PERFORMANCE EVALUATION OF GREEDY HEURISTIC FOR SIP ANALYZER IN H.264/SVC

by

JAYDEEP VIJAY INAMDAR

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 2008

॥ गुरुर्ब्रम्हा गुरुर्विष्णु गुरुर्देवो महेश्वरः ॥

॥ गुरुःसाक्षात् परब्रम्ह तस्मै श्रीगुरवे नमः ॥

ACKNOWLEDGEMENTS

I am grateful to my thesis advisor Dr. Soontorn Oraintara for the confidence he showed in me and helping me in exploring the area of Video Coding. He has been great source of inspiration and help in this thesis.

I would like to extend my sincere gratitude to UTA, for providing me opportunity and excellent facilities to excel in my graduate studies.

I would like to thank Dr. K. R. Rao for being continuous source of guidance throughout my graduate studies and thesis. I also wish to thank Dr. Manry for being part of my thesis committee. My special thanks to Dr. Yung-Lyul Lee, Sejong University, Korea for his help in obtaining the appropriate test sequences.

I am thankful to members of Multirate Signal Processing Lab and my friends in UTA for their encouragement and cooperation. I also appreciate Sangseok Park, Rahul Panchal and Harishankar Murugan for their words of wisdom.

Finally I would like express my gratitude to my parents and my brother who have always supported me in all my endeavors and the above all GOD for his blessings on all of us.

March 5, 2008

ABSTRACT

PERFORMANCE EVALUATION OF GREEDY HEURISTIC

FOR SIP ANALYZER IN H.264/SVC

Publication Number: _____

Jaydeep Vijay Inamdar, M.S.

The University of Texas at Arlington, 2008

Supervising Professor: Dr. Soontorn Oraintara

The latest scalable video coding standard, H.264/SVC uses many components of H.264/AVC standard to maintain backward compatibility and also has proposed many tools to support scalable video coding with increased coding efficiency. SIP Analyzer is one such tool incorporated in JSVM (Joint Scalable Video Model) software which is the reference software for the SVC project.

SIP Analyzer implements Selective Interlayer Prediction strategy to encode the bitstream so as to improve coding performance in scenarios where multiple adaptation is not needed without losing much if the same bitstream is used in scenario where multiple adaptation is needed. Core of this algorithm is a 0-1 Knapsack Problem that decides the right combination of lower layer frames for which interlayer prediction can be safely turned off. Current implementation solves the Knapsack Problem using Dynamic Programming approach. Even though it gives optimal solution to the problem, it is computationally complex to be implemented in real time encoders.

In this thesis we attempt to solve the problem using Greedy heuristic approach. Since it's a heuristic approach, solution given by it may differ from the optimal solution. We evaluate the performance of Greedy heuristic approach both qualitatively and quantitatively and summarize the observations which can serve as reference for the developers. It has been verified that Greedy heuristic approach greatly reduces the SIP analyzer complexity both in time and in space without compromising much with the quality.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Continuous improvement in computing power and communication technologies coupled with increasing density of storage media is enabling increased number of multimedia applications. Most of the interactive consumer multimedia devices today include video based applications. They range from multimedia messaging, high-definition video broadcasts, video telephony and video conferencing to video storage on DVD, Blu-ray, HD-DVD optical discs. All these applications attribute to the advances in video compression technology which is focus of this thesis.

1.1 Overview of Video Coding Standards

Video signal can be compressed by using various proprietary or standardized algorithms. Proprietary algorithms are developed, owned and used by smaller groups or commercial organizations for business or research purposes and they lack global compatibility and hence are of less significance. Examples include Microsoft® Windows Media Video (WMV xx) series, RealNetworks® Real Video (RV xx) series etc. More important families of compression standards are published by internationally standardized bodies such as the International Telecommunication Union (ITU), the International Organization for Standardization (ISO) or the Motion Pictures Expert Group (MPEG).

ISO/IEC (International Organization for Standardization/ International Electrotechnical Commission) and ITU-T (International Telecommunications Union/Telecommunication Standardization Sector) are two main bodies for recommending Speech/Audio/Video coding standards. ITU-T has designed the well known H.264x series. H.261 [1] being the first member

of this series, during following years ITU-T released H.262 [2], H.263 [3], H.263+, H.263++ and H.264 [4] standards targeting wide range of applications. Details can be found on ITU-T website [5].

The MPEG family of standards includes MPEG-1 (ISO/IEC 11172), MPEG-2 (ISO/IEC 13818), MPEG-4(ISO/IEC 14496), MPEG-7 (Multimedia Content Description Interface), and MPEG-21 (ISO /IEC 21000). Details can be found on MPEG website [6]. The MPEG working group is part of the Joint ISO/IEC Technical Committee on Information Technology.

In order to come up with a new video coding standard capable of providing better video quality at substantially lower bitrates than all the previous standards (MPEG-2, H.263, or MPEG-4 Part 2) without increasing design complexity, ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) formed Joint Video Team (JVT) which includes members from both the teams. They published H.264 standard/MPEG-4 Part 10 formally known as H.264/MPEG-4 AVC, in May 2003 [7]. JVT then developed extensions to the standard such as Fidelity Range Extension (FRExt) to enable higher quality by supporting increased sample bit depth precision and high resolution color information. It also added five new extensions supporting professional applications.

H.264/AVC has attracted a lot of attention from industry in recent years and is being increasingly used in variety of applications and products. It is expected to become the most popular video coding standard for almost all the industry applications.

Scalable Video Coding has been topic of research and standardization for almost 20 years [8]. Given the gaining popularity and wide industrial use of H.264/AVC standard, it scalable extension has been proposed and has officially become part of H.264/AVC standard as Annex G. It is supposed to supersede all the previously proposed scalability tools as in MPEG-2 video, H.262, H.263 and MPEG-4 Visual by overcoming their problems like significant loss in coding efficiency and increased decoder complexity. H.264/SVC is remains the most recent and revolutionary scalable video coding standard as of now.

1.2 Outline of work

The H.264/SVC standard uses many components of H.264/AVC standard to maintain backward compatibility and also has proposed many tools to support scalable video coding with increased coding efficiency. SIP Analyzer is one such tool incorporated in JSVM (Joint Scalable Video Model) software which is the reference software for the SVC project.

SIP Analyzer implements Selective Interlayer Prediction strategy [9] to encode the bitstream so as to improve coding performance in scenarios where multiple adaptation is not needed without losing much if the same bitstream is used in scenario where multiple adaptation is needed. Core of this algorithm is a 0-1 Knapsack Problem that decides the right combination of lower layer frames for which interlayer prediction can be safely turned off. Current implementation solves the Knapsack Problem using Dynamic Programming approach. Even though it gives optimal solution to the problem it is computationally complex to be implemented in real time encoders.

In this thesis we attempt to solve the problem using Greedy heuristic approach. Since it's a heuristic approach, solution given by it may differ from the optimal solution. We evaluate the performance of Greedy heuristic approach both qualitatively and quantitatively and summarize the observations which can serve as reference for the developers. It has been verified that Greedy heuristic approach reduces the SIP analyzer complexity both in time and in space without compromising much with the quality.

1.3 Reader's Guide

Rest of the document is organized as follows:

Chapter 2 gives brief overview of latest single layer video coding standard H.264/AVC.

Chapter 3 extends this idea and introduces the scalable extension to H.264/AVC, also known as H.264/SVC.

Chapter 4 discusses Selective Interlayer Prediction (SIP) strategy employed in H.264/SVC and the SIP Decision problem which boils down to 0-1 Knapsack Problem.

Chapter 5 gives basic idea of Knapsack Problem and various approaches to solve it.

Chapter 6 summarizes the contribution of this thesis to the SIP Analyzer in H.264/SVC.

Chapter 7 concludes with results, observations and future scope.

CHAPTER 2

OVERVIEW OF H.264/MPEG-4 PART 10

H.264/MPEG-4 Part 10 also known as H.264/AVC (Advanced Video Coding) is the latest single layer coding standard jointly developed by ITU-T Video Coding Expert Group (VCEG) and ISO/IEC Motion Pictures Expert Group (MPEG). The joint group is known as JVT (Joint Video Team). H.264/AVC standard aims at achieving significant enhancement in coding efficiency and error robustness in comparison to previous video coding standards like MPEG-2, H.263 and MPEG-4 Part2 with a range of features supporting better quality and low bitrate for streaming video over fixed and wireless networks and over different transport protocols.

Numerous papers and tutorials have been written about video coding theory in general and also about industry standard H.264/AVC. In this chapter we take a quick review of distinguishing features of the H.264/AVC standard. Reader is suggested to refer [10] [11] [12] [13] for comprehensive information about video coding techniques and standards. Detailed information about H.264/AVC can be found in [14] [15].

2.1 Introduction

Similar to previous standards (MPEG-1, MPEG-2 and MPEG-4) H.264/AVC standard does not explicitly define encoder-decoder pair specifications. Rather it specifies syntax of a valid encoded bitstream along with method to decode it. With this defined the implementation details of encoder are completely left to the developers.

H.264/AVC uses the same basic functional elements as in previous standards [13] i.e. block transform to exploit spatial redundancy, motion compensated prediction to exploit temporal redundancy, quantization to control bitrate, entropy encoding to reduce statistical

correlation. However important changes occur in details of each element. It introduces a new intra-picture prediction technique, new 4x4 integer transform, variable block sizes, deblocking filter, multiple reference frames, quarter pixel precision for motion compensation and improved lossless coding. In order to reduce complexity it introduces new multiplier free integer transform. Multiplier operation for exact transform is combined with quantization scaling. To cope with degradation arising due to channel noise H.264/AVC adds parameter setting, flexible macro block ordering, switched slice, redundant slice methods to data partitioning for error resilience.

2.2 H.264/AVC Profiles

Although the H.264/AVC standard proposes many tools to improve coding efficiency and better visual experience, not all the tools are needed by all the applications. If every decoder is forced to implement all these tools, it will unnecessarily increase decoder complexity. On the other hand the interoperability between certain class of applications and the related applications should also be maintained. To address this, the standard defines various subsets of coding tools intended for variety of applications. These subsets are called 'Profiles'. Currently standard defines following profiles:

- *Baseline:* Mainly used for videoconferencing and mobile video applications.
- *Main:* Used mainly for video storage and playback and also in some studio applications.
- *Extended:* Used for streaming video applications.
- *High:* Used for high quality studio distribution.

For a given profile, the performance limits of codecs are defined as collection of levels, each specifying restrictions on coding process such as sample rate, decoding speed, number of blocks per second, coded bit rate, picture buffer size etc. Additional details of each profile and level can be found in [15].

Profiles have common as well as specific coding tools as shown in fig. 2.1.

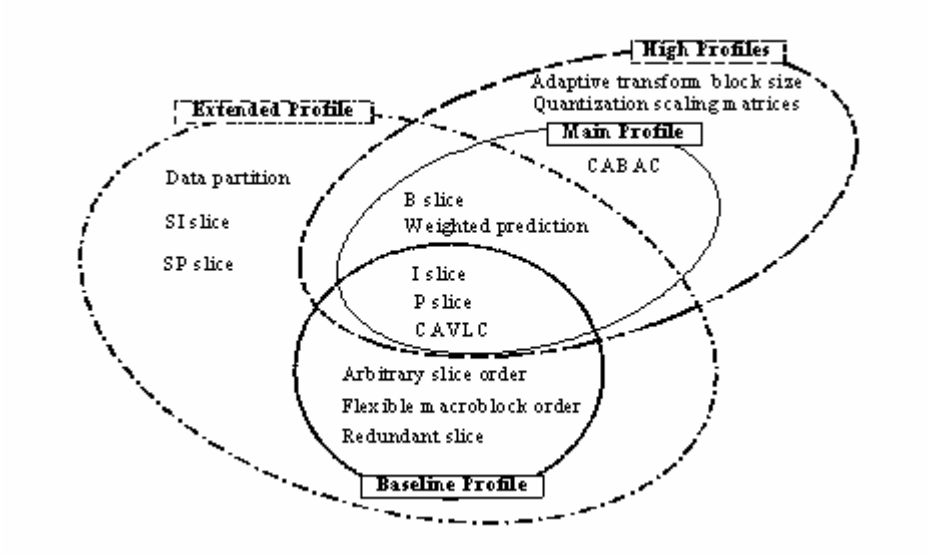

Fig. 2.1 Specific coding tools supported by H.264/AVC profiles [16]

2.3 Layered Coding Structure

The H.264/AVC bitstream has been coded in two layers: Network Abstraction Layer (NAL) and Video Coding Layer (VCL). VCL contains the actual coded video information. Purpose of NAL is to abstract VCL data such that it would be convenient to store on storage media or transmit it on variety of communication channels or networks.

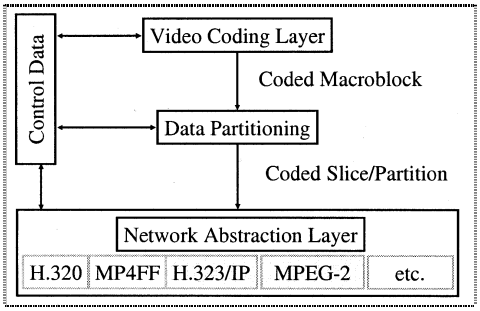

Fig. 2.2 Structure of H.264/AVC encoder [14]

2.3.1 *Network Abstraction Layer*

NAL formats the compressed video (VCL) data and provides additional non-VCL information such as parameter setting etc [16] in such a way that it can be conveniently coded as byte-stream or packet-based format.

Fig. 2.3 NAL unit syntax [17]

All data related to video stream is encapsulated in NAL Units (referred as NALU). Format of NALU is shown in fig. 2.3. First byte of each NALU is a header byte and rest all is data. First bit is always zero. Next two bits indicate whether content of NALU is sequence or picture parameter set or a slice of reference picture. Next five bits specify NALU type corresponding to payload being carried in NALU which may be VCL or non-VCL type. The picture and parameter sets play pivotal role during decoding. They define some parameters of data being encoded which are used for decoding. So during transmission, these two sets are sent frequently. If the bitstream has to be played from a random point, these parameters along with next IDR (Instantaneous Decoder Refresh) picture are used.

2.3.2 *Video Coding Layer*

The VCL design follows block based hybrid video coding approach. The basic source coding algorithm is, to exploit inter-picture and intra-picture redundancies in temporal, spatial domains and apply transforms and lossless coding to further exploit statistical redundancy. There is no single functional block in VCL which gives dramatic improvement in coding gain but it is the cumulative effect of modifications done in implementation details of these blocks in H.264/AVC standard with reference to previous standards.

Fig.2.4 and 2.5 show architecture and core building blocks of H.264/AVC coding system.

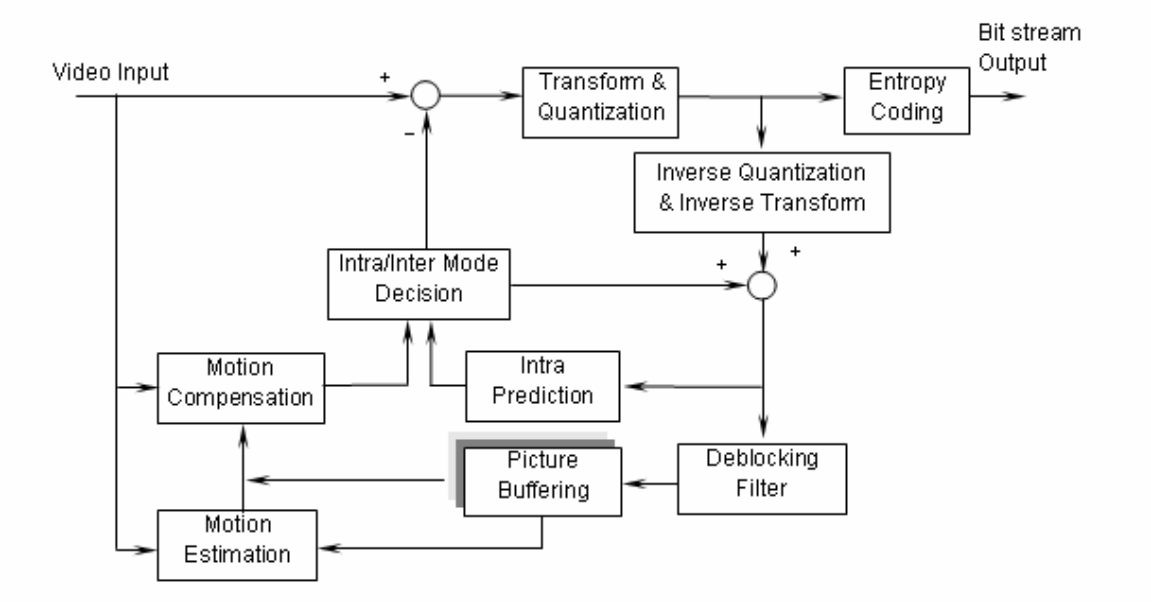

Fig. 2.4 H.264/AVC Encoder Structure [16]

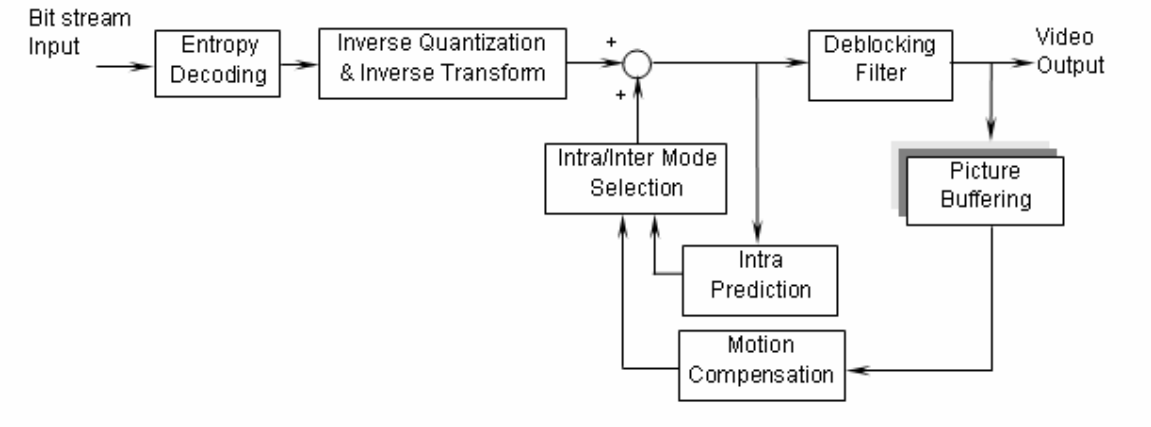

Fig. 2.5 H.264/AVC Decoder Structure [16]

We describe details of components in fig. 2.4 and 2.5:

2.3.2.1 Intra Prediction

Unlike previous standards in which intra macroblocks are coded by themselves without temporal prediction which significantly increases the bit rate, H.264/AVC proposes predicting intra macro block from original signal itself. To encode a block or macro block in Intra-coded mode a prediction is formed from previously reconstructed unfiltered blocks and this prediction is coded. The standard specifies intra-prediction as linear interpolation of pixels from adjacent edges of neighboring macroblocks that are decoded before current macro block. These interpolations are directional in nature with multiple modes implying spatial direction of prediction. For luminance pixels with 4x4 partitions, 9 prediction modes as shown in fig. 2.6 are defined.

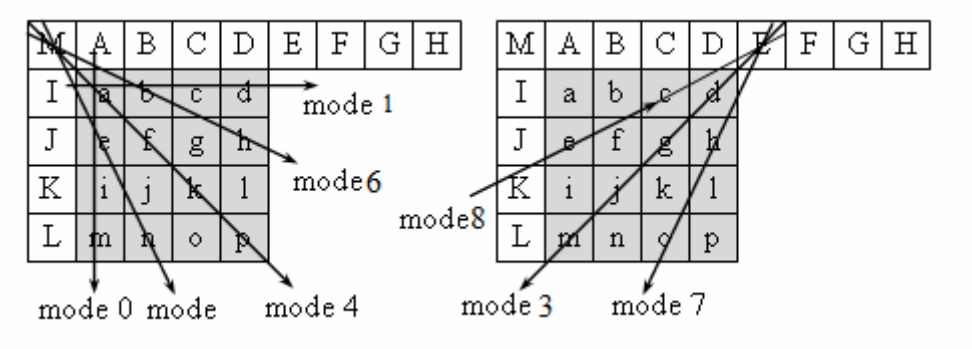

Fig. 2.6 Intra 4x4 Prediction Mode Directions [16]

For predicting 16x16 luma components of a macro block, mode 0 (vertical), mode 1(horizontal), mode 2 (DC) and mode 4 (plane) are used. Chroma prediction is defined for three possible block sizes: 8x8 in 4:2:0 format, 8x16 in 4:2:2 format and 16x16 in 4:4:4 format. There are 4 prediction modes for all chroma sizes: mode 0 (DC), mode 1 (horizontal), mode 2 (vertical) and mode 3 (plane) similar to 16x16 luma prediction modes.

2.3.2.2 Inter Prediction

This block includes both motion estimation (ME) and motion compensation (MC) to reduce temporal redundancy. Current picture can be partitioned into macroblocks or even smaller blocks. A 16x16 macro block can be partitioned into 16x16, 8x16, 16x8, 8x8 and for 8x8 macro block mode size are 8x8, 4x8, 8x4 and 4x4. Choosing smaller partition facilitates better prediction or less prediction error. But number of motion vectors and extra signaling data increases for overall picture. So there is always a tradeoff depending on input signal characteristics. The process generates a predicted version of a rectangular array of pixels by choosing another similar sized array from previously decoded and stored reference picture and translating the reference array to position of current array. This translation can be specified at quarter pixel accuracy for luma components. Motion vectors for chroma components are scaled accordingly depending on source sampling format used.

This process also involves selection of reference picture frame from a number of previously encoded (and decoded) and stored pictures. The reference picture buffer management is needed to update the reference frames depending on available system memory.

2.3.2.3 Transform

H.264 standard is based on block based transform to reduce spatial redundancy. Rather than 8x8 floating point transform as used in MPEG-2 / MPEG-4 part 2, a new 4x4 integer transform (8x8 for high profile) is proposed in H.264/AVC and its transform coefficients are explicitly specified. It is perfectly invertible. The standard avoids multiplication in transformation to reduce computational complexity and combines that as a scaling factor with quantization. Standard explicitly specifies transform and inverse transform matrices for 4x4 luma, 8x8 luma (high profile) and also specifies Hadamard transform matrices for luma and chroma DC coefficients. In addition to this, encoders can use default perceptual scaling matrices as

suggested by FRExtensions or even customize them and send to the decoder. They help in shaping the quantization error by exploiting properties of Human Visual System (HVS). Scanning of the transform coefficients is based on decreasing variances and maximizing number of zero-valued coefficients along the scan to generate run-level events to be coded with VLC. Popular methods are zig-zag or alternate scan.

2.3.2.4 Quantization

Quantization is also called as 'scaling' in the standard. High profile supports HVS based quantization scaling matrices same as in MPEG-2. The scale factor for each element in each sub block varies as function of quantization parameter associated with its associated macroblock. The rate-control algorithm in encoder decides this quantization parameter. As mentioned earlier the quantization (and inverse quantization) equations are modified to incorporate the scaling factors adjusted in transform (and inverse transform) operations.

2.3.2.5 Deblocking Loop Filter

In H.264/AVC visually disturbing blocking artifacts can be generated due to coarse quantization of block based integer transform in intra and inter residue coding. Another source of these block artifacts is motion compensated prediction. Since there is almost never a perfect match between interpolated block and the actual block from reference frame, discontinuities on the edges of the copied blocks of data arise and they are carried forward in next block which uses this block as reference. Even though small transform size (4x4) makes this artifact less visible, deblocking loop filter still enhances the performance. However it is computationally complex. It is an adaptive filter operation which is dependent on several factors such as quantization parameters, magnitude of motions vectors, macro block coding type etc. The detailed algorithm for loop filter decision logic can be found in [15]. Result of loop filtering is saved as reference picture.

(a) (b)

Fig 2.7 Details of luminance output in case of (a) No loop filtering (b) With loop filtering [18]

2.3.2.6 Mode Decision

It decides coding mode for each macroblock. To achieve highest coding efficiency mode decision may use rate-distortion optimization. It works with rate-control loop and outcome is the optimal coding mode for given macro block.

2.3.2.7 Entropy Coding

H.264/AVC uses a number of techniques such as Golomb codes, Context Adaptive Binary Arithmetic Coding (CABAC) and Context Adaptive Variable Length Coding (CAVLC) for entropy coding. All the syntax elements except the residual data are coded using Exp-Golomb codes [19]. For coding residual data more sophisticated CAVLC or CABAC (in main and high profiles) is used. CABAC is more complex than CAVLC but has more coding efficiency.

2.3.2.8 B-Slices

Bidirectional prediction is very efficient in reducing temporal correlation using many reference frames. H.264/AVC generalizes concept of B-slices in comparison to previous standards e.g. pictures containing B-slices can be used as reference frames for motion compensated prediction. In addition, H.264/AVC supports not only forward/backward prediction but also forward/forward and backward/backward prediction for scene change scenario. It also

introduces weighted prediction and direct mode for inter prediction. More details can be found in [20] [21].

2.3.2.9 SI and SP Slices

In previous standards switching between bitstream is possible only at I-picture intervals. Therefore supporting such perfect switching requires introducing I-frames frequently thereby increasing bitrate. H.264/AVC introduces two new frame types SI and SP to handle such switching.

- SP Slice: a so called switching P slice is coded such that switching between different pre-coded pictures is possible e.g. If there are two different bitstreams P(1, k) and P(2, k) corresponding to same video sequence but coded at different bitrates, within each bitstream SP frames are placed at locations where we can switch from one stream to another. For switching from P(1,3) to P(2,3) we can use SP(3) frame that produces P(2,3) from P(1,3).
- SI Slice: the switching I slice allows exact match of a macro block in SP slice for random access and error recovery.

2.3.2.10 Error Resilience

Robustness to channel noise and data errors/loss and operation over wide variety of networks is facilitated by following features. Details can be found in [14].

- Parameter set structure
- NAL unit syntax structure
- Flexible slice size
- Flexible macro block ordering (FMO)
- Arbitrary slice ordering (ASO)
- Redundant pictures

- Data partitioning
- SI/SP synchronization

2.3.2.11 H.264/AVC Decoder

It takes valid .264 bitstream as input and decodes it produce raw video sequence in YUV format. Bitstream first passes through entropy decoder which extracts header, syntax information and slice data with motion vectors. It is followed by scaling and inverse quantization and then inverse transformation to bring the information in pixel domain. If this is residue, it is added with appropriate reference frame from reference picture buffer and motion compensated to reconstruct current frame. This reconstructed frame is then passed though same deblocking filter as used on encoder side to remove blocking artifacts.

2.4 Summary

This chapter summarizes latest single layer coding standard H.264/AVC developed and standardized by JVT in May 2003. It can match the best MPEG-2 video quality at up to half the data rate [22]. It also delivers excellent video quality from 3G to HD between 40 kbps and 10 Mbps [23].

Most important characteristics are 4x4 transform, quarter pixel accuracy motion vectors, multiple reference prediction, deblocking filter, CAVLC, CABAC along with error resilience, stream switching etc.

Its wide acceptance from industry has motivated further developments to support various application dependent scenarios as extension to existing standard.

CHAPTER 3

SCALABLE EXTENSION TO H.264/AVC

3.1 Introduction

Emergence of broadband wireless technology, rapid developments in network infrastructures, storage capacity, computing power have enabled proliferation of multimedia applications on consumer PCs connected to World Wide Web and consumer devices like cell phones, PDAs, networked handheld gaming devices. Video applications are becoming important part of this revolution.

In comparison to older TV transmission systems, today's video transmission is characterized by varying connection quality. Also, receiving devices today, range from handheld cell phones, PDAs with small screens, limited processing/battery power to high definition TVs with large display panels.

It may be streaming video over packet based networks or video communication over wireless broadband networks, both face similar challenges:

- Bandwidth Fluctuations
- High bit error rate / Packet loss rate
- Heterogeneity amongst networks/receiving terminals

Due to these major problems it is difficult to develop a single video application that can meet demands of variety of receivers residing on the other side of the network. But the same source content should be provided with different bit-rates, different frame rates, different display/quality resolutions, different loss/error handling mechanisms. Variation in connection quality should not result interruption in service or unacceptably bad quality to the receiver.

3.2 Scalable Video Coding

An appealing solution to the problems posed by video transmission for modern communication systems for variety of receiving devices is **Scalable Video Coding (SVC)**. Scalable Video Coding is to produce a compressed video stream, parts of which can form other valid sub-streams and can be decoded by the given decoder. These sub-streams represent the same source video content with reduced reconstruction quality as compared to the output which is produced when the complete bitstream is decoded. The immediate advantage of such scheme is – Video Content can be coded only once with the highest desired coding gain and quality and can later on serve to demand of different applications with varied constraints on bitrate, quality, bandwidth availability etc. by partially extracting and decoding appropriate substreams out of it.

Usually scalability can be: *Spatial* Scalability, *Temporal* Scalability and *Quality* (Signal to Noise Ratio - SNR) Scalability. Also there are, *Region of Interest* (ROI) and *Object based* scalability modes but they are rarely used. A coded scalable bitstream can be configured to incorporate any combination of these basic scalable modes. Depending on the modes used in creating original coded scalable bitstream, the partially extracted and decoded bitstreams can reproduce content with degraded quality (lower SNR), lower bitrate or lower frame rate or any combination of them to serve various applications.

A video stream which is not coded using any such scalability modes, but is coded for fixed frame rate, bit rate and resolution is called **Single Layer** stream.

3.3 Applications

There is multitude of application scenarios where such scalable video stream can be used [8]:

For instance, a video server serving variety of end user devices with different display capabilities with the same source content over variety of network connections with different

bandwidths. With properly configured encoder, the bitstream is encoded only once with highest desired resolution and bitrate and then extracted and substreams are formed as per need to serve various clients.

Another interesting scenario is video transmission over a channel with unpredictable throughput variations and/or relatively high packet loss rates. Since the scalable video stream usually contains different parts with different importance in terms of quality, they can be coded with unequal error protection schemes such that stronger protection is provided to more important information i.e. base layer information and relatively weaker protection to subsequent enhancement layers. Such scheme can help in graceful degradation up to certain degree of channel error/loss rates. They can be assisted with the Media Aware Network Elements (MANE) by removing unwanted parts from the bitstream before forwarding it to terminals as per their feedback.

One different kind of application is for video archiving in video recorders, home networking or for video surveillance applications. In such scenario, the high quality parts of the video stream can be deleted after some expiration time to save storage space assuming that as time passes the probability that they will be viewed again and again lowers down.

For web browsing of video library, scalable video coding can generate a low resolution preview without decoding a full resolution picture.

In general, Scalable Video Coding addresses the issue of reliable delivery of video to diverse systems over diversified network connections using available system resources, especially in scenarios where the end system capabilities, resources and network conditions are not known beforehand.

3.4 Scalable Video Coding Standards

Given the need of scalable video coding technique to serve tremendously varied end-systems over uncertain network conditions, earlier video coding standards tried to handle this

issue. Scalability has already been present in prior video coding standards like MPEG-2 [24], H.263 [25] and MPEG-4 Visual [26] in form of scalable profiles. However the quality and spatial scalability came along with considerable increase in decoder complexity and significant loss in coding efficiency as compared to corresponding non-scalable profiles. Due to these drawbacks earlier attempts to introduce scalability were not accepted by industry.

Therefore, apart from support for various scalability modes, most important design criteria for success of a scalable coding standard are coding efficiency and decoder complexity. In addition to this, the bitstream produced by a scalable codec should also compete in performance with simulcast video transmission and also against video transcoding in multipoint control units in 3-G systems.

3.5 'H.264/SVC' - Scalable extension to H.264/AVC

H.264/AVC standard which was finalized in May 2003 is now a well established video coding standard and derivative standardization projects have started emerging out of it. Most important of them is called Scalable Video Coding (SVC) Project [27]. Initially started within MPEG by the time H.264/AVC standard was on its way of being finalized, it was later moved to Joint Video Team (JVT) in 2005. It was decided to be as an amendment of existing H.264/AVC standard. Main motivation was to remove the drawbacks of earlier scalable coding standards.

Initially to handle problem of drift [28][29] due to lost synchronization in encoder-decoder motion compensation prediction loops, a 3-d wavelet based structure was proposed [30], but it was later removed due to increased design complexity and DPCM based structure with some modifications was adopted. Latest draft of the standard [31] includes some more modifications than first model [32], like methods for non-dyadic scalability and interlaced processing.

The scalable extension to H.264/AVC standard is also referred as H.264/SVC. In this document we use SVC with reference to the H.264/SVC standard.

3.5.1 Scalability Modes in H.264/SVC

Similar to prior scalable video coding standards, in H.264/SVC standard the basic scalability modes are:

- Temporal Scalability
- Spatial Scalability
- Quality (SNR) Scalability

Here we review changes needed in H.264/AVC standard to support these scalability modes.

The scalable extension of H.264/AVC proposes a layered video codec. Usually codec configuration and structure depends upon the scalability space needed by the application. Fig 3.1 shows a typical encoder structure with two spatial layers. Each layer can be either spatial or coarse-grain SNR layer.

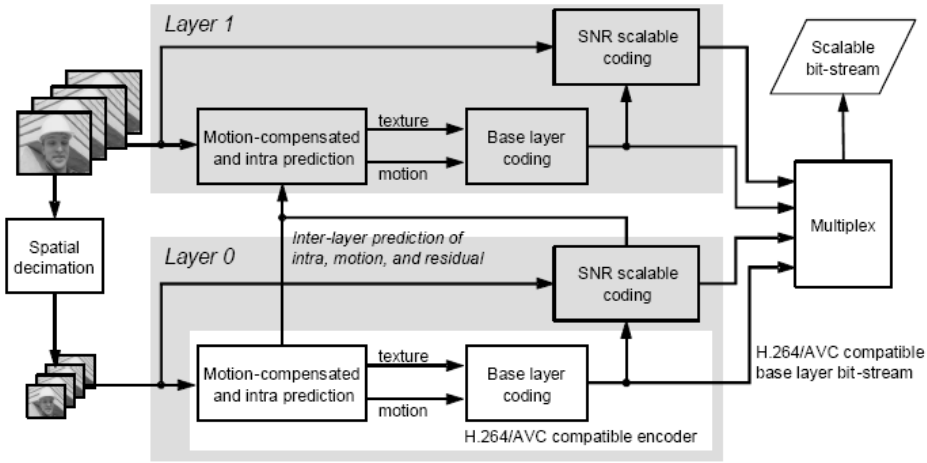

Fig. 3.1 Typical encoder structure with two spatial layers [8]

3.5.1.1 Temporal Scalability

Given bitstream is said to provide temporal scalability when set of corresponding access units can be partitioned into a temporal base layer and set of temporal enhancement layers with following property: Let temporal layers be identified by identifier *T* which is set to 0

for the base layer and incremented for each successive temporal enhancement layer. Then for each natural number *k* (0 for base layer), the bitstream that is obtained by removing all access units of all temporal layers with a temporal identifier *T* greater than *k,* forms a valid sub stream for given decoder [8].

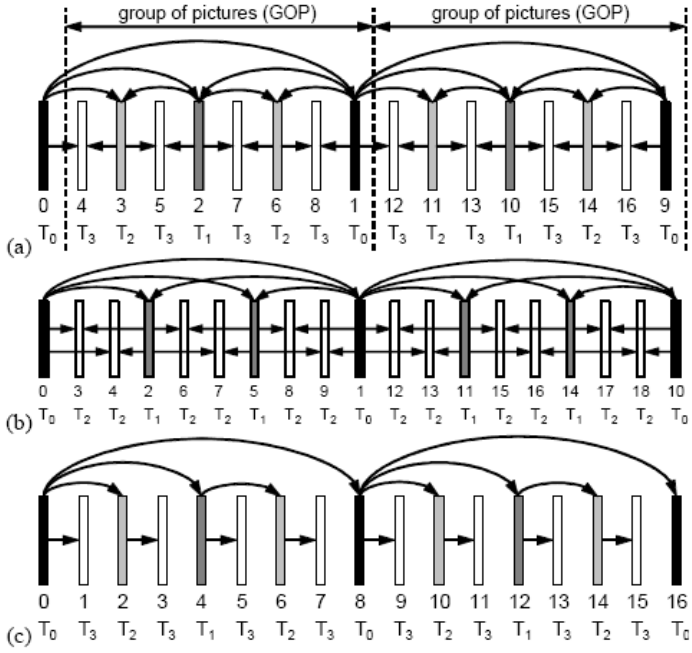

Fig. 3.2 Hierarchical prediction structures for enabling temporal scalability (a) Coding with hierarchical B-frames (b) Non-dyadic hierarchical prediction structure (c) Hierarchical prediction structure with structural encoder/decoder zero delay [8]

Actually all prior standards support temporal scalability to some degree. H.264/AVC provides significant flexibility in temporal scalability with its reference picture memory control. It allows coding of picture sequences with arbitrary temporal dependencies which are only restricted by decoded picture buffer (DPB) size. Therefore, to support temporal scalability no major changes in H.264/AVC are needed except signaling the temporal layers.

Temporal scalability with dyadic enhancement layers can be efficiently implemented with concept of hierarchical B or P pictures as shown in fig. 3.2(a). Enhancement layers are typically coded as B pictures with reference picture lists 0 and 1 corresponding to temporally preceding and succeeding picture respectively, with a temporal identifier less than that of the

picture being predicted. Since the backward prediction is not necessarily coupled with B frames, structure in fig. 3.2(a) can also be realized using P frames. Each set of temporal layers $\{T_0 \ldots T_k\}$ can be decoded independent of frames corresponding to $T > k$. Group of Pictures (GOP) corresponds to all the frames between two successive frames of temporal base layer (i.e. T_0) including the second T_0 frame.

To represent generalized non-dyadic case, hierarchical prediction structures for temporal scalability can be combined with 'multiple reference picture' concept in H.264/AVC, meaning that reference picture lists can be constructed by using more than one reference picture and they can also include frames with same temporal level as the one being predicted. Fig. 3.2(b) shows the non-dyadic case with two independently decodable sub-sequences at $1/9^{th}$ and $1/3^{rd}$ of full frame rate. Fig. 3.2(c) shows further case where it is possible to adjust encoder/decoder structural delay by restricting prediction from frames that follow the frame to be predicted in display order. Fig. 3.2(a) and 3.2(c) represent same temporal scalability but structural delay of 7 and 0 respectively. However low delay coding structures usually suffer from coding efficiency problems. In hierarchical prediction structures the reference frames should be coded before they can be used for prediction of other frames. Coding efficiency can be improved by carefully choosing quantization parameters for different temporal layers. Typically the base layer is coded with highest fidelity (or lowest quantization parameter) and quantization parameter is incremented for each subsequent temporal level. Further improvement in selection of quantization parameters can be achieved by computationally expensive rate-distortion analysis [33]. A simpler and sufficiently robust approach has been discussed in [34]. Coding efficiency of B-frames can be improved by using a weighted sum of list 0 and list 1 predictions is used during motion search [20]. It has been verified in [8] that coding efficiency of hierarchical temporal prediction structures can be improved by increasing GOP size and thus the encoding/decoding delay; the maximum coding efficiency is achieved for GOP sizes between 8 and 32.

When higher coding delay can be tolerated, hierarchical temporal prediction structure not only provides temporal scalability but also improves coding efficiency.

3.5.1.2 Spatial Scalability

Like previous scalable video coding standards, H.264/SVC follows multilayer spatial coding approach. Each layer corresponds to specific spatial resolution and is called as a 'spatial layer' with some dependency identifier *D*. Value of *D* is 0 for base layer, which has lowest spatial resolution and increments for each subsequent spatial layer. Each spatial layer is coded with most of the coding techniques used in single layer coding; like motion compensated prediction, intra prediction. In addition to this, to improve coding efficiency by exploiting correlation between adjacent layers, different '*Interlayer Prediction*' techniques are used.

Table 3.1 Video frames sampled at different display resolutions [10]

Format	Luminance Resolution (Horiz. x Verti.)	Chrominance Resolution (Horiz. x Verti.)
SQCIF	128 x 96	64 x 48
QCIF	176 x 144	128 x 96
CIF	352 x 288	176 x 144
4CIF	704 x 576	352 x 288

3.5.1.2.1 Interlayer Prediction

Spatial scalability is achieved by using an over sampled pyramid approach. For each layer and independent hierarchical motion compensation prediction structure with layer specific motion parameters is used.

As each higher layer is high resolution version of the previous layer, there exists redundancy in the information contents of consecutive layers. In order to improve coding efficiency of the enhancement layers in comparison to simulcast, various inter-layer prediction

mechanisms to re-use information from a lower spatial resolution to higher spatial resolution layer are specified.

These prediction mechanisms are made switchable so that the coder can liberally choose which base layer information should be exploited for an efficient enhancement layer coding. So the SVC conforming encoder can freely choose between intra and inter-layer prediction depending on the signal characteristics. Since the inter-layer prediction techniques employ methods for motion parameter and residual prediction, the temporal prediction structures of the layers should be aligned to maintain efficiency in enhancement layer coding.

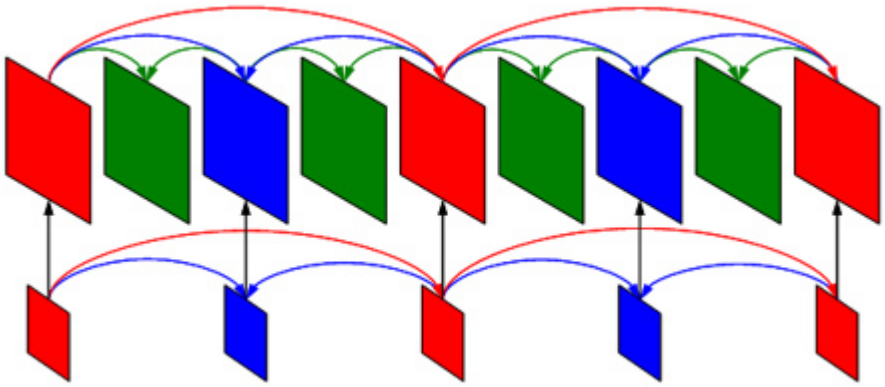

Fig. 3.3 Multi-layer structure with additional inter-layer prediction (black arrows) [35]

Following interlayer prediction techniques have been used in H.264/SVC coder design [36]:

- *Prediction of motion vectors* using the upsampled lower resolution motion vectors
- *Prediction of residual signal* using upsampled version of residual signal in the lower resolution layer
- *Prediction of macroblocks* using reconstructed and upsampled lower resolution signal

1. ***Motion Vector Prediction:*** Additional macroblock modes have been introduced in spatial enhancement layers to utilize motion information from low resolution layer. The macro block partitioning is obtained by upsampling partitioning of the corresponding 8x8

block of lower resolution layer. The reference picture indices are copied from co-located base layer blocks and corresponding motion vectors are scaled by factor of 2 in case of simple dyadic spatial scalability. For arbitrary resolution ratios reader is referred to [37] [38] [39]. For the first of these macro block modes no additional motion information is coded, for the second one, a quarter-sample refinement is transmitted for each motion vector. Additionally scaled motion vector of the lower resolution can be used as motion vector predictor.

2. ***Residual Prediction:*** In order to incorporate possibility of exploiting residual information coded in the lower resolution layer, an additional flag is transmitted for all inter-coded macroblocks that signals application of residual prediction from low resolution layer. If the flag is true the base layer residual signal is block-wise upsampled using a bi-linear filter with constant border extension and used as prediction for residual signal of present layer which is differentially coded.

3. ***Intra Prediction:*** An additional intra macro block mode has been introduced in which intra prediction signal is generated by upsampling co-located reconstruction signal of the lower layer using the 6-tap filter specified in H.264/AVC for half sample interpolation. The prediction residual is transmitted using H.264/AVC residual coding. This inter-layer prediction scheme is the only one that is supported in earlier video coding standards like MPEG-2/4 for spatial scalable coding.

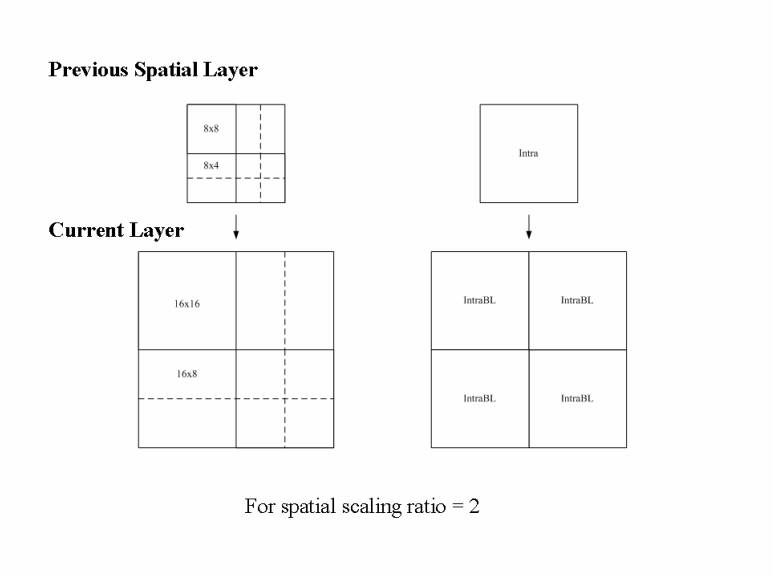

Fig. 3.4 Inheritance of modes [40]

There is also interlayer intra texture prediction. However it is not supported in the standard.

For the interlayer prediction using the reconstructed lower resolution signal it is generally required that the lower resolution layer is completely decoded including the computationally complex motion compensated prediction (or MCTF) and deblocking. It was shown in [41] that by restricting the prediction from upsampled decoded pictures to those parts of lower layer pictures which are intra coded the computational burden can be reduced with only small impact on coding efficiency for most of the test sequences. This facilitates decoding of each spatial layer with a single motion compensation loop also called as 'single loop decoding' which is computationally much less complex as compared to 'multiple loop decoding' that improves coding efficiency.

Similar to MPEG-2 video and MPEG-4 Visual, H.264/SVC supports generalized spatial scalability with arbitrary spatial resolution ratios. Only restriction is that neither horizontal nor vertical resolution should decrease from one layer to next. It also supports picture cropping.

3.5.1.3 Quality Scalability

Quality scalability can be considered as a special case of spatial scalability with identical frame sizes for base and enhancement layers. This is referred as Coarse Grain Scalability (CGS) and is supported by general spatial scalability case. Interlayer prediction techniques without upsampling operations are used. Usually the texture residue information is requantized using smaller quantization step size in enhancement layer than that used in base layer. As specific feature of this configuration, deblocking operation of base layer intra signal for interlayer intra prediction is omitted. Also, interlayer intra prediction and residue prediction are performed in transform domain to reduce complexity on decoder side.

However CGS allows only few selected bitrates in the coded scalable bit stream. In general, number of rate points supported is equal to number of layers. Switching between different CGS layers can be done only at defined points in the coded stream. Furthermore, CGS becomes less efficient when relative difference between succeeding CGS layers becomes smaller. Although CGS coding is simpler and provides low decoder complexity overhead as compared to single layer coding, it does not provide enough flexibility for all applications.

To increase flexibility of bitstream adaptation and error robustness in addition to improving coding efficiency of bitstreams that have to provide variety of bitrates, a variation of CGS coding scheme, called as Medium Grain Scalability (MGS) is introduced in SVC design. The differences with CGS concept are, modified high level signaling that allows switching between different MGS layers in any access unit, and so called ‘key picture’ concept, that allows suitable tradeoff between drift and enhancement layer coding efficiency for hierarchical prediction structures.

Earlier drafts of H.264/SVC had incorporated Fine Grain Scalability (FGS) in its design which is based on so called Progressive Refinement (PR) Slices. Each PR slice represents refinement of residual signal that corresponds to bisection of quantization size (or QP decrease of 6). These signals are represented in a way such that a single inverse transform has to be

performed at the decoder side. The ordering of transform coefficients in PR slices allows the corresponding PR refinement NAL units to be truncated at any arbitrary byte aligned point, so that the quality of the base layer can be refined in fine granular way.

The FGS method, though quite competitive in terms of R-D performance compared to single layer coding, has disadvantage of being computationally complex due to its related multipass entropy coding stage [42]. Therefore FGS was removed from the standard finalized in July 2007 [43] but it contained MGS as a low complexity version of FGS [44].

Similar to FGS, MGS operates in transform domain and allows fragmentation of a given fidelity enhancement by means of frequency selective grouping of transform coefficients. However the difference lies in re-using the bitstream syntax and entropy coding design of H.264/AVC to maximum extent. The degree of fragmentation can be chosen by the encoder without significantly compromising R-D performance.

It should be noted that a phase 2 of SVC project is under study which may contain FGS mode [45].

Drift is the effect of lost synchronization between motion compensation loops on encoder and decoder side because of unavailability of quality refinement packets due to some reason.

It can be handles using various methods, such as:

- *Base layer only control:* Fig. 3.5(a) Used in MPEG-4 Visual for Fine Grain Scalability (FGS). *Drawback*: Significantly decreases enhancement layer coding efficiency as compared to single layer coding.
- *Enhancement layer only control:* Fig. 3.5(b) Used in H.262/MPEG-2 video. *Drawback*: Any loss of quality refinement packets result in drift that can only be controlled by intra updates.

- *Two-loop control:* Fig. 3.5(c) *Drawback*: Although base layer is not influenced by enhancement layer packet loss, any loss of a quality refinement packet results in drift for enhancement layer.
- *SVC key picture concept:* Fig. 3.5(d) Used in H.264/SVC. For each picture, a flag is transmitted which indicates whether base quality reconstruction or the enhancement layer reconstruction of reference picture is used for motion compensated prediction. To limit decoded picture buffer (DPB) memory, a second syntax element signals whether base quality picture is additionally reconstructed and stored in DPB. To limit decoder overhead for such key pictures, the standard specifies that motion parameters between base and enhancement layers for key pictures must not change so that they can be decoded with single motion compensation loop.

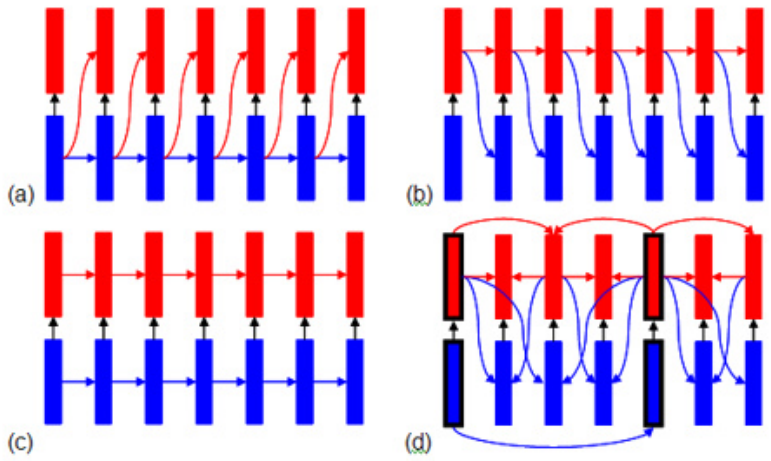

Fig. 3.5 Various drift control mechanisms for packet based quality scalable coding (a) base layer only control (b) enhancement layer only control (c) two-loop control (d) key-picture concept used in H.264/SVC (Key pictures are marked by black frames) [35]

With MGS, any enhancement layer NAL unit can be discarded from a quality scalable bitstream to form a valid sub stream and thus packet based quality scalable coding can be provided.

Additionally, H.264/SVC provides following features for quality scalable video coding [8]:

- Partitioning of transform coefficients
- SVC-to-AVC rewriting

3.5.2 NAL Unit Syntax

The 1 byte header in H.264/AVC is extended by additional 3 bytes for SVC NAL unit types. The extended header includes identifiers *D* (for spatial), *T* (for temporal) and *Q* (for quality) as well as additional information to assist easy bitstream manipulations. One such additional syntax is priority identifier *P* signaling importance of a NAL unit. Also, in order to attach SVC relayed information to non-SVC NAL unit, prefix-NAL units are introduced. SVC also specifies additional Supplementary Enhancement Information (SEI) messages, which contain information like spatial resolution or bitrate of layers included in coded scalable bitstream that can assist in bitstream adaptations.

3.5.3 SVC Profiles

SVC amendment supports following profiles for scalable video coding:

- *Scalable Baseline Profile:* Mainly targeted for mobile broadcast, conversational and surveillance applications that require low decoding complexity.
- *Scalable High Profile:* Designed for broadcast, storage and streaming applications.
- *Scalable High Intra Profile:* Mainly intended for professional applications.

3.6 Summary

In comparison to scalable video coding tools provided by prior standards, H.264/SVC provides various tools to improve coding efficiency relative to single layer coding. Important differences are:

- Possibility to employ hierarchical prediction structures for providing temporal scalability while improving coding efficiency and increasing efficiency in spatial and quality scalability.
- Methods to improve interlayer prediction of macro block modes, motion and residual that improves coding efficiency for spatial and quality scalability.
- Concept of key pictures to control drift for packet based quality scalable coding with hierarchical prediction structures.
- Low decoder complexity due to single motion compensated loop for decoding both, base and enhancement layers.
- Support for modified decoding process that allows lossless and low complexity re-writing of quality scalable bitstream into non-SVC H.264/AVC profile.

CHAPTER 4

SIP ANALYZER

4.1 Introduction

This chapter discusses Selective Inter Layer Prediction (SIP) scheme incorporated in scalable extension of H.264/AVC (H.264/SVC) that selectively disables some frames' inter layer prediction where it's not efficient, using some criterion.

SIP scheme was originally proposed by Kai Zhang (Institute of Computing Technology, Chinese Academy of Sciences) along with Jizheng Xu, Feng Wu (Microsoft Research Asia, Beijing, China) in 18th SVC Meeting held in Bangkok, Thailand (14-20 January 2006). It was submitted to JVT as proposal document JVT-R064.doc. It was originally a temporal level based selective interlayer prediction scheme which was later modified by them as a frame based selective interlayer prediction in 19th SVC Meeting held in Geneva, CH (31 March-7 April 2006). It was submitted to JVT as proposal document JVT-S051.doc.

Frame based selective interlayer prediction has been incorporated in JSVM version 5.9 and above as SIP Analyzer tool.

Following sections discuss it in more detail.

Some concepts:

Simulcast: *It is method of transmitting two or more independent single layer streams i.e. non-scalable streams coded at different resolutions, which in principle provide similar functionality as a scalable stream, although typically at the expense of increased overall bit rate.*

Multiple Adaptation Scenario (MA): *It is scenario in which the receiver of the scalable video stream needs all the resolution layers for its purpose.*

e.g. If the scalable video stream is coded with two layers viz. CIF and QCIF, the receiver end which may be a home gateway may need both of them probably to serve different resolution devices like CIF for a PDA screen and QCIF for a mobile screen. Therefore all the resolution layers need to be transmitted.

Without Multiple Adaptation Scenario (without MA): *If the receiver end needs only one particular resolution for its purpose e.g. 4CIF or CIF etc. then there is no need to transmit all the resolutions but just transmitting that particular resolution layer will be sufficient.*

e.g. a TV station broadcasting program for TV screens at 4CIF resolution

4.2 Selective Interlayer Prediction

As discussed in earlier chapter, in order to improve coding efficiency of the enhancement layers in comparison to simulcast, various inter-layer prediction mechanisms are incorporated in H.264/SVC that re-use information from a lower spatial resolution to higher spatial resolution layer.

There are numerous application scenarios where H.264/SVC video could be used. For example, a video server may save a program as an SVC file. It might be needed to serve two different end-devices with different capabilities i.e. a mobile with small display and limited power might need video with QCIF resolution and a PDA screen may need CIF resolution. The server will then use a bit stream extractor to extract appropriate resolution from the SVC file and transmit the substream to corresponding end device. This particular scalability will require combined scalability i.e. SNR and spatial scalability. Due to interlayer prediction, the decoding of high resolution layer depends on low resolution layer even though the end-device does not need the low resolution layer for display purpose.

This scenario can also be handled using simulcast i.e. encoding and storing two different copies of the same program independently, one with CIF resolution and the other with QCIF resolution. Appropriate stream is then transmitted. In this method only SNR scalability is used and there is no relation between high and low resolution streams.

Both of the approaches have advantages and disadvantages depending on the decoding scenario. In case of without MA scenario, where receiver knows beforehand, the resolution it is going to need for display purpose, method of simulcast performs better than combined scalability in the sense that for the same PSNR value simulcast achieves much lower bit rate. This is obvious as the bits are saved by not transmitting low resolution layer. So the combined scalable stream is not efficient in this scenario.

However if the receiver can not know beforehand which resolution it's going to need to serve the end-devices as mentioned in the previous example, using simulcast method will require saving both the resolutions. This is MA scenario. Experiments show that simulcast needs about 10% more storage than combined stream in MA scenario [46] because it does not exploit the correlation between consecutive layers. Again, this is not an efficient way as for the same PSNR simulcast will lead to considerably more bit rate than the combined scalable stream.

As mentioned earlier, the interlayer prediction in SVC exploits correlation between consecutive layers and saves bits while coding higher resolution layer. But the bits saved are not as many as bits used while coding the low resolution layer itself. This is the reason why JSVM performs worse under combined scalable scenario than under SNR scalable scenario.

In JSVM, not all the frames in the low resolution layer contribute similarly to the bit saving of high resolution frames. For those frames whose interlayer prediction is not so strong corresponding low layer frame bits also need to be transmitted although they contribute little to overall saving.

In [46] it was proposed to disable the inter prediction of those frames where it is not efficient using some criterion and was confirmed in [9]. Those low resolution layer frames are called 'Lazy Frames' which do little good for the high resolution layer in terms of prediction gain.

If MA is not required the bitstream extractor can just discard the packets corresponding to low layer frames. The decoding at receiving end will not be affected, as, in without MA scenario the receiving end does not need low resolution frames for display purpose. Hence bits of 'Lazy Frames' are saved. On the other hand, in MA scenario the increase in bit rate of high resolution frames by not using interlayer prediction for some low layer frames is not much because anyways those low resolution frames do only little good in terms of prediction gain. Thus selectively disabling interlayer prediction from low resolution frames is called ***Selective Inter Layer Prediction (SIP)*** scheme.

Now the problems remains as deciding which frames in the low resolution are the 'Lazy Frames'. In [46] authors proposed temporal level based SIP scheme. The proposition was disabling interlayer prediction on the low resolutions highest hierarchical B-level frames as their experiments showed that interlayer prediction of such frames is most likely to be inefficient. The extractor will drop packets corresponding to such frames if MA is not needed.

However this method has some shortcomings [9]:

- This method assumes that highest hierarchical B-frames are always the 'Lazy Frames' contributing little to the inter layer prediction. However this is not always true.
- This method can not guarantee the amount of bit-rate increase when MA is needed i.e. the loss is not under control.
- The method can not guarantee that it performance is the optimal one when MA is not needed.

4.3 SIP Decision Algorithm

To overcome these drawbacks authors proposed a new Frame based Selective Inter-layer prediction scheme in [9]. They have devised an algorithm to decide which frames are the 'Lazy Frames' for which we can safely turn off the interlayer prediction without uncontrolled increase in bitrate when MA is needed and also to get best performance when MA is not needed. The decision to selectively turn off interlayer prediction is called ***'SIP decision'***.

The assumptions made and verified by the authors are:

- Given a fixed QP, using or not using interlayer prediction does not affect high resolution frames quality i.e. PSNR value. It just helps by reducing bitrate.
- Given fixed QP, performance of one frame with interlayer prediction with its SIP decision is independent of other frames' SIP decision.
- It assumes single loop decoding scheme with which we can discard low resolution packet without worrying about its necessity for decoding other frames.

Under these assumptions the problem reduces to choosing the right combination of low layer frames which will reduce the output bitrate without MA as much as possible meanwhile avoiding output rate with MA increasing too much than specified. In other words, finding a binary vector (0 or 1) for all the low resolution frames where 0 means 'keep interlayer prediction' and 1 means 'turnoff the interlayer prediction'.

The SIP decision algorithm needs beforehand the acceptable bitrate increase in MA scenario. It may be specified as a percentage increase of original bitrate.

For two spatial layers, mathematically it is formulated as [9]:

Minimize,

$$f(X) = \sum_i [(R_i + r_i)(1 - x_i) + R'_i x_i] \qquad (4.1)$$

Subject to,

$$g(X) = \sum_i [(R_i + r'_i)(1 - x_i) + (R'_i + r'_i)\, x_i] \leq R_{max} \qquad (4.2)$$

Where

R_i , R'_i , r_i , r_i' , $R_{\max}$ >0, and they are all integer constants.

x_i is 0 or 1.

i is the index of frame i.

R_i is the output bits of the high resolution frame i if interlayer prediction is used.

R'_i is the output bits of the high resolution frame i if interlayer prediction is not used.

r_i is the output bits of the low resolution frame i when MA isn't needed.(maybe including bits of some lower layers' frames which are needed to decode frame i in the current low resolution layer)

r_i' is the sum of the output bits of frame i in all the low spatial layers.(including the current low resolution one)

x_i is the SIP decision of frame i. x_i =1 means frame i's interlayer prediction is cut off, and vice versa .

$R_{\max}$ is a given maximal output bits with multiple adaptation.

X is the SIP decision vector (x_i)

Let

$$R = \sum_i R_i$$

$$R' = \sum_i R'_i$$

$$r = \sum_i r_i$$

$$r' = \sum_i r_i'$$

let $R'_{\max} = R_{max} - R - r'$

let $\Delta_i = R'_i - R_i$

Solving it further we get,

Maximize

$$f(X) = \sum_i (r_i - \Delta_i) x_i \qquad (4.3)$$

Subject to

$$g(X) = \sum_i \Delta_i x_i \leq R'_{max} \qquad (4.4)$$

where,

Δ_i , r_i , $R'_{\max}$ >0, and they are all integer constants.

x_i is 0 or 1.

Here it is assumed that, $\Delta_i > 0 \quad for \quad all \quad i$, because if $\Delta_j \leq 0$ that means using interlayer prediction is worse than not using it. So x_j is set to 1 all the time.

It is also assumed that, $r_i - \Delta_i > 0 \quad for \quad all \quad i$. If $r_j < \Delta_j$ it means that frames interlayer prediction is so effective that its performance is better than the single layer coding. Here x_j is set to 0 all the time.

Equations (4.3) and (4.4) lead to a classical 0-1 Knapsack Problem which is a well known problem in combinatorial optimization. In this case, a frame on lower resolution layer can either be chosen ($x_i = 0$) or discarded ($x_i = 1$) for predicting corresponding higher resolution frame.

Solving this Knapsack Problem means finding such array x_i for each of the low resolution layers '*j*' involved in coding i.e. create matrix (x_i^j) that satisfies the constraints. Detailed derivation of this algorithm can be found in [9].

SIP Analyzer is the tool incorporated in JSVM (the reference software for SVC project) that implements the proposed Selective Interlayer Prediction strategy. Current implementation in JSVM code uses Dynamic programming approach to solve it.

4.4 SIP Codec Scheme

The overall coding process using SIP scheme is as follows [46]:

4.4.1 The encoder:

The encoder must encode the sequence three times. The application should mention in advance the allowed bit rate increase in with MA case e.g. It may be 3% more than original bitrate for layer 'i'.

First the encoder encodes using original configuration i.e. allowing interlayer prediction on all low resolution frames. It makes a record of bits allocated for each low and high resolution frame for the entire sequence with interlayer prediction.

Secondly it encodes the sequence by disabling interlayer prediction on all low resolution frames. It again makes record of bits allocated for all low and high resolution frames without interlayer prediction.

Then it feeds both the records i.e. bits allocated with and without interlayer prediction for all the frames in the sequence to the SIP Analyzer. Output of this algorithm is a matrix

(x_i^j) which specifies whether interlayer prediction for frame i on layer j should be turned off ((x_i^j) = 1) or it should be used ((x_i^j) =0).

Finally it encodes the sequence using this SIP decision matrix (x_i^j) to get the final bitstream.

4.4.2 The extractor

The bit stream extractor produces the bitstream as the application demands. If the application needs MA scenario extractor can be invoked in usual way thereby retaining frames on lower resolution. If the application demands without MA scenario, extractor can be invoked by specifying '-sip' option by discarding low resolution frames as specified by SIP decision matrix (x_i^j) to get desired resolution bitstream.

4.4.3 The decoder

No change is needed on decoder side except that it should support single loop decoding.

The syntax modification in H.264/SVC standard needed to support SIP Analyzer tool can be found in [9]. Actual implementation in JSVM can be found in [65].

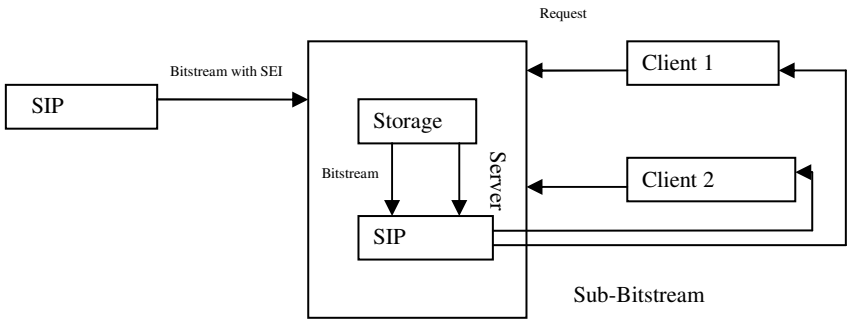

Fig. 4.1 SIP codec scheme [46]

CHAPTER 5

KNAPSACK PROBLEM

5.1 Introduction

The Knapsack Problem is a well known problem in combinatorial optimization [47]. Optimization problems are those problems in which the objective is *to find the best of all possible solutions* i.e. to find solution in the feasible region that will optimize the value of objective function.

Definition: There are '*n*' objects each having some value *'v'* and some weight *'w'*. Let i^{th} object be of value v_i and weight w_i. Let there be a knapsack which can carry objects with total weight no more than W_{max}. The objective is to select items amongst '*n*' items which will maximize total value and at the same time will not exceed the knapsack weight capacity.

Mathematically,

$$\textit{Maximize} \qquad \sum_i v_i$$

$$\textit{Under the constraint,} \quad \sum_i w_i \leq W_{max;} \text{ for i } \epsilon \ [1, n] \qquad (5.1)$$

Considering number of objects allowed selecting while filling the knapsack there are two variations of this problem.

1. *Bounded Knapsack Problem*: It puts a bound on the number of objects of each type that can be chosen while making the selection.

Mathematically,

$$\textit{Maximize} \qquad \sum_i x_i . v_i$$

$$\textit{Under the constraint,} \sum_i x_i . w_i \leq W_{max;} \text{ for i } \epsilon [1, n] \qquad (5.2)$$

where $0 \leq x_i \leq b_i$ *for some* $0 < b_i < \infty$

2. *Unbounded Knapsack Problem*: It puts no restriction on the number of objects of each type chosen while making the selection.

Considering type of objects used while filling the knapsack, we can have,

1. *The 0-1 Knapsack Problem*: It is special case of bounded Knapsack Problem with $b_i = 1$. So for every object we have choice of either choosing or discarding the object only once. If selected the object has to be taken in its entirety.

2. *The Fractional Knapsack Problem*: It is special case of Knapsack Problem with the provision that object can be selected in its fraction.

We will focus on 0-1 Knapsack Problem.

Some concepts:

P class problem: *A computational problem is in class P (the polynomial time) if there is some deterministic algorithm that solves the problem and runs in time* $O(n^k)$ *where 'k' is some integer [48]. These problems are generally considered to be feasible.*

NP class problem*: A computational problem is said to be in class NP (non-deterministic polynomial time) if there is some non-deterministic algorithm that can solve the problem in polynomial time. NP problem requires lucky guesses to work efficiently.*

NP-complete problem*: They are subset of NP problems. They are the most difficult problems in NP class in the sense that there in unavailability of any algorithm that can solve such problem in polynomial time. In other words, if a polynomial time solution exists for an NP-complete problem then that would provide solution to every other problem in NP class.*

Knapsack Problems are considered to be NP-complete or Pseudo-Polynomial time algorithms in computational complexity theory. Pseudo-polynomial time algorithms are those for which running time is polynomial in the numeric value of the input but is actually exponential with length of the input i.e. number of digits in representation of the number. So pseudo polynomial time algorithms are impractical for large values of input sequences.

5.2 Solution to the Knapsack Problem

As Knapsack Problem is supposed to be an NP-complete problem for which finding an exact solution for a large set of input is nearly impossible in practice. There are several ways of finding optimal/nearly optimal solution to the Knapsack Problem. Some of them are:

1. *Brute-force approach:* It is the most straightforward solution. Since there are '*n*' items we can have '2^n' possible combination of these items. We have to go through all combinations and find the one with maximum total value and total weight not more than W_{max}. However it is inefficient and impractical for moderate and large length of input sequences. Complexity of this algorithm is $O(n.2^n)$.

2. *Dynamic Programming:* It takes much less time than brute-force approach to reach to the optimal solution provided that the problem exhibits properties of optimal substructure and overlapping subproblems which are discussed later in this chapter.
3. *Greedy Choice Solution:* If the problems exhibit optimal substructure they can be solved by heuristic methods. They are much more efficient than dynamic programming in terms of computational complexity. Greedy approach can give optimal solution in fractional Knapsack Problems however they do not guarantee optimal solution to the 0-1 Knapsack Problems [49]. They may give good solution for practical purposes.
4. *Memory Functions:* Memory functions use the same recurrence relation as the dynamic programming approach. However they use top-down approach only to the subproblems that are necessary unlike Dynamic programming. More information can be found in [50] [51].
5. *Branch-and-Bound:* It is a generic algorithm used especially in discrete and combinatorial optimization problems. It is an improvement over exhaustive search in the sense that it generates candidate solutions one component at a time and evaluates these partly constructed solutions by using lower and upper estimated bounds of quantity being optimized. If discards those solution which are not going to lead to fruitful solution by not generating other components.
6. *Genetic Algorithm:* It is a search technique used to find optimal/sub-optimal solution to search and optimization problems. It has been inspired by Darwin's theory of evolution. Algorithm is started with set of possible solutions (called population). Solution from one population is chosen to form another population (called offspring) hoping that new population will lead to a better solution. This selection is done according to their fitness. This is repeated until some condition is satisfied [52].

Numerous papers have been written about various approaches towards finding optimal/nearly optimal solutions to generic/special cases of Knapsack Problem in the fields like business, operations research, cryptography, applied mathematics and many more. However study [53] shows that Dynamic programming remains the popular choice when it comes to finding optimal solution to a generic Knapsack Problem.

5.2.1 Dynamic Programming

Dynamic Programming is an algorithmic technique which is used to efficiently solve wide range of optimization and search problems which exhibit properties of overlapping subproblems and optimal substructure which are discussed later. This technique was originally used by Richard Bellman in the 1940s which was later modified several times.

Since the Knapsack Problem exhibits both of these properties Dynamic programming is a good candidate for solving it.

More specifically Dynamic programming makes use of following properties:

- Overlapping subproblems
- Optimal substructure
- Memoization

Some concepts:

Overlapping subproblems: *A problem is said to have overlapping subproblems if main problem can be divided into number of small problems and solution to each of them can be used then several times.*

e.g. Fibonacci series: The generalized equation for Fibonacci series is, F(n) = F(n-1) + F(n-2), where F(n-1) = F(n-2) + F(n-3) which means, F(n) = F(n-2) + F(n-3) + F(n-2). Thus F(n-2) is getting reused. So it is exhibiting overlapping subproblems property.

Optimal substructure: *Optimal substructure means optimal solution to a problem lies in the optimal solution to the subproblems that the main problem can be broken into.*

e.g. Shortest path algorithm: The shortest path from one node to another should contain shortest path from its nearest neighboring node to the other node. If this is not true then we will have shorter path than the shortest path which is a contradiction! Thus problem of finding shortest path exhibits optimal substructure property.

Memoization: *It is an optimization technique usually used in recursive algorithms by lowering functions time cost in exchange of functions space cost. In other words it speeds up the recursive program execution by avoiding repetitively calculating results needed to calculate future results by storing them in a lookup table on an as needed basis. They can be used directly while calculating future results.*

Typical way of solving problems using Dynamic programming is to break the main problem into subproblems; finding solutions to the subproblems by solving them and memorizing (storing) their results in case they need to be solved again and then trace back towards the solution to main problem by combining the solutions to the subproblems. This approach involves recursion and memorization combined together.

We will now consider Dynamic programming with reference to solving 0-1 Knapsack Problem.

To recall: There are '*n*' objects each having some value *'v'* and some weight *'w'*. Let i^{th} object be of value v_i and weight w_i. Let there be a knapsack which can carry objects with total weight no more than W_{max}. The objective is to choose combination of items amongst '*n*' items by choosing/discarding each item only once, which will maximize total value and at the same time will not exceed the knapsack weight capacity.

We define *A(i,j)* as the maximum value that can be attained by considering first 'i' items that can weigh no more than 'j' units [54].

A(0,j) = 0 and *A(i,0)* = 0 for any i ≤ N and j ≤ W_{max} which is obvious.

If w_i > j then *A(i,j) = A(i-1, j)* because we can not include item 'i'.

If $w_i \leq$ j then we have choice of including i^{th} item or excluding it. If we include it then total value will be v_i + *A(i-1, j- w_i)* whereas if we exclude it total value will become *A(i-1, j).* The choice of whether to include or exclude depends on what is the maximum value amongst both of them.

Mathematically we can put it in following recursive expression.

$$A(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ A(i-1,j) & \text{if } w_i > j \\ \max\{A(i-1,j), v_i + A(i-1, j-w_i)\} & \text{if } w_i \leq j \end{cases}$$

(5.3)

Here the subproblems overlap because at any stage *(i,j)* we may need to calculate *A(k, l)* for k < i and l < j. We have optimal substructure because at any point we need only the information about choices that we have already made.

Since we need to calculate *A(n, W_{max})* as the solution to the Knapsack Problem. While doing so, we will need to create an '*n*' by 'W_{max}' table with *A(i,j)* as entry at location *(i,j).* Calculating and storing *A(i,j)* values is Memoization which is used to exploit overlapping subproblems.

Then we iterate over all i ≤ *n* and j ≤ W_{max} by which we are basically exploiting optimal substructure. While using the recursive formula we do not recalculate previous entries but use table lookup instead. Out desired result is stored as (n, W_{max}) th entry in the table.

Pseudo code for Dynamic programming with reference to 0-1 Knapsack Problem would be:

Function dynamic (v[], w[],W_{max})

for w = 0 to W_{max}

do A[0, w] = 0

for i = 1 to n

do A[i, 0] = 0

for w = 1 to W_{max}

do if $w_i \leq w$

then if v_i + A[i-1, w-w_i]

then A[i, w] = v_i + A[i-1, w-w_i]

else A[i, w] = A[i-1, w]

else

A[i, w] = A[i-1, w]

Computational complexity of this algorithm is *O(n. W_{max})* in time and *O(n. W_{max})* in space. It should be noted that this is *not* a polynomial time solution to an NP-complete problem.

Everyday examples of Dynamic programming include:

- Viterbi algorithm used in Digital communication in connection with Hidden Markov Models.
- The Needleman-Wunsch algorithm used in Bioinformatics.
- Floyd's shortest path algorithm

- Duckworth-Lewis method used in sport of Cricket.
- Finding string-edit distance between two strings in spellcheckers.

5.2.2 Other Approaches

Presently all known algorithms that give optimal/nearly optimal solution to Knapsack Problem require time that is superpolynomial in the input size [55]. Therefore to solve such problems in real time where one is limited with computational resources and also can be satisfied with some solution that works well for all practical purposes we can use sub-optimal algorithms to solve NP-complete problems.

For most of the practical purposes other techniques like *Approximation*, *Randomization*, *Restriction*, *Parameterization* or *Heuristic* can be used. They provide reasonably good solution and significantly faster processing speed.

Most popular are *Approximation* and *Heuristic* methods. If one wants provably good solution within some tolerance limits of the optimal solution and known runtime bounds then approximation algorithms can help. But they consume considerably more time and space than heuristic methods.

For example, Ibarra and Kim [56] provide one such approximation algorithm where relative error is guaranteed to be at most E with respect to the optimal solution. However time and space complexity of this algorithm is $O(n/ E^2)$ and hence polynomial in n and 1/ E. Proof can be found in [57].

Heuristic methods like Greedy provide practically good solution considerably fast. They are simple, straightforward and require minimal amount of resources. They are easy to invent, implement and efficient for most of the times. Even though optimality of this solution can not be always guaranteed, they remain popular choice when it comes to practical implementation [58].

5.2.3 Greedy Heuristic Algorithm

Greedy heuristic algorithms are the most simple and straightforward in their approach. They are shortsighted in the sense that they work by making decision that seems most promising at the moment on the basis of information at hand without considering possible side effects and never reconsiders the decision once made. They make locally optimum choice at each stage in the hope of finding global optimum which they may or may not obtain eventually.

Greedy heuristic works in top-down manner by making one greedy choice after another reducing the given problem to smaller ones unlike Dynamic programming which solves the subproblems bottom-up and then combine the solutions to an optimal one. Even though exhaustive, Dynamic programming is guaranteed to find the optimal solution.

Characteristics of problems for which Greedy methods can be applied are:

- *Greedy Choice Property:* It means that globally optimum solution can be arrived at by making a locally optimal (greedy) choice. It iteratively makes greedy choices thereby reducing main problem to smaller ones and never reconsiders the decision once made.
- *Optimal substructure:* Optimal substructure means optimal solution to a problem lies in the optimal solution to the subproblems that the main problem can be broken into.

Typically if the problem exhibits optimal substructure property Greedy approach can be applied to find the solution. In addition if the problem also exhibits overlapping subproblems property Dynamic Programming can be a good choice, whereas if a problem does not exhibit any of these cases, then brute force approach can give optimal solution.

Typical structure of Greedy algorithm is [59]:

- Initially the set of chosen items is empty i.e. solution set
- At each step

- Items will be added to the solution set by *selection function*
- IF the solution set would no longer be feasible i.e. it can not be extended to produce optimal solution
    - Reject corresponding items and never consider them again
- ELSE IF solution set is feasible THEN
    - Add the corresponding items

We will now consider Greedy heuristic algorithm with reference to solving 0-1 Knapsack Problem.

Possible *selection functions* for making greedy choice are:

- Choosing item with maximum value from remaining items that will increase total value of knapsack quickly
- Choosing lightest item from remaining items which fills up capacity slowly thereby allowing more items to be stuffed and possibly maximizing total value
- Choosing item with highest value per unit weight at each stage from remaining items

It has been verified [53] that the third strategy i.e. choosing item with highest value per unit weight at each stage gives best results.

Using this selection function and using maximum allowed weight as the upper bound on feasibility we can write following pseudo code for the Greedy algorithm to solve 0-1 Knapsack Problem:

Function greedy (v[], w[],W_{max})

$W_{total} = 0$

for i = 0 to N-1

$p_i = v_i/w_i$

for i = 0 to N-1

sort p_i in descending order and put result in s_i. $A[s_i]$ retrieves item with cost per unit weight s_i.

for i = 0 to N-1

if $(w_i + W_{total}) > W_{max}$

discard $A[s_i]$

else

consider $A[s_i]$

$W_{total} = W_{total} + w_i$

Complexity of this algorithm is *O(N.logN)* to sort the elements plus *O(N)* to pick up the elements i.e. *O(N.logN) + O(N)* ≈ *O(N.logN)* in time. Space needed is just a single dimensional array of size 'N', which is considerably less than Dynamic programming approach. Also, this is a polynomial time algorithm which can be implemented in real-time systems.

As we can see, once the Greedy selection algorithm makes a choice to put an element in the knapsack it never reconsiders this decision later. All it is left with is the set of remaining items to make future decisions. Disadvantage of this approach is that if our selection function is not optimal it introduces error in one step and goes on accumulating it further. As a result the final set of items which Greedy heuristic has chosen to be placed in the knapsack may not be optimal. So the knapsack may not be filled completely to its capacity or if filled may not have maximum possible total value of items.

It should be noted that Greedy heuristic algorithm can give optimal solution to fractional Knapsack Problems but not to 0-1 Knapsack Problem [49]. Thus there is a tradeoff between optimality of the output versus complexity of the Greedy heuristic algorithm for 0-1 Knapsack Problems.

However Greedy approach is much faster than any other approach and the results produced by Greedy approach are good approximations to the optimal solutions [60] when

tested for large set of data. Therefore Greedy heuristic remains the popular choice when one does not need *provably good* solution but *just good* solution is acceptable.

5.3 Example

We will illustrate each of the two methods with a simple example for a 0-1 Knapsack Problem.

Problem:

Let there be 3 items and a knapsack with maximum weight allowed = 5 units

Table 5.1 Knapsack Problem: Input data

Item i	***Value v_i***	***Weight w_i***
1	2	1
2	3	2
3	4	3

Choose items using 0-1 Knapsack Problem criterion that will total maximum value without exceeding maximum weight allowed.

Approach 1: Dynamic programming solution [61]

Create a 2-D array of size 4x6 (i.e. (N+1) x (W_{max}+1)). It initially looks like,

Table 5.2 Dynamic Programming: filling table values

$i\backslash w$	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						

Then using the pseudo-code for dynamic programming we can fill up the entries in the table. After filling up all the entries the table looks like:

Table 5.3 Dynamic Programming: tracing back the table

$i \backslash w$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	<u>0</u>	2	2	2	2	2
2	0	2	<u>3</u>	4	4	4
3	0	2	3	4	6	<u>7</u>

Once we have filled up all the entries we need to trace back the table for finding entries that lead to optimal solution. If *T(i,w)* are the items that produce solution *V(i,w)* then,

T(i,w) = T(i-1, w) ; if V(i,w) = V(i-1, w)

= { i } U T(i-1, w-w_i) ; otherwise

Using this criterion we get items 2 and 3 as desired.

<u>Approach 2:</u> Greedy Heuristic Solution

We choose selection function as the one with highest value to weight ratio. First we calculate $p_i = v_i/w_i$ values and sort them as,

Table 5.4 Greedy Approach: sorting values

Item i	***Value/Weight $p_i = v_i/w_i$***
1	2
2	1.5
3	1.33

We have $W_{max} = 5$

Set $W_{total} = 0$

Consider sorted item 1: $W_{total} = W_{total} + w_i = 0 + 1 = 1$

$W_{total} \leq W_{max}$ so select item 1

Consider item 2: $W_{total} = W_{total} + w_i = 1 + 2 = 3$

$W_{total} \leq W_{max}$ so select item 2

Consider item 2: $W_{total} = W_{total} + w_i = 3 + 3 = 6$

$W_{total} > W_{max}$ so discard item 3

Therefore items selected are 1 and 2 (not an optimal choice) and their total value is 6. As we can see here, solution given by Greedy heuristic completely depends on the selection function used while making the choice. There can be several ways of choosing an item.

However as discussed in section 5.2.3 results given by using highest value to weight ratio criterion approach optimal solution over large set of data.

5.4 Summary

Knapsack Problem is classical problem in combinatorial optimization. It is an NP-complete problem for which there lacks any algorithm that can give polynomial time solution.

Dynamic Programming if the most popular method when optimality of solution is concerned. It gives optimal solution to 0-1 Knapsack Problem although it is not polynomial time algorithm.

Greedy remains popular choice when it comes to implementations. It is polynomial time algorithm however optimality of such solution can not be always guaranteed. Greedy heuristic

approach *works well* for large set of data and its solution closely matches to optimal solution for most of the practical purposes.

CHAPTER 6

CONTRIBUTION OF THE THESIS

6.1 Background

As discussed in chapter 4, Knapsack Problem is the core of the SIP decision algorithm or the SIP Analyzer tool which decides the right combination of lower resolution frames for which inter layer prediction can be safely turned off in order to produce a bitstream which can save bitrate in without MA scenario and also not exceed bitrate than restricted limits in MA scenario.

As seen in chapter 5, Knapsack Problem is an NP-complete problem for which there lacks any algorithm which can give optimal solution in polynomial time. Dynamic programming gives optimal solution to Knapsack Problem and hence current implementation of SIP Analyzer uses Dynamic Programming approach to solve the Knapsack Problem in SIP decision algorithm although it is not polynomial time algorithm.

Complexity of Dynamic programming in SIP Analyzer is *O(n. R'max)* in time and also in space. [Ref: section 4.3 to get description of terms] where '*n*' is total number of frames in lower resolution layer and $R'_{\max} = R_{max} - R - r'$ as discussed in section 4.3. As $R'_{\max}$ depends entirely on the bitrate at which the sequence is being encoded, there is no upper limit on the value. It depends solely on the application. In fact it would be sensible to use SIP strategy for high bit rate coding systems where it is useful to have any saving in the bitrate because the bandwidth might already have become bottleneck. However the factor $R'_{\max}$ will also increase as the encoding bitrate increase. Also we can not make any assumption about '*n*'. Therefore complexity of the algorithm increases when the algorithm is most needed.

Keeping these issues in mind, complexity $O(n. R'_{max})$ in time and in space is not feasible for real time systems where processing time and power have constraints. Therefore even though Dynamic programming solves the Knapsack Problem in SIP Analyzer optimally, it is not practical way of doing it, as '*n' or 'R'max'* grows in size.

As already discussed in chapter 5, Greedy heuristic remains the most popular choice to solve Knapsack Problems when it comes to real time implementations.

One good example of this is the Greedy Merge algorithm used in MPEG-4 AAC encoder implementations to perform optimal codebook search in Huffman coding where one has to find the right combination of codebook vectors that will code the frame with least possible bits under given rate-distortion constraint. This is also a 0-1 Knapsack Problem. Even though Greedy Merge may not always yield the optimal solution to the problem, it produces fairly good solution for practical purposes with considerably lower resource consumption. That is the reason it has been used in mostly all the industry implementations of MPEG-4 AAC [62] encoder.

6.2 Proposal

In this thesis, we propose applying Greedy heuristic approach to solve Knapsack Problem in SIP Analyzer in H.264/SVC and evaluate its performance both qualitatively and quantitatively.

Since Greedy heuristic does not yield optimal solution to 0-1 Knapsack Problem, by applying it to SIP Analyzer we may not get the right combination of lower layer frames for which interlayer prediction can be turned off in without MA scenario to save the bitrate. There may be mismatch between the optimal solution given by existing Dynamic programming based algorithm and the one given by proposed Greedy based approach. However motivation for doing it lies in the reduced computational complexity while making SIP decision.

We have modified the SIP Analyzer module in JSVM code (version 5.10) to incorporate Greedy heuristic approach to solve the Knapsack Problem instead of existing Dynamic Programming based approach.

Since mathematically we can not prove optimality of solution provided by Greedy heuristic approach, we have tried to judge its qualitative performance by conducting large number of tests.

We conducted tests for all possible scalability scenarios with the standard test vectors specified by JVT for H.264/SVC testing and also with some more standard test cases used in MPEG and H.264 world. The results are promising.

Chapter 7 discusses the results in more detail, however in essence we can say that Greedy approach works very well in terms of computational complexity reduction whereas in terms of optimality of the solution, the deviation from bitrates from optimal solution in with and without MA scenarios is considerably small.

6.3 Implementation

Proposed code for Greedy heuristic approach to solve Knapsack Problem in SIP Analyzer module in JSVM code [65] is as follows:

```
// SSS Solve the Knapsack Problem using Greedy heuristic
// Original routine from [65] is modified here

ErrVal SIPAnalyser::xProcessKnapsack(int iNumber,
                                     int *piWeight,
                                     int *piPrice,
                                     int iBagCubage,
                                     int *piDecision)
{

       int i,j;

       // Create a 2-D array ppiM[value/weight][index]
       // and fill in the values

       double **ppfM;
       ROF(ppfM=new double* [iNumber+1]); // One extra space
```

```
                                                // to avoid exception
        for(i=0;i<=iNumber;i++)
        {
                ppfM[i]=new double[2];
                ROF(ppfM[i]);

                ppfM[i][0] = double(piPrice[i])/double(piWeight[i]);
                ppfM[i][1] = (int)i;
        }

        // Sort the array in descending order
        RNOK(xQuickSort(&ppfM[0], 0, iNumber-1));

        // Call Greedy
        RNOK(xGreedy(iNumber, iBagCubage, piWeight, ppfM, piDecision));

        // Free memory
        for(i=0;i<=iNumber;i++)
                delete [] ppfM[i];
         delete [] ppfM

        return Err::m_nOK;
}


// Greedy Heuristic approach to solve the 0-1 Knapsack Problem using
// largest value to weight ratio criterion

ErrVal SIPAnalyser::xGreedy(int iItemsLeft,
                            int iBagCubage,
                            int *piWeight,
                            double **ppfM,
                            int *piDecision)
{

        // Do until no items left
        if(iItemsLeft <= 0)
                return Err::m_nOK;


        // Pick up the item if it fits
        if(piWeight[(int)ppfM[0][1]] < iBagCubage)
        {
                iBagCubage-=piWeight[(int)ppfM[0][1]];
                piDecision[(int)ppfM[0][1]]=1;
        }
        // Discard if it does not fit
        else
                piDecision[(int)ppfM[0][1]]=0;

        ppfM+=1;

        // Do it recursively for remaining items
        RNOK(xGreedy(iItemsLeft-1, iBagCubage, piWeight, ppfM, piDecision));

        return Err::m_nOK;
}
```

```
// Sort the array in descending order using QuickSort [63][64]

ErrVal SIPAnalyser::xQuickSort(double** ppfM,
                               int iLower,
                               int iHigher)
{
       //  iLower is the lower index and iHigher is the upper index
       //  of the array to be sorted

       int i = iLower, j = iHigher;
       double x = ppfM[(iLower+iHigher)/2][0],temp1, temp2;

       //  Perform partitioning
       do
       {
              while (ppfM[i][0] > x)
                     i++;
              while (ppfM[j][0] < x)
                     j--;

              if (i<=j)
              {
                     temp1 = ppfM[i][0];
                     temp2 = ppfM[i][1];
                     ppfM[i][0] = ppfM[j][0];
                     ppfM[i][1] = ppfM[j][1];
                     ppfM[j][0] = temp1;
                     ppfM[j][1] = temp2;
                     j--;
                     i++;
              }
       } while (i<=j);

       //  Sort both the parts recursively
       if (iLower < j)
              xQuickSort(ppfM, iLower, j);
       if (i < iHigher)
              xQuickSort(ppfM, i, iHigher);

       return Err::m_nOK;
}
```

CHAPTER 7

RESULTS AND CONCLUSION

As discussed in previous chapters, the optimality of the solution given by Greedy Heuristic approach can not be mathematically proven. Therefore we conducted sufficiently large number of tests with possible scenarios where SIP Analyzer could be used.

The main aim while conducting these tests was, to show quantitative performance improvement by measuring computing time and memory consumption for each algorithm.

To measure qualitative performance of Greedy approach, we can either show change in bitrate at given PSNR value or show change in PSNR value at given bitrate. We have chosen the later option because it is easy to extract a subsequence at target bitrate using bitstream extractor provided by JSVM [65]. However it should be mentioned that using SIP decision does not affect coded frame's quality. It just helps reducing bitrate.

While presenting the results, we have compared decisions given both the algorithms (and corresponding decision error for Greedy approach) and the effect of that decision error in terms of PSNR degradation. Both the errors are measured for MA and without MA scenarios.

7.1 Test Scenario

The test sequences used include standard test cases suggested by JVT [66]. They include – Bus, City, Crew, Football, Foreman, Harbour, Mobile and Soccer.

In addition we also used some more popular test vectors used in H.264 and MPEG World [67] [68] [69]. They are – Stefan, Deadline, Suzie, Miss-America, Flower, Tempete, Waterfall, Students, MaD9003 (Mother and Daughter), Paris, Pamphlet, News, Silent, Akiyo,

Coastguard, Container, Hall, Bowing, Carphone, Claire, Grandma, Bridge-Close (Bridge Close View).

All test sequences have CIF@30, QCIF@15 resolutions at 4:2:0 sampling. In addition, City, Crew, Harbour and Soccer also have 4CIF@60 resolution. The minimum frame size is 90 and the maximum is 2000 for the sequences used.

The JSVM version used is 5.10 (just next version after actual SIP Analyzer was integrated in JSVM) mainly because we found it more stable than any other later version as far as SIP Analyzer is concerned. Other aspect is: our experiment is mainly concerned with evaluating performance of Greedy heuristic with respect to Dynamic Programming based approach in SIP Analyzer. Later JSVM versions have mainly improved the VCL components. Therefore applying same Greedy based approach to SIP Analyzer in any later version should give equivalent results except minor differences in PSNR values.

All the tests were conducted on Dell Dimension DM061 machine with Intel ® Core ™ 2 CPU @ 2.13 GHz, 3070 MB RAM and 32-bit Windows ® Vista ™ Home Premium Edition Operating system.

For profiling the routines, we used `GetSystemTimeAsFileTime()` function for Windows ® which gives execution time with accuracy in milli-seconds.

Possible application scenarios for SIP based codec use Spatial and Combined Scalability. Therefore we have tested our algorithm for these two scalability modes. For all test scenarios, allowed bitrate increase in 'with MA' scenario is restricted to 3% of original bitrate.

For all scenarios, the error is calculated as:

$$\%\ Decision_{error} = \frac{(Decision_{Dynamic} - Decision_{Greedy})}{Decision_{Dynamic}} \ x\ 100 \qquad (7.1)$$

$$\% PSNR_{error} = \frac{(PSNR_{Dynamic} - PSNR_{Greedy})}{PSNR_{Dynamic}} \times 100 \quad (7.2)$$

where, $PSNR_{error}$ is the error caused in PSNR value of frame decoded at given bitrate, corresponding to error in decision because of sub-optimal Greedy approach.

7.2 Results

While presenting the results, error values are averaged out after measuring errors for different quantization parameters. However we would like to mention that in most of the test cases, maximum error never exceeded 0.1%.

The results for possible scalable modes are as follows:

7.2.1 Spatial Scalability

Each sequence was coded at three different Quantization Parameter (QP) values with two spatial layers. They are (28, 30), (30, 32), (32, 36) in (QP_{base}, $QP_{enhanced}$) format. Error shown is calculated by averaging errors at all three points. Some of the encoder settings include – GOP Size 32, AVC Compatible Base layer mode, Loop filter on (idc = 0), Motion search with FastSearch (mode = 4), NumFGSLayers = 1.

Following are results for QCIF15 and CIF30 combination: (The error is measured for enhanced layer i.e. CIF30 layer in this case)

Table 7.1 Results for spatial scalability (Enhancement layer resolution CIF30)

Sequence	# of frames	% Error in SIP Decision		% Error in PSNR		Avg Processing Time (u –sec)		Avg Memory Saving **
		with MA	wo MA	with MA	wo MA	Dynamic	Greedy *	
stefan	90	0.0307	-0.0985	-0.0111	-0.0363	146666.6667	0.0000	3976.6667
bus	150	0.0341	-0.0940	-0.0024	-0.0049	354666.6667	0.0000	1797.8333
deadline_1	150	0.0191	-0.0366	-0.0062	-0.0064	110000.0000	0.0000	591.8333
deadline_2	150	0.0470	-0.2084	0.0703	-0.0579	86000.0000	0.0000	3473.0000
deadline_3	150	0.0000	0.0000	0.0000	0.0000	80000.0000	0.0000	3184.3333
deadline_4	150	0.0569	-0.1854	-0.0191	-0.0216	113000.0000	0.0000	2954.5000
deadline_5	150	0.0072	-0.0172	-0.0017	-0.0118	120666.6667	0.0000	474.5000
deadline_6	150	0.0064	-0.0176	-0.0059	-0.0051	138000.0000	0.0000	1025.5000
deadline_7	150	0.0028	-0.0166	-0.0039	-0.0040	127000.0000	0.0000	710.6667
suzie	150	0.0129	-0.0103	0.0000	0.0034	52000.0000	0.0000	648.1667
miss-america	150	0.0170	-0.0164	-0.0105	-0.0052	41666.6667	0.0000	24.1667
flower	250	0.0043	-0.0066	-0.0001	0.0012	516333.3333	333.3333	20540.1667
tempete	260	0.0097	-0.0283	-0.0020	0.0023	605333.3333	333.3333	660.6667
waterfall	260	0.7602	-0.4201	0.0016	-0.0200	328000.0000	333.3333	3717.3333
football	260	0.0130	-0.0184	0.0007	0.0003	556666.6667	333.3333	2654.3333
students1	300	0.0006	-0.0023	-0.0010	-0.0002	351666.6667	0.0000	237.8333
students2	300	0.2889	-0.3879	0.0047	-0.0098	242000.0000	0.0000	1491.3333
MaD9003	300	0.0026	-0.0093	0.0000	0.0009	342000.0000	0.0000	79.8333
paris	300	0.0051	-0.0077	0.0006	0.0019	664000.0000	0.0000	2070.1667
foreman	300	0.0024	-0.0171	0.0188	-0.0102	511000.0000	0.0000	188.5000
mobile	300	0.0378	-0.0341	0.0113	-0.0339	630666.6667	666.6667	21992.8333
soccer	300	0.0114	-0.0218	0.0000	0.0026	682333.3333	333.3333	501.1667
harbour	300	0.0058	-0.0221	-0.0031	0.0001	585000.0000	0.0000	286.1667
city	300	0.0000	0.0000	0.0000	0.0000	520000.0000	333.3333	19274.3333
crew	300	0.0015	-0.0144	0.0120	0.0016	788333.3333	333.3333	47.1667
pamphlet	300	0.0018	-0.0688	0.0053	-0.0010	165666.6667	333.3333	342.8333
news	300	0.0084	-0.0260	-0.0028	-0.0026	410666.6667	0.0000	223.8333
silent	300	0.0028	-0.0070	0.0003	-0.0003	490000.0000	0.0000	213.3333
akiyo	300	0.0056	-0.0124	-0.0056	-0.0016	111000.0000	0.0000	145.5000
coastguard	300	0.0060	-0.0136	-0.0028	-0.0036	769000.0000	333.3333	354.1667
container	300	0.0007	-0.0197	0.0649	-0.0259	333333.3333	0.0000	2797.6667
hall	300	0.0147	-0.0003	-0.0028	-0.0039	470000.0000	333.3333	2685.3333
bowing	300	0.0020	-0.0169	0.0467	0.0341	145666.6667	0.0000	157.6667
carphone	382	0.0112	-0.0067	-0.0032	-0.0003	747666.6667	0.0000	18.5000
claire	494	0.0017	-0.0041	-0.0007	-0.0002	373000.0000	333.3333	239.0000
grandma	870	0.0009	-0.0615	0.0469	0.0594	1308000.0000	1000.0000	53.0000
bridge-close	2000	0.0000	0.0000	0.0000	0.0000	3746000.0000	1000.0000	38367.6667

** zero processing time for Greedy means actual processing time is less than 1 m-sec*

*** Average memory saving = (Bytes needed by Dynamic Programming) / (Bytes needed by Greedy)*

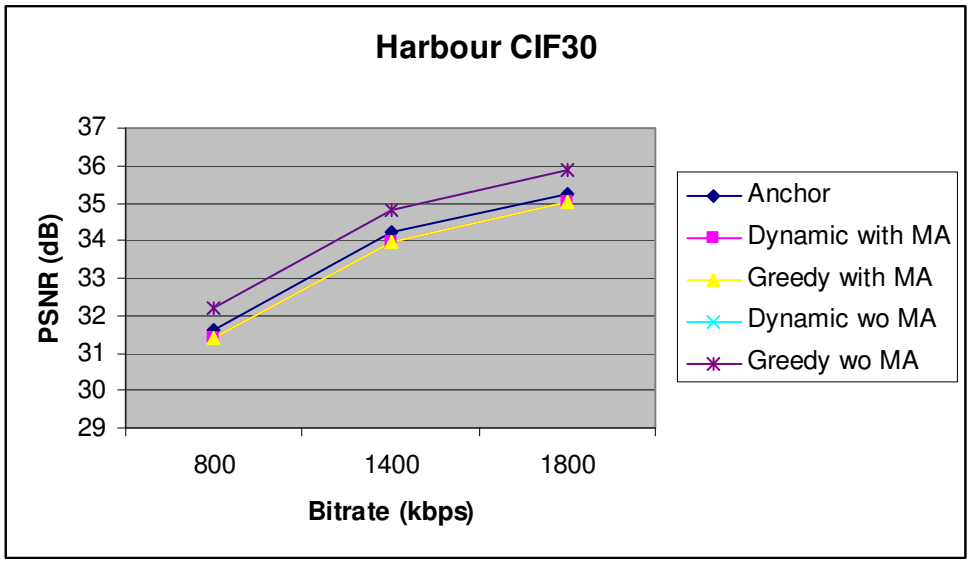

Fig. 7.1 Results for spatial scalability: Harbour

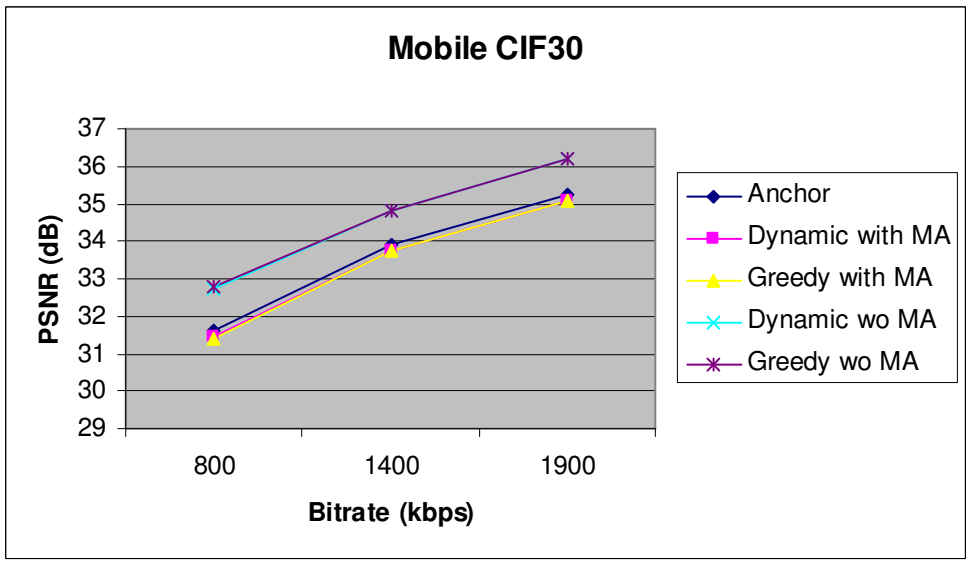

Fig. 7.2 Results for spatial scalability: Mobile

From fig. 7.1 and 7.2 we can see that, in 'with MA' scenario the overall bitrate increases than anchor (i.e. sequence coded without using SIP decision) within restricted limits, whereas, in 'without MA' scenario there is considerable saving in bitrate as compared to anchor. We can

also see that, in both cases, Greedy performance matches closely with that using Dynamic Programming approach.

It should be noted here that, since Greedy can never perform better than Dynamic Programming, the decision error is always positive for 'with MA' scenario (it means Greedy may not achieve allowed maximum bitrate increase margin) whereas decision error is always negative for 'without MA' case (which means, Greedy may not achieve minimum possible bitrate in this case). However since each coded bitstream has limitation on bitrate resolution to which it could be extracted with bitstream extractor provided by JSVM, the PSNR error may be positive or negative depending on deviation in bitrates between extracted substreams. However from magnitude of the PSNR error for all test cases we can see that deviation is extremely small with Greedy approach.

Following are results for CIF30 and 4CIF60 combination: (The error is measured for enhanced layer i.e. 4CIF60 layer in this case)

Table 7.2 Results for spatial scalability (Enhancement layer resolution 4CIF60)

Sequence	# of frames	% Error In Decision		% Error In PSNR		Avg Processing Time (u-sec)		Avg Memory Saving
		with MA	wo MA	with MA	wo MA	Dynamic	Greedy	
City	600	0.0002	-0.0005	0.0008	0.0010	1054000.0000	666.6666	7988.5000
Crew	300	0.0049	-0.0192	0.0096	0.0023	202333.3333	0.0000	175.0000
Soccer	300	0.0095	-0.0081	-0.0016	-0.0001	782666.6667	0.0000	500.6667
Harbour	300	0.0099	-0.0197	-0.0010	0.0060	207533.3333	0.0000	1012.3333

From both the tables we can observe that Greedy heuristic approach is extremely fast and consumes considerably less memory than Dynamic Programming based approach. Also, the deviation from PSNR value (and thereby the bitrates) is very small for all practical purposes.

7.2.2 Combined Scalability

Two spatial layers and three quality layers were used. The base and enhanced layers were coded with different QP values (28, 30), (30, 32) and (32, 36) with base layer and enhanced layer coded at 3 FGS layers. Some of the encoder settings include – GOP Size 32, AVC Compatible Base layer mode, Loop filter on (idc = 0), Motion search with FastSearch (mode = 4).

Following are results for QCIF15 and CIF30 combination: (The error is measured for enhanced layer i.e. CIF30 layer in this case)

Table 7.3 Results for combined scalability (Enhancement layer resolution CIF30)

Sequence	# of frames	% Error In Decision		% Error In PSNR		Avg Processing Time (u-sec)		Avg Memory Saving
		with MA	wo MA	with MA	wo MA	Dynamic	Greedy	
stefan	90	0.0781	-0.0313	0.0167	-0.0497	202666.6667	0.0000	19872.6667
bus	150	0.0000	0.0000	0.0000	0.0000	251666.6667	0.0000	35851.6667
deadline_1	150	0.0000	0.0000	0.0000	0.0000	189666.6667	0.0000	57812.8333
deadline_2	150	0.0000	0.0000	0.0000	0.0000	192666.6667	0.0000	69214.6667
deadline_3	150	0.0000	0.0000	0.0000	0.0000	237333.3333	0.0000	80123.6667
deadline_4	150	0.0000	0.0000	0.0000	0.0000	174666.6667	0.0000	56894.6667
deadline_5	150	0.0000	0.0000	0.0000	0.0000	201333.3333	333.3333	53058.8333
deadline_6	150	0.0000	0.0000	0.0000	0.0000	227333.3333	333.3333	44838.1667
deadline_7	150	0.0000	0.0000	0.0000	0.0000	207666.6667	0.0000	48625.8333
suzie	150	0.0607	-0.0591	-0.0179	0.0108	248000.0000	0.0000	3487.8333
miss-america	150	0.0942	-0.1066	-0.0252	0.0518	295333.3333	333.3333	3681.1667
flower	250	0.0000	0.0000	0.0000	0.0000	525333.3333	0.0000	83219.1667
tempete	260	0.0000	0.0000	0.0000	0.0000	568000.0000	0.0000	26574.6667
waterfall	260	0.0000	0.0000	0.0000	0.0000	444666.6667	0.0000	52988.1667
football	260	0.0183	-0.0321	-0.0079	0.0088	515333.3333	0.0000	895.8333
students1	300	0.0000	0.0000	0.0000	0.0000	640000.0000	0.0000	45508.3333
students2	300	0.0000	0.0000	0.0000	0.0000	481000.0000	0.0000	42798.8333
MaD9003	300	0.0016	-0.0010	-0.0026	-0.0006	696000.0000	0.0000	31905.5000
paris	300	0.0000	0.0000	0.0000	0.0000	512000.0000	0.0000	46942.1667
foreman	300	0.0000	0.0000	0.0000	0.0000	556666.6667	0.0000	37569.3333
mobile	300	0.0000	0.0000	0.0000	0.0000	560333.3333	0.0000	71443.6667
soccer	300	0.0000	0.0000	0.0000	0.0000	322333.3333	333.3333	58653.3333
harbour	300	0.0000	0.0000	0.0000	0.0000	591666.6667	333.3333	24721.3333

Table 7.3 - Continued

city	300	0.0000	0.0000	0.0000	0.0000	252333.3333	0.0000	84901.3333
crew	300	0.0154	-0.0306	-0.0016	0.0028	603333.3333	0.0000	665.3333
pamphlet	300	0.0015	-0.0384	-0.0068	-0.0019	435000.0000	0.0000	8383.3333
news	300	0.0069	-0.0145	-0.0019	0.0024	503000.0000	0.0000	43683.5000
silent	300	0.0000	0.0000	0.0000	0.0000	426666.6667	333.3333	24069.5000
akiyo	300	0.0110	-0.0231	-0.0364	-0.0121	532000.0000	0.0000	384.3333
coastguard	300	0.0029	-0.0063	0.0029	0.0022	618000.0000	333.3333	5692.0000
container	300	0.0000	0.0000	0.0000	0.0000	369000.0000	0.0000	25607.8333
hall	300	0.0138	-0.0187	-0.0049	0.0022	646666.6667	0.0000	884.1667
bowing	300	0.0078	-0.0086	-0.0006	0.0008	415000.0000	333.3333	6913.1667
carphone	382	0.0171	-0.0188	-0.0049	0.0022	837000.0000	333.3333	5.3333
claire	494	0.0078	-0.0213	-0.0053	0.0003	906333.3333	0.0000	600.6667
grandma	870	0.0035	-0.0089	-0.0029	-0.0001	1725666.6667	333.3333	323.3333
bridge-close	2000	0.0000	0.0000	0.0000	0.0000	4574333.3333	1333.3333	50349.6667

In this particular coding configuration, for some test cases, the factor W_{max} in Knapsack Problem becomes big enough to accommodate all input items. Therefore Greedy approach works correctly and we get zero error.

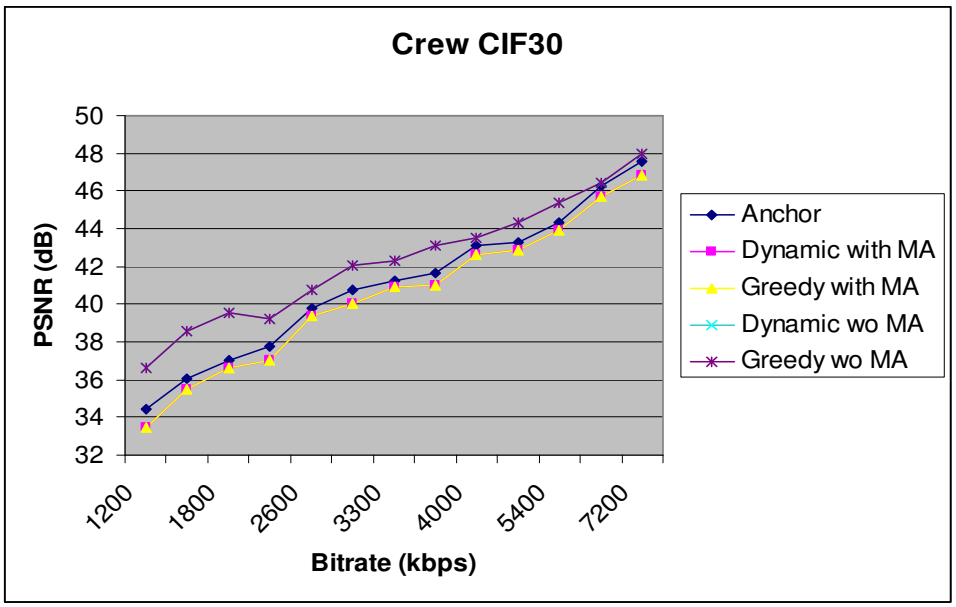

Fig. 7.3 Results for combined scalability: Crew

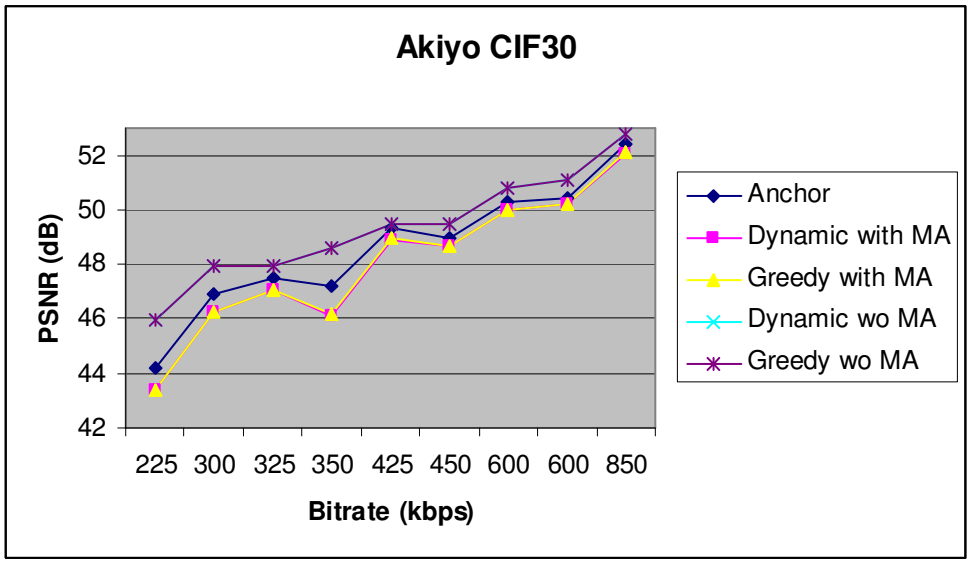

Fig. 7.4 Results for combined scalability: Akiyo

We can observe from fig. 7.3 and 7.4 that similar to spatial scalability, in combined scalability mode too, SIP strategy is useful as it saves bitrate in 'without MA' scenario.

Following are results for CIF30 and 4CIF60 combination where the base and enhanced layers were coded with different QP values (28, 30), (30, 32) and (32, 36) and 3 FGS layers: (The error is measured for enhanced layer i.e. 4CIF60 layer in this case)

Table 7.4 Results for combined scalability (Enhancement layer resolution 4CIF60)

Sequence	# of frames	% Error In Decision		% Error In PSNR		Avg Processing Time (u-sec)		Avg Memory Saving
		with MA	wo MA	with MA	wo MA	Dynamic	Greedy	
City	300	0.0000	0.0000	0.0000	0.0000	261400	0.0000	60342.3750
Crew	300	0.0019	-0.0020	0.0029	-0.0018	315200	0.0000	3094.5000
Soccer	300	0.0000	0.0000	0.0000	0.0000	416800	0.0000	28556.0000
Harbour	300	0.0069	-0.0187	-0.0036	0.0051	380000	333.3333	1712.8750

Thus, similar to spatial scalability, Greedy heuristic approach works well for combined scalability mode, too.

7.3 Conclusions

- Dynamic Programming is computationally complex approach to solve 0-1 Knapsack Problem in SIP Analyzer in H.264/SVC. Its complexity increases as number of input frames or coded rate increases. Time and memory consumption becomes considerably high for real time applications.
- Greedy Heuristic approach though may not always yield optimal solution to 0-1 Knapsack Problem, has advantage of considerably small resource consumption as compared to Dynamic Programming based approach. Our experiments verify this supposition.
- In terms of optimality of solution, the decision error introduced in SIP decision by sub-optimal Greedy approach (using highest value per unit weight criterion) is considerably small for most of the practical applications. Also its effect on bitrate deviation (or PSNR in our case) is substantially lesser in scope which should work for all real time applications where *reasonably good* solution is acceptable. We can not mathematically predict performance of Greedy approach; however our tests show that the maximum decision error is much less than 0.1% for almost all test scenarios.

7.4 Future Scope

This thesis is aimed at evaluating performance of Greedy heuristic for the 0-1 Knapsack Problem in SIP Analyzer tool in H.264/SVC reference software both qualitatively and quantitatively. As expected, it performs excellently as compared to existing Dynamic Programming based approach in terms of reduction in computational complexity. The

experiments show that, in terms of optimality of the solution, the deviation from bitrates/quality is extremely small for practical applications.

However since it is not mathematically provable, for those concerned with provably good solution, Greedy may not be the right approach. It may be a good idea to evaluate performance of E-optimal algorithms for this problem as compared to Dynamic Programming and Greedy Heuristic based approaches.

Current SIP scheme needs the video sequence to be encoded thrice. Any improvement in this strategy would also be highly useful in reducing overall processing time and resource consumption.

REFERENCES

[1] ITU-T Website: http://www.itu.int/rec/T-REC-H.261

[2] ITU-T Website: http://www.itu.int/rec/T-REC-H.262

[3] ITU-T Website: http://www.itu.int/rec/T-REC-H.263

[4] ITU-T Website: http://www.itu.int/rec/T-REC-H.264

[5] ITU-T Website: http://www.itu.int/ITU-T/

[6] MPEG Website: www.mpeg.org

[7] ITU-T and ISO/IEC JTC 1, “Advanced video coding for generic audiovisual services,” ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG4-AVC), Version 1: May 2003, Version 2: Jan. 2004, Version 3: Sep. 2004, Version 4: July 2005

[8] Heiko Schwarz et al, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", To appear in IEEE Transactions on Circuits and Systems for Video Technology, September 2007

[9] K. Zhang, J. Xu and F. Wu, "Frame Based Selective Interlayer Prediction," JVT-S051, Apr. 2006

[10] I.E.G Richardson, “H.264 and MPEG-4 Video Coding for Next Generation Multimedia”, John Wiley & Sons, 2003

[11] I.E.G Richardson,” Video Codec Design: developing image and video compression systems”, Chichester: John Wiley & Sons, 2002

[12] K.R.Rao and J.J.Hwang, “Techniques and standards for Image, Video and Audio Coding”, Prentice Hall, 1996

[13] M. Ghanbari, “Standard Codecs: Image Compression to Advanced Video Coding”, London, U.K.: Institution of Electrical Engineers, 2003

[14] T. Wiegand, et al, “Overview of the H.264/AVC video coding standard”, IEEE Trans. CSVT, Vol.13, pp. 560-576, July 2003

[15] A. Puri, H. Chen and A. Luthra, "Video Coding using the H.264/MPEG-4 AVC compression standard", Signal Processing: Image Communication Vol. 19, pp.793-849, Oct. 2004

[16] Soon-kak Kwon, A. Tamhankar and K.R. Rao, "Overview of H.264 / MPEG-4 Part 10", Special issue on " Emerging H.264/AVC video coding standard", J. Visual Communication and Image Representation, vol. 17,pp. 2006

[17] MPEG-4: ISO/IEC JTC1/SC29 14496-10: Information technology – Coding of audio-visual objects - Part 10: Advanced Video Coding, ISO/IEC, 2005

[18] Peter List et al, "Adaptive Deblocking Filter", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 13, NO. 7, JULY 2003

[19] S. W. Golomb, "Run-Length Encoding," *IEEE Trans. on Information Theory*, IT-12, pp. 399-401, Dec. 1966

[20] M. Flierl, T.Wiegand, and B. Girod, "A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction," in Proc. Data Compression Conf., Snowbird, UT, Mar. 1998, pp. 239–248

[21] M. Flierl and B. Girod, "Generalized B Picture and the Draft H.264/AVC Video-Compression Standard," *IEEE Trans. CSVT*, Vol. 13, pp. 587-597, July 2003

[22] J. Ostermann, et al, "Video coding with H.264/AVC: Tools, performance and complexity", IEEE CAS Magazine, Vol.4, pp.7-34, I quarter, 2004

[23] Apple Website: Apple QuickTime: FAQ - H.264

[24] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2 Video), Nov. 1994

[25] ITU-T, "Video coding for low bit rate communication," ITU-T Recommendation H.263, Version 1: Nov. 1995, Version 2: Jan. 1998, Version 3: Nov. 2000

[26] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14492-2 (MPEG-4 Visual), Version 1: Apr. 1999, Version 2: Feb. 2000, Version 3: May 2004

[27] Thomas Wiegand - Scalable video model 3.0. Joint Video Team (JVT), Jan 2005

[28] Athanasios Leontaris et al, "Drift-Resistant SNR Scalable Video Coding", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 15, NO. 8, AUGUST 2006

[29] JENS-RAINER OHM, "Advances in Scalable Video Coding", Invited Paper, PROCEEDINGS OF THE IEEE, VOL. 93, NO. 1, JANUARY 2005

[30] H. Schwarz, T. Hinz, H. Kirchhoffer, D. Marpe, and T. Wiegand, "Technical description of the HHI proposal for SVC CE1," ISO/IEC JTC 1/SC 29/WG 11, doc. M11244, Palma de Mallorca, Spain, Oct.2004

[31] T. Wiegand, G. J. Sullivan, J. Reichel, H. Schwarz, and M. Wien, eds., "Joint Draft 11 of SVC Amendment," Joint Video Team, doc. JVTX201, Geneva, Switzerland, July 2007

[32] J. Reichel, M. Wien, and H. Schwarz, eds., "Scalable Video Model 3.0," ISO/IEC JTC 1/SC 29/WG 11, doc. N6716, Palma de Mallorca, Spain, Oct. 2004

[33] K. Ramchandran, A. Ortega, M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," IEEE Transactions on Image Processing, vol. 13, no. 5, Sep. 1994

[34] J. Reichel, H. Schwarz, M. Wien, eds., "Joint scalable video model 11 (JSVM 11)," Joint Video Team, doc. JVT-X202, Geneva, Switzerland, July 2007

[35] HHI Website: http://ip.hhi.de/imagecom_G1/savce/

[36] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand, "The Scalable H.264/MPEG4-AVC Extension: Technology and Applications", EuMob'06, paper Alghero, Italy, September 20, 2006

[37] J. Reichel, H. Schwarz, and M. Wien (eds.), "Scalable Video Coding – Joint Draft 6," Joint Video Team, Doc. JVT-S201,Geneva, Switzerland, Apr. 2006.

[38] J. Reichel, H. Schwarz, and M. Wien (eds.), "Joint Scalable Video Model JSVM-6," Joint Video Team, Doc. JVT-S202, Geneva, Switzerland, Apr. 2006.

[39] E. François and J. Vieron, "Extended spatial scalability: a generalization of spatial scalability for non-dyadic configurations, "Proceedings of ICIP 2006, Atlanta, GA, USA, Oct. 2006.

[40] Y. C. Lin: "Introduction to H.264/SVC", May 10, 2006

[41] H. Schwarz, et al, "Constrained inter-layer prediction for single-loop decoding in spatial scalability," Proceedings of ICIP 2005, Geneva, Italy, Sep. 2005.

[42] H. Kirchhoffer et al, "A LOW-COMPLEXITY APPROACH FOR INCREASING THE GRANULARITY OF PACKET-BASED FIDELITY SCALABILITY IN SCALABLE VIDEO CODING", To be presented at Picture Coding Symposium (PCS 2007), November 7-9, 2007, Lisbon, Portugal

[43] Text of ISO/IEC 14496-10:2005/FDAM 3 Scalable Video Coding, Joint Video Team (JVT) of ISO-IEC MPEG & ITU-T VCEG, Lausanne, N9197, Sep. 2007

[44] H. Kirchhoffer, H. Schwarz, and T. Wiegand, CE1: Simplified FGS, Joint Video Team (JVT) of ISO-IEC MPEG & ITU-T VCEG, JVTW090 Apr. 2007

[45] J. Ridge and M. Karczewicz, AHGreport: FGS applications and design simplification, Joint Video Team (JVT) of ISO-IEC MPEG & ITU-T VCEG, JVT-X 006 Jul. 2007

[46] K. Zhang, J. Xu and F. Wu, "Selective Inter-layer Prediction," JVT-R064, Jan. 2006

[47] Wikipedia: http://en.wikipedia.org/wiki/Knapsack_problem

[48] "Greedy Algorithms, Dynamic Programming and Approximation Algorithms", CITS3210 Algorithms Presentation, The University of Western Australia:

[49] Thomas H. Cormen et al, "Introduction to Algorithms", 2nd ed, MIT Press.

[50] Wikipedia: http://en.wikipedia.org/wiki/Memory_bound_function

[51] Levitin, Anany, “The Design and Analysis of Algorithms”, New Jersey: PearsonEducation Inc., 2003.

[52] “Genetic Algorithm Tutorial”, by Marek Obitko

[53] Maya Hristakeva et al, "Different Approaches to Solve the 0/1 Knapsack Problem", The Midwest Instruction and Computing Symposium 2005 paper 102.

[54] http://20bits.com/2007/05/08/introduction-to-dynamic-programming/

[55] Wikipedia: http://en.wikipedia.org/wiki/NP-complete

[56] O.H. Ibarra and C.E. Kim (1975) “Fast approximation algorithms for Knapsack and sum of subset problem”, Journal of ACM 22, 463-468

[57] David Pisinger, “Algorithms for Knapsack Problems”, PhD Thesis Feb 1995 Dept of Computer Science, University of Copenhagen

[58] Diptesh Ghosh et al., “Sensitivity analysis of the Greedy Heuristic for Binary Knapsack Problems”, SOM - reports University of Groningen publication 00A18

[59] Ref: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Greedy/greedyIntro.htm

[60] Wikipedia: http://en.wikipedia.org/wiki/Greedy_algorithm

[61] Ref: http://web.csse.uwa.edu.au/__data/page/87405/heuristic-greedy4.pdf

[62] MPEG 4 AAC - ISO/IEC 14496-3

[63] Wikipedia: http://en.wikipedia.org/wiki/Quicksort

[64] Quick Sort Tutorial: http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/quick/quicken.htm

[65] JVT, "Joint Scalable Video Model JSVM ", http://ftp3.itu.int/av-arch/jvt-site JSVM_5_10)

[66] Test sequences: ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/

[67] Test sequences: http://ftp3.itu.int/av-arch/video-site/sequences

[68] Test sequences: http://media.xiph.org/video/derf/

[69] Test sequences: http://trace.eas.asu.edu/yuv/cif.html

BIOGRAPHICAL INFORMATION

Jaydeep received his Bachelor of Engineering (B.E.) degree in Electronics & Telecommunication Engineering from Pune University, India in May 2003. After working in industry for three years, developing embedded software for Speech/Audio DSP applications mainly for mobile handsets, he pursued his graduate studies in Electrical Engineering at University of Texas at Arlington, USA. He was member of Multirate Signal Processing lab under guidance of Dr. Soontorn Oraintara. He received his Master of Science (M.S.) degree in Electrical Engineering in May 2008. He has experience working with companies like Cirrus Logic, STMicroelectronics, Qualcomm and Intel in capacity of intern or full time engineer. His research interests are signal processing and algorithm development for real time multimedia applications and latest trends in computer architecture and embedded software.