

Kemal Ugur

Improved Prediction Methods for Low Complexity, High Quality Video Coding

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Sähkötalo Building, Auditorium S3, at Tampere University of Technology, on the 8th of November 2010, at 13 o'clock.

ISBN 978-952-15-2471-4
ISSN 1459-2045

Abstract

Video coders are an essential part of any digital video application, such as digital TV broadcasting or video conferencing as the bitrate of an uncompressed video signal is too high for any practical usage. The goal of all video coders is to maximize the video quality while minimizing the bitrate, with as low complexity as possible. This is achieved by exploiting different redundancies present in the video signal. One type of redundancy is present between different pictures captured either at different time instants from the same camera (temporal redundancy) or from different cameras at the same time instant (view redundancy). The focus of this thesis is to develop algorithms to exploit the redundancies between different pictures more effectively with the goals of improving coding efficiency of video coders and decreasing their implementation complexity.

The first part of this thesis presents a detailed complexity analysis of motion compensated prediction part of the most recent H.264/AVC standard that is used to exploit the temporal redundancy. More specifically, the interpolation function of the standard is described in detail and its implementation complexity is analyzed. This part presents a novel algorithm to generate easy to decode bitstreams with high coding efficiency utilizing the flexibility of the H.264/AVC standard.

The second part of this thesis discusses adaptive interpolation filtering (AIF) to improve the coding efficiency of video coders. In AIF, the coefficients of the interpolation filter are updated dynamically to compensate the time variant nature of the aliasing present in the video signal. In this part, firstly novel algorithms are presented to improve the coding efficiency of AIF schemes. This is achieved by making the filter structure and the filter symmetry more flexible so that the AIF tool becomes useful for a larger set of sequences exhibiting different statistical behavior and at different bitrates. Secondly, novel algorithms are presented to reduce both the encoding and decoding complexity of AIF schemes. This is done by using a novel structure that uses less number of taps than the state-of-art schemes, enabling an efficient implementation using 16-bit integer arithmetic. In addition, novel algorithms are developed to reduce the encoding complexity while maintaining the coding efficiency gain.

The third and the final part of this thesis discusses three dimensional video coding, and specifically the Multiview Video Coding (MVC) extension of the H.264/AVC standard. MVC is a recent addition to the H.264/AVC family of video coding standards, to enable the emerging 3D video applications. The focus of this part is on the implementation complexity of MVC, paying special attention to parallel decoding of multiple views on different processors or cores. A novel algorithm is presented that encodes the views with certain restrictions so that the views could be decoded in parallel.

Preface

The research presented in this thesis has been carried out at Nokia Research Center, and at the Department of Signal Processing of Tampere University of Technology. During the preparation of the thesis, the author worked within the Visual Content Processing and Audiovisual Content Representation teams.

The work would not be possible without the help and support of many people. First of all, I would like to thank my supervisor Prof. Moncef Gabbouj for his guidance and encouragement throughout the years. I would also like to thank Prof. Gabbouj for enabling a healthy and fruitful collaboration environment between Nokia Research Center and Department of Signal Processing of Tampere University of Technology.

I would like to thank the reviewers, Prof. Rabab Ward and Prof. Murat Tekalp, for their valuable feedback on the draft of the thesis.

From Nokia Research Center, I would firstly like to thank Jani Lainema for giving me the opportunity to take responsibility in this field and creating a pleasant and friendly working atmosphere at Nokia. I would also like to thank my current manager, Mauri Väänänen, for his encouragement and support for finishing my thesis. I would like to thank Antti Hallapuro and Dr. Dmytro Rusanovskyy for the fruitful collaboration over the years that resulted in many publications, some of which also formed the essential part of this thesis. I would also like to thank all my colleagues at Nokia and friends making my time in Finland pleasant.

Financial support of Nokia Foundation is gratefully acknowledged.

Last but not least, I wish to express my deepest gratitude for my parents, Ayfer and Avni Ugur and my sister Burcu, for their continuous encouragement and support over the years.

Tampere, September 2010

Kemal Ugur

Contents

Abstract	i
Preface	iii
Contents.....	v
List of Publications	vii
List of Supplementary Publications.....	ix
List of Acronyms	xi
List of Tables.....	xiii
List of Figures	xv
1. Introduction	17
1.1. Fundamentals of video coding	18
1.2. Outline and Objectives of the thesis	21
1.3. Author's Contributions to the Publications.....	22
2. Motion Compensated Prediction and Interpolation	25
2.1. Luminance Interpolation in H.264/AVC	27
2.2. Complexity Analysis of H.264/AVC Interpolation	29
2.3. Generating bitstreams with low decoding complexity.....	31
3. Adaptive Interpolation Filtering.....	37
3.1. Fundamentals of adaptive interpolation Filtering (AIF).....	37
3.1.1. <i>Encoding Complexity of 2D-AIF</i>	38
3.1.2. <i>Decoding Complexity of 2D-AIF</i>	39
3.2. Directional adaptive interpolation filtering.....	40
3.2.1. <i>16-Bit Integer Arithmetic Implementation for DAIF</i>	42
3.3. Low Complexity Encoding methods for AIF	51
3.3.1. <i>Efficient Calculation of Filter Coefficients with Distortion Modeling</i>	51
3.3.2. <i>Adaptive Interpolation with Restricted Second Pass</i>	54
3.4. Coding Efficiency improvements for Adaptive interpolation.....	57
3.4.1. <i>Adaptive Interpolation with Flexible Symmetry</i>	58
3.4.2. <i>Adaptive Interpolation with Flexible Filter Structures</i>	62
4. 3D and Multiview Video Coding.....	67
4.1. MVC Amendment of H.264/AVC	67

4.1.1. <i>Features of MVC</i>	67
4.1.2. <i>Performance of MVC</i>	68
4.2. Parallel processing of multiple views	70
4.3. Adaptive interpolation filtering for MVC	76
5. Conclusions	77
5.1. Future work	78
Bibliography	79

List of Publications

This thesis is written in a summary style followed by the following publications listed below as appendices. The publications are referenced in the thesis as [P1], [P2], etc.

- [P1] Ugur, K.; Lainema, J.; Hallapuro, A.; Gabbouj, M., "Generating H.264/AVC Compliant Bitstreams for Lightweight Decoding Operation Suitable for Mobile Multimedia Systems," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* , vol.2, no., pp.II-II, 14-19 May 2006
- [P2] Ugur, K.; Rusanovskyy, D.; Hallapuro, A.; Lainema, J.; Gabbouj, M., "Efficient implementation of adaptive interpolation filters for low complexity video coding," *Consumer Electronics, IEEE Transactions on* , vol.55, no.1, pp.257-263, February 2009
- [P3] Rusanovskyy, D.; Ugur, K.; Gabbouj, M., "Fast encoding algorithms for video coding with adaptive interpolation filters," *Multimedia Signal Processing, 2008 IEEE 10th Workshop on* , vol., no., pp.317-321, 8-10 Oct. 2008
- [P4] Ugur, K.; Rusanovskyy, D.; Gabbouj, M., "Efficient calculation of adaptive interpolation filter with distortion modelling," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on* , vol., no., pp.673-676, March 31 2008-April 4 2008
- [P5] Rusanovskyy, D.; Ugur, K.; Gabbouj, M.; Lainema, J., "Video coding with pixel-aligned directional adaptive interpolation filters," *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* , vol., no., pp.704-707, 18-21 May 2008
- [P6] Rusanovskyy, D.; Ugur, K.; Hallapuro, A.; Lainema, J.; Gabbouj, M., "Video Coding With Low-Complexity Directional Adaptive Interpolation Filters," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.19, no.8, pp.1239-1243, Aug. 2009
- [P7] Ugur, K.; Lainema, J.; Gabbouj, M., "Adaptive Interpolation Filter with Flexible Symmetry for Coding High Resolution High Quality Video," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* , vol.1, no., pp.I-1013-I-1016, 15-20 April 2007
- [P8] Rusanovskyy D., Ugur K., Gabbouj M., "Adaptive Interpolation with Flexible Filter Structures", *Proceedings of IEEE International Conference on Image Processing, ICIP 2009*, Cairo, Egypt, Nov. 2009, pp. 1025-1028.

- [P9] Ugur, K.; Hui Liu; Lainema, J.; Gabbouj, M.; Houqiang Li, "Parallel Encoding - Decoding Operation for Multiview Video Coding with High Coding Efficiency," *3DTV Conference, 2007* , pp.1-4, 7-9 May 2007

- [P10] Y. Chen, Y.-K. Wang, Ugur K.; Hannuksela M.M.; Lainema J.; Gabbouj M., "The emerging MVC standard for 3D video services," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 786015, 13 pages, 2009.

List of Supplementary Publications

The following supplementary publications support the novel techniques and results of the thesis but have not undergone a thorough academic review process or are not as essential for the thesis as the publications listed earlier.

- [S1] Rusanovskyy, D.; Ugur, K.; Lainema, J., “Adaptive interpolation with directional filters”, ITU-T SG16/Q6 Doc. VCEG-AG21, Shenzhen, China, July 2007.
- [S2] Ugur, K.; Hallapuro, A.; Rusanovskyy, D.; Lainema, J., “Implementation of adaptive filters with 16-bit arithmetic”, ITU-T SG16/Q6 Doc. VCEG-AG22, Shenzhen, China, July 2007.
- [S3] Rusanovskyy, D.; Ugur, K.; Hallapuro, A.; Lainema, J., “Implementation of directional interpolation filter with 16-bit Integer Arithmetic”, ITU-T SG16/Q6 Doc. VCEG-AH18, Antalya, Turkey, January 2008.
- [S4] Fuldseth, A.; Bjontegaard, G.; Rusanovskyy, D.; Ugur, K.; Lainema, J., “Low Complexity Directional Interpolation Filter”, ITU-T SG16/Q6 Doc. VCEG-AI12, Berlin, Germany, July 2008.
- [S5] Rusanovskyy, D.; Ugur, K.; Karczewicz, M.; Ye, Y.; Chen, P.; Motta, G.; "Improvements on Enhanced Directional Adaptive Filtering (EDAIF-2)," ITU-T Q.6/SG16 VCEG, Doc. C125, Geneva, Switzerland, January 2009.
- [S6] Ugur, K.; Andersson, K. R.; Fuldseth, A., “Video coding technology proposal by Tandberg, Nokia, and Ericsson”, JCTVC-A119, first meeting of the JCT-VC, Dresden, Germany, April 2010
- [S7] Ugur, K.; Lainema, J.; H. Liu, Y.-K. Wang, “Parallel Decoding Info SEI Message for MVC“, JVT-V098. Marrakech, Morocco, January 2007.

List of Acronyms

3GPP	The 3rd Generation Partnership Project
4CIF	Four times resolution of CIF
720p	1280 pixels wide by 720 pixels height in progressive scan
1080p	1920 pixels wide by 1080 pixels height in progressive scan
1D, 2D, 3D	One-, Two-, Three-Dimensional
AIF	Adaptive interpolation filter
ARM	Processor family used commonly in mobile environments
CIF	Common intermediate format
DAIF	Directional adaptive interpolation filter
DCT	Discrete Cosine Transform
DVB	Digital video broadcast standardization organization
DVB-H	DVB-H (Handheld), multimedia delivery standard
DVD	Digital Versatile Disc
Blu-ray	Optical disc storage medium with capacity higher than DVD
MCP	Motion Compensated Prediction
H.264/AVC	Video compression standard jointly developed by MPEG and VCEG
HEVC	High efficiency video coding, name for the next generation standard
KTA	Key technological area, experimental software maintained by VCEG
Macroblock	A block containing one or more transform blocks
Mbits/s	Megabits per second
MPEG-4	Family of video compression standards developed by MPEG
FIR	Finite impulse response
MVC	Multiview video coding
JVT	Joint video team of MPEG and VCEG
MPEG	Moving picture experts group, standard developing organization
PSNR	Peak signal to noise ratio
QCIF	Quarter common intermediate format
RD	Rate distortion
RDO	Rate distortion organization
SSD	Sum of squared differences
SEI	Supplemental enhancement information
TV	Television
VCEG	Video coding experts group, standard development organization
YUV	A color space

List of Tables

Table 1 Interpolation complexity for each sub-pixel position in terms of required number of interpolation for a 4x4 block	30
Table 2 Coding efficiency and complexity performance of the proposed algorithm	34
Table 3 Interpolation Complexity of various schemes (number of 32-bit airthmetic operations to interpolate a sub-pixel value)	41
Table 4 Simulation Results of the Proposed DAIF schemes comparing to the H.264/AVC Interpolation.	50
Table 5 Simulation results of low complexity AIF encoding	54
Table 6 Motion vector variations between first and second pass of encoding	56
Table 7 Coding efficiency of the proposed algorithms with respect to H.264/AVC interpolation filter	57
Table 8 Encoding complexity of the proposed algorithms in percentage of H.264/AVC reference encoding	57
Table 9 Coding efficiency of adaptive interpolation with flexible symmetry	61
Table 10 Simulation results of adaptive interpolation with flexible filter structures	65
Table 11 Decoding Speed Increase Compared to Single-View Decoder for Three Approaches	74
Table 12 Coding Efficiency of the proposed parallel multiview scheme relative to anchor and simulcast approaches	75

List of Figures

Figure 1 Illustration of spatial redundancy in a video frame	18
Figure 2 Illustration of temporal redundancy in a video signal	19
Figure 3 Block diagram of a typical H.264/AVC video encoder	20
Figure 4 Overview of Motion Compensated Prediction	21
Figure 5 Motion partitions used in H.264/AVC	26
Figure 6 Illustration of multiple reference pictures [2]	26
Figure 7 Luminance interpolation. Gray boxes indicate samples at integer positions and white boxes indicate fractional samples.	28
Figure 8 Quarter-pixel interpolation defined in H.264/AVC	28
Figure 9 The sub-pixel interpolation process for position j .	31
Figure 10 Approximate interpolation complexities	32
Figure 11 Analysis of the proposed method. Above graph shows the coding efficiency drop due to using the proposed method. Below graph shows the achieved complexity reduction in total decoding complexity.	35
Figure 12 Flowchart of video encoder with adaptive interpolation.	38
Figure 13 Execution time for computation of correlation functions in percentage of all steps combined	39
Figure 14 Illustration of DAIF structure for (a) sub-pixel locations $\{e,o\}$ and (b) sub-pixel locations $\{g,m\}$	42
Figure 15 Illustration of intermediate clipping	44
Figure 16 Frequency response of filters with different rounding methods.	48
Figure 17 Effect of rounding on interpolation quality	49
Figure 18 Prediction error due to quantization. Dotted lines indicate the calculated error model.	53
Figure 19 Refinement of fractional component of motion vectors (a) half-pixel refinement, (b) quarter-pixel refinement	55
Figure 20 Number of independent filters for different symmetries	59
Figure 21 Number of coefficients for each sub-pixel position for different symmetries	60
Figure 22 Coding efficiency of the proposed algorithm compared to 2D-AIF (denoted as KTA filter)	62
Figure 23 An interpretation of 2-tap length filters diversity analysis. Euclidian distance (solid arrow) between two filters (dashed arrows) is compared against the threshold.	63
Figure 24 12-tap filter structures: a) – diagonal cross filter; b) – radial support filter.	64
Figure 25 Cut-off frequencies (at the level -3dB) for three $h(j)$ filters: (thick solid line) – 2D 6x6 filter of H.264/AVC, (thin solid line) – estimate of 12-tap Radial Filter, (thin dash-dotted line) – estimate of Diagonal-Cross Filter.	64
Figure 26 Illustrative prediction structure for MVC using Hierarchical B pictures	69
Figure 27 Performance of MVC compared to simulcast [36]	70
Figure 28 Sample prediction structure for two views	71
Figure 29 Systematic restriction of reference area	72
Figure 30 Sample parallel decoding process for two views	73
Figure 31 Comparing the proposed parallel encoding scheme with anchor and simulcast	75
Figure 32 Workflow for asymmetric MVC coding [32]	76

Chapter 1

Introduction

Digital video compression is a key technology that enables a number of services in many industries ranging from entertainment to communications. Every time a user enjoys a video clip on a popular video sharing site, records a video on a mobile phone or on a digital camera, plays a movie from a DVD or a Blu-ray disc, attends a video-conferencing meeting, uses video surveillance for security, etc. video compression technologies are seamlessly being used. The need for video compression in these services arises from the fact that a significant amount of space or bandwidth is required to store and transmit the uncompressed video. Consider a video sequence with a resolution of 720x480 pixels (a common resolution used in DVD discs), played at 25 frames-per-second. The bitrate of this video with three color components at 8 bits per pixel is over 200 Mbits/s. To store this video in current DVD discs, a compression by a factor of at least 20 is required.

The resolutions used in current video services are increasing at a fast pace. High Definition (HD) resolutions of 720p (1280x720 pixels) and 1080p (1920x1080 pixels) are becoming increasingly common for digital TV broadcast and other services. It is foreseen that with the advances in display and capture technologies, even higher resolutions such as 4096x2160 pixels will be common in consumer entertainment applications. In addition to increased spatial resolution, the frame rates of the video sequences are expected to increase to around 60 frames-per-second and higher. In addition to increased temporal and spatial resolutions, stereoscopic 3DTV's are becoming a reality in consumer domain with the advances in display technologies. In 3DTV, pictures for both views are transmitted, effectively doubling the information that need to be transmitted. To enable digital video services for these emerging applications that need to transmit a large amount of information, more efficient compression schemes need to be designed.

In addition to compression capabilities of video coders, the complexity required to implement them is a major concern. Consider the case of watching video a sampled at 1080p resolution at 60 frames-per-second. This means that a video decoder needs to process more than 6 millions pixels in less than 20 milliseconds. For the case of stereoscopic 3D video, the computation requirements are doubled due to the need to process pictures for both left and right eye. Therefore, a practical video coder needs to achieve significant compression and at the same time operate at suitable complexity levels. The implementation complexity is espe-

cially important in realizing video services over mobile devices, where the computational resources are severely limited. This requirement is expected to be more important as mobile devices are becoming increasingly popular to record and watch video.

At the time of writing the thesis, increased compression efficiency at high resolutions and improved computational efficiency are key requirements identified in the video standardization community for the next generation video standardization project [1].

1.1. FUNDAMENTALS OF VIDEO CODING

A video sequence is essentially a series of frames, captured at very short time intervals (40 milliseconds for a 25 frame-per-second video). Video coder achieves compression by exploiting the redundancies in the video signal and also removing some information that is difficult to see with the human visual system.

There exist three types of redundancies in the video signal. First type of redundancy is called *Spatial redundancy*, which means that a pixel in a given video frame is correlated to neighboring pixels within the same frame. Instead of coding each pixel in a frame independently, typical video coders exploit this redundancy and use *intra-frame* coding techniques similar to techniques present in still image compression, such as transform coding, intra prediction, etc. An example of spatial redundancy is illustrated in Figure 1.

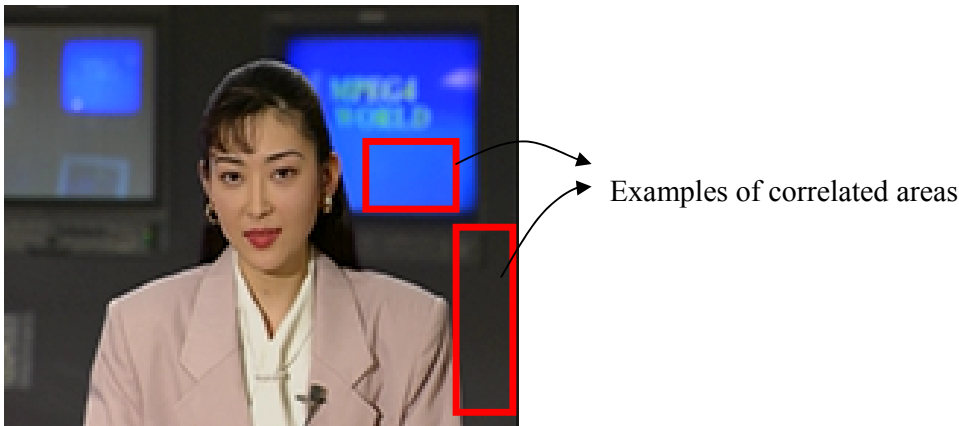


Figure 1 Illustration of spatial redundancy in a video frame

Second type of redundancy is present between pictures and it is called temporal redundancy (or view redundancy if the pictures are captured from different viewpoints), which means a frame in the video sequence is highly correlated to temporally neighboring frames. In other words, frames in a video sequence look very similar to neighboring frames and differ slightly due to movements of an object in the scene or due to global camera motion. Thus, instead of coding each frame independently, a typical video coder uses *inter-frame* coding techniques and codes each frame predictively with respect to other frames. Temporal redundancy in a video sequence is further illustrated in Figure 2 that shows three consecutive frames from a video sequence. As seen in Figure 2, only changing area in these frames are the eyes of the

woman, whereas all the other areas remain unchanged or very slightly moving. Instead of coding these frames independently, video coder only codes the difference between those frames.



Figure 2 Illustration of temporal redundancy in a video signal

In addition to these two redundancies present in the video signal, another type of redundancy is present in the video bitstream, and it is called *statistical redundancy*. Video coders utilize different syntax elements to exploit the temporal and spatial redundancies, such as motion vectors or transform coefficients. Statistical redundancy refers to the correlations that exist among those syntax elements, and video coders utilize entropy coding tools, such as variable length coding to exploit this.

A video coder that utilizes *inter-frame* and *intra-frame* coding tools to reduce temporal and spatial redundancy along with entropy coding tools to reduce statistical redundancy is generally referred to as a *hybrid* video coder. The latest video coding standard, H.264/AVC, is a hybrid video coder and its simplified encoder block diagram is shown in Figure 3 [2]. As seen in Figure 3, the prediction signal for the current frame is first generated either by intra-frame prediction using pixels from the same frame or by inter-frame prediction using pixels from other frames. Then the difference between the original and prediction signal is transformed and quantized. The quantized transform coefficients and the prediction information (such as motion vectors for inter-frame prediction) are entropy coded and transmitted to the decoder.

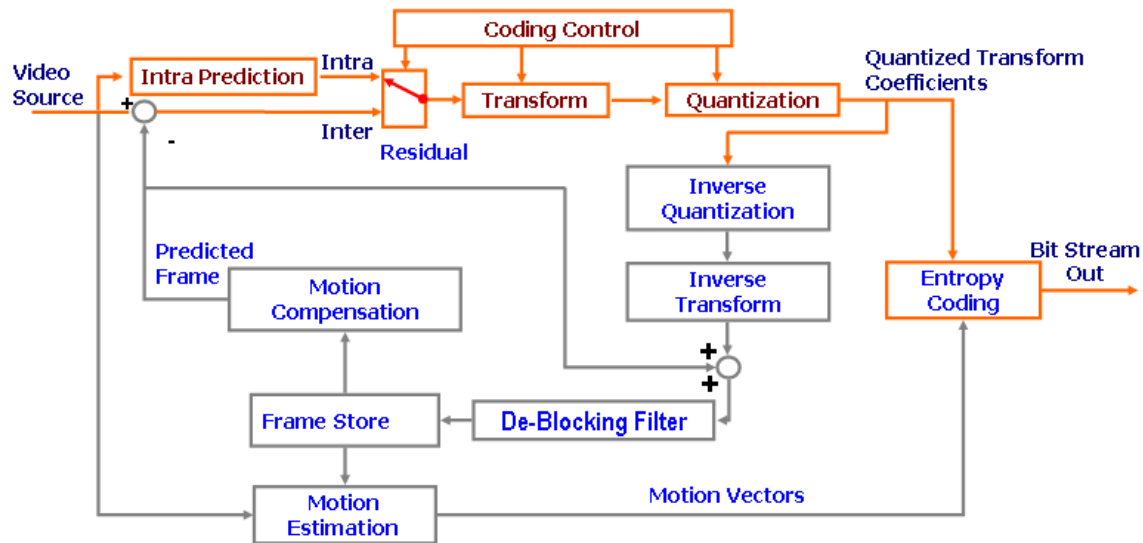


Figure 3 Block diagram of a typical H.264/AVC video encoder

As described above, the main difference of a video signal compared to still images is the additional temporal redundancy present in the video signal. Therefore, *inter-frame* coding tools exploiting this temporal redundancy play an important role in video compression. The most widely used *inter-frame* coding technique is called Motion Compensated Prediction (MCP), and it is used by many popular video coding standards, such as MPEG-2, MPEG-4, and H.264/AVC. In MCP, each frame of the video sequence is divided into blocks and the reference frame is searched for every block in the original frame to find their best match, and the best matching block in the reference frame forms the prediction for the current block. The best matching block is called the prediction for the current block, and its relative position with respect to the original one is called the block's motion vector. The video coder subtracts the prediction from the original block and obtains the residual signal. The residual signal is then transformed, quantized, entropy coded and transmitted to the decoder along with the block's motion vector. A simple illustration of this process is given in Figure 4.

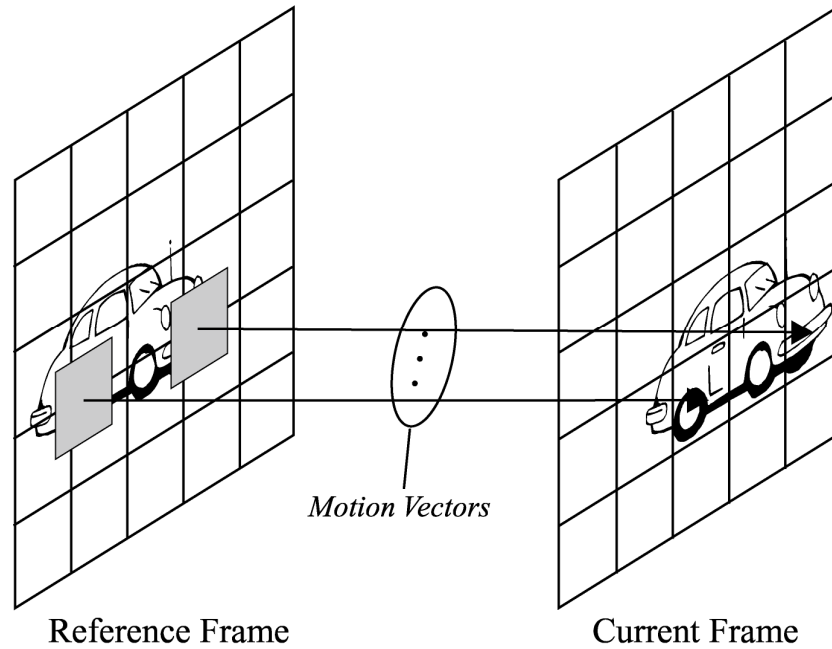


Figure 4 Overview of Motion Compensated Prediction

For each motion compensated block, the decoder receives the residual signal and the block's motion vector. Using the motion vector, the decoder first generates the prediction signal. Then, the decoder inverse transforms and inverse quantizes the residual signal and adds it to the prediction signal, which results in the final reconstruction of the current block. In the H.264/AVC standard, this reconstruction signal is further smoothed using a mandatory in-loop deblocking filter, to reduce the visually annoying effects of quantization.

In order to enable interoperable implementations of various video services from different companies, international standardization bodies create video coding specifications. H.264/AVC is the most recent video coding standard, and it is created jointly by the Video Coding Experts Group (VCEG) of ITU-T and the Moving Pictures Experts Group (MPEG) of ISO-IEC. Like other video coding standards, H.264/AVC specifies how the coded video bitstream is decoded and leaves particular encoder implementations to the market to foster further development in the industry. As all coding tools related to MCP lay within the coding loop, MCP is one of the areas specified in the standard. In other words, the H.264/AVC standard specifies exactly the tools used in MCP, such as interpolation, motion vector decoding and motion partition block sizes.

1.2. OUTLINE AND OBJECTIVES OF THE THESIS

This thesis presents novel algorithms and methods to improve the coding efficiency and reduce the complexity of video coders. The developed algorithms have a particular focus on high-resolution, low complexity use-cases to enable the emerging high quality high resolution video services on mobile terminals.

This thesis presents three main contributions. First, the motion compensation of the most recent video coding standard, H.264/AVC, is introduced in Chapter 2 and its complexity is analyzed with a particular focus on interpolation complexity. In this chapter, novel encoding algorithms to generate easy to decode bitstreams are also presented. The aim of this work is to demonstrate the flexibility of the H.264/AVC standard, and study the complexity-coding efficiency trade-off that could be achieved whilst preserving the H.264/AVC standard specification.

The second main contribution of this thesis consists of novel algorithms for adaptive interpolation filtering (AIF) presented in Chapter 3. AIF is a tool that improves the coding efficiency of video coders, where the coefficients of the interpolation filter used in motion compensation are not fixed but change dynamically. Novel techniques for lowering the encoding and decoding complexity of AIF schemes are presented first. Then, a novel structure based on directional filters is presented. Thirdly, novel algorithms are presented to further improve the coding efficiency of various AIF schemes. It should be noted that, some of the developed algorithms could also be utilized for fixed interpolation schemes, where the filter coefficients are pre-defined and do not change. In addition, many of the proposed algorithms are not limited to interpolation but could be extended to adaptive loop filtering or post-filtering.

The third major contribution of the thesis is a novel parallel implementation method for decoding different views of a multiview video coding (MVC) system. Multiview video refers to a video signal describing a scene captured from more than one camera each having a different viewpoint. Therefore, in a multiview video signal there exists an additional redundancy among different view-points, called *view-redundancy*. This thesis presents novel methods specifically designed for multiview video coding to enable parallel processing of different views so that the implementation complexity of MVC decoders is reduced.

The main goal of the research work presented in this thesis is to develop novel algorithms to be used in future video coding standards. Therefore, parts of the work presented in this thesis are also proposed to and some have been adopted by international standardization bodies. Novel methods that enable parallel processing of multiple views are adopted to MVC extension of the H.264/AVC standard as a standard essential feature. The directional filters used for adaptive interpolation and their low complexity implementation is implemented to the experimental software maintained by VCEG with the aim of collecting promising tools for future video coding standards. The directional filters using fixed coefficients are also adopted into the initial design of the next generation video standardization project by MPEG and VCEG, named High Efficiency Video Coding (HEVC).

1.3. AUTHOR'S CONTRIBUTIONS TO THE PUBLICATIONS

Publication [P1] analyses the interpolation used in the H.264/AVC and presents novel encoder algorithms to generate H.264/AVC compliant bitstreams to generate a lightweight decoding operation. The original idea was conceived by the author and Jani Lainema. The author implemented the idea, performed most of the encoding simulations and wrote most of the publi-

cation. The decoder complexity analysis presented in the publication is mostly performed by Antti Hallapuro.

Novel algorithms to enable 16-bit implementation of adaptive interpolation filters are presented in Publication [P2]. The author was one of the co-inventors of the presented ideas and wrote most of the publication. Dr. Dmytro Rusanovskyy implemented the ideas in the H.264/AVC software and performed most of the simulations presented in the paper.

Publications [P3] and [P4] present various methods to improve the encoding speed of adaptive interpolation filtering. All the ideas presented in these papers are co-invented by the research team, including the author, Dr. Dmytro Rusanovskyy, Antti Hallapuro and Jani Lainema. The author implemented the idea presented in [P4], performed the necessary simulations, and wrote most of the publication. The implementation of the ideas in [P3], respective simulations and the writing of the publications is mostly done by Dmytro Rusanovskyy with assistance of the co-authors.

Publication [P5] presents a novel structure based on directional filters for adaptive interpolation filtering, and [P6] extends the concept and presents additional results utilizing fixed coefficients and also presents some methods for 16-bit implementation. The ideas presented in [P5] and [P6] are co-invented by the research team including the author, Dr. Dmytro Rusanovskyy, Antti Hallapuro and Jani Lainema. The implementation of the ideas presented in these publications, and the respective simulations are mostly performed by Dr. Dmytro Rusanovskyy with assistance of the co-authors.

Publications [P7] and [P8] present novel methods for improving the coding efficiency of adaptive interpolation schemes. In [P7], the symmetry of the filter bank used in adaptive interpolation is made flexible, rather than pre-defined. Publication [P8] extends this idea by adding support for lower complexity encoding. In addition, a novel algorithm to switch between different filter structures is proposed in [P8]. The idea presented in [P7] is invented by the author and Jani Lainema. The author implemented this idea in the H.264/AVC software, performed all the simulations and most of the writing of the publication. The ideas presented in [P8] are co-invented by the research team including the author, Dr. Dmytro Rusanovskyy, Antti Hallapuro and Jani Lainema. The implementation of the ideas and the respective simulations are performed by Dr. Dmytro Rusanovskyy. Publication [P8] is mostly written by Dr. Dmytro Rusanovskyy, with assistance of the co-authors.

Publication [P9] presents novel prediction methods specifically designed for multiview video coding to enable parallel processing of different views for multiview video coding. The idea is co-invented by the author, Jani Lainema and Miska Hannuksela. The author also wrote most of the publication. The implementation and simulation work is mostly performed by Liu Hui. Publication [P10] provides an overview of the novel features of the MVC standard. The author wrote the section on parallel processing and its relationship with system standards. Most of the remaining sections were written by Dr. Ying Chen, with assistance of the co-authors.

Chapter 2

Motion Compensated Prediction and Interpolation in Video Coding

The basic principle of Motion Compensated Prediction (MCP) is to reduce the amount of information transmitted to the decoder, by exploiting the temporal redundancy present in the video signal. In MCP, the picture to be coded is first divided into blocks, and for each block, the encoder searches the reference frame to find its best matching block. The best matching block is called the prediction of the corresponding block and the difference between the original and the prediction signal is coded by various means, such as transform coding and quantization, and transmitted to the decoder. The relative position of the prediction with respect to the original block is called the block's motion vector and it is transmitted to the decoder along with the residual signal. Upon receiving the motion information, the decoder can generate the prediction signal and by adding the received residual signal to the prediction, the decoder can form the reconstructed block. Due to errors in quantization of the residual signal, the reconstructed block and the original block are not identical.

The total number of bits transmitted to the decoder for a motion compensated block comprises the bits used to describe the motion of the block and the bits used to code the corresponding residual signal. Therefore, the coding efficiency of MCP based video coders could be increased if this total amount of bits is reduced. This could be done by designing efficient tools that can provide more accurate prediction, hence reduce the bits needed to code the residual signal. Even though these tools could increase the bits used for describing the motion, having more accurate prediction usually compensates this additional overhead. This principle is used heavily in the design of H.264/AVC video coding standard that includes many tools to improve motion prediction. These tools could be summarized as follows:

Hierarchical Partitioning of Motion Blocks: Similar to previous standards, H.264/AVC divides the picture into macroblocks of size 16x16. In addition, macroblocks are further divided into transform blocks. In H.264/AVC, each macroblock could use different partitioning to describe the underlying motion of the macroblock more accurately. For example, each 16x16 macroblock could use only one motion vector or be partitioned into a two 16x8 or 8x16 motion partitions or four 8x8 motion partitions. Furthermore, each 8x8 motion partitions could be

further divided into a two 8x4 or 4x8 partitions or four 4x4 partitions. This is illustrated in Figure 5 in more detail.

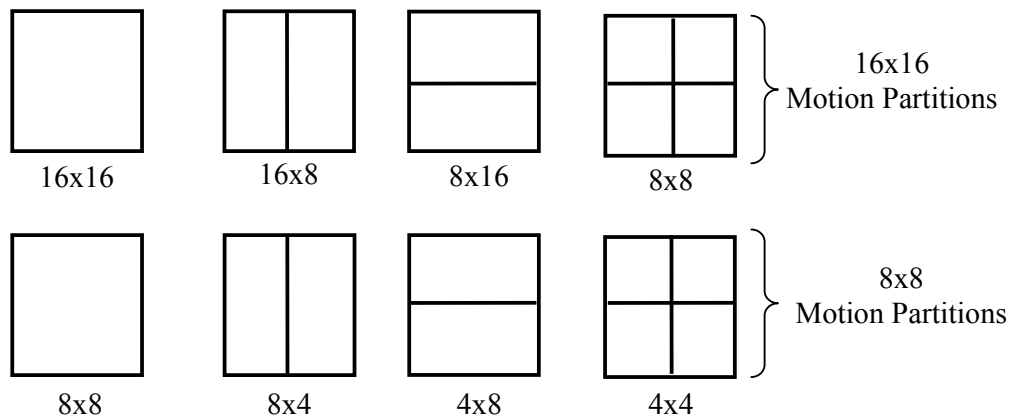


Figure 5 Motion partitions used in H.264/AVC

Multiple Reference Pictures: In H.264/AVC, more than one reference picture could be utilized in motion compensation. In this case, both the encoder and the decoder store the reference frames in a reference picture buffer to be used in motion prediction. If multiple reference pictures are used, the encoder signals the reference picture index, in addition to the motion vector, to indicate which reference pictures are used for prediction. This process is indicated in Figure 6, for the case of using three reference pictures [2].

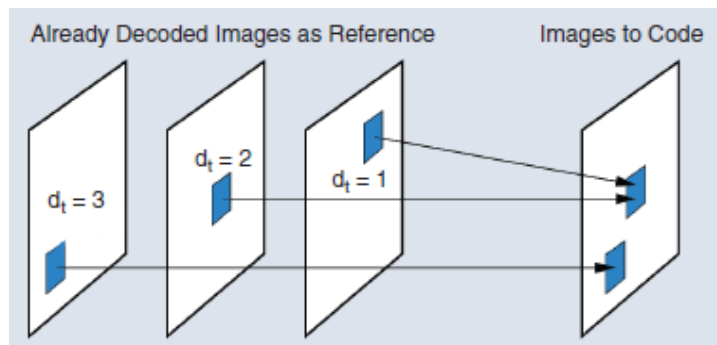


Figure 6 Illustration of multiple reference pictures [2]

Multi-Hypothesis and Weighted Prediction: H.264/AVC includes a number of tools that generate more accurate prediction by linearly weighting different prediction signals. One of these tools is called Weighted Prediction, where the reference signal is weighted by multiplying it with a scaling factor and adding an offset value. The scale and offset used in weighted prediction could be signaled explicitly to the decoder or derived implicitly using the temporal

distance of the reference frames. In multi-hypothesis prediction, the prediction signal could be generated by super-imposing multiple signals either from the same or different reference pictures [5].

Motion Vectors with High Accuracy: H.264/AVC utilizes motion vectors with quarter-pixel accuracy. If the motion vector has a fractional pixel accuracy, interpolation needs to be performed to obtain samples at fractional pixel locations. To obtain luminance sub-pixels, H.264/AVC defines 6-tap Finite Impulse Response (FIR) filters to obtain samples at half-pixel locations and bilinear filters to obtain the samples at quarter-pixel locations [6]. To obtain the chrominance sub-pixels, bilinear interpolation is used. More details regarding the luminance interpolation defined in H.264/AVC standard is given in Section 2.1.

2.1. LUMINANCE INTERPOLATION IN H.264/AVC

If the motion vector has a half or quarter-pixel accuracy, the samples at non-integer positions need to be interpolated from the original pixels at integer positions. Figure 7 illustrates the image sample grid, where the original pixels at integer locations are labeled by upper-case letters within the shaded boxes, and fractional pixels that are obtained by interpolation are indicated by white squares with lower case letters. The samples at half-pixel locations, b , h are interpolated from integer samples by applying a one dimensional 6-tap FIR filter on the integer sampled pixels. For example, the sample at half-pixel location b is obtained from the samples at integer locations $C1$, $C2$, $C3$, $C4$, $C5$, $C6$. More specifically, b is given as

$$b = ((C1 - 5 \times C2 + 20 \times C3 + 20 \times C4 - 5 \times C5 + C6) + 16) \gg 5 \quad (1)$$

The half-pixel sample at position j is interpolated by applying a separable 6-tap filter with the same filter taps given in Equation (1). In other words, in order to interpolate j , the samples at locations cc , dd , h , ee , ff and gg are interpolated first using vertical 6-tap filtering, followed by horizontal interpolation (the order of the horizontal, vertical interpolation could be changed, which would yield still the same value). Due to the additional need of interpolating other half-sample locations, interpolation of the sample at position j has the highest complexity among other half-pixel samples.

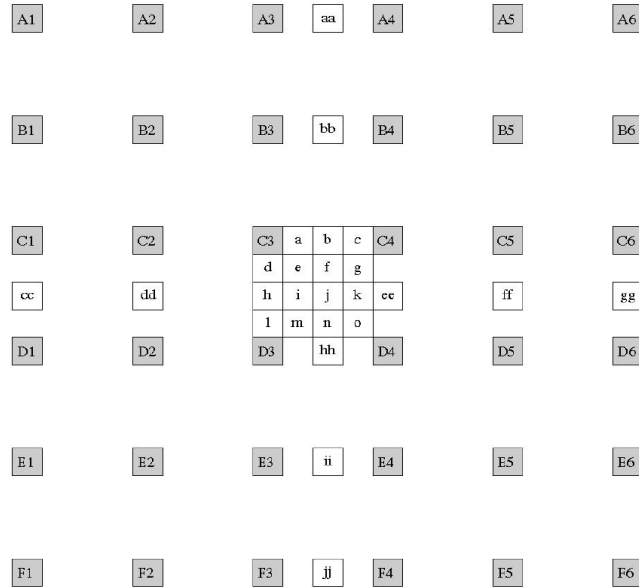


Figure 7 Luminance interpolation. Gray boxes indicate samples at integer positions and white boxes indicate fractional samples.

The samples at quarter pixel positions are obtained by averaging the two nearest samples at half or integer positions. Figure 8 illustrates the details of the quarter pixel interpolation scheme employed in the H.264/AVC standard. To obtain each quarter pixel, two values are averaged by rounding upwards. In Figure 8, quarter pixels are denoted by letters *a, c, d, e, f, g, i, k, l, m, n, o* and are placed within shaded boxes. As seen in Figure 8, the quarter-pixel sample *d* is obtained by averaging the integer pixel C3 and the half pixel sample *h*. This means that the half-pixel *h* must be interpolated first using the 6-tap filter, in order to calculate the value of the quarter pixel sample *d*. Similarly, the half-pixel sample *j* needs to be interpolated first to obtain the quarter pixels at locations *f, i, k, n*. This means that the interpolation complexities of different sub-pixels are different. This is analyzed in more detail in Section 2.2.

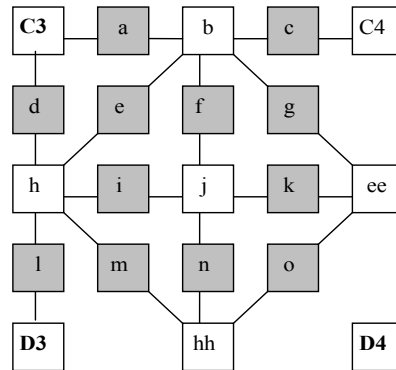


Figure 8 Quarter-pixel interpolation defined in H.264/AVC

2.2. COMPLEXITY ANALYSIS OF H.264/AVC INTERPOLATION

The complexity of the H.264/AVC standard was analyzed in [7][8][9] and it was found that the luminance interpolation is one of the most complex parts of an H.264/AVC decoder constituting 25-30% of the whole decoding complexity¹. As mentioned above, different sub-pixels have different interpolation complexities. The number of filtering operations needed to be performed for each sub-pixel is given as follows:

Sub-pixel Positions b,h : Sub-pixel positions b,h are horizontally and vertically aligned with integer pixels respectively, and they are obtained using a 6-tap FIR filter with coefficients given in Equation (1).

Sub-pixel Position j : In order to obtain the value of the sub-pixel at position j , 6-by-6 separable filtering is needed. First, the intermediate sub-pixel samples are obtained by a 6-tap horizontal (vertical) filtering and using these intermediate values the final value for the position j is obtained using a vertical (horizontal) filtering. Therefore, seven 6-tap filtering operations are required to obtain the value of the sub-pixel at position j . This complexity could be reduced by performing the interpolation in blocks and reusing the values of intermediate sub-pixel samples that are already computed, instead of performing the corresponding operations independently for each pixel. This process is illustrated in Figure 9 for the case of interpolating a 4x4 block. As seen in Figure 9, a total of 36 6-tap filtering operations are needed for obtaining the intermediate sub-pixel values and an additional 16 6-tap operations are needed, bringing the total number to 52.

Sub-pixel Positions a,c,d,l : The quarter-pixels at positions a,c,d,l are obtained by bi-linear averaging the half-pixel position b or h , and the corresponding integer sample. Therefore, a bi-linear averaging operation is required to obtain the value of sub-pixel positions a,c,d,l in addition to the 6-tap filtering operation.

Sub-pixel Positions e,g,m,o : The quarter-pixels at positions e,g,m,o are obtained by bi-linear averaging two half-pixel values that require 6-tap filtering each. Therefore, a total of two 6-tap filtering operations followed by a bi-linear averaging operation are needed to obtain each sample value of sub-pixel positions e,g,m,o .

Sub-pixel Positions f,i,k,n : The quarter-pixels at positions f,i,k,n are obtained by bi-linear averaging the two half-pixel samples. One half-pixel position needed is the middle j position and the other half-pixel position is aligned either horizontally or vertically by the integer sample. The value of the half-pixel position that is horizontally or vertically aligned to integer sample is already obtained during the interpolation of position j . Therefore, in addition to 6-tap filter-

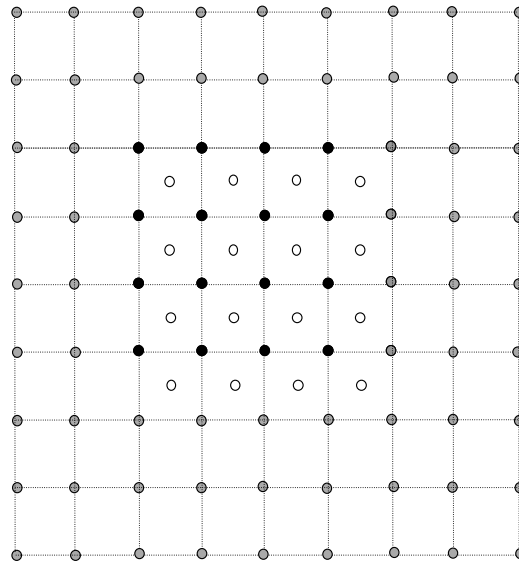
¹ It should be noted that implementation complexity may mean different things in different contexts. In this thesis, implementation complexity refers to software implementation complexity.

ing operations to obtain position j , an additional bi-linear averaging operation is needed, to obtain sub-pixel positions f, i, k, n .

Based on the analysis above, the interpolation complexity of each sub-pixel position could be summarized in Table 1, assuming the interpolation is performed based on 4x4 blocks.

Table 1 Interpolation complexity for each sub-pixel position in terms of required number of interpolation for a 4x4 block

Sub-pixel position	Total number of 6-tap filtering	Total number of bilinear filtering
a	16	16
b	16	0
c	16	16
d	16	16
e	32	16
f	52	16
g	32	16
h	16	0
i	52	16
j	52	0
k	52	16
l	16	16
m	32	16
n	52	16
o	32	16



- Integer position pixels
- Integer position pixels - extended
- 1/2 pixels to be interpolated

Figure 9 The sub-pixel interpolation process for position j .

2.3. GENERATING BITSTREAMS WITH LOW DECODING COMPLEXITY

In the previous section, it was shown that the interpolation of different quarter and half pixel positions have different complexities. Therefore, the decoding complexity depends on both the horizontal and vertical sub-pixel components of the motion vector. For example, if a motion vector has integer values for both its horizontal and vertical components, the decoder does not need to perform any interpolation to obtain the prediction signal, and hence have minimal complexity. However, the complexity would increase, if the motion vector points to locations that are difficult to interpolate, such as the half pixel location j , or any quarter-pixel location that requires the value of j .

In order to generate bitstreams that have lower decoding complexity, a novel encoder algorithm is developed in [P1] that exploits this variation in complexity. This is achieved in two stages. At the motion estimation step, the candidate motion vectors having less decoding complexity are biased using a Lagrangian based cost function. At the mode decision step, the interpolation complexity of each mode is estimated and the modes with less complexity are favored using a similar cost function.

Motion Estimation: At the motion estimation stage, the reference picture is searched for the candidate motion vectors, and the motion vector that results in the best prediction is chosen.

The H.264/AVC test model maintained by JVT chooses the best motion vector by minimizing the following cost function.

$$J(\mathbf{m}, \lambda_{\text{motion}}) = SAD(s, c(\mathbf{m})) + \lambda_{\text{motion}} \cdot R(\mathbf{m}) \quad (2)$$

with $\mathbf{m} = (mx, my)^T$ being the motion vector, and λ_{motion} being the Lagrange multiplier. The first term of the above cost function is the distortion term and it is given as the Sum of Absolute Difference (SAD) between the original signal s , and the reference signal $c(\mathbf{m})$. The rate term $R(\mathbf{m})$ represents the number of bits that would be used to code the motion vector $\mathbf{m}$.

In order to favor motion vectors with less interpolation complexity and penalize those with higher complexity, the conventional cost function is modified as follows.

$$J'(\mathbf{m}, \lambda_{\text{motion}}, \lambda_{ME}) = J(\mathbf{m}, \lambda_{\text{motion}}) + \lambda_{ME} \cdot C_{ME}(\mathbf{m}) \quad (3)$$

The proposed cost function has an additional term, $C_{ME}(\mathbf{m})$, that represents the decoding complexity of the candidate motion vector, $\mathbf{m}$. $C_{ME}(\mathbf{m})$ is basically a two dimensional array calculated using Table 1 and it is given in Figure 10. For example, if the candidate motion vector has integer-pixel accuracy in both horizontal and vertical directions, then $C_{ME}(\mathbf{m})$ is zero. Similarly, if $\mathbf{m}$ points to location j , the value of $C_{ME}(\mathbf{m})$ is 4. The value of $C_{ME}(\mathbf{m})$ is further multiplied by the Lagrangian term λ_{MEC} , to adjust the complexity–video quality trade-off. Larger values of λ_{MEC} imply less decoding complexity of the resulting bitstream and a higher penalty on the coding efficiency.

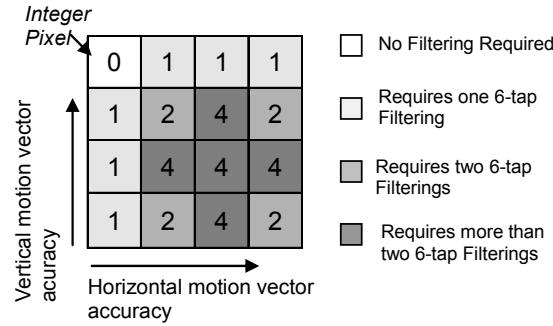


Figure 10 Approximate interpolation complexities

Mode Decision: After the motion estimation is performed, the coding results of all the candidate modes are compared. The conventional mode decision function implemented in the H.264/AVC test model chooses the best mode for the macroblock which minimizes the following Lagrangian cost function.

$$J(M, \lambda_{MODE}) = SSD(s, r) + \lambda_{MODE} \cdot R(s, c, M) \quad (4)$$

with r being the reconstruction signal for the given mode, M , and λ_{MODE} is the Lagrangian multiplier.

The first term of the above cost function is the distortion term, and it is given as the Sum of Squared Difference (SSD) between the original and the reconstructed signal. $R(s, c, M)$ is the rate term that represents the number of bits used to code the mode M . In order to favor the modes with less interpolation complexity, Equation (4) is modified as follows:

$$J'(M, \lambda_{MODE}) = J(M, \lambda_{MODE}) + \lambda_{MDC} \cdot C_{MODE}(M) \quad (5)$$

Similar to the proposed cost function used in Motion Estimation, Equation (5) has the additional term $C_{MODE}(M)$ representing the decoding complexities of candidate modes. $C_{MODE}(M)$ is the sum of all the interpolation complexities for all motion vectors involved in the candidate mode, and is illustrated below in Equation (6).

$$C_{MODE}(M) = \sum_{i=1}^{num_of_MVs} C_{ME}(\mathbf{m}_i) \quad (6)$$

where num_of_MVs refers to the number of motion vectors used in the given mode. For example, if the candidate mode is INTER_8x8, in which the macroblock is divided into four blocks and each block has its own motion vector, the value of num_of_MVs is four. $C_{MODE}(M)$ is multiplied by λ_{MDC} to adjust the complexity-quality trade-off at the mode decision stage. The values for λ_{MDC} and λ_{MEC} affect the encoding performance and should be carefully selected. It should be noted that, conventional cost function is used in the proposed method when deciding between INTER and INTRA modes.

The performance of this scheme is tested as follows. Firstly, low-complexity bitstreams for different test sequences are generated using the methods described above. Then, using a highly optimized H.264/AVC decoder implementation on an ARM-11 platform, all the bitstreams are decoded and the required numbers of processor cycles are recorded for each of them. Then, the same process is repeated using the conventional encoder with the same encoder settings, to get the reference RD Optimized bitstreams. Table 2 presents the coding efficiency and complexity reduction performance of the proposed algorithm for various sequences. Figure 11 presents the results for the test sequence Foreman. When compared to the RD Optimized bitstream, the low complexity bitstream can have up-to 14% less decoding complexity with a penalty on video quality of around 3%. On average, the total decoding complexity reduction is 6%, with 1.8% penalty on coding efficiency. It should be noted that the complexity reduction numbers presented in Figure 11 and Table 2 refer to the total decoding complexity. Since the proposed algorithm aims to reduce only the interpolation complex-

ity, the reduction in interpolation complexity is significantly higher than those reported numbers.

Table 2 Coding efficiency and complexity performance of the proposed algorithm

Sequence	Total complexity reduction	Delta bitrate
Container	-2.6%	+0.8%
Foreman	-11.8%	+3.3%
News	-4.5%	+1.8%
Silent	-7.4%	+1.9%
Paris	-5.2%	+1.6%
Mobile	-3.3%	+0.9%
Tempete	-7.5%	+2.1%
Total	-6.0%	+1.8%

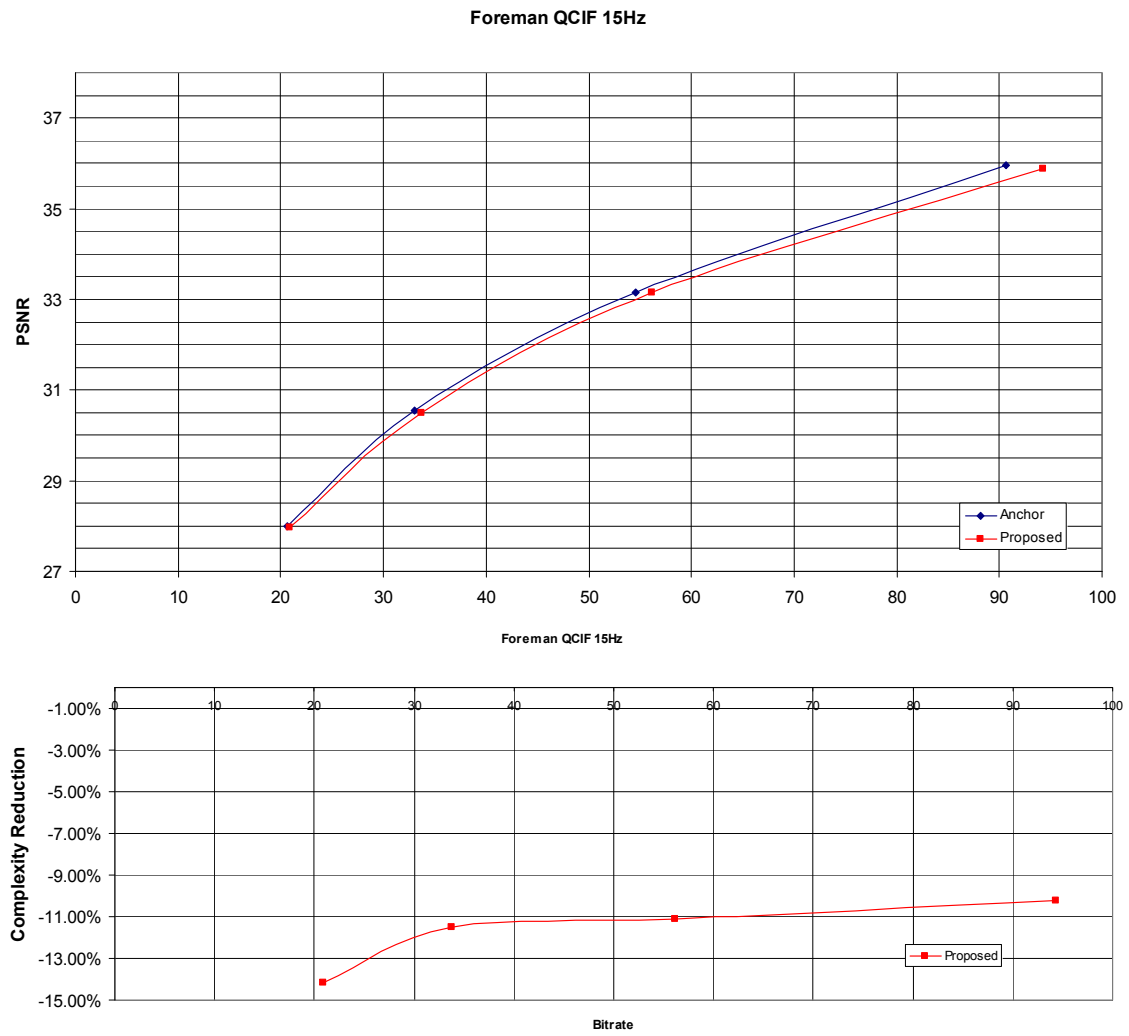


Figure 11 Analysis of the proposed method. Above graph shows the coding efficiency drop due to using the proposed method. Below graph shows the achieved complexity reduction in total decoding complexity.

Chapter 3

Adaptive Interpolation Filtering

It is known that aliasing present in the original video sequence deteriorates the coding efficiency of a video coders [9]. The interpolation filter defined in the H.264/AVC standard was designed to minimize these effects of aliasing present in the input image sequence [9][11]. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. In other words, due to many factors of the video sequence, such as change of camera parameters, change in scene, the aliasing component of the video signal changes in time. Adaptive interpolation filtering (AIF) that change the filter coefficients at each frame have been proposed in the literature to combat this non-stationary effect of aliasing [12][13][14][15][16].

3.1. FUNDAMENTALS OF ADAPTIVE INTERPOLATION FILTERING (AIF)

The basic principle of AIF is that the interpolation filter used to obtain pixel values at sub-pixel locations dynamically changes according the underlying statistics of video signal. Encoder transmits the necessary information about the interpolation filter as side information to the decoder, so that identical prediction could be achieved at the decoder side.

In [12], the interpolation filter coefficients to obtain half-pixel samples are varied at every coded picture, and fixed bi-linear interpolation is used to compute the quarter-pixel samples. In [14][15][16], the interpolation filter for both the half-pixel samples and quarter-pixel samples are varied at each picture. In addition, some studies have been performed to vary the filter not only at every picture, but also spatially within the picture [17].

AIF in video coding was first introduced by Wedi et. al as a proposal to the H.264/AVC standard [12][13]. In [12], the interpolation filter coefficients, used to obtain half-pixel samples are assumed to be changing at each frame. The quarter-pixel samples are obtained with the same bilinear filtering configuration as described in the previous subsection. The filter coefficients are obtained by solving an iterative optimization problem and had significant encoding complexity. For this reason, it was not included in the final H.264/AVC standard.

In [14], Vatis et. al. extended this idea and proposed to use 2D non-separable adaptive interpolation filter (2D-AIF). This scheme utilizes a 2D non-separable filter for each sub-pixel

and the coefficients for each filter are calculated analytically by minimizing the prediction error energy. The complexity of the 2D-AIF for the encoder and decoder are analyzed respectively in the following subsections.

3.1.1. Encoding Complexity of 2D-AIF

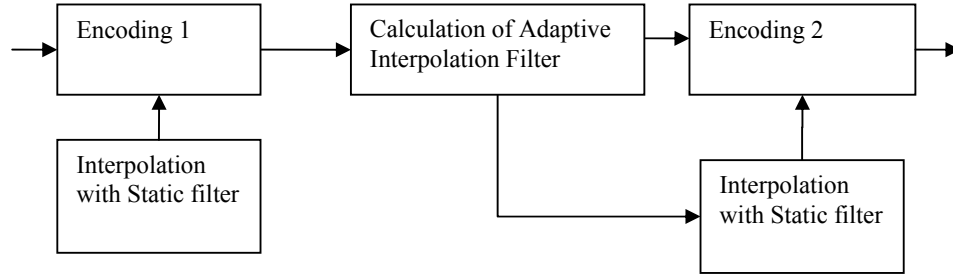


Figure 12 Flowchart of video encoder with adaptive interpolation.

The encoding process of 2D-AIF is illustrated in Figure 12, and could be summarized with the following steps.

- Step-1: Interpolate the reference frame(s) using a fixed filter.
- Step-2: Encode the current frame and obtain the residual error and motion vectors.
- Step-3: Utilizing the motion vectors and the prediction error, construct a system of linear equations by minimizing the residual error energy.
- Step-4: Solve the system of linear equations and obtain the optimal filter coefficients.
- Step-5: Interpolate the reference frame(s) using the optimal filter.
- Step-6: Final encoding of the frame using the reference frame(s) upsampled with the optimal filter.

As seen above, there is a significant encoding complexity increase associated with the adaptive interpolation due to two reasons: i) Constructing and solving the system of linear equations as it requires computing auto-correlation and cross-correlation functions that are difficult to compute (Steps 3-4-5) and i) The additional encoding pass needed for each frame (Step-6). The additional encoding pass doubles the encoding time as every frame needs to be coded twice. Regarding the additional complexity associated with filter coefficient calculation, most of its complexity is due to the need to construct the linear equations as it requires computation of correlation functions (Step-3). This is also verified by performing various simulations and measuring the percentage of the time it takes for an encoder to compute correlation functions over three steps combined (Steps 3,4,5) The simulation results are illustrated in Figure 13 for three different sequences coded with different quantization parameters. In Figure 13, the time encoder spends on constructing the linear equations is shown in terms of percentage of the three steps combined. As expected, it was found out that computing the

correlation functions takes most of the time spent in the filter calculation process (around 75-85%).

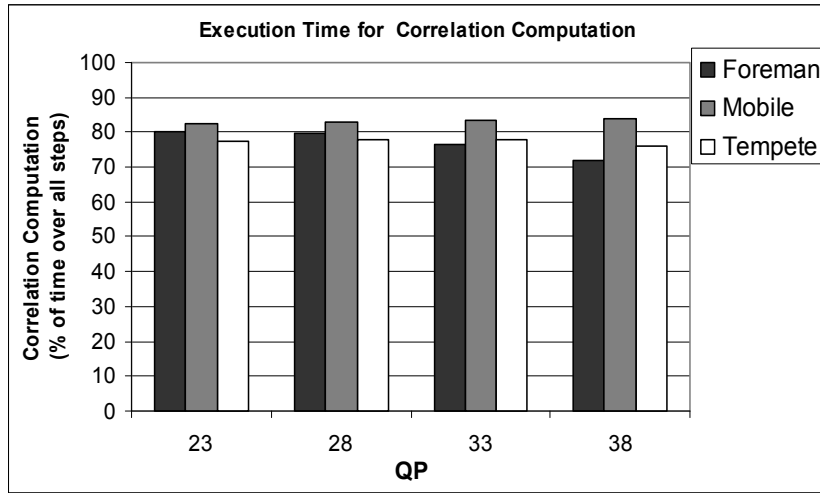


Figure 13 Execution time for computation of correlation functions in percentage of all steps combined

Novel algorithms are developed both for reducing the decoding complexity by designing a novel filter structure based on directional filters [P5][P6] and its 16-bit implementation [P2], and also the encoding complexity by designing novel encoder algorithms with high coding efficiency [P3][P4].

3.1.2. Decoding Complexity of 2D-AIF

The decoding complexity of 2D –AIF is analyzed in [18] by counting the number of operations required to interpolate each sub-pixel sample. It is found out that the interpolation complexity of 2D-AIF is approximately three times that of the fixed interpolation filter of H.264/AVC. The main reasons of this additional complexity is the need to perform 32-bit floating point arithmetic and the 2D non-separable filters utilized in the interpolation. Section 3.2 presents a novel structure and its 16-bit implementation that uses directional filters to reduce the decoding complexity significantly [P2][P6].

3.2. DIRECTIONAL ADAPTIVE INTERPOLATION FILTERING

As mentioned above, 2D-AIF has a significant increase in the encoding and decoding complexities compared to the interpolation method in H.264/AVC. The reason for the increase in the decoding complexity is that interpolating each sub-pixel requires a 2D non-separable filter of size 6x6 and the filtering operation is done in 32-bit floating point arithmetic. In order to reduce this complexity without decreasing the coding efficiency, a novel structure based on directional filters is developed.

Consider Figure 14, where the pixel locations in the original image grid (integer-pixel locations) are labeled by upper-case letters $\{A1, \dots, F6\}$ within the shaded boxes, and lower-case symbols $\{a, b, \dots, o\}$ represent locations of samples at sub-pixel image grid (sub-pixel locations) that are to be interpolated. Let us denote an interpolation filter used for each sub-pixel Sp , $Sp \in \{a, b, \dots, o\}$, with $h(Sp)$ being the interpolation filter used at Sp . The direction of these filters is determined according to the alignment of the corresponding sub-pixel Sp with the integer-pixels. The alignment for each sub-pixel and the corresponding integer samples used to interpolate each sub-pixel are given as follows:

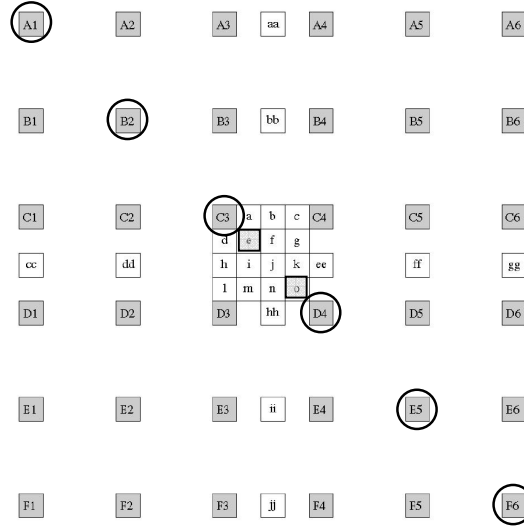
- The sub-pixel locations e, o are diagonally aligned with integer-pixels in the North-West—South-East direction. The integer-pixel positions that are aligned with these Sp are $\{A1, B2, C3, D4, E5, F6\}$, thus only these integer-pixels are utilized to estimate $h(e)$ and $h(o)$ and to interpolate sub-pixels e and o . This is illustrated in Figure 14.a.
- The sub-pixel locations g, m are diagonally aligned with the integer-pixels in the North-East—South-West direction. Therefore, integer samples $\{A6, B5, C4, D3, E2, F1\}$ are used for interpolating at sub-pixel locations g and m . This is illustrated in Figure 14.b.
- Sub-pixel locations $\{a, b, c\}$ and $\{d, h, l\}$ are horizontally and vertically aligned with integer pixels as shown by the dashed arrows in Fig 2. In this case, $\{C1, C2, \dots, C6\}$ are used for interpolating at $\{a, b, c\}$; and $\{A3, B3, \dots, F3\}$ are used for interpolating at $\{d, h, l\}$.
- Locations $\{f, i, j, k, n\}$ are either aligned with integer-pixel samples in both diagonal directions (location j), or have no integer-pixel samples to be aligned with (this is the case for positions f, i, k, n). For this reason, $\{f, i, j, k, n\}$ are interpolated using the integer-pixels that lie on a diagonal cross structure comprised of the integer samples $\{A1, B2, \dots, F6\} \cup \{F1, E2, \dots, A6\}$.

The directional interpolation structure described above utilizes sparse filters for interpolation. For example, only 6 integer samples are utilized to interpolate the value at sub-pixel position e , compared to 36 samples if a 2D separable 6x6 filter is used. This approach significantly reduces the implementation complexity as less filter taps are used in interpolation. The reduction in complexity is shown in

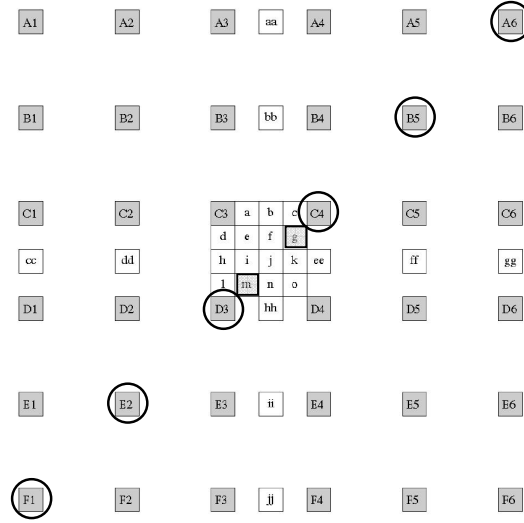
Table 3 Interpolation Complexity of various schemes (number of 32-bit airthmetic operations to interpolate a sub-pixel value)

Sub-Pixel Location	Proposed Scheme	2D AIF
{a,c,d,l}	13	13
{b,h}	10	10
{e,g,m,o}	13	58
{f,n}	19	55
{i,k}	19	55
{j}	16	41
Worst-case complexity	19	58
Average complexity	14.4	37.7

As seen on the above table, the complexity of the proposed scheme is significantly less than that of 2D AIF, both in terms of average complexity and in terms of worst-case complexity. However, the utilization of 32-bit arithmetic is still problematic for implementation. Next sub-section presents a novel method for 16-bit implementation of DAIF. The coding efficiency of DAIF is also compared with 2D AIF in the next section.



(a)



(b)

Figure 14 Illustration of DAIF structure for (a) sub-pixel locations $\{e,o\}$ and (b) sub-pixel locations $\{g,m\}$

3.2.1. 16-Bit Integer Arithmetic Implementation for DAIF

The filtering operation for the AIF schemes is done in floating point arithmetic as the filter coefficients are calculated without any restrictions with floating point accuracy. All parts of the H.264/AVC standard could be implemented using 16-bit integer arithmetic, and this need to perform 32-bit floating point arithmetic significantly increases the implementation com-

plexity of AIF schemes. In this section, a novel algorithm is presented for 16-bit implementation of AIF schemes with high coding efficiency [P2].

In order to perform interpolation with integer arithmetic, the filter coefficients should be represented by integer values with finite word length as follows:

$$H_i = \text{sign}(h_i) \cdot \lfloor |h_i| \cdot 2^Q + 0.5 \rfloor, \quad (7)$$

where $\text{sign}(\cdot)$ function returns -1 if the input is negative and +1 if the input is positive, h_i and H_i represent the floating point and integer filter coefficients and $\lfloor \cdot \rfloor$ is the floor function returning the nearest integer smaller than the input value. Q is the number of bits representing the magnitude with the leftmost bit is reserved for the sign and i represent the filter coefficient index.

Using this representation, 6-tap interpolation filtering is performed using integer arithmetic as follows:

$$Sp = \left(\sum_{i=0}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (8)$$

where Y_i are the integer samples inside the filtering mask and Sp is the sub-pixel sample to be interpolated. In order to have the convolution sum given in Equation (8) fit in a 16-bit register, the filter coefficients h_i should be represented with maximum of 4 bit accuracy ($Q = 4$). It was found empirically, that having Q equal to 4 significantly reduces the coding efficiency of DAIF.

In order to achieve a guaranteed 16-bit operation for the interpolation operation and still keep the coding efficiency high, a novel filter-specific calculation structure is developed. The proposed method achieves 16-bit operation using the following tools:

Using Partial Convolution Sums:

The proposed structure performs interpolation not using Equation (8) but using partial sums, denoted as:

$$Sp = \left(\sum_{i=0}^2 Y_i \cdot H_i + \sum_{i=3}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (9)$$

Assume that the partial sums $\sum_{i=0}^2 Y_i \cdot H_i$ and $\sum_{i=3}^5 Y_i \cdot H_i$ are stored in 16-bit accumulation registers, denoted as *res1* and *res2*, respectively. The sign of the adaptive filter coefficients are unknown, thus the accumulation registers *res1* and *res2* should support signed number representation. However, for typical interpolation filter coefficients and data samples, these

partial sums will always be positive. The reason for this can be best explained using the following example:

Consider the case where the interpolation filter is used to obtain the pixel positioned at a half-pixel sample, and further consider that there is no illumination change in the signal. This means the gain of interpolation filter is unity and the sum of filter coefficients $\sum_{i=0}^2 Y_i$ is equal to $\sum_{i=3}^5 Y_i$, which is equal to 2^{Q-1} . The value of partial sum $\sum_{i=0}^2 Y_i \cdot H_i$ could be less than zero when the negative filter coefficients are multiplied by a very large magnitude of pixel values, compared to the pixel values that are multiplied by positive pixel values. For example, for the case of H.264/AVC interpolation filter, the filter coefficients are 1, -5, 20 and Q is 5. If the pixel values, H_i are [10 100 10], then the partial sum will be -300, which is less than 0. However, in natural image signal, this rapid changing pixel values happen very rarely as neighboring pixel values are spatially correlated. That's why, the partial sums are always positive except in rare cases.

One bit accuracy for filter coefficients representation can be gained by clipping negative *res1* and *res2* (i.e. if they are negative, set their values to zero) and performing the final addition using unsigned, instead of signed arithmetic. This process is illustrated in Figure 15. As will be shown in experimental results, this clipping does not deteriorate the coding efficiency, due to two reasons. Firstly, this clipping happens very rarely in traditional video sequences. Secondly, the H.264/AVC performs clipping as well so that the final result can be represented in 8 bits. In other words, for the rare cases when the clipping operation results in loss of information, it is very likely that H.264/AVC interpolation would also lose the information.

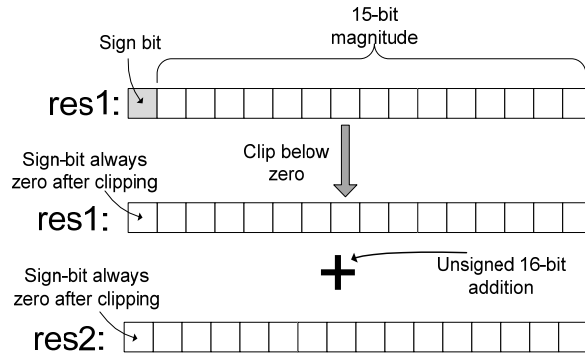


Figure 15 Illustration of intermediate clipping

Restriction on Filter Coefficients

In order for the partial sums to fit in 16-bit registers, two requirements are derived on the filter coefficient values:

$$\left(\sum_{i=j}^{j+2} \max(0, H_i) < 128 \right) AND \left(\sum_{i=j}^{j+2} \min(0, H_i) > -128 \right) \quad (10)$$

where j is 0 and 3 for a 6-tap filter. With these restrictions in place, the accuracy of the filter coefficients is increased from 4 bits to 7 bits and still guaranteeing a 16-bit integer implementation. These requirements reduce the dynamic range of filter coefficients to ensure a 16-bit operation. There are mainly two families of interpolation filters for which these requirements cannot be met. These are:

1. Interpolation filters to obtain sub-pixels that are positioned very close to integer position. For these cases, the filter coefficients become highly asymmetric and one of the filter coefficients would be significantly larger than the rest of the coefficients. This asymmetry in filter coefficients violates one of the restrictions given in Eq. 10. To illustrate this behavior, let's consider a widely used interpolation kernel that utilizes a 3-lobed Lanczos-windowed sinc function. This interpolation kernel is defined as:

$$Lanczos3(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} \cdot \frac{\sin(\pi \frac{x}{3})}{\pi \frac{x}{3}}, & |x| < 3 \\ 0, & |x| \geq 3 \end{cases} \quad (11)$$

where x is the distance between the sub-pixel and the integer samples. This kernel yields the following filter coefficients in a floating point representation to obtain a sub-pixel at position 92/100:

$$h = [0.0143, -0.0586, 0.9883, 0.0733, -0.0186, 0.0007]$$

$$H = [2, -8, 127, 9, -2, 0]$$

As seen above, the filter coefficients used to obtain a sub-pixel position very close to the integer sample does not fulfill the restriction given in Eq. 10.

2. Interpolation filters with a gain larger than unity. Consider the case again of using the interpolation filter defined in H.264/AVC using 7 bits of accuracy. But instead of having unity gain, let us assume the gain is 2. The filter coefficients in integer form is given as:

$$H_{AVC} = [8, -40, 160, 160, -40, 8]$$

These filter coefficients do not meet the restrictions defined above. However, utilizing interpolation filters with a gain that is very different than unity in video coding happens in rare cases. Those rare cases are when the sequence is fading to a white or a black picture, drastically changing the average brightness of the entire scene, and for those cases the video coder may choose to use different tools, such as Weighted Prediction.

For the rare cases, where these restrictions can not be met, the standard H.264/AVC interpolation filter coefficients are used for the adaptive filter.

Filter Coefficients with Variable Accuracy

Additional bit accuracy for the filter coefficients is gained by observing the fact that the statistical distribution of the coefficients of different sub-pixels is different. If a sub-pixel is interpolated using a 6-tap filter, the statistical distribution of the coefficients would be different than that of the 12-tap interpolation filter. For example, the coefficients of a 12-tap filter most often fall in the range $[-0.5, 0.5]$, instead of $[-1, 1]$. We exploit this fact and restrict the filter coefficients of sub-pixel locations that use a higher number of taps. By using a restricted range, additional accuracy is gained for the respective sub-pixel locations, which increases the coding efficiency of 16-bit implementation and helps to bring it back to 32-bit level, as will be shown in the experimental results.

Adaptive Rounding of Filter Coefficients

The rounding operation given in Eq. 7 is not optimal in terms of minimizing the total rounding error of the interpolation filter. For some filters, the total rounding error accumulates to a large value, and hence penalizes the coding efficiency. To alleviate this effect, a correction factor is added to a number of filter coefficients after the rounding operation to reduce the total rounding error. This is done as follows. After the integer filter coefficients are computed using Equation (7), the total rounding error of the filter is found using:

$$rnd_error = \sum_{i=0}^S (h_i \cdot 2^Q - H_i) \quad (12)$$

If the total rounding error is positive, this means that most of the coefficients were down-rounded. +1 is added to the coefficient that has the largest down-rounding error, then to the coefficient that has the second largest down-rounding error etc. This is repeated until the magnitude of the total rounding error falls below 0.5. The same process is done using the correction factor -1 for negative rounding errors. After this process, the modified coefficients have a higher rounding error, but the total rounding error of the filter is decreased.

In order to illustrate how adaptive rounding works, consider the following filter obtained using the Lanczos windowed sinc interpolation kernel, given in Equation (11)

$$h = [0.0034, -0.0435, 0.1716, 0.9481, -0.1064, 0.0251]$$

For the case of regular rounding (rounding using Equation (7)), filter coefficients in integer form with a 7-bit accuracy are given as:

$$H = [0, -6, 22, 121, -14, 3]$$

The total rounding error is found to be 1.7897 and the rounding error for each coefficient is found as:

$$rnd_error_H = [0.4379, 0.4356, -0.0413, 0.3579, 0.3872, 0.2125]$$

As seen above, the rounding error of all the coefficients is less than 0.5, but the total rounding error is 1.7897. Using the above procedure, +1 is firstly added to the filter coefficient that has the largest down-rounding error, which is the first coefficient. Then the total rounding error decreases to 0.7897. +1 is then added for one more filter coefficient that has the second largest down rounding error, and that is the second filter coefficient. This way the total rounding error becomes -0.2103. The modified filter coefficients and their respective rounding errors are then given as:

$$\begin{aligned} H' &= [1, -5, 22, 121, -14, 3] \\ rnd_error_{H'} &= [-0.5621, -0.5644, -0.0413, 0.3579, 0.3872, 0.2125] \end{aligned}$$

As seen clearly in the above example, the magnitude of the total rounding error of the filter is decreased from 1.7897 to 0.2103, even if some of the filter coefficients have a higher rounding error. The frequency response of the two rounding methods, rounding to nearest integer given in Equation (7) and the proposed rounding as described above are presented in. As seen in Figure 16, rounding to the nearest integer value changes the frequency response of the filter significantly, but the frequency response comes closer to the original one when the proposed rounding is used. More important than the change in frequency response, adaptive rounding corrects the gain of the interpolation filter. In the above example, the original filter with floating point accuracy has unity gain; whereas filter h obtained using rounding to the nearest integer has a gain of 0.984375. In video coding, this change in gain means the encoder has to code more residual and degrade its coding efficiency. After adaptive rounding is applied, this gain is moved back to unity, which would help the video coder improve its coding efficiency.

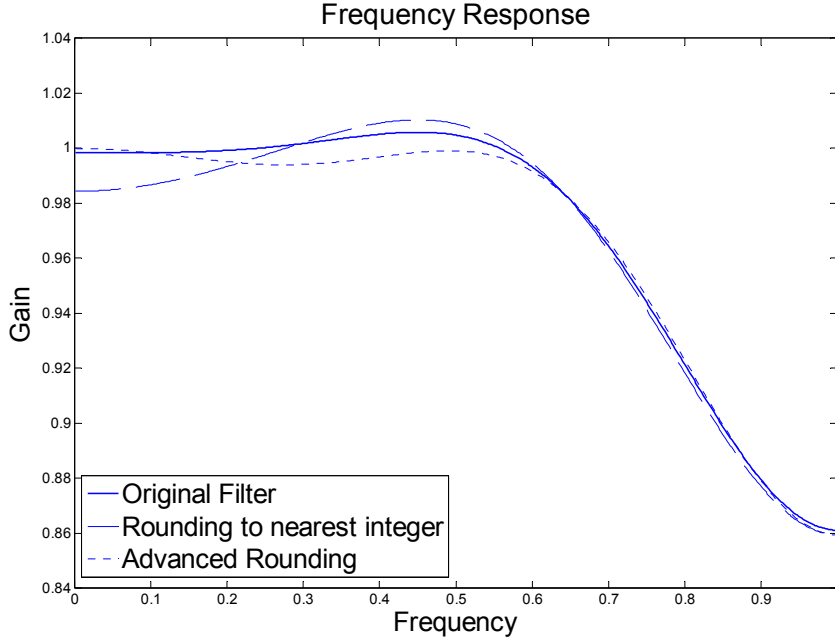


Figure 16 Frequency response of filters with different rounding methods.

In order to analyze the performance of the proposed rounding scheme, let's use the same lanczos-windowed sinc function given in Equation (11). Each sub-pixel position is interpolated first using floating point filter coefficients. Then the same sub-pixel position is interpolated using integer filter coefficients obtained with rounding to the nearest integer, as given in Equation (7). Finally, the corresponding sub-pixel positions are interpolated using the filter coefficients obtained by the proposed rounding operation. The PSNR between the floating point interpolation and the two respective integer interpolations is plotted in Figure 17. As seen in Figure 17, the proposed rounding operation constantly improves the interpolation quality compared to rounding to the nearest integer scheme. For some sub-pixels the difference in PSNR is significantly high, and exceeds 1 dB.

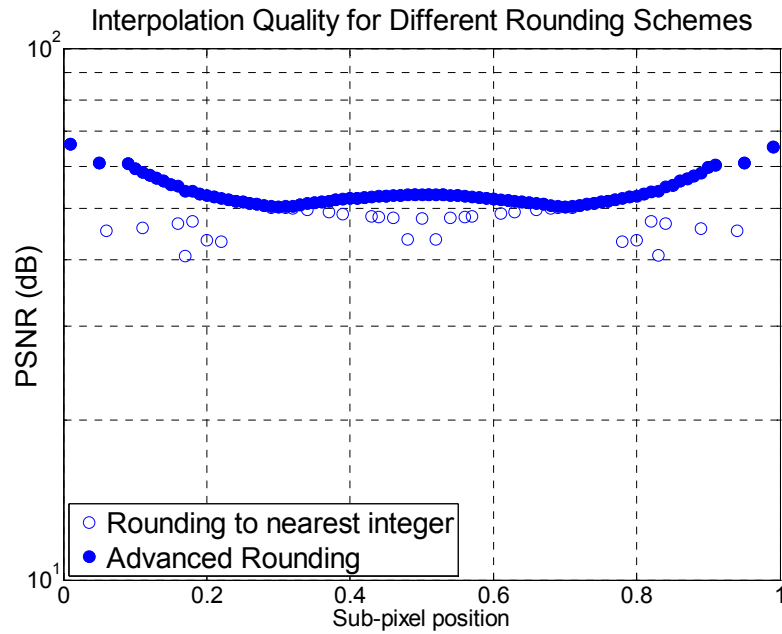


Figure 17 Effect of rounding on interpolation quality

Table 4 presents the simulation results of the Directional Adaptive Interpolation (DAIF) and its 16-bit implementation. The simulation results are performed using the conditions recommended by the ITU-T/VCEG group [28] with the baseline profile settings. Four interpolation schemes were compared; two integer implementations (32-bit and 16-bit) of the proposed DAIF scheme, noted as DAIF-32 and DAIF-16, respectively; the scheme with 2D non-separable adaptive filters (2D-AIF) [14] and the standard H.264/AVC interpolation [2], which was used for benchmarking.

Table 4 Simulation Results of the Proposed DAIF schemes comparing to the H.264/AVC Interpolation.

Resolution	Sequence	Delta Bit-rate, %		
		<i>2D-AIF</i>	<i>DAIF-32</i>	<i>DAIF-16</i>
QCIF	Container	-7.19	-8.06	-9.06
	Foreman	-1.82	-2.27	-2.1
	Silent	0.34	0.31	-0.09
CIF	Paris	-1.07	-0.86	-0.92
	Foreman	-6.63	-5.67	-5.46
	Mobile	-5.33	-1.95	-2.04
	Tempete	-3.6	-1.6	-2.09
720p	City	-18.22	-16.74	-16.18
	BigShips	-12.05	-11.61	-11.57
	ShuttleStart	-19.33	-18.33	-18.55
	Crew	-18.82	-18.68	-19.14
	Night	-6.86	-6	-5.9
Average		-8.38	-7.62	-7.76

As seen in the above table, the DAIF scheme provides an average coding efficiency gain over H.264/AVC using around 8% less bitrate (the savings in bitrate go as high as 19%). The gains of DAIF are comparable to 2D-AIF, whereas DAIF needs significantly less implementation complexity compared to 2D-AIF. The gains for both techniques are more visible at higher resolution sequences. This is because high resolution sequences need higher bitrate, which increases the bit-budget to transmit the filter coefficients. At low resolutions, the bit overhead associated with transmitting the filter coefficients becomes very high, which reduces the coding efficiency. It should also be noted that the coding efficiency of 16-bit implementation of DAIF is on average very similar to that of 32-bit DAIF. On rare cases, there is even a slight improvement in coding efficiency. This is due to the fact that the dynamic range of filter coefficients is less in 16-bit case, which reduces the overhead of coefficient coding.

The directional adaptive interpolation filter and its low complexity implementation using 16-bit integer arithmetic were presented in ITU-T/VCEG standardization group and it was adopted to the Key Technological Area (KTA) experimental software that is developed with the aim of collecting promising tools for the next generation video coding standards [S1][S2][S3][23]. The directional interpolation filter structure is then extended with fixed filter coefficients and was also adopted to the KTA experimental software [S4] [23]. In addition, directional filter with fixed coefficients was also included in the initial High Efficiency Video Coding (HEVC) standardization effort of MPEG and VCEG [S6][39].

3.3. LOW COMPLEXITY ENCODING METHODS FOR AIF

As described in Section 3.1.1, using adaptive interpolation filters increases the encoding complexity due to the following two reasons: i) constructing the system of linear equations and ii) additional encoding pass needed for each frame. This section presents novel methods to reduce the encoding complexity of AIF schemes by reducing the complexities of these two sources. Firstly, a novel method to reduce the complexity due to the need of constructing the system of linear equations is presented in Section 3.3.1 and secondly, Section 3.3.2 presents a novel method to restrict the number of passes used in adaptive interpolation.

3.3.1. Efficient Calculation of Filter Coefficients with Distortion Modeling

The coefficients of adaptive interpolation are computed by solving a system of linear equations that are constructed by computing the cross correlation and auto-correlation functions using the original frame, reference frames and the motion vectors found at the first encoding pass. Constructing these equations is a computationally demanding task due to the need to perform many multiplication and addition operations for every motion vector with sub-pixel accuracy. For example, the computation of the correlation function for a given sub-pixel position requires more operations than interpolating the corresponding sub-pixel.

To reduce the encoding complexity due to the need to perform correlation operation, a novel algorithm is designed [P4]. The proposed algorithm aims to reduce this complexity by using only a subset of the data within the frame, and hence reduce the total number of computationally complex correlation calculations. This is achieved by first identifying motion blocks that do not suffer from prediction errors due to aliasing and motion estimation mismatch, and marking them redundant. Since those redundant blocks do not suffer from aliasing and motion estimation mismatch as much as other blocks, it is assumed that excluding them in the adaptive interpolation calculation process would not hurt coding efficiency. This identification process is done at the first encoding pass, by comparing the prediction error of each block against a threshold value that is dynamically computed for every frame of the video sequence. In order not to add any additional complexity, blocks are marked as redundant using the cost values that are already computed in the motion estimation and mode decision stages. If a block is marked redundant, fractional pixels within the block are excluded in the filter calculation process. The condition to mark a 4x4 block, as redundant is given as:

$$SAD_{4 \times 4}(i) < D_1(Q) + D_2(N) \quad (13)$$

where $SAD_{4 \times 4}(i)$ is the motion prediction error of the i^{th} block given as the Sum of Absolute Differences (SAD). N denotes the frame index and Q the quantization step size. It should be noted that the block size of 4x4 is chosen to match the smallest motion block partition supported by the H.264/AVC standard. In Equation (13), there are two terms contributing to the distortion threshold. The first term depends only on the quantization step-size and its purpose

is to remove the effect of quantization in motion prediction signal, so that the proposed algorithm provides reliable results at all quantization step sizes. The second term achieves a stable coding efficiency-complexity trade-off over the entire sequence. These two terms in Equation (13) are derived as follows.

Error Threshold due to Quantization

In motion compensated prediction, Motion prediction errors may result from three possible sources, i) quantization of the reference signal, ii) aliasing present in the original signal and iii) errors in displacement estimation [6]. The redundant blocks need to be identified where the majority of the prediction mismatches are due to aliasing and displacement estimation errors, and not due to quantization. For this purpose, an empirical model is developed that relates motion prediction error of blocks to quantization step size using statistical analysis, to estimate the prediction error due to quantization and compute the distortion threshold $D_I(Q)$.

The relationship between prediction error and quantization has been well studied in rate control context, where the goal is to estimate the quantization step size given the target bits and the prediction error. One well known rate control algorithm developed for MPEG-4 utilizes a second-order Rate-Distortion model for this purpose [19]. In this context, a model is needed that relates the prediction error to quantization step size, without requiring the use of bitrate information. Therefore a similar second order model as in [19] could be used with a slight modification so that the rate term is not included. The model used is given by:

$$D_I(Q) = a_1 \cdot Q^2 + a_2 \cdot Q + a_3 \quad (14)$$

where $D_I(Q)$ is the prediction error due to quantization of the reference signal, Q is the quantization step size and a_1, a_2, a_3 are the model parameters that are calculated empirically using many test sequences with different motion characteristics using a wide-range of quantization step sizes. The model parameters a_1, a_2, a_3 are calculated using the following statistical analysis. A test sequence is first encoded and the prediction error of every 4x4 motion block is recorded. According to the resulting prediction errors, a threshold is calculated, which results in M% of blocks having less prediction error than the threshold value. This threshold is indicated as $SAD_{4 \times 4}Threshold(Q, M)$, and it is calculated for many test sequences with different motion characteristics using a wide-range of quantization step sizes. It should be noted that, the percentage M controls the trade-off between encoding complexity reduction and coding efficiency. After these thresholds are calculated for all the sequences and quantization step-sizes, the model parameters are found by fitting to data to the second order polynomial defined by Equation (14) using least-squares method.

This process is illustrated using the following example. Let's consider the test sequence Foreman, which is encoded with varying quantization step sizes and the resulting prediction errors of 4x4 blocks are recorded. For each of the quantization step sizes, the threshold values $SAD_{4 \times 4}Threshold(Q, M)$ are found using the above described process. Figure 19 pre-

sents the threshold values plotted against the quantization size for two values of M , 50% and 80%. Circles in Figure 19 indicate the threshold values for M equal to 50% and pluses indicates threshold for M being equal to 80%. After these data are computed, the second order polynomial is found and shown with dotted lines in Figure 19.

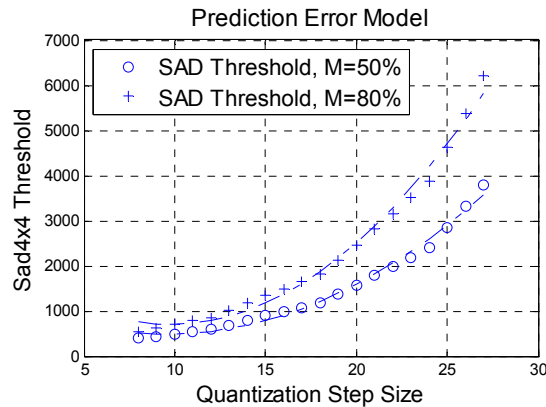


Figure 18 Prediction error due to quantization. Dotted lines indicate the calculated error model.

Dynamically Modifying the Error Threshold

The distortion threshold used to determine redundant blocks needs to be dynamically computed for every frame of the sequence. This is because the motion characteristics of a sequence can vary dramatically from one segment to another, and can significantly affect the performance of the proposed scheme. Consider the case where a certain segment of the video has relatively low motion, which results in lower than usual prediction error for the blocks. In this case, most or all of the motion blocks are identified as redundant and excluded in the filter calculation process. This results in significant degradation of coding efficiency for that segment of the video due to the fact that the adaptive filter, which is calculated by excluding majority of the data within the frame, would not be optimal.

This problem is solved in the proposed algorithm by adding the second term in Equation (13) to provide a stable trade-off between coding efficiency and complexity reduction throughout the entire sequence. $D_2(N)$ is the distortion threshold computed for every frame with frame index N , and it is independent from the quantization step size. $D_2(N)$ is computed as follows. After the encoding of a frame with index $N-1$ is finished, the resulting motion prediction error of each 4×4 motion block is recorded and $D_2(N)$ is calculated so that $M\%$ of the blocks in frame $N-1$ has less prediction error than the value of $D_2(N)$. This way, if a high motion segment starts in a video sequence, the value of $D_2(N)$ increases, which in turn increases the final distortion threshold and the same complexity reduction-coding efficiency trade-off is achieved throughout the sequence.

The encoding complexity and the coding efficiency of the proposed algorithm are compared with [14], where the encoder does not utilize the proposed encoding algorithm. The results are shown in Table 5, where $\Delta Q\%$ indicates the percentage of complexity reduction

estimated using the number of fractional pixels used in the filter calculation process, and Δ PSNR refers to the coding efficiency difference with respect to [14] calculated using Bjontegaard Delta PSNR method [20]. In Table 5, the low complexity bitstreams are generated using M as 50% and lower complexity bitstreams are generated using M being 80%.

As seen in Table 5, the proposed algorithm achieves practically the same coding efficiency with 83% less encoding complexity on average. It can also be seen that for some sequences, a small gain in coding efficiency is realized. This is because adaptive interpolation filter is optimized to certain areas of the image that suffer the most from aliasing and motion prediction errors. This is also inline with the conclusions drawn in [17], where the interpolation filter is optimized locally inside a frame, using several encoding passes.

Table 5 Simulation results of low complexity AIF encoding

	Low Complexity		Lower Complexity	
	$\Delta Q, \%$	$\Delta PSNR$	$\Delta Q, \%$	$\Delta PSNR$
Container	53.13	0.02	93.63	0.08
Foreman	31.93	0	76.19	-0.01
ForemanCIF	46.97	0	84.90	-0.05
Mobile	22.71	0	58.87	0.02
Tempete	29.50	0	83.70	-0.05
ShuttleStart	71.14	0	97.28	-0.02
City	24.41	-0.02	86.40	-0.03
Average	39.97	0.00	83.00	-0.01

3.3.2. Adaptive Interpolation with Restricted Second Pass

As discussed above, AIF schemes have high encoding complexity mainly due to the need to construct a system of linear equations to compute the coefficients of the adaptive filter and the additional encoding pass with the newly calculated filter. The complexity of the filter calculation process could be reduced by using data from a subset of the macroblocks using the method described in the previous subsection. In this section, two novel methods to reduce the complexity of AIF due to the additional encoding pass are presented.

First algorithm assumes that the motion information obtained by the static filter in the first pass is highly correlated with the motion information obtained by the adaptive filter in the second pass. This assumption is verified in the next section by collecting statistics from various sequences. By exploiting this correlation, very lightweight motion estimation could be performed in the second encoding pass by reusing many of the coding results of the first pass. The second algorithm assumes that the effect of aliasing has a content depend nature and does not undergo a fast temporal variation. Using this assumption the second encoding pass is not performed after the adaptive filter is calculated. Instead the filter coefficients of the previous frame are used to code the subsequent frame. This way, the second pass is completely eliminated and a single-pass coding is achieved using adaptive interpolation.

Coding with Restricted Second Pass

This algorithm assumes that the motion information obtained using the static filter in the first pass is highly correlated to the motion information obtained using the adaptive filter in the second pass. This assumption is based on the fact that changing the interpolation filter only changes the values of the sub-pixel samples in the reference frame; therefore, the integer component of the motion vector stays the same in the first and second pass. By exploiting this assumption, full motion estimation is not performed in the second encoding pass, but the fractional component of the motion vectors obtained in the first pass are refined. This refinement is illustrated in Figure 19. Firstly, eight half-pixel locations neighboring the integer motion vector from the first pass are searched then the eight quarter-pixel locations neighboring the best half-pixel location are searched to find the best motion vector.

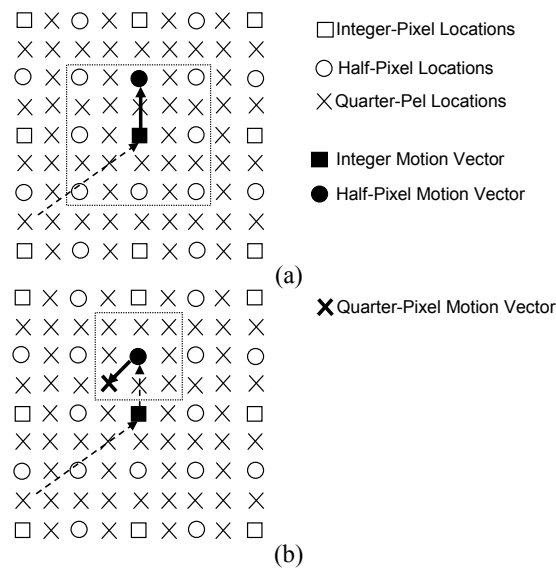


Figure 19 Refinement of fractional component of motion vectors (a) half-pixel refinement, (b) quarter-pixel refinement

It should be noted that practical motion estimation algorithms adapt the motion search range based on many factors, including the results of the sub-pixel motion estimation. Therefore, it is possible that the motion estimation in the first and second pass also yield different motion vectors in terms of their integer component. Extensive simulations are performed to see what percentage of the motion vectors actually differ also in integer component and it is found out that around 10% of the cases the integer component of motion vectors also change. The results of this experiment are illustrated in Table 6.

Table 6 Motion vector variations between first and second pass of encoding

Sequence	<i>No Change of MV's (%)</i>	<i>Fractional Components Change (%)</i>	<i>Complete Change of MV's (%)</i>
Container	79.49	19.96	0.55
Foreman (QCIF)	55.89	28.63	15.49
Paris	79.73	14.60	5.67
Foreman (CIF)	47.22	29.26	23.52
Mobile	76.82	16.40	6.78
Tempete	71.17	25.24	3.59
BigShips	48.97	42.47	8.55
City	62.71	24.76	12.52
Crew	34.46	45.47	20.07
Night	49.09	32.18	18.73
ShuttleStart	52.63	33.12	14.25
Average	59.83	28.37	11.7

Single-Pass Coding with Adaptive Interpolation

The first method described above reduces the encoding complexity by performing lightweight motion estimation in the second encoding pass. The second proposed algorithm reduces the encoding complexity even further by completely removing the second pass. This is achieved based on the observation that in majority of the cases content does not change dramatically from frame to frame, thus the effect of aliasing stays almost constant between subsequent frames. Thus, it is assumed that an adaptive filter computed for frame N, could accurately compensate the effect of aliasing for frame N+1. Experimental results show that, for sequences where the content does not change very fast between frames, the proposed algorithm achieves similar coding efficiency as using two passes. However, it is also noted that for the sequences where content is changing fast between frames, the coding efficiency of this algorithm suffers relatively more. On average, the encoding complexity is reduced by 65% with 3% increase in bitrate.

The detailed experimental results summarizing the coding efficiency and encoding time are given in Table 7 and Table 8 for both coding with restricted second pass technique (denoted as Fast1 in Table 7 and Table 8) and for single-pass coding technique (denoted as Fast2 in Table 7 and Table 8). Experimental results show 40% reduction in encoding complexity on average is achieved, with minor decrease in coding efficiency (around 0.01% change in coding efficiency) if only coding with restricted second pass technique is used. For the case of using single-pass AIF, encoding complexity is reduced further to 60% but with a higher penalty on coding efficiency (more than 3% increase in bitrate). The loss in coding efficiency is more visible at video sequences with fast movement, where the interpolation filter from previous frame is not very useful for the current frame.

Table 7 Coding efficiency of the proposed algorithms with respect to H.264/AVC interpolation filter

Resolution	Sequence	Delta Bit-rate, %		
		<i>DAIF</i>	<i>Fast1</i>	<i>Fast2</i>
QCIF	Container	-10.24	-10.02	-3.37
	Foreman	-6.23	-6.34	1.18
CIF	Paris	-0.96	-0.89	1.6
	Foreman	-5.98	-5.81	-1.52
	Mobile	-3.33	-3.32	1.01
	Tempete	-2.19	-2.16	1.05
720p	BigShips	-12.15	-11.95	-10.79
	City	-15.41	-15.26	-13.13
	Crew	-13.32	-12.74	-8.15
	ShuttleStart	-11.4	-10.9	-9.49
	Night	-4.53	-4.35	-3.05
Average		-7.76	-7.61	-4.06

Table 8 Encoding complexity of the proposed algorithms in percentage of H.264/AVC reference encoding

Resolution	Sequence	Delta Time, %		
		<i>DAIF</i>	<i>Fast1</i>	<i>Fast2</i>
QCIF	Container	121.52	60.33	22.68
	Foreman	174.37	114.96	63.46
CIF	Paris	126.53	64.65	24.17
	Foreman	173.66	110.12	60.02
	Mobile	152.63	86.12	55.69
	Tempete	167.88	103.61	73.85
720p	BigShips	155.26	84.65	64.95
	City	168.05	99.79	76.69
	Crew	156.10	86.71	48.36
	ShuttleStart	167.19	101.67	56.17
	Night	149.24	84.76	44.54
Average		155.68	90.67	53.69

3.4. CODING EFFICIENCY IMPROVEMENTS FOR ADAPTIVE INTERPOLATION

The previous subsections presented several novel algorithms to reduce both the encoding and decoding complexity of adaptive interpolation filtering. In this subsection, several algorithms are presented to improve the coding efficiency of AIF schemes. The first algorithm proposes using interpolation filters whose symmetry could be adapted at each frame [P7], while the second algorithm extends this idea, by defining a more flexible approach and also presents flexible filter structures [P8].

3.4.1. Adaptive Interpolation with Flexible Symmetry

At the time this algorithm was developed, the Directional Adaptive Interpolation Filtering (DAIF) described in Section 3.2 has not been developed yet. Therefore, the below description of the algorithm is done for the 2D-AIF case. However, it should be noted that the same algorithm could be applied to other filter structures including DAIF.

Consider Figure 7, where the pixels at integer positions are labeled by upper-case letters within the shaded boxes, and the other symbols represent sub-pixel positions to be interpolated. For each fractional pixel position, an interpolation filter is defined. For the 2D-AIF case [14], a one dimensional 6-tap filter for positions a, b, c, d, h, n that are horizontally or vertically aligned with an integer pixel are used (horizontal filter for positions a, b, c , and vertical filter for positions d, h, n). For other positions, two-dimensional a non-symmetric 6x6 tap filter is utilized. This scheme results in 6 independent 1-D filters and 9 independent 2-D filters, resulting in 360 coefficients, which is a significant overhead to be transmitted to the decoder.

In order to decrease this overhead, it was proposed to use symmetrical filters as follows [14]. The same filter coefficients are used in case the distance of the corresponding full-pixel positions to the current fractional-pixel position is the same. It is also assumed that the distance in the horizontal and vertical directions are treated equal. In other words, this assumption could be characterized as an interpolation filter with horizontal, vertical and diagonal symmetry, and is denoted as *HVD-Filter*.

With the symmetry assumption in place, the number of coefficients needed for defining the interpolation filter is reduced in two ways. Firstly, the number of filters to be transmitted is reduced from 15 to 5 (only filters for positions a, b, e, f, j need to be transmitted). Also, a smaller number of coefficients are needed for each filter (for example, only 3 coefficients are needed to define the 1-D 6-tap filter for sub-pixel location b as horizontally symmetric filter is being used). The proposed algorithm is motivated from the fact that this symmetry assumption results in transmitting fixed number of coefficients and hence cannot provide the best trade-off between the accuracy of the interpolation filter and the overhead used to code the filter coefficients for all resolutions and bitrates. In addition, the statistical properties of the video frames may not be stationary, and the symmetry assumption for the interpolation filter may not hold for every frame of the video. Some frames of a video may contain complex scenes for which neither horizontal nor vertical symmetry holds, whereas some other frames could be relatively simpler and both horizontal and vertical symmetry assumptions are valid.

In order to exploit the varying statistical characteristics of a video sequence and provide a better trade-off for high quality-high resolution material, several filters are defined with different characteristics. Each filter holds a different symmetry type, and is defined with a different number of coefficients. More specifically, the following filters are proposed:

- Filter with Horizontal Symmetry (*HOR-Filter*): Image signal is assumed to have only horizontal, but no vertical or diagonal symmetry.
- Filter with Vertical Symmetry (*VER-Filter*): Image signal is assumed to have only vertical, but no horizontal or diagonal symmetry.

- Filter with Horizontal and Vertical Symmetry (*H+V-Filter*): Image signal is assumed to have horizontal and vertical, but no diagonal symmetry.
- Filter with No Symmetry Assumptions (*FULL-Filter*): Image signal is assumed to have no horizontal, vertical and diagonal symmetry.

Figure 20 illustrates the number of filters used for each one of the above symmetry assumptions. In Figure 20, each square represents a sub-pixel pixel position, and the number inside each square represents the identification number of the interpolation filter used for the corresponding sub-pixel position. Sub-pixel positions with the same identification number share the same interpolation filter (for example, for the *HVD-Filter*, the sub-pixel locations a, c, d, n use the same filter that is denoted by ID number 1). The coefficients of the filter that are shared by more than one sub-pixel positions are transmitted only once. Coefficients are transmitted only for the shaded sub-pixel positions as denoted in Figure 20, whereas for other positions denoted as white squares no coefficients are transmitted.

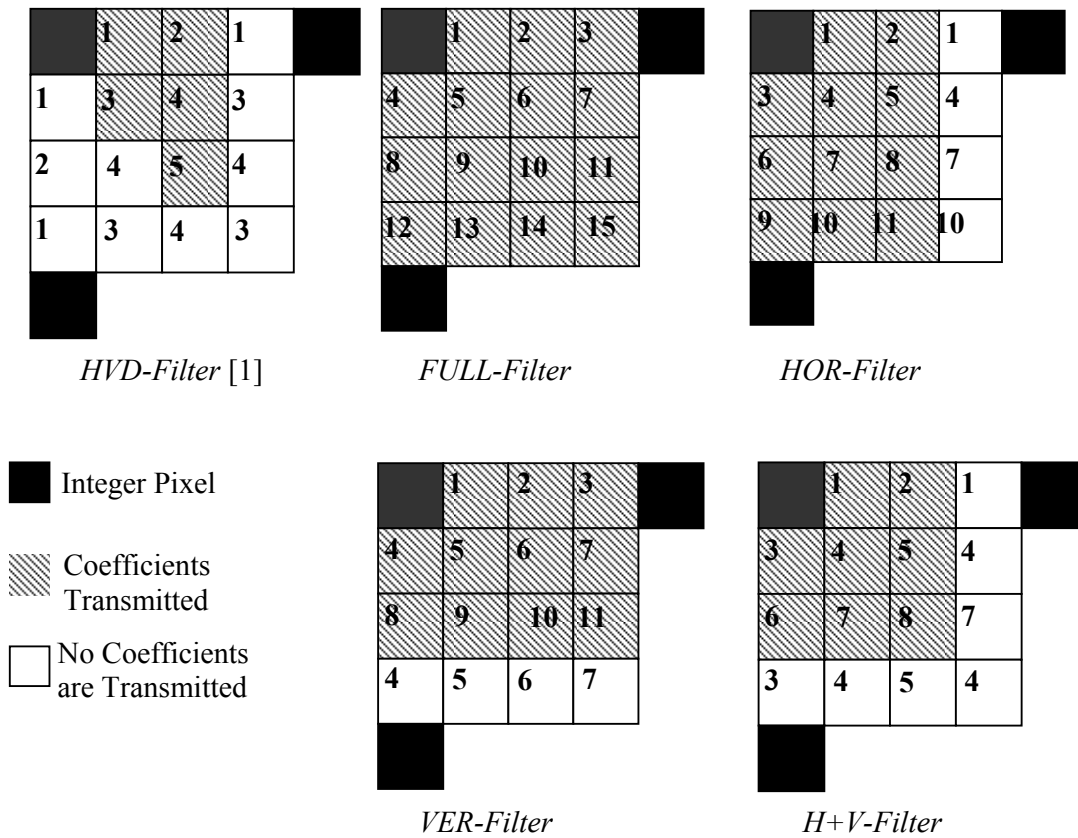


Figure 20 Number of independent filters for different symmetries

In addition to the number of filters, the number of coefficients used for each filter depends on the filter's symmetry assumption. For example, consider *HOR-Filter* where only horizontal symmetry is assumed. In this case, sub-pixel position b uses a symmetric horizontal filter, whereas position h uses a 6-tap non-symmetric vertical filter. Similar to *HVD-Filter*, all filter types except the *FULL-Filter* use a 1D filter for sub-pixel positions that are horizontally or vertically aligned with an integer position (sub-pixel positions a, b, c, d, h, n). For the *FULL-Filter* these sub-pixel positions do not use a 1D filter, but a 2D filter. It is known that using a 2D filter is not necessary to remove the aliasing effect for positions aligned with the integer pixel. However, using a 2D filter for those positions is useful to combat the adverse effects of quantization and motion estimation errors that also decrease the prediction quality.

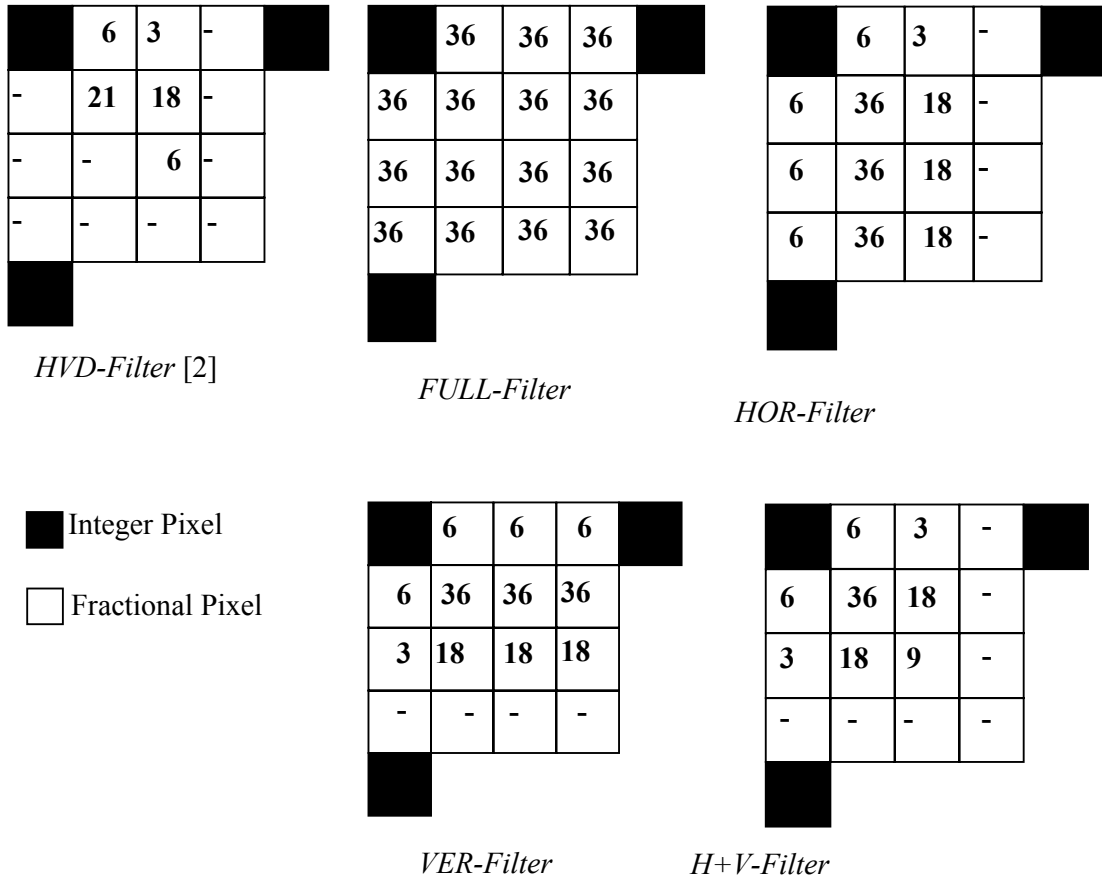


Figure 21 Number of coefficients for each sub-pixel position for different symmetries

Figure 21 illustrates the number of coefficients used by each sub-pixel position for the filters with different symmetry assumptions. The numbers inside each square represent how many coefficients the interpolation filter uses for the corresponding sub-pixel position. For example, for position j , *HVD-Filter* uses 6 coefficients, whereas in *HOR-Filter* and *VER-Filter* 18 coefficients are used. The total numbers of coefficients are 54, 540, 189, 189 and 99

for the *HVD*, *FULL*, *HOR*, *VER* and *H+V-Filters*, respectively. In order to find the symmetry type of the interpolation filter and its corresponding coefficients, each frame is encoded several times for each one of the candidate filter symmetry and the optimal filter symmetry is found using rate distortion optimization. In other words, this algorithm encodes each frame 6 times (one for the fixed filter and once for each candidate symmetry type), compared to encoding each frame 2 times. This means that the encoding time is increased approximately three times. The coding efficiency of the proposed algorithm is compared with 2D-AIF in Table 9, where Delta-Bitrate is calculated using [20]. To analyze where the coding efficiency improvements are coming from Figure 23 illustrates the rate-distortion curve of the proposed algorithm with respect to 2D-AIF. As seen in Figure 23, most of the gain is coming at high bitrates (improvements coming close to 10%), validating the earlier assumption that filters with fixed symmetry are not optimal especially at high bitrates.

Table 9 Coding efficiency of adaptive interpolation with flexible symmetry

Sequence	Bitrate Reduction (delta-bitrate)
Paris	-0.14 %
Foreman	-0.82%
Mobile	-0.66 %
City (4CIF)	-5.39 %
Ice	-0.1 %
Soccer	-4.5 %
BigShips	-1.04 %
ShuttleStart	-0.18%
City	-3.84 %
Jets	-0.66 %
Preakness	-1.36 %
Raven	-2.75 %

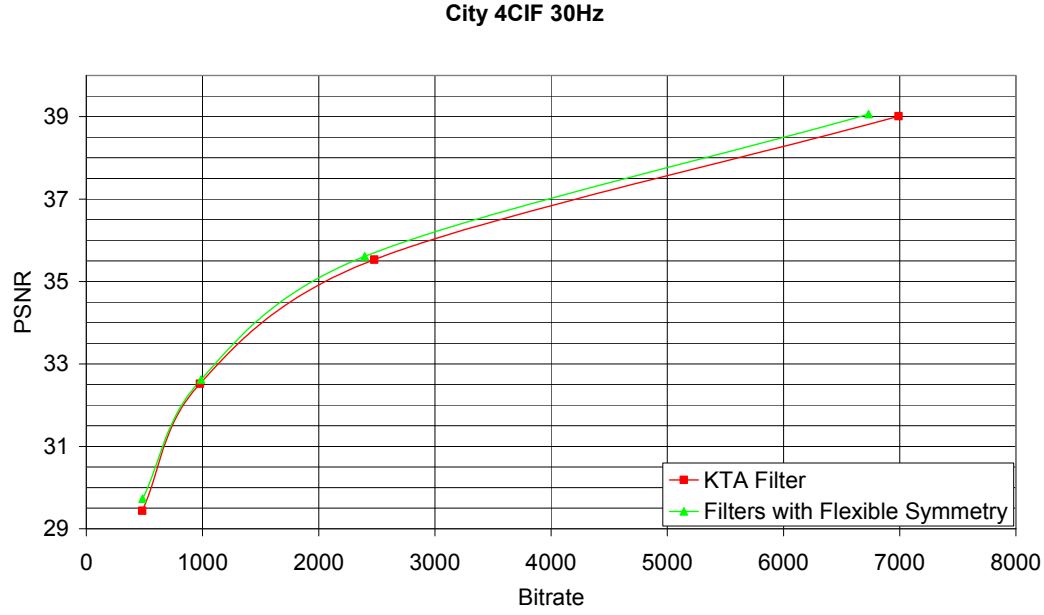


Figure 22 Coding efficiency of the proposed algorithm compared to 2D-AIF (denoted as KTA filter)

As mentioned above, the encoding complexity of the proposed algorithm is high due to the need to perform additional encoding passes for each frame. In the next subsection, a filter structure that exploits a higher degree of flexibility is proposed to reduce this number of additional encoding passes.

3.4.2. Adaptive Interpolation with Flexible Filter Structures

The previous subsection presented a novel algorithm of using adaptive interpolation filters with flexible symmetry. However, due to the need to encode each frame multiple times, the encoding complexity is quite high. In addition, for the same reason, the number of different symmetries that could be used for defining the interpolation filter is quite limited.

This subsection presents a novel algorithm that extends this idea by allowing more variety for filter symmetries, and hence being more flexible. This variety is achieved by signaling a symmetry descriptor for each sub-pixel, which carries the information if the corresponding sub-pixel uses an independent filter or shares the same filter with another sub-pixel. In addition to making the symmetry of the filter flexible, the filter structure is not fixed but also allowed to change based on the properties of the signal. It should be noted that both of these algorithms are introduced without any need to perform multiple-pass encoding, and hence minimizing the additional complexity.

Highly Flexible Symmetry for Adaptive Interpolation

The symmetry descriptor for each sub-pixel is calculated as follows. First, for each sub-pixel location Sp , the adaptive filter $h(Sp)$ is computed without assuming any symmetry. Then each

N-tap length filter is treated as a vector in N-space, and the Euclidean distance $d_{Sp1,Sp2}$ between filters of different sub-pixel positions $Sp1$ and $Sp2$ are computed as follows:

$$d_{Sp1,Sp2} = \sqrt{\sum_{i=0}^{N-1} (h_i(Sp1) - h_i(Sp2))^2} \quad (15)$$

This distance is compared against a threshold, and if the distance between two filters is lower than a threshold, those sub-pixel samples share a single filter which is re-computed from the merged covariance matrixes of those two sub-pixel locations. Figure 23, illustrates this process, where filters $h(a)$ and $h(c)$ will be merged, as their Euclidian distance is lower than the threshold, whereas sub-pixels d and l will use independent filters.

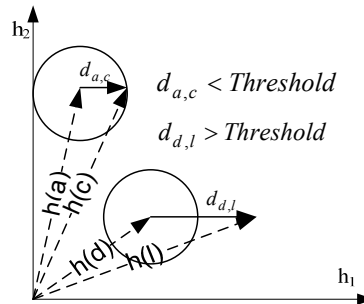


Figure 23 An interpretation of 2-tap length filters diversity analysis. Euclidian distance (solid arrow) between two filters (dashed arrows) is compared against the threshold.

The threshold is calculated as a function of bitrate. If the coded bitrate is high, this means there is a higher budget for transmitting the filter coefficients, so the threshold is decreased. If the coded bitrate is low, a higher threshold is used for the same reason. The threshold value is empirically calculated by running extensive test simulations over a wide set of test video sequences coded at different bit-rates.

Flexible Filter Structures for Adaptive Interpolation

In addition to making the symmetry of the filters flexible, two different 12-tap filter structures are used for the middle sub-pixels $\{f,i,j,k,n\}$. The reason to add a second set of filter structure for those sub-pixels is because the diagonal cross filter has a weaker support for the diagonal edges and this result in a loss in the coding efficiency for some of the sequences. In order to illustrate this phenomenon, the cut-off frequencies at -3dB level of frequency response of the diagonal cross filter is illustrated in Figure 25 and compared against the standard 2D 6x6-tap H.264/AVC filter, using the same filter coefficients. One can see that the 12-tap diagonal-cross filter provides better support for frequencies in the horizontal and vertical directions (dash-dotted line). It's frequency support in these directions is close to the cut-off frequency of

the 6x6 filter of H.264/AVC. However, the diagonal cross filter has a “weaker” support (low cut-off frequency) in the diagonal directions.

To compensate this effect, a second set of filter structure (called radial filter) is introduced. The spatial support of the radial filter is shown in Figure 24 and is defined as:

$$s(\text{Sp}) = \{B3, B4, C2, C3, C4, C5, D2, D3, D4, D5, E3, E4\}$$

The frequency response of the radial filter is also shown in Figure 25. It is seen in Figure 25 that diagonal cross and radial filters have complementary characteristics. For example the radial filter has a stronger support than the diagonal cross filter in the diagonal directions but a weaker support in the horizontal and vertical directions.

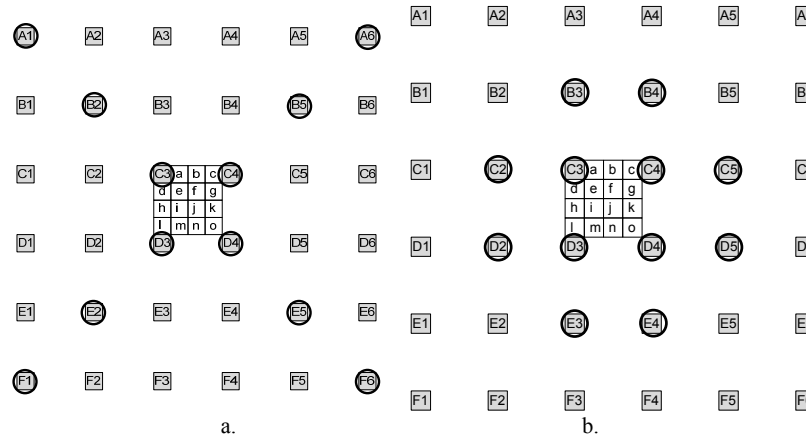


Figure 24 12-tap filter structures: a) – diagonal cross filter; b) – radial support filter.

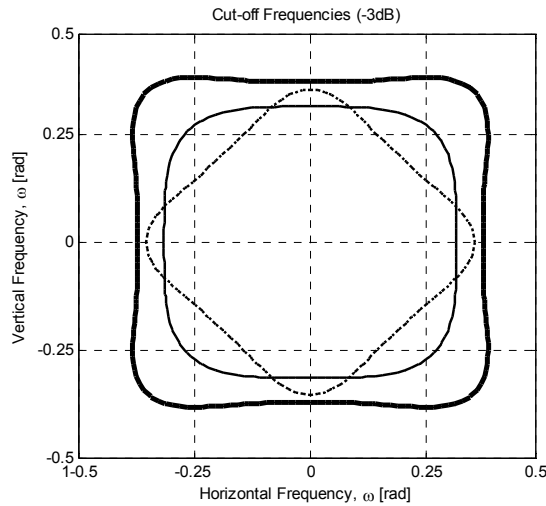


Figure 25 Cut-off frequencies (at the level -3dB) for three $h(j)$ filters: (thick solid line) – 2D 6x6 filter of H.264/AVC, (thin solid line) – estimate of 12-tap Radial Filter, (thin dash-dotted line) – estimate of Diagonal-Cross Filter.

The encoder algorithm to determine the filter symmetry and structure is detailed step by step below;

1. Frame is coded with the fixed filter given in H.264/AVC and motion vectors are computed
2. Using the motion vectors and the reference frame, encoder computes independent filters for each sub-pixel $\{a, \dots, o\}$. For sub-pixel locations $\{f, i, j, k, n\}$ filters for two different structures are computed, diagonal-cross and radial.
3. The decision between radial and diagonal cross filter is made based on the motion prediction error. That is, by using the motion vectors obtained at the first step, the error between original and prediction image is computed for both radial and diagonal cross structure. The filter structure that gives the smallest error for each sub-pixel is chosen.
4. The symmetry between each sub-pixel location is calculated as described in Section 3.4.1. According the calculated symmetry the co-variance matrices are modified, and the filter coefficients are re-computed
5. Input frame is encoded with adaptive interpolation filter.

The coding efficiency of the proposed algorithm is shown in Table 10. In Table 10 the coding efficiency of the proposed algorithm is compared both to H.264/AVC interpolation and using directional adaptive filtering. It should be noted that the proposed algorithm requires no additional encoding passes, thus the encoding complexity increase can be considered negligible. At the decoder side, the complexity is not changed, as the interpolation for each sub-pixel is performed using the same number of taps.

Table 10 Simulation results of adaptive interpolation with flexible filter structures

Sequence	Δ Bit-rate,%	
	DAIF	Flexible filter structure and symmetry
BigShips	-8.56	-9.37
Night	-5.79	-6.50
City	-12.77	-14.41
Crew	-10.50	-10.67
Raven	-21.74	-24.60
Jets	-4.23	-6.49
CrowdRun	-1.02	-1.48
ParkJoy	-1.34	-2.10
Toys&Call	-8.53	-10.14
Sunflower	-21.97	-23.64
Average	-9.65	-10.94

The flexible filter structure and the symmetry described in this section was presented in ITU-T/VCEG standardization group and it was adopted to the Key Technological Area

(KTA) experimental software that is developed with the aim of collecting promising tools for the next generation video coding standards [S5][23] .

Chapter 4

3D Video and Multiview Video Coding

With the advances in acquisition and display technologies, 3D video is becoming a reality in consumer domain with different application opportunities. To address these emerging applications and to ensure interoperability between different manufacturers, the Joint Video Team (JVT) recently standardized the Multiview Video Coding (MVC) extension to the H.264/AVC standard.

The MVC standard was designed to efficiently code videos obtained by cameras pointing to the same scene but from different viewpoints (e.g. videos obtained by stereo-cameras). One of the benefits of MVC is the additional compression efficiency compared to coding each view independently (simulcast) or coding them side-by-side. This is achieved by exploiting the redundancy present between views and applying the same motion compensation methods used to exploit temporal redundancy. As similar tools are used to exploit the view redundancy as to exploit temporal redundancy, similar complexity problems exist for coding 3D video. In addition, due to the need to process many views and output the pictures at the same time, an additional challenge exists to enable parallel processing of the views.

This section first presents a brief overview of the MVC standard and then presents tools to efficiently code parallel views (more specifically, Parallel Decoding Supplemental Enhancement Information message is presented). Finally, this section discusses various applications of adaptive interpolation filtering in MVC.

4.1. MVC AMENDMENT OF H.264/AVC

4.1.1. Features of MVC

The Joint Video Team (JVT) worked on finalizing the Multiview Video Coding (MVC) extension to H.264/AVC standard starting in July 2006, and finalized it in June 2008. Before the start of this project, the requirements for the MVC standard were identified. Some of those

requirements, such as ability to support low delay processing, are applicable to any video coding standard. Some of the other requirements have MVC specific features that could be summarized as:

Improved Compression Efficiency: The MVC standard shall allow significantly improved coding efficiency compared to coding each view independently (i.e. simulcast), by exploiting the redundancies between views.

Backwards Compatibility: One of the views contained in a bitstream conforming to the MVC standard shall be decodable by H.264/AVC decoders that do not support the MVC standard.

Ability to Transmit Camera Parameters: The MVC standard shall include efficient ways to transmit extrinsic and intrinsic camera parameters. These parameters could be used for many purposes, such as rectification of stereo-views for improving the 3D video quality, synthesizing virtual views in between original views, etc.

Parallel Processing: MVC standard shall include efficient ways to process views in parallel. One of the ways to support parallel processing is by using Parallel Decoding Info SEI message and is further discussed in Section 4.1.2.

View Scalability & Random Access: View scalability means that MVC standard shall allow efficient ways to extract a portion of the bitstream to output a limited number of the views out of the N original views. View random access means that it shall be possible to access any picture in any view without the need to decode the entire bitstream.

It should be noted that, the above features of MVC are achieved by only modifying the high level syntax of the H.264/AVC standard without changing any low level coding tools. This makes the introduction of MVC to the market easier as existing H.264/AVC hardware and software implementations could be reused with minor tweaking.

4.1.2. Performance of MVC

The coding efficiency of MVC has been analyzed in much detail in [36] by comparing different prediction structures and also using subjective evaluation in addition to objective metrics. One of the sample prediction structure used by MVC is illustrated in Figure 26. In Figure 26, T_0, T_1 , etc. represent video frames captured at different time instants and S_0, S_1 , etc. represent video frames captured from different cameras. As seen in Figure 26, the multiview video sequence are coded using the hierarchical-B frame concept, which was found to be useful in improving coding efficiency for traditional 2D video sequence, but used in both in temporal and view directions [42]. It should be noted that simpler prediction structures than the one depicted in Figure 26 was shown to have similar or even better coding efficiency [37].

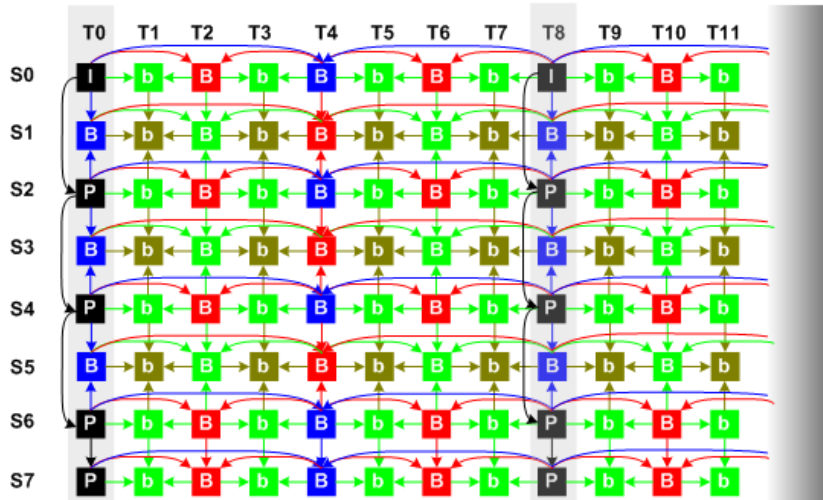


Figure 26 Illustrative prediction structure for MVC using Hierarchical B pictures

In order to assess the performance of MVC, extensive experiments have been carried out during the course of standardization. In [36], multi-view test sequences with 5 to 16 camera views have been evaluated with the developed coding scheme and compared against anchor coding results in terms of PSNR vs. bitrate. The anchor as provided by MPEG was coded with H.264/AVC simulcast and an IBBPBBP temporal decomposition. It is shown that coding gains up to 3.2 dB and an average gain of 1.5 dB could be achieved by additionally using interview reference pictures that is enabled by MVC. In addition, subjective results show that majority of the cases the same quality is achieved at half the bitrate of simulcast

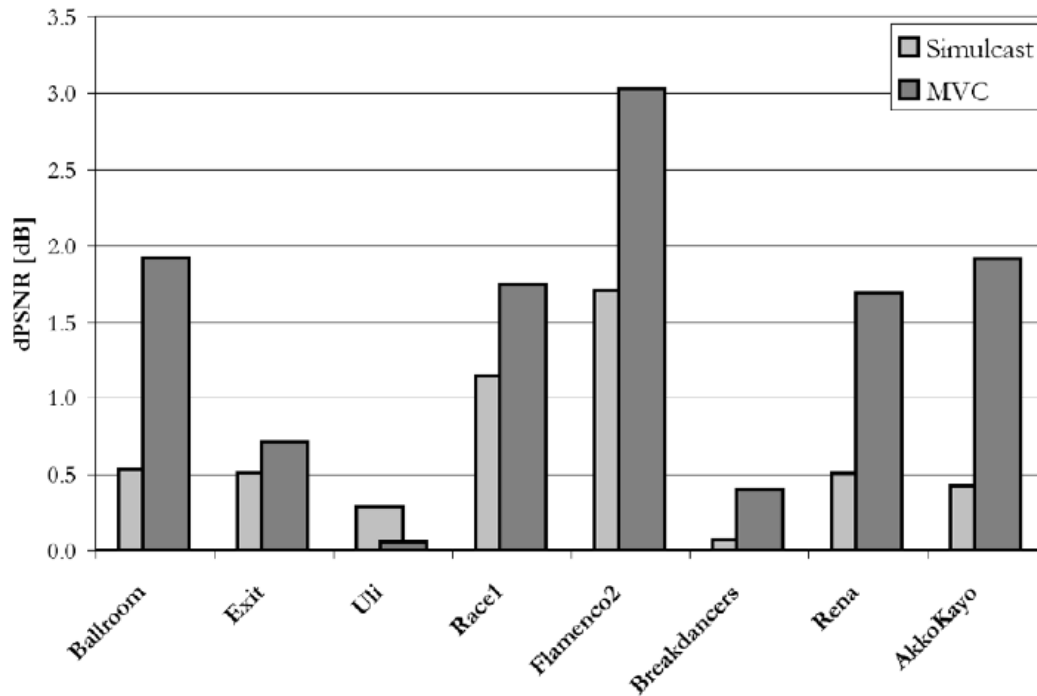


Figure 27 Performance of MVC compared to simulcast [36]

4.2. PARALLEL PROCESSING OF MULTIPLE VIEWS

As mentioned earlier, one of the key identified requirements for the MVC standard is its ability to support parallel processing of different views [26]. The parallel processing of different views is especially important for 3D broadcasting use cases, where displays need to output many views simultaneously to support head-motion parallax. However, inter-view dependencies between pictures may impose serious parallelism issues to the video system, because two pictures at different views need to be decoded sequentially. To illustrate this problem, let's consider a 3DTV system displaying simultaneously two views, where the views are coded with the coding structure as illustrated in Figure 28.

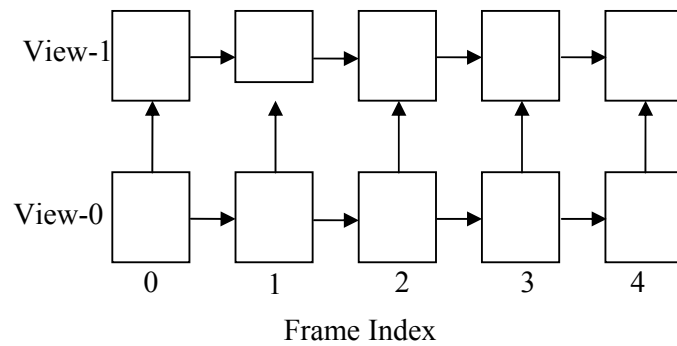


Figure 28 Sample prediction structure for two views

In order to decode a picture in view 1 at any temporal instant, the picture in view 0 at the same temporal instant shall be decoded first. One way to display two views at the same time is by having an MVC decoder running two times faster than a regular single-view decoder. Even though two independent decoders running on different platforms might be available, both decoders need to run twice faster than the single-view decoder because decoding has to be performed sequentially. Another alternative is to run each decoder at the same speed as a single-view case but delay the display by one picture. This way the display of pictures does not happen immediately after pictures are received and decoded, but lag the decoding process. Although this might not be a problem for a two-view stereo use-case, the situation gets worse, with the increasing number of views that is supported by a 3D display. One other way to increase the parallelism is to code each view independently. However, this kind of simulcast approach results in a significant penalty in coding efficiency as inter-view redundancies are not exploited at all.

In order to facilitate parallel processing of different views, while maintaining the coding efficiency, a novel algorithm is developed. The proposed algorithm encodes each view with certain constraints on inter-view prediction, so that the decoding of macroblocks from different view could be pipelined. The proposed algorithm is also proposed to MVC standardization and is included in the standard in the form of Parallel Decoding Information Supplemental Enhancement Information (SEI) message [S7]. This SEI message indicates the views are encoded with systematic constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of the macroblocks in other views.

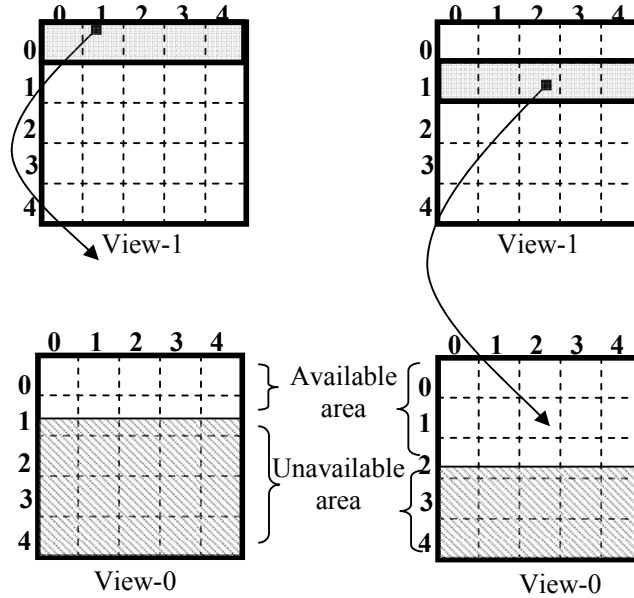


Figure 29 Systematic restriction of reference area

In order to describe how the proposed parallel processing is achieved using Parallel Decoding Information SEI message; let's consider an example, where two pictures from view 1 and view 0 are going to be decoded. Assume view 1 picture references view 0 picture as illustrated in Figure 29 (for simplicity the size of the frames are five macroblocks both horizontally and vertically). Parallel Decoding Information SEI message indicates the video is encoded in a way that macroblocks in view 1 picture could only use reconstruction values of macroblocks that belong to certain rows in the view 0 picture. For example, the macroblocks in the first macroblock row of view 1 picture could only use reconstruction values from the first two macroblock rows in view 0 picture. In other words, the available reference area for the first macroblock row of view 1 picture constitutes only data from the first two macroblock rows of view 0 picture (i.e. the motion vectors for the view 1 macroblocks are restricted). This systematic restriction of reference area enables parallel decoding of first row of view 1 with any row below the second of view 0, as they are not referring each other.

In order to illustrate how this feature is used, let's assume an MVC decoder running on two processors (or processor cores) and decoding a bitstream containing two views, where view-1 references view-0. Further assume that the bitstreams are coded with the restrictions as described above. The parallel decoding operation of these two views is illustrated in Figure 30, where processor P0 is decoding view-0 pictures and processor P1 is decoding view-1 pictures. The decoding operation for both views start simultaneously, but decoding of the first row of macroblocks in view 1 picture does not start before view 0 notifies the view 1 decoder. This notification is done after all the macroblocks in the first two macroblock rows in view 0 are decoded, and their reconstruction data are placed in the memory. This notification tells the decoder of view 1 that, all data required to decode the first macroblock row in view 1 are ready. This way, the decoder of view 1 could start decoding of macroblocks of the first row,

while the decoder of view 0 proceeds with decoding macroblocks in the third row and the two decoders run in parallel. This parallel operation continues with two macroblock rows of delay between two views till the decoding of all the macroblocks is finished.

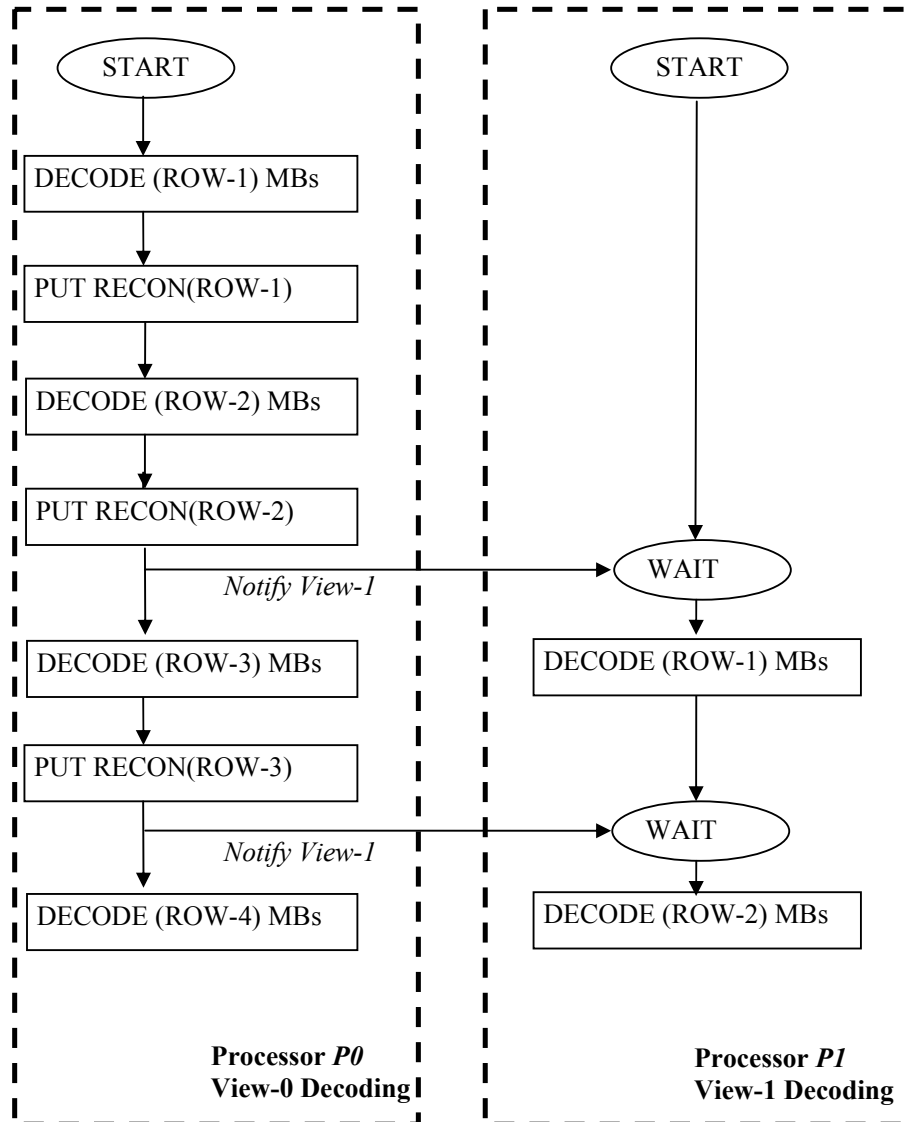


Figure 30 Sample parallel decoding process for two views

The benefit of using Parallel Decoding Information SEI message is that, significant coding gain is achieved over simulcast, while maintaining almost the same desirable parallelism characteristics. When compared to anchor method, where encoding happens without utilizing the SEI message and systematic restrictions, it is seen that parallel operation is achieved with almost no penalty on the coding efficiency: maximum 0.08 and in average 0.03 dB loss for all the test sequences defined in the common test condition for MVC [24]. Compared to

simulcast, similar parallelism is achieved with 0.9 dB gain on coding efficiency as inter-view prediction is still utilized.

In order to test the performance of this scheme, the proposed algorithm is implemented on the Joint Multi-view Test Software (JMVM) [40] and simulations have been performed following the test conditions advised by the JVT standardization committee [41]. For the simulcast case, where each view is coded independently, each decoder could run at the same speed as the single-view case to output pictures simultaneously, as there are no inter-view dependencies. For coding structures that utilize inter-view redundancy, it was assumed that decoder needs to run N times faster than a single view decoder (N being number of dependent views), to simultaneously output pictures from different views. For the bitstreams generated with the proposed method, there exists a lot less speed requirement. Table 11 compares the decoder speed requirement for the proposed method with different approaches for different sequences. The data for anchor in Table 11, utilizes inter-view prediction structure using hierarchical B pictures as shown in [27].

Table 11 Decoding Speed Increase Compared to Single-View Decoder for Three Approaches

	Decoder Speed Requirement	
Sequence	Anchor	Proposed
Ballroom	5xsimulcast	1.3xsimulcast
Exit	5xsimulcast	1.3xsimulcast
Race1	5xsimulcast	1.3xsimulcast
Flamenco2	2xsimulcast	1.03xsimulcast
BreakDance	5xsimulcast	1.2xsimulcast
Rena	8xsimulcast	1.5xsimulcast
Uli	5xsimulcast	1.2xsimulcast
Akko&Kayo	6xsimulcast	1.4xsimulcast

The coding results for the proposed scheme are compared against two approaches, simulcast and anchor. The anchor data is generated without any encoder restrictions and simultaneous output of views is possible only by running the decoder N times faster, N being the number of dependent views. Comparison against anchor is useful to see the penalty on the coding efficiency to enable parallel decoding of different views. The proposed algorithm is also compared against simulcast approach. The encoding details for simulcast are exactly the same as the proposed scheme and anchor, only difference being the inter-view prediction is disabled for simulcast. Comparison against simulcast indicates the amount of improvement proposed scheme brings with similar parallelism requirements as simulcast. Table 12 presents the coding efficiency comparison of the proposed approached against anchor and simulcast. The differences in PSNR are calculated using the method described in [20]. It is shown in Table 12 that the proposed scheme achieves up-to 0.9 dB gain compared to simulcast, with very similar parallelism requirements. Compared to anchor approach, where the parallel operation is not possible, the penalty to achieve parallelism is negligible (less than 0.025 dB on

average). Similar results could be seen in Figure 31, where the RD curve for the sequence Flamenco is plotted. As also seen in Figure 31, proposed scheme has significant gains compared to simulcast with similar requirements on parallelism.

Table 12 Coding Efficiency of the proposed parallel multiview scheme relative to anchor and simulcast approaches

Sequence	PSNR Results of Proposed Scheme	
	Δ_{dB} relative to anchor	Δ_{dB} relative to simulcast
Ballroom	-0.005 dB	+ 1.17 dB
Exit	-0.004 dB	+ 0.39 dB
Race1	-0.06 dB	+ 0.92 dB
Flamenco2	-0.07 dB	+ 1.18 dB
BreakDance	-0.002 dB	+ 0.41 dB
Rena	-0.003 dB	+ 1.55 dB
Uli	-0.01 dB	- 0.19 dB
Akko&Kayo	-0.08 dB	+ 1.44 dB

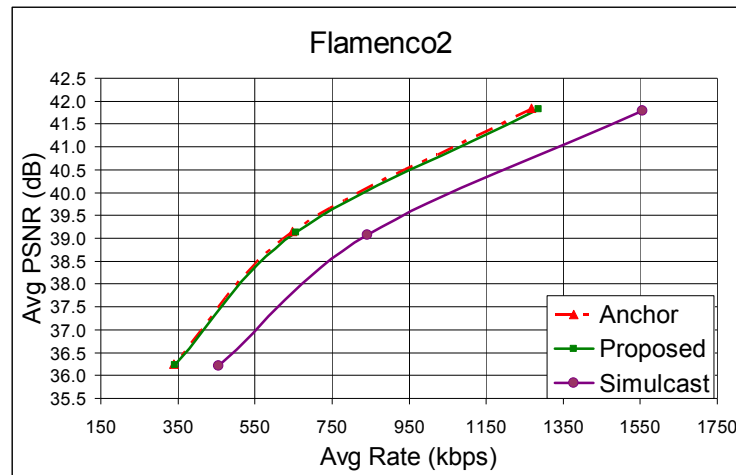


Figure 31 Comparing the proposed parallel encoding scheme with anchor and simulcast

It should be noted that, Parallel Decoding Information SEI message does not change the worst case complexity of MVC decoders. This is due to the fact that SEI messages are not mandatory, and hence encoders need not follow these restrictions to comply with the standard. This means that the MVC decoders need to be designed to handle bitstreams where the encoding restrictions have not been applied, and Parallel Decoding Information SEI message is not present. However, system standards such as DVB-H [29] references can mandate the usage of this SEI message in their respective environments. This would ensure parallel operation for all the decoders operating on these services.

4.3. ADAPTIVE INTERPOLATION FILTERING FOR MVC

In Section 3, applications of adaptive interpolation filtering on video coding is presented. AIF techniques are useful also for coding 3D video based on MVC. Firstly, similar to coding 2D video, AIF could be used to exploit temporal redundancy better in 3D video as well and improve the coding efficiency. However, it is found there are two techniques specifically useful for 3D video coding that AIF is useful.

Asymmetric video coding

Asymmetric coding is a concept where one of the views in the stereo 3D setup is coded with lower fidelity, while providing unnoticeable or small perceptual quality degradation for the stereoscopic display [31]. Lower fidelity for the other view could be achieved either by coding the view with low quality or at lower spatial resolution. A multiview video coding system based on coding one of the views at lower resolution was proposed in [32] and illustrated in Figure 32

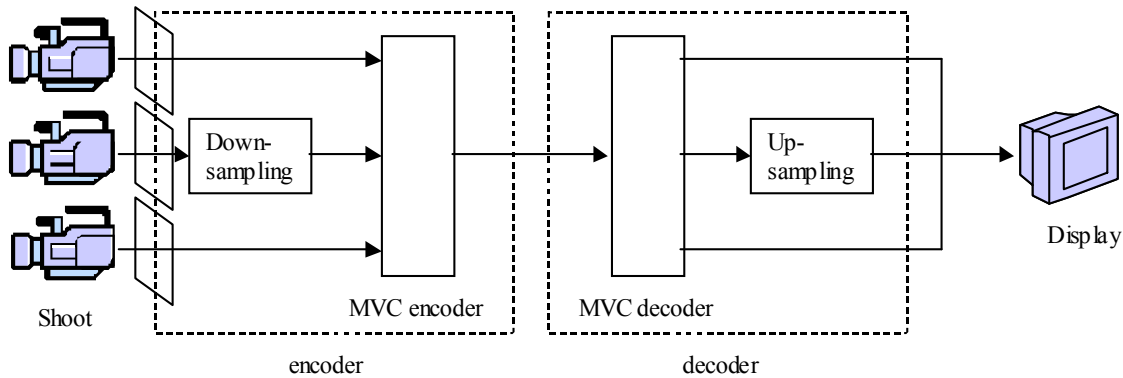


Figure 32 Workflow for asymmetric MVC coding [32]

In this scenario, the samples from the views that are coded at high resolution need to be downsampled prior to using them in the predicting the lower resolution views. This down-sampling could be done either using a fixed or a temporally varying filter. In [33], it was found that applying an adaptive filter to generate downsampled inter-view predictors using adaptive interpolation techniques improve the coding efficiency significantly.

Focus mismatch compensation

In multiview video coding, multiple cameras captures the same scene at the same temporal instant, from different viewpoints. This means that to differences in camera settings and/or shooting positions, frames from different views are prone to suffer from mismatches other than simple displacement [32]. Due to the differences in camera distances to objects, the object in one view might appear sharper than other views. To compensate for this type of focus mismatches that result in blurriness/sharpness discrepancy among different views, algorithms based on adaptive interpolation filtering concept have been proposed [32][35]. This means that the novel concepts developed in Section 3 could be useful to improve the coding efficiency of 3D video coding and/or reduce the implementation complexity.

Chapter 5

Conclusions

Video coders are an essential part of any digital video application, such as digital TV broadcasting or video conferencing as the bitrate of an uncompressed video signal is too high for any practical usage. The main goal of all video coders is to maximize the video quality while keeping the bitrate as low as possible. In addition, all video coders try to achieve compression of the video signal with as low complexity as possible. The implementation complexity is especially important for mobile devices, such as mobile phones where the computational resources are severely limited. In addition, since those devices are battery operated, an increase in complexity directly translates to a decrease in operating time.

This thesis presented novel methods for improving the compression capability of video coders and reducing their implementation complexities. The goal of this thesis is to come up with novel algorithms, which could be useful in video coding standards to improve the quality of experience of the future mobile video applications. Many of the ideas presented in the thesis have either been adopted by standardization bodies or currently being considered for international standards.

Section 2 presented a detailed analysis of the motion compensation part of the recent H.264/AVC standard with a special attention to luminance interpolation. A novel algorithm is presented that generates standard compliant bitstreams that are easy to decode, by exploiting the difference in interpolation complexities of different sub-pixels. This work was also presented at the ITU-T VCEG meeting to demonstrate the flexibility of the H.264/AVC standard. Section 3 discusses novel algorithms for adaptive interpolation filtering (AIF) schemes that adaptively change the filter coefficients at each frame. A novel structure based on directional filters and its efficient 16-bit implementation with high coding efficiency is presented in Section 3.2. This structure achieves significant reduction in complexity compared to other state-of-the-art methods, while maintaining the high coding efficiency. Both of these methods are adopted by VCEG standardization group into the experimental KTA (Key Technology Area) software aiming to collect promising tools for the future[S1][S2][S3]. The low complexity alternative of directional filter structure with fixed coefficients was also proposed to VCEG and got adopted into the KTA experimental software [S4]. In addition, the directional filter structure with fixed coefficients was accepted to the initial design of the High Efficiency Vid-

eo Coding (HEVC) standard [S6][39]. Section 3.3 presented novel algorithms aiming to reduce the encoding complexity of AIF schemes significantly, without hurting the coding efficiency. Section 3.4 presents methods to further improve the coding efficiency of AIF schemes, by making the filter structure and the filter symmetry flexible. These methods were also adopted to KTA software maintained by the VCEG standardization group[S5].

Section 4 discusses 3D video coding and the recent Multiview Video Coding (MVC) standard with special emphasis on its implementation complexity. More specifically, parallel decoding of multiple views are discussed and a novel algorithm is presented in Section 4.2 that improves the parallel processing of multiple views with high coding efficiency. This tool is also proposed to JVT (Joint Video Team) and included in the final MVC standard as Parallel Decoding Info SEI Message [S7]. In addition, the potential of adaptive interpolation filtering techniques presented in thesis for improved 3D video coding is presented in Section 4.3.

5.1. FUTURE WORK

The video coding is an active research area with significant improvements in coding efficiency continuously being demonstrated. At the time of writing the thesis, VCEG and MPEG, the two international standardization organizations who jointly created the very successful MPEG-2 and the H.264/AVC standards, have joined once again and formed the Joint Collaborative Team on Video Coding (JCT-VC). JCT-VC embarked on a new project to develop the next generation video coding standard called High Efficiency Video Coding (HEVC). Many of the techniques presented in the thesis are found promising and considered by many proposals brought to JCT-VC. Among those, various adaptive interpolation techniques have been used extensively, as summarized in [38], and the techniques presented in Section 3 could be utilized to further improve the performance of those proposals. In addition, the joint proposal made by Nokia, Tandberg and Ericsson included directional interpolation filter technique, introduced in Section 3.2, but using fixed coefficients. Adaptive in-loop filtering techniques that filter the reconstructed picture prior to placing it in reference frame memory in a spatio-temporal adaptive manner have also been found promising. Many of the techniques presented in Section 3 are developed for interpolation, but could be extended in the future for adaptive in-loop filtering techniques.

In addition to advances in traditional two dimensional video coding, 3D video coding is receiving increased attention from the research community. The emerging use-cases require transmission of more than 2 views, and increased effort is put on compressing multiple views in an efficient manner. One of the promising areas is to code depth maps of certain views and synthesize intermediate views using the neighboring views and the associated depth maps, instead of transmitting them separately. This approach will require significant complexity challenges. Novel algorithms need to be designed for managing the complexity and increasing the parallelism of the architecture.

Bibliography

- [1] ISO/IEC JTC1/SC29/WG11, Vision, Applications and Requirements for High-Performance Video Coding, MPEG Document N11096, January 2010
- [2] ISO/IEC 14496–10:2003, “Coding of Audiovisual Objects—Part 10:Advanced Video Coding,” 2003, also ITU-T Recommendation H.264 “Advanced video coding for generic audiovisual services.”
- [3] Wiegand T. and Girod B., “Multi-Frame Motion-Compensated Prediction for Video Transmission”, Kluwer Academic Publishers, September 2001.
- [4] Ostermann, J.; Bormans, J.; List, P.; Marpe, D.; Narroschke, M.; Pereira, F.; Stockhammer, T.; Wedi, T.; , "Video coding with H.264/AVC: tools, performance, and complexity," *Circuits and Systems Magazine, IEEE* , vol.4, no.1, pp. 7- 28, First Quarter 2004
- [5] Flierl, M.; Girod, B.; , "Generalized B pictures and the draft H.264/AVC video-compression standard," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp. 587- 597, July 2003
- [6] M. Karczewicz and A. Hallapuro, “Interpolation Solution with Low Encoder Memory Requirements and Low Decoder Complexity”, VCEG-N31, 24-27 Sept. 2001.
- [7] V. Lappalainen, A. Hallapuro, and T. Hämäläinen, “Performance Analysis of Low Bit Rate H.26L Video Encoder”, In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001*, May 2001, pp. 1129-1132.
- [8] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, “Complexity of Optimized H.264/AVC Video Decoder Implementation”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 717-725, July 2003.
- [9] Horowitz, M.; Joch, A.; Kossentini, F.; Hallapuro, A.; , "H.264/AVC baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp. 704- 716, July 2003
- [10] Wedi, T.; Musmann, H.G.; , "Motion- and aliasing-compensated prediction for hybrid video coding," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp. 577- 586, July 2003
- [11] O.Werner, “Drift analysis and, drift reduction for multiresolution hybrid video coding,” *Signal Processing: Image Commun.*, vol. 8, no. 5, July 1996.

- [12] T. Wedi, "Adaptive interpolation filter for motion and aliasing compensated prediction," in Proc. Visual Communications and Image Processing (VCIP), San Jose, CA, Jan. 2002.
- [13] T. Wedi, "Adaptive Interpolation Filter for H.26L", ITU-T SG 16/Q.6 Doc. VCEG-N28, Sep. 2001, Berlin Germany.
- [14] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion and Aliasing-Compensated Prediction Using a Two-dimensional Non-Separable Adaptive Wiener Interpolation Filter", *Proc. ICIP 2005*, Genova, Italy, September 2005.
- [15] S. Wittmann and T. Wedi, "Separable Adaptive Interpolation Filter for Video Coding", In Proceedings International Conference on Image Processing, ICIP 2008, San-Diego, USA, Oct. 2008, pp. 2500-2503.
- [16] Y. Ye and M. Karczewicz, "Enhanced Adaptive Interpolation Filter", ITU-T SG16/Q6, doc. C-0464, Geneva, Switzerland, Apr. 2008.
- [17] Vatis, Y.; Ostermann, J.; , "Locally Adaptive Non-Separable Interpolation Filter for H.264/AVC," *Image Processing, 2006 IEEE International Conference on* , vol., no., pp.33-36, 8-11 Oct. 2006
- [18] Y. Vatis and J. Ostermann, "Comparison of Complexity between Two-Dimensional Non-Separable Adaptive Interpolation Filter and Standard Wiener Filter", ITU-T SG16/Q.6 Doc. VCEG-AA11, Nice, France, Oct. 2005.
- [19] Hung-Ju Lee; Tihao Chiang; Ya-Qin Zhang; , "Scalable rate control for MPEG-4 video," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.10, no.6, pp.878-894, Sep 2000
- [20] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-curves" ITU-T SGI 6/Q.6 Doc. Doc. VCEG-M33, April 2001.
- [21] Merkle, P.; Muller, K.; Smolic, A.; Wiegand, T.; , "Efficient Compression of Multi-View Video Exploiting Inter-View Dependencies Based on H.264/MPEG4-AVC," *Multimedia and Expo, 2006 IEEE International Conference on* , vol., no., pp.1717-1720, 9-12 July 2006
- [22] Merkle, P.; Smolic, A.; Muller, K.; Wiegand, T.; , "Efficient Prediction Structures for Multiview Video Coding," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.17, no.11, pp.1461-1473, Nov. 2007
- [23] Reference ITU-T VCEG-KTA Software [Online]. Available: <http://iphome.hhi.de/suehring/tml/download/>.
- [24] Common Test Conditions for Multiview Video Coding, JVT-T207, Klagenfurt, Austria, Jul. 2006.

- [25] Sullivan, G.J.; Wiegand, T.; , "Rate-distortion optimization for video compression," *Signal Processing Magazine, IEEE* , vol.15, no.6, pp.74-90, Nov 1998
- [26] ISO/IEC JTC1/SC29/WG11, Requirements on Multi-view Video Coding v.5, N7539, Nice, France, October 2005.
- [27] P. Merkle, A. Smolic, K. Mueller, T. Wiegand, Comparative Study of MVC Structures, JVT-V132, Marrakech, Morocco, Jan. 2007
- [28] T. K Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments", ITU-T Q.6/SG16, VCEG-AE010/AI010/AJ010, 2007-2009.
- [29] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "*DVB-H: Digital Broadcast Services to Handheld Devices*," Proceedings of the IEEE, vol. 94, no. 1, pp. 194-209, Jan. 2006
- [30] K. Ugur, J. Lainema, H. Liu, Y.-K. Wang, Parallel Decoding Info SEI Message for MVC, JVT-V098. Marrakech, Morocco, Jan. 2007.
- [31] Julesz B., "Foundations of Cyclopean Perception", University of Chicago Press, Chicago, IL, USA, 1971
- [32] H. Kimata, S. Shimuzu, "Inter-view prediction with down-sampled reference pictures", JVT-W079. San Jose, California, USA, April 2007.
- [33] Ying Chen; Ye-Kui Wang; Hannuksela, M.M.; Gabbouj, M.; , "Picture-level adaptive filter for asymmetric stereoscopic video," Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on , vol., no., pp.1944-1947, 12-15 Oct. 2008
- [34] J.-H. Kim, P. Lai, J. Lopez, A. Ortega, Y. Su, P. Yin, and C. Gomila, "New coding tools for illumination and focus mismatch compensation in multi-view video coding," IEEE Trans. Circuits Systems and Video Technologies (CSVT), vol. 17, no. 11, pp. 1519–1535, Nov 2007.
- [35] P. Lai, Y. Su, P. Yin, C. Gomila, and A. Ortega, "Adaptive filtering for cross-view prediction in multi-view video coding," in Proc. SPIE 2007 Visual Communications and Image Processing (VCIP), Jan 2007.
- [36] Merkle, P.; Muller, K.; Smolic, A.; Wiegand, T.; , "Efficient Compression of Multi-View Video Exploiting Inter-View Dependencies Based on H.264/MPEG4-AVC," Multimedia and Expo, 2006 IEEE International Conference on , vol., no., pp.1717-1720, 9-12 July 2006
- [37] Merkle, P.; Smolic, A.; Muller, K.; Wiegand, T.; , "Efficient Prediction Structures for Multiview Video Coding," Circuits and Systems for Video Technology, IEEE Transactions on , vol.17, no.11, pp.1461-1473, Nov. 2007

- [38] Joint Collaborative Team – Video Coding, “Architectural outline of proposed High Efficiency Video Coding (HEVC) design elements”, JCTVC-A202 output document, first meeting of the JCT-VC, Dresden, Germany, 15-23 April, 2010.
- [39] Joint Collaborative Team – Video Coding, “Test Model under Consideration”, JCTVC-A205 output document, first meeting of the JCT-VC, Dresden, Germany, 15-23 April, 2010.
- [40] “Joint Multiview Video Model (JMVM) 1.0”, JVT-T209, Klagenfurt, Austria, July 2006
- [41] “Common Test Conditions for Multiview Video Coding”, JVT-U221, Hangzhou, China, October 2006.
- [42] Schwarz, H.; Marpe, D.; Wiegand, T.; , "Analysis of Hierarchical B Pictures and MCTF," *Multimedia and Expo, 2006 IEEE International Conference on* , vol., no., pp.1929-1932, 9-12 July 2006

Publications

- [P1] Ugur, K.; Lainema, J.; Hallapuro, A.; Gabbouj, M., "Generating H.264/AVC Compliant Bitstreams for Lightweight Decoding Operation Suitable for Mobile Multimedia Systems," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol.2, no., pp.II-II, 14-19 May 2006

© 2006 IEEE. Reprinted with permission

GENERATING H.264/AVC COMPLIANT BITSTREAMS FOR LIGHTWEIGHT DECODING OPERATION SUITABLE FOR MOBILE MULTIMEDIA SYSTEMS

Kemal Ugur¹, Jani Lainema¹, Antti Hallapuro¹, Moncef Gabbouj²

¹Nokia Research Center, Tampere, Finland

²Tampere University of Technology, Tampere, Finland

ABSTRACT

In this work, we propose novel encoder algorithms for the state-of-the-art video coding standard H.264, to generate decoder friendly video bitstreams. Using the proposed algorithms, it is possible to generate bitstreams requiring significantly less decoding complexity, with negligible effect on picture quality. This is achieved by using novel algorithms for mode decision and motion estimation that bias easy-to-decode motion vectors in a Rate-Distortion optimized fashion. Experimental results show that, more than 15% decoding complexity reduction is achieved with less than a 0.1 dB penalty on the average video quality. We believe that this approach has potential in various use cases especially in mobile multimedia systems, where the video decoder operation is often dominating the handsets power consumption.

1. INTRODUCTION

H.264/AVC is the state-of-the-art video coding standard that is jointly developed by ISO/MPEG and ITU-T/VCEG study groups. When compared to the earlier video coding standards, H.264/AVC achieves significantly better video quality at similar bitrates. Due to its high compression efficiency and network friendly design, H.264/AVC is gaining momentum in industry ranging from third generation mobile multimedia services, digital video broadcasting to handheld (DVB-H) to high definition digital versatile discs (HD-DVD). H.264 achieves increased compression efficiency with the expense of increased complexity for both the encoders and the decoders. Similar to previous standards, the complexity of the H.264 encoder is typically much higher than that of the decoder. There exist several tools that significantly reduce the encoding complexity by resulting slightly lower video quality. These tools include for example fast motion estimation [1], fast mode decision [2] and disabling the use of some of the motion modes etc. The commonality of all these is, they achieve complexity reduction by performing a complexity-video quality trade-off.

For decoders, the situation is completely different. Since the decoder is strictly defined by the standard, it is not possible to have lower complexity decoding by performing a similar video quality-decoding complexity trade-off. Even though the decoder has less complexity than the encoder, the importance of low complexity decoding is equally important as low complexity encoding, if not more. The reason for that is, in many applications such as DVD players, digital TV receivers etc, the end-user equipment has only the decoder implemented and the decoder block is the only codec related functional block adding complexity to the system. Even if

the encoder and decoder co-exist in a system, such as in a video conferencing application, the decoding complexity could become very important if the encoder and decoder are running on different hardware platforms.

In this work, we propose several encoding algorithms that can be used to generate fully H.264 compliant *low-complexity bitstreams*, which require significantly less decoding complexity than bitstreams created with traditional encoding algorithms. More specifically, the focus of this work is to generate bitstreams that would require less amount of half-pixel and quarter pixel interpolations at the decoder, as interpolation step consumes most of the decoding processor cycles [3]. This is achieved in two stages. At the motion estimation stage, the candidate motion vectors having less decoding complexity are biased using a Lagrangian based cost function. At the mode decision stage, the decoding complexity of each mode is estimated and the modes with less decoding complexity are favoured using a similar cost function. Using the proposed methods, the encoder can generate *low-complexity bitstreams* having over 15% less decoding complexity with less than a 0.1 dB penalty on the video quality on average.

This paper is organized as follows; Section 2 provides a brief analysis of the H.264 decoder complexity and the interpolation scheme. Section 3 presents the proposed motion estimation and mode decision algorithms to generate the low complexity bitstreams. Section 4 presents the simulation environment and the experimental results. Conclusions and discussions are presented in Section 5.

2. H.264 DECODER COMPLEXITY ANALYSIS

Previous analyses on H.264 decoder complexity show that the motion compensation is the most computationally complex step at the decoder, followed by the deblocking filter process [3][4]. The high complexity in motion compensation is due to interpolation needed to decode motion vectors with half or quarter pixel accuracy. It was shown that, for a baseline H.264 decoder, this interpolation step takes around 39% of the execution time on average, and it can go up to 44% for some sequences. In the next subsection, we first analyze the details of the H.264's interpolation scheme and present source of its complexity.

2.1 Half-pixel and Quarter-pixel Interpolation

H.264 allows usage of motion vectors with quarter and half pel precision to increase the accuracy of the motion prediction. For the case of integer motion vectors, the prediction signal contains the original values of the reference picture; otherwise the values at non-integer positions need to be in-

terpolated from the original pixels at integer positions. In Figure 2, the original pixels at integer locations are labelled by upper-case letters within shaded boxes, other symbols represent other locations to be interpolated. The samples at half-pixel locations, b , h , m and s , are interpolated from integer samples by applying a one dimensional 6-tap FIR filter on the integer sampled pixels. For example, the sample at half-pixel location b is obtained from the samples at integer locations E, F, G, H, I and J given in Figure 1. More specifically, b is given as

$$b = \frac{((E - 5.F + 20.G + 20.H - 5.I + J) + 16)}{32} \quad (1)$$

The half-pel sample at position j is interpolated by applying the 6-tap filter on the 6 samples at half pixel locations either in vertical or horizontal direction. In other words, in order to interpolate j , the samples at locations cc , dd , h , m , ee and ff must be interpolated first (alternatively, j is obtained using the samples at locations aa , bb , b , s , gg and hh , which would yield the same value). Due to the additional need of interpolating other half-sample locations, the interpolation of sample at position j has the most complexity among other half-pixel samples.

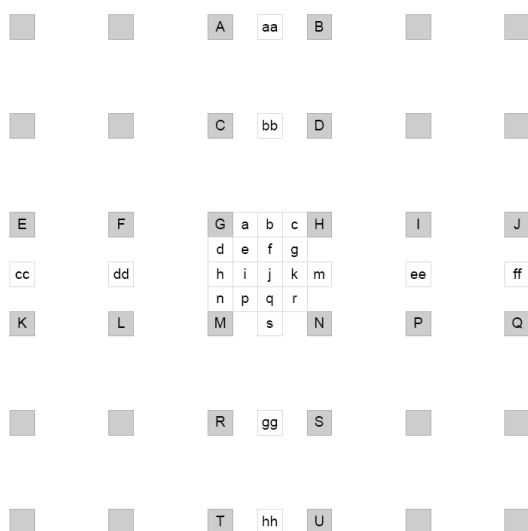


Figure 1 H.264 Sub-Pixel Interpolation

The samples at quarter pixel positions are obtained by averaging two nearest samples at half or integer positions. Figure 2 illustrates the details of the quarter pixel interpolation scheme employed in the H.264 standard. For each quarter pixel, two values are averaged by up-rounding. In Figure 2, quarter pixels are denoted by letters a , c , d , e , f , g , i , k , n , p , q , r and are placed within shaded boxes. Each quarter-pixel is connected to two other half or integer pixels that will be used to calculate the corresponding quarter pixel. For example, in order to interpolate quarter-pixel d , the integer pixel G and the half pixel h must be averaged. So, the half-pixel h must be interpolated first using the 6-tap filter, in order to calculate the value of quarter pixel d . Therefore, the quarter pixels, for which the half pixel j is used for averaging, have the most complexity among other quarter pixel locations (i.e. the quar-

ter pixels at locations f , i , k , q have the highest interpolation complexity).

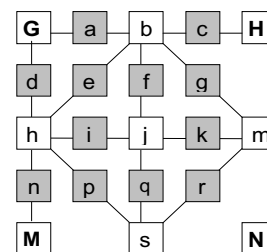


Figure 2 Quarter Pixel Interpolation Scheme

3. ENCODING FOR LOW COMPLEXITY DECODING

In the previous section, it was shown that the interpolation complexities of different quarter and half pixel locations are not the same. Therefore, the decoding complexity depends on both the horizontal and vertical sub-pixel components of the motion vector. For example, if a motion vector has integer values for both its horizontal and vertical components, the decoder does not need to perform any interpolation to obtain the prediction signal, and hence have minimal complexity. However, the complexity would increase, if the motion vector points to locations that are difficult to interpolate, such as the half pixel location j , or any quarter-pixel location that requires the value of j .

We first analyzed the amount of operations needed to interpolate each sub-pixel location, using a highly optimized H.264 decoder implementation [4]. Using this data, we approximated the interpolation complexity of each motion vector with different horizontal and vertical sub-pixel accuracies. Figure 3 presents the approximate interpolation complexities required to decode motion vectors with different accuracies. The reader is referred to [4][5] for a detailed analysis of interpolation complexity.

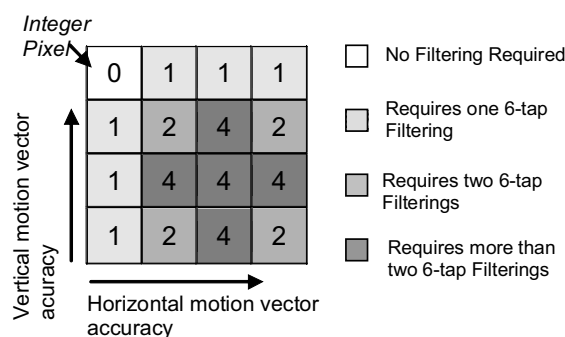


Figure 3 Approximate Interpolation Complexities of Motion Vectors with different sub-pixel components.

In Figure 3, each location is represented by a box and the numbers in each box indicate the approximate interpolation complexity required to decode the motion vector, larger number indicating a higher complexity. For example, if the motion vector has integer values at both directions, the interpolation complexity is zero. However, if the motion vec-

tor points to the half-pixel location just next to the integer pixel in horizontal direction (indicated as b in Figure 2), the interpolation complexity increased to 1, as the decoder needs to perform an additional 6-tap filtering.

We use this approximate complexity data to generate bitstreams that have more motion vectors pointing to easy-to-interpolate locations, in order to reduce the decoding complexity. This is achieved in two stages. At the motion estimation step, the candidate motion vectors having less decoding complexity are biased using a Lagrangian based cost function. At the mode decision step, the decoding complexity of each mode is estimated and the modes with less decoding complexity are favoured using a similar cost function. The details of these stages are presented in the next subsections.

3.1 Motion Estimation

At the motion estimation stage, the reference picture is searched for the candidate motion vectors, and the motion vector that results in the best prediction is chosen. The conventional motion estimation that is implemented in the current H.264 test model chooses the motion vector that minimizes the following cost function.

$$J(\mathbf{m}, \lambda_{\text{motion}}) = SAD(s, c(\mathbf{m})) + \lambda_{\text{motion}} \cdot R(\mathbf{m}) \quad (2)$$

with $\mathbf{m} = (m_x, m_y)^T$ being the motion vector, and λ_{motion} being the Lagrange multiplier. The first term of the above cost function is the distortion term and it is given as the Sum of Absolute Difference (SAD) between the original signal s , and the reference signal $c(\mathbf{m})$. The rate term $R(\mathbf{m})$ represents the number of bits that would be used to code the motion vector $\mathbf{m}$.

In order to favour motion vectors with less interpolation complexity and penalize ones with higher complexity, we modify the conventional cost function, and use the one indicated as in Equation 3.

$$J'(\mathbf{m}, \lambda_{\text{motion}}, \lambda_{\text{ME}}) = J(\mathbf{m}, \lambda_{\text{motion}}) + \lambda_{\text{MEC}} \cdot C_{\text{ME}}(\mathbf{m}) \quad (3)$$

The proposed cost function has an additional term, $C_{\text{ME}}(\mathbf{m})$, that represents the decoding complexity of the candidate motion vector, $\mathbf{m}$. $C_{\text{ME}}(\mathbf{m})$ is basically a two dimensional array and it is calculated using the data given in Figure 3. For example, if the candidate motion vector has integer-pixel accuracy in both horizontal and vertical directions, then $C_{\text{ME}}(\mathbf{m})$ is zero. Similarly, if $\mathbf{m}$ points to location j , the value of $C_{\text{ME}}(\mathbf{m})$ is 4. The value of $C_{\text{ME}}(\mathbf{m})$ is further multiplied by the Lagrangian term λ_{MEC} , to adjust the complexity-video quality trade-off. Larger the value of λ_{MEC} , less decoding complexity the resulting bitstream has, with a higher penalty on the coding efficiency.

3.2 Mode Decision

After the motion estimation is performed for all candidate INTER modes, the coding results of the modes are compared and the one that minimizes the following Lagrangian cost function is chosen.

$$J(M, \lambda_{\text{MODE}}) = SSD(s, r) + \lambda_{\text{MODE}} \cdot R(s, c, M) \quad (4)$$

with r being the reconstruction signal for the given mode, M , and λ_{MODE} is the Lagrangian multiplier. M is referring to one of the candidate INTER modes as illustrated in Figure 4.

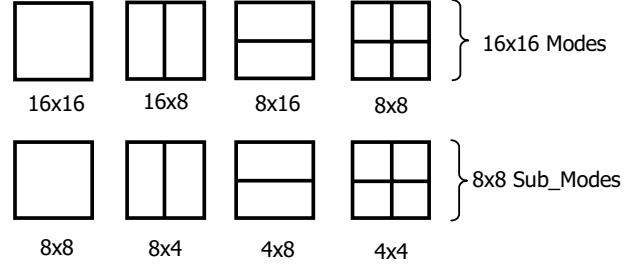


Figure 4 INTER modes supported by H.264 standard

The first term of the above cost function is the distortion term, and it is given as the Sum of Square Difference (SSD) between the original and the reconstructed signal. $R(s, c, M)$ is the rate term that represents the number of bits used to code the mode, M . In order to favour the INTER modes with less interpolation complexity, we modify Equation 4, and use the following cost function:

$$J'(M, \lambda_{\text{MODE}}) = J(M, \lambda_{\text{MODE}}) + \lambda_{\text{MDC}} \cdot C_{\text{MODE}}(M) \quad (5)$$

Similar to the proposed cost function used in Motion Estimation, Equation 5 has the additional term $C_{\text{MODE}}(M)$ representing the decoding complexities of candidate modes. $C_{\text{MODE}}(M)$ is the sum of all the interpolation complexities for all motion vectors involved in the candidate mode, and is illustrated below in Equation 6.

$$C_{\text{MODE}}(M) = \sum_{i=1}^{\text{num_of_}MV\text{'s}} C_{\text{ME}}(\mathbf{m}_i) \quad (6)$$

where $\text{num_of_}MV\text{'s}$ refers to the number of motion vectors used in the given mode. For example, if the candidate mode is INTER_8x8, in which the macroblock is divided into four blocks and each block has its own motion vector, the value of $\text{num_of_}MV\text{'s}$ is four. $C_{\text{MODE}}(M)$ is multiplied by λ_{MDC} to adjust the complexity-quality trade-off at the mode decision stage. The values for λ_{MDC} and λ_{MEC} affect the encoding performance and should be carefully selected. Results of our simulations suggest that values 2, 50 for λ_{MDC} and λ_{MEC} respectively result in the best complexity-quality trade-off. The effect of changing the values of Lagrangian parameters, λ_{MEC} and λ_{MDC} , could be observed more in the experimental results.

It should be noted that, conventional cost function is used in the proposed method when deciding between INTER and INTRA modes.

4. EXPERIMENTAL RESULTS

In order to test the performance of our scheme, we first generated the *low-complexity* bitstreams for different test sequences, using the methods described above. Then using a highly optimized H.264 decoder implementation on an ARM-11 platform, we decoded all the bitstreams and recorded the required number of processor cycles for each one. We repeated the same process using the conventional encoder, with the same encoder settings, to get the reference *RD Optimized* bitstreams.

Figure 5 presents the results for the test sequence Foreman. When compared to the *RD Optimized* bitstream, the *low complexity* bitstream has 14% less decoding complexity. The

penalty on video quality, to achieve lower complexity decoding is 0.1 dB. We also present the decoded pictures from both the bitstreams in Figure 6. It is observed that the *low complexity* bitstream has practically the same visual quality as the *RD Optimized* one.

As mentioned before, the values of λ_{MDC} and λ_{MEC} are chosen as 2, 50 respectively. However, our method allows one to decrease the decoding complexity more by allowing a higher penalty on the coding efficiency. This could be achieved by using larger values for λ_{MDC} and λ_{MEC} . In order to demonstrate the effect of changing those values, we repeated the above experiment with λ_{MDC} being 4 instead of 2, to favour modes having less decoding complexity, and generated the *lower complexity* bitstream. As it is seen in Figure 7, the decoding complexity is decreased even more, with slightly less coding efficiency.

5. CONCLUSIONS

In this work, we proposed novel encoder algorithms that are used to generate H.264 compliant decoder-friendly bitstreams. Using the proposed algorithms, it is possible to generate bitstreams requiring significantly less decoding resource consumption, with negligible effect on picture quality. This is achieved by using novel algorithms for mode decision and motion estimation that bias easy-to-decode motion vectors in a Rate-Distortion optimized fashion. It was shown that, more than 15% decoding complexity reduction is achieved with less than a 0.1 dB penalty on average video quality. It was also shown that proposed methods allow further complexity reduction, by having slightly less picture quality.

It is believed that, this approach has many important use cases in mobile multimedia systems, where the video decoder operation is often dominating the handsets power consumption.

REFERENCES

- [1] Peng Yin; Tourapis, H.-Y.C.; Tourapis, A.M.; Boyce, J. , "Fast mode decision and motion estimation for JVT/H.264", in ICIP 2003, vol.3, pg: 853-856, 4-17 Sept. 2003
- [2] J. Lee; B. Jeon, "Fast mode decision for H.264," IEEE International Conference on Multimedia and Expo, 2004., vol.2, no.pp. 1131- 1134 Vol.2, 27-30 June 2004
- [3] Horowitz, M.; Joch, A.; Kossentini, F.; Hallapuro, A., "H.264/AVC baseline profile decoder complexity analysis", in CSVT, IEEE Trans. on, vol.13, pg: 704- 716, July 2003
- [4] Lappalainen, V.; Hallapuro, A.; Hamalainen, T.D., "Complexity of optimized H.26L video decoder implementation", CSVT, IEEE Trans. on, vol.13, pg: 717- 725, July 2003
- [5] Karczewicz, M.; Hallapuro, A.; "Interpolation solution with low encoder memory requirements and low decoder complexity.", VCEG-N31, 24-27 Sept. 2001.

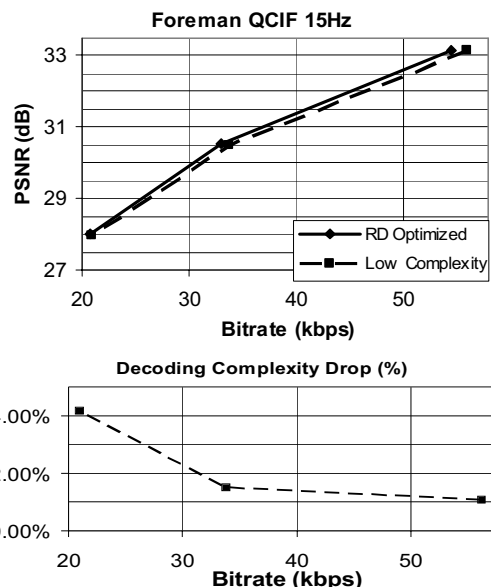


Figure 5 Analysis of the proposed method

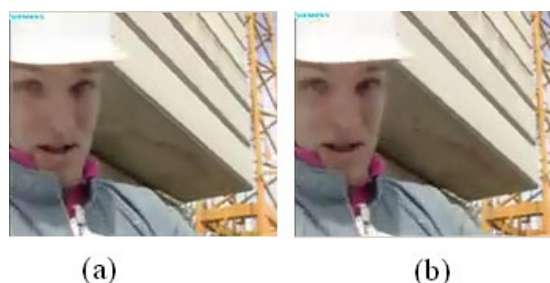


Figure 6 Reconstructed Frame # 89 Foreman
a: RD Optimized Reference b: Low Complexity

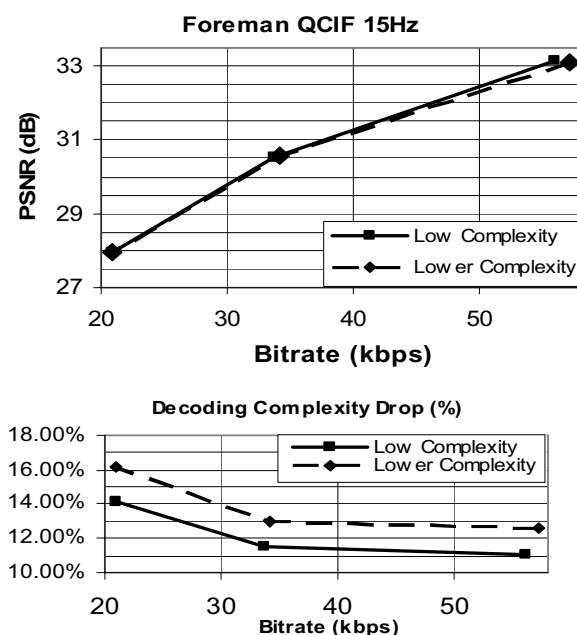


Figure 7 Effect of Changing λ_{MDC}

- [P2] Ugur, K.; Rusanovskyy, D.; Hallapuro, A.; Lainema, J.; Gabbouj, M., "Efficient implementation of adaptive interpolation filters for low complexity video coding," *Consumer Electronics, IEEE Transactions on* , vol.55, no.1, pp.257-263, February 2009

© 2009 IEEE. Reprinted with permission

Efficient Implementation of Adaptive Interpolation Filters for Low Complexity Video Coding

K. Ugur, *Member, IEEE*, D. Rusanovskyy, *Member, IEEE*, A. Hallapuro, J. Lainema, M. Gabbouj, *Senior Member, IEEE*

Abstract — *This paper presents novel methods for implementing adaptive interpolation filtering techniques for video coding on 16-bit integer architectures. The proposed methods offer significant complexity reduction over the state-of-the-art, making them especially valuable for resource constrained mobile multimedia devices, with high coding efficiency¹.*

Index Terms — Adaptive interpolation, video coding, interpolation filter.

I. INTRODUCTION

Motion Compensated Prediction (MCP) is a technique used by all popular video coding standards to exploit the temporal redundancy present in a video sequence. In MCP, each frame of a video sequence is divided into blocks and a reference frame is searched for each block to find its best match. The relative position of the best matching block with respect to the original one is called the block's motion vector. Motion Vectors may have either integer or fractional pixel accuracies. If the motion vector has fractional pixel accuracy, the decoder needs to perform interpolation to obtain the samples at sub-pixel positions. The state-of-the art video coding standard, H.264/AVC, supports motion vectors with half and quarter pixel accuracies. The half-pixel samples are obtained by using a 6-tap filter and quarter-pixel samples are obtained by bi-linear filtering using the two nearest samples at half or integer pixel positions [1].

The interpolation filter defined in H.264/AVC was designed to minimize the adverse effects of aliasing present in the input image sequence [2]. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in the literature to combat this non-stationary effect of aliasing [3-5]. Because the coefficients of the adaptive filter can take any value and thus have high dynamic range, the filtering process requires 32-bit floating point registers. This result in a serious

complexity increase for adaptive interpolation schemes compared to 16-bit integer arithmetic implementation of the H.264/AVC interpolation filter. In our previous work [5], we presented adaptive interpolation scheme utilizing directional filters and its 16-bit integer implementation. These schemes were adopted to the reference software maintained by the ITU-T VCEG standardization group.

In this paper, we present a more detailed description of the 16-bit implementation of adaptive interpolation filters and extend it by including novel tools. The proposed algorithms achieve 16-bit integer implementation by restricting the dynamic range of the filter coefficients. The filter coefficients for different sub-pixels are defined with a variable accuracy by exploiting the different statistical distributions of the filter coefficients. In addition, a novel structure is proposed that utilizes partial convolution sums to gain additional accuracy and improve the coding efficiency further. Finally, we propose an adaptive rounding scheme to decrease the rounding error of the filter coefficients. The proposed algorithms could work with any adaptive interpolation filtering structure, but as an example we implemented them on our previously proposed Directional Adaptive Interpolation Filtering (DAIF) structure. Experimental results show that the proposed method achieves 16-bit operation with no penalty on coding efficiency compared to the 32-bit floating point implementation. When compared to H.264/AVC, the proposed methods achieve up to 19% bitrate reduction, with comparable implementation complexity.

This paper is organized as follows; Section 2 provides a brief background of adaptive interpolation filters and the DAIF structure. Section 3 presents the details of the proposed tools to achieve low complexity 16-bit implementation. Section 4 presents detailed theoretical analysis on the effect of the proposed tools on interpolation quality. Section 5 presents experimental results and detailed complexity analysis. Section 6 concludes the paper.

II. ADAPTIVE INTERPOLATION FILTERS FOR VIDEO CODING

Consider Fig. 1.a., where the pixels at integer pixels are labeled by upper-case letters {A1,...,F6} within shaded boxes, and lower-case symbols {a,b,...,o} represent sub-pixels locations that are to be interpolated. In H.264/AVC, the sub-pixels at half-pixel positions (denoted with b , h , j) are obtained using a 6-tap filter. The quarter-pixels are obtained by averaging the nearest half or full-pixel samples [1].

¹ This work was supported by Nokia Research Center and the Academy of Finland, (application number 213462, Finnish Programme for Centres of Excellence in Research 2006-2011)".

Kemal Ugur, Antti Hallapuro and Jani Lainema are with the Nokia Research Centre, Nokia Corp., Tampere, Finland. (e-mail: firstname.lastname@nokia.com)

Dmytro Rusanovskyy and Moncef Gabbouj are with the Department of Signal Processing at Tampere University of Technology, Tampere, Finland. (e-mail: firstname.lastname@tut.fi).

In the adaptive interpolation schemes, the encoder calculates the optimal filter coefficients that minimize the prediction error energy for each frame and transmits them to the decoder with the rest of the frame data. The calculations of the filter coefficients are performed as follows. Firstly, the frame is encoded with a fixed filter, such as the one defined in H.264/AVC standard and the initial motion vectors are found. Using these initial motion vectors and the reference frames, the interpolation filter coefficients for each sub-pixel are calculated analytically by minimizing the prediction error energy. Finally, the reference frame is interpolated using the optimal filter and the current frame is coded. In order to reduce the number of bits used to code the filter coefficients, it is assumed that statistical properties of the image signal are symmetrical. With this assumption, the filter coefficients are assumed to be the same, in case the distance of the corresponding full-pixel to the current sub-pixel is equal. For more details on calculating the filter coefficients and filter symmetry, the reader is referred to [3]

A. Directional Adaptive Interpolation

In [3], Vatis *et al.* proposed the use of a 2D non-separable adaptive interpolation filter to reduce the prediction error energy and improve the coding efficiency of video coders. For each sub-pixel position that is not aligned with integer pixels (sub-pixel positions $e, f, g, i, j, k, m, n, o$), this scheme utilizes an 2D non-separable filter having a size of 6x6 taps. However, the improvement in the coding efficiency comes with the expense of having approximately three times higher interpolation complexity compared to the standard H.264/AVC [6]. The reasons for such complexity increase are the non-separable nature of 36-tap filters and the need to perform operations in 32-bits arithmetic. Since interpolation is already one of the most complex parts of H.264/AVC decoders, a further increase in the complexity is very problematic, especially for resource constrained devices, such as portable video players or recorders.

In [5], we proposed the Directional Adaptive Interpolation Filtering (DAIF) scheme to reduce the complexity of adaptive interpolation filtering. Instead of using 2D non-separable filters, DAIF structure uses directional filters, where the direction of the interpolation filter for different sub-pixel locations is determined based on the alignment of the corresponding sub-pixel with integer pixel samples. For example, consider Figure 1.a. and Figure 1.b. where the locations of the integer samples that are used in the filtering process are denoted by e, o and g, m ; respectively. The sub-pixel locations e, o are diagonally aligned with integer-pixels in the North-West—South-East direction. The integer-pixel positions that are aligned with these sub-pixels are $\{A1, B2, C3, D4, E5, F6\}$, thus only these integer-pixels are utilized to estimate $h(e)$ and $h(o)$ and to interpolate sub-pixels at locations e and o . Similarly, sub-pixel locations g, m are diagonally aligned with integer-pixels in the North-East—South-West direction. Therefore, integer

samples $\{A6, B5, C4, D3, E2, F1\}$ are used for interpolating at sub-pixel locations g and m . Sub-pixel locations $\{a, b, c\}$ and $\{d, h, l\}$ are horizontally and vertically aligned with integer pixels. In this case, $\{C1, C2, \dots, C6\}$ is used for interpolating at $\{a, b, c\}$ and $\{A3, B3, \dots, F3\}$ are used for interpolating at $\{d, h, l\}$. Locations $\{f, i, j, k, n\}$ are either aligned with integer-pixel samples in both diagonal directions (location j), or have no integer-pixel samples to be aligned with (positions f, i, k, n). For this reason, $\{f, i, j, k, n\}$ are interpolated using the integer-pixels that lie on a diagonal cross structure.

Since most of the sub-pixels are obtained using a single 6-tap filtering and the rest are obtained using 12-taps the interpolation complexity of DAIF is significantly less than its counterparts.

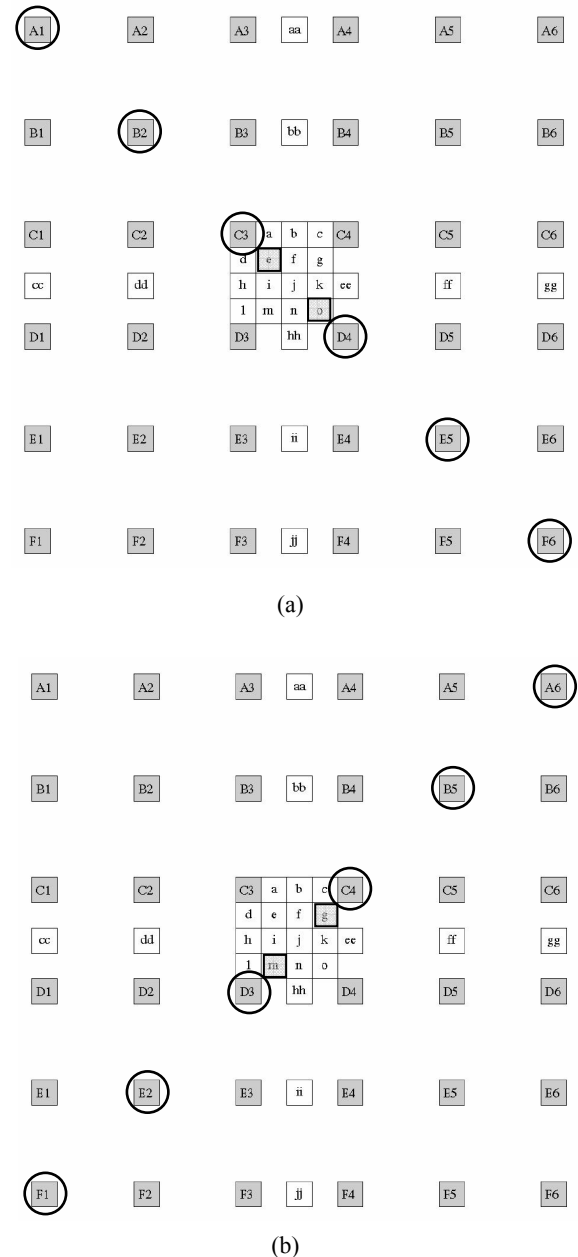


Fig. 1. DAIF structure for a) sub-pixel locations e, o and (b) g, m

III. LOW COMPLEXITY ADAPTIVE INTERPOLATION FILTERING

Because the coefficients of the adaptive filter can take any value and thus have high dynamic range, the filtering process requires 32 bit arithmetic. This result in a serious complexity increase for adaptive interpolation schemes compared to 16-bit integer arithmetic implementation of the H.264/AVC interpolation filter. One straightforward way to achieve a 16-bit operation for adaptive interpolation is to define the filter coefficients using smaller number of bits and decrease their accuracy. However, this results in a significant coding efficiency penalty and diminishes all the benefits of adaptive interpolation.

To investigate this problem, let us assume a one-dimensional 6-tap filter with coefficients in floating point arithmetic given as h_i , where i is the coefficient index. The filter coefficients in integer form are written as H_i , where Q determines the accuracy for the coefficients:

$$H_i = \text{sign}(h_i) \cdot \lfloor |h_i| \cdot 2^Q + 0.5 \rfloor \quad (1)$$

,where $\text{sign}()$ function returns -1 if the input is negative and +1 if the input is positive, and $\lfloor \cdot \rfloor$ is the floor function returning the nearest integer smaller than the input value.

The filter output is computed as follows:

$$Sp = \left(\sum_{i=0}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (2)$$

Where Y_i are the integer samples inside the filtering mask and Sp is the sub-pixel sample to be interpolated. Larger Q improves the accuracy of filter coefficients representation but increases the dynamic range of the convolution sum in (2). In order to have the convolution sum in (2) fit in a 16-bit register, the filter coefficients should be represented with maximum of 4 bits accuracy ($Q=4$). It was found empirically, that having Q less than 8 significantly reduces the coding efficiency of adaptive interpolation schemes. The following subsections describe the proposed methods in order to increase the accuracy of the filter coefficients representation and thus preserve the coding efficiency of the adaptive interpolation scheme.

A. Partial Convolution Sums

The proposed structure performs interpolation not using equation in (2) above, but partial sums. Let us rewrite Equation (2) as:

$$Sp = \left(\sum_{i=0}^2 Y_i \cdot H_i + \sum_{i=3}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (3)$$

Assume that the partial sums $\sum_{i=0}^2 Y_i \cdot H_i$ and $\sum_{i=3}^5 Y_i \cdot H_i$ are stored in 16-bit registers. The sign of the adaptive filter coefficients are unknown, thus the accumulation registers $res1$ and $res2$ should support signed number representation. However, for typical interpolation filter coefficients and data samples, these partial sums will always be positive. We can gain 1 bit accuracy for filter coefficients representation by clipping negative $res1$ and $res2$ (i.e. if they are negative, set

their values to zero) and performing the final addition using unsigned, instead of signed arithmetic. This process is illustrated in Figure 2.

In order to see how this process increases the accuracy of filter coefficients, assume that the filter coefficients are represented in integer form with 7 bits accuracy (i.e. $Q = 7$). For this example, we consider the interpolation filter defined in the H.264/AVC, denoted with h_{AVC} and H_{AVC} for floating point and in integer form, respectively, and are given as:

$$h_{AVC} = [1, -5, 20, 20, -5, 1]/32$$

$$H_{AVC} = [4, -20, 80, 80, -20, 4]$$

In order to see if the 16-bit register used for the interpolation given in Eq. 2 overflows, we need to find out the maximal value the convolution sum could take before shifting and see if it fits in the 16-bit register. The maximal value happens when the image samples that are multiplied with positive filter coefficients are 255 (assuming an 8-bit image), and the image samples that are multiplied with negative filter coefficients are 0. In other words, the maximum value for the convolution sum happens for filter H_{AVC} when the image samples Y take the following values:

$$Y = [255, 0, 255, 255, 0, 255]$$

In this case, the convolution sum $\sum_{i=0}^5 Y_i \cdot H_i + 2^{Q-1}$ takes the value 42904 and requires a 17-bit signed register (16 bits for magnitude and 1 bit for sign), and therefore does not fit into a 16-bit register.

For the same example, let us now use partial convolution sums and clipping the intermediate values as described above. Each partial convolution sum given in Eq. 3 would take the value 21420 and fits in a 16-bit signed register. As the final summation is done using unsigned addition, and the magnitude of 42904 could be defined in a 16-bit, the interpolation operation does not overflow the 16-bit registers.

It should be noted that clipping the partial sums below zero loses some information to gain additional bit accuracy for the coefficients. However, for typical interpolation filters and input samples, partial sums would very often be greater than zero and this clipping is exploiting this fact. More detailed analysis of the effect of this step on interpolation quality is given in Section 4.

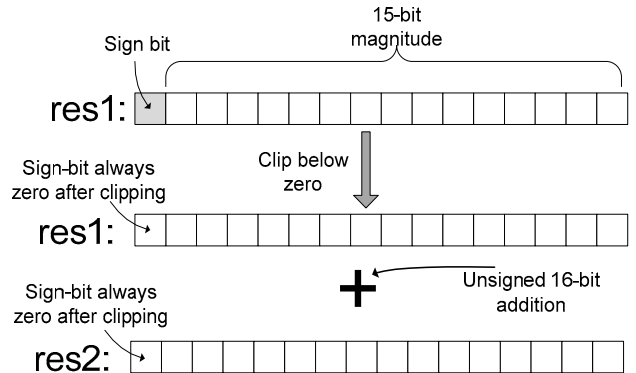


Fig. 2. Illustration of Intermediate Clipping

B. Restriction on Filter Coefficients

In order for partial sums given in Eq. 3 to fit in 16-bit registers, we define two requirements on the filter coefficient values:

$$\left(\sum_{i=j}^{j+2} \max(0, H_i) < 128 \right) \text{AND} \left(\sum_{i=j}^{j+2} \min(0, H_i) > -128 \right) \quad (4)$$

where j is 0 and 3 for a 6-tap filter. With these restrictions in place, the accuracy of the filter coefficients is increased from 4 bits to 7 bits and still guaranteeing a 16-bit integer implementation. These requirements reduce the dynamic range of filter coefficients to ensure a 16-bit operation. Meanwhile, the dynamic range defined with the above restrictions cover typical interpolation filters used in video coding so that the coding efficiency is kept high. There are mainly two families of interpolation filters for which these requirements cannot be met. These are:

1. Interpolation filters with a gain larger than unity. Consider the case again of using the interpolation filter defined in H.264/AVC using 7 bits of accuracy. But instead of having unity gain, let's assume the gain is 2. The filter coefficients in integer form is given as:
 $H_{AVC} = [8, -40, 160, 160, -40, 8]$
 These filter coefficients do not meet the restrictions defined in Eq. 4. However, utilizing interpolation filters with a gain that is very different than unity in video coding happens in rare cases. And for those cases the video coder may choose to use different tools, such as Weighted Prediction [1].
2. Interpolation filters to obtain sub-pixels that are positioned very close to integer position. For these cases, the filter coefficients become highly asymmetrical and one of the filter coefficients would be significantly larger than the rest of the coefficients. This asymmetry in filter coefficients violates one of the restrictions given in Eq. 4. To illustrate this behavior, let's consider a widely used interpolation kernel that utilizes a 3-lobed Lanczos-windowed sinc function. This interpolation kernel is defined as:

$$\text{Lanczos3}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} \cdot \frac{\sin(\pi \frac{x}{3})}{\pi \frac{x}{3}}, & |x| < 3 \\ 0, & |x| \geq 3 \end{cases} \quad (5)$$

where x is the distance between the sub-pixel and the integer samples. This kernel yields the following filter coefficients in a floating point representation to obtain a sub-pixel at position 92/100:

$$h = [0.0143, -0.0586, 0.9883, 0.0733, -0.0186, 0.0007]$$

$$H = [2, -8, 127, 9, -2, 0]$$

As seen above, the filter coefficients used to obtain a sub-pixel position very close to the integer sample does not fulfill the restriction given in Eq. 4.

For the rare cases, where these restrictions can not be met, the standard H.264/AVC interpolation filter coefficients are used for the adaptive filter. As it will be seen in Section 5 these restrictions do not deteriorate the performance of adaptive interpolation filters in video coding. The effect of reducing the dynamic range of the filter coefficients on interpolation quality is also analyzed further in Section 4.

C. Filter Coefficients with Variable Accuracy

Additional bit accuracy for the filter coefficients is gained by observing the fact that the statistical distribution of the coefficients of different sub-pixels is different. If a sub-pixel is interpolated using a 6-tap filter, the statistical distribution of the coefficients would be different than that of the 12-tap interpolation filter. For example, the coefficients of a 12-tap filter most often fall in the range $[-0.5, 0.5]$, instead of $[-1, 1]$. We exploit this fact and restrict the filter coefficients of sub-pixel locations that use a higher number of taps. By using a restricted range, additional accuracy is gained for the respective sub-pixel locations.

D. Adaptive Rounding of Filter Coefficients

The rounding operation given in Eq. 1 is not optimal in terms of minimizing the total rounding error of the interpolation filter. For some filters, the total rounding error accumulates to a large value, and hence penalizes the coding efficiency. To alleviate this effect, we add a correction factor to a number of filter coefficients after the rounding operation and reduce the total rounding error. This is done as follows. After the integer filter coefficients are computed using Equation 1, the total rounding error of the filter is found using:

$$\text{rnd_error} = \sum_{i=0}^5 (h_i \cdot 2^0 - H_i) \quad (6)$$

If the total rounding error is positive, this means that most of the coefficients were down-rounded. +1 is added to the coefficient that has the largest down-rounding error, then to the coefficient that has the second largest down-rounding error etc. This is repeated until the magnitude of the total rounding error falls below 0.5. Same process is done using the correction factor -1 for negative rounding errors. After this process, the modified coefficients have a higher rounding error, but the total rounding error of the filter is decreased.

In order to illustrate how adaptive rounding works, consider the following filter obtained using the Lanczos windowed sinc interpolation kernel, given in Eq. 5:

$$h = [0.0034, -0.0435, 0.1716, 0.9481, -0.1064, 0.0251]$$

For the case of regular rounding (rounding using Eq. 1), filter coefficients in integer form with a 7-bit accuracy are given as:

$$H = [0, -6, 22, 121, -14, 3]$$

Using Eq. 5, the total rounding error is 1.7897 and the rounding error for each coefficient is found as:

$$\text{rnd_error}_H = [0.4379, 0.4356, -0.0413, 0.3579, 0.3872, 0.2125]$$

As seen above, the rounding error of all the coefficients is less than 0.5, but the total rounding error is 1.7897. Using the

above procedure, we first add +1 to the filter coefficient that has the largest down-rounding error, which is the first coefficient. Then the total rounding error decreases to 0.7897. We need to add +1 for one more filter coefficient that has the second largest down rounding error, and that is the second filter coefficient. This way the total rounding error becomes -0.2103. The modified filter coefficients and their respective rounding errors are then given as:

$$H' = [1, -5, 22, 121, -14, 3]$$

$$\text{rnd_error}_{H'} = [-0.5621, -0.5644, -0.0413, 0.3579, 0.3872, 0.2125]$$

As seen clearly in the above example, the magnitude of the total rounding error of the filter is decreased from 1.7897 to 0.2103, even if some of the filter coefficients have a higher rounding error. The frequency response of the two rounding methods, rounding to nearest integer given in Eq. 1 and the proposed rounding as described above are presented in Fig. 3. As seen in Figure 3, rounding to the nearest integer value changes the frequency response of the filter significantly, but the frequency response comes closer to the original one when the proposed rounding is used. More important than the change in frequency response, adaptive rounding corrects the gain of the interpolation filter. In the above example, the original filter with floating point accuracy has unity gain; whereas filter h obtained using rounding to the nearest integer has a gain of 0.984375. In video coding, this change in gain means the encoder has to code more DC residual and degrade its coding efficiency. After adaptive rounding is applied, this gain is moved back to unity, which would help the video coder improve its coding efficiency.

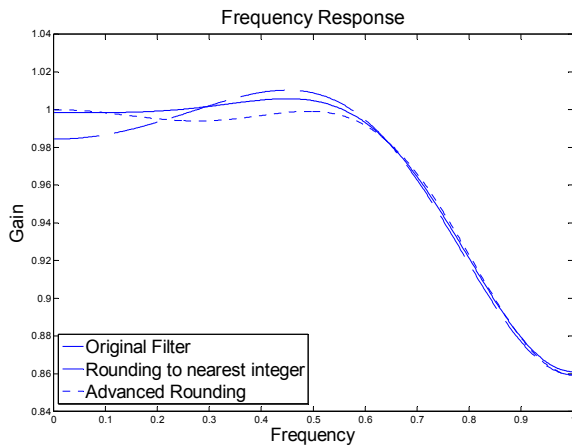


Fig. 3. Frequency response of filters with different rounding.

IV. EFFECT ON INTERPOLATION QUALITY

In this section, we analyze the effect of some of the proposed tools on the interpolation quality in more detail. The proposed tools are used in the context of video coding, where the filter coefficients are found at each frame by minimizing the prediction error energy. In order to simulate this process, we first define lanczos-windowed sinc function as the interpolation kernel. The coefficients of the adaptive interpolation filter are then given as a function of the sub-pixel

position using this interpolation kernel. As an example see Fig. 4, where the coefficients of the interpolation filter are given for the quarter-pixel position. We change the sub-pixel positions to vary the adaptive filter coefficients and analyze the effect of each proposed tool on this adaptive filter.

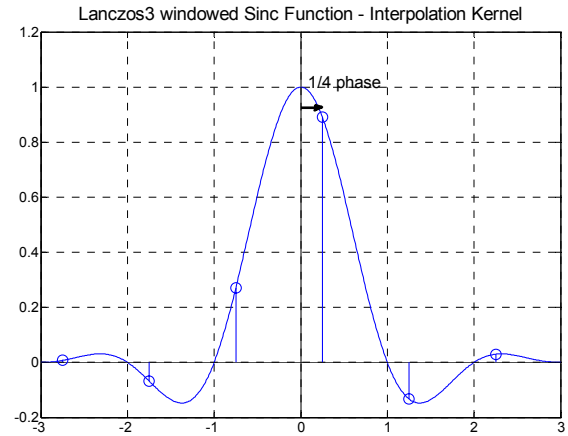


Fig. 4. Lanczos3 interpolation kernel and the respective filter coefficients for quarter-pixel position

A. Effect of restricting filter coefficients

We first analyze the effect of restricting the dynamic range of the filter coefficients using Eq. 4. This is done by utilizing different interpolation filters and checking if their respective coefficients satisfy the constraints given in Eq. 4. For this purpose, we use the lanczos-windowed sinc function given in Eq. 5, for sub-pixels whose position varies by increments of $1/100$. In other words, we utilize 99 different interpolation filters corresponding to these sub-pixel positions located between two consecutive integer samples. If a high number of sub-pixel positions do not satisfy the constraints, it means that those sub-pixel positions can not be interpolated using the proposed tool. We also analyze how the gain of the interpolation filter changes the number of sub-pixel positions satisfying the constraints. The results of this experiment are given in Table 1.

TABLE 1
EFFECT OF RESTRICTING THE FILTER COEFFICIENTS

Filter Gain	Number of filters not satisfying Equation 4
1	10: [2,3,4,7,8,92,93,96,97,98]/100
1.1	50
0.9	0

As seen in Table 1, for the unity gain, filters for the 10 sub-pixel positions out of 99 do not satisfy the constraints given in Eq. 4. As previously noted, those sub-pixel positions are the ones that are located very closely to an integer pixel. It should be noted that, in the context of video coding this much of fine-grain motion is not usually needed, and the fact that these sub-pixel positions do not satisfy the dynamic range requirements does not hinder the coding efficiency of a video coder. This is also shown in Section 5, where the proposed tools are integrated in a video coder.

B. Effect of the proposed rounding

In order to analyze the effect of the proposed rounding scheme, we again use the same lanczos-windowed sinc function given in Eq. 5. Each sub-pixel position is interpolated first using floating point filter coefficients. Then the same sub-pixel position is interpolated using integer filter coefficients obtained with rounding to the nearest integer, as given in Eq. 1. Finally, the corresponding sub-pixel positions are interpolated using the filter coefficients obtained by the proposed rounding operation. The PSNR between the floating point interpolation and the two respective integer interpolations is plotted in Fig. 5.

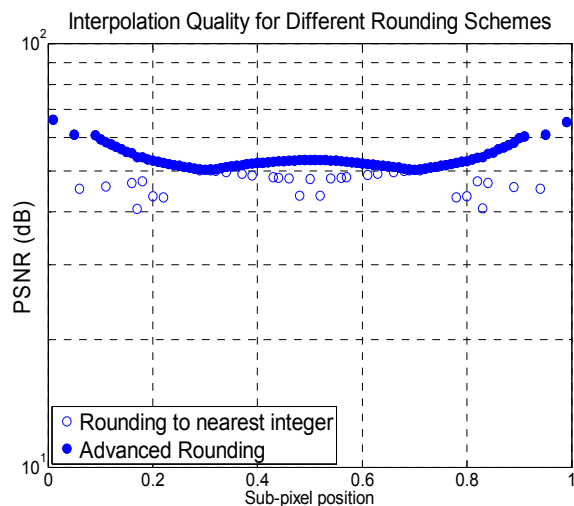


Fig. 5. Effect of rounding on interpolation quality

As seen in Fig. 5, the proposed rounding operation constantly improves the interpolation quality compared to rounding to the nearest integer scheme. For some sub-pixels the difference in PSNR is significantly high, and exceeds 1 dB.

V. EXPERIMENTAL RESULTS

A. Coding Efficiency Analysis

To evaluate the performance of the proposed algorithms in the context of video coding, we implemented them using the Directional Adaptive Interpolation Filtering structure [5] and performed extensive simulations. For the simulations, common conditions recommended by the ITU-T VCEG standardization group are used [7]. Table 2 summarizes the results for the Baseline profile settings, for 720p sequences. As seen in Table 2, the proposed methods achieve 16-bit operations with no penalty on coding efficiency, compared to 32-bit implementation. For some sequences, minor gains are also realized due to the adaptive rounding of the filter coefficients, and the lower dynamic range of filter coefficients that reduces the required number of bits to code them. When compared to H.264/AVC, the proposed methods achieve significant bitrate reduction (up to 19.39% gain).

TABLE 2
CODING EFFICIENCY OF THE PROPOSED METHOD

Sequence	DAIF-16bits vs. H.264/AVC	DAIF-16bit vs. DAIF-32 bit
	Δ Bitrate [%]	Δ Bitrate [%]
City	-16.51	-0.95
Bigships	-11.42	0.2
Crew	-19.39	-1.24
Shuttlestart	-19.31	-0.66
Night	-5.86	0.14

B. Computational Complexity Analysis

In order to estimate the complexity of the proposed tools, we follow a similar methodology previously used to analyze the decoder complexity of H.264/AVC [8]. This methodology uses two basic steps to estimate the computational complexity of the interpolation filters. In step one, we count the number of arithmetic operations required to execute a filter for each sub-pixel. It is assumed that interpolation is done on a block-based and the number of arithmetic operations is counted for both 4x4 and 8x8 block sizes. In step two, we multiply the complexity numbers found in step one by the appropriate entry in a sub-function frequency table. The sub-function frequency table contains information regarding the use frequency of various sub-functions, the filters for each sub-sample location in this work, over the set of common condition sequences. The sub-function frequency table is generated by collecting the statistics for each sub-pixel location for both the H.264 and DAIF case using the Baseline profile's common conditions, as defined in [7]. To get more realistic numbers, it is assumed that motion blocks of size equal to or greater than 8x8 use the 8x8 interpolation whereas smaller block sizes use 4x4 interpolation.

Table 3 presents the number of arithmetic operations in 16-bits required to obtain each sub-pixel (i.e. the result of Step-1 as outline above). As seen in Table 3, the maximum increase in operations for DAIF-16-bit compared to H.264 interpolation occurs at locations b and h that are horizontally and vertically aligned half-pixel samples, respectively. On the other hand, diagonally aligned locations, e, g, m and o provide DAIF with a 43.5% computational advantage, as those locations are obtained using a single one-dimensional filter. On average, DAIF-16-bit has 21.7% less complexity than H.264 interpolation for the 4x4 case and 13.7% less complexity for the 8x8 case.

TABLE 3
NUMBER OF 16-BIT ARITHMETIC OPERATIONS REQUIRED TO INTERPOLATE EACH SUB-PIXEL

Sub-pixel position	H.264/AVC		DAIF-16bit	
	4x4	8x8	4x4	8x8
b,h	10	10	17	10
a,c,d,l	13	13	17	13
e,g,m,o	23	23	17	17
j	32.5	26.25	35	35
f,i,k,n	35.5	29.25	35	35
Average for sub-pixels	22.56	20.48	23	23

When the complexity numbers of Table 3 are weighted with the appropriate entry in the sub-function frequency table, we arrive to a realistic estimate of complexity of the proposed tools, in which DAIF-16bits is 7% less complexity than H.264/AVC interpolation.

It should be noted that, the above analysis is based on counting the number of arithmetic operations and does not take into account several issues such as memory bandwidth, parallelism etc. A more detailed study analyzing the complexity of DAIF-16-bit, taking into account Single-Instruction-Multiple-Data (SIMD) parallelism, can be found in [9].

VI. CONCLUSIONS

In this paper, novel algorithms are proposed that guarantee 16-bit integer implementation for adaptive interpolation filtering schemes with a high coding efficiency. When compared to H.264/AVC, the proposed methods achieve up to 19% bitrate reduction, with a lower computational complexity.

REFERENCES

- [1] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Rec. H.264 & ISO/IEC 14496-10
- [2] T. Wedi and H. G. Musmann, "Motion- and Aliasing-Compensated Prediction for Hybrid Video Coding", IEEE Trans. on CSVT, vol. 13, No. 7, July 2003.
- [3] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion-and Aliasing-compensated Prediction using a Two-Dimensional Non-Separable Adaptive Wiener Interpolation Filter", in Proc. IEEE ICIP, Sept. 2005.
- [4] S. Wittman and T. Wedi, "Separable adaptive interpolation filter", ITU-T SG16/Q6, Doc. C-0219 July, 2007.
- [5] D. Rusanovskyy, K. Ugur, A. Hallapuro, J. Lainema, M. Gabbouj, "Video Coding with Low Complexity Directional Adaptive Interpolation Filters", *accepted*, IEEE Trans on CSVT.
- [6] Y. Vatis and J. Ostermann, "Comparison of complexity between twodimensional non-separable adaptive interpolation filter and standard wiener filter", ITU-T SGI 6/Q.6 Doc. VCEG-AA11, Nice, France, October 2005.
- [7] T.K Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments", ITU-T Q6/SG16, VCEG-AE010, Marrakech, Morocco, January 2007.
- [8] Complexity Analysis: H.264 vs. Computationally Efficient H.264, S. Perschau and M. Horowitz, ITU SG16 COM 16-D 140-E, presented in Geneva, CH, 26 July - 5 August 2005.
- [9] A. Hallapuro, K. Ugur, D. Rusanovskyy, J. Lainema, "Complexity Analysis: Directional Adaptive Interpolation Filters vs. H.264 Interpolation", ITU-T SGI 6/Q.6 Doc. C438, Geneva, Switzerland, April-May 2008.



Kemal Ugur received his M.Sc. degree in electrical and computer engineering from University of British Columbia, Canada in 2003 and joined Nokia Research Center in year 2004. Currently, he is working as a member of research staff and leading a project on next generation video coding technologies. Since he joined Nokia, he has been actively participating to several standardization forums, such as Joint Video Team (JVT) for the standardization of Multiview Video Coding extension of H.264/AVC, Video Coding Experts Group (VCEG) for exploration towards next generation video coding standard and 3GPP for mobile broadcast and multicast standard. He has more than 20 publications in academic conferences and journals and around 20 patent applications. In parallel, he is pursuing his PhD degree at Tampere University of Technology. He is a member of the research team that won the highly-prestigious Nokia Quality Award in the year 2006.



Dmytro Rusanovskyy received the Dipl.Eng. degree from National University of Radio Electronics, Kharkiv, Ukraine in 2000, and the M.Sc. degree in computer science from University of Jyväskylä, Finland in 2002. Currently, he is working as an external researcher in Nokia Research Center, and pursuing his PhD degree at Tampere University of Technology. His research interests include image, video processing and coding. He has been actively contributing to the work of ITU-T Video Coding Experts Group (VCEG) and he is an author/co-author around of 10 academic papers, several patent applications and adopted VCEG proposals.

Antti Hallapuro joined Nokia Research Center in 1998 where he has been working on video coding related topics. He has participated in H.264 video codec standardization and is author of several input documents. He has been involved in productization of H.264 codec in various forms. His interests are in practical video coding and processing algorithms and their high performance implementations.



Jani Lainema received the M.Sc. degree in computer science from the Tampere University of Technology, Finland in 1996. He joined the Visual Communications Laboratory of Nokia Research Center in 1996. Since then he has contributed to the designs of ITU-T's and MPEG's video coding standards as well as to the evolution of different multimedia service standards in 3GPP, DVB and DLNA. Recently he has been working as a Research Leader and Principal Member of Research Staff at Nokia Research Center Tampere. His research interests include video, image and graphics coding and communications, as well as practical applications of game theory.



Moncef Gabbouj (M'85-SM'95) received his BS degree in electrical engineering in 1985 from Oklahoma State University, Stillwater, and his MS and PhD degrees in electrical engineering from Purdue University, West Lafayette, Indiana, in 1986 and 1989, respectively. Dr. Gabbouj is a Professor at the Department of Signal Processing at Tampere University of Technology, Tampere, Finland. He was Head of the Department during 2002-2007. Dr. Gabbouj was a visiting professor at the American University of Sharjah, UAE, in 2007-2008 and Senior Research Fellow of the Academy of Finland in 1997-1998 and 2007-2008. His research interests include multimedia content-based analysis, indexing and retrieval; nonlinear signal and image processing and analysis; and video processing and coding.

Dr. Gabbouj served as Distinguished Lecturer for the IEEE Circuits and Systems Society in 2004-2005. He served as associate editor of the *IEEE Transactions on Image Processing*, and was guest editor of *Multimedia Tools and Applications*, the European journal *Applied Signal Processing*. He is the past chairman of the IEEE Finland Section, the IEEE CAS Society, Technical Committee on DSP, and the IEEE SP/CAS Finland Chapter.

Dr. Gabbouj was the recipient of the 2005 Nokia Foundation Recognition Award and co-recipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems and co-recipient of the NORSIG 94 Best Paper Award from the 1994 Nordic Signal Processing Symposium. He is a member of IEEE SP and CAS societies.

- [P3] Rusanovskyy, D.; Ugur, K.; Gabbouj, M., "Fast encoding algorithms for video coding with adaptive interpolation filters," *Multimedia Signal Processing, 2008 IEEE 10th Workshop on* , vol., no., pp.317-321, 8-10 Oct. 2008

© 2008 IEEE. Reprinted with permission

Fast Encoding Algorithms for Video Coding with Adaptive Interpolation Filters

Dmytro Rusanovskyy¹, Kemal Ugur², Moncef Gabbouj¹

¹Tampere University of Technology, Finland

¹ dmytro.rusanovskyy@tut.fi

³ moncef.gabbouj@tut.fi

²Nokia Research Center, Finland

² kemal.ugur@nokia.com

Abstract—In order to compensate for the temporally changing effect of aliasing and improve the coding efficiency of video coders, adaptive interpolation filtering schemes have been recently proposed. In such schemes, encoder computes the interpolation filter coefficients for each frame and then re-encodes the frame with the new adaptive filter. However, the coding efficiency benefit comes with the expense of increased encoding complexity due to this additional encoding pass. In this paper, we present two novel algorithms to reduce the encoding complexity of adaptive interpolation filtering schemes. First algorithm reduces the complexity of the second encoding pass by using a very lightweight motion estimation algorithm that reuses the data already computed in the first encoding pass. Second algorithm eliminates the second coding pass and re-uses the filter coefficients already computed for previous frames. Experimental results show that the proposed methods achieve between 1.5 to 2 times encoding complexity reduction with practically negligible penalty on coding efficiency.

I. INTRODUCTION

The interpolation filter defined in H.264/AVC is designed to minimize the adverse effects of aliasing present in the input image sequence. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive Wiener interpolation filters that change the filter coefficients at each frame have been proposed in literature to combat this non-stationary effect of aliasing [1][2][3]. In [1], Vatis et. al. proposed a scheme that uses 2D non-separable adaptive interpolation filter, a scheme using separable filters was proposed in [2] and finally adaptive interpolation using directional filters was proposed in [3]. These schemes provide similar coding gains (up-to 1.1dB) over the standard H.264/AVC scheme and they have been adopted by ITU-T VCEG standardization group in the reference KTA software [8].

In spite of difference in filter structures, these algorithms share the same coding conception, and require two encoding passes for each P- or B-frame as illustrated in Figure 1. In the first coding pass (Encoding 1), the current frame is coded with the reference frame interpolated with static filter. During this pass, motion estimation (ME) and rate distortion optimization (RDO) methods are performed and motion prediction information (motion vectors and macroblock partitions) are found. Using this information, encoder then calculates the coefficients of the adaptive filter, by solving system of linear

equations that minimize the prediction error energy (reader is referred to [1] for more details). Next, frame is re-encoded including ME and RDO using the computed adaptive interpolation filter.

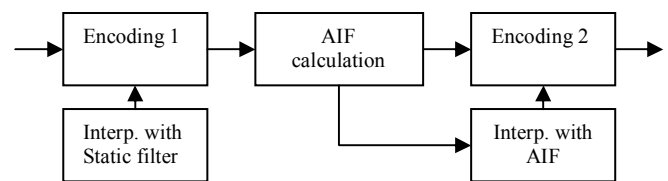


Fig. 1. Flowchart of video encoder with adaptive interpolation

It is clear from Figure 1 that increase in complexity compared to encoding with static filter is due to three additional steps. Those are the AIF calculation step, interpolation of the reference frame with the new filter step and the second encoding pass. In our earlier work, we significantly reduced the complexity of the AIF calculation step [4]. In this work we study simplifying the second encoding pass and propose two novel methods to significantly reduce its complexity.

For the first method, we assume that the motion information obtained by the static filter in the first pass is correlated to the motion information obtained by the adaptive filter in the second pass. This correlation is exploited and the complexity of motion estimation in the second encoding pass is reduced by reusing many of the coding results of the first pass. For the second method, we assume that although aliasing is not stationary, it does not undergo a fast temporal variation. Using this assumption we do not perform the second encoding pass after the adaptive filter is calculated. Instead we use the adaptive filter computed for the current frame to code the subsequent frame. This way, single-pass coding is achieved for the adaptive interpolation filtering schemes.

This paper is organized as follows; Section 2 describes video coding with adaptive Wiener interpolation filtering. Section 3 presents the details of the proposed fast encoding algorithms. Experimental results are given in Section 4, and Section 5 concludes the paper.

II. VIDEO CODING WITH ADAPTIVE INTERPOLATION FILTERING

Figure 2 shows a generalized block diagram of a hybrid video coding scheme, e.g. H.264/AVC that uses adaptive interpolation filtering. The current image X is first predicted by motion compensated prediction (MCP) from the already encoded reference frame Y . The result of the motion compensated prediction is the prediction X' . The difference between current image X and prediction X' is the prediction error E' . In order to minimize the prediction error E' , an independent filter for each different sub-pixel position Sp is calculated (denoted as $h(Sp)$). Filter coefficients are computed by solving the system of Wiener-Hopf equations (1), which are constructed from local signal statistics of Y and X using the motion vectors from the first pass. The system of linear equations needed to calculate such N-tap Wiener filter $h(Sp)$ is given as:

$$\sum_{t=0}^{N-1} R_{t-i,i}^r(Sp) \cdot h_t(Sp) = R_i^{cr}(Sp), \quad i, t = 0, \dots, N-1 \quad (1)$$

where R^{cr} and R^r are estimations of local cross-correlation and auto-correlation functions averaged over the reference (Y) and the current (X) images. Solving (1) provide a filter optimal in terms of minimal prediction error E' , in other words a Wiener interpolation filter.

After the adaptive filter h is calculated, reference frame is interpolated with the new filter and the second encoding pass is performed.

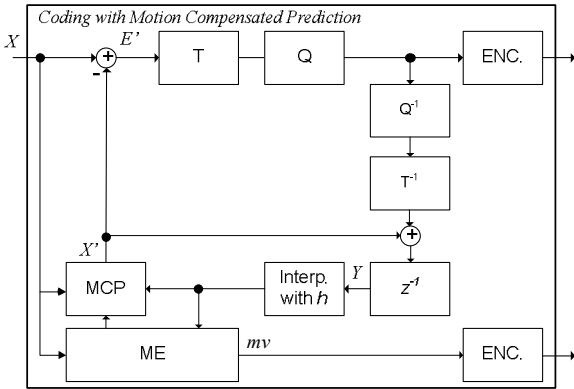


Fig. 2 Flowchart of video encoder with adaptive interpolation

III. PROPOSED ENCODER ALGORITHMS

As discussed above, AIF schemes have high encoding complexity mainly due to the need to compute the adaptive filter and the additional encoding pass with the new calculated filter. The complexity of the filter calculation process could be reduced by using data from subset of the macroblocks (see [4] for more details). To reduce the complexity of AIF due to the additional encoding pass, two fast encoding algorithms are presented. First algorithm assumes that the motion information obtained by the static filter in the first pass is highly correlated with the motion information obtained by the adaptive filter in the second pass. By exploiting this correlation, very lightweight motion estimation could be

performed in the second encoding pass by reusing many of the coding results of the first pass. Second algorithm assumes that the effect of aliasing has a content depend nature and does not undergo a fast temporal variation. Using this assumption we do not perform the second encoding pass after the adaptive filter is calculated. Instead we use the computed adaptive filter to code the subsequent frame. This way, second pass is completely eliminated and single-pass coding is achieved using adaptive interpolation. Details of the proposed algorithms are given in the next subsections.

A. Coding with Restricted Second Pass

Our first design is based on the assumption that a full-scale ME and RDO during the second coding pass are unnecessary and they can be restricted to operate over fractional-pixel locations only. To verify our assumption, we performed extensive simulations over VCEG test sequences as defined in [10] and compared the resulting motion vectors from the first and second pass that use static and adaptive filter respectively. The motion vectors resulting from the second pass could differ from the first pass motion vectors in three possible ways. The first case is that the motion vector estimated for a coded block at the second coding pass stays the same as the motion vector selected at the first coding pass. Second case is that the motion vector selected at the second coding pass can be equal to motion vector from the first pass up to integer-pixel accuracy and differ only in the fractional components. Thirdly, second pass results a completely different motion vectors estimated at the first pass either due to an RDO mode change or due to a different ME decision.

The results of this simulation and the probabilities of each one of the three possible cases are shown in Table-1. Analysis of Table I show that on average only 11.7% of the coded blocks retrieve completely new motion vectors at the second coding pass. Whereas, 60% of coded blocks have the same motion vectors and 28% of blocks retrieve motion vectors different in fractional component only. These results show that ME on integer-pixel locations at the second coding pass is unnecessary for 88% of the coded blocks on average. Thus, we can avoid ME on integer-pixel locations at the second coding pass and use motion information from the first coding pass instead, without significant penalty on coding efficiency.

TABLE I
MOTION VECTOR VARIATIONS BETWEEN THE FIRST AND THE SECOND CODING PASSES

Sequence	No Change of MVs (%)	Fractional Components Change, (%)	Complete Change of MVs, (%)
Container	79.49	19.96	0.55
ForemanQ	55.89	28.63	15.49
Paris	79.73	14.60	5.67
Foreman	47.22	29.26	23.52
Mobile	76.82	16.40	6.78
Tempete	71.17	25.24	3.59
BigShips	48.97	42.47	8.55
City	62.71	24.76	12.52
Crew	34.46	45.47	20.07
Night	49.09	32.18	18.73
ShuttleStart	52.63	33.12	14.25
Average	59.83	28.37	11.7

Using the assumption described above, proposed algorithm is designed as follows.

- 1) *Initial Interpolation*: Interpolate the reference image with the static interpolation filter.
- 2) *First-Pass Encoding*: Encode the input frame, perform ME and RDO.
- 3) *Store Motion Information*: Store motion data (best motion vectors at integer-pixel accuracy) for each of macroblocks' partitions:

$$MD = \{mv\}_S,$$

where $S = \{16 \times 16, 16 \times 8, 8 \times 16, 8 \times 8, 4 \times 8, 8 \times 4 \text{ and } 4 \times 4\}$ is a set of all available partitions.

- 4) *Calculate AIF*: Calculate the optimal filter coefficients by minimizing the prediction error using the motion information (optimal modes, partitions and motion vectors) obtained at the first step.

- 5) *Adaptive Interpolation*: Interpolate the reference frame with the computed adaptive filter.

- 6) *Second Encoding Pass*: Avoid ME on integer-pixel locations and use instead motion data from the Step 3. For each macroblock partition we re-use integer-pixel motion vectors preserved in the Step 3 and refine the fractional component of the motion vector. This refinement is illustrated in Figure 3. First, eight half-pixel locations neighboring the integer motion vector is searched, as shown in Fig. 3.a, and then similarly the eight quarter-pixel locations neighboring the best half-pixel location are searched.

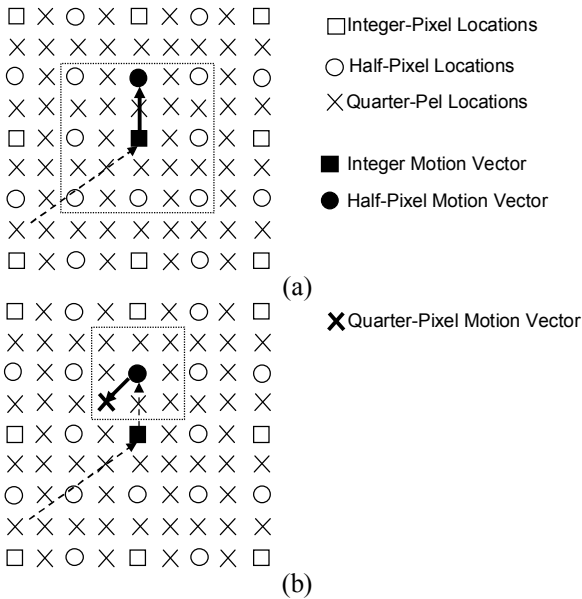


Fig. 3. Refinement of fractional component of motion vectors (a) half-pixel refinement, (b) quarter-pixel refinement

Based on the analysis reported in [5]-[7], the most time-consuming coding modules in a typical H.264/AVC video encoder include the integer pixel Motion Estimation (ME) and Rate-Distortion Optimization (RDO). These modules typically

consume 60% (1 reference frame) to 80% (5 reference frames) of the total encoding time [5], and the proportion can be much higher with larger search range. Whereas quarter-pixel ME is usually estimated as 10% of the total processing time [6]. Based on this data, the encoding complexity of the second encoding pass is significantly reduced (up-to 50-70%) by avoiding the integer-pixel ME.

B. Single-Pass Coding with Adaptive Interpolation

First method described above reduces the encoding complexity by performing lightweight motion estimation in the second encoding pass. The second proposed algorithm reduces the encoding complexity even further by completely removing the second pass. This is achieved based on the observation that in majority of the cases content does not change dramatically from frame to frame, thus the effect of aliasing stays almost constant between subsequent frames. Thus, we assume that an adaptive filter computed for frame N , could accurately compensate the effect of aliasing for frame $N+1$. The proposed algorithm could be described as:

- 1) *Initialization 1*: Initialize interpolation filter h in Fig. 1 with static filter coefficients.
- 2) *Interpolation*: Interpolate the reference frame Y with h .
- 3) *Encoding*: Encode current frame X with reference frame Y .
- 4) *Calculate AIF*: Calculate the filter coefficients that minimize prediction error for frame X by using the coding results of previous step.
- 5) *Initialization 2*: Initialize interpolation filter h with the computed filter, go on to next frame and do Step 2.

IV. EXPERIMENTAL RESULTS

Performances of the proposed fast encoder algorithms are evaluated by comparing them against Directional Adaptive Interpolation Filtering (DAIF) without any encoder optimizations, and against the static H.264/AVC interpolation filter [9]. The simulations are performed using the Baseline common conditions and IPP prediction structure as detailed in [10] using a single reference frame. The complexity reduction is estimated by measuring the execution times for all mentioned algorithms using ANSI C instructions.

The coding efficiency results of the proposed algorithms are presented in Table I, with respect to static interpolation filter. In addition, exemplary rate-distortion curves are given for "City" and "Crew" in Fig. 4 and 5 respectively. Table III presents the achieved complexity reduction of the proposed algorithms with respect to not using any optimizations at all. *Fast1* and *Fast2* in Tables II and III refer to Restricted Second Pass, and One Pass Coding methods respectively as described in previous section.

As seen from Tables II and III, video coding with adaptive interpolation provide a significant coding gain over the static interpolation. Two pass video coding with DAIF [3] reduces the bit-rate by 11% for high resolution videos but it is also around 2.5 times more complicated than encoding with static filter. The fast encoding algorithm *Fast1* with restricted ME

and RDO provides very close to two-pass DAIF coding performance whereas reducing the execution time delta almost by two times (see Table III Fast1 vs. DAIF). Fast encoding algorithm Fast2 reduces the complexity even further in three times (156% vs. 54%) with higher coding efficiency penalty (-7.76% vs. -4.06% of bit-rate reduction on average). It should be noted, Fast2 is penalized mostly for QCIF and CIF sequences, whereas its performance for 720p sequences is pretty close to two regular two pass encoding scheme.

TABLE II
CODING EFFICIENCY OF THE PROPOSED ALGORITHMS WITH RESPECT TO H.264/AVC INTERPOLATION FILTER

Resolution	Sequence	Delta Bit-rate, %		
		DAIF[3]	Fast1	Fast2
QCIF	Container	-10.24	-10.02	-3.37
	Foreman	-6.23	-6.34	1.18
CIF	Paris	-0.96	-0.89	1.6
	Foreman	-5.98	-5.81	-1.52
	Mobile	-3.33	-3.32	1.01
	Tempete	-2.19	-2.16	1.05
720p	BigShips	-12.15	-11.95	-10.79
	City	-15.41	-15.26	-13.13
	Crew	-13.32	-12.74	-8.15
	ShuttleStart	-11.4	-10.9	-9.49
	Night	-4.53	-4.35	-3.05
Average		-7.76	-7.61	-4.06

TABLE III
ENCODING COMPLEXITY OF THE PROPOSED ALGORITHMS WITH RESPECT TO H.264/AVC

Resolution	Sequence	Delta Time, %		
		DAIF[3]	Fast1	Fast2
QCIF	Container	121.52	60.33	22.68
	Foreman	174.37	114.96	63.46
CIF	Paris	126.53	64.65	24.17
	Foreman	173.66	110.12	60.02
	Mobile	152.63	86.12	55.69
	Tempete	167.88	103.61	73.85
720p	BigShips	155.26	84.65	64.95
	City	168.05	99.79	76.69
	Crew	156.10	86.71	48.36
	ShuttleStart	167.19	101.67	56.17
	Night	149.24	84.76	44.54
Average		155.68	90.67	53.69

V. CONCLUSIONS

In order to compensate for the temporally changing effect of aliasing and improve the coding efficiency of video coders, adaptive interpolation filtering schemes have been recently proposed. In such schemes, encoder computes interpolation filter coefficients for each frame and re-encodes the frame with the new adaptive filter. However, the coding efficiency comes with the expense of increased encoding complexity due to this additional encoding pass. In this paper we presented two fast encoding algorithms for adaptive interpolation filtering schemes. First method performs quarter-pixel refinement of motion vectors in the second coding pass, instead of performing full motion estimation. And secondly, a single-pass encoding scheme is achieved with adaptive interpolation by reusing the filter coefficients already computed for previous frames. Experimental results show that the proposed methods achieve between 1.5 to 2 times encoding complexity reduction with practically negligible penalty on coding efficiency.

REFERENCES

- [1] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion and Aliasing-Compensated Prediction Using a Two-dimensional Non-Separable Adaptive Wiener Interpolation Filter," in *Proc. ICIP'05*, 2005.
- [2] S. Wittman and T. Wedi, "Separable adaptive interpolation filter", ITU-T SG16/Q6, Doc. C-0219 July, 2007.
- [3] D. Rusanovskyy, K. Ugur, J. Lainema, M. Gabbouj, "Video Coding with Pixel-Aligned Directional Adaptive Interpolation Filters," in *Proc. ISCAS'08*, 2008.
- [4] K. Ugur, D. Rusanovskyy, M. Gabbouj, "Efficient Calculation of Adaptive Interpolation Filter with Distortion Modeling," in *Proc. ICASP'08*, 2008.
- [5] L. Su, Y. Lub, F. Wub, S. Lib, W. Gao, "Real-time Video Coding under Power Constraint Based on H.264 Codec", SPIE-IS&T/Vol.6508.
- [6] J. Ostermann, J. Bormans, P. List, etc., "Video coding with H.264/AVC: Tools, Performance, and Complexity," *IEEE Circuits and System Magazine*, Vol. 4, No. 1, pp. 7-28, 2004.
- [7] Z. Chen, P. Zhu and Y. He, "Fast integer pel and fractional pel motion estimation for JVT," JVT-F017.doc.
- [8] Reference ITU-T VCEG-KTA Software [Online]. Available: <http://iphome.hhi.de/suehring/tml/download/>
- [9] ITU-T Recommendation H.264 & ISO/IEC 14496-10 (MPEG-4) AVC, "Advanced Video Coding for Generic Audiovisual Services", v.3: 2005.
- [10] T. K. Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments," ITU-T Q.6/SG16, VCEG-AE010, Marrakech, Morocco, January 2007

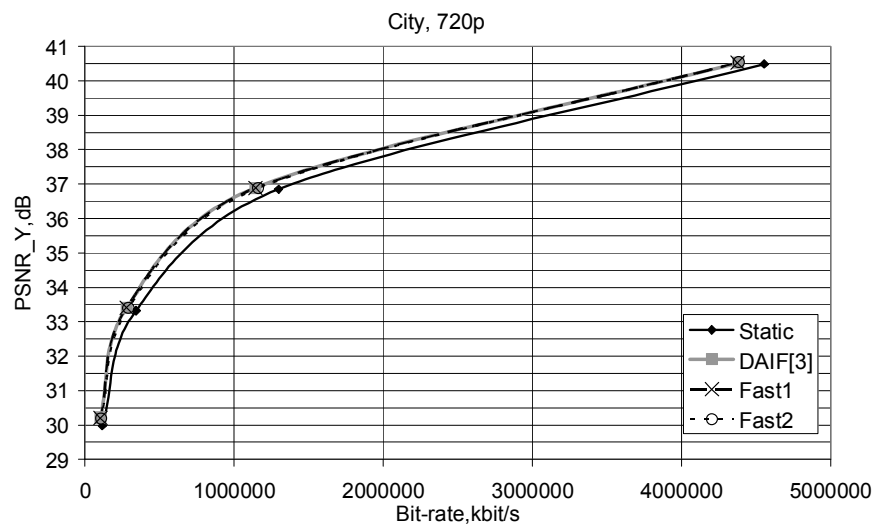


Fig.3. Rate-Distortion Curves for “City”, 720p test sequence

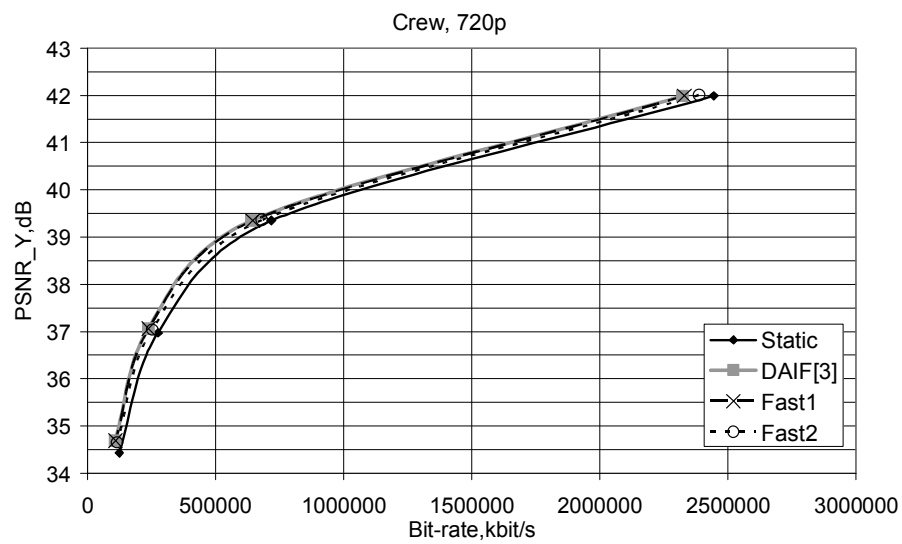


Fig.4. Rate-Distortion Curves for “Crew”, 720p test sequence

- [P4] Ugur, K.; Rusanovskyy, D.; Gabbouj, M., "Efficient calculation of adaptive interpolation filter with distortion modelling," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, vol., no., pp.673-676, March 31 2008-April 4 2008

© 2008 IEEE. Reprinted with permission

EFFICIENT CALCULATION OF ADAPTIVE INTERPOLATION FILTER WITH DISTORTION MODELLING

Kemal Ugur¹, Dmytro Rusanovskyy², Moncef Gabbouj²

¹Nokia Research Center, Tampere, Finland

²Institute of Signal Processing, Tampere University of Technology, Tampere, Finland

ABSTRACT

A novel method is proposed to calculate the coefficients of adaptive interpolation filter used in hybrid video coders for improving the coding efficiency. The proposed algorithm first selects the motion blocks where the majority of prediction errors result from mismatches in motion estimation and from aliasing present in the signal. This is realized by using a second order distortion model to estimate the effect of quantization on motion prediction error and coding results of the previous frames. Then, the filter coefficients are calculated analytically by minimizing the prediction error of those selected blocks. Experimental results show that the proposed method achieves up-to 0.6 dB gain compared to the standard H.264/AVC. Compared to other methods that calculate the filter coefficients using all motion blocks of the frame, the proposed method has significantly less encoding complexity (83% on average) with practically no penalty on coding efficiency.

Index Terms— Adaptive Interpolation, Video Coding

1. INTRODUCTION

The interpolation filter defined in H.264/AVC is designed to minimize the adverse effects of aliasing present in the input image sequence. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in the literature to combat this non-stationary effect of aliasing [1], [2]. In [1], Vatis *et. al.* proposed a 2D non-separable adaptive interpolation filter to reduce the prediction error energy and improve the coding efficiency of video coders. For each fractional pixel position, this scheme utilizes an independent filter and each filter is calculated analytically by solving a system of linear equations that are constructed by minimizing the prediction error energy. This construction step is the main source of complexity due to the need to compute complex cross-correlation and auto-correlation functions for every fractional pixel sample. After the coefficients of the adaptive filter are found, the reference frame is interpolated with this filter and the frame is encoded.

In this work, we propose a novel algorithm to calculate the coefficients of the adaptive filter with significantly less complexity than its counterparts but still improves the

coding efficiency significantly over H.264/AVC. The proposed algorithm first identifies the motion blocks that are not suffering from prediction errors due to aliasing and motion estimation mismatch, marks them as redundant and excludes them in the filter calculation process. This identification process uses a second order distortion model and coding results of previous frames to provide a stable performance over various quantization levels and sequences with different motion characteristics. Using the proposed method, the encoder computes cross-correlation and auto-correlation functions not for every fractional pixel but only for a significantly lower subset of them. As computing the correlation functions is by far the most complex part of the filter calculation process, reduction in its complexity translates in large gains in the overall encoding complexity. Experimental results show that, the proposed method achieves up-to 0.6 dB gain compared to standard H.264/AVC. Compared to other methods that calculate filter coefficients using all the motion blocks of the frame, proposed method has significantly less encoding complexity (83% on average) with practically no penalty on coding efficiency.

This paper is organized as follows; Section 2 describes the 2D adaptive interpolation filter that is used as a basis for our method and presents an analysis of its encoding complexity. Section 3 presents the details of the proposed algorithm. Experimental results are given in Section 4, and Section 5 concludes the paper.

2. NON-SEPARABLE 2D-ADAPTIVE INTERPOLATION FILTER

The adaptive interpolation scheme proposed in [1] is based on 2D non-separable adaptive filters defined independently for each fractional pixel location. For each fractional pixel position, the filter coefficients are calculated analytically by minimizing the prediction error energy. For positions that are horizontally or vertically aligned with an integer pixel, one-dimensional 6-tap filter is used. For other positions, two-dimensional non-separable 6x6 tap filter is utilized.

The filter coefficients are calculated for every frame of the video sequence using the following steps. First, the motion vectors are found by performing motion estimation using the standard non-adaptive filter. In order to find the optimal filter that minimizes the prediction error, a system of linear equations is constructed using the motion vectors

found in the previous step (see [1] and [5] for more details). The 2D non-separable filter for the sub-pixel location SP is given by $h_{m,n}(SP)$, where m,n are the horizontal and vertical coefficient indexes, respectively. The system of linear equations needed to calculate $h_{m,n}(SP)$ is illustrated in (1).

$$\sum_{j=0}^5 \sum_{i=0}^5 R_{m-i,n-j}^r h_{m,n}(SP) = R_{m,n}^{cr}, \quad m,n = 0,\dots,5 \quad (1)$$

where R^{cr} and R^r are the cross-correlation and auto-correlation functions computed over the reference and the predicted image. Equation-1 is then solved to find the coefficients of the adaptive interpolation filter. The reference frame is interpolated with the new filter and the frame is encoded.

2.1 Complexity Analysis

As seen from the above description, an encoder that supports adaptive interpolation filter needs to perform several additional operations, compared to standard non-adaptive interpolation filtering. These additional operations are, *i*) constructing the system of linear equations (1) using cross correlation and auto-correlation functions, *ii*) solving Equation-1 and *iii*) interpolating using the adaptive filter. Solving Equation-1 is not a complex problem itself, while constructing it is a computationally demanding task. This is mainly because of the need to perform many multiplication and addition operations for each sub-pixel that motion vectors over the entire frame. For example, the computation of the correlation function for a given sub-pixel position requires more operations than interpolating the corresponding sub-pixel.

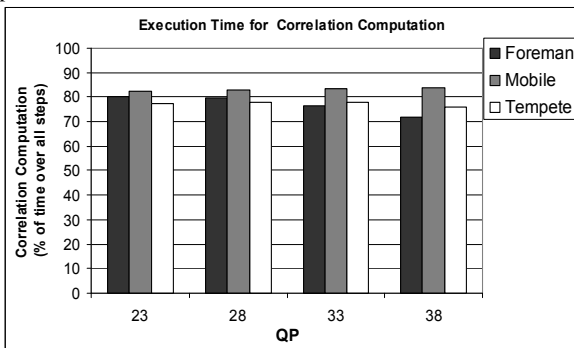


Figure 1 Execution time for computation of correlation functions in percentage of all steps combined

In order to measure how much time it takes for an encoder to construct Equation (1) compared to other operations, we performed several simulations with different sequences and quantization parameters. For every coded frame, we compared the time needed for each of the above steps. The simulation results are illustrated in Figure-2 for three different sequences coded with different quantization parameters. In Figure-2, the time encoder

spends on computing the correlation functions is shown in terms of percentage of the three steps combined. As expected, it was found out that computing the correlation functions takes most of the time spent in the filter calculation process (around 75-85%). Therefore, reducing the complexity of constructing Equation-1 has significant importance on reducing the overall encoding complexity.

3. THE PROPOSED ALGORITHM

As it was shown above, the main complexity of calculating the filter coefficients is due to the need of computing correlation functions for every fractional pixel location over the entire frame. The proposed algorithm aims to reduce this complexity by using only a subset of data, and hence reduce the total number of complex correlation calculations. This is achieved by identifying motion blocks that do not suffer from prediction errors due to aliasing and motion estimation mismatch, marking them as redundant. This identification process is done at the first encoding pass, by comparing the prediction error of each block against a threshold value that is dynamically computed for every frame of the video sequence. If a block is marked redundant, fractional pixels within the block are excluded in the filter calculation process. Equation-2 shows the condition to mark a 4x4 block as redundant, whose coordinates are i,j , frame index is N and quantization step is Q .

$$SAD_{4 \times 4}(i, j) < D_1(Q) + D_2(N) \quad (2)$$

where $SAD_{4 \times 4}(i,j)$ is the motion prediction error of the 4x4 block given in Sum of Absolute Differences (SAD). It should be noted that the block size of 4x4 is chosen to match the smallest motion block partition supported by the H.264/AVC standard. In Equation-2, there are two terms contributing to the distortion threshold. First term depends only on quantization step-size and it is related to removing the effect of quantization in motion prediction signal, so that proposed algorithm provides reliable results at all quantization step sizes. Second term is related to achieve a stable coding efficiency-complexity trade-off over the entire sequence.. The details of how these two terms are calculated are given in the following subsections.

3.1 Error Threshold due to Quantization

In motion compensated prediction, Motion prediction errors may result from three possible sources, *i*) quantization of the reference signal, *ii*) aliasing present in the original signal and *iii*) errors in displacement estimation [2]. Our goal is to find those blocks in which the majority of the prediction mismatches are due to aliasing and displacement estimation errors, and not due to quantization. For this purpose, an empirical model is developed that relates motion prediction error of blocks to quantization step size using statistical analysis, to estimate the prediction error due to quantization and compute the distortion threshold $D_1(Q)$.

The relationship between prediction error and quantization has been well studied in Rate-Control context, where the goal is to estimate the quantization step size given the target bits and the prediction error. One well known Rate-Control algorithm developed for MPEG-4 utilizes a second-order Rate-Distortion model for this purpose [3]. In the context of this paper, a model is needed that relates the prediction error to quantization step size, without requiring the use of bitrate information. Therefore, we adopt a similar second order model as in [3] but modify it accordingly so that the rate term is not included. The model used is given by:

$$D_1(Q) = a_1 \cdot Q^2 + a_2 \cdot Q + a_3 \quad (3)$$

where $D_1(Q)$ is the prediction error due to quantization of the reference signal, Q is the quantization step size and a_1, a_2, a_3 are the model parameters.

The model parameters a_1, a_2, a_3 are calculated empirically using the following statistical analysis. A test sequence is first encoded and the prediction error of every 4x4 block is recorded. According to the resulting prediction errors, a threshold is calculated, which results in $M\%$ of blocks having less prediction error than the threshold value. This threshold is indicated as $SAD_{4x4_Thr}(Q, M)$, and it is calculated for many test sequences with different motion characteristics using a wide-range of quantization step sizes. It should be noted that, the percentage M controls the trade-off between encoding complexity reduction and coding efficiency. The details of how M affects the coding efficiency-complexity tradeoff is shown in the Experimental Results section. After these thresholds are calculated for all the sequences and quantization step-sizes, the model parameters are found by fitting to data, the second order polynomial defined by Equation-3 using least-squares method.

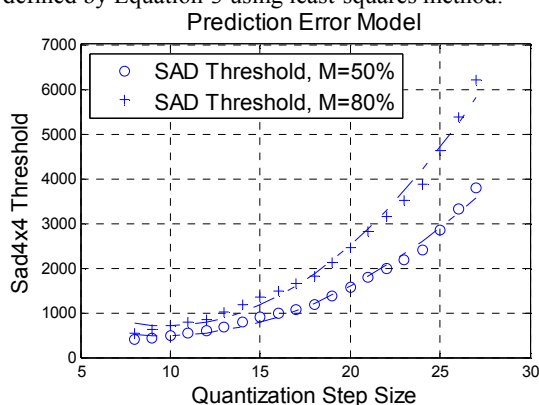


Figure 2. Prediction error due to quantization. Dotted lines indicate the calculated error model.

This process is illustrated using the following example. Let us consider the test sequence Foreman, which is encoded with varying quantization step sizes and the resulting prediction errors of 4x4 blocks are recorded. For each of the quantization step sizes, the threshold values $SAD_{4x4_Thr}(Q, M)$ are found using the above described process. Figure-2 presents the threshold values plotted against the quantization size for two values of M , 50% and

80%. Circles in Figure-2 indicate the threshold values for M equal to 50% and pluses indicates threshold for M being equal to 80%. After these data are computed, the second order polynomial is found and shown with dotted lines in Figure-2.

3.1 Dynamically Modifying the Error Threshold

The distortion threshold used to determine redundant blocks needs to be dynamically computed for every frame of the sequence. This is because the motion characteristics of a sequence can vary dramatically from one segment to another, and can significantly affect the performance of the proposed scheme. Consider the case where a certain segment of the video has relatively low motion, which results in lower than usual prediction error for the blocks. In this case, most or all of the motion blocks are identified as redundant and excluded in the filter calculation process. This results in significant degradation of the coding efficiency for that segment of the video as the calculated coefficients of the filter will not be optimal. Similarly, if some parts of the video undergo a large steady motion, the prediction error for most of the blocks will be large. This would result in including many blocks in the filter calculation process, even though it is not needed, and complexity could not be reduced efficiently.

In Figure-3 this problem is illustrated for two of the frames from the low and high motion segments of the Foreman sequence. The vertical dashed line indicates the threshold value due to quantization ($D_1(Q)$ in Equation-2), which is calculated empirically as described in Section 3.1. The histogram of the prediction errors of blocks for those frames is also plotted. As seen in Figure-3.a, the prediction error of almost all of the blocks is smaller than the threshold $D_1(Q)$ (%), which means many blocks are marked as redundant. This results in coding efficiency penalty, as the resulting filter coefficients are far from optimal for that frame. Similarly, as seen in Figure 3.b, majority of the blocks have larger error than the threshold value, which results in high interpolation complexity for that frame.

This problem is solved in the proposed algorithm by adding the second term in Equation-2 to provide a stable trade-off between coding efficiency and complexity reduction throughout the entire sequence. $D_2(N)$ is the distortion threshold computed for every frame with frame index N , and it is independent from the quantization step size. $D_2(N)$ is computed as follows. After the encoding of a frame $N-1$ is finished, the resulting motion prediction error of each 4x4 motion block is recorded and $D_2(N)$ is calculated so that $M\%$ of the blocks in frame $N-1$ has less prediction error than the value of $D_2(N)$. This way, if a high motion segment starts in a video sequence, the value of $D_2(N)$ increases, which in turn increases the final distortion threshold and the same complexity reduction-coding efficiency trade-off is achieved throughout the sequence.

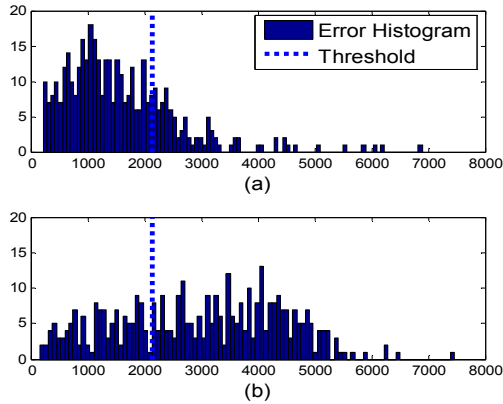


Figure 3. Error Histogram of different frames of Foreman sequence. (a) from low motion segment and (b) from high motion segment.

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of our proposed scheme, we first encoded several sequences at different resolutions using standard H.264/AVC without employing any adaptive interpolation filtering and using the method [1]. Same sequences are finally encoded using the proposed algorithm with two different values of M (50% and 80% respectively) to illustrate how the tradeoff between complexity reduction and coding efficiency could be controlled. The simulations are performed using the Baseline common conditions as detailed in [6]. The encoding complexity and the coding efficiency of the proposed algorithm is compared with [1] in Table 1, where $\Delta Q\%$ indicates the percentage of complexity reduction estimated using the number of fractional pixels used in the filter calculation process, and $\Delta PSNR$ refers to the coding efficiency difference with respect to [1]. As seen in Table-1, the proposed algorithm achieves practically the same coding efficiency with 83% less encoding complexity on average. It can also be seen that for some sequences, a small gain in coding efficiency is realized. This is because adaptive interpolation filter is optimized to certain areas of the image that suffer the most from aliasing and motion prediction errors. This is also inline with the conclusions drawn in [4], where the interpolation filter is optimized locally inside a frame, using several encoding passes. In Figure-4, the performance of the proposed algorithm is compared against H.264/AVC for one of the test sequences, ShuttleStart. It is seen that the proposed algorithm achieves a coding efficiency gain of up-to 0.6 dB at high bitrates compared to H.264/AVC.

5. CONCLUSIONS

In this work, a novel method is proposed to calculate the coefficients of the adaptive interpolation filter used in hybrid video coders for improving the coding efficiency. The proposed algorithm first identifies the motion-blocks

that do not suffer from prediction errors due to aliasing and motion estimation mismatch and excludes them from the filter calculation process. This identification process uses a second order distortion model and coding results of previous frames to provide a stable performance over various quantization levels and sequences with different motion characteristics. It was shown that the proposed method achieves up-to 0.6 dB gain compared to standard H.264/AVC. Compared to other methods that calculate filter coefficients using all the motion blocks of the frame, the proposed method achieves practically the same coding efficiency with significantly less encoding complexity (83% on average).

6. REFERENCES

- [1] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion and Aliasing-Compensated Prediction Using a Two-dimensional Non-Separable Adaptive Wiener Interpolation Filter", *Proc. ICIP 2005*, Genova, Italy, September 2005.
- [2] T. Wedi and H. G. Musmann, "Motion- and Aliasing-Compensated Prediction for Hybrid Video Coding", *IEEE Trans. on CSVT*, vol. 13, No. 7, July 2003.
- [3] Hung-Ju Lee; Tihao Chiang; Ya-Qin Zhang, "Scalable rate control for MPEG-4 video," *Circuits and Systems for Video Technology*, IEEE Transactions on , vol.10, no.6pp.878-894, Sep 2000
- [4] Y. Vatis, J. Ostermann "Locally Adaptive Non-Separable Interpolation Filter for H.264/AVC" *Proc. ICIP 2006*, Atlanta, USA, October 2006
- [5] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1991.
- [6] T.K. Tan, G. Sullivan, T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments", VCEG contribution VCEG-AA10, Nice, Oct. 2005.

Table 1. Simulation Results

	Percentage M			
	50%		80%	
	$\Delta Q, \%$	$\Delta PSNR$	$\Delta Q, \%$	$\Delta PSNR$
Container	53.13	0.02	93.63	0.08
Foreman	31.93	0	76.19	-0.01
ForemanCIF	46.97	0	84.90	-0.05
Mobile	22.71	0	58.87	0.02
Tempete	29.50	0	83.70	-0.05
ShuttleStart	71.14	0	97.28	-0.02
City	24.41	-0.02	86.40	-0.03
Average	39.97	0.00	83.00	-0.01

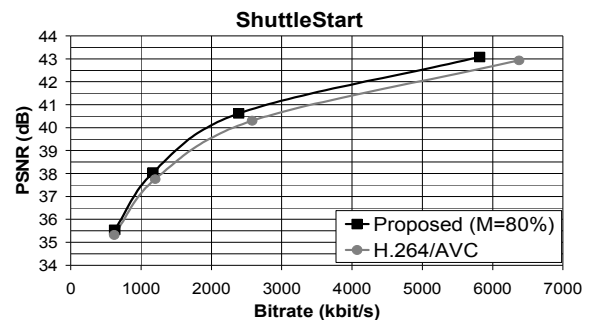


Figure 4. RD performance of proposed algorithm compared with H.264/AVC

- [P5] Rusanovskyy, D.; Ugur, K.; Gabbouj, M.; Lainema, J., "Video coding with pixel-aligned directional adaptive interpolation filters," *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* , vol., no., pp.704-707, 18-21 May 2008

© 2008 IEEE. Reprinted with permission

Video Coding with Pixel-Aligned Directional Adaptive Interpolation Filters

¹⁾Dmytro Rusanovskyy, ²⁾Kemal Ugur,

¹⁾Institute of Signal Processing,
Tampere University of Technology,
Tampere, Finland

¹⁾Moncef Gabbouj, ²⁾Jani Lainema

²⁾Nokia Research Centre,
Nokia Corp.,
Tampere, Finland

Abstract— In this paper a novel adaptive interpolation filter structure is proposed to improve the coding efficiency of video coders. Proposed scheme utilizes one dimensional directional adaptive filter for every of the sub-pixel location, whose coefficients are calculated analytically for every frame by minimizing the prediction error energy. The direction of the interpolation filter is different for every sub-pixel position and it is determined based on the alignment of the corresponding sub-pixel with integer pixel samples. Experimental results show that, the proposed method achieves up-to 1.1 dB gain compared to the standard non-adaptive interpolation scheme of H.264/AVC, requiring less number of operations for interpolation. Compared to two-dimensional non-separable adaptive interpolation, proposed scheme has practically the same coding efficiency with approximately 3 times less complexity. Since significant coding efficiency is achieved without increasing the complexity, it is believed that proposed method has important use-cases in mobile multimedia environments where the resources are severely constrained.

I. INTRODUCTION

The state-of-the art video coding standard, H.264/AVC supports use of motion vectors with quarter pixel accuracy for motion prediction. If the motion vector of a block has fractional pixel accuracy, interpolation needs to be performed to obtain the samples at sub-pixel positions. The half-pixel samples are obtained by using a 6-tap FIR filter and quarter-pixel samples are obtained by bi-linear filtering using the two nearest samples at half or integer pixel positions [1]. The interpolation filter of H.264/AVC was designed to minimize the adverse effects of aliasing present in the input image sequence [2]. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in literature to overcome this non-stationary effect of aliasing and increase the coding efficiency of video codecs [2],[3].

In [3], Vatis *et al.* proposed to use 2D non-separable adaptive interpolation filter (2D-AIF) to reduce the prediction error energy. For each fractional pixel position, this scheme utilizes an independent filter and the coefficients for each filter are

calculated by minimizing the prediction error energy. It was reported that coding gains of up to 1.1 dB is achieved over the standard H.264/AVC especially at high resolution, high quality videos. However, the improvement in the coding efficiency with 2D-AIF scheme comes with the expense of having approximately three times more interpolation complexity compared to the standard H.264/AVC [4]. The reason for this additional complexity is mainly because of the non-separable nature of the filters and the need to perform 32 bit arithmetic. Since interpolation is one of the most complex part of H.264/AVC decoders, a significant increase in its complexity is very problematic, especially for resource constrained devices, such as portable video players or recorders.

In this paper, we propose a novel adaptive interpolation scheme to increase the coding efficiency of video coders, with significantly less complexity than its counterparts. The proposed method uses directional adaptive interpolation filters (DAIF) to compute each sub-pixel sample. The direction of the interpolation filter is different for every sub-pixel position and it is determined based on the alignment of the corresponding sub-pixel with integer pixel samples. Compared to previous schemes that use 2D non-separable adaptive filters, proposed method achieves practically the same coding efficiency with approximately 3 times less complexity. Proposed scheme outperforms the standard H.264/AVC interpolation by up-to 1.1 dB and requires less number of operations for filtering.

This paper is organized as follows; Section 2 describes the proposed filter architecture and its complexity analysis is given in Section 3. Section 4 presents experimental results, and Section 5 concludes the paper.

II. PROPOSED ADAPTIVE FILTER STRUCTURE

Consider Figure 1, where the pixels at integer positions are labeled by upper-case letters {A1,...,F6} within shaded boxes, and lower-case symbols {a,b,...,o} represent sub-pixel positions to be interpolated. Proposed algorithm uses a single directional adaptive filter to interpolate each sub-pixel location.

Let's denote the interpolation filter coefficients used for each sub-pixel Sp , with $h(Sp)$. The direction of these interpolation filters is determined according to the alignment of the corresponding sub-pixel with integer pixel samples. The sub-pixel locations e,o are diagonally aligned with integer pixels in NorthWest-SoutEast direction. The integer samples that are aligned with sub-pixel location in this direction are A1,B2,C3,D4,E5,F6, thus $h(e)$ and $h(o)$ only utilize these integer samples. Similarly, sub-pixel locations g, m are diagonally aligned with integer pixels but in NorthEast-SouthWest direction. Therefore, integer samples A6,B5,C4,D3,E2,F1 are used to interpolate sub-pixel locations g,n. Diagonal pixels-alignment is shown in Fig 1 with red solid arrows. The sub-pixel positions {a,b,c,d,h,l} are either horizontally or vertically aligned with integer image samples. In this case, {C1,C2,...,C6} is utilized for filters $\{h(a), h(b), h(c)\}$ and {A3,B3,...,F3} are used for interpolation of {d,h,l}, shown with green dashed arrows in Fig 1. The sub-pixel positions {j,f,i,k,n} are either aligned with integer-pixel samples in both of the two diagonal directions (position j), or have no integer-pixel samples to be aligned with (positions f,i,k,n). These sub-pixel locations are interpolated using the integer pixels that lie in a diagonal cross structure (A1,B2,C3,D4,E5,F6,F1,E2,D3,C4,B5,A6). The set of integer samples that participate for each sub-pixel position are denoted with $s(Sp) \in \{A1,...,F6\}$ and are given as

$$\begin{aligned} s(a) &= s(b) = s(c) = \{C1, C2, C3, C4, C5, C6\}, \\ s(d) &= s(h) = s(l) = \{A3, B3, C3, D3, E3, F3\}, \\ s(e) &= s(o) = \{A1, B2, C3, D4, E5, F6\}, \\ s(m) &= s(g) = \{F1, E2, D3, C4, B5, A6\}, \\ s(j) &= s(f) = s(i) = s(k) = s(n) = \\ &\{A1, B2, C3, D4, E5, F6, F1, E2, D3, C4, B5, A6\}. \end{aligned}$$

A. Calculating Filter Coefficients

The filter coefficients used to interpolate each sub-pixel are calculated for every P- or B- coded frame analytically by minimizing the motion prediction error. First, initial motion vectors with $\frac{1}{4}$ -pixel accuracy are found by performing motion estimation with the standard H.264/AVC interpolation. Using the motion vectors found in this step, a sub-pixel location Sp that fractional motion vector points to for every motion block is found. For each location Sp an independent filter $h(Sp)$ is estimated utilizing the integer pixels of reference and predicted images that are used to obtain the corresponding sub-pixel. Integer samples used to interpolate each sub-pixel location are given as $s(Sp)$ in previous section. Filter-coefficients estimation is done by solving a system of Wiener-Hopf equations constructed independently for each sub-pixel positions using all motion vectors over the coded frame.

In our description, we utilize relative coordinates for image pixels participating in adaptive interpolation, as it is shown in Fig.1. Considering an sample $X_{x,y}$ located at the (x,y) coordinates of current video frame X , its local neighborhood size of 6x6 pixels is shown as $\{A1,...,F6\}$, where $X_{x,y}$ is located at the C3 position. The motion vector

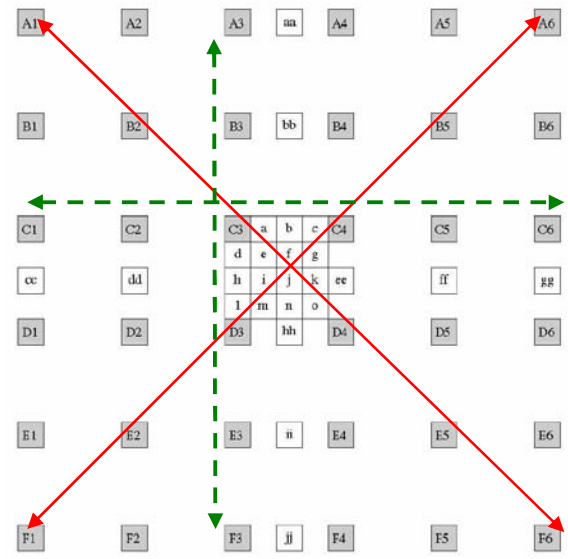


Figure 1. Utilized image samples notation and interpolations filters alignment.

$mv_{x,y} = (dx, dy)$ that is associated with the predicted sample $Y_{x,y}$ points to the location $(x+dx, y+dy)$ in the reference frame Y . Integer part of this motion vector is computed by rounding (dx, dy) towards the negative infinity and it specifies the relative base C3 and its local neighborhood $\{A1,...,F6\}$ in reference frame. Thus, a relative coordinate A1 in current frame is corresponding to a sample $X_{x-2,y-2}$ while A1 in the reference frame corresponds to a sample $Y_{x+dx-2,y+dy-2}$. The fractional part of motion vector specifies sub-pixel position $Sp = \{a, b, ..., o\}$ and thus determines the corresponding adaptive filter $h(Sp)$ computed from these image samples.

Adaptive filter-coefficients for sub-pixel location Sp is given with $h_t(Sp)$, where t is the coefficients index, are analytically computed by solving the system of linear equation [5]. A system for N-tap Wiener filter, where $N=6$ for $\{a,b,c,d,h,l,e,g,m,o\}$ and $N=12$ for $\{f,i,j,k,n\}$ is given in (1):

$$\sum_{t=0}^{N-1} R_{t-i,i}^r(Sp) \cdot h_t(Sp) = R_i^{cr}(Sp), \quad i, t = 0, ..., N-1 \quad (1)$$

where $R^{cr}(Sp)$ and $R^r(Sp)$ are expected cross-covariance and auto-covariance functions computed independently for each Sp as shown in (2) and (3) from current and reference images.

$$R_i^{cr}(Sp) = E[Y(s_i(Sp)) \cdot X(C3)] \quad , \quad (2)$$

$$R_{i,i}^r(Sp) = E[Y(s_i(Sp)) \cdot Y(s_i(Sp))] \quad , \quad (3)$$

where $s(Sp) \in \{A1,...,F6\}$ is the sub-pixel specific set of relative coordinates of integer pixel samples that are used for interpolation. A practical approach to compute expectations (2) and (3) is to average local covariances over all motion vectors which were estimated for currently coded frame.

B. Filter-coefficients symmetry

The filter structure introduced above defines a set of 15 independent filters $\{h(Sp), Sp=\{a,b,...,o\}\}$ for each coded frame and is fully described with 120 filter coefficients that should be transmitted to the decoder side. However, transmitting such amount of data at each frame brings a significant overhead and may degrade the coding efficiency. In order to reduce this overhead, we assume the statistical properties of image signal are symmetric [3]. This way, same filter-coefficients are used in case the distance of the corresponding full-pixel positions to the current sub-pixel position are equal. These assumption leads to the following restrictions on the filter coefficients.

Sub-pixel locations $\{a,c,d,l\}$ share the same filter, but horizontally or vertically mirrored, which is defined with 6 coefficients.

$$h_t(a) = h_{-t}(c) = h_t(d) = h_{-t}(l) .$$

Sub-pixel locations $\{b,h\}$ are interpolated with symmetrical 6-tap filter with 3 independent filter coefficients:

$$h_t(b) = h_t(h) .$$

Diagonally aligned sub-pixel locations $\{e,g,m,o\}$ are described with one filter using 6 filter coefficients, but diagonally mirrored:

$$h_t(e) = h_{-t}(o) = h_t(g) = h_{-t}(m) .$$

Sub-pixel location j has the central position, thus it is interpolated with a symmetric filter with only 3 independent filter coefficients:

$$\begin{aligned} h_0(j) &= h_5(j) = h_6(j) = h_{11}(j) , \\ h_1(j) &= h_4(j) = h_7(j) = h_{10}(j) , \\ h_2(j) &= h_3(j) = h_8(j) = h_9(j) . \end{aligned}$$

A single filter with 6 independent filter coefficients is utilized for $\{f,i,k,n\}$ positions. Filters retaliation is following:

$$\begin{aligned} h_0(f) &= h_6(f) = h_5(n) = h_{11}(n) = \\ h_0(i) &= h_{11}(i) = h_5(k) = h_6(k) , \\ h_1(f) &= h_7(f) = h_{10}(n) = h_4(n) = \\ h_1(i) &= h_{10}(i) = h_7(k) = h_4(k) , \\ &\dots \\ h_{11}(f) &= h_5(f) = h_0(n) = h_6(n) = \\ h_6(i) &= h_5(i) = h_0(k) = h_{11}(k) . \end{aligned}$$

Following these derivations, total amount of filter-coefficients to be transmitted to the decoder is reduced to 24.

C. Interpolation process

After adaptive filter-coefficients are calculated, we interpolate reference image samples Y at sub-pixels locations as following:

$$Y(Sp) = \sum_{t=0}^N Y(s_t(Sp)) \cdot h_t(Sp) \quad (4)$$

TABLE I. NUMBER OF ARITHMETIC OPERATIONS REQUIRED TO INTERPOLATE SUB-PIXEL POSITIONS.

Subpel position	H.264/AVC[1]	2D-AIF[3]	DAIF
b,h	10	10	10
a,c,d,l	13	13	13
e,g,m,o	23	58	13
J	32.5	41	16
f,i,k,n	35.5	55	19
Total for all 1/4-pixels	338.5	565	216

Interpolation for $\{f,i,k,n,b,h,j\}$ locations is modified due to a symmetry assumption introduced above, following are exemplary filtering implementation for $\{b,f,j\}$ sub-pixels:

$$X(b) = (C1 + C6) \cdot h_0(b) + (C2 + C5) \cdot h_1(b) + (C3 + C4) \cdot h_3(b)$$

$$X(f) = (A1 + A6) \cdot h_0(f) + (B2 + B5) \cdot h_1(f) + (C3 + C4) \cdot h_2(f) + (D3 + D4) \cdot h_3(f) + (E2 + E5) \cdot h_4(f) + (F1 + F6) \cdot h_5(f)$$

$$\begin{aligned} X(j) &= (A1 + A6 + F1 + F6) \cdot h_0(j) + \\ &(B2 + B5 + E2 + E5) \cdot h_1(j) + \\ &(C3 + C4 + D3 + D4) \cdot h_2(j) \end{aligned}$$

III. COMPLEXITY ANALYSIS

We estimated the complexity of the proposed in this paper interpolation (DAIF) assuming the image symmetry and compared it with complexities of 2D-AIF and H.264/AVC interpolations reported in [4]. Our complexity estimation was done as a number of basic arithmetic operations, such sum, multiplication and shift required to interpolate each sub-pixel location, similarly to the method described in [4]. These complexity estimates for each filter are given in the Table I. For every sub-pixel we show the number of operations for H.264/AVC, 2D-AIF and proposed interpolations. We realize that number of arithmetic operations is a simple approximation of the complexity and do not consider computational architecture, e.g. memory bandwidth, use of dual MAC units and so on. However, we believe these estimates are useful for illustration purposes.

Analysis of the Table I shows that proposed DAIF scheme requires about 38% of that number of operations needed for 2D-AIF to interpolate reference image. In comparison to the standard H.264/AVC interpolation, our method requires about 36% less arithmetic operations.

The worst-case complexity is an important parameter that determines the upper-bound of decoding performance. As seen in Table-I, the worst case complexity of the proposed scheme is reduced by 41% compared to H.264/AVC and reduced to one-third of the 2D-AIF scheme.

IV. EXPERIMENTAL RESULTS

Proposed in this paper DAIF scheme was integrated into the official VCEG-KTA software [6]. In order to test performance of the proposed scheme, we simulated rate-distortion curves with VCEG-KTA software following the common test conditions defined in the ITU-T/VCEG group [7] with the baseline profile settings. Encoded bit-streams included side information for adaptive filters such as quantized filter coefficients, thus generated streams were fully decodable.

Simulation were performed with test video sequences listed in [7] with total amount of 300 coded frames for sequences at QCIF and CIF resolutions, and 150 coded frames for video materials at 720p resolution. In addition, we included 4CIF video with total number of 150 coded frames for each sequence. Three interpolation schemes have been compared; those are proposed in this paper (DAIF), the scheme with 2D non-separable adaptive filters (2D-AIF) [3], and H.264/AVC [1], which was utilized for benchmarking. Table 2 presents the simulation results for 2D-AIF and DAIF interpolation schemes in terms of difference in average PSNR (Δ PSNR [7]) compared to the standard H.264/AVC method. In addition to these, exemplary rate-distortion curves are given in Fig. 2 and 3 for “ShuttleStart” and “Crew” test sequences respectively.

As seen from Table II, proposed scheme significantly outperforms in terms of coding efficiency the standard H.264/AVC. The achieved coding gain is up to 1.1dB for high resolution video (see Fig.2 and 3) and 0.6 dB on average for 720p sequences. In comparison to the method with 2D non-separable adaptive interpolation, proposed scheme achieves comparable results, with significantly lower complexity. Compared to 2D non non-separable filtering scheme, proposed scheme algorithm has practically the same coding efficiency (0.02 dB less in average), with significant benefits in complexity.

V. CONCLUSIONS

The recent H.264/AVC video coding standard supports motion compensated prediction with up to quarter-pixel accuracy motion vectors. In this paper we consider a quarter-pixel accuracy motion compensated prediction and propose a novel interpolation scheme using directional adaptive filter for each sub-pixel location. The direction of the interpolation filter is different for every sub-pixel position and it is determined based on the alignment of the corresponding sub-pixel with integer pixel samples. Experimental results show that, the proposed method achieves up-to 1.1 dB gain compared to the standard non-adaptive interpolation scheme of H.264/AVC, requiring less number of operations for interpolation. Compared to two-dimensional non-separable adaptive interpolation, proposed scheme has practically the same coding efficiency with approximately 3 times less complexity.

REFERENCES

- [1] M. Karczewicz, A. Hallapuro, “Interpolation solution with low encoder memory requirements and low decoder complexity”, ITU-T SGI, VCEG-N31, 24-27 Sept. 2001.
- [2] T. Wedi, “Adaptive Interpolation Filter for H.26L”, ITU-T SG16/Q6, doc. VCEG-N28, Santa Barbara, CA, USA, Sep. 2001.
- [3] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, “Motion and Aliasing-Compensated Prediction Using a Two-dimensional Non-Separable Adaptive Wiener Interpolation Filter”, Proc. ICIP 2005, Genova, Italy, September 2005.
- [4] Y.Vatis and J. Ostermann, “Comparison of complexity between two-dimensional non-separable adaptive interpolation filter and standard wiener filter”, ITU-T SGI 6/Q.6 Doc. VCEG-AA11, Nice, France, October 2005.
- [5] Papoulis, Probability, Random Variables, and Stochastic Processes. New York: McGraw-Hill, 1991.

[6] Reference ITU-T VCEG-KTA Software [Online]. Available: <http://iphome.hhi.de/suehring/tml/download/>

[7] T. K Tan, G. Sullivan and T. Wedi, “Recommended Simulation Common Conditions for Coding Efficiency Experiments,” ITU-T Q.6/SG16, VCEG-AE010, Marrakech, Morocco, January 2000

TABLE II. SIMULATION RESULTS OF THE 2D-AIF AND PROPOSED INTERPOLATION COMPARING TO THE H.264/AVC INTERPOLATION.

Resolution	Sequence	Delta PSNR,dB	
		2D-AIF[3]	DAIF
QCIF	Container	0.33	0.38
	Foreman	0.1	0.11
	Silent	-0.02	0
CIF	Paris	0.05	0.06
	Foreman	0.3	0.27
	Mobile	0.3	0.12
	Tempete	0.16	0.07
4CIF	City	0.45	0.42
	Soccer	0.55	0.49
	Ice	0.1	0.1
	Crew	0.4	0.36
720p	City	0.61	0.57
	BigShips	0.36	0.35
	ShuttleStart	0.75	0.75
	Crew	0.59	0.56

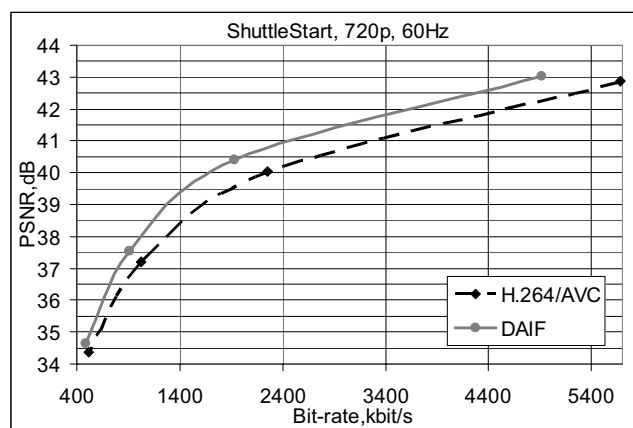


Figure 2. Rate-distortion curves for “ShuttleStart” test sequence.

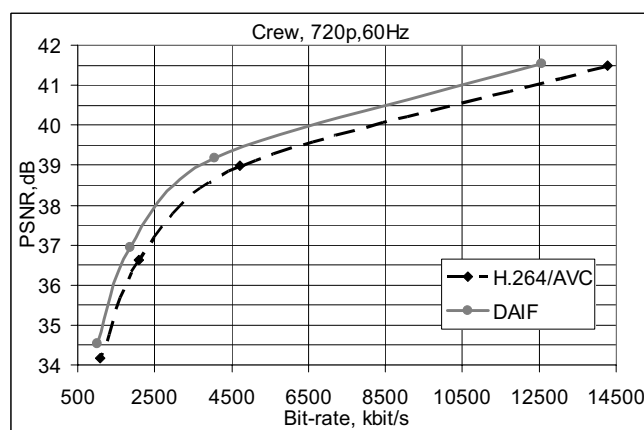


Figure 3. Rate-distortion curves for “Crew” test sequence.

- [P6] Rusanovskyy, D.; Ugur, K.; Hallapuro, A.; Lainema, J.; Gabbouj, M., "Video Coding With Low-Complexity Directional Adaptive Interpolation Filters," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.19, no.8, pp.1239-1243, Aug. 2009

© 2009 IEEE. Reprinted with permission

Video Coding With Low-Complexity Directional Adaptive Interpolation Filters

Dmytro Rusanovskyy, Kemal Ugur, Antti Hallapuro, Jani Lainema, and Moncef Gabbouj

Abstract—A novel adaptive interpolation filter structure for video coding with motion-compensated prediction is presented in this letter. The proposed scheme uses an independent directional adaptive interpolation filter for each sub-pixel location. The Wiener interpolation filter coefficients are computed analytically for each inter-coded frame at the encoder side and transmitted to the decoder. Experimental results show that the proposed method achieves up to 1.1 dB coding gain and a 15% average bit-rate reduction for high-resolution video materials compared to the standard nonadaptive interpolation scheme of H.264/AVC, while requiring 36% fewer arithmetic operations for interpolation. The proposed interpolation can be implemented in exactly 16-bit arithmetic, thus it can have important use-cases in mobile multimedia environments where the computational resources are severely constrained.

Index Terms—Adaptive filters, interpolation, motion-compensation, video coding.

I. INTRODUCTION

STANDARDIZED hybrid video coding systems such as the recent H.264/AVC coding scheme [1] as well as older standards, e.g., MPEG-1, 2, 4, H.262, and H.263, are all based on motion-compensated prediction. The state-of-the-art video coding standard H.264/AVC supports the use of motion vectors with quarter pixel accuracy for motion-compensated prediction. The fractional-pixel samples are obtained using a six-tap Wiener filter that is designed to minimize the adverse affects of aliasing [2]. However, aliasing in a video sequence is not a stationary process but has varying characteristics. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in the literature to overcome this nonstationary effect of aliasing and thus increase the coding efficiency of video codecs [3], [4].

Fig. 1 shows a generalized block diagram of a hybrid video encoder with adaptive interpolation filters. The upper blocks of the diagram enclosed within a solid thick

line rectangle represent a conventional encoder structure with fractional-pixel motion-compensated prediction, e.g., H.264/AVC, while those enclosed within a dash-dotted rectangle depict the estimation process of the interpolation filter.

In hybrid video coding scheme, the current image X is to be predicted by motion-compensated prediction (MCP) from the already encoded reference image Y . The result of the motion-compensated prediction is the predicted image X' . The prediction error E' is undergoing the transform coding (T) and quantization (Q) following the entropy encoding (ENC). In addition to encoding the prediction error E' , the motion information mv that is obtained by the motion estimation (ME) is coded and transmitted to the decoder side.

Video coding with adaptive interpolation assumes that coefficients of the interpolation filter h are computed for each inter-predicted frame, and this process is shown in Fig. 1 within the dash-dotted block. The adaptive filter h is the optimal Wiener filter minimizing the error energy of the motion-compensated prediction.

In [4], Vatis *et al.* proposed a 2-D nonseparable adaptive interpolation filter (2-D-AIF). For each sub-pixel position, this scheme utilizes an independent 2-D nonseparable filter size of 6×6 tap. A total of 36 filter coefficients for each filter are calculated by minimizing the prediction error energy of each coded frame. It was reported that coding gains of up to 1.1 dB are achieved over the standard H.264/AVC especially at high-resolution high-quality videos. However, the improvement in the coding efficiency with 2-D-AIF scheme comes at the expense of having approximately three times higher interpolation complexity compared to the standard H.264/AVC [5]. The reasons for such increase are the nonseparable nature of 36-tap filters and the need to perform operations in 32-bit arithmetic. Since interpolation is already one of the most complex parts of H.264/AVC decoders [6], a further increase in the complexity is very problematic, especially for resource-constrained devices, such as portable video players or recorders.

In this letter, we propose a novel adaptive Wiener interpolation scheme to improve the coding efficiency of video coders, with significantly less complexity than its counterparts [4]. The proposed method uses directional filters to interpolate each sub-pixel sample. The direction of the interpolation filter for each sub-pixel position is determined based on the alignment of the corresponding sub-pixel with integer-pixel samples. The proposed filter structure allows the use of 16-bit integer operations at the decoder, thus providing a very low computational

Manuscript received January 16, 2008; revised October 21, 2008; accepted December 5, 2008. First version published May 12, 2009; current version published August 14, 2009. This work was supported by the Nokia Research Center and by the Academy of Finland, Application Number 213462, Finnish Programme for Centers of Excellence in Research 2006–2011. This paper was recommended by Associate Editor S. Li.

D. Rusanovskyy and M. Gabbouj are with the Department of Signal Processing at Tampere University of Technology, 33720 Tampere, Finland (e-mail: dmytro.rusanovskyy@tut.fi; moncef.gabbouj@tut.fi).

K. Ugur, A. Hallapuro, and J. Lainema are with the Nokia Research Center, Nokia Corporation, 33720 Tampere, Finland (e-mail: ugur.kemal@nokia.com; antti.hallapuro@nokia.com; jani.lainema@nokia.com).

Digital Object Identifier 10.1109/TCSVT.2009.2022708

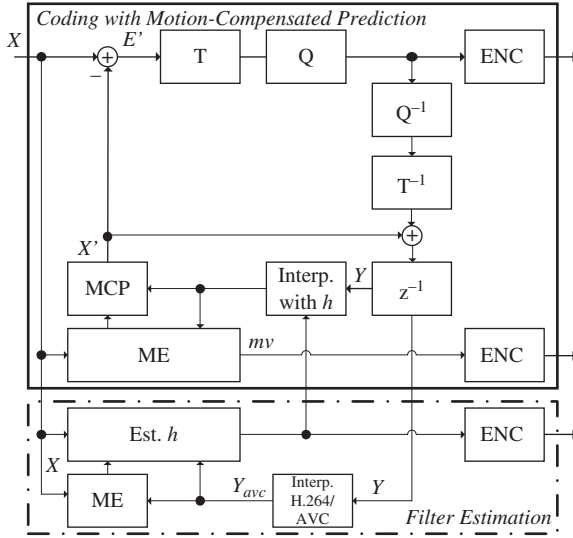


Fig. 1. Block diagram of a hybrid video encoder based on motion-compensated prediction with adaptive interpolation filter.

complexity of interpolation at the decoder. Since significant coding efficiency is achieved with low decoding complexity, it is believed that the proposed method has important use-cases in mobile multimedia environments where the computational resources are severely constrained.

This letter is organized as follows. The proposed filter architecture is given in Section II. In Section III we discuss the interpolation process, describe its integer implementation, and provide its complexity estimates. Section IV presents simulation results, while Section V concludes the letter.

II. PROPOSED FILTER STRUCTURE

Consider Fig. 2, where the pixel locations in the original image grid (we call them integer-pixels locations) are labeled by upper case letters $\{A1, \dots, F6\}$ within the shaded boxes, and lower case symbols $\{a, b, \dots, o\}$ represent locations of samples at sub-pixel image grid (sub-pixel locations) that are to be interpolated. The proposed algorithm uses a single directional adaptive interpolation filter (DAIF) to generate each sub-pixel.

Let us denote an interpolation filter used for each sub-pixel Sp , $Sp \in \{a, b, \dots, o\}$, with $h(Sp)$. The direction of these filters is determined according to the alignment of the corresponding sub-pixel Sp with the integer-pixels. Such sub-pixel-specific filter alignment leads to different sets of integer-pixels, which are used to estimate the filter $h(Sp)$ and interpolate Sp -sample. We denote these sets as $s(Sp) \subseteq \{A1, \dots, F6\}$. The following describes the alignment of the filters' directions and therefore $s(Sp)$ in detail.

A. Alignment of Filters' Directions

Sub-pixel locations $\{a, b, c\}$ and $\{d, h, l\}$ are horizontally and vertically aligned with samples at integer locations $s(a|b|c) = \{C1, C2, \dots, C6\}$ and $s(d|h|l) = \{A3, B3, \dots, F3\}$, respectively, as it shown in the Fig. 2(a). Therefore, only those image samples at the respective $s(Sp)$ locations are utilized for filter $h(Sp)$ estimation and interpolating of $\{a, b, c\}$ and $\{d, h, l\}$.

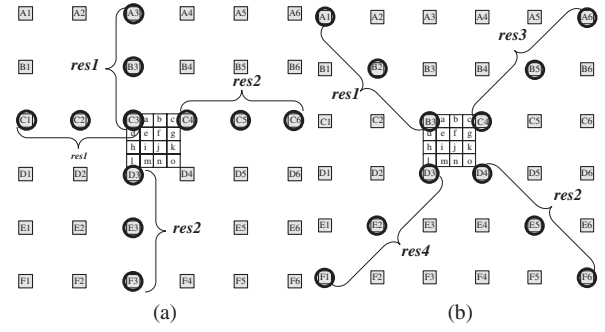


Fig. 2. Interpolation process; (a) two partial sums ($res1$ and $res2$) required for each of $\{a, b, c, d, h, k\}$ and (b) four partial sums for $\{f, i, j, k, n\}$.

Sub-pixel locations e, o are diagonally aligned with integer-pixels in the northwest–southeast direction, whereas for locations g, m such alignment is in northeast–southwest direction, see Fig. 2(b). Therefore, image samples at coordinates $s(e|o) = \{A1, B2, C3, D4, E5, F6\}$ and $s(g|m) = \{A6, B5, C4, D3, E2, F1\}$ are used for interpolating sub-pixel locations $\{e, o\}$ and $\{g, m\}$, respectively. Locations $\{f, i, j, k, n\}$ are either aligned with integer-pixel samples in both diagonal directions (location j), or have no integer-pixel samples to be aligned with (positions f, i, k, n). For this reason, $\{f, i, j, k, n\}$ are interpolated using the integer-pixels that lie on a diagonal cross structure $\{A1, B2, \dots, F6\} \cup \{F1, E2, \dots, A6\}$. The sets of integer-pixel locations that contribute to the interpolation at each sub-pixel location are denoted with $s(Sp)$ and are given explicitly as following:

$$\begin{aligned} s(a) &= s(b) = s(c) = \{C1, C2, C3, C4, C5, C6\} \\ s(d) &= s(h) = s(l) = \{A3, B3, C3, D3, E3, F3\} \\ s(e) &= s(o) = \{A1, B2, C3, D4, E5, F6\} \\ s(m) &= s(g) = \{F1, E2, D3, C4, B5, A6\} \\ s(j) &= s(f) = s(i) = s(k) = s(n) = \\ &\quad \{A1, B2, C3, D4, E5, F6, F1, E2, D3, C4, B5, A6\}. \end{aligned} \quad (1)$$

B. Calculating the Filter Coefficients

The interpolation filter estimation diagram is depicted in Fig. 1 within a dash-dotted block. The filter coefficients are analytically calculated for every P- or B-coded frame X . First, initial motion vectors mv with 1/4-pixel accuracy are found by performing motion estimation with the standard H.264/AVC interpolation (Y_{avc}). Using this motion information, filters $h(Sp)$ are calculated by solving a system of Wiener–Hopf equations [7], which are constructed from Y_{avc} and X images independently for each Sp . Coordinates of image samples which are used for $h(Sp)$ estimations are given in (1) as $s(Sp)$ in terms of relative (local coordinates) $\{A1, \dots, F6\}$. The link between these relative coordinates and the absolute image coordinates (x, y) can be established as follows.

Consider a sample $X_{x,y}$ located at the (x, y) coordinate in the original video frame X . Its local neighborhood with a size of 6×6 pixels is shown in Fig. 2 as $\{A1, \dots, F6\}_{x,y}$, whereas $X_{x,y}$ is associated with $C3$ position. The sample $X_{x,y}$ is predicted from the reference image Y_{avc} with motion vector $mv_{x,y} = (dx, dy)$. The integer part of this motion vector $\lfloor mv_{x,y} \rfloor = (\lfloor dx \rfloor, \lfloor dy \rfloor)$ specifies the absolute coordinate

of C3 position in the up-sampled reference frame Y_{avc} as $(4x + 4 \cdot \lfloor dx \rfloor, 4y + 4 \cdot \lfloor dy \rfloor)$. The absolute coordinates of $\{A1, \dots, F6\}_{x,y}$ positions in Y_{avc} can be derived accordingly. The fractional component of the current $mv_{x,y}$ determines a sub-pixel location $Sp \in \{a, b, \dots, o\}$ and hence the interpolation filter $h(Sp)$ which is to be computed from $\{A1, \dots, F6\}_{x,y}$ samples of the reference and the original images.

Filter coefficients $h_t(Sp)$, where t is the coefficient's index, are calculated by solving a system of Wiener-Hopf equations. For an N -tap filter, the system of equations is given in (2)

$$\sum_{t=0}^{N-1} R_{t-i,i}^r(Sp) \cdot h_t(Sp) = R_i^{cr}(Sp), \quad i, t = 0, \dots, N-1 \quad (2)$$

where $R^{cr}(Sp)$ and $R^r(Sp)$ are the expected cross-covariance and auto-covariance functions computed from the original and the reference images independently for each Sp

$$R_i^{cr}(Sp) = E[Y(s_t(Sp)) \cdot X(C3)] \quad (3)$$

$$R_{t,i}^r(Sp) = E[Y(s_t(Sp)) \cdot Y(s_i(Sp))] \quad (4)$$

where $s(Sp)$ is a set of integer-pixels locations given in (1). A practical approach to compute expectations (3) and (4) is to average local covariances over all motion vectors which were estimated from Y_{avc} .

III. INTERPOLATION PROCESS

After the adaptive filter coefficients are calculated, the reference image samples Y are interpolated at sub-pixels locations Sp as follows:

$$Y(Sp) = \sum_{t=0}^{N-1} Y(s_t(Sp)) \cdot h_t(Sp). \quad (5)$$

Interpolation at sub-pixel positions $\{a, c, d, l, e, g, m, o\}$ is accomplished with a 6-tap filter ($N = 6$) and $\{f, i, j, k, n\}$ samples are interpolated with $N = 12$, as shown in (5). And, the filter calculation process described in Section II places no restrictions on floating point filter coefficients h_t .

However, in order to perform interpolation with integer arithmetic, h_t should be represented by integer values with finite word length. A signed h_t value is represented with $Q+1$ bits as follows:

$$h_t^Q = \text{sign}(h_t) \cdot \lfloor |h_t| \cdot 2^Q + 0.5 \rfloor \quad (6)$$

where Q is the number of bits representing the magnitude and the leftmost bit is reserved for the sign. Using this representation, interpolation is performed using integer arithmetic as follows:

$$Y(Sp) = \left(\sum_{t=0}^{N-1} Y(s_t(Sp)) \cdot h_t^Q + 2^{Q-1} \right) \gg Q. \quad (7)$$

Larger Q improves the accuracy of filter coefficient representation but increases the required memory size and memory-access bandwidth. In order to have the convolution sum in (7) fit in a 16-bit memory storage, the filter coefficients h_t should be represented with a maximum of 4-bit accuracy ($Q = 4$). It was found empirically, that having $Q < 8$ significantly reduces the coding efficiency.

A. 16-Bit Integer Arithmetic Implementation

In order to achieve guaranteed 16-bit operation for the interpolation operation and still keep the coding efficiency high, a filter-specific calculation structure is proposed. The proposed method exploits the typical characteristics of interpolation filters and places certain restrictions on the filter coefficient values.

The proposed structure performs interpolation not using the equation given in (7), but using partial sums as shown in Fig. 3. Two partial sums are required to interpolate $Y(Sp)$ for $Sp \in \{a, b, c, d, h, l, e, g, m, o\}$ and they are given as

$$\begin{aligned} res1 &= \left(\sum_{t=0}^2 Y(s_t(Sp)) \cdot h_t^Q(Sp) \right), \\ res2 &= \left(\sum_{t=3}^5 Y(s_t(Sp)) \cdot h_t^Q(Sp) \right). \end{aligned}$$

Two additional partial sums are required to interpolate $Y(Sp)$ for $Sp \in \{f, i, k, n, j\}$ and these sums are given as

$$\begin{aligned} res3 &= \left(\sum_{t=6}^8 Y(s_t(Sp)) \cdot h_t^Q(Sp) \right), \\ res4 &= \left(\sum_{t=9}^{11} Y(s_t(Sp)) \cdot h_t^Q(Sp) \right). \end{aligned}$$

For typical filter coefficients and image samples, positive values of partial sums prevail in observed statistics. So we can gain additional accuracy for filter coefficients by clipping the partial sums below zero, e.g., $res1 = \max(0, res1)$. Since all intermediate results $res1$, $res2$, $res3$, and $res4$ are positive, their representation format is converted from a signed 16-bit integer to an unsigned 16-bit integer. This allows one additional bit accuracy for defining the magnitude of the coefficients.

In order to avoid overflow and underflow and keep the partial sums fit in the signed 16-bit registers, we define two requirements on filter coefficient values

$$\begin{aligned} &\left(\sum_{t=j}^{j+2} \max(0, h_t^Q(Sp)) < 128 \right) \text{ and} \\ &\left(\sum_{t=j}^{j+2} \min(0, h_t^Q(Sp)) > -128 \right) \end{aligned}$$

where $j = 0, 3$ and $j = 0, 3, 6, 9$ for $\{a, c, d, l, e, g, m, o\}$ and $\{f, i, k, n, j\}$ filters, respectively. With these restrictions in place, Q in (6) could be set as $Q = 7$ and the coefficients take values of $-127/128, -126/128, \dots, 126/128, 127/128$, with accuracy $1/128$. The accuracy of filter coefficients is increased from 4 bits to 7 and still guaranteeing 16-bit integer implementation.

Additional bit accuracy for filter coefficients of sub-pixel locations $\{f, i, k, n, j\}$ is gained by observing the fact that the statistical distribution of those coefficients most often fall in the range $[-0.5, 0.5]$, instead of $[-1, 1]$. We exploit this fact and restrict the filter coefficients of sub-pixel locations $\{f, i, k, n, j\}$ to the range $[-0.5, 0.5]$. More specifically,

the floating point filter coefficients of sub-pixel locations $\{f, i, k, n, j\}$ is converted to integer representation as shown in (6), with $Q = 8$, instead of 7.

Using the filtering operation and the restrictions described above, the accuracy of the filter-coefficients is increased from 4 bits to 7 for $\{a, c, d, l, e, g, m, o\}$ and to 8 for $\{f, i, k, n, j\}$ sub-pixel locations. The interpolation operations final expressions (8) and (9) are given for 16-bits interpolation of $\{a, c, d, l, e, g, m, o\}$ and $\{f, i, k, n, j\}$ respectively

$$Y(Sp) = (res1 + res2 + 2^{Q-1}) \gg Q, \quad (8)$$

for $Q = 7$ and $Sp \in \{a, b, c, d, h, l, e, g, m, o\}$

$$Y(Sp) = ((res1 + res2) \gg 1 + (res3 + res4) \gg 1 + 2^{Q-2}) \gg Q - 1$$

$Q = 8$ and $Sp \in \{f, i, j, k, n\}$. (9)

In rare cases where the coefficient calculation process (2) results in coefficients that do not fulfill the conditions given in this section, the static interpolation filter is used for the corresponding sub-pixel location.

B. Interpolation Complexity Analysis

In this section we provide complexity analysis of the interpolation process that should be implemented at the decoder side. The complexity of the filter estimation process is beyond the scope of this letter, as it is an encoder algorithm and has much less restriction on the complexities.

First, we estimate the interpolation complexity of the proposed DAIF scheme as given in (7) and compare it with the complexities of the standard H.264/AVC and the most recently proposed 2-D-AIF interpolations, which are reported in [5]. This estimation was done in terms of the number of basic arithmetic operations, such as summation, multiplication, and shift, required to interpolate each sub-pixel location. The complexity estimate for each sub-pixel as well as the average of all sub-pixels complexities are given in Table I. Analysis of Table I shows that DAIF scheme requires about 38% of the number of operations needed for 2-D-AIF to interpolate the reference image (all sub-pixel locations). Moreover, our method requires about 36% less arithmetic operations in comparison to the standard H.264/AVC interpolation. As seen in Table I, the worst case complexity (which is an important parameter for decoding complexity) of the proposed scheme is reduced by 41% compared to H.264/AVC and reduced to one-third of the 2-D-AIF scheme.

Second, we estimate the complexity of the proposed interpolation implemented with 16-bit arithmetic, as given in (8) and (9), and compare it with complexity estimates for the 16-bit implementation of the H.264/AVC interpolation, available in [5]. Estimation results are given in Table II. Analysis of Table II shows that the proposed 16-bit implementation of DAIF interpolation has a complexity that is comparable with the highly optimized H.264/AVC on average for all sub-pixels as well as in the worst case scenario. We realize that the number of arithmetic operations is a simple approximation of the complexity and does not

TABLE I
NUMBER OF ARITHMETIC OPERATIONS REQUIRED TO INTERPOLATE
SUB-PIXEL POSITIONS

Sub-pixel position	H.264/AVC [1]	2-D-AIF [4]	DAIF
b, h	10	10	10
a, c, d, l	13	13	13
e, g, m, o	23	58	13
J	32.5	41	16
f, i, k, n	35.5	55	19
Average per sub-pixels	22.56	37.66	14.4

TABLE II
NUMBER OF 16-BIT ARITHMETIC OPERATIONS REQUIRED TO
INTERPOLATE SUB-PIXEL POSITIONS WITHIN 16-BITS ARITHMETIC

Sub-pixel position	H.264/AVC [1]	DAIF-16-bit
b, h	10	17
a, c, d, l	13	17
e, g, m, o	23	17
J	32.5	35
f, i, k, n	35.5	35
Average per sub-pixels	22.56	23

TABLE III
SIMULATION RESULTS OF THE 2-D-AIF AND PROPOSED DAIF SCHEMES
COMPARING TO THE H.264/AVC INTERPOLATION

Sequence	Delta Bit-rate, %			
	2-D-AIF [4]	SDIF [11]	DAIF-32	DAIF-16
Container ^a	-7.19	-4.78	-8.06	-9.06
Foreman ^a	-1.82	1.85	-2.27	-2.1
Silent ^a	0.34	0.03	0.31	-0.09
Paris ^b	-1.07	0.22	-0.86	-0.92
Foreman ^b	-6.63	-0.37	-5.67	-5.46
Mobile ^b	-5.33	3.53	-1.95	-2.04
Tempete ^b	-3.6	1.49	-1.6	-2.09
City ^c	-18.22	-13.89	-16.74	-16.18
BigShips ^c	-12.05	-9.15	-11.61	-11.57
ShuttleStart ^c	-19.33	-6.43	-18.33	-18.55
Crew ^c	-18.82	-2.95	-18.68	-19.14
Night ^c	-6.86	-3.85	-6	-5.9
Average	-8.38	-2.68	-7.62	-7.76

^a-QCIF resolution, ^b-CIF, and ^c-720p resolution of video sequences.

consider computational architecture, e.g., memory bandwidth, use of parallel arithmetic, and so on. However, we believe these estimates are useful for illustration purposes.

IV. EXPERIMENTAL RESULTS

Both of the proposed implementations of DAIF scheme, i.e., the 32-bit version as given in (7) with $Q = 8$ and the 16-bit version as given in (8) and (9), were adopted for integration into the official reference ITU-T VCEG-KTA software [12].

In order to test the performance of the proposed interpolations, we produced rate-distortion curves with VCEG-KTA software [8] following the common test conditions defined in the ITU-T/VCEG group [9] with the baseline profile settings.

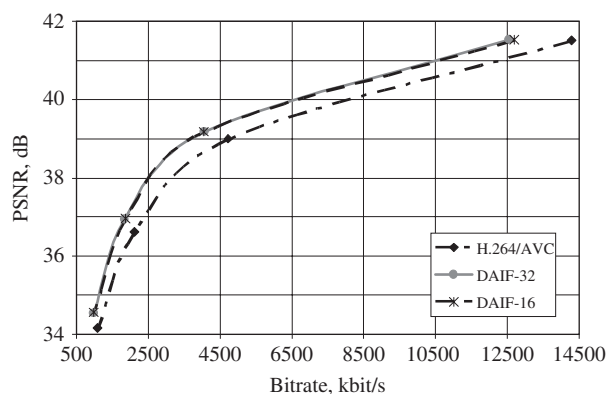


Fig. 3. Rate-distortion curves for "Crew" sequence, 60 frames/s.

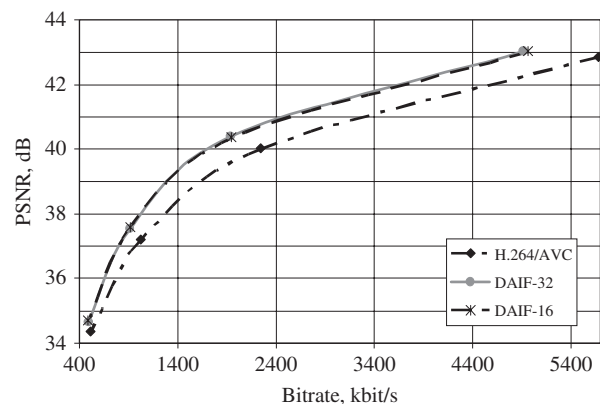


Fig. 4. Rate-distortion curves for "ShuttleStart" sequence, 60 frames/s.

Simulations were performed with test video sequences at QCIF, CIF, and 720p resolutions with a total of 300 coded frames for sequences at QCIF and CIF resolutions, and 150 coded frames for video materials at 720p resolutions. The standard H.264/AVC interpolation [1] was used as an anchor, and all simulation results are presented in Table III in terms of bit-rate reduction (%) compared to the anchor. Table III provides the simulation results for two integer implementations (32-bit and 16-bit) of the proposed DAIF scheme, noted as DAIF-32 and DAIF-16. Encoded bit streams for these schemes included side information for adaptive filters such as quantized filter coefficients, and thus the generated streams were fully decodable.

Table III also illustrates the simulation results for two more interpolation schemes: the 2-D-AIF which denotes the video coding with 2-D nonseparable adaptive filters (2-D-AIF) [4]; and the SDIF which is the most recently proposed static directional interpolation filtering [11]. Additionally, rate-distortion curves achieved with the proposed DAIF scheme for "Crew" and "ShuttleStart" test sequences are given in Figs. 3 and 4 respectively.

As seen from Table III and the figures, a significant coding gain is achieved by the DAIF scheme compared to H.264/AVC. This gain comes from both the directional filter structure (SDIF outperforms the H.264/AVC interpolation by 2.7% in bit-rate reduction) and from adapting filter coefficients (~7.7% of bit-rate reduction compared to H.264/AVC). Both DAIF schemes, the DAIF-32 and DAIF-16, provide comparable

coding gain, about -7.7% of delta bit rate on average and 15% of bit-rate reduction for 720p video material. However, DAIF-16 slightly outperforms the DAIF-32 on low-resolution video sequences. This is because of reduced number of bits needed for transmitting the filter coefficients for DAIF-16 due to the lower dynamic range of the filter.

In comparison to the 2-D-AIF, the proposed scheme achieves comparable results as well (-0.03 dB on average), with significantly lower complexity. Additionally, it should be noted that the proposed DAIF scheme achieves about 1% of bit-rate reduction over the 2-D-AIF interpolation for low-resolution video sequences (QCIF) due to the significantly reduced bit overhead required for filter coefficients transmitting.

V. CONCLUSION

The recent H.264/AVC video coding standard supports motion-compensated prediction with up to 1/4-pixel accuracy motion vectors. In this letter, we proposed a novel interpolation scheme using DAIF for each sub-pixel location. Experimental results show that DAIF achieves about 7.7% of bit-rate reduction on average and 15% for high-resolution video compared to the standard nonadaptive interpolation scheme of H.264/AVC, while requiring 36% fewer operations for interpolation. Compared to the 2-D-AIF scheme, the proposed scheme has practically the same coding efficiency with approximately three times less complexity. The DAIF scheme as well as its 16-bit implementation described in this letter were both applied to the ITU-T/VCEG standardization and successfully adopted in the reference VCEG-KTA software [12].

REFERENCES

- [1] Recommendation ITU-T H.264 & ISO/IEC 14496-10 (MPEG-4) AVC, "Advanced Video Coding for Generic Audiovisual Services," v.3: 2005.
- [2] T. Wedi and H. G. Musmann, "Motion and aliasing-compensated prediction for hybrid video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 577-587, Jul. 2003.
- [3] T. Wedi, "Adaptive interpolation filters and high-resolution displacements for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 484-491, Apr. 2006.
- [4] Y. Vatis, B. Edler, D. T. Nguyen, and J. Ostermann, "Motion and aliasing-compensated prediction using a 2-D nonseparable adaptive wiener interpolation filter," in *Proc. ICIP 2005*, vol. 2. Genova, Italy, Sep. 2005, pp. 894-897.
- [5] Y. Vatis and J. Ostermann, "Comparison of complexity between 2-D nonseparable adaptive interpolation filter and standard wiener filter," Nice, France, ITU-T SGI 6/Q.6 Doc. VCEG-AA11, Oct. 2005.
- [6] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, "Complexity of optimized H.264/AVC video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 717-725, Jul. 2003.
- [7] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. 3rd ed. New York: McGraw-Hill, 1991, ch. 13, pp. 580-629.
- [8] ITU-T VCEG. (2007, Aug. 7). version 1.4. *Reference ITU-T VCEG-KTA Software* [Online]. Available: <http://iphone.hhi.de/suehring/ttml/download/>
- [9] T. K. Tan, G. Sullivan, and T. Wedi, "Recommended simulation common conditions for coding efficiency experiments," Marrakech, Morocco, ITU-T Q.6/SG16, VCEG-AE010, Jan. 2007.
- [10] J. K. Tungait, "Modeling and identification of symmetric noncausal impulse responses," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 5, pp. 1171-1181, Oct. 1986.
- [11] A. Fuldseth, G. Bjontegaard, D. Rusanovskyy, K. Ugur, and J. Lainema, "Low-complexity directional interpolation filter," Berlin, Germany, ITU-T Q.6/SG16, VCEG-AI12, Jul. 2008.
- [12] D. Rusanovskyy, K. Ugur, and J. Lainema, "Adaptive interpolation with directional filters," Shenzhen, China, VCEG-AG21, Oct. 20, 2007.

- [P7] Ugur, K.; Lainema, J.; Gabbouj, M., "Adaptive Interpolation Filter with Flexible Symmetry for Coding High Resolution High Quality Video," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* , vol.1, no., pp.I-1013-I-1016, 15-20 April 2007

© 2007 IEEE. Reprinted with permission

ADAPTIVE INTERPOLATION FILTER WITH FLEXIBLE SYMMETRY FOR CODING HIGH RESOLUTION HIGH QUALITY VIDEO

Kemal Ugur¹, Jani Lainema¹, Moncef Gabbouj²

¹ Nokia Research Center, Tampere, Finland

² Tampere University of Technology, Tampere, Finland

ABSTRACT

In this work, a novel sub-pixel interpolation algorithm is proposed for video coders targeted towards high resolution and high fidelity use cases. Proposed scheme is based on adapting the interpolation filter's symmetry assumptions in a Rate-Distortion-Optimized fashion, taking into account the coding rate and the statistical properties of each image of the video sequence. Experimental results show that, using the proposed algorithm a gain of up-to 1.1 dB is achieved compared to non-adaptive sub-pixel interpolation of H.264/AVC. Compared to other state-of-the-art adaptive sub-pixel interpolation methods, a gain of up-to 0.5 dB is achieved. Proposed scheme outperforms H.264/AVC for all test cases; however, improvement is more significant at high bitrates and at high resolutions. This is especially important for future video coding solutions targeting high fidelity video applications.

Index Terms— Video Coding, Interpolation

1. INTRODUCTION

Motion Compensated Prediction (MCP) is a technique used by all popular video coding standards to exploit the temporal redundancy present in a video sequence. In MCP, each frame of a video sequence is divided into blocks and reference frame is searched for each block to find its best match. The relative position of the best matching block with respect to the original one is called the block's motion vector. Motion Vectors may have either integer or fractional pixel accuracies. If the motion vector has fractional pixel accuracy, decoder needs to perform interpolation to obtain the samples at sub-pixel positions. The state-of-the-art video coding standard, H.264/AVC, supports motion vectors with half or quarter pixel accuracy. The half-pixel samples are obtained by using a 6-tap FIR filter and quarter-pixel samples are obtained by bi-linear filtering using the two nearest samples at half or integer pixel positions [1].

The interpolation filter defined in H.264/AVC was designed to minimize the adverse effects of aliasing present in the input image sequence [2][3]. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in

literature to combat with this non-stationary effect of aliasing. [3] [4].

In [4], Vatis *et. al.* proposed to use 2D non-separable adaptive interpolation filter to reduce the prediction error energy. For each fractional pixel position, this scheme utilizes an independent filter and the coefficients for each filter are calculated by minimizing the prediction error energy. In order to reduce the number of bits used to code the filter coefficients, it was assumed that statistical properties of the image signal are symmetrical. With this assumption, filter coefficients are assumed to be the same, in case the distance of the corresponding full-pixel to the current sub-pixel is equal. This assumption greatly reduces the number of coefficients that need to be transmitted to the decoder.

In this paper, we claim that symmetry assumption in [4] has two shortcomings. Firstly, the statistical properties of the image sequence are not stationary and fixed symmetry assumption may not hold for every frame of the image sequence. For example, some frames of the video may contain complex scenes that do not hold neither horizontal nor vertical symmetry, whereas some other frames are relatively simpler and both horizontal and vertical symmetry assumption is valid. Secondly, the symmetry assumption in [4] does not provide a good trade-off between the accuracy of the interpolation filter and the overhead for coding the filter coefficients for high resolution, high quality material, and better trade-offs are possible to improve the coding efficiency significantly.

In order to exploit the varying statistical properties of the video sequence, and provide a better trade-off for high quality-high resolution material, we defined several filters with different symmetry assumptions, each one utilizing a different number of coefficients. It is proposed to adapt the filter's symmetry type for each frame of the video sequence in a Rate-Distortion Optimized fashion. Experimental results show that, when compared to H.264/AVC, proposed scheme improves the coding efficiency by up-to 1.1 dB. The improvement is 0.5 dB when compared to the scheme presented in [4]. This paper is organized as follows; Section 2 provides a brief background of 2D adaptive non-separable interpolation filtering. Section 3 presents the details of the proposed scheme. Section 4 presents the experimental results, and Conclusions are presented in Section 5.

2. OVERVIEW OF 2-D NON-SEPERABLE ADAPTIVE INTERPOLATION FILTERING

Consider Figure 1, where the pixels at integer positions are labeled by upper-case letters within shaded boxes, and other symbols represent sub-pixel positions to be interpolated. For each fractional pixel position, an interpolation filter is defined. Let's first assume that interpolation filters do not hold any symmetry. For positions a, b, c, d, h, n that are horizontally or vertically aligned with an integer pixel, one-dimensional 6-tap filter is used (Horizontal filter for positions a, b, c , and vertical filter for positions d, h, n). For other positions, two-dimensional 6x6 tap filter is utilized. Without any symmetry, this scheme results in 6 independent 1-D filters and 9 independent 2-D filters, resulting in 360 coefficients, which is a significant overhead to be transmitted to the decoder.

In order to decrease this overhead, it was assumed that statistical properties of the image signal are symmetric. Thus, the same filter coefficients are used in case the distance of the corresponding full-pixel positions to the current fractional-pixel position is the same. It is also assumed that distance in horizontal and vertical directions are treated equal. In other words, this assumption could be characterized as an interpolation filter with horizontal, vertical and diagonal symmetry, and is denoted as *HVD-Filter*.

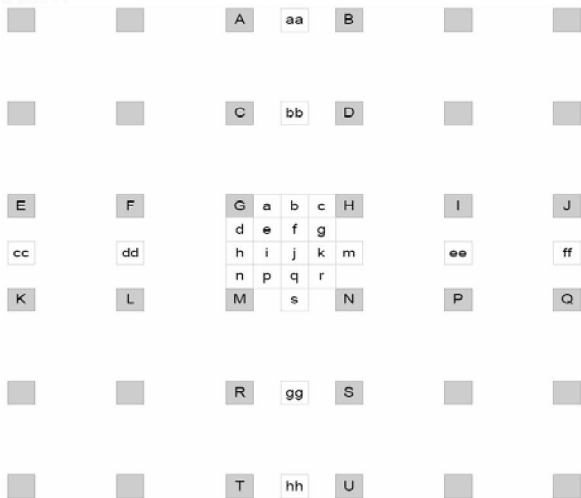


Figure 1 Pixel Positions Used in Interpolation

With the symmetry assumption in place, the number of coefficients needed for defining the interpolation filter is reduced in two ways. First, the number of filters to be transmitted are reduced from 15 to 5 (only filters for positions a, b, e, f, j need to be transmitted). Also, a smaller number of coefficients are needed for each filter (for example, only 3 coefficients are needed to define the 1-D 6-tap filter for sub-pixel location b as horizontally symmetric filter is being used).

3. PROPOSED INTERPOLATION FILTER WITH FLEXIBLE SYMMETRY

The symmetry assumption described in Section 2.1 provides a good trade-off between the accuracy of the interpolation filter and the overhead used to code the filter coefficients for low and medium resolution material. However, for coding higher resolutions at higher qualities, this tradeoff does not hold. In addition, the statistical properties of the image sequence may not be stationary, and the symmetry assumption for the interpolation filter may not hold for every frame of the video sequence. Some frames of the video may contain complex scenes that neither holds horizontal nor vertical symmetry, whereas some other frames could be relatively simpler and both horizontal and vertical symmetry assumption is valid.

In order to exploit the varying statistical characteristics of the video sequence, and provide a better trade-off for high quality-high resolution material, we defined several filters with different characteristics. Each filter holds a different symmetry type, and is defined with different number of coefficients.

3.1. 2-D Non-Separable Interpolation Filters with Flexible Symmetries

In addition to the *HVD-Filter* as described above, we define four new filters with different combinations of symmetry assumptions. More specifically, following filters are being used in our proposed scheme:

- Filter with Horizontal Symmetry (*HOR-Filter*): Image signal is assumed to have only horizontal, but no vertical or diagonal symmetry.
- Filter with Vertical Symmetry (*VER-Filter*): Image signal is assumed to have only vertical, but no horizontal or diagonal symmetry.
- Filter with Horizontal and Vertical Symmetry (*H+V-Filter*): Image signal is assumed to have horizontal and vertical, but no diagonal symmetry.
- Filter with No Symmetry Assumptions (*FULL-Filter*): Image signal is assumed to have no horizontal, vertical and diagonal symmetry.

Figure 2 illustrates the number of filters used for each one of the above symmetry assumptions. In Figure 2, each square represents a sub-pixel pixel position, and the number inside each square represents the identification number of the interpolation filter used for the corresponding sub-pixel position. Sub-pixel positions with the same identification number share the same interpolation filter (For example, for the *HVD-Filter*, the sub-pixel locations a, c, d, n use the same filter that is denoted by ID number **1**). The coefficients of the filter that is shared by more than one sub-pixel positions are transmitted only once. Coefficients are transmitted only for the shaded sub-pixel positions as denoted in Figure 2, whereas for other positions denoted as white squares no coefficients are transmitted.

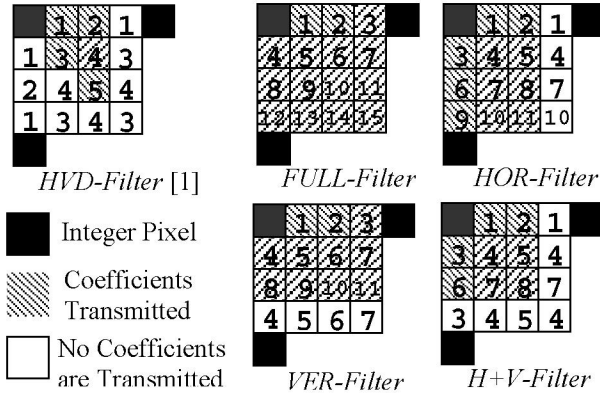


Figure 2 Number of Independent Filters for Different Symmetries

As seen from Figure 2, the number of filters changes for each of different symmetry assumptions. For example, 5 filters are needed for *HVD-Filter*, whereas 11 filters are used for *HOR* and *VER* Filters.

In addition to the number of filters, number of coefficients used for each filter depends on the filter's symmetry assumption. For example, consider *HOR-Filter* where only horizontal symmetry is assumed. In this case, sub-pixel position *b* uses a symmetrical horizontal filter, whereas position *h* uses a 6-tap non-symmetric vertical filter. Similar to *HVD-Filter*, all filter types except the *FULL-Filter* use a 1D filter for sub-pixel positions that are horizontally or vertically aligned with an integer position (sub-pixel positions *a, b, c, d, h, n*). For the *FULL-Filter* these sub-pixel positions do not use a 1D filter, but a 2D filter. It is known that using a 2D filter is not necessary to remove the aliasing effect for positions aligned with the integer pixel. However, using a 2D filter for those positions is useful to combat with the adverse effects of quantization and motion estimation errors that also decrease the prediction quality [3].

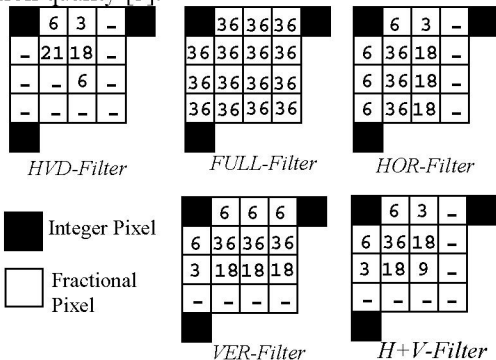


Figure 3 Number of Coefficients for Each Sub-Pixel Position at Different Symmetries

Figure 3 illustrates the number of coefficients used by each sub-pixel position for the filters with different symmetry assumptions. The numbers inside each square

represent how many coefficients the interpolation filter uses for the corresponding sub-pixel position. For example, for position *j*, *HVD-Filter* uses 6 coefficients, whereas in *HOR-Filter* and *VER-Filter* it uses 18 coefficients. The total numbers of coefficients are 54, 540, 189, 189 and 99 for the *HVD*, *FULL*, *HOR*, *VER* and *H+V-Filters* respectively.

3.2. Calculating Interpolation Filter's Symmetry Type and Coefficients

In order to find the symmetry type of the interpolation filter and its corresponding coefficients, we make use of the following encoder algorithm that is useful to estimate the upper bound performance of our proposed scheme.

1. Motion Estimation is performed using the standard non-adaptive H.264/AVC interpolation filter, and motion vectors for every macroblock are found.
2. For each symmetry type, the filter coefficients are calculated to minimize the prediction error energy. This step is done by constructing and solving a set of linear equations using the previously reconstructed frame and motion vectors found in Step-1. The number of linear equations is equal to the number of coefficients that need to be transmitted to the decoder for the given symmetry type (For details of constructing the linear equations, reader is referred to [4]).
3. Using the interpolation filter found in Step-2, frame is re-encoded. The cost for using the given filter symmetry type is calculated using Equation 1.

$$J(f) = D(f, \mathbf{m}) + \lambda_{mode} \cdot (R_1 + R_2) \quad (1)$$

where $\mathbf{m} = (mx, my)^T$ is the new motion vector set for the frame, λ_{mode} being the Lagrange multiplier as defined in [6] and *f* denoting the interpolation filter's symmetry type. *D* is the distortion term, and it is defined as the Mean Square Error (MSE) between the original and reconstruction signal. *R₁* is the number of bits used to code the residual signal including the header information (such as motion vectors, macroblock modes etc.) and *R₂* is the number of bits used to code the filter coefficients. When calculating *R₂* it is assumed that filter coefficients are quantized and coded using 10 bits per coefficient.

4. Steps 2&3 are repeated for each candidate symmetry type. Filter symmetry with the minimum cost is chosen, and its corresponding coefficients are signaled to the decoder.

It should be noted that, this encoding scheme requires several encodings per frame, in order to achieve the practical upper-bound performance. However, the number of encodings could be reduced to two per frame, by removing Step-3 and using a modified cost function that does not need the reconstruction for each candidate filter. This approach is conceptually similar to fast mode decision in H.264/AVC software that does not require the reconstructed macroblock [5]. The decoder complexity of our proposed scheme is practically same as in [4], except that *FULL-Filter* has slightly higher complexity because of using 2D filters for positions *a, b, c, d, h, n*.

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of our proposed scheme, several sequences at 4CIF (704x576) and 1080p (1920x1080) resolutions are first encoded using the H.264/AVC standard without any adaptive sub-pixel interpolation. Same sequences are then encoded utilizing the adaptive interpolation scheme described in [4] where only *HVD-Filter* is used and then using our proposed scheme that adaptively changes the interpolation filter's symmetry assumptions. Coding tools common in the Baseline profile of H.264/AVC standard are used in all the simulations and we limit the number of reference frames to one. Quantization Parameter (QP) is kept constant for each simulation (no rate control is employed), and QP takes the values between 24 and 32. Figures 4 and 5 present the results for two standard test sequences. For sequence *Soccer*, an average gain of 0.9 dB is achieved compared to H.264/AVC, with gains up-to 1.1 dB at high bitrates. Our scheme outperforms [4] by 0.5 dB at high bitrates and by around 0.25 dB at lower bitrates. Having a larger gain at high bitrates validates our claim that symmetry assumption in [4] does not provide a good trade-off for higher bitrate encoding. In order to see how bitrate affects the choice of filter types, we compare the selection frequency of different filters for two different QP's for the *Soccer* sequence in Figure 6 (For the sake of clarity, only three filters are shown in the figure). It is seen that at low bitrates (with large QP), encoder mostly chooses filters that does not require much overhead for the coefficients. However, with the increasing bitrate more accurate filters utilizing more coefficients are chosen and the coding efficiency is improved further.

5. CONCLUSIONS

In this paper, we presented a novel sub-pixel adaptive interpolation filtering scheme to improve the coding efficiency of a video coder. Proposed scheme is based on adapting the interpolation filter's symmetry at each frame in a Rate-Distortion-Optimized fashion, taking into account the coding rate and the statistical properties of each image. Experimental results show that the proposed scheme outperforms the H.264/AVC standard by up to 1.1 dB. When compared to [4], a gain of up-to 0.5 dB is achieved. Proposed scheme outperforms H.264/AVC and [4] for all test cases; however, improvement is more significant at high bitrates and at high resolutions. This is especially important for future video coding solutions targeting high fidelity video applications.

6. REFERENCES

[1] Karczewicz, M.; Hallapuro, A.; "Interpolation solution with low encoder memory requirements and low decoder complexity.", VCEG-N31, 24-27 Sept. 2001.

[2] O. Werner, "Drift analysis and drift reduction for multiresolution hybrid video coding", Signal Processing: Image Commun., vol. 8, no. 5, July 1996.

[3] T. Wedi and H. G. Musmann, "Motion- and Aliasing-Compensated Prediction for Hybrid Video Coding", IEEE Trans. on CSVT, vol. 13, No. 7, July 2003.

[4] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion-and Aliasing-compensated Prediction using a Two-Dimensional Non-Separable Adaptive Wiener Interpolation Filter", in Proc. IEEE ICIP 2005, Genova, Italy, September 2005.

[5] H.264/AVC Reference Software [Online]. Available: <http://iphome.hhi.de/suehring/tml/download/>

[6] Wiegand, T.; Schwarz, H.; Joch, A.; Kossentini, F.; Sullivan, G.J., "Rate-constrained coder control and comparison of video coding standards," IEEE Transactions on CSVT, vol.13, no.7pp. 688- 703, July 2003

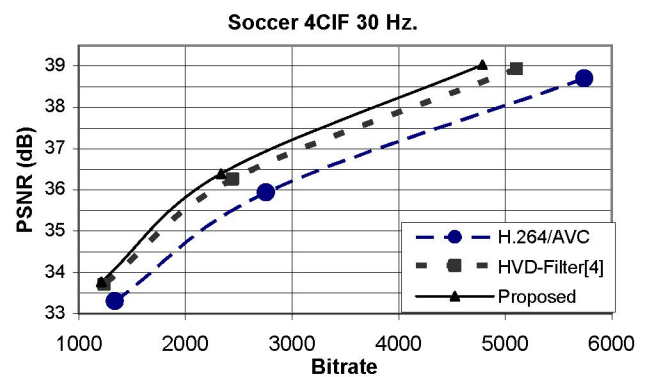


Figure 4 Coding Results for Soccer 4CIF Sequence

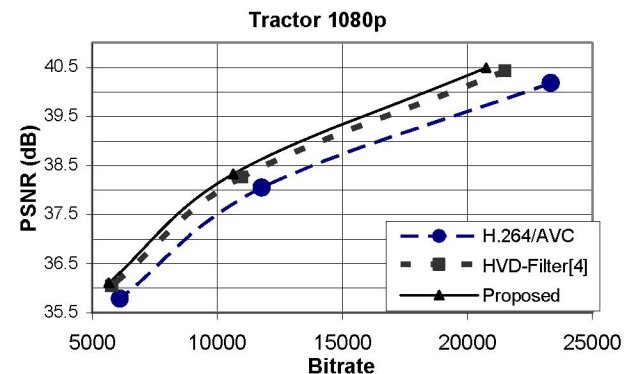


Figure 5 Coding Results for Tractor 1080p Sequence

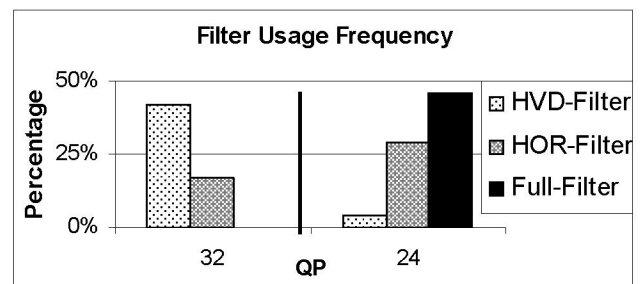


Figure 6 Filter Usage Statistics at Different Bitrates

- [P8] Rusanovskyy D., Ugur K., Gabbouj M., “Adaptive Interpolation with Flexible Filter Structures”, *Proceedings of IEEE International Conference on Image Processing, ICIP 2009*, Cairo, Egypt, Nov. 2009, pp. 1025-1028.

© 2009 IEEE. Reprinted with permission

ADAPTIVE INTERPOLATION WITH FLEXIBLE FILTER STRUCTURES FOR VIDEO CODING

^{1,2)}Dmytro Rusanovskyy, ²⁾Kemal Ugur, ¹⁾Moncef Gabbouj

¹⁾Institute of Signal Processing, Tampere University of Technology, Tampere, Finland

²⁾Nokia Research Center, Nokia Inc., Tampere, Finland

ABSTRACT

Two novel algorithms are proposed for improving the coding efficiency of adaptive interpolation schemes for video codecs, without increasing implementation complexity. Proposed algorithms utilize two different filter structures with equal tap length, but with complementary frequency responses. Depending on the content being coded, encoder selects which one of the two filter structure is optimal and signals this information to the decoder. In addition, the symmetry of filters is not pre-defined but is flexible. Encoder selects the optimal filter symmetry depending on the coding rate and the content and signals this information to the decoder. Experimental results show, that proposed improvements bring up to 7% of bit-rate reduction at high bit-rate over conventional adaptive interpolation. When compared to H.264/AVC, average gain over the test set is 11%. Coding efficiency is improved without increasing the complexity, thus proposed algorithms are suitable for mobile multimedia use-cases, where the computational resources are very limited.

Index Terms— Adaptive interpolation, video coding, interpolation filter.

1. INTRODUCTION

The standard H.264/AVC codec utilizes motion compensated prediction with motion vectors at fractional (up-to 1/4-pixel) resolution. In order to estimate and compensate fractional-pixel displacements, samples at sub-pixel positions need to be interpolated. Interpolation filters of H.264/AVC were designed to minimize the adverse effects of aliasing present in wide range of image sequences. However, aliasing in a video data is not a stationary process, but has a varying characteristic. Adaptive interpolation filters (AIF) that change the filter coefficients at each frame have been proposed in literature to combat this non-stationary effect of aliasing [1-4]. These schemes all provide comparable coding performance over the standard

H.264/AVC scheme and they have been adopted by ITU-T VCEG standardization group in the reference KTA software [5].

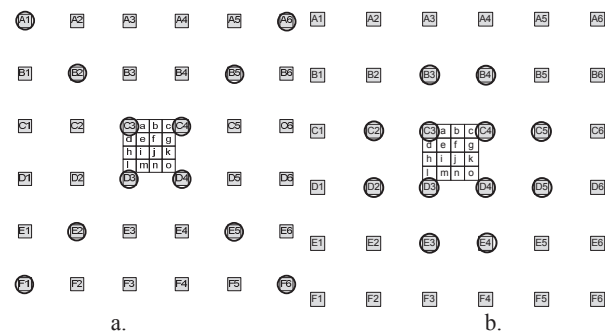


Figure 1. 12-tap filter structures: a. – diagonal cross filter; b. – radial support filter.

In [1], Vatis et. al. proposed a scheme with 36-taps 2D non-separable adaptive interpolation filters, a scheme with separable 36-tap filters was proposed in [2]. Following this, two low complexity adaptive interpolation schemes that use a maximum of 12-tap filters were proposed in [3] and [4]. Directional adaptive interpolation filters (DAIF) [3] uses 12-tap sparse filter structure in a form of diagonal cross, and authors of [4] utilize 12-tap filter structure with a radial support, as shown in Fig. 1. Algorithms [1-4] utilizes filter symmetry to reduce the overhead of transmitting filter coefficients. Same filter-coefficients are used in the case of equal distances between the corresponding full-pixel positions and the current sub-pixel position. This assumption allowed to AIF schemes be applicable for coding of the low resolution sequences, but restricts AIF performance for HD videos at high bitrates.

In this paper, we modified the DAIF scheme [3] with two new techniques allowing larger flexibility for interpolation filters. First tool relaxes the symmetry assumption described above and instead uses an adaptive symmetry that can change depending on the coded bitrate. Symmetry descriptor information is transmitted to the decoder along with the filter coefficients. Decoder reads the symmetry

descriptors and performs filter coefficient decoding accordingly. The second tool is switching 12-tap filter structure for five central sub-pixel locations $\{f,i,j,k,n\}$. Selection is done among two candidates, which are 12-tap diagonal cross of [3] and 12-tap radial support filter of [4], see Fig. 1 and have complementing frequency responses. Proposed improvements of the DAIF scheme require no complexity increase to interpolation at decoder and bring up-to 7% of bit-rate reduction for HD sequences. Complexity increase at the encoder side is negligible comparing to the original DAIF scheme.

This paper is organized as follows; Section 2 describes the basics of the DAIF scheme. Section 3 presents the details of the proposed techniques. Experimental results are given in Section 4, and Section 5 concludes the paper.

2. DIRECTIONAL ADAPTIVE INTERPOLATION

Consider Fig. 1, where integer pixel locations are labeled by upper-case letters $\{A1,...,F6\}$ within the shaded boxes, and lower-case symbols $\{a,b,...,o\}$ represent samples at sub-pixel locations that are to be interpolated. Let's denote an interpolation filter used for each sub-pixel Sp , $Sp \in \{a,b,...,o\}$, with $h(Sp)$. The direction of these filters is determined in the DAIF according to the alignment of the corresponding sub-pixel Sp with the integer-pixels sets. We denote these sets as $s(Sp) \subseteq \{A1,...,F6\}$.

To interpolate locations $\{a,b,c\}$, $\{d,h,l\}$, and $\{e,g,o,m\}$, the DAIF scheme utilizes 1D 6-tap filters aligned horizontally, vertically or diagonally respectively. Five central sub-pixel locations $\{f,i,j,k,n\}$ are interpolated in with a sparse 12-tap filter spanning the integer-pixels that lie on a diagonal cross structure: $s(f|i|j|k|n)=\{A1,B2,...,F6\} \cup \{F1,E2,...,A6\}$, see Fig.1.a.

The sets of integer-pixel locations that contribute to the interpolation at each sub-pixel locations are denoted with $s(Sp)$ and are given explicitly as following.

$$\begin{aligned} s(a) &= s(b) = s(c) = \{C1, C2, C3, C4, C5, C6\}, \\ s(d) &= s(h) = s(l) = \{A3, B3, C3, D3, E3, F3\}, \\ s(e) &= s(o) = \{A1, B2, C3, D4, E5, F6\}, \\ s(m) &= s(g) = \{F1, E2, D3, C4, B5, A6\}, \\ s(j) &= s(f) = s(i) = s(k) = s(n) = \\ &\quad \{A1, B2, C3, D4, E5, F6, F1, E2, D3, C4, B5, A6\}. \end{aligned} \quad (1)$$

The DAIF defines a set of 15 independent filters $\{h(Sp), Sp \in \{a,b,...,o\}\}$ for each coded frame and is fully described with 120 filter coefficients that are computed for each coded frame and transmitted to the decoder. In order to reduce this bit overhead, the original DAIF imposed a full symmetry assumption on the image signal characteristics. The same filter-coefficients are used in the case of equal distances between the corresponding full-pixel positions and the current sub-pixel position. Sub-pixels $\{a,...,o\}$ were

clustered in several groups (defined by symmetry) and a single independent filter is computed for each of these groups. A single 6-tap filter is computed and transmitted for sub-pixel locations $\{a,c,d,l\}$. Similarly, independent filters were defined for locations $\{e,g,o,m\}$, $\{f,i,k,n\}$, $\{c,h\}$ and $\{j\}$ correspondingly. Due to such arrangement, the total number of filter coefficients to be transmitted in the DAIF scheme was reduced from 120 to 24. Such symmetry made the DAIF be applicable for coding of low resolution sequences, but penalized the coding performance at high bit-rate.

3. THE PROPOSED ALGORITHMS

We propose two new coding tools to improve coding performance of the DAIF scheme. First tool relaxes the symmetry assumption imposed in [3] and instead uses an adaptive symmetry that can change depending on the coded bitrate. The second tool utilizes switching between two different 12-tap filter structures with complementary frequency responses.

3.1. Flexible Filter Symmetry

Adaptive interpolation with flexible symmetry was first proposed in [7]. Authors defined a number of filter sets with different symmetry assumptions. Encoder selects the optimal filter by encoding each frame multiple times for each candidate filter, and selecting the filter giving the smallest Rate-Distortion cost. Such approach limits the number of possible symmetries and drastically increases the encoding complexity, as each frame to be encoded multiple times.

Our proposed tool is more flexible and it allows more variety for filter symmetries. This variety is achieved by signaling a symmetry descriptor for each sub-pixel, which carries the information if the corresponding sub-pixel uses an independent filter or shares the same filter with another sub-pixel. If the same filter is shared with more than one sub-pixel locations, only one filter is transmitted to the decoder.

The symmetry descriptor for each sub-pixel is calculated as follows. First, for each sub-pixel location Sp adaptive filter $h(Sp)$ is computed without assuming any symmetry. We treat each N-tap length filter as a vector in N-space, and compute the Euclidean distance (2) between coefficients of different sub-pixel positions $Sp1$ and $Sp2$.

$$d_{Sp1,Sp2} = \sqrt{\sum_{i=0}^{N-1} (h_i(Sp1) - h_i(Sp2))^2} \quad (2)$$

This distance is compared against a threshold that is function of the bit-rate. If the distances between filters are lower than the threshold, those sub-pixel samples share a single filter which is re-computed from merged covariance matrixes of those two sub-pixel locations. Fig. 2, illustrate this process, where filters $h(a)$ and $h(c)$ to be merged, as

their Euclidian distance is lower than the threshold, whereas sub-pixels d and l will use independent filters.

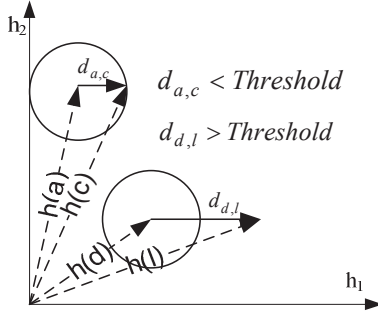


Figure 2. An interpretation of 2-tap length filters diversity analysis. Euclidian distance (solid arrow) between two filters (dashed arrows) is compared against the threshold (circle of allowed).

The threshold value is calculated by running extensive test simulations over a wider set of test video sequences coded at different bit-rates. By doing this, we found an empirical threshold which is modelled a quadratic function of coded frame bit-rate $threshold = (nBits / 2 \cdot 10^4)^2$.

3.2. Switching Filter Structure

In addition, we propose interpolation of $\{f,i,j,k,n\}$ sub-pixels with switching between two different 12-tap filter structures. Those structures are diagonal-cross and radial filters, their spatial supports are show in Fig. 1.a and Fig. 1.b respectively.

The spatial supports of these 12-tap filter are explicitly defined as

$$s(Sp) = \{A1, B2, C3, D4, E5, F6, F1, E2, D3, C4, B5, A6\} \quad (3)$$

for diagonal cross filter, and as

$$s(Sp) = \{B3, B4, C2, C3, C4, C5, D2, D3, D4, D5, E3, E4\} \quad (4)$$

for radial support filter.

In order to compare frequency responses of diagonal-cross and radial filters, we display their cut-off frequencies (-3dB level) in Fig. 3 and compared against the standard 2D 6x6-tap H.264/AVC filter. One can see that the 12-tap diagonal-cross filter provides better support for frequencies in horizontal and vertical directions (dash-doted line). Its frequency support in these directions outperforms the support of the radial filter and is close to the cut-off frequency of 36-tap filter of H.264/AVC. However, the diagonal cross filter has “weaker” support (low cut-off frequency) in the diagonal directions, whereas 12-tap radial filters, in contrast, performs much better (solid thin line).

Considering this, we propose an interpolation scheme which has an ability to switch between these 12-tap filters and thus has an advantage of efficient use of tap-length against interpolation with a fixed filter structure.

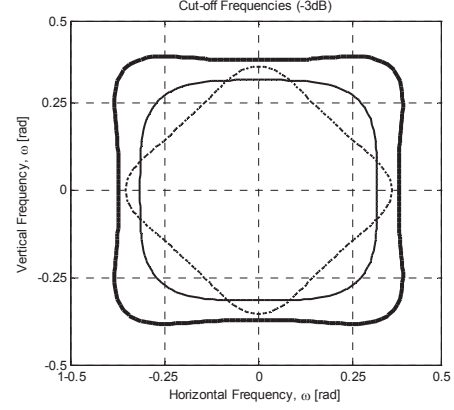


Figure 3. Cut-off frequencies (at the level -3dB) for three $h(j)$ filters: thick solid line – 2D 6x6 filter of H.264/AVC, thin solid line – estimate of 12-tap Radial Filter, thin dash-dotted line – estimate of Diagonal-Cross Filter.

3.3. Encoder Algorithm

In this section, we describe in detail how the encoder selects the optimal filter symmetry and the filter structure and also we present a brief analysis of the proposed tools on the encoder and decoder complexity. The encoder algorithm is detailed step by step below;

1. Frame is coded with the fixed filter given in H.264/AVC and motion vectors are computed
2. Using the motion vectors and the reference frame, encoder computes independent filters for each sub-pixel $\{a, \dots, o\}$. For sub-pixel locations $\{f,i,j,k,n\}$ filters for two different structures are computed, diagonal-cross and radial.
3. The decision between radial and diagonal cross filter is made based on the motion prediction error. That is, by using the motion vectors obtained at the first step, the error between original and prediction image is computed for both radial and diagonal cross structure. The filter structure that gives the smallest error for each sub-pixel is chosen.
4. The symmetry between each sub-pixel location is calculated as described in Section 3.1, using the bit-rate obtained in the Step-1. According the calculated symmetry, the filter coefficients are re-computed.
5. Input frame is encoded with adaptive interpolation filter, identically as it is done in [3].

Proposed algorithm requires no additional encoding passes compared to the original DAIF scheme, thus encoding complexity increase can be considered negligible. At the decoder side, the complexity is identical to the DAIF design, as the interpolation for each sub-pixel is performed using the same number of taps.

4. EXPERIMENTAL RESULTS

Proposed algorithms were integrated and proposed for adoption into the JMKA software maintained by ITU-T VCEG standardization group [5]. In order to test performance, we simulated rate-distortion curves with the reference KTA software following the common test conditions defined in the ITU-T/VCEG group [6] with the High Profile settings. Simulation were performed with test sequences listed in [6] with total amount of 150 coded frames for sequences at 720p resolutions, and 60 coded frames for video materials at 1080p resolution.

Three interpolation schemes have been compared; those are proposed in this paper tools ("Proposed"), original algorithm ("Conventional"), and the standard H.264/AVC, which was utilized for benchmarking. Table I presents the simulation results for interpolation schemes in terms of delta bit-rate reduction [6] compared to the standard H.264/AVC. Additionally, Fig. 4 presents simulated rate-distortion curves for "Raven" 720p test sequences.

As seen in Table I proposed algorithms provide an up-to 24.60% bitrate reduction compared to the standard H.264/AVC interpolation, with an average improvement of 10.94%. We then compared the performance of the proposed tools with conventional adaptive interpolation filtering that does not utilize the flexible tools proposed in this paper. As seen in Figure 4, improvement obtained using the proposed scheme is very visible at high bitrates and becomes up-to 7%. This verifies our assumption that having fixed filter symmetry used in previous schemes is not optimal for coding at high bitrates, and using adaptive filter symmetry improves the coding efficiency significantly.

5. CONCLUSIONS

In this paper, we proposed two novel algorithms for adaptive interpolation that allow flexibility in filter structure. First tool replaces the fixed symmetry of adaptive interpolation filters with a flexible structure. Encoder calculates the optimal filter symmetry between independent filters of different sub-pixel locations based on the coded rate and signals this information to the decoder. The second tool is switching the 12-tap filter structure for central sub-pixel locations, between two candidates that have complementary frequency responses.

Experimental results shows, that proposed improvements bring up-to 7% of bit-rate reduction at high bit-rates in comparison to the original DAIF design and require no complexity increase at the decoder side.

11. REFERENCES

- [1] Y. Vatis, B. Edler, D. T. Nguyen and J. Ostermann, "Motion and Aliasing-Compensated Prediction Using a Two-dimensional Non-Separable Adaptive Wiener Interpolation Filter", Proc. ICIP 2005, Genova, Italy, September 2005.
- [2] S. Wittman and T. Wedi, "Separable adaptive interpolation filter", ITU-T SG16/Q6, Doc. C-0219 July, 2007.
- [3] D. Rusanovskyy, K. Ugur, J. Lainema, M. Gabbouj, "Video Coding with Pixel-Aligned Directional Adaptive Interpolation Filters," In Proc. ISCAS'08, 2008.
- [4] Y. Ye, M. Karczewicz, "Enhanced adaptive interpolation filter", ITU-T SG16/Q6, doc. C-0464 April, 2008
- [5] Reference ITU-T VCEG-KTA Software [Online]. Available: <http://iphone.hhi.de/suehring/tml/download/>
- [6] T. K Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments," ITU-T Q.6/SG16, VCEG-AJ010, San-Diego, CA, USA, October 2007
- [7] K. Ugur, J. Lainema, M. Gabbouj, "Adaptive Interpolation Filter with Flexible Symmetry for Coding High Resolution High Quality Video", In Proc. Acoustics, Speech and Signal Processing (ICASSP) 2007, Honolulu, HA, USA.

Table I Coding efficiency of the proposed scheme in terms of delta bit-rate [6] with respect to the H.264/AVC interpolation.

Sequence	Δ Bit-rate, %	
	Conventional	Proposed
BigShips(720p)	-8.56	-9.37
Night(720p)	-5.79	-6.50
City(720p)	-12.77	-14.41
Crew(720p)	-10.50	-10.67
Raven(720p)	-21.74	-24.60
Jets(720p)	-4.23	-6.49
CrowdRun(1080p)	-1.02	-1.48
ParkJoy(1080p)	-1.34	-2.10
Toys&Call(1080p)	-8.53	-10.14
Sunflower(1080p)	-21.97	-23.64
Average	-9.65	-10.94

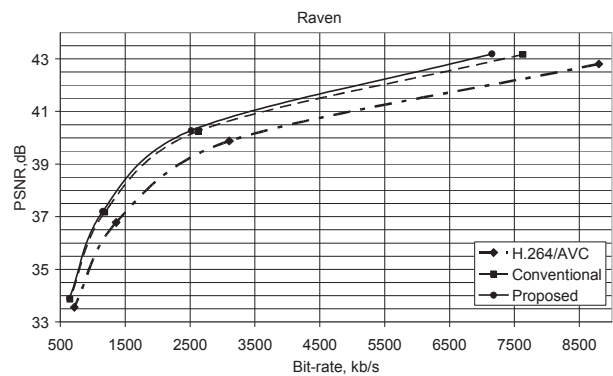


Figure 4. Rate-distortion curves for "Raven" 720p test video sequence at 60 fps.

- [P9] Ugur, K.; Hui Liu; Lainema, J.; Gabbouj, M.; Houqiang Li, “Parallel Encoding - Decoding Operation for Multiview Video Coding with High Coding Efficiency,” *3DTV Conference, 2007* , pp.1-4, 7-9 May 2007

© 2007 IEEE. Reprinted with permission

PARALLEL ENCODING - DECODING OPERATION FOR MULTIVIEW VIDEO CODING WITH HIGH CODING EFFICIENCY

Kemal Ugur¹, Hui Liu², Jani Lainema¹, Moncef Gabbouj³, Houqiang Li²

¹: Nokia Research Center, Tampere, Finland

²: University of Science and Technology of China, Hefei, China

³: Tampere University of Technology, Tampere, Finland

ABSTRACT

Multiview Video Coding (MVC) standardization is an on-going effort aiming to extend H.264/AVC by developing novel tools optimized for 3D and multiview video use cases. One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views. The parallel operation is known to be especially important for 3DTV applications, where the display needs to output many views simultaneously to support head-motion parallax. In this paper, we present a novel coding structure that enables parallel encoder/decoder operation for different views, without compromising from the coding efficiency. This is achieved by systematically restricting the reference area of each view, so that encoding and decoding of macroblocks from different views could be efficiently pipelined and parallel operation of separate views becomes possible. As the inter-view prediction is still used, proposed structure achieves up to 0.9 dB gain compared to simulcast, maintaining very similar desirable parallelism characteristics.

1. INTRODUCTION

With recent advances in capture and display technologies, 3D video communication and entertainment services are becoming a reality and will enter into consumer domain in the near future. To enable these upcoming applications that range from stereoscopic video broadcasting to 3DTV, an efficient Multi-view Video Coding (MVC) standard is necessary. Recently, the Joint Video Team (JVT) comprising VCEG and MPEG standardization groups undertook the effort to standardize an MVC standard by extending H.264/AVC and including novel tools optimized for MVC.

One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views [1]. The parallel processing of different views is especially important for 3DTV use cases, where the displays need to output many views simultaneously to support head-motion parallax. However, inter-view dependencies between pictures may impose serious parallelism issues to the video system, because two pictures at different views need to be decoded sequentially. Let's

consider a 3DTV system displaying simultaneously two views, and views are coded with the coding structure as illustrated in Figure-1. In order to decode a picture in view-1 at any temporal instant, the picture in view-0 at the same temporal instant shall be decoded first. Only way to display two views at the same time is by having an MVC decoder running two times faster than a regular single-view decoder. Even though two independent decoders running on different platforms might be available, both decoders need to run twice faster than the single-view decoder because decoding has to be performed sequentially. The situation gets worse, with the increasing number of views that is supported by the 3D display. Currently there are commercial displays which can display 100 views simultaneously, and if all the views depend on each other, then the decoder must run 100 times faster, which is very challenging.

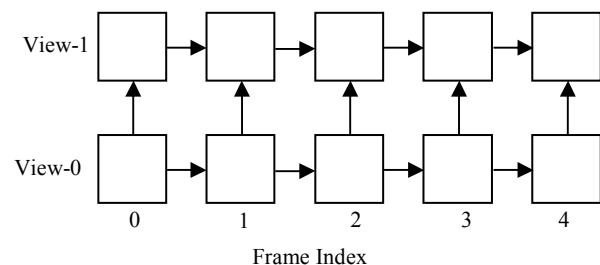


Figure 1. Sample Prediction Structure for Two Views

One way to increase the parallelism is to code each view independently. However, this kind of simulcast approach results in a significant penalty in coding efficiency as inter-view redundancies are not exploited at all. In this paper, we present a coding structure that enables parallel encoder/decoder implementation for different views, even though there are dependencies between views. This is achieved by encoding views with some constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Resulting bitstreams generated with the proposed structure could be decoded in parallel, because macroblocks belonging to different views could be efficiently pipelined. As the inter-view prediction is still used, significant gain compared to simulcast is achieved, while maintaining almost the same desirable parallelism characteristics. Also,

because the parallelism is achieved by encoding constraints, no low-level changes are introduced to the MVC decoder. Experimental results show that, when compared to simulcast approach, proposed system achieves 0.9 dB gain on average with similar parallelism requirements.

This paper is organized as follows: Section 2 provides the details of the proposed coding scheme and details a parallel decoder operation for two views as an example. Section 3 presents some design considerations and trade-offs possible with the scheme. Section-4 presents the experimental results. Conclusions are presented in Section 5.

2. PARALLEL DECODING OPERATION FOR MVC

In order to better understand the parallel processing requirements for MVC, consider the structure illustrated in Figure-1, where two views are encoded using both temporal and view prediction. The decoding of a frame in view-1 cannot start before all its references are fully decoded and corresponding reconstructions are made available in the memory. This means that, before the decoding of any frame in view-1 could start, both its temporal and view references must be decoded first. Let's assume the decoding time for view-0 and view-1 pictures are τ_0 and τ_1 respectively. Then, the time needed to decode two pictures captured at the same time instant from both views would be $\tau_0 + \tau_1$. Even if there are resources for having separate decoders for each view, time needed to decode views at the same time instant cannot be reduced, as decoders cannot run in parallel, but have to wait for each other. This need to perform sequential decoding in view direction constitutes a serious problem with the increasing number of dependent views, and may become a very challenging task for implementation.

The main concept behind our proposal is to use certain encoder restrictions so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Let's consider a similar example as above, where two pictures from view-1 and view-0 are going to be decoded, and view-1 picture references view-0 picture as illustrated in Figure 2 (for simplicity the size of the frames are five macroblocks both horizontally and vertically). Assume the video is encoded in a way that macroblocks in view-1 picture could only use reconstruction values of macroblocks that belong to certain rows in view-0 picture. For example, the macroblocks in the first row of view-1 picture could only use reconstruction values from the first two macroblock rows in view-0 picture. In other words, the available reference area for the first macroblock row of view-1 picture constitutes only data from the first two macroblock rows of view-0 picture (i.e. the motion vectors for the view-1 macroblocks are restricted). Similarly, the second macroblock row of view-1 picture only uses reconstruction values of the first three macroblock rows of the View-0 picture. This systematic restriction of reference area enables parallel decoding of first row of

view-1 with any row below the second of view-0, as they are not referring each other (the details of the parallel decoding operation is presented in the next section).

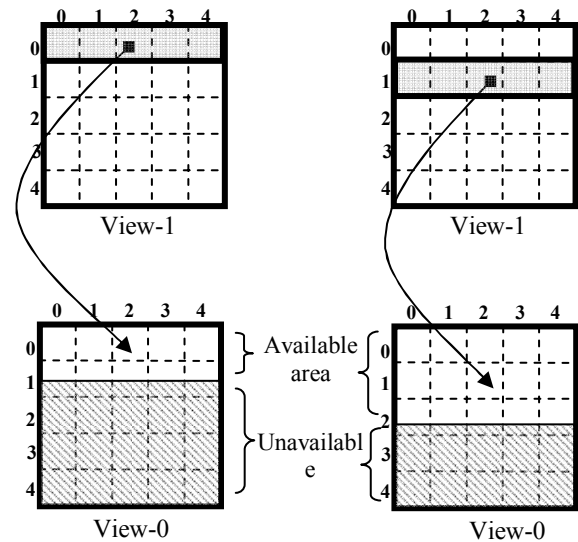


Figure 2. Reference Area Restriction

It should be noted that the available reference area for the first macroblock row in view-1 does not include all the pixels of first two macroblock rows of view-0, but cropped by three pixels from bottom. The reason for cropping is to avoid using pixels that depend on reconstruction values from unavailable reference area, due to deblocking and interpolation. In H.264/AVC interpolation is used to obtain sub-pixel values and is defined using a 6-tap FIR filter. Because of the 6-tap filtering step, some fractional pixels in the second macroblock row would need pixels from third macroblock row in the vertical sub-pixel interpolation process. To avoid this, the available reference area is restricted by cropping three pixels from the boundary. Second reason for cropping is related to the adaptive deblocking process. The deblocking filter in H.264/AVC modifies the reconstruction pixels close to the macroblock boundary by utilizing data from neighboring macroblocks. The number of pixels affected depends on many things such as macroblock mode, but at most three pixels from the macroblock boundary could be changed in deblocking. By cropping the available reference area by three pixels, those pixels that depend on macroblocks from the unavailable reference area are left out. For more details of deblocking filter and sub-pixel interpolation in the H.264/AVC standard, the reader is referred to [2][3]

In order to illustrate the parallel decoding operation, let's assume there are two decoders for each view running on separate platforms. Through proper signaling, the MVC decoder knows which macroblocks of view-0 and view-1 pictures could be independently decoded. Figure-3 illustrates the parallel decoding operation of macroblocks for the bitstream generated with the above example.

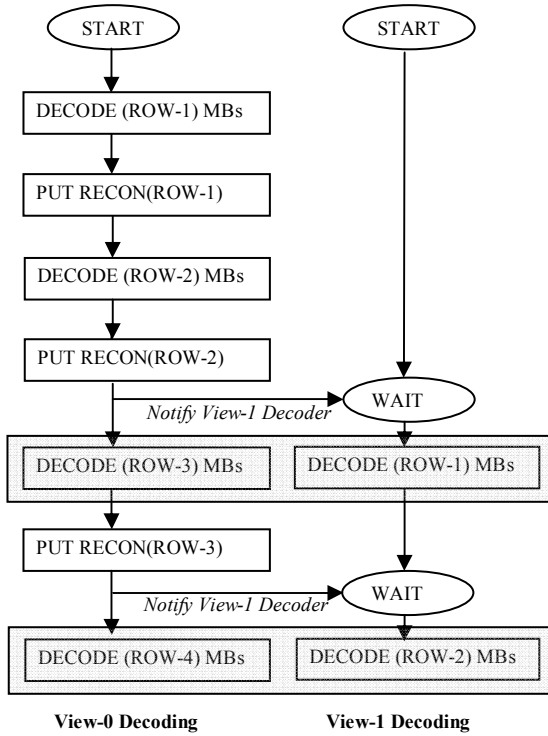


Figure 3. Sample Parallel Decoding Process for Two Views

As shown in Figure-3, decoding for both views start simultaneously, but decoding of first row of macroblocks in view-1 picture does not start before view-0 notifies the view-1 decoder. This notification is done after all the macroblocks in the first two rows in view-0 are decoded, and their reconstruction data are placed in the memory. This notification tells decoder of view-1 that, all data required to decode macroblocks of first row in view-1 are ready. This way, decoder of view-1 could start decoding of macroblocks of first row, while decoder of view-0 proceeds with decoding macroblocks in the third row and two decoders run in parallel (this parallel operation is indicated with a shaded rectangle in Figure-3). This parallel operation continues with two rows of delay between two views till the decoding of all the macroblocks are finished.

3. DESIGN CONSIDERATIONS

In the previous section, we presented encoder restrictions and details of the parallel decoding process. The total decoding time for a parallel implementation as illustrated above could be given as

$$\tau_0 + \text{delay_overhead} + \text{synch_overhead} \quad (1)$$

where, τ_0 is the time needed to decode the frame for the single-view case. *delay_overhead* is the time spent waiting for the initial macroblock(s) to be decoded, and *synch_overhead* is the time spent in the WAIT states. If the initial delay between views is large (*delay_overhead* is large), coding efficiency of the system would increase

because of the increased area available for referencing, with negative impact on parallelism. So, there exists a trade-off between coding efficiency and the parallelism of the multi-view system. Therefore, it is advantageous to allow an encoder to utilize this trade-off efficiently. For this purpose, we use the parameter *initial_delay* to adjust how many delays in macroblock rows exist between views. For the example illustrated in Figure 2, *initial_delay* is given as 2.

4. EXPERIMENTAL RESULTS

In order to test the performance of our scheme, we have followed the test conditions advised by the JVT standardization committee [4]. The proposed scheme was implemented on the Joint Multi-view Test Software (JMVM) [5]. As explained in Section 3, proposed scheme allows trade-offs between coding efficiency and the parallelism that could be adjusted using the parameter *initial_delay*. For this experiment, *initial_delay* is set to 2 (same as the example illustrated in Figure 3). For the simulcast case, where each view is coded independently, each decoder could run at the same speed as the single-view case to output pictures simultaneously, as there are no inter-view dependencies. For coding structures that utilize inter-view redundancy, it was assumed that decoder needs to run N times faster than a single view decoder (N being number of dependent views), to simultaneously output pictures from different views. For the bitstreams generated with the proposed method, there exists a lot less speed requirement. Table 1 compares the decoder speed requirement for the proposed method with different approaches for different sequences. The data for anchor in Table-1, utilizes inter-view prediction structure using hierarchical B as shown in [7].

Sequence	Anchor	Proposed
	Decoder Speed Requirement	Decoder Speed Requirement
Ballroom	5xsingle	1.3xsingle
Exit	5xsingle	1.3xsingle
Race1	5xsingle	1.3xsingle
Flamenco2	2xsingle	1.03xsingle
BreakDance	5xsingle	1.2xsingle
Rena	8xsingle	1.5xsingle
Uli	5xsingle	1.2xsingle
Akko&Kayo	6xsingle	1.4xsingle

Table 1. Decoding Speed Increase Compared to Single-View Decoder for Three Approaches

The coding results for the proposed scheme are compared against two approaches, simulcast and anchor. The anchor data is generated without any encoder restrictions and simultaneous output of views is possible only by running the decoder N times faster, N being the number of dependent views. Comparison against anchor is useful to see the penalty on the coding efficiency to enable parallel decoding of different views. We also compared our

method against simulcast approach. The encoding details for simulcast are exactly the same as the proposed scheme and anchor, only difference being the inter-view prediction is disabled for simulcast. Comparison against simulcast indicates the amount of improvement proposed scheme brings with similar parallelism requirements as simulcast. Table-2 presents the coding efficiency comparison of the proposed approach against anchor and simulcast. The differences in PSNR are calculated using the method described in [6]. It is shown in Table-2 that the proposed scheme achieves up-to 0.9 dB gain compared to simulcast, with very similar parallelism requirements. Compared to anchor approach, where the parallel operation is not possible, the penalty to achieve parallelism is negligible (less than 0.025 dB on average). Similar results could be seen in Figure 3, where the RD curve for the sequence Flamenco is plotted. As also seen in Figure 3, proposed scheme has significant gains compared to simulcast with similar requirements on parallelism.

Sequence	PSNR Results of Proposed Scheme	
	Δ_{dB} relative to anchor	Δ_{dB} relative to simulcast
Ballroom	-0.005 dB	+ 1.17 dB
Exit	-0.004 dB	+ 0.39 dB
Race1	-0.06 dB	+ 0.92 dB
Flamenco2	-0.07 dB	+ 1.18 dB
BreakDance	-0.002 dB	+ 0.41 dB
Rena	-0.003 dB	+ 1.55 dB
Uli	-0.01 dB	- 0.19 dB
Akko&Kayo	-0.08 dB	+ 1.44 dB

Table 2. Coding Efficiency of the Proposed Scheme relative to Anchor and Simulcast Approaches

5. CONCLUSIONS

Parallel encoding/decoding operation for the MVC standard is especially important for 3DTV systems that support head-motion parallax, where the receiving end needs to decode and display multiple views simultaneously. In this paper, we presented a coding structure for the upcoming MVC standard that enables parallel encoder/decoder operation for different views, even though there are dependencies between views. This is achieved by coding views with some constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Because of the systematic restriction of dependencies, a decoder could efficiently pipeline different views and achieve parallel operation. As the inter-view prediction is still used, significant gain is achieved compared to simulcast, while maintaining almost the same desirable parallelism characteristics. Experimental results show that, when compared to simulcast, proposed system achieves 0.9 dB gain on average with similar parallelism requirements. The proposed algorithm has also been proposed to MVC standardization [8], and has been

adopted in the form of Supplemental Enhancement Information (SEI) message.

It should be noted that, utilizing simpler prediction structures that only use inter-view prediction for frames where temporal prediction is not available (i.e. I-frames in temporal direction) also helps to improve the parallelism of the system significantly, without hurting the coding efficiency [7]. The proposed method could be used in conjunction with those simpler prediction structures, so that I frames in different views could also be decoded in parallel.

6. REFERENCES

- [1] "Requirements on Multi-view Video Coding v.5", document N7539, MPEG, Nice, France, Oct. 2005
- [2] P. List, A. Joch; J. Lainema.; G. Bjontegaard. M. Karczewicz, "Adaptive deblocking filter," IEEE Transactions on CSVT, vol.13, no.7pp. 614- 619, July 2003
- [3] M. Karczewicz, A. Hallapuro "Interpolation solution with low encoder memory requirements and low decoder complexity.", VCEG-N31, 24-27 Sept. 2001.
- [4] "Common Test Conditions for Multiview Video Coding", *JVT-U221*, Hangzhou, China, Oct. 2006
- [5] "Joint Multiview Video Model (JMVM) 1.0", *JVT-T209*, Klagenfurt, Austria, July 2006
- [6] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-curves" ITU-T SGI 6/Q.6 Doc. Doc. VCEG-M33, April 2001.
- [7] P. Merkle, A. Smolic, K. Mueller, T. Wiegand, "Comparative Study of MVC Structures", *JVT-V132*, Marrakech, Morocco, January 2007
- [8] K. Ugur, J. Lainema, H. Liu, Y.K. Wang, "Parallel Decoding Info SEI Message for MVC", *JVT-V098*, Marrakech, Morocco, January 2007

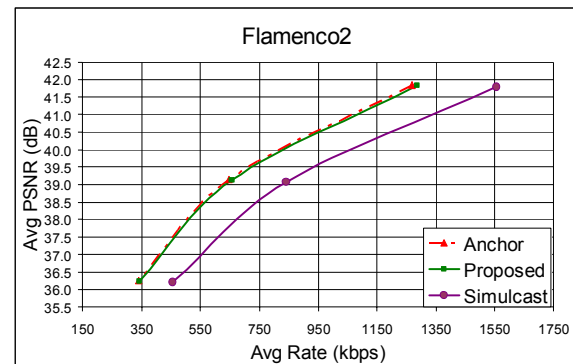


Figure 3. Comparing the Proposed Scheme with Anchor and Simulcast

- [P10] Y. Chen, Y.-K. Wang, Ugur K.; Hannuksela M.M.; Lainema J.; Gabbouj M., "The emerging MVC standard for 3D video services," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 786015, 13 pages, 2009.

© 2009 Y. Chen et. al.

Review Article

The Emerging MVC Standard for 3D Video Services

**Ying Chen,¹ Ye-Kui Wang,² Kemal Ugur,² Miska M. Hannuksela,²
Jani Lainema,² and Moncef Gabbouj¹**

¹ Department of Signal Processing, Tampere University of Technology, 33720 Tampere, Finland

² Nokia Research Center, Visiokatu 1, 33720 Tampere, Finland

Correspondence should be addressed to Ying Chen, ying.chen@tut.fi

Received 1 October 2007; Revised 7 February 2008; Accepted 5 March 2008

Recommended by Aljoscha Smolic

Multiview video has gained a wide interest recently. The huge amount of data needed to be processed by multiview applications is a heavy burden for both transmission and decoding. The joint video team has recently devoted part of its effort to extend the widely deployed H.264/AVC standard to handle multiview video coding (MVC). The MVC extension of H.264/AVC includes a number of new techniques for improved coding efficiency, reduced decoding complexity, and new functionalities for multiview operations. MVC takes advantage of some of the interfaces and transport mechanisms introduced for the scalable video coding (SVC) extension of H.264/AVC, but the system level integration of MVC is conceptually more challenging as the decoder output may contain more than one view and can consist of any combination of the views with any temporal level. The generation of all the output views also requires careful consideration and control of the available decoder resources. In this paper, multiview applications and solutions to support generic multiview as well as 3D services are introduced. The proposed solutions, which have been adopted to the draft MVC specification, cover a wide range of requirements for 3D video related to interface, transport of the MVC bitstreams, and MVC decoder resource management. The features that have been introduced in MVC to support these solutions include marking of reference pictures, supporting for efficient view switching, structuring of the bitstream, signalling of view scalability supplemental enhancement information (SEI) and parallel decoding SEI.

Copyright © 2009 Ying Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Three-dimensional video has gained significant interest recently. Furthermore, with the advances in acquisition and display technologies, 3D video is becoming a reality in consumer domain with different application opportunities. Given a certain maturity of capture and display technologies and with the help of multiview video coding (MVC) techniques, a number of different envisioned 3D video applications are getting feasible [1]. 3D video applications can be grouped under three categories: free-viewpoint video, 3D TV, and immersive teleconferencing. The requirements of these applications are quite different and each category has its own challenges to be addressed.

1.1. Application Scenarios. To illustrate these challenges, consider Figure 1, where the end-to-end architecture of different applications is shown. In this illustration, a multiview video is first captured and then encoded by a multiview

video coding (MVC) encoder. A server transmits the coded bitstream(s) to different clients with different capabilities, possibly through media gateways. The media gateway is an intelligent device, also referred to as a media-aware network element (MANE), which is in the signaling context and may manipulate the incoming video packets (rather than simply forward packets). At the final stage, coded video is decoded and rendered with different means according to the application scenario and capabilities of the receiver. To provide smoothly immersive experience when a user adjusting its viewing position, view synthesis [2, 3] may be required at the client to generate “virtual” views of a real-world scene. However, till now, this process is out of the scope of any existing coding standard.

In free-viewpoint video, the viewer can interactively choose his/her viewpoint in 3D space to observe a real-world scene from preferred perspectives [4]. It provides realistic impressions with interactivity, that is, the viewer can navigate freely in the scene within a certain range, and

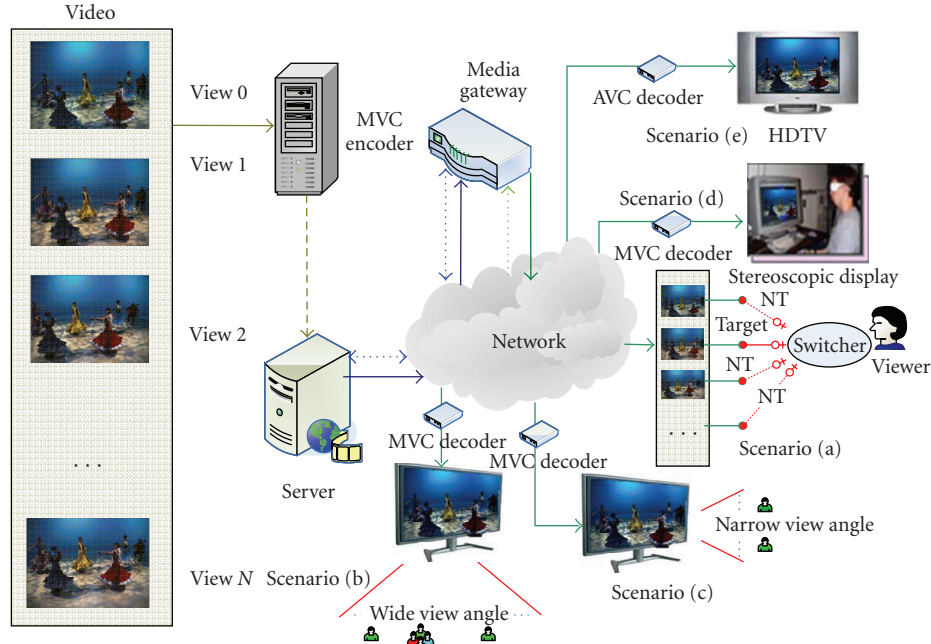


FIGURE 1: MVC system architecture.

analyze the 3D scene from different viewing angles. Such a video communication system has been reported in [5]. Unlike holography, which generates 3D representation and requires changing of the relative geometry position of a viewer to switch view point, this scenario is actually realized by switching between rendered view(s) using interface such as remote controller. In case the desired viewpoint is not available, interpolating a virtual view from other available views can be employed. Scenario (a), in Figure 1, illustrates this application, where there exist several candidate views for the viewer, and one of them is selected as the target view that is displayed (views that are not targeted and thus are not outputted are denoted as “NT” for simplicity in Figure 1). In this scenario, not all the candidate views are required to be decoded, thus the decoder can focus its resources only on decoding of the target view. For this purpose, the target view needs to be efficiently extracted from the bitstream and thus only the packets that are required for successfully decoding the desired views are transmitted. To enable navigation in a scene, important functionality to be achieved by the system is efficient switching between different views.

3D TV refers to the extension of traditional 2D TV displays-to-displays capable of 3D rendering. In this application, more than one view is decoded and displayed simultaneously [6]. A simple 3D TV application can be realized by stereoscopic video. Stereoscopic display can be achieved by using data glasses or other means. However, it is nicer for the user to get the 3D feeling directly through 3D appliances with added feature of rendering binocular depth cues [7], which can be realized by autostereoscopic displays. Advanced autostereoscopic displays can support head-motion parallax, by decoding and displaying multiple

views from different viewpoints simultaneously. That is, a viewer without extra facilities like data glasses can move to different geometry angle ranges, each of which contains typically two views rendered and shed by 3D displays. 3D TV displays are discussed in [8]. The viewer then can experience a slightly different scene by moving his/her head (for example, user may look what is behind a certain object in the scene). In this scenario, multiple views need to be decoded simultaneously; therefore parallel processing of different views is very important to realize this application. In addition, displaying multiple views is important also to realize wide viewing angle as shown in Figure 1(b). This scenario is also referred to as autostereoscopic 3D TV for multiple viewers [7]. However, if the decoder capability is limited or the transmission bandwidth decreases, the client at a receiver may simply decode and render just a subset of the views but still provide 3D display with a narrow view angle, as shown in Figure 1(c). The media gateway plays an important role to provide the adaptation functionality to support this use case. Such 3D TV broadcast or multicast system must then support flexible stream adaptation. Stream adaptation can be achieved at the server or media gateway, where only the sub-bitstreams, with less bandwidth and desired by the client are transmitted and other packets are discarded. After bitstream extraction, the sub-bitstream must be decodable for by MVC decoders.

Free-viewpoint video focuses on its functionality in free navigation while 3D TV emphasizes on 3D experience. In immersive teleconference, both interactivity and virtual reality may be preferred by the participants and thus free viewpoint or 3DTV style can be both supported. In the immersive teleconferencing, where there is interactivity among viewers, immersiveness can be achieved either in

a free-viewpoint video or 3D TV manner. So, the problems or requirements in free-viewpoint video or 3D TV are still existing and valid.

Typically, two mechanisms can make people perceptually feel immersed in a 3D environment. A typical technique, known as head-mounted display (HMD), needs a device worn on the head, as a helmet, which has a small display optic in front of each eye. This scenario is shown in Figure 1(d). Substitutions for HMD need to introduce head tracking [9] or gaze tracking [10] techniques, as shown in the solutions discussed in [7]. In 3D TV, however, each stereoscopic display can have effect on a certain small range of a view angle, thus, a viewer can change his/her viewing position when he/she is trying to view the scene in another viewpoint, as if there was a natural object.

For rendering of 3D TV content or view synthesis, depth information is needed. Depth-images storing the depth information as a monoscopic color video can be coded with existing coding standards, for example, as auxiliary pictures in H.264/AVC [11].

As the normal 2D TV or HDTV applications are still dominating the market, the MVC content will provide a way for those 2D decoders, for example, H.264/AVC decoder in the set-top box (STB) of digital TV to generate a display from an MVC bistream, as shown in Figure 1(e). This requires MVC bitstreams to be backward compatible, for example, to H.264/AVC.

1.2. Requirements of MVC. Due to the huge amount of data, particularly when the number of views to be decoded is large, transmission of multiview video applications relies heavily on the compression of the video captured by cameras. Therefore, efficient compression of multiview video contents is the primary challenge for realizing multiview video services.

A natural way to improve compression efficiency of multiview video content is to exploit the correlation between views, in addition to the use of inter prediction in monoview coding. This requires buffering of additional decoded pictures. When the number of views is large, the required memory buffer may be prohibitive. In order to make efficient implementations of MVC feasible, the codec design should include efficient memory management of decoded pictures.

The above challenges and requirements, among others [12], are the basis of the objectives for the emerging MVC standard, which is under development by the joint video team (JVT), and will become the multiview extension of H.264/AVC [11]. MVC standardization in the JVT started in July 2006 and is expected to be finalized in mid-2008. The most recent draft of MVC is available in [13].

In the MVC standard draft, redundancies among views are utilized to improve compression efficiency compared to independent coding of views. This is allowed with the so-called interview prediction, in which decoded pictures of other views can be used as reference pictures when coding a picture as long as they all share the same capturing or output time. View dependencies for interview prediction are defined for each coded video sequence.

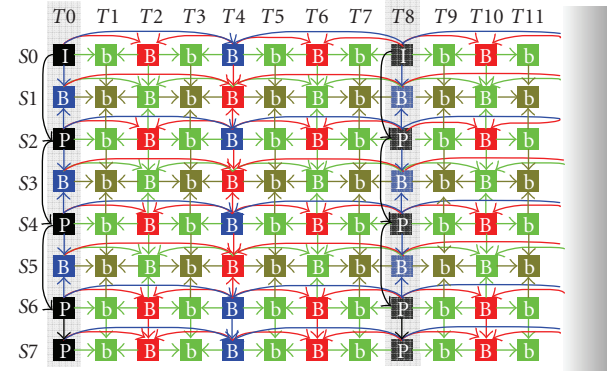


FIGURE 2: Typical MVC prediction structure.

With the exception of interview prediction, pictures of each view are coded with the tools supported by H.264/AVC. In particular, hierarchical temporal scalability was found to be efficient for multiview coding [14]. A typical prediction structure of MVC, utilizing both interview prediction and hierarchical temporal scalability, is shown in Figure 2. It is noted that the MVC standard provides a greater deal of flexibility than depicted in Figure 2 for arranging temporal or view prediction references [15].

Except the coding efficiency requirement, the following important aspects of the MVC requirements [12] for the design of the MVC standard are listed.

1.2.1. Scalabilities. View scalability and temporal scalability are considered in the MVC design for the adaptation of user preference, network bandwidth, and decoder complexity. View scalability is useful in the scenario shown in Figure 1(c), wherein some of the views are not transmitted and decoded.

1.2.2. Decoder Resource Consumption. In 3D TV scenarios, as shown in Figures 1(b) and 1(c), a number of views are to be decoded and displayed, an optimal decoder in terms of memory and complexity is of vital importance to make the real-time decoding of MVC bitstreams possible.

1.2.3. Parallel Processing. In the 3D TV scenarios, since multiple views need to be decoded simultaneously, parallel processing of different views is very important to realize this application and to reduce the computation time to achieve real-time decoding.

1.2.4. Random Access. Besides temporal random access, view random access is to be supported to enable accessing a frame in a given view with minimal decoding of frames in the view dimension. For example, free-viewpoint video described in Figure 1(a) needs advanced view random access functionality to support smooth navigation.

1.2.5. Robustness. When transmitted in a lossy channel, the MVC bitstream will have error resiliency capabilities. There are error resilient tools in H.264/AVC which can benefit the

MVC applications. Other techniques, which are designed only for MVC and discussed later, can also be utilized to improve error resilience of MVC bitstreams.

1.3. Contributions of this Paper. JVT has recently finalized the scalable extension of H.264/AVC, also known as scalable video coding (SVC) [16]. MVC shares some design principles with SVC, such as backward compatibility with H.264/AVC, temporal scalability, and network friendly adaptation, and many features in SVC have been reused in MVC.

However, new mechanisms are needed in MVC at least related to view scalability, interview prediction structure, coexisting of decoded pictures from multiple dimensions (i.e., both the temporal and view dimensions) in the decoded picture buffer, multiple representations in the display, and parallel decoding at the decoder.

These mechanisms cover the challenges and requirements, identified above, for 3D video services, except for the compression efficiency challenge. In this paper, we will describe how these mechanisms are realized in the existing draft MVC standard.

The main MVC features discussed in this paper include reference picture management to achieve optimal memory consumption at the decoder, time-first coding to support consistent system level design, SEI messages, and other features for view and scalability information provisioning, adaptation, random access, view switching, and reference picture list construction.

The rest of this paper is organized as follows. In Section 2, we discuss the MVC bitstream structure and the backward compatibility which is mentioned in Scenario (e). In Section 3, with a typical application scenario, we discuss how adaptation works when connectivity between server and client or decoder capacity varies. Then, view scalability information SEI message, which is designed to facilitate the storage, exaction, and adaptation of MVC bitstream, is reviewed. The features discussed in this section are of importance for efficient file composition, bitstream exaction, and stream adaptation in intermediate media gateways, which has been mentioned in Scenario (c). Random access and view switching functionalities are described in Section 4, which is desirable in Scenario (a). In Section 5, the decoded picture buffer management is discussed. This topic is crucial to enable a system to minimize the required memory for decoding MVC bitstreams. In Section 6, the parallel decoding SEI message, which is important for real-time MVC decoder solutions, is discussed. Other related issues are summarized in Section 7. Finally, Section 8 concludes the paper.

2. Structure of MVC Bitstreams

This section reviews the concept of network abstraction layer units (NAL units) and summarizes how the NAL unit types defined in H.264/AVC and SVC are reused for MVC. Syntax elements in the NAL unit header in the MVC context are also discussed.

In H.264/AVC, the coded video bits are organized into NAL units. NAL units can be categorized to video coding

layer (VCL) NAL units and non-VCL NAL units. The supported VCL NAL unit types and non-VCL NAL units in H.264/AVC are defined in [11] and well categorized in [17].

In MVC, there is a base view, which is coded independently and is compliant with H.264/AVC, this meets the requirement in Scenario (e) of the MVC system architecture, as shown in Figure 1. Consequently, coded picture information for the base view is included in the VCL NAL units specified in H.264/AVC. A new NAL unit type, called coded slice of MVC extension, is used for containing coded picture information for nonbase views. When an MVC bitstream containing NAL units of the new NAL unit type is fed to an H.264/AVC decoder, NAL units of any new NAL unit type can be ignored and the decoder only decodes the bitstream subset containing NAL units of the existing NAL unit types defined in H.264/AVC.

There are useful properties of the coded pictures in the H.264/AVC-compliant base view, such as temporal level, which are not indicated in the VCL NAL units of H.264/AVC. To indicate those properties for the base view-coded pictures, the prefix NAL unit, of another new NAL unit type, has been introduced. Note that prefix NAL unit is also specified in SVC. A prefix NAL unit precedes each H.264/AVC VCL NAL unit and contains its essential characteristics in multiview context. As H.264/AVC decoders ignore prefix NAL units, the backward compatibility to H.264/AVC is still maintained.

Non-VCL NAL units include parameter set NAL units and SEI NAL units among others. Parameter sets contain the sequence-level header information (in sequence parameter sets (SPS)) and the infrequently changing picture-level header information (in picture parameter sets (PPS)). With parameter sets, this infrequently changing information needs not to be repeated for each sequence or picture, hence coding efficiency is improved. Furthermore, the use of parameter sets enables out-of-band transmission of the important header information, avoiding the need of redundant transmissions for error resilience. In “out-of-band” transmission, parameter set NAL units are transmitted in a more different channel than the ones for transmission of other NAL units. More discussions on parameter sets can be found in [18].

In MVC, coded pictures from different views may use different sequence parameter sets. An SPS in MVC can contain the view dependency information for interview prediction. This enables signaling-aware media gateways to construct the view dependency tree. Therefore, each view can be mapped to the view dependency tree and view scalability can be fulfilled, without any extra signaling inside NAL unit headers [19].

The scalable nesting SEI message [19], which was also introduced in SVC with the same name, is set apart from other SEI messages in that it contains one or more ordinary SEI messages, but in addition it indicates the scope of views or temporal levels for which the messages apply. In doing so, it enables the reuse of the syntax of H.264/AVC SEI messages for a specific set of views and temporal levels.

Some of the other SEI messages specified in MVC are related to the indication of output views, available operation points, and information for parallel decoding.

In H.264/AVC, an NAL unit consists of a 1-byte header and an NAL unit payload of varying size. In MVC, this structure is retained except for prefix NAL units and MVC-coded slice NAL units, which consist of a 4-byte header and the NAL unit payload. New syntax elements in MVC NAL unit header include *priority_id*, *temporal_id*, *anchor_pic_flag*, *view_id*, *idr_flag* and *inter_view_flag*.

anchor_pic_flag indicates whether a picture is an anchor picture or nonanchor picture. Anchor pictures and all the pictures succeeding in output order (i.e., display order) can be correctly decoded without decoding of previous pictures in decoding order (i.e., bitstream order) and thus can be used as random access points. Anchor pictures and nonanchor pictures can have different dependencies, both of which are signaled in the sequence parameter set.

More discussions on anchor pictures will be given in Section 4. *idr_flag* is introduced in Section 4, *inter_view_flag* is discussed in Section 5, and the other new MVC NAL unit header fields are introduced in Section 3.

3. Extraction and Adaptation of MVC Bitstreams

MVC supports temporal scalability and view scalability. A portion of an MVC bitstream can correspond to an operation point that gives output representation for a certain frame rate and a number of target views. Data representing higher frame rate, views closer to the leaves of the dependency tree, or views that are not preferred by the client can be truncated during the stream bandwidth adaptation at the server or media gateway, or ignored at the decoder for complexity adaptation.

The bitstream structure defined in MVC is characterized by two syntax elements: *view_id* and *temporal_id*. The syntax element *view_id* indicates the identifier of each view. This indication in NAL unit header enables easy identification of NAL units at the decoder and quick access of the decoded views for display. The syntax element *temporal_id* indicates the temporal scalability hierarchy or, indirectly, the frame rate. An operation point including NAL units with a smaller maximum *temporal_id* value has a lower frame rate than an operation point with a larger maximum *temporal_id* value. Coded pictures with a higher *temporal_id* value typically depend on the coded pictures with lower *temporal_id* values within a view, but never depend on any coded picture with higher *temporal_id*.

The syntax elements *view_id* and *temporal_id* in the NAL unit header are important for both bitstream extraction and adaptation. Another important syntax element in the NAL unit header is *priority_id* [19], which is mainly used for the simple one-path bitstream adaptation process.

Whenever the operation point contains only a subset of the entire MVC bitstream, such as in Scenario (a) and Scenario (c) shown in Figure 1, a bitstream extraction process is then needed to exact the required NAL units from the entire bitstream. The bitstream extraction process should be a lightweight process without heavy parsing of

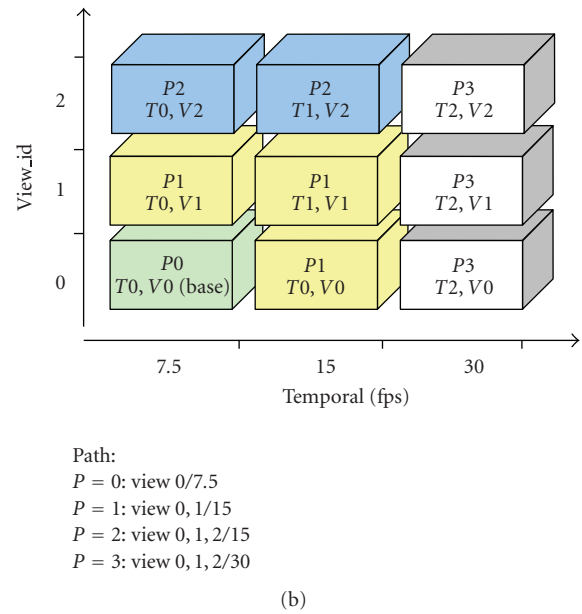
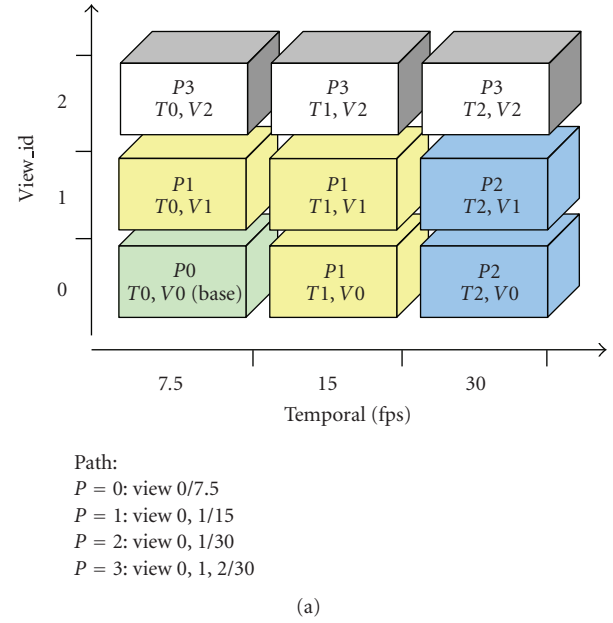


FIGURE 3: Assignment of *priority_id* for NAL units of a 3-view bitstream with two levels of temporal resolution. T : *temporal level*; V : *view identifier*; P : *priority identifier*. *Temporal level equal to 0 corresponds to 7.5 fps (frame per second), it equal to 1 corresponds to 15 fps, and it equal to 2 corresponds to 30 fps.*

the bitstream. For this purpose, the mapping between each operation point (identified by the combination of required *view_id* values and *temporal_id* values) and the required NAL units is specified as part of the view scalability information SEI message (VSSEI) [20]. After the operation point is agreed upon, the server can simply extract the required bitstream subset by discarding nonrequired NAL units by checking the *view_id* and *temporal_id* values in the fixed-length coded NAL unit headers.

Media gateways can perform single-path adaptation by simply discarding NAL units with `priority_id` greater than a certain value. The `priority_id` has no normative effect on the decoding process. The only constraint to `priority_id` values is that any bitstream subset extracted based on any value of `priority_id` must be a conforming MVC bitstream. It is the encoder responsibility to set `priority_id` values for the NAL units and the values can be rewritten, for example, when the preference of the decoder changes.

Figure 3 depicts two examples of `priority_id` assignments which yield two different adaptation paths for the same MVC bitstream that contains 3 views with 3 temporal levels. In Figure 3(a), the `priority_id` is assigned such that the 7.5 Hz base view is with `priority_id` equal to 0, and then frame rate of 15 Hz including both view 0 and view 2 is with `priority_id` equal to 1, and then higher frame rate is preferred to more views. In Figure 3(b), the first two steps are the same as in Figure 3(a), while in the last two steps, more views are preferred to higher frame rate.

Although a simple media gateway may perform stream adaptation exclusively based on `priority_id`, more intelligent implementations may jointly employ the values of `priority_id`, `view_id`, and `temporal_id`, in order to perform combined adaptation. For example, for the bitstream discussed in Figure 3, there can be two adaptation steps, the first step is to have NAL units with `temporal_id` equal to 1 (15 Hz) and `view_id` through 0 to 1; the second step is to increase frame rate directly to 30 Hz and include all the NAL units in view 2. Note that in this case, the NAL units corresponding to each adaptation step can have different values of `priority_id`, for example, when the `priority_id` assignment follows Figure 3(a).

An MVC bitstream may contain a large number of views (the `view_id` in the current MVC draft specification is of 10 bits). This makes the possible number of combinations of `view_id` values and `temporal_id` values huge. However, in practical applications, typically only limited combinations, that is, operation points, would be used. The VSSEI has been designed to be flexible to signal any subset of all the possible operation points. Beside the mapping of operation points and NAL units, the following information for each indicated operation point is also included in the VSSEI, either to enable the establishment of the communication session or more efficient bitstream extraction or adaptation.

Profile and level: This information describes the capacity a decoder requires to decode a bitstream. Profile and level can be signaled in the SPS. However, the total number of SPS is limited to a certain value in the bitstream and it may happen that for all the operation points, many of them share the same SPS, the level inside which is not accurate enough to describe the minimum required capacity of the decoders for different operation points. Therefore, profile and level are signaled in the VSSEI for each operation point.

Bit rate: Similar as profile and level, this information is needed in the session negotiation process for the server and the client to agree upon a certain operation point. This information is also useful in rate adaptation by MANEs. For example, to better adapt the bandwidth, it is necessary for intelligent media gateways to know the

bandwidth of a session when it switches to another operation point.

Operation point dependencies: In the VSSEI, each operation point is identified by the `view_id` values of the target views and the `temporal_id` values. The dependent views as well as the dependent pictures may be known from the active SPS which contains the view dependency information. However, within the view dependency, pictures may have more flexible relationship. For example, assume in a two-view bitstream with 30 fps, 4 temporal levels and according to the SPS MVC extension anchor pictures and nonanchor pictures in view 1 are, respectively, dependent on anchor and nonanchor pictures in view 0. And if we have two operation points (OPs), OP 0 has the pictures in view 0 with temporal level up to 3, that is, 15 fps and OP 1 has pictures with all the pictures in view 1, however, the pictures with the highest temporal level in view 1 do not really rely on interview pictures for reference. Then, OP 1 actually depends only on OP 0, which contains half of the pictures in view 0 and the highest temporal level pictures in view 0 can be neglected for transmission and decoding. However, with only the view dependency signaled in the SPS MVC extension, those pictures are still required to be transmitted and decoded. Thus, operation point dependency information included in the VSSEI would enable simply identification and discarding of the nonrequired NAL units that are not indicated by the view dependency information signaled in SPS.

In the following are some MVC stream adaptation examples in a broadcasting system (see Figure 1). Assume that the entire bitstream contains coded pictures of 8 views.

For Scenario (e), NAL units are filtered by the MANE so that only the NAL units that can be recognized by H.264/AVC decoders (by checking the NAL unit type) are fed to the STB of an HDTV.

For Scenario (d), an operation point containing, for example, only view 0 and view 1 is in use. The MANE controls the bitstream in a way that only allows the NAL units with `view_id` (by checking the `view_id` in the NAL unit header) equal to 0 or 1 to be sent to the client.

Depending on the bandwidth, a client with enough decoding capability for 3D TV may switch between Scenarios (b) and (c), wherein the sub-bitstream corresponding to Scenario (b) forms an operation point that contains only a subset of the views within a narrow view angle. The MANE filters out the views outside the view angle.

4. Random Access and View Switching

4.1. Random Access. Random access refers to starting decoding of a bitstream from a point other than the beginning. The support of random access is required for traditional trick play modes such as fast forward and fast backward. In streaming applications, random access is used to seek the desired playback position requested by the users. In broadcast and multicast applications, random access points are required to allow for newcomers to tune in or switching of program channels.

Random access with MVC for the above purposes is not much different from that with single-view coding, as

all the target views of an operation point are accessed simultaneously. The only difference is that there may be views dependent on by the target views; hence these dependent views need also to be accessed and decoded.

To access to a picture in a given view at a specified time, the decoder should first find the closest preceding temporal locations that are random access points to the specific target view and all the dependent views, collectively referred to as the required views. Then the decoder starts decoding the required views from a found location. In average, how many view pictures need to be decoded to access to a specific target picture is therefore proportional to the random access period (i.e., the length of the temporal dependency chain) and the number of dependent views (i.e., the length of the interview dependency chain).

Instantaneous decoding refresh (IDR) pictures are natural random access points. In an MVC bitstream, IDR pictures in the base view have NAL units of type 5. If the bitstream also contains NAL units that are unknown to plain H.264/AVC decoders, then the base view IDR picture NAL units are each preceded by a prefix NAL unit, which has `idr_flag` equal to 1. IDR pictures of nonbase views, also referred to as view-IDR (V-IDR) pictures in the draft MVC standard, all have `idr_flag` equal to 1. V-IDR pictures may rely on pictures from other views but only within the same access units though interview prediction [21].

An access unit contains all the NAL units pertaining to a certain time instance. According to the draft MVC standard, an IDR access unit is an access unit wherein the pictures of all the views are IDR pictures. Such an IDR access units provide random access support at the time instance to all the views. Note that the draft MVC standard allows for such access unit wherein pictures of some views are IDR pictures while pictures of other views are non-IDR pictures.

IDR pictures disallow any picture succeeding the IDR picture in decoding order (i.e., bitstream order) to be inter-predicted from earlier pictures in the same view. This leads to a reduced compression efficiency compared to the typical open GOP (group of pictures) coding structures such as the IBBP structure, where the B pictures after the I picture in decoding order precede the I picture in display order, and can use pictures before the I picture in decoding order for inter prediction. The I pictures in such open GOP coding structures are defined as anchor pictures in the draft MVC standard and are identified by the NAL unit header syntax element `anchor_pic_flag` equal to 1. Anchor pictures can therefore also be used as random access points, while application implementers must bear in mind that a few pictures after such random access points may not be correctly decoded when random access is carried out at these points. Actually, in this situation these pictures can be dropped from the bitstream sent to the user. Like V-IDR pictures, anchor pictures in nonbase views can also use interview prediction.

It is also possible to perform random access at non-intra pictures, for example, using gradual decoding refresh (GDR) based on the isolated regions technology [22]. In this case, the GDR random access points can be indicated by the recovery point SEI message as specified in H.264/AVC,

but included in the scalable nesting SEI message that tells to which views the semantics apply.

4.2. View Switching. View switching refers to changing the target view(s). The number of target view(s) may be one or more. In case the number of target view(s) change or any of the target view is changed from one view to another, a view switching occurs. View switching must happen at view-switching points, after which the new target view(s) can be correctly decoded. A typical application for view switching is free-viewpoint video, which has been shown in Scenario (a) of Figure 1.

All random access points can also be used as view switching points. There is another type of switching points that are not random access points. For example, if at picture X the target views can be switched to view subset C from view subset A but not from view subset B, then picture X is a view-switching point from view subset A to view subset C. This type of switching points can be realized by specifically setting the interview prediction relationship, or by using the SP/SI coding technology [23].

5. Decoded Picture Buffer Management

In this section, we first introduce the decoding order arrangement of coded view pictures, which is closely related to decoded picture buffer management. After that, we present an analysis of the buffer requirement for decoding of MVC bitstreams, which has been discussed in more details in [24]. Finally, reference picture management methods both inside a view and related to interview pictures are discussed.

5.1. Decoding-Order Arrangement. In H.264/AVC, the order how NAL units are placed inside the bitstream is referred to as the decoding order. In multiview video, where two dimensions, time and view, are involved, prescription of the decoding order gets more complicated.

Two fundamentally different decoding order arrangements, view-first coding and time-first coding, have been considered by the JVT. In view-first coding [25], within each group of pictures (GOP), pictures of each view are contiguous in decoding order, as shown in Figure 4, where the horizontal direction denotes time (each time instance is represented by T_m), and the vertical direction denotes view (each view is represented by S_n). Pictures of each view are grouped into GOPs, for example, pictures T1 to T8 for any view in Figure 2 form a GOP.

View-first coding causes a fundamental problem for storage of multiview video bitstreams in media container files based on ISO base media file format [26]. Coded pictures belonging to different views but with the same time instance are interleaved with pictures of other time instances in a bitstream, and thus cannot be in the same access unit. These different access units, when composed into a file according to the ISO base media file format, correspond to different samples. The ISO base media file format requires samples to be ordered in their decoding order. According to the ISO base media file format, the decoding time of a sample is an

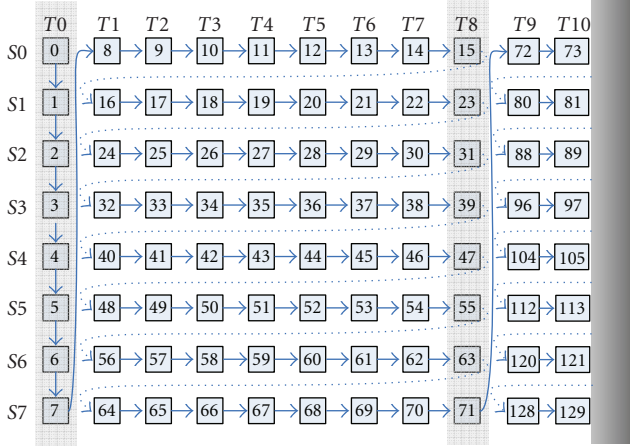


FIGURE 4: View-first coding.

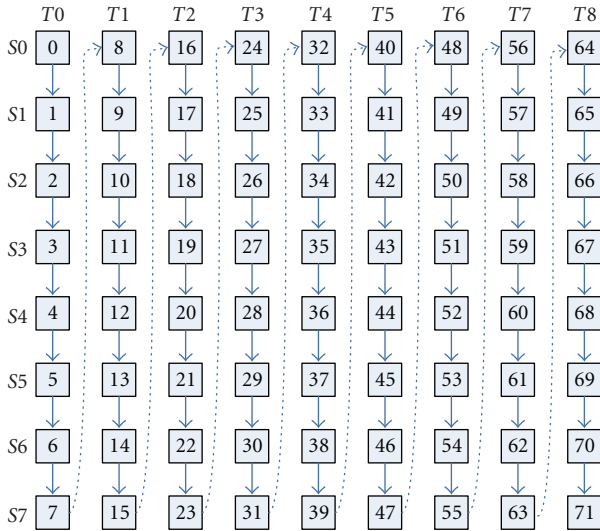


FIGURE 5: Time-first coding.

increasing function of sample number, and the composition time (also used as presentation time) of a sample is indicated as a nonnegative increment compared to its decoding time. Consequently, view-first coding would require a composition time offset proportional to the GOP size multiplied by the number of views, which would be perceived as significant initial buffering delay. Furthermore, possibility for parallel decoding would be hard to realize when view-first coded streams are included in files compliant with ISO base media file format, because the indicated decoding and composition times assumed single-processor operation.

To overcome the mentioned problems, time-first coding was introduced in MVC [27]. In time-first coding, pictures of any temporal location are contiguous in decoding order, as shown in Figure 5. In this case, we can define pictures of the same time instance but belonging to different views as one access unit. Note that the decoding order of access units may not be identical to the presentation order.

With time-first coding, an access unit contains NAL units continuous in decoding order. This definition is similar to the access unit definition in SVC. Therefore, many mechanisms designed in the SVC file format, such as extractors and aggregators, are useful for MVC too. Some design principle for MVC file format can be found in [28].

The following subsections on buffer requirement analysis and buffer management are all for time-first coding only.

5.2. Buffer Requirement Analysis. In MVC, pictures in the same time instance are assumed to be outputted simultaneously. Decoded pictures used for prediction or future output are buffered in the decoded picture buffer (DPB). To efficiently utilize the buffer memory, the DPB management processes have been specified, which include a storage process of decoded pictures into the DPB, a marking process of reference pictures, and an output and removal process of decoded pictures from the DPB.

Assume that we have a prediction structure similar to the one shown in Figure 2, where each GOP includes a number of views (nv) and in each view gl (GOP length) pictures.

The optimal DPB size, as discussed in [29], is $TL + 1$, where TL is the highest temporal level of all the pictures and $TL = \lceil \log_2(gl) \rceil$.

The DPB sizes for time-first coding in different scenarios are summarized in the following, while more details can be found in [24, 30].

5.2.1. DPB When Output is not Taken into Consideration. In time-first coding, the pictures in the same time instance will be stored in the DPB longer and each view preserves the hierarchical B coding structure. So there are two steps to reach the maximum DPB size for time first:

- (1) take the pictures in the same time instance as a whole and form a hierarchical B coding structure, the DPB size would then be $nv \cdot (TL + 1)$;
- (2) for the nonreference pictures in the highest temporal level, interview prediction requires them to be stored in the DPB.

These two steps are shown in Figure 6.

So, in the typical prediction structure, the maximum DPB size for time-first coding is $nv \cdot (TL + 1) + 2$.

In both results, there is a “2”, which actually means the maximum interview reference pictures in the typical prediction structure.

5.2.2. DPB When the Output is Taken into Consideration. When the output is considered, the case is even worse for view-first coding, especially for 3D TV application scenario, which requires the display of all the views. The reason is that, in view-first coding, all the pictures of the already coded view in a GOP must be kept in the buffer at least till the last view starts decoding.

For simplicity, we give the DPB buffer sizes for view-first coding and time-first coding in both 3D TV and free-viewpoint video scenarios without detailed analysis, which can be found in [24].

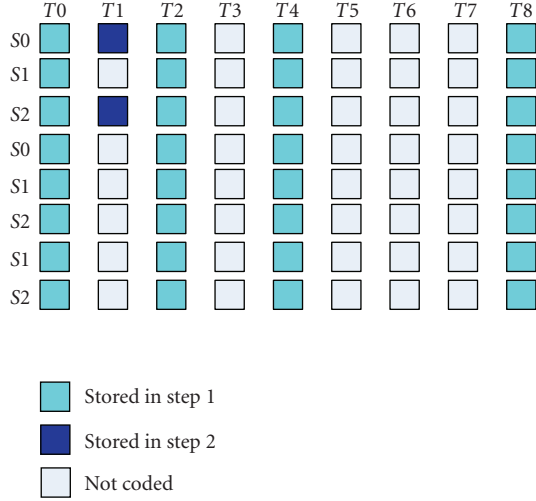


FIGURE 6: DPB status for time-first coding.

In 3D TV scenario, the total DPB sizes for time-first coding are $(nv - 1)gl + TL + 1$ and $nv(2 \cdot TL - \log_2 |TL - 1|)$, respectively.

In free-viewpoint video, the total DPB size for time-first coding is $(nv \cdot (TL + 1) + 3)/2 + TL - 1 - \log_2 |TL - 1|$.

Table 1 gives the example values for all the compared scenarios when the GOP length is 16 and number of views are 8. Time-first coding, as shown by the formula as well as the example values, requires less DPB size.

Note. Scenarios through (1) to (4) are the following scenarios, respectively: (1) DPB w/o output; (2) 3D TV DPB with output; (3) Free viewpoint video DPB with output, maximum; (4) Free viewpoint video DPB with output, average gl is 16 and nv is 8.

5.3. Buffer Management Inside a View. Because of the time-first coding structure, whether a picture is a reference picture or nonreference picture can be decided only by its temporal prediction structure. Because for any two pictures in a view, if picture A follows picture B, then in the whole bitstream, picture A also follows any picture with the same time instance as picture B. This is not the case in view-first coding, so it may require cross-view explicitly or implicit marking to make those pictures with the same time instance as B but with early decoding time as A as “unused for reference”.

So, all the memory management control operation commands, if present, are effective inside a view. And the sliding window also takes effect inside a view, which was proposed into JVT in the same time in [30–32].

5.4. Buffer Management for Interview Reference Pictures. In each time instance, if dependency exists, for one current decoding picture, there can be one or more interview reference pictures. Those interview reference pictures, although there are not used for temporal prediction within a view, are required to be somehow stored in the decoded picture buffer.

TABLE 1: Comparison examples between view-first and time-first when different scenarios are utilized.

	(1)	(2)	(3)	(4)
view-first	44	117	44	32
time-first	44	56	56	23.5

However, whether to store these pictures as “used for reference” or “unused for reference” is still an issue. In the AVC specification, if a picture is not used as a reference picture for others, it is with a `nal_ref_idc` value equal to 0 and is a nonreference picture. Those pictures, however, in MVC context can be used for interview reference picture, for example, the highest temporal level pictures in view 0 when view 1 is decoded.

If there are stored as a reference picture, when only base view sub-bitstream is decoded, it is definitely an extra memory burden for the H.264/AVC decoder and the encoder may need to design extra memory management control operation (MMCO) commands. So, in [33], we proposed that those pictures are not required to be stored as a reference picture. This solution solves the problem we mentioned above and another question arises: how would those pictures used only for interview prediction be managed to reach the optimal buffer management. One argument is if an interview picture is not used for temporal prediction and is a nonreference picture, it may be not available in the DPB.

Because of the time-first coding structure and the assumption that pictures are outputted at the same time, the concern mentioned above is solved.

So there is no extra marking process for those pictures if all views are required for output. If some views are not required for output, those pictures can be implicitly removed from the DPB earlier. The implicit removal is based on the view dependency defined in the MVC SPS extension [19, 34]. The implicit removal is defined in the hypothetical reference decoder (HRD) part of the MVC specification. The current HRD design of MVC focuses mostly on output conformance.

Although the interview prediction structure is in the scope of MVC SPS extensions, for each time instance, a picture can be used as interview picture or not based on real uses. For example, pictures in higher temporal levels may be more helpful for the efficiency while picture in lower temporal levels may be less helpful. The decoding of those pictures, if they are nonreference pictures and belong to the views that are not required for output, can be avoided. So an `inter_view_flag` was proposed by [35, 36] and was introduced into the MVC specification.

6. Parallel Coding of Multiple Views

One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views [11]. The parallel processing of different views is especially important for 3D broadcasting use cases, where the displays need to output many views simultaneously to support head-motion parallax. However, interview dependencies between

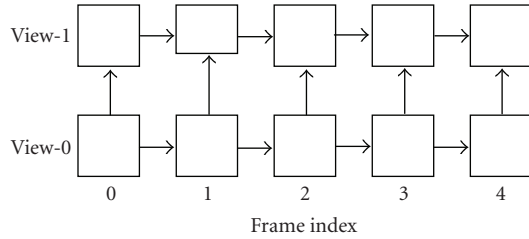


FIGURE 7: Sample prediction structure for two views.

pictures may impose serious parallelism issues to the video system, because two pictures at different views need to be decoded sequentially. Let us consider a 3DTV system displaying simultaneously two views, and views are coded with the coding structure as illustrated in Figure 7.

In order to decode a picture in view 1 at any temporal instant, the picture in view 0 at the same temporal instant will be decoded first. The only way to display two views at the same time is by having an MVC decoder running two times faster than a regular single-view decoder. Even though two independent decoders running on different platforms might be available, both decoders need to run twice faster than the single-view decoder because decoding has to be performed sequentially. The situation gets worse, with the increasing number of views that is supported by the 3D display. Currently there are commercial displays which can display 100 views simultaneously, and if all the views depend on each other, then the decoder must run 100 times faster, which is very challenging.

One way to increase the parallelism is to code each view independently. However, this kind of simulcast approach results in a significant penalty in coding efficiency as interview redundancies are not exploited at all. The draft MVC standard includes a more efficient method that allows parallel decoding/encoding operation of multiple views with high coding efficiency. This is achieved by utilizing the parallel decoding information SEI message that indicates that the views are encoded with systematic constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views [37, 38].

In order to describe how parallel processing is achieved using parallel decoding information SEI message, let us consider an example, where two pictures from view 1 and view 0 are going to be decoded. Assume view 1 picture references view 0 picture as illustrated in Figure 8 (for simplicity, the sizes of the frames are five macroblocks both horizontally and vertically). Parallel decoding information SEI message indicates the video is encoded in a way that macroblocks in view 1 picture could only use reconstruction values of macroblocks that belong to certain rows in view 0 picture. For example, the macroblocks in the first macroblock row of view 1 picture could only use reconstruction values from the first two macroblock rows in view 0 picture. In other words, the available reference area for the first macroblock row of view 1 picture constitutes only data from the first

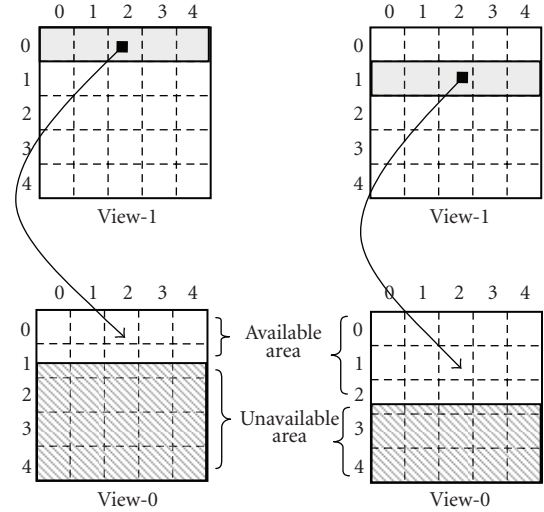


FIGURE 8: Systematic restriction of reference area.

two macroblock rows of view 0 picture (i.e., the motion vectors for the view 1 macroblocks are restricted). Similarly, the second macroblock row of view 1 picture only uses reconstruction values of the first three macroblock rows of the view 0 picture. This systematic restriction of reference area enables parallel decoding of first row of view 1 with any row below the second of view 0, as they are not referring each other.

In order to illustrate how this feature is used, let us assume an MVC decoder running on two processors (or processor cores) and decoding a bitstream containing two-views, where view 1 references view 0. Further assume that the bitstreams are coded with the restrictions as described above. The parallel decoding operation of these two views is illustrated in Figure 9, where processor P0 is decoding view 0 pictures and processor P1 is decoding view 1 pictures. The decoding operations for both views start simultaneously, but decoding of the first row of macroblocks in view 1 picture does not start before view 0 notifies the view 1 decoder. This notification is done after all the macroblocks in the first two macroblock rows in view 0 are decoded, and their reconstruction data are placed in the memory. This notification tells decoder of view 1 that all data required to decode first macroblock row in view 1 are ready. This way, the decoder of view 1 could start decoding the macroblocks of the first row, while the decoder of view 0 proceeds with decoding macroblocks in the third row and two decoders run in parallel. This parallel operation continues with two macroblock rows of delay between two views till the decoding of all the macroblocks is finished.

The benefit of using parallel decoding information SEI message is that significant coding gain is achieved over simulcast, while maintaining almost the same desirable parallelism characteristics. When compared to anchor method, where encoding happens without utilizing the SEI message and systematic restrictions, it is seen that parallel operation is achieved with almost no penalty on coding efficiency:

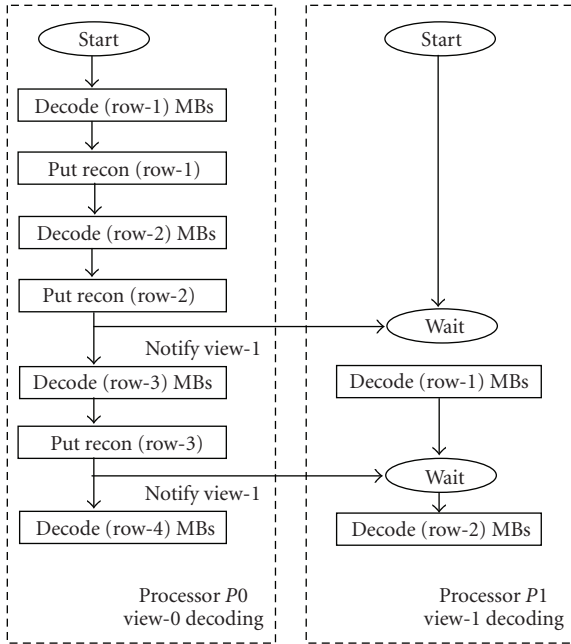


FIGURE 9: Sample parallel decoding process for two views.

maximum 0.08 and in average 0.03 dB loss for all the test sequences defined in the comment test condition for MVC [39]. Compared to simulcast, similar parallelism is achieved with 0.9 dB gain on coding efficiency as interview prediction is still utilized [37, 38].

In addition to using this SEI message, parallel processing could also be achieved by using simpler prediction structures. For example, consider the case where nonanchor pictures do not use interview prediction but only temporal prediction [40]. This approach achieves parallel operation as nonanchor pictures can be independently decoded without referencing other views. The parallelism of this structure could be further improved by using the parallel decoding information SEI message for anchor pictures.

It should be noted that parallel decoding information SEI message does not change the worst case complexity of MVC decoders. This means that the MVC decoders need to be designed to handle bitstreams where the encoding restrictions have not been applied, and parallel decoding information SEI message is not present. However, system standards such as DVB-H [41] can mandate the usage of this SEI message on their respective environments. This would ensure parallel operation for all the decoders operating on these services.

7. Other Related Techniques

Beside those discussed in earlier sections, the joint draft of MVC includes the following related techniques as summarized below.

7.1. Reference Picture List Construction. The reference picture list construction process can flexibly arrange temporal and

view prediction references. This provides not only potential coding efficiency gain but also error resilience, since reference picture section and redundant picture mechanisms can then be extended to the view dimension [15]. This strengthens the error robustness of the MVC bitstreams.

7.2. Active View Information SEI Message. The decoder may prefer to display a subset of the views encoded in an MVC bitstream. If this preference can be known by the decoder, then only the output views and the dependent views need to be decoded and stored in the DPB. The active view information SEI message was introduced to indicate the views that are to be output [42].

7.3. Multiview Scene Information and Multiview Acquisition Information SEI Messages. Two SEI messages related to acquisition and rendering were introduced in MVC, namely multiview scene information SEI message and multiview acquisition information SEI message, to signal camera parameters, which are helpful in view interpolation by a renderer [43].

8. Concluding Remarks

In this paper, we reviewed the key aspects of the system, transport interface, and decoder designs of MVC. We also introduced techniques crucial in meeting the requirements of typical 3D services and system architectures. These solutions, as adopted to the draft MVC standard, focus on two parts: features to facilitate storage and transport of MVC bitstreams and features to achieve minimum decoder resource consumption.

For the MVC system and transport interface, bandwidth adaptation, decoder capability adaptation, view random access, and view switching are the main concerns of the design. For the MVC decoder, minimizing the memory consumption and computational complexities are addressed.

The following key points of the MVC design are highlighted.

(i) MVC shared the same network abstraction layer (NAL) unit types designed in SVC, while differs a little in some specific syntax elements.

(ii) The base view of an MVC bitstream was designed to be H.264/AVC compatible in a way that it can be reconstructed by a standard H.264/AVC decoder. At the same time, backward-compatible extensions allow to utilize MVC-specific features with an MVC-compliant decoder.

(iii) New supplemental enhancement information (SEI) messages have been introduced to signal operation points as well as their dependency information. It is designed for adaptation and bitstream extraction. In addition, a mechanism has been specified that allows reusing all the original H.264/AVC SEI messages.

(iv) Time-first coding order was introduced to facilitate the file format design of MVC. This coding order is essential to achieve optimal buffer management at the decoder.

(v) Parallel decoding information SEI message was introduced to enable parallel encoder/decoder operation

for different views. This is especially important for 3D broadcast systems that support head-motion parallax, where the receiving end needs to decode and display multiple views simultaneously.

Acknowledgment

This work was supported in part by Nokia and the Academy of Finland, Finnish Centre of Excellence Program 2006–2011 under Project 213462.

References

- [1] A. Smolic, H. Kimata, and A. Vetro, "Development of MPEG standards for 3D and free viewpoint video," in *Three-Dimensional TV, Video, and Display IV*, vol. 6016 of *Proceedings of SPIE*, Boston, Mass, USA, October 2005.
- [2] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1020–1037, 2003.
- [3] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291 of *Proceedings of SPIE*, pp. 93–104, San Jose, Calif, USA, May 2004.
- [4] H. Kimata, M. Kitahara, K. Kamikura, Y. Yashima, T. Fujii, and M. Tanimoto, "System design of free viewpoint video communication," in *Proceedings of the 4th International Conference on Computer and Information Technology (CIT '04)*, pp. 52–59, Wuhan, China, September 2004.
- [5] A. Smolic and P. Kauff, "Interactive 3-D video representation and coding technologies," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 98–110, 2005.
- [6] A. Vetro, W. Matusik, H. Pfister, and J. Xin, "Coding approaches for end-to-end 3D TV systems," in *Proceedings of the 23rd Picture Coding Symposium (PCS '04)*, pp. 319–324, San Francisco, Calif, USA, December 2004.
- [7] C. Fehn, R. de la Barré, and S. Pastoor, "Interactive 3-DTV-concepts and key technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 524–538, 2006.
- [8] J. G. Eden, "Information display early in the 21st century: overview of selected emissive display technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 567–574, 2006.
- [9] B. Fröba and C. Küblbeck, "Face detection and tracking using edge orientation information," in *Visual Communications and Image Processing*, vol. 4310 of *Proceedings of SPIE*, pp. 583–594, San Jose, Calif, USA, January 2001.
- [10] L. Young and D. Sheena, "Methods & designs: survey of eye movement recording methods," *Behavior Research Methods & Instrumentation*, vol. 7, no. 5, pp. 397–429, 1975.
- [11] ITU-T Rec. H.264—ISO/IEC IS 14496-10, "Advanced video coding for generic audiovisual services," v3, 2005.
- [12] ISO/IEC JTC1/SC29/WG11, "Requirements on multi-view video coding v.5," N7539, Nice, France, October 2005.
- [13] "Joint draft 6.0 on multi-view video coding," JVT-Z209, Antalya, Turkey, January 2007.
- [14] D. Tian, M. M. Hannuksela, and M. Gabbouj, "Sub-sequence video coding for improved temporal scalability," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 6, pp. 6074–6077, Kobe, Japan, May 2005.
- [15] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "On MVC reference picture list construction," JVT-V043, Marrakech, Morocco, January 2007.
- [16] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien, Eds., "Joint draft 11 of SVC amendment," JVT-X201, Geneva, Switzerland, June–July 2007.
- [17] Y.-K. Wang, M. M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and transport interface of SVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1149–1163, 2007.
- [18] S. Wenger, "H.264/AVC over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.
- [19] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "Comments on MVC JD 2.0," JVT-W035, San Jose, Calif, USA, April 2007.
- [20] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "View scalability information SEI message for MVC," JVT-W037, San Jose, Calif, USA, April 2007.
- [21] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "MVC comments on JD 3.0," Geneva, Switzerland, July 2007.
- [22] M. M. Hannuksela, Y.-K. Wang, and M. Gabbouj, "Isolated regions in video coding," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 259–267, 2004.
- [23] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 637–644, 2003.
- [24] Y. Chen, Y.-K. Wang, and M. Gabbouj, "Buffer requirement analyses for multi-view video coding," in *Proceedings of the 26th Picture Coding Symposium (PCS '07)*, Lisbon, Portugal, November 2007.
- [25] "Joint multiview video model (JMVM) 1.0," JVT-T208, Klagenfurt, Austria, July 2006.
- [26] ISO/IEC IS 14496-2, "Information technology—coding of audio-visual objects—part 12: ISO base media file format," 2005.
- [27] Y.-K. Wang, Y. Chen, and M. M. Hannuksela, "Time-first coding for multi-view video coding," JVT-U104, Hangzhou, China, October 2006.
- [28] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "On MVC file format," M14634, Lausanne, Switzerland, July 2007. 1pt0.8pt
- [29] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 1929–1932, Toronto, Ontario, Canada, July 2006.
- [30] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "MVC reference picture management," JVT-U105, Hangzhou, China, October 2006.
- [31] A. Vetro and S. Yea, "Comments on MVC reference picture marking," JVT-U062, Hangzhou, China, October 2006.
- [32] P. Pandit, Y. Su, P. Yin, and C. Gomila, "Comments on high-level syntax for MVC," JVT-U026, Hangzhou, China, October 2006.
- [33] Y. Chen, Y.-K. Wang, M. M. Hannuksela, S. Liu, and H. Li, "On MVC reference picture marking," JVT-V044, Marrakech, Morocco, January 2007.
- [34] A. Vetro and S. Yea, "MVC clarification of marking process," JVT-V085, Marrakech, Morocco, January 2007.
- [35] Y.-K. Wang, Y. Chen, and M. M. Hannuksela, "Comments to JMVM 1.0," JVT-U103, Hangzhou, China, October 2006.
- [36] J. Choi, W. Shim, H. Song, and Y. Moon, "Inter-view prediction reference picture marking," JVT-W056, San Jose, Calif, USA, April 2007.

- [37] K. Ugur, H. Liu, J. Lainema, M. Gabbouj, and H. Li, "Parallel encoding-decoding operation for multi-view video coding with high coding efficiency," in *Proceedings of the Conference on True Vision, Capture, Transmission, and Display of 3D Video (3DTV '07)*, pp. 1–4, Kos Island, Greece, May 2007.
- [38] K. Ugur, J. Lainema, H. Liu, and Y.-K. Wang, "Parallel decoding info SEI message for MVC," JVT-V098, Marrakech, Morocco, January 2007.
- [39] "Common test conditions for multiview video coding," JVT-T207, Klagenfurt, Austria, July 2006.
- [40] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Comparative study of MVC structures," JVT-V132, Marrakech, Morocco, January 2007.
- [41] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "DVB-H: digital broadcast services to handheld devices," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 194–209, 2006.
- [42] Y.-K. Wang, M. M. Hannuksela, and Y. Chen, "MVC output related conformance," JVT-W036, San Jose, Calif, USA, April 2007.
- [43] A. Vetro, S. Yea, W. Matusik, H. Pfister, and M. Zwicker, "Anti-aliasing for 3D displays," JVT-W060, San Jose, Calif, USA, April 2007.

