3] > 11.558594
|   |   |   |   nx <= 216
|   |   |   |   |   var_mean_16x8 <= 9.093994
|   |   |   |   |   |   var_4x4[3] <= 302.464844: 1 (12.0/1.0)
|   |   |   |   |   |   var_4x4[3] > 302.464844: 3 (37.0)
|   |   |   |   |   var_mean_16x8 > 9.093994
|   |   |   |   |   |   mean_4x4[1] <= 91.75: 3 (5.0)
|   |   |   |   |   |   mean_4x4[1] > 91.75: 1 (165.0)
|   |   |   |   nx > 216: 4 (11.0)
|   |   mean_4x4[1] > 232.6875: 3 (38.0)
|   mean_8x8[2] > 239.84375: 2 (22.0)

Number of Leaves  :     9

Size of the tree :      17


Time taken to build model: 0.06 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         379               99.7368 %
```

Figure 4.3 The classifier tree in terms of statements.

Tree obtained in Figure 4.2 or figure 4.3 is implemented in terms of if-else statement inside the AVS-M code replacing the RDO procedure block selection mode. The if-else statements in AVS-M are shown in Figure 4.4. After implementing the if-else statements inside the encoder, the thesis tried to study the effect of some evaluation parameters of sequence and performance parameter of the encoder. Table 4.2 shows the reduction in encoding time, which is the main aim of this thesis.



```
//-------------------Decision tree implemented------------------------------------------------------------
//***********************************************************************************************

if(cmb.mean_8_4[7] <- 99.0625)
{
        cmb.final_block_type - 4;
}
else
{
        if(cmb.mean_8_8[2] > 239.84375)
        {
                cmb.final_block_type = 2;
        }
        else
        {
                if(cmb.mean_4_4[1] > 232.6875)
                {
                        cmb.final_block_type  = 3;}
                else
                {
                        if(cmb.variance_4_8[3] <- 11.558594)
                        {
                                cmb.final_block_type = 4;}
                        else
                        {
                                if(nx > 216 )
                                {
                                        cmb.final_block_type = 4;}
                                else
                                {
                                        if(cmb.variance_mean_16_8 <= 9.093994)
                                        {
                                                if(cmb.variance_4_4[3] <= 302.464844)
                                                {
                                                        cmb.final_block_type - 1;
                                                }
                                                else
                                                {
                                                        cmb.final_block_type = 3;}
                                        }
                                        else
                                        {
                                                if(cmb.mean_4_4[1] <= 91.75)
                                                {
                                                        cmb.final block type = 3;}
                                                else
                                                {
                                                        cmb.final_block_type = 1;
                                                }}}}}}}
```

Figure 4.4 Implementing the tree in AVS-M.

Table 4.2 Comparison of the encoding time between AVS-M and the proposed encoder

Sequence no:	Sequence	Encoding time of the sequence with original AVS-M (seconds)	Encoding time of the sequence with machine learning algorithm implemented (seconds)	% reduction in encoding time of the sequence **
1.	Akiyo_qcif	1.497	0.390	73.94
2.	Highway_qcif	1.934	0.390	79.83
3.	Coastguard_qcif	2.699	0.390	85.56
4.	Bridge-close	2.075	0.359	82.70

Table 4.2 - *continued*

5.	News_qcif	1.732	0.374	78.41
6.	Miss-america_qcif	1.747	0.452	74.13
7.	Container_qcif	1.903	0.405	78.72
8.	Carphone_qcif	1.872	0.405	78.36
9.	Foreman_qcif	1.997	0.405	79.71

(** % reduction in time = $\frac{\text{(encoing time AVS−VII)− (encoding time proposed encoder)}}{\text{(encoing time AVS−VII)}}$ X 100 %)

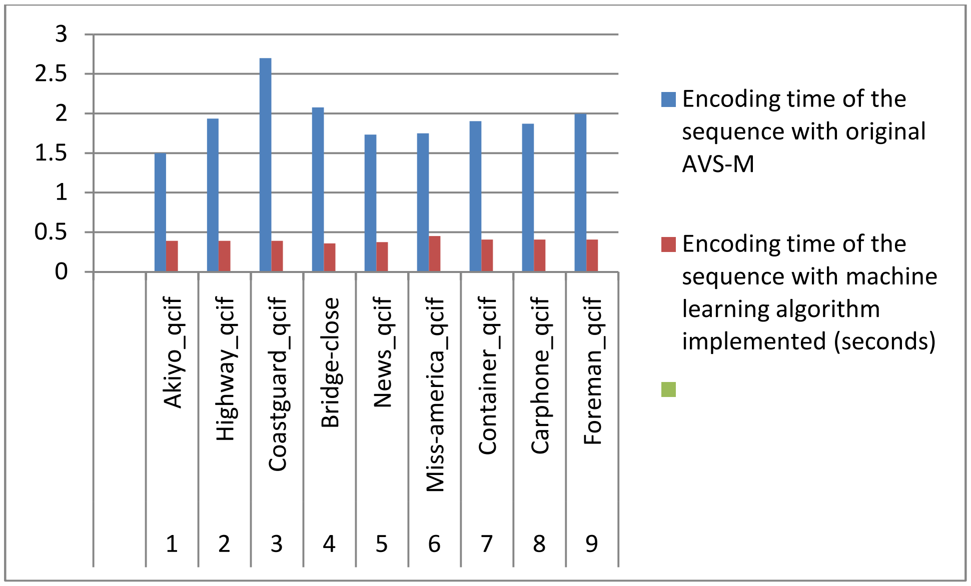


Figure 4.5 Bar chart to compare encoding time of AVS-M and the proposed encoder.

Table 4.3 shows the comparison of PSNR of luma component of the video sequences, when encoded by AVS-M and the proposed encoder. Unexpected drop in PSNR for carphone_qcif and foreman_qcif sequences is observed.

Table 4.3 Comparison of the Y-component when encoded by AVS-M and the proposed encoder.

Sequence number	Sequence	PSNR of the video sequence encoded with AVS-M (Y-component in dB)	PSNR of the video sequence encoded with AVS-M, with machine learning algorithm (Y-component in dB)	% *** reduction in PSNR of Y-component while encoding with proposed encoder
1.	Akiyo_qcif	37.316987	37.176920	0.375
2.	Highway_qcif	37.622730	37.116283	1.34
3.	Coastguard_qcif	34.206012	33.834510	1.08
4.	Bridge-close	34.361937	33.416094	2.70
5.	News_qcif	35.964311	35.575041	1.08
6.	Miss-america_qcif	39.417091	38.895065	1.32
7.	Container_qcif	35.798379	35.076394	2.01
8.	Carphone_qcif	36.517526	34.486256	5.56
9.	Foreman_qcif	35.977242	34.749983	3.41

(*** % reduction in PSNR(Y) = $\frac{(\mathrm{PSNR(Y,AVS\_VII)}) - (\mathrm{PSNR(Y,Proposed\ encoder)})}{(\mathrm{PSNR(Y,AVS\_VII)})}$ X 100 %)

Figure 4.6 represents the Y-component of the video sequences in both encoder and compares them.

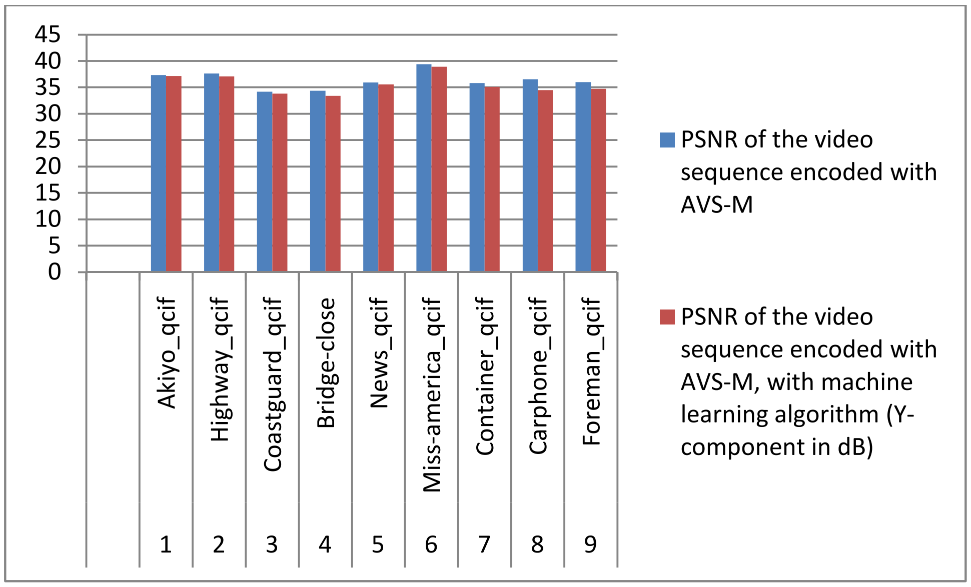


Figure 4.6 Bar chart to compare PSNR of luma component when encoded by AVS-M and the proposed encoder

Table 4.4 Comparison of the U-component when encoded by AVS-M and proposed the encoder.

Sequence number	Sequence	PSNR of the video sequence encoded with AVS-M (U-component in dB)	PSNR of the video sequence encoded with AVS-M, with machine learning algorithm (U-component in dB)	%**** reduction in PSNR of U-component while encoding with proposed encoder
1.	Akiyo_qcif	39.285791	39.180394	0.268
2.	Highway_qcif	37.678196	37.656085	0.059
3.	Coastguard_qcif	43.270733	43.169228	0.023
4.	Bridge-close	36.929451	36.741130	0.050
5.	News_qcif	38.555165	38.355257	0.052
6.	Miss-america_qcif	38.947323	38.895372	0.013

Table 4.4 - *continued*

7.	Container_qcif	39.586359	39.325356	0.065
8.	Carphone_qcif	39.951664	39.775975	0.044
9.	Foreman_qcif	40.318186	40.114908	0.050

(**** % reduction in PSNR (U) = $\frac{(\text{PSNR(U,AVS\_VII)}) - (\text{PSNR(U,Proposed encoder)})}{(\text{PSNR(U,AVS\_VII)})}$ X 100 %)

Table 4.4 compares the U component of the video sequences when encoded with AVS-M and with proposed encoder.

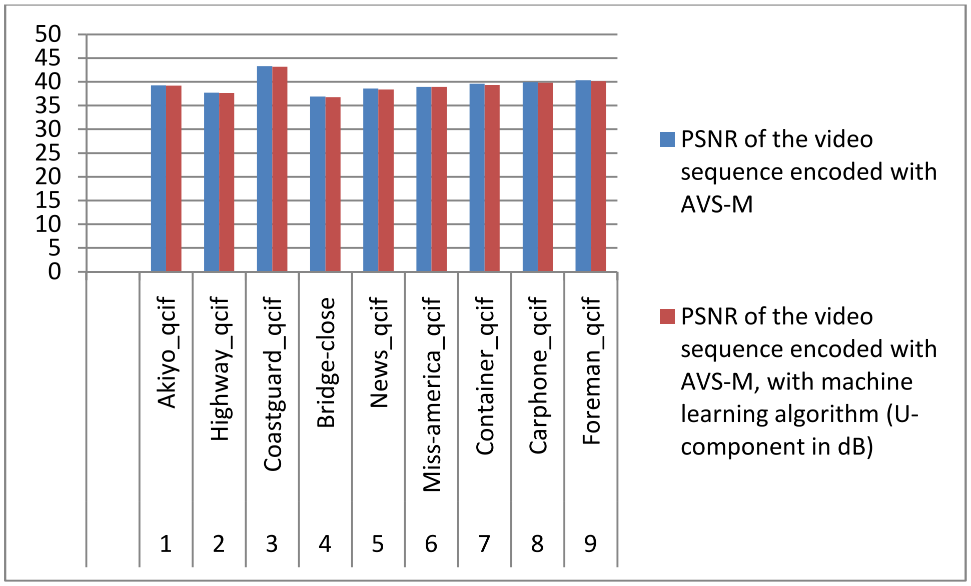


Figure 4.7 Bar chart to compare PSNR of U component when encoded by AVS-M and the proposed encoder.

Table-4.5 compares the PSNR of V component of video sequence when encoded with AVS-M and encoded with proposed encoder.

Table 4.5 Comparison of the V-component when encoded by AVS-M and the proposed encoder.

Sequence number	Sequence	PSNR of the video sequence encoded with AVS-M (V-component in dB)	PSNR of the video sequence encoded with AVS-M, with machine learning algorithm (V-component in dB)	%***** reduction in PSNR of V-component while encoding with proposed encoder
1.	Akiyo_qcif	40.506783	40.452600	0.013
2.	Highway_qcif	38.493726	38.502246	-0.002
3.	Coastguard_qcif	44.820145	44.577757	0.054
4.	Bridge-close	37.376204	37.257109	0.031
5.	News_qcif	39.619799	39.583642	0.009
6.	Miss-america_qcif	38.929404	38.647452	0.072
7.	Container_qcif	39.617169	39.488338	0.033
8.	Carphone_qcif	40.275814	40.190305	0.021
9.	Foreman_qcif	40.925833	40.786655	0.034

(***** % reduction in PSNR (V) = $\frac{(\mathrm{PSNR(V,AVS\_VII)}) - (\mathrm{PSNR(V,Proposed\ encoder)})}{(\mathrm{PSNR(V,AVS\_VII)})}$ X 100 %)

Figure 4.8 shows the bar-chart to compare the PSNR of V-component of video sequence, encoded with AVS-M and encoded with proposed encoder. The study observes that there is almost no loss in PSNR in V component. There was unexpected increase in PSNR of V-component for highway_qcif.

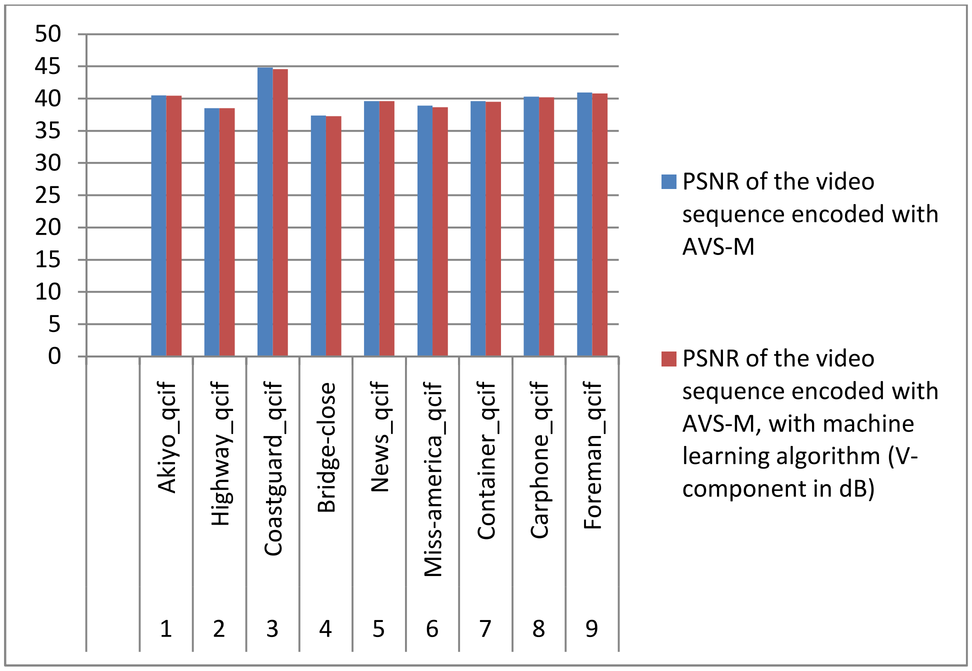


Figure 4.8 Bar chart to compare PSNR of V component when encoded by AVS-M and the proposed encoder.

Table 4.6 Comparison of YUV-component when encoded by AVS-M and the proposed encoder.

Sequence number	Sequence	PSNR of the video sequence encoded with AVS-M (YUV-component in dB)	PSNR of the video sequence encoded with AVS-M, with machine learning algorithm (YUV-component in dB)	% reduction in PSNR of YUV-component while encoding with proposed encoder
1.	Akiyo_qcif	38.006087	37.878926	0.033
2.	Highway_qcif	37.732928	37.390016	0.090
3.	Coastguard_qcif	35.741460	35.380718	1.009
4.	Bridge-close	35.074161	34.300780	2.204

Table 4.6 - *continued*

5.	News_qcif	36.766114	36.429744	0.091
6.	Miss-america_qcif	39.244104	38.850075	1.004
7.	Container_qcif	36.732660	36.100675	1.720
8.	Carphone_qcif	37.412563	35.663827	4.742
9.	Foreman_qcif	37.044786	35.856083	3.208

(***** % reduction in PSNR (YUV) = $\frac{(\text{PSNR(YUV,AVS\_VII)}) - (\text{PSNR(YUV,Proposed encoder)})}{(\text{PSNR(YUV,AVS\_VII)})}$ X 100 %)

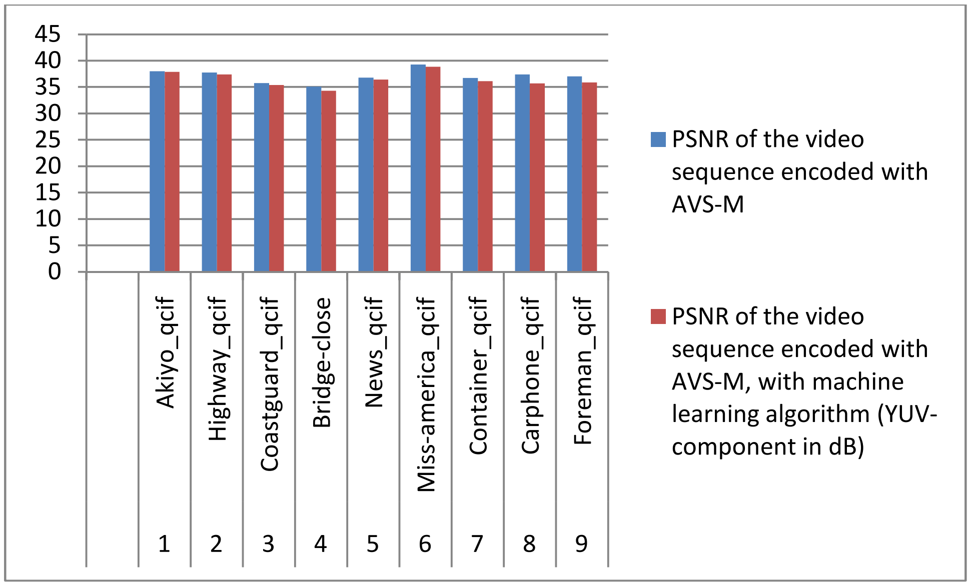


Figure 4.9 Bar chart to compare PSNR of YUV component when encoded by AVS-M and the proposed encoder.

PSNR (YUV) is calculated from PSNR(Y), PSNR (U) and PSNR (V) by equations 4.1 through 4.4. [8]

tmpy = (4.0 / pow(10, PSNR(Y) / 10)) (4.1)

tmpu = (1.0 / pow(10, PSNR(U) / 10)) (4.2)

tmpv = (1.0 / pow(10, PSNR(V)/ 10)) (4.3)

PSNR(YUV) = (10 * log10(6 / (tmpy + tmpu + tmpv))) (4.4)

Table 4.7 Comparison of number of bits used by AVS-M and the proposed encoder.

Sequence number	Sequence	No. of bits used to encode the sequence encoding with AVS-M	No. of bits used to encode the sequence encoding with the proposed encoder.	%****** increase in bits while encoding with proposed encoder
1.	Akiyo_qcif	27283	26867	-1.524
2.	Highway_qcif	27346	27098	-0.090
3.	Coastguard_qcif	55790	54702	-1.950
4.	Bridge-close	49424	44405	-10.154
5.	News_qcif	42013	41877	-0.324
6.	Miss-america_qcif	20689	20385	-1.469
7.	Container_qcif	37725	36061	-4.410
8.	Carphone_qcif	47101	45925	-2.496
9.	Foreman_qcif	47029	46557	-1.003

(****** % reduction in bits = $\frac{(\text{no.of bits(AVS}-\text{M))}- \ (\text{no.of bits(Proposed encoder))}}{(\text{no.of bits(AVS\_VII))}}$ X 100 %)

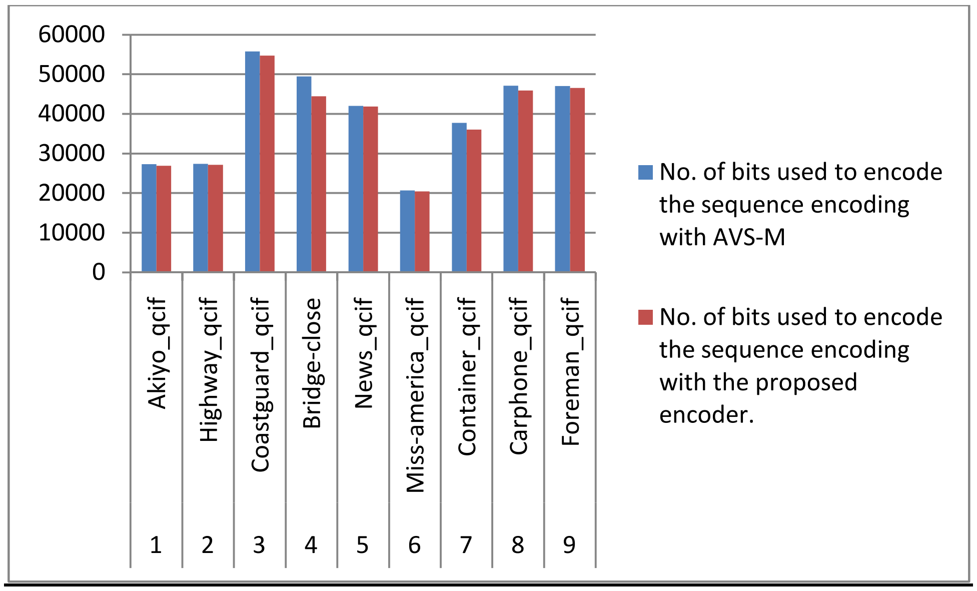


Figure 4.10 Bar chart to compare number of bits used by AVS-M and the proposed encoder.

Table 4.8 shows the performance of proposed encoder for 100 frames. Even though considerable loss in PSNR(Y) is observed, there is almost the same saving in encoding time. One reason behind the significant loss in %dB in PSNR (Y) may be that the tree that is trained for 4 frames only and we are testing it on 100 frames.

Table 4.8 Comparison of encoding performances by AVS-M and the proposed encoder

Sequence	Akiyo_qcif	Container_qcif	Bridge-close_qcif	Miss-america_qcif	Foreman_qcif
PSNR(Y) dB	38.876830	36.852617	35.705636	41.028109	37.394567
PSNR(Y') dB	36.958418	33.763365	33.269825	38.091754	35.005621
%decrease in PSNR	4.92	7.95	6.82	7.15	6.38
PSNR(U) dB	40.388664	40.580305	37.145837	39.639008	41.00515186
PSNR(U') dB	40.167291	40.127251	36.935639	39.508902	40.930060

Table 4.8 - *continued*

%decrease in PSNR	0.05	1.11	0.056	0.032	0.018
PSNR(V) dB	41.280113	40.484948	37.877927	40.017777	42.023883
PSNRV') dB	41.207557	40.255924	37.772218	39.779168	41.934222
%decrease in PSNR	0.02	0.056	0.027	0.059	0.021
PSNR(YUV) dB	39.430119	37.769947	36.222158	40.586585	38.380581
PSNR(YUV') dB	37.854104	35.0472241	34.251494	38.481546	36.272297
%decrease in PSNR	3.99	7.2	5.44	5.18	5.49
No. of bits used(B)	204723	252364	549384	230396	766770
No. of bits used(B')	209955	253908	398959	227076	841282
% saving in bits	2.5	0.06	-27.38	-1.44	9.71
Encoding time(t) (t)	49.873	66.143	65.692	54.615	72.634
Encoding time (t')	10.186	14.352	11.730	14.071	13.915
%decrease in t	79.57	78.30	82.14	74.23	80.84

(Y, U, V, YUV, B, t components of sequences when encoder with AVS-M and Y', U', V', YUV', B', t' are the components of video sequences when encoded with the proposed encoder)

4.4 Conclusions and Future work

By observing quality matrices from Table 4.2 through Table 4.8, it is concluded that a highly time efficient AVSM codec can be obtained by implementing machine learning algorithm C4.5 in place of mode-decision block of AVS-M. That has resulted in, an average 75% reduction in the encoding time of the AVS-M, also saving the number of bits required to encode the sequence with negligible loss of quality in terms of PSNR (Y, U, V, YUV components).

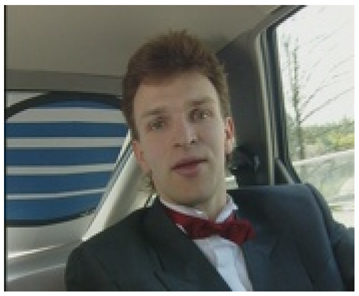
This thesis considered a tree with moderate accuracy in predicting the block mode decision, to be implemented in the encoder, so that a unique tree can be applied for all the sequences to be tested. Two sequences carphone_qcif and foreman_qcif show a loss in PSNR value for Y component which is a bit more than expected. Intense study in video data statistics and identifying the key attributes, and also preprocessing these attributes before getting the tree may help improving the tree efficiency. There are large numbers of encoding modes for a macroblock, but only inter-predicted macroblock and that too only macroblock modes ranging from 16x16 to 8x8 are considered, leaving 8x8 to 4x4 and skip mode decision in its original decision making process, to minimize the overfitting problem that would occur in trees leading to more decision errors. So intense study can be under taken in the field of statistics and pattern recognition to improve the decision tree, and also for embodying all decision modes in terms of if-else statements. This effort can decrease the encoding time exponentially without affecting the video quality too much.

APPENDIX A

SNAPSHOTS OF THE VIDEO SEQUENCES USED

Sequence 1: akiyo_qcif.

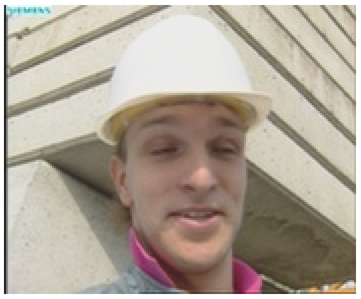
Sequence 2: bridge-close_qcif.



Sequence 3: carphone_qcif

Sequence 4: coastguard_qcif

Sequence 5: container_qcif



Sequence 6: foreman_qcif

Sequence 7: highway_qcif

Sequence 8: miss-america_qcif

Sequence 9: news_qcif.

APPENDIX B

ATTRIBUTES EXTRACTED FROM THE VIDEO FRAMES

```
@attribute mean_4x4[0] numeric
@attribute mean_4x4[1] numeric
@attribute mean_4x4[2] numeric
@attribute mean_4x4[3] numeric
@attribute mean_4x4[4] numeric
@attribute mean_4x4[5] numeric
@attribute mean_4x4[6] numeric
@attribute mean_4x4[7] numeric
@attribute mean_4x4[8] numeric
@attribute mean_4x4[9] numeric
@attribute mean_4x4[10] numeric
@attribute mean_4x4[11] numeric
@attribute mean_4x4[12] numeric
@attribute mean_4x4[13] numeric
@attribute mean_4x4[14] numeric
@attribute mean_4x4[15] numeric
@attribute var_4x4[0] numeric
@attribute var_4x4[1] numeric
@attribute var_4x4[2] numeric
@attribute var_4x4[3] numeric
@attribute var_4x4[4] numeric
@attribute var_4x4[5] numeric
@attribute var_4x4[6] numeric
@attribute var_4x4[7] numeric
@attribute var_4x4[8] numeric
@attribute var_4x4[9] numeric
@attribute var_4x4[10] numeric
```

```
@attribute var_4x4[11] numeric

@attribute var_4x4[12] numeric

@attribute var_4x4[13] numeric

@attribute var_4x4[14] numeric

@attribute var_4x4[15] numeric

@attribute mean_4x8[0] numeric

@attribute mean_4x8[1] numeric

@attribute mean_4x8[2] numeric

@attribute mean_4x8[3] numeric

@attribute mean_4x8[4] numeric

@attribute mean_4x8[5] numeric

@attribute mean_4x8[6] numeric

@attribute mean_4x8[7] numeric

@attribute mean_8x4[0] numeric

@attribute mean_8x4[1] numeric

@attribute mean_8x4[2] numeric

@attribute mean_8x4[3] numeric

@attribute mean_8x4[4] numeric

@attribute mean_8x4[5] numeric

@attribute mean_8x4[6] numeric

@attribute mean_8x4[7] numeric

@attribute var_4x8[0] numeric

@attribute var_4x8[1] numeric

@attribute var_4x8[2] numeric

@attribute var_4x8[3] numeric

@attribute var_4x8[4] numeric

@attribute var_4x8[5] numeric
```

```
@attribute var_4x8[6] numeric

@attribute var_4x8[7] numeric

@attribute var_8x4[0] numeric

@attribute var_8x4[1] numeric

@attribute var_8x4[2] numeric

@attribute var_8x4[3] numeric

@attribute var_8x4[4] numeric

@attribute var_8x4[5] numeric

@attribute var_8x4[6] numeric

@attribute var_8x4[7] numeric

@attribute mean_8x8[0] numeric

@attribute mean_8x8[1] numeric

@attribute mean_8x8[2] numeric

@attribute mean_8x8[3] numeric

@attribute var_8x8[0] numeric

@attribute var_8x8[1] numeric

@attribute var_8x8[2] numeric

@attribute var_8x8[3] numeric

@attribute mean_8x16[0] numeric

@attribute mean_8x16[1] numeric

@attribute mean_16x8[0] numeric

@attribute mean_16x8[1] numeric

@attribute var_8x16[0] numeric

@attribute var_8x16[1] numeric

@attribute var_16x8[0] numeric

@attribute var_16x8[1] numeric

@attribute mean_16x16 numeric
```

```
@attribute mean_var_4x4 numeric

@attribute var_mean_4x4 numeric

@attribute mean_var_4x8 numeric

@attribute var_mean_4x8 numeric

@attribute mean_var_8x4 numeric

@attribute var_mean_8x4 numeric

@attribute mean_var_8x8 numeric

@attribute var_mean_8x8 numeric

@attribute mean_var_8x16 numeric

@attribute var_mean_8x16 numeric

@attribute mean_var_16x8 numeric

@attribute var_mean_16x8 numeric

@attribute edge_4[0] numeric

@attribute edge_4[1] numeric

@attribute edge_4[2] numeric

@attribute edge_4[3] numeric

@attribute edge_4[4] numeric

@attribute edge_4[5] numeric

@attribute edge_4[6] numeric

@attribute edge_4[7] numeric

@attribute edge_4[8] numeric

@attribute edge_4[9] numeric

@attribute edge_4[10] numeric

@attribute edge_4[11] numeric

@attribute edge_4[12] numeric

@attribute edge_4[13] numeric

@attribute edge_4[14] numeric
```

```
@attribute edge_4[15] numeric

@attribute edge_4[16] numeric

@attribute edge_4[17] numeric

@attribute edge_4[18] numeric

@attribute edge_4[19] numeric

@attribute edge_4[20] numeric

@attribute edge_4[21] numeric

@attribute edge_4[22] numeric

@attribute edge_4[23] numeric

@attribute edge_48[0] numeric

@attribute edge_48[1] numeric

@attribute edge_48[2] numeric

@attribute edge_48[3] numeric

@attribute edge_48[4] numeric

@attribute edge_48[5] numeric

@attribute edge_48[6] numeric

@attribute edge_48[7] numeric

@attribute edge_48[8] numeric

@attribute edge_48[9] numeric

@attribute edge_84[0] numeric

@attribute edge_84[1] numeric

@attribute edge_84[2] numeric

@attribute edge_84[3] numeric

@attribute edge_84[4] numeric

@attribute edge_84[5] numeric

@attribute edge_84[6] numeric

@attribute edge_84[7] numeric
```

```
@attribute edge_84[8] numeric

@attribute edge_84[9] numeric

@attribute edge_88[0] numeric

@attribute edge_88[1] numeric

@attribute edge_88[2] numeric

@attribute edge_88[3] numeric

@attribute edge_16[0] numeric

@attribute edge_16[1] numeric

@attribute lmean_4_4[0] numeric

@attribute lmean_4_4[1] numeric

@attribute lmean_4_4[2] numeric

@attribute lmean_4_4[3] numeric

@attribute lmean_4_4[4] numeric

@attribute lmean_4_4[5] numeric

@attribute lmean_4_4[6] numeric

@attribute lmean_4_4[7] numeric

@attribute lmean_4_4[8] numeric

@attribute lmean_4_4[9] numeric

@attribute lmean_4_4[10] numeric

@attribute lmean_4_4[11] numeric

@attribute lmean_4_4[12] numeric

@attribute lmean_4_4[13] numeric

@attribute lmean_4_4[14] numeric

@attribute lmean_4_4[15] numeric

@attribute lvar_4_4[0] numeric

@attribute lvar_4_4[1] numeric

@attribute lvar_4_4[2] numeric
```

```
@attribute lvar_4_4[3] numeric
@attribute lvar_4_4[4] numeric
@attribute lvar_4_4[5] numeric
@attribute lvar_4_4[6] numeric
@attribute lvar_4_4[7] numeric
@attribute lvar_4_4[8] numeric
@attribute lvar_4_4[9] numeric
@attribute lvar_4_4[10] numeric
@attribute lvar_4_4[11] numeric
@attribute lvar_4_4[12] numeric
@attribute lvar_4_4[13] numeric
@attribute lvar_4_4[14] numeric
@attribute lvar_4_4[15] numeric
@attribute lmean_4_8[0] numeric
@attribute lmean_4_8[1] numeric
@attribute lmean_4_8[2] numeric
@attribute lmean_4_8[3] numeric
@attribute lmean_4_8[4] numeric
@attribute lmean_4_8[5] numeric
@attribute lmean_4_8[6] numeric
@attribute lmean_4_8[7] numeric
@attribute lmean_8_4[0] numeric
@attribute lmean_8_4[1] numeric
@attribute lmean_8_4[2] numeric
@attribute lmean_8_4[3] numeric
@attribute lmean_8_4[4] numeric
@attribute lmean_8_4[5] numeric
```

```
@attribute lmean_8_4[6] numeric

@attribute lmean_8_4[7] numeric

@attribute lvar_4_8[0] numeric

@attribute lvar_4_8[1] numeric

@attribute lvar_4_8[2] numeric

@attribute lvar_4_8[3] numeric

@attribute lvar_4_8[4] numeric

@attribute lvar_4_8[5] numeric

@attribute lvar_4_8[6] numeric

@attribute lvar_4_8[7] numeric

@attribute lvar_8_4[0] numeric

@attribute lvar_8_4[1] numeric

@attribute lvar_8_4[2] numeric

@attribute lvar_8_4[3] numeric

@attribute lvar_8_4[4] numeric

@attribute lvar_8_4[5] numeric

@attribute lvar_8_4[6] numeric

@attribute lvar_8_4[7] numeric

@attribute lmean_8_8[0] numeric

@attribute lmean_8_8[1] numeric

@attribute lmean_8_8[2] numeric

@attribute lmean_8_8[3] numeric

@attribute lvar_8_8[0] numeric

@attribute lvar_8_8[1] numeric

@attribute lvar_8_8[2] numeric

@attribute lvar_8_8[3] numeric

@attribute lmean_8_16[0] numeric
```

```
@attribute lmean_8_16[1] numeric

@attribute lmean_16_8[0] numeric

@attribute lmean_16_8[1] numeric

@attribute lvar_8_16[0] numeric

@attribute lvar_8_16[1] numeric

@attribute lvar_16_8[0] numeric

@attribute lvar_16_8[1] numeric

@attribute lmean_16_16 numeric

@attribute nx numeric

@attribute ny numeric

@attribute modes {0,1,2,3,4}
```

APPENDIX C

CONFIGURATION FILE- ENCODER.CFG

```
# New Input File Format is as follows

# <ParameterName> = <ParameterValue> # Comment

# See configfile.h for a list of supported ParameterNames

##########################################################################################

# Files

##########################################################################################

InputFile              = "C:\E\research\test_sequences\akiyo_qcif.yuv"       # Input sequeextern FILE *bits;ce,

YUV 4:2:0

FramesToBeEncoded     = 60     # Number of frames to be coded

SourceWidth          = 176    # Image width in Pels, must be multiple of 16

SourceHeight         = 144    # Image height in Pels, must be multiple of 16

ReconFile            = "test_rec.yuv"

OutputFile           = "test.avs"

AttributeFile            = "Attributes.dat"

###########################################################################################

# Encoder Control

###########################################################################################

ProfileIDC           = 16  # Profile IDC (16=Jiben)

LevelIDC             = 16  # Level IDC (e.g. 10=Level 1.0, 12 = Level 1.1)

IntraPeriod          =  0  # Period of I-Frames (0=only first)

QPFirstFrame         =  40 # Quant. param for first frame (intra) (0-63)

QPRemainingFrame      =  28 # Quant. param for remaining frames (0-63)

FrameSkip            =  0  # Number of frames to be skipped in input (e.g 2 will code every third frame)

UseHadamard          =  1  # Hadamard transform (0=not used, 1=used)

SearchRange          = 16  # Max search range

NumberReferenceFrames =  2  # Number of previous frames used for inter motion search (1-5)

RDOptimization        =  1  # rd-optimized mode decision (0:off, 1:on)
```

```
HalfPixelMVEnable     =  0  # if half pixel MV is enabled (0: off, 1:on)

ConstrainedIntraPred  =  0  # 0: No constrained; 1 constrained

##########################################################################################

#  High Level Syntax Set

#########################################################################################

# SeqID

##########################################################################################

# Fixed QP Stuff

##########################################################################################

FixedPictureQP       =  0  # 0: QP is variable in picture 1: QP is constant in picture

FixedSliceQP         =  0  # 0: QP is variable in Slice   1: QP is constant in Slice

SliceDeltaQP         =  0  # slice delta qp set, must be in [-32:31]

MbDeltaQP            =  0  #macroblock delta qp, must be in [-32:31]

EPMVFAST             =  0  # Enhanced PMVFAST for Fast ME (0=disable, 1=enable)

#Have to Do before next Release

#InterSearch16x16      =  1  # Inter block search 16x16 (0=disable, 1=enable)

#InterSearch16x8       =  1  # Inter block search 16x8  (0=disable, 1=enable)

#InterSearch8x16       =  1  # Inter block search  8x16 (0=disable, 1=enable)

#InterSearch8x8        =  1  # Inter block search  8x8  (0=disable, 1=enable)

#InterSearch8x4        =  1  # Inter block search  8x4  (0=disable, 1=enable)

#InterSearch4x8        =  1  # Inter block search  4x8  (0=disable, 1=enable)

#InterSearch4x4        =  1  # Inter block search  4x4  (0=disable, 1=enable)

#FME

##########################################################################################

# Loop filter parameter

##########################################################################################

LoopFilterDisable        =  0  # Disable loop filter in slice header (0=Filter, 1=No Filter)
```

```
LoopFilterParameter            =   1   # Configure loop filter parameter (0=parameter below ingored,
1=parameters sent)
SliceLFDisable          =  0  # 0: Filter in Slice edge   1: No filter in slice edge
LoopFilterAlphaCIOffset  =  0  # alpha_ci_offset, [-8:8]
LoopFilterCPOffset      =  -1 # cp_offset, [-16:16]
LoopFilerQPOffset       =  1  # qp_offset, [-40:23]
##########################################################################################
#Slice
##########################################################################################
Slice_Enable         =  1   # Must be enabled(0= no slice, 1= slice)
Slice_Parameter      =  11  # (1,2,3....Total_Mb_nr in one frame: slice format indicate the MBs in one
slice)
##########################################################################################
#SEI
##########################################################################################
SEIEnable           =  1   # 1: enable, 0 : disable
HRDEnable            =  1   # 1: enable, 0 : disable
##########################################################################################
#Isolated Region
##########################################################################################
IREGEnable          = 0   # 1 : enable, 0 : disable
IREGEvolutionRate    = 11 # evolution rate of isolated region
LoopfilterMode       = 1   # not used yet !
##########################################################################################
# Hypothetical reference decoder (HRD)
##########################################################################################
NumberofLeakyBuckets    =  8                    # Number of Leaky Bucket values
```

```
LeakyBucketRateFile      =  "leakybucketrate.cfg"  # File from which encoder derives rate values

LeakyBucketParamFile     =  "leakybucketparam.cfg" # File where encoder stores leakybucketparams

##########################################################################################

#Rate control

##########################################################################################

PictureRate          =    30   # frame/s

RateControlEnable    =    0    # 0 Disable, 1 Enable

Bitrate              =    64000  # Bitrate(bps)

InitialQP            =    0    # Initial Quantization Parameter for the first I frame

                            # InitialQp depends on two values: Bits Per Picture,

                            # and the GOP length

ChannelType          =    0    # type of channel( 1=time varying channel; 0=Constant channel)
```

REFERENCES

[1]http://ee.uta.edu/Dip/Courses/EE5359/Multimedia%20Processing%20project%20report%20final.pdf ; course website UTA

[2]P. Carrillo, H. Kalva and T. Pin, "Low complexity H.264 video encoding", SPIE. vol.7443, Paper # 74430A, Aug. 2009

[3]Kusrini and S. Hartati, "Implementation of C4.5 algorithm to evaluate the cancellation possibility of new student applicants at STMIK AMIKOM YOGYKARTA", Proceedings of the International Conference on Electrical Engineering and Informatics Institute Teknologi Bandung, Indonesia June 17-19, 2007

[4]S. Saponara, et al, "Adaptive algorithm for fast motion estimation in H.264/MPEG-4 AVC", Proc. Eusipco2004, pp. 569 – 572, Wien, Sept. 2004

[5]Décisions tree basics : http://dms.irb.hr/tutorial/tut_dtrees.php

[6]Weka tool software :http://www.cs.waikato.ac.nz/ml/weka/

[7]X. Jing and L. P. Chua, "An efficient inter mode decision approach for H.264 video coding ", IEEE, International Conference on Multimedia and Expo (ICME), pp. 1111-1114, July 2004.

[8]Download AVS-M software from:ftp://159.226.42.57/public/avs_doc/avs_software(password protected)

[9]Power-point slides by L. Yu, chair of AVS video:

http:// www-uta.edu/dip/Courses/EE5351/ISPACSAVS.pdf

[10]L.Fan, "Mobile multimedia broadcasting standards", ISBN: 978-0-387-78263-8, Springer US, 2009

[11]AVS working group official website, http://www.avs.org.cn

[12]Test sequences can be downloaded from the site http://trace.eas.asu.edu/yuv/index.html

[13]Y.Xiang et al, "Perceptual evaluation of AVS-M based on mobile platform", Congress on image and Signal Processing, 2008, vol. 2, pp.76 – 79, 27-30 May 2008.

[14]M.Liu and Z.Wei, "A fast mode decision algorithm for intra prediction in AVS-M video coding", vol.1, ICWAPR apos; 07, Issue, 2-4, pp.326 – 331, Nov. 2007.

[15]L.Yu et al, “Overview of AVS-Video: Tools, performance and complexity,” SPIE VCIP, vol. 5960, pp. 596021-1~ 596021-12, Beijing, China, July 2005.

[16]L. Yu, S. Chen, and J. Wang, “Overview of AVS-video coding standards” special issue on AVS, SP:IC, vol. 24, p. 247-262, April 2009.

[17]Y. Shen, et al, “A simplified intra prediction method”, AVS Doc. AVS-M 1419, 2004.

[18]F. Yi, et al, “An improvement of intra prediction mode coding”, AVS Doc. AVS-M 1456, 2004.

[19]L. Xiong, “Improvement of chroma intra prediction”, AVS Doc. AVS-M1379, 2004

[20]X. Mao, et al, “Adaptive block size coding for AVS-X profile”. AVS Doc. AVS-M2372, 2008.

[21]R. Wang, et al, “Sub-pixel motion compensation interpolation filter in AVS”, 2004 IEEE International Conference on Multimedia and Expo, pp. 1:93-96, 2004.

[22]F.Yi, et al, “Low-complexity tools in AVS Part 7”, J. Computer Science Technology, vol.21, pp. 345-353, May, 2006

[23]W.Gao and T.Huang “AVS Standard -Status and Future Plan”, Workshop on Multimedia New Technologies and Application, Shenzhen, China, Oct., 2007.

[24]W.Gao, et al, “AVS the Chinese next-generation video coding standard,” National Association of Broadcasters, Las Vegas, 2004.

[25]Z.Ma, et al, “Intra Coding of AVS Part 7 Video Coding Standard”, J. Comput. Sci. Technol, vol.21, Feb.2006.

[26]W.Hua, M.Cuiqin and Z.Lijuan, “A brief review of machine learning and its application”. International conference on ICIECS, pp. 1-4, 2009.

[27]] 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.

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

[29] 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.

[30] P.Hart, “The condensed nearest neighbor rule”, IEEE Trans. on Information theory, Vol. 14, pp. 515–516, May 1968.

[31] G. Gates, “The reduced nearest neighbor rule”, IEEE Trans. on Information
Theory”, Vol. 18, pp. 431–433, May 1972.

[32] B. Dasarathy,”Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments”. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 2, pp. 67–71. Jan. 1980.

[33] T. Cover and P. Hart, “Nearest neighbor pattern classification”. IEEE Trans. Inform. Theory, IT-13(1), pp. 21–27, Jan.,1967.

[34] C. Hoare. "Partition: Algorithm 63," "Quicksort: Algorithm 64," and "Find: Algorithm 65." Comm. ACM 4(7), pp. 321-322, 1961.

[35] X. Wu et al, “top 10 algorithms in data mining” survey paper, Springer-Verlag London Limited 2007.

[36] D. W. Aha, D. Kibler and M.Albert, “Instance based learning algorithms”, Kluwer Academic
Publishers, Boston, pp. 37-66 (1991).

[37] L. Yu, S. Chen, J. Wang, “Overview of AVS-video coding standards”, Signal Processing: Image
Communication, vol. 24, pp. 247–262, May 2009.

[38] H. Abbas, R. Sarker and C. Newton. “Data mining: a heuristic approach”, idea group publishing, 2002.

[39] S.Bow, “Pattern recognition and image processing” signal processing and communications series, 2002.

[40] R.Kennedy et al, “ Solving data mining problems through pattern recognition”, the data warehousing institute series , Prentice Hall ptr, 1997.

[41] O.Maimon and L.Rokach, “ The data mining and knowledge discovery handbook”, Springer, 2005.

[42] J.R.Quinlan, “Improved use of continuous attributes in C4.5”, J. Artificial intelligence research, vol. 4, pp. 77-90, 1996.

[43] J.R.Quinlan,”C4.5: programmes for machine learning”, san mateo, calif.: Morgan Kaufmann, 1993.

[44] S.Dua and X.Du, “Data mining and machine learning in cyber security”, Boca Raton, FL CRC press, 2011.

[45] Information technology- advanced coding of audio and video part -7: mobile video

[46]S. Sridhar, “Multiplexing and demultiplexing of AVS China video with AAC Audio and maintaining lip synchronization during playback”, M.S.Thesis, EE Dept. UTA, Dec. 2010.

BIOGRAPHICAL INFORMATION

Pragnesh Ramolia was born in Rajkot city of Gujarat state in India. He received his Bachelor of engineering degree in Electronics and Communication from Veer narmad south Gujarat University, Gujarat, India in May 2007. He started pursuing his master’s degree at The University of Texas at Arlington from January 2009. He did his internship at 3S Network Inc. in summer 2010 and fall 2010 in RF Field testing. He will be awarded Master of Science in electrical engineering in May 2011. His areas of interest are multimedia processing, wireless communications and networking.