Analysis, Coding, and Processing for High-Definition Videos

DONG, Jie

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Electronic Engineering

The Chinese University of Hong Kong
January 2010

題獻/Dedication

獻給我的丈夫劉雨

致謝/Acknowledgements

在博士論文即將完成之際，謹允許我借此一隅向幫助、鼓勵和支持過我的老師、朋友和家人致以深深的敬意和衷心的感謝。

首先要感謝的是我的博士論文導師顏慶義教授。顏教授學識淵博，洞察敏銳，給我很多啟發性的建議同時也讓我避免了不少彎路。我在攻讀博士學位期間取得的點滴進步都離不開顏教授的悉心指導。而顏教授嚴謹的治學態度和孜孜不倦的工作作風，更是讓我受益終身。

同時要感謝的是圖像與視訊處理實驗室的徐孔達教授、湛偉權教授和Thierry Blu教授。他們對各種學術問題的真知灼見以及從不同角度給我提出的建議，同樣給我很大幫助和啟發，讓我在自己博士研究課題之外學到了很多知識。

我也要感謝實驗室的伙伴們。他們是李宏亮、楊文姍、張帆、陳震中、李傑、劉雨、魏振宇、崔春暉、張茜、劉強、李鬆南、馬林、鄧啟霖、姚劍、麥振文、馮志強、孫德慶、張偉、歐陽萬裡、張任奇和趙叢。共同的研究興趣讓我們這些年輕人走到一起，從他們身上我學到了很多。和他們在一起科研和學習，讓我感到充實而快樂，給我留下了非常美好的回憶。

最後，讓我把最真摯的謝意獻給我的家人，他們是我的堅實後盾。父母的辛勤培養和殷殷期望，是我漫長求學生涯中不變的支柱。而我的丈夫劉雨，也是我的師兄，總是在我困難的時候幫助我，在我軟弱的時候鼓勵我，給我力量和勇氣去面對困難，迎接挑戰。

摘要

近年來，隨著信息技術的快速發展，高清視頻正逐步取代傳統的標清視頻，得到越來越廣泛的應用，包括數字電視廣播，高密度光盤存儲，視頻監控等等。然而，高清視頻相對于其他較低分辨率視頻，除了採樣頻率，還有哪些不同之處，以及如何利用這些不同之處來設計特別針對高清視頻的編碼和處理技術仍是一個亟待解決的研究課題。

首先，本論文定量研究了高清視頻的統計特性，包括空間相關性和功率譜密度。實驗結果顯示，高清視頻不僅有較高的空間相關性，其功率譜密度也較為特殊。主要的能量分部于垂直和水平方向，而其他方向，比如對角方向，能量很低。

其次，基于以上兩點實驗結論，本論文提出了兩項編碼技術。針對空間特性，本論文提出採用二維16階變換來壓縮空間域像素，以期有效地去除較高的空間相關性。為了滿足不同的性能和復雜度需求，本論文所提出的二維16階變換又包括非正交整數餘弦變換(NICT)和改進的整數餘弦變換(MICT)兩類。另一方面，針對功率譜密度特性，本論文提出了參數化內插濾波器(PIF)，並將其運用于分像素運動補償。參數化內插濾波器不但可以像其他自適應內插濾波器一樣隨著視頻信號統計特性的變化而不斷調整，而且僅用五個參數來表示濾波器，而非逐一表示各個濾波器系數，從而大大節省了傳輸濾波器所需的比特數也提高了濾波器精度。實驗結果顯示，本論文所提出的兩項編碼技術，相對于最新國際視頻壓縮標準H.264/AVC中所採用的相應技術，都顯著提高了對高清視頻的率失真性能。

最後，本論文研究了隔行高清視頻的特性，並提出了兩種實時去隔行技術以適用于不同的時延需求。這兩種去隔行技術是特別針對由H.264/AVC編碼的隔行視頻信號而設計的，所以碼流中豐富的語法元素值可以有效地用于估計局部的運動和紋理，從而使得去隔行算法作出相應有效調整。考慮到真實運動和紋理與估計值有可能會不一致，本論文又引入了對估計值可靠性的分析。實驗結果顯示，這兩項去隔行技術，相對于其他常用的實時去隔行技術，能提供較好的視覺質量，同時在不同平台上均能達到對1080i視頻實時去隔行的需求。

Abstract

Today, High-Definition (HD) videos become more and more popular with many applications. This thesis analyzes the characteristics of HD videos and develops the appropriate coding and processing techniques accordingly for hybrid video coding.

Firstly, the characteristics of HD videos are studied quantitatively. The results show that HD videos distinguish from other lower resolution videos by higher spatial correlation and special power spectral density (PSD), mainly distributed along the vertical and horizontal directions.

Secondly, two techniques for HD video coding are developed based on the aforementioned analysis results. To exploit the spatial property, 2D order-16 transforms are proposed to code the higher correlated signals more efficiently. Specially, two series of 2D order-16 integer transforms, named modified integer cosine transform (MICT) and non-orthogonal integer cosine transform (NICT), are studied and developed to provide different trade-offs between the performance and the complexity. Based on the property of special PSD, parametric interpolation filter (PIF) is proposed for motion-compensated prediction (MCP). Not only can PIF track the non-stationary statistics of video signals as the related work shows, but also it represents interpolation filters by parameters instead of individual coefficients, thus solving the conflict of the accuracy of coefficients and the size of side information. The experimental results show the proposed two coding techniques significantly outperform their equivalents in the state-of-the-art international video coding standards.

Thirdly, interlaced HD videos are studied, and to satisfy different delay constraints, two real-time de-interlacing algorithms are proposed specially for H.264 coded videos. They adapt to local activities, according to the syntax element (SE) values. Accuracy analysis is also introduced to deal with the disparity between the SE values and the real motions and textures. The de-interlacers provide better visual quality than the commonly used ones and can de-interlace $1080i$ sequences in real time on PCs.

Publications

Book Chapter

- **Jie Dong** and King Ngi Ngan, "Present and future video coding standards," in *Intelligent Multimedia Communication: Techniques and Applications*, Springer-Verlag Publisher, 2009.

Journal Papers

- **Jie Dong**, King Ngi Ngan, Chi Keung Fong and Wai Kuen Cham, "2D order-16 integer transforms for HD video coding," *IEEE Transactions on Circuits and System for Video Technology*, U.S.A., vol. 19, no. 10, pp. 1462-1474, Oct. 2009.

- **Jie Dong** and King Ngi Ngan, "Real-time de-interlacing for H.264 coded HD videos," *IEEE Transactions on Circuits and System for Video Technology*, U.S.A., under review.

- **Jie Dong** and King Ngi Ngan, "Parametric interpolation filter for high-definition video coding," *IEEE Transactions on Circuits and System for Video Technology*, U.S.A., under review.

- Cixun Zhang, Lu Yu, Jian Lou, Wai Kuen Cham, and **Jie Dong**, "The technique of prescaled integer transform: concept, design and applications," *IEEE Transactions on Circuits and System for Video Technology*, U.S.A., vol. 18, no. 1, pp. 84-97, Jan. 2008.

Conference Papers

- **Jie Dong** and King Ngi Ngan, "Parametric interpolation filter for motion compensated prediction," In *Proceedings of 2009 IEEE International Conference on Image Processing (ICIP2009)*, Cairo, Egypt, Nov. 7-11, 2009.

- **Jie Dong** and King Ngi Ngan, "An adaptive and parallel scheme for HD video de-interlacing," In *Proceedings of 2008 IEEE International Conference on Multimedia and Expo (ICME2008)*, Hannover, Germany, Jun. 23-26, 2008.
- **Jie Dong**, King Ngi Ngan, Chi Keung Fong and Wai Kuen Cham, "A universal approach to developing fast algorithm for simplified order-16 ICT," In *Proceedings of 2007 IEEE International Symposium on Circuits and Systems (ISCAS2007)*, New Orleans, USA, May 27-30, 2007.
- **Jie Dong** and King Ngi Ngan, "16×16 integer cosine transform for HD video coding," In *Proceedings of the Seventh IEEE Pacific Rim Conference on Multimedia (PCM2006)*, Hangzhou, China, Nov. 2-4, 2006.

Patents

- Wai Kuen Cham, Chi Keung Fong, **Jie Dong**, King Ngi Ngan, Hoi Ming Wong, Lu Wang, Yan Huo and Thomas Pun, "Method and device for order-16 integer transform from order-8 integer cosine transform," US Non-Provisional Patent Application, Dec. 2007.
- King Ngi Ngan, **Jie Dong** and Yan Huo, "Method and apparatus of de-interlacing video," US Non-Provisional Patent Application, May 2008.
- King Ngi Ngan and **Jie Dong**, "Parametric interpolation filter for motion-compensated prediction," US Provisional Patent Application, Oct. 2009.

Standard Contributions

- **Jie Dong**, King Ngi Ngan and Wai Kuen Cham, "Adaptive block-size transforms for AVS X-profile," Audio and Video Coding Standard Workgroup of China, AVS M1771, Mar. 2006.
- Wai Kuen Cham, Chi Keung Fong, **Jie Dong** and King Ngi Ngan (The Chinese University of Hong Kong), Hoi-Ming Wong, Lu Wang, Yan Huo and Thomas Pun (Hong Kong Applied Science and Technology Research Institute Company Limited), "Adaptive block-size transform for AVS-X," Audio and Video Coding Standard Workgroup of China, AVS M2182, Dec. 2007.

- Wai Kuen Cham, Chi Keung Fong, **Jie Dong** and King Ngi Ngan (The Chinese University of Hong Kong), Hoi-Ming Wong, Lu Wang, Yan Huo and Thomas Pun (Hong Kong Applied Science and Technology Research Institute Company Limited), "Adaptive block-size transform for AVS-X," Audio and Video Coding Standard Workgroup of China, AVS M2284, Mar. 2008.

- Xunan Mao, Yunfei Wang and Yun He (Tsinghua University), Wai Kuen Cham, Chi Keung Fong, **Jie Dong** and King Ngi Ngan (The Chinese University of Hong Kong), Hoi-Ming Wong, Lu Wang, Yan Huo, Thomas Pun and Carmen Cheng (Hong Kong Applied Science and Technology Research Institute Company Limited), "Adaptive block-size transform coding for AVS," Audio and Video Coding Standard Workgroup of China, AVS M2372, Jun. 2008.

- Chun Man Mak, King Ngi Ngan and **Jie Dong**, "Subjective and objective quality comparison of H.264/AVC and AVS at high fidelity," Audio and Video Coding Standard Workgroup of China, AVS M2530, Mar. 2009.

Nomenclature

Abbreviations

1-D	One-Dimensional
2-D	Two-Dimensional
3-D	Three-Dimensional
3DRS	3-D recursive searching
ABT	Adaptive Block-size Transform
AIF	Adaptive Interpolation Filter
AR	AutoRegressive
ASO	Arbitrary Slice Ordering
AVC	Advanced Video Coding
AVS	Audio and Video coding Standard of China
BS	Boundary Strength
CA-2D-VLC	Context-Adaptive 2-D Variable-Length Coding
CABAC	Context-Adaptive Binary Arithmetic Coding
CBAC	Context-Based Arithmetic Coding
CAVLC	Context-Adaptive Variable-Length Coding
CBP	Coded Block Pattern
CfP	Call for Proposals
CIF	Common Intermediate Format
CMC	Complementary Motion Compensation
DAIF	Directional Adaptive Interpolation Filter
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DPB	Decoded Picture Buffer
DTFT	Discrete-Time Fourier Transform
EOB	End Of Block
ELA	Edge-dependent Linear Averaging
EP	Enhanced Profile
FDIF	Fixed Directional Interpolation Filter
FLC	Fixed-Length Coding
FMO	Flexible Macroblock Ordering
fps	frame per second
FRExt	Fidelity Range Extensions
GMC	Global Motion Compensation

GOP	Group Of Pictures
GST	Generalized Sampling Theorem
HD	High-Definition
HP	High Profile
HVS	Human Visual System
ICT	Integer Cosine Transform
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IPP	Integrated Performance Primitives
i.i.d.	independent and identically-distributed
ISO	International Standardization Organization
ITU-T	International Telecommunications Union - Telecommunications sector
JM	Joint Model
JVT	Joint Video Team
KLT	Karhunen-Loeve Transform
LA	Line Averaging
LMMSE	Linear Minimum Mean Squared Error
LPS	Least Probable Symbol
LSB	Least Significant Bit
LTI	Linear Time-Invariant
MB	MacroBlock
MBAFF	MB Level Adaptive Frame/Field Coding
MC	Motion Compensation
MCP	Motion-Compensated Prediction
ME	Motion Estimation
MF	Median Filtering
MICT	Modified Integer Cosine Transform
MMCO	Memory Management Control Operations
MPEG	Moving Picture Experts Group
MPS	Most Probable Symbol
MSB	Most Significant Bit
MSE	Mean Squared Error
MV	Motion Vector
MVC	Multiview Video Coding
NICT	Non-orthogonal Integer Cosine Transform
NAL	Network Abstraction Layer
NALU	NAL Unit
PAFF	Picture Level Adaptive Frame/Field Coding
PCM	Pulse-Code Modulation

p.d.f.	probability distribution function
PIT	Pre-scaled Integer Transform
PIF	Parametric Interpolation Filter
PSD	Power Spectral Density
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
R-D	Rate-Distortion
RGB	Red Green Blue
ROI	Region Of Interests
SAD	Summation of Absolute Difference
SD	Standard-Definition
SE	Syntax Element
SEI	Supplemental Enhancement Information
SFP	Strong Filter Position
SIF	Source Input Format
SIFO	Switched Interpolation Filter with Offset
SSD	Summation of Squared Difference
SSS	Strict-Sense Stationary
SVC	Scalable Video Coding
TBP	Temporal Backward Projection
UHD	Ultra-High-Definition
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VCR	VideoCassette Recorder
VLC	Variable-Length Coding
VTF	Vertical-Temporal Filter
VO	Video Object
VOD	Video On Command
VOP	Video Object Plane
VQM	Visual Quality Metric
WHT	Walsh-Hadamard Transform
WSS	Wide-Sense Stationary

Notations

$\mathbf{x}$	A vector
$\mathbf{x}^T$	Transposition of $\mathbf{x}$
$\mathbf{x}^H$	Vector transposition followed by element conjugation
$\mathbf{x}(t, \zeta)$	A random process
$\mathbf{R}_{xx}$	Autocorrelation matrix of the random process $\mathbf{x}$
$\mathbf{C}_{xx}$	Autocovariance matrix of the random process $\mathbf{x}$
$\mathbf{r}_{xx}$	Correlation coefficient matrix of the random process $\mathbf{x}$
$\mathbf{S}_{xx}$	Power spectral density of the random process $\mathbf{x}$
$p(x)$	p.d.f. of random variable x
$h(n)$	Impulse response of a linear filter
$H(z)$	Transfer function of a linear system
λ	Lagrangian multiplier
D	Distortion
R	Rate
$J(\cdot)$	Lagrangian cost function
$E[\cdot]$	Expectation operation
$\det(\cdot)$	Determinant operation
$x\%y$	Modulo operation, which is x modulo y
$\lfloor x \rfloor$	The largest integer value less than or equal to x
$\lceil x \rceil$	The smallest integer value greater than or equal to x
$>>$	Right bit shift
$<<$	Left bit shift
$\psi(m, n, k)$	A pixel located at (m, n) of the k^{th} frame in a digital video signal
ψ_{max}	Peak (maximum) intensity value of the video signal ψ
A	Alphabet of a discrete source
σ_e^2	Variance (energy) of prediction error
∇f	The first derivative of function f

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A digital video comprises a sequence of two-dimensional (2-D) digital images representing a dynamic three-dimensional (3-D) scene on certain sampling grids and at regular time intervals. Digital video is more robust against channel disturbance and provides higher visual quality and more functionalities, compared with analog video.

In digital video communication, raw video signals, which consist of a huge amount of data, have to be compressed to a bit-rate constrained by the channel capacity. Hence, video compression, a.k.a. video coding, is mainly targeted at the best rate-distortion (R-D) performance, although also providing other functionalities, such as scalability, error resilience, random access, and random switching, in order to adapt to various applications and network/terminal conditions. The distortion herein is measured almost exclusively by pixel-wise fidelity, mean squared error (MSE) or peak signal-to-noise ratio (PSNR), because of mathematical tractability and feasibility of optimization. Recently, visual quality metric (VQM) based on the characteristics of human visual system (HVS) has received increasing attention, as it correlates better with perceived distortion. Nevertheless, how VQM influences the design of video coding scheme is still an open research issue. In this work, PSNR is used as the distortion measure.

Video coding systems may greatly differ from each other, as they consist of complicated functional modules and each module can be realized in different ways. For the interoperability in communication, video coding standards are developed to restrict various video coding systems, such that manufacturers can successfully interwork with each other by producing compliant encoders and decoders, and at the same time still have the freedom to develop competitive and innovative products. Generally speaking, a video coding standard provides a significant amount of innovations and represents the state-of-the-art technology at the time.

Video coding standards have been developing for about 20 years, driven by applications and advances in hardware capability. All the standards essentially follow the block-based hybrid coding scheme, where the designation "block-based" means each video frame is divided into non-overlapped blocks and each block is the unit the hybrid coding applies to. In hybrid coding, raw video signals are modeled as realizations of a random field, of which the statistical redundancy is removed by joint predictive coding and transform coding. Predictive coding reduces the temporal and spatial correlation of successive frames of a video sequence, whereas transform further exploits the spatial redundancy of the prediction error. The phrase "hybrid coding" will refer to the block-based hybrid coding scheme here and subsequently in this thesis for convenience. The development of hybrid coding is mainly reflected in the evolution of video coding standards. From H.261 [1] published in 1990 to the latest H.264/AVC [2], the coding efficiency has been improved more than five times.

Apart from the rapid development of video coding technology, the digital video representation has also been experiencing a tremendous growth in the past two decades. The typical video formats, such as common intermediate format (CIF) and quarter-CIF (QCIF) for mobile applications and standard-definition (SD) for broadcast and storage media, are gradually superseded by high-definition (HD) videos. Today, HD video formats have been adopted for broadcast all over the world and are also popular with many other applications, including high-density storage media, filmmaking, video gaming, and surveillance.

Generally speaking, HD video refers to any video of higher spatial resolution than SD video. Nonetheless, the HD video formats commonly used in communication are specified by [3]. Table 1.1 compares various formats of digital video source in different aspects. As indicated in the table, HD videos have higher spatial/temporal/color sampling rate, higher bit depth, and wider aspect ratio (16:9), and therefore provide richer details, more brilliant color, and more content. On the other hand, the raw data rate of HD video becomes two to six times higher than that of the SD videos, which makes HD video coding very challenging and also the key enabler of HD applications.

The state-of-the-art video coding technology, represented by H.264/AVC, is powerful enough to provide satisfactory quality of HD videos subject to the bit-rate constrained by today's network capacity. Nevertheless, the R-D performance can be further

Format	Full resolution	Color subsampling	Bit depth	Frame rate
QCIF [1]	176×144	4:2:0	8 bits	30p
CIF [1]	352×288	4:2:0	8 bits	30p
SIF [4]	352×240/288	4:2:0	8 bits	30p/25p
SD [5]	720×480/576	4:2:0/4:2:2/4:4:4	8-10 bits	30i/25i
HD [3]	1280×720	4:2:0/4:2:2/4:4:4	8-12 bits	24p/30p/60p
	1920×1080			24p/30p/30i
Digital cinema [6]	4096×2160 2048×1080	4:4:4	12 bits	24p/48p
UHD [7]	7680×4320 3840×2160	4:2:0/4:2:2/4:4:4	10 bits	60p

Table 1.1: *Formats of digital video source*

improved, since the technical developments in H.264/AVC are all originally designed as general coding tools for videos of various resolutions and do not fully exploit the special properties of HD videos. Actually, little research effort has been made to find the statistical difference between HD and other low resolution videos, although the characteristics of HD videos are intuitively expected to be unique.

In this work, the statistical characteristics of HD videos are studied quantitatively. The results show that HD videos distinguish from other lower resolution videos by higher spatial correlation and special power spectral density (PSD): the signal varies more slowly and the high frequency energy is mainly distributed along the vertical and horizontal directions. These findings are instructive to improve the coding tools in the latest hybrid coding scheme and make them more efficient for HD videos.

Based on the above analysis results, two techniques for HD video coding are developed. To exploit the spatial property, 2D order-16 transforms are proposed to code the higher correlated signals more efficiently. Specially, two series of 2D order-16 integer transforms, named modified integer cosine transform (MICT) and non-orthogonal integer cosine transform (NICT), are studied and developed to provide different trade-offs between the performance and the complexity. Based on the property of special PSD, parametric interpolation filter (PIF) is proposed for motion-compensated prediction (MCP). Not only can PIF track the non-stationary statistics of video signals as the related work can, but also it represents interpolation filters by parameters instead of individual coefficients, thus solving the conflict of the accuracy of coefficients and the

size of side information. The proposed two techniques are integrated into the state-of-the-art international video coding standard, H.264/AVC, and compared to their equivalents therein. The experimental results show the proposed coding techniques can significantly improve the overall R-D performance of H.264/AVC, i.e., on average about 10% bit-rate reduction at the same PSNR.

In the near future, video materials with even higher definitions, such as ultra-high-definition (UHD) (see Table 1.1), will be captured and distributed, but their bit-rates produced by the latest coding technology will go up faster than the increased capacity of the wireless or wired network infrastructure [8]. Therefore, high-performance video coding techniques specially for UHD videos are required. The study results in this work are quite instructive to imply the trend of coding strategy with increasing definition.

In addition to coding tools, processing technique like de-interlacing is also studied for interlaced HD videos. To satisfy different delay constraints, two real-time de-interlacing algorithms are proposed specially for H.264 coded videos. They adapt to local activities, according to the syntax element (SE) values. Accuracy analysis is also introduced to deal with the disparity between the SE values and the real motions and textures. The de-interlacers provide better visual quality than the commonly used ones and can de-interlace $1080i$ sequences in real time on PCs.

Organization of the Work

The work is organized as follows. Chapter 2 reviews the fundamentals of hybrid coding, which will be used later for HD video characterization and performance evaluation. A detailed survey on the history of hybrid coding standards is provided in Chapter 3, where the state-of-the-art video coding standard H.264/AVC, the platform on which the proposed coding and processing techniques are evaluated, is emphasized. In Chapter 4, the methodologies of analyzing the statistical characteristics of HD videos are described and the results are reported as well. Based on the analysis results in Chapter 4, two coding techniques for HD videos, 2-D order-16 integer transform and PIF, are proposed in Chapter 5 and Chapter 6, respectively. In Chapter 7, two real-time de-interlacing techniques are presented for H.264 coded interlaced HD videos, as post-processing tools. The thesis is concluded in Chapter 8, where the contributions of the Ph.D. work are summarized and the future research directions are pointed out as well.

Chapter 2

Fundamentals in Hybrid Video Coding

This chapter begins with a brief introduction of the digital video source, including formation and representation. Since in hybrid coding the digital video source is modeled as a random process, basic elements of probability theory and random process are then reviewed in Section 2.2. Three fundamental techniques for hybrid coding, predictive coding, transform coding, and entropy coding, are introduced in Sections 2.3, 2.4, and 2.5, respectively. Then, in Section 2.6, the concept of the hybrid coding is presented, which is essentially the core of all the video coding standards.

For further reading on probability theory and random process, the readers are referred to [9]. Fundamentals in hybrid video coding are given by Wang *et al.* [10].

2.1 Video Formation and Representation

Digital video is captured in pictures, which are actually the snapshots of a dynamic scene taken at regular time intervals. Each picture is represented by a rectangular array of pixels. In monochrome videos, the value of each pixel is the luminance intensity at the corresponding sampling position in the scene, whereas in color videos, each pixel carrying luminance and chrominance information is made of the combination of the intensities of primary colors, e.g., YCbCr, RGB, or YUV. In other words, each color picture is composed of three rectangular arrays for three color channels. The samples in each color channel have discrete and finite magnitudes, which typically have 256 levels, i.e., 8-bit bit depth.

The formats of digital video source widely used in communications are summarized in Table 1.1. In Sections 2.1.1 and 2.1.2, two characteristics of raw video related to color representation and raster scan are introduced, respectively. Section 2.1.3 introduces the quality measurement for distorted digital videos.

2.1.1 Color Space and Subsampling

Color Spaces

The red, green, and blue (RGB) primary is widely used in video capture and display systems. However, in video transmission, signals in the RGB space is usually converted into other color spaces with luminance/chrominance coordinates, such as YUV for PAL and SECAM TV systems and YIQ for NTSC TV systems, in order to reduce the bandwidth requirement and be compatible with monochrome video applications. The value of the Y component represents the brightness of a pixel, while the other two components bear the chrominance information. The specifications of the color space conversion from RGB to YUV and YIQ are given in (2.1) and (2.2), respectively,

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} \tilde{R} \\ \tilde{G} \\ \tilde{B} \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} \tilde{R} \\ \tilde{G} \\ \tilde{B} \end{bmatrix} \tag{2.2}$$

where $\tilde{R}$, $\tilde{G}$, and $\tilde{B}$ are normalized gamma-corrected values, so that $(\tilde{R}, \tilde{G}, \tilde{B})$ equal to (1,1,1) corresponds to white. The color space defined in BT.601 [5] is known as YCbCr, which is the scaled and shifted version of the analog YUV space. The transformation matrix for deriving YCbCr coordinate from the RGB coordinate is given as in (2.3),

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{2.3}$$

where $R = 255\tilde{R}$, $G = 255\tilde{G}$, and $B = 255\tilde{B}$ are the digital equivalents of the normalized RGB primary $\tilde{R}\tilde{G}\tilde{B}$. If the bit depth of the video source is 8-bit, the scaling and shifting operations guarantees that the resulting Y, Cb, and Cr components take values in the range of (0, 255). The color space transformation for 10-bit representation of the video source is also defined in [5].

Color Subsampling

Since HVS is less sensitive to color than to brightness, the chrominance components, Cb and Cr, may be subsampled without much degradation of the perceived video

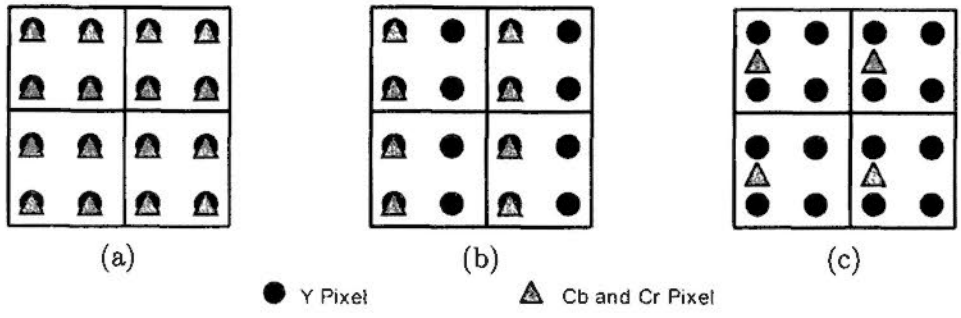


Figure 2.1: *Color subsampling formats. (a) 4:4:4. (b) 4:2:2. (c) 4:2:0.*

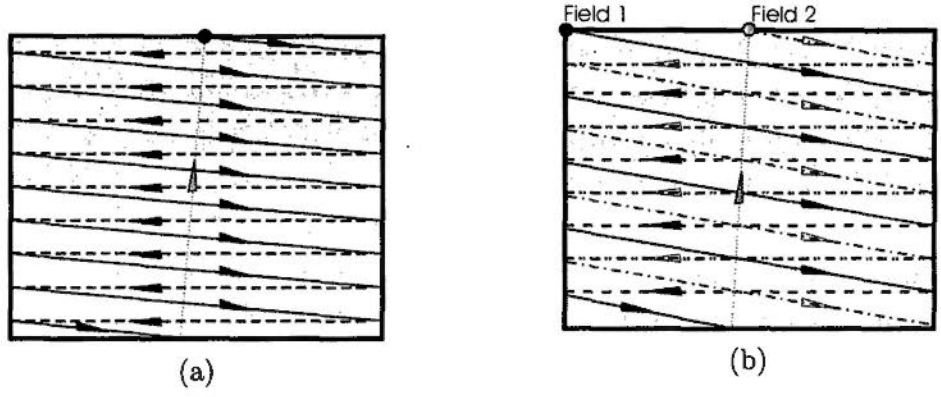


Figure 2.2: *Raster scan formats for analog video. (a) Progressive scan. (b) Interlaced Scan.*

quality. The color subsampling format is indicated by a triple of digits separated by colons. In 4:2:2 format, the horizontal sampling rate for the chrominance components reduces to half but the vertical sampling rate keeps unchanged. To further reduce the data rate, the sampling rate for chrominance components is reduced to half in both horizontal and vertical directions; the subsampling format is known as 4:2:0. However, for applications requiring very high quality, the chrominance components should have exactly the same sampling rate as for the luminance component, which is known as 4:4:4 format. The sampling grids of the luminance and chrominance samples for different color subsampling formats are shown in Fig. 2.1.

2.1.2 Progressive and Interlaced Scan

Progressive and interlaced scans are two types of raster scan first introduced in analog video capture, storage and display. With raster scan, analog video is represented by a continuous one-dimensional (1-D) waveform, of which the intensity values are captured

along contiguous scan lines over consecutive pictures. A picture refers to either a frame with progressive scan or a field with interlaced scan.

In progressive scan, the imaged region is scanned from left to right and from top to bottom. The resulting video signal consists of a sequence of frames captured at regular frame intervals Δ_t; each frame consists of a consecutive set of horizontal scan lines separated by a regular vertical spacing. Fig. 2.2 (a) depicts the scanning of a frame. The number of frames captured every second is known as frame rate.

In the interlaced scan, the horizontal lines in a frame are not scanned successively. Instead, each frame is scanned into two fields, each comprising half the number of lines in a frame. The time interval between two fields, i.e., the field interval, is equal to $\Delta_t/2$, while the line spacing in a field is twice of that desired for a frame. As shown in Fig. 2.2 (b), the scan lines in two successive fields are shifted one line spacing in a frame. Interlaced scan trades off the vertical resolution for an enhanced temporal resolution, given the total number of lines that can be recorded within a given time.

Following the terminology used in analog video, a digital video can also be progressive or interlaced, depending on the capturing process. In progressive video, a frame is captured at one time instance, whereas in interlaced video, a frame is formed by interleaving two fields, each having half vertical resolution and captured at a filed interval. The field containing the first line and following alternating lines in a frame is called the top field, and the field containing the second line and following alternating lines is called the bottom field. If an interlaced video contains fast moving objects with vertical edges, the serration artifacts will become annoying.

2.1.3 Quality Measurement

Visual quality is an important aspect of the performance of video processing techniques. Although HVS makes the ultimate judgment, objective criterion is necessary to design the algorithms and measure the distortion between the original and processed videos.

Most video processing systems are designed to minimize MSE between two video sequences ψ_1 and ψ_2, which is defined as

$$MSE = \sigma_e^2 = \frac{1}{N} \sum_k \sum_{m,n} (\psi_1(m,n,k) - \psi_2(m,n,k))^2 \,, \tag{2.4}$$

where N is the total number of pixels in either sequence. For a color video, MSE is

computed separately for each chrominance component.

Instead of MSE, PSNR in decibel (dB) is more often used as a distortion measure in video coding. PSNR is defined as

$$PSNR = 10 \log_{10} \frac{\psi_{max}^2}{\sigma_e^2} , \tag{2.5}$$

where ψ_{max} is the peak (maximum) intensity value of the video signal. For video signal with 8-bit bit depth, ψ_{max} equals 255. PSNR is more commonly used than MSE, because people tend to associate the visual quality with a certain range of PSNR. As a rule of thumb, for the luminance component, a PSNR over 40 dB indicates invisible distortion, between 30 and 40 dB usually means the distortion is visible but acceptable, between 20 and 30 dB is quite poor, and a PSNR lower than 20 dB is unacceptable.

It is well known that MSE or PSNR only reflects a rough classification of the visual quality of the distorted signal and does not precisely correlate with it. However, these measures have been used almost exclusively as objective distortion measures in image/video coding and restoration, partly because of their mathematical tractability and the lack of better alternatives. At the time of writing, the Video Coding Experts Group (VCEG), a video coding standardization body in ITU-T, is considering involving VQM as one of the distortion measures in its future standardization [11] and is seeking reliable VQM and convincing experimental methodology. Anyway, PSNR will be relevant in quality measurement for quite a long time. In this work, PSNR is used as the distortion measure.

2.2 Source Models

Video coding schemes can be classified into two categories: waveform-based coding and content-based coding, depending on how the video source is modeled. The former models a given video signal as the realization of a random process, whereas the latter, assuming that a group of pixels may represent a physical object, models a given video signal as the combination of moving objects and background. In this work, only waveform-based coding is discussed, because it is the context in which the proposed techniques are placed. Before moving into source coding techniques, the characterizations of general random process and image source are reviewed in Sections 2.2.1 and 2.2.2, respectively. The definitions and theorems introduced here are based primarily

on books [9; 10; 12; 13; 14]. Theories of 1-D and 2-D random processes can be found in [9] and [12], image statistical characterization is given in [13], and hybrid coding and its fundamental techniques are comprehensively introduced in [10].

2.2.1 Random Process

A random process $\mathbf{x}(t,\zeta)$ is a family of time functions depending on the argument ζ representing the set of all experimental outcomes. The argument t can be either a real number or an integer, making $\mathbf{x}(t,\zeta)$ a continuous-time process or a discrete-time process, respectively. On the other hand, considering the value of $\mathbf{x}(t,\zeta)$, one can classify $\mathbf{x}(t,\zeta)$ as a discrete-state process if its values are countable, or otherwise a continuous-state process. In this thesis, only discrete-time discrete-state process, denoted as $\mathbf{x}[n]$ (ζ is omitted for convenience), is discussed, because it appropriately models digital video signals.

Each realization of a random process, denoted as $x[n]$, is considered to be a member of an ensemble of digital signals characterized by a set of probability distributions. The complete joint probability distribution of $x[n]$ is described in (2.6) for all N samples,

$$f(x_1, x_2, \ldots, x_N; n_1, n_2, \ldots, n_N) \tag{2.6}$$

where x_i, $i = 1, \cdots, N$, is the random variable $\mathbf{x}[n_i]$. In general, complete and high-order joint probability distributions of a random process are unknown, nor are they easily modeled. Nevertheless, the first-order probability distribution $f(x;n)$ can sometimes be modeled successfully based on the physics of the process or the histogram measurements. Conditional probability distributions of a random process, evaluated at sample n_1 given knowledge of the sample value at n_2 as shown in (2.7), are also useful in characterizing a random process.

$$f(x_1; n_1 | x_2; n_2) = \frac{f(x_1, x_2; n_1, n_2)}{f(x_2; n_2)} \tag{2.7}$$

Correlation and Covariance

Although realizations of a random process are not deterministic, their average properties are often considered deterministic, including correlation and covariance.

The autocorrelation of $\mathbf{x}[n]$, denoted as $R_{xx}[n_1, n_2]$, is the expectation of the product

$\mathbf{x}[n_1]\mathbf{x}[n_2]$, as given in (2.8).

$$R_{xx}[n_1, n_2] = E\left[\mathbf{x}[n_1]\mathbf{x}[n_2]\right] \tag{2.8}$$

The autocovariance of $\mathbf{x}(n)$, denoted as $C_{xx}[n_1, n_2]$, is the covariance of the random variables $\mathbf{x}[n_1]$ and $\mathbf{x}[n_2]$, as given in (2.9).

$$C_{xx}[n_1, n_2] = R_{xx}[n_1, n_2] - \eta_x[n_1]\eta_x[n_2] \tag{2.9}$$

where $\eta_x[n]$ is the expectation of $\mathbf{x}[n]$. Obviously, for a zero-mean random process, R_{xx} is equal to C_{xx}. The correlation coefficient of $\mathbf{x}[n]$, denoted as r_{xx}, is by definition the normalized C_{xx}, as given in (2.10),

$$r_{xx}[n_1, n_2] = \frac{C_{xx}[n_1, n_2]}{\sqrt{C_{xx}[n_1, n_1]C_{xx}[n_2, n_2]}} = \frac{C_{xx}[n_1, n_2]}{\sigma_{x1}\sigma_{x2}} \tag{2.10}$$

where σ_{x1}^2 and σ_{x2}^2 in (2.10) are the variances of $\mathbf{x}[n_1]$ and $\mathbf{x}[n_2]$, respectively. r_{xx} satisfies (2.11) as below.

$$|r_{xx}[n_1, n_2]| \leq 1, \qquad r_{xx}[n, n] = 1 \tag{2.11}$$

Similarly, given two random processes $\mathbf{x}[n]$ and $\mathbf{y}[n]$, their cross-covariance is shown in (2.12).

$$C_{xy}[n_1, n_2] = R_{xy}[n_1, n_2] - \eta_x[n_1]\eta_y[n_2] \tag{2.12}$$

Stationary Processes

Strict Sense Stationary A random process is strict-sense stationary (SSS), if its statistical properties are invariant to a shift of the origin: the processes $\mathbf{x}[n]$ and $\mathbf{x}[n + c]$ have the same statistics for any c. Hence, the n^{th}-order probability distribution of an SSS process satisfies (2.13) for any c, the first-order probability distribution is independent of n, as shown in (2.14), and the second-order probability distribution only depends on τ, the temporal distance between n_1 and n_2 (see (2.15)).

$$f(x_1, \ldots, x_k; n_1, \ldots, n_k) = f(x_1, \ldots, x_k; n_1 + c, \ldots, n_k + c) \tag{2.13}$$

$$f(x; n) = f(x) \tag{2.14}$$

$$f(x_1, x_2; n_1, n_2) = f(x_1, x_2; \tau), \quad \tau = n_1 - n_2 \tag{2.15}$$

Wide Sense Stationary In contract to SSS, wide-sense stationarity refers only to the first- and second-order statistics. A random process $\mathbf{x}[n]$ is wide-sense stationary (WSS) if its expectation is constant and its autocorrelation depends only on τ equal to $n_1 - n_2$, as given in (2.16) and (2.17), respectively.

$$E[\mathbf{x}[n]] = \eta_x \tag{2.16}$$

$$R_{xx}[n_1, n_2] = E[\mathbf{x}[n_1]\mathbf{x}[n_2]] = R(\tau), \quad \tau = n_1 - n_2 \tag{2.17}$$

It can be shown that $R(\tau) = R(-\tau) = R(|\tau|)$ and $|R(\tau)| \leq R(0)$ for all τ. Substituting (2.16) and (2.17) into (2.9), one can find the autocovariance of a WSS process also depends only on τ, as shown in (2.18),

$$C(\tau) = R(\tau) - \eta_x^2 \tag{2.18}$$

The variance of the random process, denoted as σ_x^2, is also a constant equal to $C(0)$. For a WSS random process, the correlation coefficient can be derived from (2.10) and is usually denoted as ρ_τ in (2.19).

$$\rho_\tau = r(\tau) = C(\tau)/\sigma_x^2 \tag{2.19}$$

Correlation coefficient ρ_1, representing the correlation of adjacent samples, is an important criterion to measure the redundancy among samples, which increases with $|\rho_1|$. A value of ρ_1 equal to 1 implies perfect positive correlation, e.g., in a constant random process; a value of ρ_1 equal to -1 implies perfect negative correlation, e.g., in a random process formed by alternating -1 and 1; a value of ρ_1 equal to 0 implies the samples are uncorrelated at all.

At this point, it is more convenient to introduce the matrix terminology. The random process with length N can be considered as an N-dimensional random column vector, i.e., $\mathbf{x} = [\mathbf{x}[0], \mathbf{x}[1], \cdots, \mathbf{x}[N-1]]^T$, where T means the vector transposition. Then the autocorrelation matrix and the autocovariance matrix can be formulated by (2.20) and (2.21), respectively,

$$\mathbf{R}_{xx} = E[\mathbf{x}\mathbf{x}^T] \tag{2.20}$$

$$\mathbf{C}_{xx} = \boldsymbol{R}_{xx} - \boldsymbol{\eta}_x \boldsymbol{\eta}_x^T \tag{2.21}$$

where $\boldsymbol{\eta}_x$ equal to $E[\boldsymbol{x}]$ is the mean-value column vector. The autocovariance matrix is expressed by (2.22)

$$\mathbf{C}_{xx}=\begin{bmatrix} C(0) & C(1) & C(2) & \cdots & C(N-1) \\ C(1) & C(0) & C(1) & \cdots & C(N-2) \\ C(2) & C(1) & C(0) & \cdots & C(N-3) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ C(N-1) & \cdots & \cdots & \cdots & C(0) \end{bmatrix} = \sigma_x^2 \begin{bmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{N-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{N-2} \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{N-1} & \cdots & \cdots & \cdots & 1 \end{bmatrix} = \sigma_x^2 \mathbf{r}_{xx} \tag{2.22}$$

Autoregressive Processes

An autoregressive (AR) process is generated by passing the white-noise process $\mathbf{z}[n]$ with PSD $S_{zz}(\omega) = \sigma_z^2$ through a filter with the transfer function given in (2.23).

$$B(z) = \frac{1}{1 - \sum_{j=1}^{N} b_j e^{-j\omega}} \tag{2.23}$$

The equivalent difference equation generating the AR process $\mathbf{x}[n]$ is shown in (2.24),

$$\mathbf{x}[n] = \mathbf{z}[n] + \sum_{j=1}^{N} b_j \mathbf{x}[n-j] \tag{2.24}$$

where $\mathbf{x}[n]$ is called an AR(N) process or an N^{th} order Markov process. It can be shown that an AR process is WSS, if b_j, $j = 1, \cdots, N$, is less than 1. Multiplying $\mathbf{x}[n-k]$, $k \neq 0$, and taking expectations on both sides of (2.24), one can find the autocorrelation of $\mathbf{x}[n]$ is calculated by (2.25).

$$R(k) = \sum_{j=1}^{N} b_j R(k-j), \qquad k > 0 \tag{2.25}$$

Note that $E[\mathbf{z}[n]\mathbf{x}[n-k]]$ equals 0 for any $k > 0$, because $\mathbf{z}[n]$ is by definition uncorrelated with the past outputs. If k equals 0, $R(0)$ is calculated as below.

$$R(0) = \sum_{j=1}^{N} b_j R(j) + \sigma_z^2 \tag{2.26}$$

First-Order Markov (AR(1)) Process According to the definition of AR(N) process in (2.24), the AR(1) process with b_1 equal to ρ is generated by (2.27),

$$\mathbf{x}[n] = \mathbf{z}[n] + \rho\mathbf{x}[n-1] = \mathbf{z}[n] + \sum_{j=1}^{\infty} \rho^j \mathbf{z}[n-j] \tag{2.27}$$

i.e., by passing white-noise process $\mathbf{z}$ with variance σ_z^2 through a filter with impulse response and frequency response in (2.28) and (2.29), respectively.

$$h(n) = \begin{cases} \rho^n & \text{for } n \geq 0 \\ 0 & \text{for } n < 0 \end{cases} \tag{2.28}$$

$$H(\omega) = (1 - \rho e^{-j\omega})^{-1} \tag{2.29}$$

Substituting ρ into (2.25), one can obtain the variance and autocorrelation of $\mathbf{x}[n]$ as in (2.30) and (2.31), respectively.

$$\sigma_x^2 = R(0) = \frac{\sigma_z^2}{1-\rho^2} \tag{2.30}$$

$$R(k) = \sigma_x^2 \rho^{|k|} \tag{2.31}$$

According to (2.30) and (2.31), an AR(1) process is WSS, because it is zero-mean and its autocorrelation $R(k)$ depends on k only (see (2.31)). However, the filter $h(n)$ in (2.28) must be stable; therefore $|\rho|$ less than 1 is required.

In the matrix terminology, the autocorrelation matrix, derived from (2.31), is shown in (2.32) with the elements $\mathbf{R}_{xx}[k,l]$ equal to $\sigma_x^2 \rho^{|k-l|}$. $\mathbf{R}_{xx}$ is a symmetric Toeplitz matrix, in which all the elements on the same diagonal line are identical and the corresponding upper and lower triangle lines are the same.

$$\mathbf{R}_{xx} = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{L-1} \\ \rho & 1 & \rho & \cdots & \rho^{L-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{L-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho^{L-1} & \cdots & \cdots & \cdots & 1 \end{bmatrix} \tag{2.32}$$

Power Spectral Density

A random process $\mathbf{x}[n]$ in the frequency domain is represented by power spectral density (PSD), denoted as $S_{xx}(\omega)$, instead of the discrete-time Fourier transform (DTFT) of any realization of $\mathbf{x}[n]$, because of the nondeterministic nature of random process. Suppose $\mathbf{x}[n]$ is WSS with the autocorrelation $R(\tau)$. The PSD and $R(\tau)$ constitutes a Fourier transform pair, of which the forward and inverse transforms are shown in (2.33) and (2.34), respectively.

$$S_{xx}(\omega) = \sum_{\tau=-\infty}^{\infty} R(\tau)e^{-j\omega\tau} \tag{2.33}$$

$$R(\tau) = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_{xx}(\omega)e^{j\omega\tau}d\omega \tag{2.34}$$

When $R(\tau)$ is unknown or unreliable, PSD can be estimated using the periodogram of the random process, as given in (2.35),

$$\hat{S}_{xx}(\omega) = \frac{1}{N}|X(\omega)|^2 = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \tag{2.35}$$

where $\hat{S}_{xx}$ means an estimate of S_{xx} and $X(\omega)$ is the DTFT of an available realization of $\mathbf{x}[n]$. A better estimate can be obtained, if the N-sequence is divided into M non-overlapped subsequences, of which the periodograms are averaged.

Considering a random process $\mathbf{z}[n]$ with the PSD S_{zz} is input to a linear time-invariant (LTI) and stable system with the transfer function $H(\omega)$, one can calculate the PSD of the output process $\mathbf{x}[n]$ by (2.36),

$$S_{xx}(\omega) = S_{zz}(\omega)|H(\omega)|^2 \tag{2.36}$$

Substituting the transfer function as in (2.29) to (2.36), one obtains the PSD of the AR(1) process, given in (2.37).

$$S_{xx}(\omega) = \sigma_z^2|H(\omega)|^2 = \frac{1-\rho^2}{1+\rho^2-2\rho\cos\omega}\sigma_x^2 \tag{2.37}$$

2.2.2 Image Statistical Characterization

In characterizing images, the aforementioned 1-D random process $\mathbf{x}[n]$ is extended to 2-D random field, denoted as $\mathbf{x}[m,n]$, of which a realization, representing a possible image,

is denoted as $x[m,n]$. Random field representing nature images is always considered WSS, where the expectation does not depend on spatial coordinates, i.e.,

$$\eta_x[m,n] = \eta_x = \text{constant}, \quad \text{for all } m,\ n, \tag{2.38}$$

and the autocorrelation is translation-invariant. The autocorrelation can then be written as a function of two variables as shown in (2.39),

$$R(\tau_h, \tau_v) = E[\mathbf{x}(m,n)\mathbf{x}(m+\tau_h, n+\tau_v)] \tag{2.39}$$

where h and v refer to horizontal and vertical directions. The correlation coefficient, i.e., normalized $R(\tau_h, \tau_v)$, denoted as $r(\tau_h, \tau_v)$, is defined as in (2.40).

$$r(\tau_h, \tau_v) = \frac{R(\tau_h, \tau_v) - \eta_x^2}{\sigma_x^2} \tag{2.40}$$

Furthermore, an image field is often modeled as a realization of a first-order Markov, i.e., AR(1), field, where the autocorrelation is proportional to the geometric separation between pixels. In continuous coordinates, the autocorrelation of an AR(1) field is

$$R(\tau_h, \tau_v) = \sigma_x^2 \exp\{-\sqrt{\alpha^2\tau_h^2 + \beta^2\tau_v^2}\}, \tag{2.41}$$

where α and β are spatial scaling constants. The corresponding PSD is

$$S_{xx}(\omega_h, \omega_v) = \frac{1}{\sqrt{\alpha\beta}} \frac{2\sigma_x^2}{1 + (\omega_h^2/\alpha^2 + \omega_v^2/\beta^2)}. \tag{2.42}$$

Separable Autocorrelation Model As a simplifying assumption in image and video coding and processing, the AR(1) field is separable. With the separable autocorrelation model, the spatial autocorrelation is the product of the two along the horizontal and vertical directions, which means $R(\tau_h, \tau_v)$ can be calculated by (2.43).

$$R(\tau_h, \tau_v) = R(\tau_h)R(\tau_v) \tag{2.43}$$

In continuous coordinates, the 2-D joint autocorrelation in (2.41) and the PSD in (2.42) are simplified as in (2.44) and (2.45), respectively.

$$R(\tau_h, \tau_v) = \sigma_x^2 \exp\{-\alpha|\tau_h| - \beta|\tau_v|\} \tag{2.44}$$

$$S_{xx}(\omega_h, \omega_v) = \frac{4\alpha\beta\sigma_x^2}{(\alpha^2 + \omega_h^2)(\beta^2 + \omega_v^2)} \tag{2.45}$$

Denoting $e^{-\alpha}$ and $e^{-\beta}$ as ρ_h and ρ_v, respectively, one can obtain the discretized description of (2.44) as in (2.46),

$$R(\tau_h, \tau_v) = \sigma_x^2 \rho_h^{|\tau_h|} \rho_v^{|\tau_v|} \tag{2.46}$$

where τ_h and τ_v are integer-valued variables. In the special but prevalent case that ρ_h and ρ_v are both equal to as ρ, $R(\tau_h, \tau_v)$ in (2.46) is rewritten by (2.47),

$$R(\tau_h, \tau_v) = \sigma_x^2 \rho^{|\tau_h| + |\tau_v|} \tag{2.47}$$

which is a generalization of the 1-D formulation in (2.31). Therefore, the autocorrelation in the horizontal or vertical direction can also be expressed by (2.31) or (2.32), but the correlation coefficient between diagonal neighbors, i.e., $R(1,1)$, is equal to ρ^2. This separable AR(1) field, which can be generated by the difference equation (2.48), has been proved to be an accurate model for natural images, when ρ is about 0.95 [13].

$$\mathbf{x}(m,n) = \rho\mathbf{x}(m,n-1) + \rho\mathbf{x}(m-1,n) - \rho^2\mathbf{x}(m-1,m-1) + \mathbf{z}(m,n) \tag{2.48}$$

In (2.48), $\mathbf{z}(m,n)$ is a 2-D zero-mean random field of uncorrelated random variables with common variance σ_z^2.

2.3 Predictive Coding

Since pixels in video signals are highly correlated, it is a waste of bits to specify each pixel independently. In a predictive encoder, a pixel is predicted from the spatial or temporal neighbors that have been stored in the memory, and then only the prediction error is quantized and coded. In the corresponding decoder, the reconstructed value is the predicted value plus the quantized prediction error. To guarantee that the encoder and the decoder use exactly the same prediction, the encoder must repeat the same process as in the decoder to reproduce reconstructed samples. This is called closed-loop prediction, a key technique in hybrid coding.

In general, one can use various kinds of predictors, including linear and non-linear

ones. In practice, linear predictors are used almost exclusively for the ease of implementation. Optimal linear predictor is designed by minimizing the variance of the prediction error. Let random variable s_0 represent the pixel to be coded, let random variable s_k, $k = 1, \cdots, K$, represent the neighboring pixels used to predict s_0, and then the prediction for s_0 is given by (2.49),

$$s_p = \sum_{k=1}^{K} a_k s_k \tag{2.49}$$

where a_k is called prediction coefficient. From another viewpoint, (2.49) can be considered as a filtering process, where a_k is the filter coefficient. Then, the variance of the prediction error is defined as in (2.50).

$$\sigma_e^2 = E[|s_0 - s_p|^2] = E\left[\left|s_0 - \sum_{k=1}^{K} a_k s_k\right|^2\right] \tag{2.50}$$

Let $\partial\sigma_e^2/\partial a_l = 0$, which yields (2.51), and then the optimal prediction coefficients, achieving the minimum of σ_e^2, can be calculated.

$$E\left[\left(s_0 - \sum_{k=1}^{K} a_k s_k\right) s_l\right] = 0, \quad l = 1, 2, \cdots, K \tag{2.51}$$

As shown in (2.51), the prediction error is orthogonal to each pixel used for prediction, which is known as the orthogonal principle for the linear minimum mean squared error (LMMSE) estimator. Letting $R_{ss}[k, l]$ represent the autocorrelation between s_k and s_l, one can derive the following set of linear equations from (2.51):

$$\sum_{k=1}^{K} a_k R_{ss}[k, l] = R_{ss}[0, l], \quad l = 1, 2, \cdots, K \tag{2.52}$$

or, in the matrix terminology,

$$\begin{bmatrix} R_{ss}[1,1] & R_{ss}[2,1] & \cdots & R_{ss}[K,1] \\ R_{ss}[1,2] & R_{ss}[2,2] & \cdots & R_{ss}[K,2] \\ \cdots & \cdots & \cdots & \cdots \\ R_{ss}[1,K] & R_{ss}[2,K] & \cdots & R_{ss}[K,K] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_K \end{bmatrix} = \begin{bmatrix} R_{ss}[0,1]) \\ R_{ss}[0,2] \\ \cdots \\ R_{ss}[0,K] \end{bmatrix} \tag{2.53}$$

or

$$\mathbf{R}_{ss}\mathbf{a} = \mathbf{r} \tag{2.54}$$

These equations for solving the LMMSE predictor is called the Yule-Walker equations. Based on (2.54), the optimal prediction coefficients and the variance of the prediction error are given in (2.55) and (2.56), respectively.

$$\mathbf{a} = \mathbf{R}_{ss}^{-1}\mathbf{r} \tag{2.55}$$

$$\sigma_e^2 = E[(s_0 - s_p)s_0] = R_{ss}[0,0] - \sum_{k=0}^{K} a_k R_{ss}[k,0] = R_{ss}[0,0] - \mathbf{r}^T\mathbf{a} = R_{ss}[0,0] - \mathbf{r}^T\mathbf{R}_{ss}^{-1}\mathbf{r} \tag{2.56}$$

This solution for the optimal predictor assumes that the reference pixels s_k ($k = 1, 2, \cdots, K$) used for prediction are original. In a lossy predictive coder, s_k must be replaced by the reconstructed pixels $\hat{s}_k$ and the error in (2.50) is replaced by (2.57) accordingly.

$$\sigma_e^2 = E\left[\left|s_0 - \sum_{k=1}^{K} a_k \hat{s}_k\right|^2\right] \tag{2.57}$$

The prediction coefficients $\mathbf{a}$ can be derived by following the above reasoning steps, but the autocorrelation matrix of the prediction vector is derived from reconstructed pixels, denoted as $\mathbf{s}_p = [\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_K]^T$, and $\mathbf{r}$ is the cross-correlation between the pixel to be predicted s_0 and $\mathbf{s}_p$.

2.4 Transform Coding

After predictive coding, transform coding is applied to the prediction error for further de-correlation. Ideally, a transform should be applied to an entire residual frame, to fully exploit the spatial correlation. However, to the reduce computational complexity, block-based transform coding is more often used in practice, which divides a residual frame into non-overlapping blocks, and applies the transform to each block. An introduction of block-based transform is given in Section 2.4.1.

A good transform should compact the energy of the block into as few coefficients as possible. The performance is measured by transform coding gain, the maximum gain of transform coding over Pulse-code Modulation (PCM) coding. Section 2.4.2 derives transform coding gain for a given transform and discusses how to achieve it by optimal bit allocation.

2.4.1 Block-based Unitary Transform

1-D Unitary Transform

Let $\mathbf{s} = [s_0, s_1, \cdots, s_{N-1}]^T$ and $\mathbf{t} = [t_0, t_1, \cdots, t_{N-1}]^T$ represent the vectors before and after transform, respectively. The transform matrix of an order-N transform, denoted as $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{N-1}]$, is formed by linearly independent basis vectors $\mathbf{u}_k = [u_{k0}, u_{k1}, \cdots, u_{k(N-1)}]^T$, $k = 0, 1, \cdots, N-1$. Since $\mathbf{s}$ is represented as a linear combination of the basis vectors, of which the contributions are shown by $\mathbf{t}$, the relationship between $\mathbf{s}$ and $\mathbf{t}$ is expressed by the inverse transform as in (2.58).

$$\mathbf{s} = \sum_{k=0}^{N-1} t_k \mathbf{u}_k = \mathbf{U}\mathbf{t} \tag{2.58}$$

Since the basis vectors are linearly independent, $\mathbf{U}$ is invertible. Therefore, the transform coefficients $\mathbf{t}$ are obtained by forward transform (2.59).

$$\mathbf{t} = \mathbf{U}^{-1}\mathbf{s} = \mathbf{V}\mathbf{s} \tag{2.59}$$

The transform is called unitary transform, if the basis vectors are orthonormal to each other, as defined in (2.60),

$$\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}_N \tag{2.60}$$

where $\mathbf{I}_N$ represents the $N \times N$ identity matrix and the superscript H denotes vector transposition followed by element conjugation. Hence, $\mathbf{U}$ and $\mathbf{U}^H$ are the inverses of each other. To summarize, the forward and inverse unitary transforms in (2.59) and (2.58) are re-written by (2.61) and (2.62), respectively.

$$\mathbf{t} = \mathbf{U}^H\mathbf{s} = \mathbf{V}\mathbf{s} \tag{2.61}$$

$$\mathbf{s} = \sum_{k=0}^{N-1} t_k \mathbf{u}_k = \mathbf{U}\mathbf{t} = \mathbf{V}^H\mathbf{t} \tag{2.62}$$

Usually, a transform is defined by the forward transform matrix $\mathbf{V}$ and the basis vectors of the transform are the conjugates of the row vectors in $\mathbf{V}$.

2-D Unitary Transform for Video Coding

The unitary transform is extended to 2-D, when applied to residual blocks in video coding. In this case, an $M \times N$ block $\mathbf{S}$ is represented as a linear combination of orthonormal basis images $\mathbf{U}_{k,l}$ that also have size $M \times N$, as shown in (2.63),

$$\mathbf{S} = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T_{k,l} \mathbf{U}_{k,l} \tag{2.63}$$

where $T_{k,l}$ is the transform coefficient in $\mathbf{T}$. The inner product of any two basis images $\mathbf{U}_{i,j}$ and $\mathbf{U}_{k,l}$ satisfies (2.64).

$$\langle \mathbf{U}_{k,l} \cdot \mathbf{U}_{i,j} \rangle = \begin{cases} 1, \text{ if } k = i, \ l = j \\ 0, \text{ otherwise} \end{cases} \tag{2.64}$$

With an orthonormal set of basis images, one can find the transform coefficient $T_{k,l}$ by the inner product of $\mathbf{U}_{k,l}$ and $\mathbf{S}$: $T_{k,l}$ equals $\langle \mathbf{U}_{k,l} \cdot \mathbf{S} \rangle$.

A 2-D transform is separable, when each basis image $\mathbf{U}_{k,l}$ is the outer product of two basis vectors from two 1-D transforms, $\mathbf{h}_k = [h_{k0}, h_{k1}, \cdots, h_{k(M-1)}]^T$ and $\mathbf{g}_l = [g_{k0}, g_{k1}, \cdots, g_{k(N-1)}]^T$, respectively, as shown in (2.65)

$$\mathbf{U}_{k,l} = \mathbf{h}_k \mathbf{g}_l^T \tag{2.65}$$

It can be shown that $\mathbf{U}_{k,l}$ will form an orthonormal basis set as long as $\mathbf{h}_k$ and $\mathbf{g}_l$ form orthonormal basis sets, respectively. Separable 2-D transform is usually accomplished by sequentially performing 1-D transforms to rows and columns using the transform matrices $\mathbf{G} = [\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_{N-1}]$ and $\mathbf{H} = [\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_{M-1}]$, respectively. The forward and inverse separable 2-D transforms are shown in (2.66) and (2.67), respectively.

$$\mathbf{T} = \mathbf{H}^H \mathbf{S} \mathbf{G} \tag{2.66}$$

$$\mathbf{S} = \mathbf{H} \mathbf{T} \mathbf{G}^H \tag{2.67}$$

At present, the 2-D transforms used in video coding are all separable, including Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Walsh-Hadamard Transform (WHT), and Karhunen-Loeve Transform (KLT). See [14] for the definitions of these transforms. DFT is the most widely used to analyze discrete signals in the

frequency domain, whereas DCT, WHT, and KLT are more useful for source coding, especially image and video coding. In the following, these three transforms are reviewed.

KLT KLT is the optimal transform in terms of de-correlation; the transform coefficients are uncorrelated at all, which means the autocorrelation of the transform coefficients, denoted as $\mathbf{R}_T$, is a diagonal matrix. Since $\mathbf{R}_T$ can be obtained by transforming $\mathbf{R}_S$, the autocorrelation of the input source, KLT actually diagonalizes $\mathbf{R}_S$, as in (2.68),

$$\mathbf{T}_{KLT}\mathbf{R}_S\mathbf{T}_{KLT}^T = \mathbf{R}_T = \text{diag}\boldsymbol{\lambda} \tag{2.68}$$

where $\mathbf{T}_{KLT}$ is the transform matrix and $\boldsymbol{\lambda}=[\lambda_0, \lambda_1, \cdots, \lambda_{N-1}]^T$ are the eigenvalues of $\mathbf{R}_s$. Denoting the corresponding normalized eigenvectors of $\boldsymbol{\lambda}$ as $\mathbf{V}=[\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_{N-1}]$, one can constructed the KLT transform matrix by transposing $\mathbf{V}$. However, KLT is seldom used in video coding because of the high computational complexity. Firstly, the transform matrix is derived based on the second-order statistics of the source and it is difficult to accurately track such time- and space-variant statistics. Secondly, KLT is usually implemented using floating point operations without fast algorithm.

DCT 2-D DCT is the most commonly used transform in signal processing applications in general and especially in image and video compression. A lot of literature deals with DCT; a comprehensive discussion with many further references is provided by Rao and Yip [15]. The k^{th} basis vector of 1-D order-N DCT are defined by (2.69),

$$u_{kn} = \alpha(k)\cos\frac{(2n+1)k\pi}{N}, \quad n = 0, 1, \cdots, N-1 \tag{2.69}$$

$$\alpha(k) = \begin{cases} \sqrt{1/N} & k = 0 \\ \sqrt{2/N} & k = 1, 2, \cdots, N-1 \end{cases}$$

The forward and inverse transforms are described by (2.70) and (2.71), respectively.

$$t_k = \sum_{n=0}^{N-1} u_{k,n}s_n \tag{2.70}$$

$$s_n = \sum_{k=0}^{N-1} u_{k,n}t_k \tag{2.71}$$

When applied to an $M \times N$ block, 2-D $M \times N$ DCT is usually accomplished by sequentially applying 1-D order-M DCT to each column of the input block and 1-D order-N DCT to each row of the intermediate block. Several fast algorithms for computing order-N 1-D DCT have been developed to further reduce the complexity [16; 17].

The DCT basis vectors are real, varying in a sinusoidal pattern with different frequencies. When the input source is highly correlated, such as image and video signals, the basis vectors of DCT approximates those of KLT. Therefore, 2-D DCT is considered as the sub-optimal choice for image and video coding, as it achieves comparable performance with KLT but has much lower computational complexity.

WHT As a computational simpler transform, WHT is also useful in video coding. The order of WHT should be a power of two. WHT is also separable, and the k^{th} basis vector of 1-D order-N WHT, $N = 2^m$, is defined by (2.72),

$$u_{kn} = \frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{m-1} b_i(n)p_i(k)} \tag{2.72}$$

where $b_i(n)$ is the k^{th} bit (from right to left) in the binary representation of n and $p_i(k)$ are computed using (2.73) as below.

$$\begin{aligned} p_0(k) &= b_{m-1}(k) \\ p_1(k) &= b_{m-1}(k) + b_{m-2}(k) \\ p_2(k) &= b_{m-2}(k) + b_{m-3}(k) \\ &\cdots \\ p_{m-1}(k) &= b_1(k) + b_0(k) \end{aligned} \tag{2.73}$$

The basis vectors of the WHT matrix consist of coefficients differing only in the sign. Hence, WHT decomposes signals by a set of rectangular waveforms instead of sinusoidal waveforms. Harmuth [18] has suggested a frequency interpretation for the WHT matrix: the number of sign changes in each basis vector divided by two is called the frequency of that basis vector. It can be shown that the basis vector $\mathbf{u}_k$ has the frequency $k/2$, i.e., the frequency increases with the basis vector's index, just like DCT. DWT is not so efficient in de-correlation as DCT, but its rectangular basis vectors suggest the high efficiency in compressing images containing many edge and discontinuities.

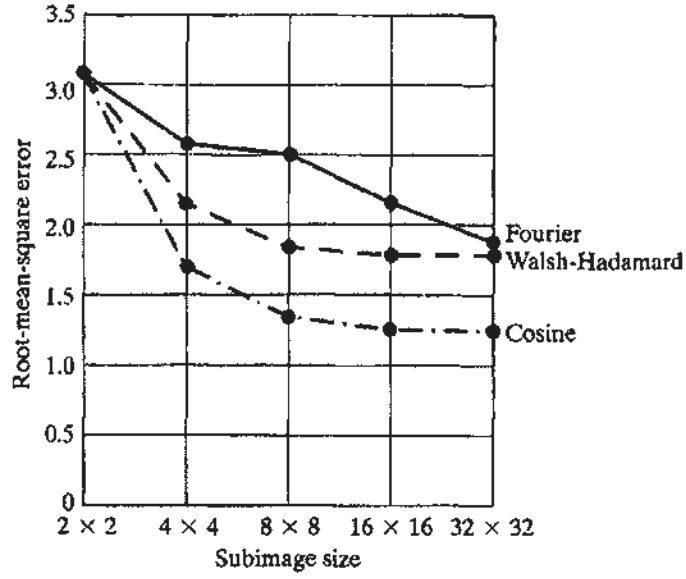


Figure 2.3: *Reconstruction error versus transform block size [14]*

Selection of Transform Size Apart from transform type, the size of the subimage, where the transform is applied, also significantly affects the coding performance and computational complexity. Fig. 2.3 illustrates the impact of subimage size on the reconstruction error of transform coding. The data plotted are obtained by dividing a set of monochrome images into subimages of size $n \times n$, for n=2, 4, 8, 16, and 32, computing the transform for each subimage, truncating 75% of the resulting coefficients, and taking the inverse transform of the truncated coefficients. The largest reconstruction error is resulted by using 2×2 subimage, where the three curves intersect, and the reconstruction error decreases with the subimage size. Obviously, DCT is the most efficient in image coding, because it has the smallest reconstruction error with any subimage size larger than 2×2. As the subimage size becomes greater than 8×8, the DCT and WHT curves flatten, but the computational complexity increases rapidly. Therefore, 2D order-8 DCT is a good trade-off between performance and complexity, and thus has been widely adopted in image and video coding.

2.4.2 Bit Allocation and Transform Coding Gain

Given the desired average bit rate R, the optimal bit allocation ensures that the total bits RN are appropriate allocated among the N transform coefficients, such that the average reconstruction error is minimized and the maximum gain of transform coding

over PCM coding, known as transform coding gain, is achieved.

The transform discussed in this sub-section is 1-D unitary, as all the 2-D unitary transforms used in image and video coding are separable. The transform coefficients are assumed to be coded individually by fixed-length coding (FLC).

Optimal Bit Allocation

Minimizing the reconstruction error in the time domain is equivalent to minimizing the quantization error in the transform domain, because of the orthogonality of unitary transform. Using the Lagrange multiplier method, this constraint minimization problem can be converted to the minimization of (2.74),

$$J(R_k) = D_{t,k}(R_k) + \lambda \left(\sum_{k=0}^{N-1} R_k - RN \right) \tag{2.74}$$

where $D_{t,k}(R_k)$ is the quantization error of the k^{th} coefficients t_k coded by R_k bits. Suppose the scalar quantizer for t_k is optimal; the minimum of $D_{t,k}(R_k)$ can be approximated by (2.75) [17],

$$D_{t,k}(R_k) = \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k} \tag{2.75}$$

where $\epsilon_{t,k}^2$ is a constant depending on the probability distribution function (p.d.f.) of t_k and $\sigma_{t,k}^2$ is the variance of t_k. Letting $\frac{\partial J}{\partial R_k} = 0$ and substituting (2.75) into (2.74), one obtains (2.76)

$$\frac{\partial D_{t,k}}{\partial R_k} = -2 \ln 2 D_{t,k} = -(2 \ln 2) \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k} = -\lambda, \quad k = 0, 1, \cdots, N-1 \tag{2.76}$$

After some reasoning steps, (2.77) shows the optimal number of bits allocated to t_k.

$$R_k = R + \frac{1}{2} \log_2 \frac{\epsilon_{t,k}^2 \sigma_{t,k}^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right)^{1/N}} \tag{2.77}$$

With this bit allocation, the distortions of all the coefficients are equal; that is

$$D_{TC} = D_t = D_{t,k} = \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right)^{1/N} 2^{-2R} \tag{2.78}$$

The solution implies that a coefficient with a larger variance should be given more bits, whereas a coefficient with a smaller variance should be given fewer bits. The optimal

allocation ensures that all the coefficients have the same quantization error.

Transform Coding Gain

As image and video sources are widely considered WSS, all samples have the same variance, denoted as σ_s^2. If the samples of the source is separately quantized using the optimal scalar quantization and coded using PCM, the distortion is related to the rate by (2.79) [12],

$$D_{PCM} = D_{s,k} = \epsilon_s^2 \sigma_s^2 2^{-2R} \tag{2.79}$$

where ϵ_s^2 depends on the p.d.f. of the source sample and $D_{s,k}$ is the quantization error of the k^{th} sample coded by R bits.

The performance of transform coding is measured by transform coding gain over PCM, defined as in (2.80)

$$G_{TC} = \frac{D_{PCM}}{D_{TC}} \tag{2.80}$$

Substituting (2.78) and (2.79) into (2.80), one can calculate G_{TC} by (2.81).

$$G_{TC} = \frac{\epsilon_s^2 \sigma_s^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/N}} = \frac{\epsilon_s^2}{\left(\prod_k \epsilon_{t,k}^2\right)^{1/N}} \frac{\frac{1}{N}\sum_k \sigma_{t,k}^2}{\left(\prod_k \sigma_{t,k}^2\right)^{1/N}} \tag{2.81}$$

According to (2.81), G_{TC} is proportional to the ratio of the arithmetic mean of transform coefficient variances to the geometric mean of the variances. G_{TC} is greater than or equal to one, because the arithmetic mean is greater than or equal to the geometric mean for any arbitrary set of values and ϵ_s^2 is also greater than or equal to $\left(\prod_k \epsilon_{t,k}^2\right)^{1/N}$ in general. The more unevenly valued the transform coefficients' variances are, the smaller their geometric mean is and thus the higher G_{TC} is.

2.5 Entropy Coding

The predictive coding and transform coding remove the temporal and spatial redundancies, respectively, and convert the video signal to a compact representation, which consists of syntax elements (SE) instead of samples. Then, the SEs are losslessly converted to binary bitstreams; this process is known as entropy coding. Because of the imperfect predictive and transform coding, statistical redundancy still exists among SEs, reflected by the non-uniform distribution of individual SEs and the correlation

among different SEs. The former can be exploited by variable-length coding (VLC) introduced in Section 2.5.1, whereas the latter can be removed by joint coding or conditional coding, which will be introduced in Sections 2.5.2 and 2.5.3, respectively.

2.5.1 Variable-Length Coding (VLC)

With VLC, each sample x_n of a discrete source $\mathbf{x}$ with an alphabet $A = \{a_1, a_2, \cdots, a_L\}$ is assigned a binary codeword c_n, of which the length, i.e., the number of bits, depends on the p.d.f. of $\mathbf{x}$. The codebook $C = \{c(a_1), c(a_2), \cdots, c(a_L)\}$ is pre-defined, where $c(a_i)$ is the codeword for symbol a_i. The codeword c_n for x_n is $c(x_n)$. A useful codebook ensures that a coded sequence can be uniquely decodable, i.e., a sequence of codewords corresponds to one and only one possible sequence of source symbols. Let $l(a_i)$ denote the length of $c(a_i)$, and the bit rate R for coding one sample is by definition the expectation of $l(a_i)$, calculated as in (2.82).

$$R = \sum_{a_i \in A} p(a_i) l(a_i) \tag{2.82}$$

At the same time, the entropy (measured by bit) of $\mathbf{x}$ with an alphabet A and a p.d.f. $p(x)$ is defined by

$$H_1(\mathbf{x}) = -\sum_{x \in A} p(x) \log_2 p(x). \tag{2.83}$$

If the codebook is defined based on an accurate p.d.f., the lower bound of bit rate $\bar{R}_1(\mathbf{x})$ satisfies

$$H_1(\mathbf{x}) \leq \bar{R}_1(\mathbf{x}) \leq H_1(\mathbf{x}) + 1. \tag{2.84}$$

Popular VLC methods used in image and video coding includes Huffman coding and arithmetic coding.

Huffman Coding

Given a discrete source with alphabet $A = \{a_1, a_2, \cdots, a_L\}$ and p.d.f. $p(a_l)$, Huffman coding designs a codebook for all possible symbols, so that a symbol appearing more frequently is assigned a shorter codeword and vice versa. The basic procedure for codebook design using Huffman coding is as follows:

Step 1: Arrange the symbol probabilities $p(a_l)$, $l = 1, \cdots, L$, in a decreasing order. If only one node left, the codebook designing is finished. Otherwise, go to Step 2.

Step 2: Find the two nodes with the smallest probabilities. Arbitrarily assign 1 and 0 to these two nodes and merge them to form a new node whose probability is the sum of the two merged nodes. Go back to Step 1 and consider them as leaf nodes of a tree.

With Huffman coding, at least one bit has to be used for each sample, which becomes inefficient when the probability of a sample is much larger than 0.5. To further reduce the bit rate, one can design Huffman codebook for a vector sample instead of a signal sample or based on conditional p.d.f., leading to joint coding and conditional coding, which will be introduced later.

Arithmetic Coding

Arithmetic coding represents a sequence of symbols by an interval in the line segment from 0 to 1, with length equal to the probability of the sequence. The intervals corresponding to all possible sequences, of which the probabilities sum to 1, fill up the entire line segment. The coded bits for a sequence is the binary representation of any point in the interval corresponding to the sequence. However, it does not mean the coding process cannot start until the entire sequence appears. Actually, one starts from an initial interval determined by the first symbol, and then recursively divides the previous interval after each new symbol appears. To specify an interval, the lower and upper bounds of the interval are represented in binary. Whenever the most significant bit (MSB) of the lower boundary is the same as that of the upper bound, this bit is shifted out. At the end of the source sequence, all the bits that have been output will be the binary representation of the interval corresponding to the sequence. The larger probability a sequence has, the longer will be the interval and the fewer bits will be needed to specify the interval.

The advantage of arithmetic coding is that one can keep track of the varying source statistics simply by updating the probability model. One can also easily realize conditional coding, which will be introduced later, by using different probability models for different conditioning states. With Huffman coding, one has to redesign the codebook based on an updated p.d.f. to track the un-stationary statistics, or design multiple codebooks for different conditioning states. Because of the higher coding efficiency and ease of adaptation, arithmetic coding is a better alternative than Huffman coding, as long as the computational complexity involved is acceptable.

2.5.2 Joint Coding

If the samples in a discrete source are dependent, one can further reduce the bit rate by considering each successive N samples as a vector sample and jointly coding them by assigning one codeword for each possible vector symbol in A^N, where A^N represents the N-fold Cartesian product of A. The first-order entropy of the vector source is actually the N^{th} order entropy of the original source, denoted as $H_N(\mathbf{x})$. With the N^{th} order joint p.d.f. $p(x_1, x_2, \cdots, x_N)$ known, $H_N(\mathbf{x})$ can be calculated by re-writing (2.83) as in (2.85).

$$\begin{aligned} H_N(\mathbf{x}) &= H(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N) \\ &= -\sum_{[x_1, x_2, \cdots, x_N] \in A^N} p(x_1, x_2, \cdots, x_N) \log_2 p(x_1, x_2, \cdots, x_N). \end{aligned} \tag{2.85}$$

Similar to (2.84), the lower bound of the bit rate $\bar{R}^N$ required to represent each vector is given in (2.86).

$$H_N(\mathbf{x}) \leq \bar{R}^N(\mathbf{x}) \leq H_N(\mathbf{x}) + 1 \tag{2.86}$$

Letting $\bar{R}_N(\mathbf{x})$, equal to R^N/N, represent average bit rate per sample yeilds (2.87).

$$H_N(\mathbf{x})/N \leq \bar{R}_N(\mathbf{x}) \leq H_N(\mathbf{x})/N + 1/N \tag{2.87}$$

With N tending to ∞, the bit rate of joint coding, $\bar{R}_N(\mathbf{x})$, approaches the source entropy rate $\bar{H}(\mathbf{x})$ defined in (2.88).

$$\bar{H}(\mathbf{x}) = \lim_{N \to \infty} \frac{1}{N} H_N(\mathbf{x}). \tag{2.88}$$

The entropy rate is the lower bound of the bit rate required to code a discrete source losslessly, which is achieved only when an infinite number of samples are coded together.

2.5.3 Conditional Coding

To improve the efficiency of scalar coding, an alternative to joint coding is conditional coding, a.k.a., context-based coding. With M^{th} order conditional coding, the codebook for a discrete source $\mathbf{x}$ is designed according to the conditional p.d.f. of single sample, where the condition means the pattern formed by M coded samples. Such a pattern is called the context. For each possible context, a codebook is designed. If the source

alphabet size is L, the maximum number of contexts and consequently the maximum number of codebooks are both L^M. Before moving to the lower bound of conditional coding, conditional entropy is defined.

The M^{th} order conditional entropy $H_{C,M}(\mathbf{x})$ of a discrete source $\mathbf{x}$ with the M^{th} order joint p.d.f. $p(x_1, x_2, \cdots, x_N)$ and conditional p.d.f. $p(x_{M+1}|x_M, x_{M-1}, \cdots, x_1)$ is the conditional entropy of a sample x_{M+1} given its previous M samples $\mathbf{x}_M, \cdots, \mathbf{x}_1$,

$$\begin{aligned} H_{C,M}(\mathbf{x}) &= H(\mathbf{x}_{M+1}|\mathbf{x}_M, \mathbf{x}_{M-1}, \cdots, \mathbf{x}_1) \\ &= \sum_{[x_1, x_2, \cdots, x_M] \in A^M} p(x_1, x_2, \cdots, x_M) H(\mathbf{x}_{M+1}|x_M, x_{M-1}, \cdots, x_1) \end{aligned} \quad (2.89)$$

where

$$\begin{aligned} & H\left(\mathbf{x}_{M+1}|x_M, x_{M-1}, \cdots, x_1\right) \\ &= -\sum_{x_{M+1} \in A} p(x_{M+1}|x_M, x_{M-1}, \cdots, x_1) \log_2 p(x_{M+1}|x_M, x_{M-1}, \cdots, x_1). \end{aligned} \quad (2.90)$$

Similar to (2.84) and (2.86), it can be shown that the minimal bit rate $\bar{R}_{C,M}(\mathbf{x})$ required to represent a discrete source $\mathbf{x}$ using M^{th} order conditional coding satisfies

$$H_{C,M}(\mathbf{x}) \leq \bar{R}_{C,M}(\mathbf{x}) \leq H_{C,M}(\mathbf{x}) + 1. \quad (2.91)$$

The limit of $\bar{R}_{C,M}(\mathbf{x})$ with M tending to ∞ varies in the interval as in (2.92).

$$\bar{H}(\mathbf{x}) \leq \lim_{M \to \infty} \bar{R}_{C,M}(\mathbf{x}) \leq \bar{H}(\mathbf{x}) + 1 \quad (2.92)$$

Compared with (2.84), conditional coding can achieve a lower bit rate than scalar coding, because the source entropy rate $\bar{H}(\mathbf{x})$ is less than $H_1(\mathbf{x})$, unless the source is independent and identically-distributed (i.i.d.). However, as one still codes one sample at a time, the upper bound always differs from the entropy rate by one bit, even when the conditioning order M tends to infinity.

2.6 Hybrid Coding

Hybrid coding refers to the combination of predictive coding and transform coding. Fig. 2.4 shows the generic block diagrams of the hybrid coding scheme for a Macroblock (MB). Hybrid coding is essentially the core of all video coding standards, but the

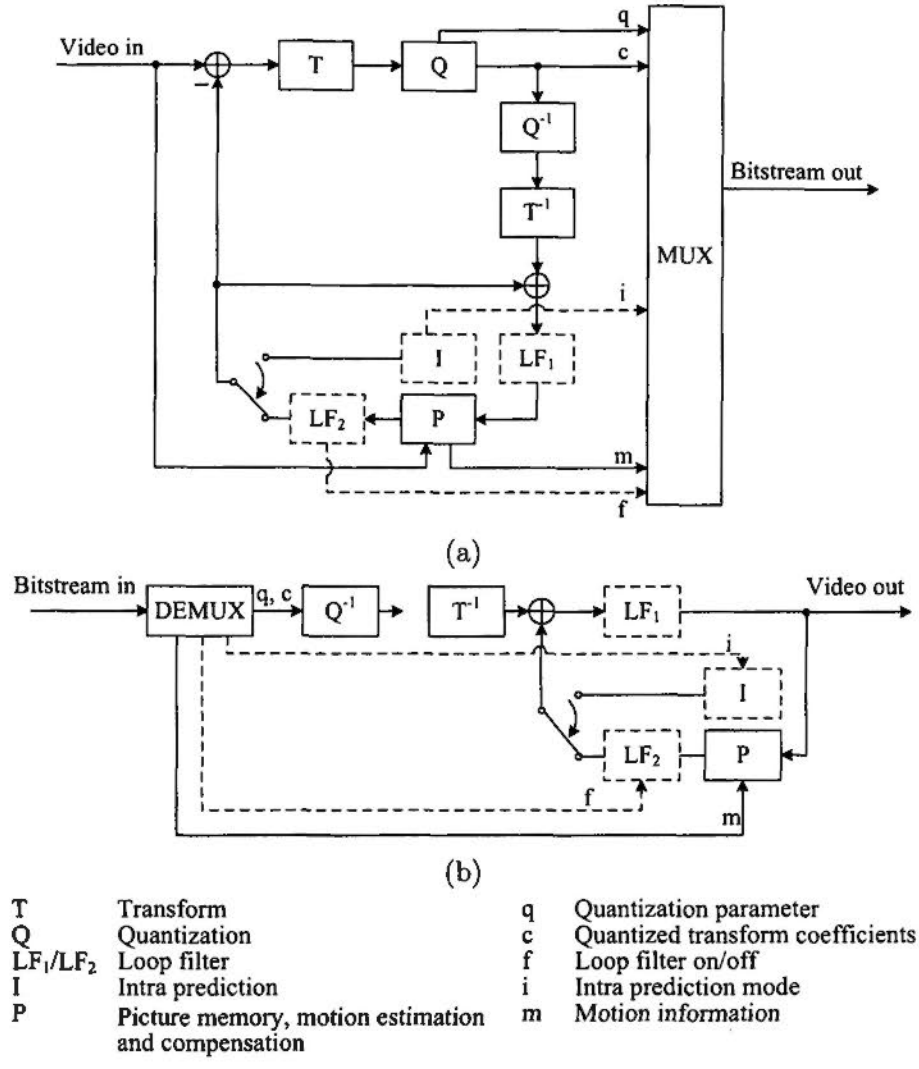


Figure 2.4: *Block diagrams of the hybrid (a) encoder and (b) decoder*

modules with dashed boxes are not necessarily included. Each MB consists of 16×16 luminance pixels and the associated chrominance pixels, depending on the color format, e.g., in 4:2:0 format, Cb and Cr pixels within one MB are both 8×8.

At the encoder (see Fig. 2.4 (a)), an MB is predicted from a coded frame, known as reference frame, stored in P. A motion vector (MV) consisting of horizontal and vertical displacements is used to indicate the position of the prediction block, called target block, in the reference frame, and the target block is simply copied for MCP. Besides storing the reference frames, the other function of P is to search the MV, known as motion estimation (ME).

The difference of the current MB and its prediction is input to T, where 2-D transform is applied to compact the error energy to fewer coefficients, and the coefficients are then quantized in Q. In most standards, the transform is 2-D order-8 DCT and the quantization is uniform. The quantization stepsize is signaled by the quantization parameter (QP), with which the stepsize increases. The coefficients after quantization are converted to a 1-D array following a zig-zag scan order putting the low frequency coefficients in front of the high frequency ones.

Such hybrid coding process converts the original video to a compressed representation that consists of SEs. It is followed by entropy coding, where all SEs are losslessly converted to binary codewords by using VLC or FLC and multiplexed together to a bitstream.

If a coded MB belongs to a reference frame that will be used to predict the following frames, it should be reconstructed by sequentially applying dequantization and inverse transform and then being compensated by its prediction. A reconstructed reference frame may contain blockiness near the transform block boundaries, caused by quantization and leads to inaccurate MCP. Some standards apply an in-loop deblocking filter on reference frames. The filtering can be enabled before the reconstructed frame is stored in P (see LF_1) as in H.264/AVC, or immediately before the target block is used for prediction (see LF_2) as in H.261.

At the decoder (see Fig. 2.4 (b)), the SEs in a bitstream are demultiplexed. The transform coefficients are dequantized, inverse transformed, and compensated by the intra (optional) or inter prediction according to the coding mode. Reference frames used in encoder and decoder should be the same to reconstruct identical videos.

Using MCP to code an MB is known as inter-mode, which is further classified as P-mode, if the prediction is restricted in the past frame, and B-mode, if the prediction is from the past frame or the future frame or both. In general, B-mode can achieve more accurate MCP than P-mode. The coding method without MCP is known as intra-mode, which can be used when inter-mode is not efficient or possible. In early standards, intra-mode directly applies the transform and quantization to original samples. In the recent standards, intra prediction (see I) is employed in the spatial or transform domain to reduce the spatial redundancy and only the intra prediction errors are coded. Extended concepts of I-, P-, and B-modes to a frame level are I-, P-, and B-frames.

2.7 Summary

This chapter reviews the fundamentals of hybrid coding. First of all, the formation and representation of video source are introduced. It is stated that the video source with YCbCr color space, 4:2:0 color subsampling format, and 8-bit bit depth will be used as the test material and PSNR will be used as the distortion measure in this work. Second, basic elements of probability theory and random process are reviewed, including correlation, covariance, and PSD, which are useful to characterize source videos. Based on the statistics characterization, video source is modeled as a separable AR(1) field with the correlation coefficient ρ about 0.95. Hence, video source contains high spatio-temporal redundancy, which should be removed as much as possible for efficient compression. Third, three fundamental techniques, predictive coding, transform coding, and entropy coding, are successively presented, which jointly accomplish the redundancy reduction and form the framework of the hybrid coding.

Chapter 3

Hybrid Video Coding Standard

In the past two decades, hybrid coding has been experiencing a tremendous growth, which is mainly reflected in the evolution of video coding standards. The timeline is shown in Fig. 3.1, in which each standard represents the state-of-the-art technology at the time. The standard series H.26x (x=$1\cdots4$) are recommended by VCEG in ITU-T, and MPEG-x (x=$1,2,4$) are developed by the Moving Picture Experts Group (MPEG) in ISO/IEC. A survey on the history of MPEG video coding has been written by C. Reader [19]. A comprehensive presentation of the present and future video coding standards can be found in [20]. In the following, the standards operating on HD videos are introduced, whereas the other standards, such as H.261 [1], H.263 [21], and MPEG-1 [4], dealing with only QCIF, CIF, and SIF video formats (see Table 1.1) for video telephony and streaming, are omitted. H.264/AVC and the Audio and Video coding Standard of China (AVS) are introduced in detail, because they are the platforms, where the proposed techniques are integrated, and their performances are the benchmarks this work compares to.

3.1 MPEG-2 Video

MPEG-2 [22] is the superseder of MPEG-1, as it also provides videocassette recorder (VCR)'s functionalities, such as random access, fast forward and rewind. The VCR-like functionalities are enabled by the concept of group of pictures (GOP), which starts with an I-frame followed by interleaving P- and B-frames. One can access any GOP without decoding previous GOPs, accomplish fast forward by decoding only I- or I- and P-frames, realize fast rewind by decoding only I-frames in a backward order.

The main feature distinguishing MPEG-2 from MPEG-1 is the efficient coding of interlaced videos, the key enabler of digital storage media and digital TV broadcast.

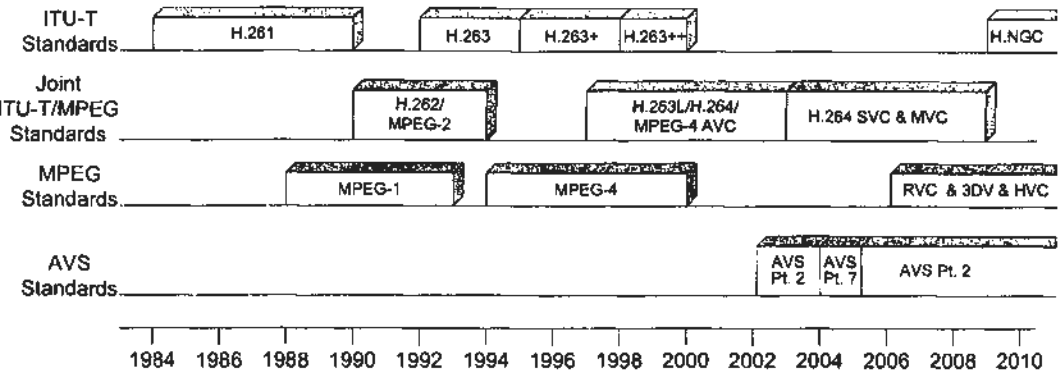


Figure 3.1: *The timeline of video coding standards*

Two opposite fields in an interlaced frame can be coded in an interleaving manner, known as frame-mode, or separately, known as field-mode. These two modes adapt to the frame level by picture level adaptive frame/field coding (PAFF) or to the MB level by MB level adaptive frame/field coding (MBAFF). Coding a field-mode MB basically follows the diagram in Fig. 2.4, but the reference pictures are fields instead of frames. Other modifications, such as zig-zag scan, are also made.

MPEG-2 mainly operates on the formats specified in ITU-R BT.601 [5] (formerly CCIR 601) with 4:2:0 color format and produces broadcast quality videos at the bit-rates of 4 to 8 mbits/s and high quality videos at 10 to 15 mbits/s [10]. It also handles HD videos [3] or other color formats at even higher bit-rates. MPEG-2 has been a great success with worldwide adoption for DVD-Video and digital TV broadcast via cable, satellite and terrestrial channels.

As a generic coding standard, full MPEG-2 syntax covers many features and parameters to meet a wide spectrum of bit-rates, video formats, quality levels and functionalities. However, certain application requires only a subset of the features and consumes limited processing power. To reduce the implementation cost and keep the interoperability, MPEG-2 first introduces the concept of profile and level, such that encoders and decoders compliant to the same profile and level can interwork with each other without supporting the full syntax. The profile describes the necessary features to support, while the level indicates the necessary processing capability by specifying the upper limits of spatial resolution, frame rate, and bit-rate. Among the seven profiles of MPEG-2, the Main Profile at the Main Level is the most widely used for TV broadcast.

3.2 MPEG-4 Visual

MPEG-4 is the first attempt to standardize the content-based coding techniques. Although further improving coding efficiency, MPEG-4 Visual [23] goes significantly beyond the pure hybrid coding scheme and provides a rich set of tools to code natural and synthetic objects.

The unit of video content is called video object (VO), e.g., a talking person without background. A VO is taken as a sequence of snapshots of an arbitrarily shaped object and its time instance are called video object planes (VOP). As successive VOPs are highly correlated, MCP is also efficient in spite of the arbitrary shape, but the aforementioned concepts of I-frame, P-frame, and B-frame are extended to I-VOP, P-VOP, and B-VOP, respectively.

If the shape of a VOP is equal to a frame, which means VO is equivalent to a video sequence, the content-based coding scheme is reduced to the conventional hybrid coding. The Simple Profile of MPEG-4 is compatible with the baseline H.263. However, MPEG-4 has some new technical developments to improve the coding efficiency.

- AC/DC intra prediction. The DC coefficient of either the block to the left or the upper block is used to predict the DC coefficient in the current block. For AC coefficients, the first row or column in the block predicting the DC coefficient is used to predict the corresponding row or column in the current block. Switching AC prediction can be performed at the MB level.

- Horizontal scan. In addition to the two scans in MPEG-2, a scan favoring the horizontal direction is introduced, which is the transposition of the vertical scan in MPEG-2. The scan mode is coupled with the AC prediction mode.

- Global motion compensation (GMC) for VO. In order to efficiently compensate the global motion caused by camera motion and zooming, GMC is introduced for a VO, using a small number of parameters. GMC is based on global ME, image warping, motion trajectory coding, and transform coding of the prediction errors.

- GMC based on Sprite. A sprite is a still image transmitted in the first frame, describing the panoramic background. For each consecutive frame in a sequence, the background is the affine transform of part of the Sprite, controlled by only

eight global motion parameters.

- Improved block-based MCP. The size of motion compensation (MC) blocks can be 8×8 or 16×16 and the resolution of ME and MV is increased to 1/4-pixel, which provides even more accurate MCP than the preceding standards.

If VO is of arbitrary shape and location, the coding scheme is extended by additionally coding the shape, represented by a binary alpha map defining whether a pixel belongs to an object or, alternatively, an 8-bit gray-scaled alpha map to describe the transparency information. The alpha map is coded on the MB base. For each opaque and transparent MB, the encoder just signals whether the MB is inside or outside the VOP. For a boundary MB, which contains VOP boundaries, the alpha map is coded by a context-based arithmetic coder and the VOP texture inside the boundary MB is transformed by the shape-adaptive DCT.

Meanwhile, MPEG-4 takes into account specificities of a wide variety of networks and terminals and allows universal access to multimedia information, by employing techniques for a high degree of scalability and error resilience. More technical details and the elaborately defined profiles are presented in [24] and [25].

3.3 H.264/AVC

H.264/AVC is the latest international video coding standard, jointly developed by VCEG and MPEG. In contrast to MPEG-4 Visual covering the ambitious breadth of functionalities, H.264/AVC returns to the narrower and more traditional focus on two goals, efficient compression and robustness to network environments, for generic rectangular-picture camera-shot video content. This section first illustrates the history of H.264/AVC. Then, key technical designs to improve coding efficiency and facilitate the network transportation are presented, followed by the explanation of its profiles and the corresponding application areas.

3.3.1 History

The H.264/AVC standardization project was launched by VCEG in early 1998, initially entitled H.26L (L stood for "long-term"). It was targeted at doubling the coding efficiency in comparison to any other existing video coding standards for a broad variety

of applications. The first draft was created in August 1999. In 2001, MPEG issued a Call for Proposals (CfP) with the similar target of H.26L and then adopted the developing H.26L as the starting point. Meanwhile, VCEG and MPEG formed a Joint Video Team (JVT) to jointly develop the new standard and the reference software was named joint model (JM). In 2003, the first phase of the standard was finalized and submitted for formal approval, which was later be adopted by ITU-T under the name of H.264 and by ISO/IEC as MPEG-4 Part 10 Advanced Video Coding (AVC) in the MPEG-4 suite of standards [26]. An overview of the first phase of H.264/AVC was presented in [27] and a comprehensive introduction can be found in [28]. The first phase of H.264/AVC [26] primarily focused on entertainment-quality video with 8-bit bit depth and 4:2:0 color format and did not support the highest video resolutions used in the most demanding professional environments. To address the needs of those professional applications, a continuation of the joint project was launched to add new extensions to the capabilities of the original standard. These extensions were finalized in 2005, named fidelity range extensions (FRExt) [29], which produced a suite of four new profiles, collectively called high profiles. Later in 2007, the breadth of the high profiles were enlarged to eight profiles [2]. In this thesis, the referred H.264/AVC includes the high profiles. More details of H.264/AVC FRExt are given in [30] and [31] for the 2005 and 2007 versions, respectively. In recent years, scalable video coding (SVC) [32] and multiview video coding (MVC) [33] were developed as two amendments to H.264/AVC. A detailed introduction of H.264/AVC, including all the extensions and amendments, can be found in [20].

3.3.2 Video Coding Layer (VCL)

The VCL design of H.264/AVC essentially follows the hybrid coding scheme, as introduced in Section 2.6, but significantly outperforms all previous video coding standards. The gain is due to more accurate prediction and more efficient entropy coding, contributed by a plurality of coding elements.

Adaptive Frame/Field Coding

A coded video sequence of H.264/AVC consists of a sequence of coded pictures. In an interlaced video, a coded picture represents either an entire frame or a single field. Two

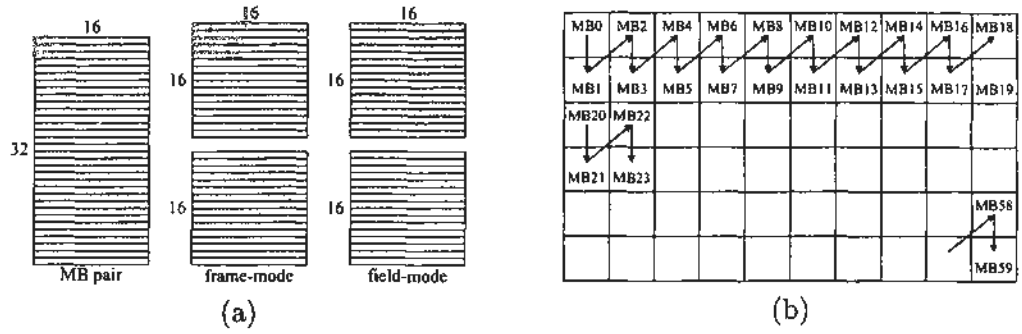


Figure 3.2: *The concept of MB pair. (a) Coding modes. (b) Coding procedure.*

coding modes, PAFF and MBAFF, as in MPEG-2 (see Section 3.1) are also employed. As shown in Fig. 3.2, the difference from the MBAFF in MPEG-2 is that the MBAFF in H.264/AVC makes the frame/field decision at the MB pair level rather than the MB level, so that the basic MB processing structure is kept intact. Fig. 3.2 (b) illustrates the MBAFF coding process in an interlaced frame.

If an interlaced frame consists of mixed regions where some regions are moving and others are not, MBAFF coding is efficient to code the non-moving regions in frame-mode and the moving regions in field-mode. However, in the case of rapid global motion and scene change, MBAFF coding is not so efficient as PAFF coding, because one field cannot use the opposite field in the same frame as a reference picture for MCP.

Slice

H.264/AVC supports flexible slice structure, which means the sizes and shapes of slices are not necessarily the multiples of MB rows as in the previous video coding standards. There are five types of slices as explained below and a coded picture may be composed of different types of slices.

- I-Slice. A slice in which all MBs of the slice are coded in intra-mode.
- P-Slice. In addition to intra-mode, some MBs can be coded using MCP from picture list 0.
- B-Slice. In addition to the coding modes available in a P-slice, some MBs can be coded using bi-directional prediction from picture list 0 and list 1.
- SP-Slice. A so-called switching P-slice that enables efficient switching between different pre-coded pictures.

- SI-Slice. A so-called switching I-slice that allows an exact match of an MB in an SP-slice for random access and error recovery.

Multipicture Reference

H.264/AVC supports multipicture reference, which means more than one previously coded pictures can be used as reference for MCP. At the decoder, the buffer storing reference pictures is named decoded picture buffer (DPB), which synchronously replicates the buffer of the encoder according to memory management control operations (MMCO) specified in the bitstream. DPB can hold up to 16 pictures, depending on the conformance point and the picture size. The flexible operations on DPB include the following two aspects.

Firstly, the concepts of P- and B-slices are generalized. Any picture type, including B-picture, is allowed to be stored in DPB for future reference, according to the indication in its NAL unit header (see 3.3.3). Thus, the picture type and the referencing capability are decoupled. Two lists of reference pictures, denoted as list 0 and list 1, can be arbitrarily constructed using the available pictures in the DPB. P-slices use only list 0 pictures for reference; B-slices can use both list 0 and list 1. However, it is not restricted that list 0 and list 1 consist of temporally preceding and succeeding pictures, respectively. Instead, list 0 may contain temporally succeeding pictures and vice versa. It is also possible for list 0 and list 1 to have same pictures. Fig. 3.3 shows the possible prediction cases for a B-picture, which have never been valid in any previous standards. By this means, the prediction direction and the picture type are decoupled. Therefore, the only substantial difference between P- and B-slices is that blocks in B-slices can be predicted by a weighted average of two distinct MCP blocks.

Secondly, the reference pictures in list 0 or list 1 are not necessarily to be temporally neighboring, but instead, they are classified as short-term and long-term. Short-term reference picture behaves the same as a conventional one: it is recently coded and involved in the first-in-first-out sliding window operation on the DPB. A reference picture is short-term by default. On the contrary, operations on long-term pictures, including marking, removing, and replacing, are explicitly signaled by MMCO specified in bitstream. This reference picture management introduced in H.264/AVC provides a high degree of flexibility in selecting reference picture and is especially efficient for

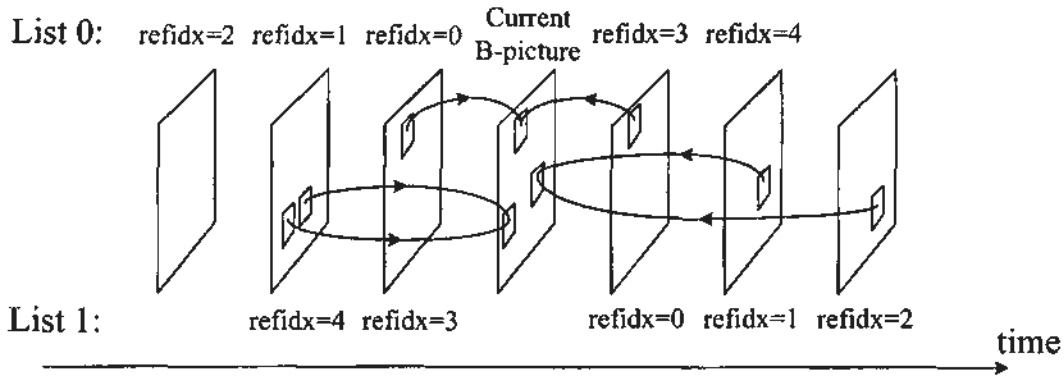


Figure 3.3: *The construction of list 0 and list 1, and possible bi-directional predictions for B-pictures (reference index is denoted as refidx.)*

coding videos with periodic transitions, such as interview.

Intra Coding

Intra prediction in H.264/AVC is performed in the spatial domain. For the luminance pixels in an MB, three modes are allowed, denoted as INTAR_4×4, INTAR_8×8 and INTAR_16×16, while the prediction size for chrominance pixels is fixed to be 8×8.

There are totally nine modes for INTAR_4×4. As shown in Fig. 3.4, the shaded area is the 4×4 block to predict and the squares labeled with capital letters A-M are the reference pixels in the neighboring coded blocks. Each mode indicates a prediction direction except the DC mode, of which the predicted value is the average of all the available reference pixels. INTAR_8×8 for luminance pixels uses basically the same concepts as INTAR_4×4 except the size, but the reference pixels are lowpass filtered before used for prediction, in order to reduce prediction error in a larger block.

The intra prediction with smaller block sizes and more directions is well suited for coding parts of a picture with rich details. On the contrary, for coding very smooth areas, INTAR_16×16 is employed with four modes (see Fig. 3.5). The 8×8 chrominance pixels within an MB are predicted using a similar prediction as for INTAR_16×16, since chrominance is usually smooth over large areas.

Besides, the standard includes a PCM mode, denoted as I_PCM, with which the values of the samples are sent directly using FLC. This mode enables lossless coding for region of interests (ROI) and avoids data expansion when representing the values in anomalous pictures.

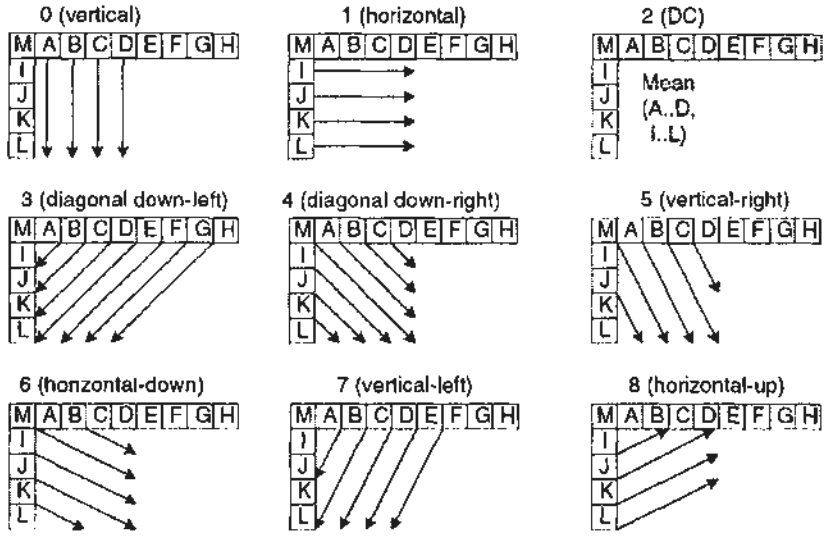


Figure 3.4: *Nine modes for INTAR_4×4 prediction*

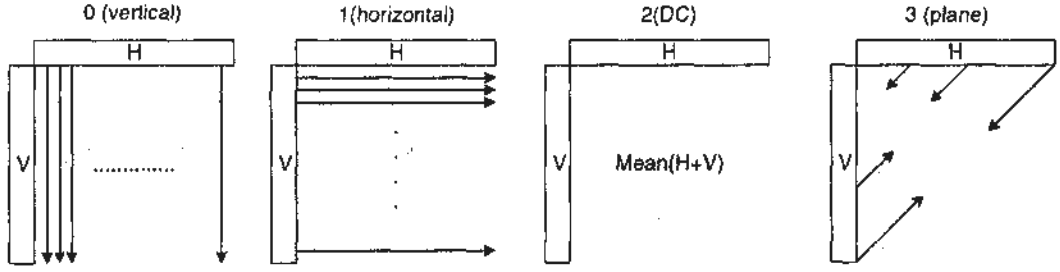


Figure 3.5: *Four modes for INTAR_16×16 prediction and chrominance intra prediction*

Variable MC Block Sizes

In inter-mode, an MB may be partitioned into multiple MC blocks. Fig. 3.6 (a) shows four partition types for luminance pixels. In case 8×8 are chosen, each partition is named sub-MB and can be further partitioned into 8×4, 4×8, or 4×4 MC blocks (see Fig. 3.6 (b)). Bigger MC blocks use less bits to signal the motion information but may result in large MCP errors, and thus are more suitable for homogeneous areas of a picture. Meanwhile, smaller MC blocks provide more accurate MCP but require more bits to code the motion information, so are beneficial to detailed areas.

MV Prediction

MV in H.264/AVC is predicted from the MVs of the neighboring coded MC blocks and only the prediction error, MVD, is coded. Similar concept has been adopted in previous standards, but the MV prediction in H.264/AVC is improved and more accurate. As

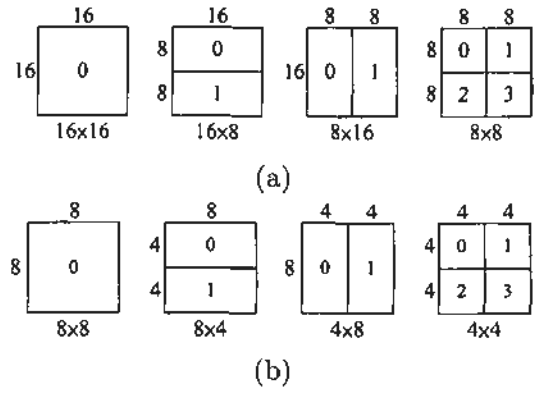


Figure 3.6: *The partitions of (a) MB and (b) sub-MB for MCP*

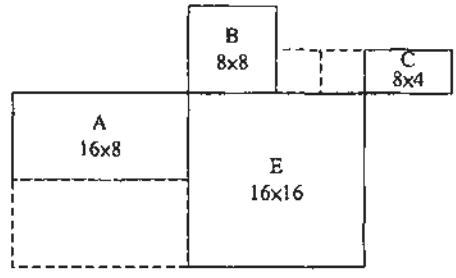


Figure 3.7: *The adjacent MC blocks involved in MV prediction*

shown in Fig. 3.7, E is the current block, of which the MV is to predict, and only three adjacent blocks, A, B, and C, are involved to predict the MV of E. If the size of E is neither 16×8 nor 8×16, the median of the MVs of A, B, and C is used as MVP. For 16×8 MB partition, MVP for the upper 16×8 block is B's MV and MVP for the lower 16×8 block is A's MV. For 8×16 MB partition, MVP for the left 8×16 block is A's MV and MVP for the right 8×16 block is C's MV. In case A, B, and C in Fig. 3.7 are not all available, e.g., outside the current slice or in intra-mode, the derivation of MVP will be modified accordingly.

Interpolation for MCP

For luminance components, the accuracy of MCP is in units of 1/4 distance between pixels, so the resolution of MV is up to 1/4-pixel. As shown in Fig. 3.8, 15 types of sub positions, labeled with lowercases a-o, in between the integer pixels, labeled with capital letters, are needed to be interpolated for ME and MCP. The values of half-pixel sub positions are obtained by sequentially applying a 1-D 6-tap FIR filter horizontally and vertically, whilst the values of 1/4-pixel sub positions are interpolated using bilinear

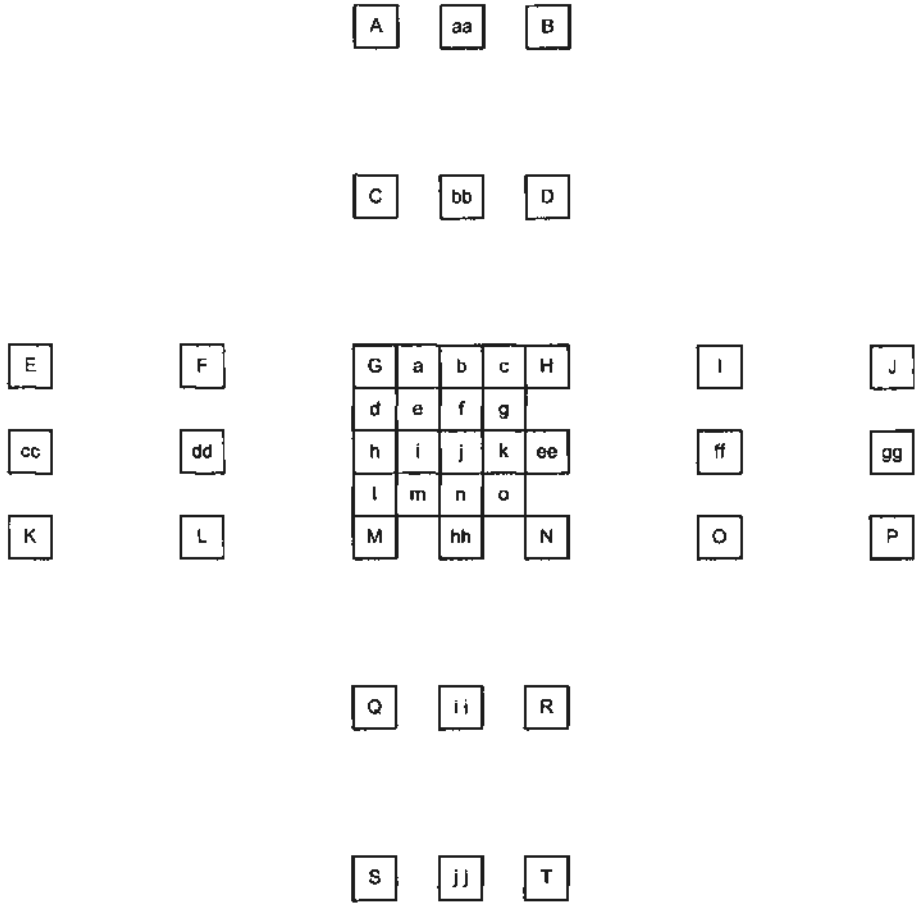


Figure 3.8: *The sub position pixels to be interpolated and the supporting integer pixels*

filter supported by integer pixels and the interpolated half-pixel values.

Before the half-pixel sub positions b and h are interpolated, intermediate values b_1 and h_1 are first calculated by applying the 6-tap filter in (3.1) and (3.2), respectively.

$$b_1 = (E - 5F + 20G + 20H - 5I + J) \tag{3.1}$$

$$h_1 = (A - 5C + 20G + 20M - 5Q + S) \tag{3.2}$$

The values of b and h are obtained by (3.3) and (3.4), respectively.

$$b = (b_1 + 16) >> 5 \tag{3.3}$$

$$h = (h_1 + 16) >> 5 \tag{3.4}$$

The other half-pixel sub position j is obtained by (3.5),

$$j = (cc - 5dd + 20h_1 + 20ee - 5ff + gg + 512) >> 10 \tag{3.5}$$

where the intermediate values, denoted as cc, dd, ee, ff and gg, are obtained in the same way to h_1. The 1/4-pixel sub positions, a, c, d, f, i, k, l, and n, are derived by averaging with upward rounding of the two nearest samples at integer and half-pixel positions, for example, as in (3.6).

$$a = (G + b + 1) >> 1 \tag{3.6}$$

The 1/4-pixel sub positions, e, g, m, and o, are derived by averaging with upward rounding of the two nearest samples at half-pixel sub positions in the diagonal direction, for example, as in (3.7).

$$e = (b + h + 1) >> 1 \tag{3.7}$$

The sub positions in between chrominance pixels are interpolated by bilinear filtering. Assuming 4:2:0 color subsampling format, 1/4-pixel MCP resolution for luminance component corresponds to 1/8-pixel resolution for chrominance components.

Weighted Prediction

In all the previous video coding standards, the target block is directly copied for MCP, or in B-mode, the MCP is the average of the preceding and succeeding target blocks. Such a prediction method is supported by H.264/AVC by default. In addition, H.264/AVC introduces weighted prediction, where the samples of the target blocks are scaled by weighting factors before used for MCP. The weighting factors, w_0 and w_1, for the target blocks from list 0 and list 1, respectively, can be customized by encoders and transmitted in the slice header, or implicitly derived specially for B-mode in the inverse proportion to the relative temporal distance from the current picture to the reference pictures. Weighted prediction can adaptively control the relative contributions from target blocks and is especially useful for scene transition and illumination change.

Transform and Quantization

H.264/AVC is the first coding standard employing an Integer Cosine Transform (ICT) instead of DCT that allows implementation approximations within some specified tolerances [34] (replaced by [35] in 2006). An advantage of using ICT is that, with an exact integer inverse transform, mismatch between the encoder and the decoder is avoided.

The intra- or inter-prediction error of the luminance pixels in an MB may be split into 4×4 or 8×8 blocks, to which 2-D order-4 and order-8 ICTs are applied, respectively. The two transforms can be adaptively selected for luminance prediction error at the MB level, but chrominance pixels are all transformed by the 2-D order-4 ICT. The two transform matrices are given as below.

$$T_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{3.8}$$

$$T_8 = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \tag{3.9}$$

Given 8-bit input video signal, the transforms can be easily implemented using 16-bit arithmetic. Especially, the 4×4 transform is so simple that can be implemented using just a few additions and bit shifts.

As mentioned above, INTAR_16×16 prediction modes are intended to code smooth areas and chrominance components are usually smooth over large areas. For that reason, the DC coefficients of the sixteen transformed 4×4 luminance blocks in the MB coded by INTAR_16×16 are extracted to form a 4×4 block, which is transformed by a secondary WHT. Similarly, the DC coefficients of chrominance blocks with all MB types are also transformed using a secondary WHT, which is 2-D order-2 for 4:2:0 color format. Fig. 3.9 illustrates such a hierarchical transform procedure for 4:2:0 videos, where the index labeled in each block shows the coding order.

After transform, quantization is applied to meet the target bit-rate. By default, the coefficients in an MB are quantized by the common stepsize determined by QP, an integer ranging from 0 to 51. The value of QP is not linearly related to the stepsize as in all previous standards but influences the stepsize in such a way that the stepsize increases about 16% for one increament and exactly doubles for every six increments.

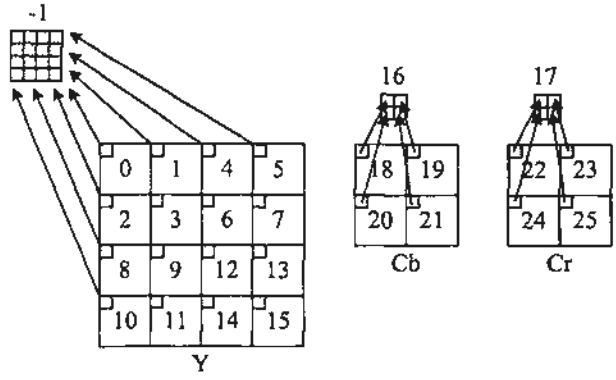


Figure 3.9: *The hierarchical transform procedure for 4:2:0 videos and the block coding order*

Rather, the value of QP is linearly related to the actual bit-rate.

Besides the default one, H.264/AVC also supports perceptual-based quantization, which has been a mainstay of prior use in MPEG-2. H.264/AVC defines a default quantization matrix, but also allows an encoder to specify customized quantization matrices and send them at the sequence or picture level. Such perceptual-based quantization typically does not improve objective quality measured by MSE, but does improve subjective quality, which is eventually more important.

Entropy Coding

In H.264/AVC, two entropy coding modes are supported. One is a Huffman-like coding, which means each symbol is converted to a codeword with integer number of bits, by look-up tables or dynamic codeword construction. The other is the context-adaptive binary arithmetic coding (CABAC). The two modes are switchable at the picture level and the SEs at the higher levels are all coded by FLC or VLC with Exp-Golomb codes. In the mode of Huffman-like coding, the SEs other than the residual data are also coded using Exp-Golomb codes, while the residual data are coded using context-adaptive VLC (CAVLC). In the following, the Exp-Golomb codes, CAVLC, and CABAC are introduced, respectively.

Exp-Golomb Codes Exp-Golomb codes have a generic form of [M zeros][1][INFO] (see Table 3.1), where INFO is an M-bit field carrying information. Hence, the Exp-Golomb codebook can be constructed in a logical way and is conceptually infinite in length. With Exp-Golomb codes, the standard only defines the mapping from the SE

codeNum	Codeword
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

Table 3.1: *The Exp-Golomb codes*

value to the look-up index, an unsigned integer denoted as codeNum (see Table 3.1), according to the data statistics.

CAVLC CAVLC is originally designed for coding 4×4 transform blocks. The coding process is in the reverse zig-zag order, because the trailing non-zero coefficients have high probability to be ±1, whereas the values of leading non-zero coefficients are more random. The SEs necessary to represent all the information of a block include the numbers of non-zero coefficients and zeros before the last non-zero coefficient, denoted as total_coeff and total_zeros, respectively, the number of successive ±1 coefficients at the end of the coefficient sequence, denoted as T1s, and their signs, the levels of coefficients other than T1s, and the number of successive zeros before each coefficient, denoted as run_before. For each SE except the sign of T1s, which is a 1-bit flag, conditional coding is employed, which means multiple VLC tables are designed for different conditioning states. The switching of these tables depends on the local activities, i.e., the context.

The SE total_coeff and T1s are jointly coded. As shown in the context template (see Fig. 3.10), total_coeff in A and B are used as the context to select an appropriate codebook. For coding total_zeros, the choice of the codebook depends on total_coeff in the current block, since the summation of the two will not exceed 16. Large total_coeff implies small total_zeros and vice versa. For run_before, the choice of the table depends on the number of zeros that have not yet been processed, with which the range of run_before decreases. The levels of each non-zero coefficients are coded by seven conceptual tables indexed from 0 to 6. The tables in order of ascending indices cater

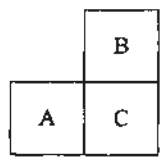


Figure 3.10: *The context template*

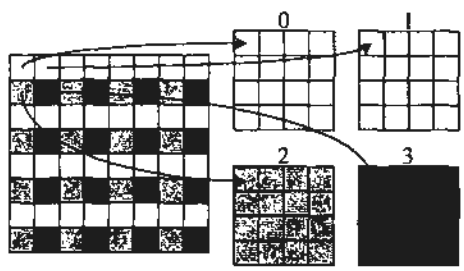


Figure 3.11: *Separate coefficients of an 8×8 transform block into four 4×4 transform blocks for 8×8 CAVLC*

for seven types of contexts from the high frequency coefficients to low frequency coefficients. The switching among the tables is determined by the maximum magnitude of the coded levels in the current block.

No special CAVLC scheme is designed for coding 8×8 transform blocks. The 64 coefficients have to be separated into four 4×4 blocks (see Fig. 3.11), which are successively coded using CAVLC.

CABAC is used for coding a broader range of SEs than CAVLC, including slice header, MB header, and residual data. Compared to CAVLC, CABAC typically provides a bit-rate reduction between 5%-15%, although much more computationally complex. The improvement is mainly contributed by three design aspects. Firstly, arithmetic coding allows one to assign a non-integer number of bits per symbol, which is especially beneficial to symbol probabilities greater than 0.5. Secondly, the context modeling allows each SE to have more than one probability models to estimate the conditional probability distributions for different local activities, i.e., different contexts. Thirdly, the probability models can be updated to keep track of the actual statistics experienced during the coding process. The diagram of CABAC is depicted in Fig. 3.12.

The first step is binarization. Since CABAC uses binary arithmetic coding, a non-binary valued SE should first be converted into a binary string. Note that the binary string still contain much statistical redundancy and is not part of the bitstream. It

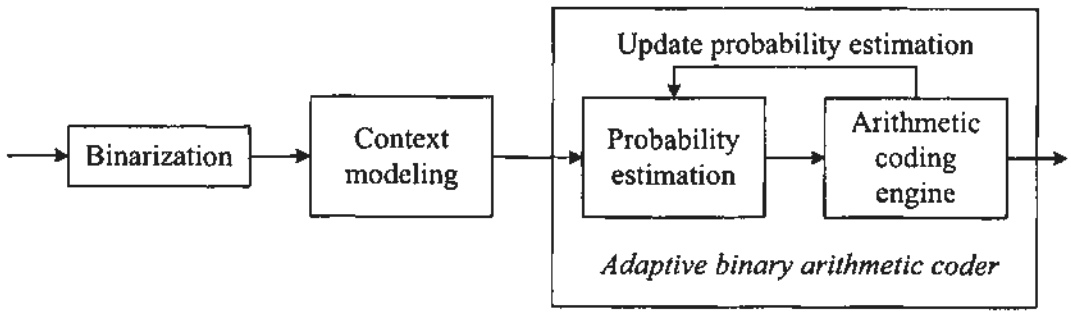


Figure 3.12: *Generic block diagram of the CABAC entropy coding scheme*

is only the input to the arithmetic coding engine, of which the output is the compact bitstream.

In the second step of context modeling, each binary decision in a binary string, called "bin", is assigned an appropriate context model, which estimates the probability distribution of the bin. A context model records two types of information. One is the most probable symbol (MPS), which is either 0 or 1. The other is the probability of the least probable symbol (LPS) P_{LPS}, which ranges from 0 to 0.5 and is discretized into 64 probability states rather than continuous. In case a context model is used for a single bin or is shared by several bins, there is no need to do the context model selection. Otherwise, the selection among available models is made according to the values of the relevant SEs in context template (see Fig. 3.10). After context modeling, each bin as well as its probability distribution described by the associated context model are input to the arithmetic coding engine.

The third step is the arithmetic coding, of which the concept has been introduced in Section 2.5.1. Conventional arithmetic coding is based on the principle of recursive interval subdivision, which introduces multiplications. In H.264/AVC, the arithmetic coding is specified as a multiplication-free low complexity method using only shifts and table look-ups, mainly facilitated by the following three properties.

1. Probability estimation is performed by a transition process between 64 probability states for LPS.

2. The range R representing the current state of the arithmetic coder is quantized to a small range of pre-set values before calculating the new range at each step, making it possible to calculate the new range using a look-up table.

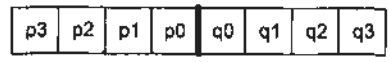


Figure 3.13: *Pixels on either side of a vertical boundary of adjacent blocks P and Q*

3. A simplified coding process is defined for SEs with a near-uniform probability distribution, in which the context modeling is bypassed,

Each context model has an initial probability estimate defined by the standard and is reset at the beginning of every slice. After the coding of each bin, the context model that has just been used is updated depending on the coded bin. Such an update behavior will change the probability state or even cause the transition between LPS and MPS.

In-loop Deblocking Filter

In-loop deblocking filters, as the name implies, is adopted to reduce the blockiness introduced by discontinuity of motion and block-based transform and quantization. In H.264/AVC, it is brought within the MCP loop (see LF_1 in Fig. 2.4), which leads to not only better subjective and objective qualities but also improved capability to predict other pictures. The deblocking filter is applied to the vertical and horizontal boundaries of 8×8 and 4×4 transform blocks. Each filtering operation affects up to three pixels on either side of the boundary. Fig. 3.13 shows four pixels on either side of a vertical boundary in adjacent blocks P and Q, where the values of $p0$, $p1$, $p2$ and $q0$, $q1$, $q2$ may be changed by filtering. There are three steps for filtering one block.

First of all, the parameter, boundary strength (BS), is calculated for each boundary, ranging from 0 (no filtering) to 4 (strongest filtering). The value of BS implies the possible extent to which the two adjacent blocks suffer from the blockiness and thus is used to select an appropriate filter. Fig. 3.14 shows the BS determination procedure used in progressive video coding, which depends on the boundary location, the coding mode (intra or inter), the existence of non-zero coefficients, and the motion information. In the figure, refP and refQ are the reference information for P and Q, including the direction (list 0, list 1, or bi-prediction) and the reference picture for each direction. "DMVX≥1" means the horizontal distance of the P and Q's target blocks in the same reference picture is greater than or equal to one integer pixel. For coding interlaced videos, the concept is similar but more logical decisions are involved.

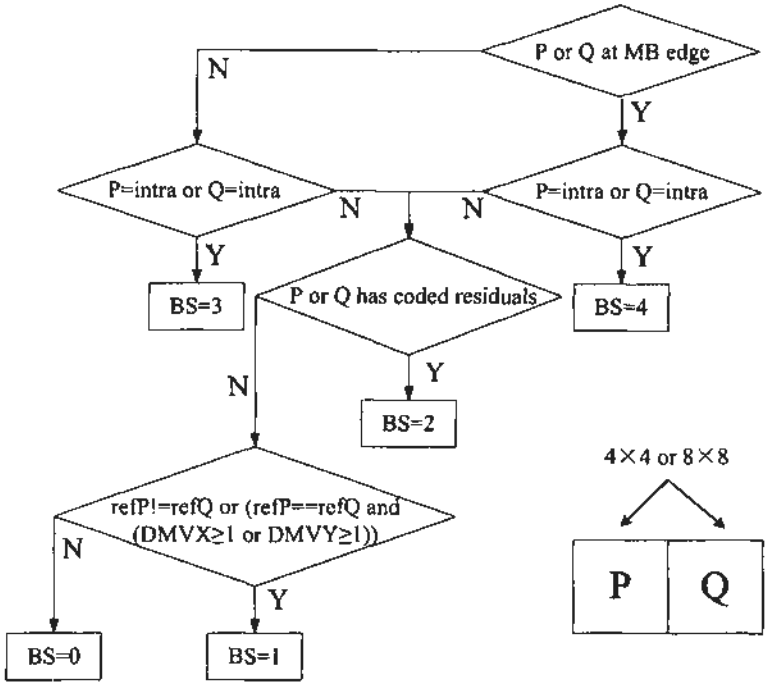


Figure 3.14: *BS determination procedure used in progressive video coding*

Secondly, after the filter has been selected for a block according to BS, whether to apply the filter is left to be decided for each set of pixels across the boundary (see Fig. 3.13). The filter decision is made by thresholds $\alpha(x)$ and $\beta(x)$, which decrease with the index x. By default, x is the average of the QPs used in P and Q, but x can be adjusted by adding an offset to the average QP. The filtering of $p0$ and $q0$ only takes place if the following condition is true,

$$|p0 - q0| < \alpha(x) \quad \text{and} \quad |p1 - p0| < \beta(x) \quad \text{and} \quad |q1 - q0| < \beta(x) \tag{3.10}$$

where $\beta(x)$ is considerably smaller than $\alpha(x)$, and filtering of $p1$ and $q1$ takes place if the following condition is true.

$$|p2 - p0| < \beta(x) \quad \text{and} \quad |q2 - q0| < \beta(x) \tag{3.11}$$

The basic idea is that a relatively large absolute difference between pixels near a block boundary implies blockiness and should therefore be reduced. However, if the magnitude of that difference is so large that it cannot be explained by the coarseness of the quantization, the boundary is more likely to reflect the actual edge in the source picture and should be preserved.

Finally, the filtering operation is applied to where is necessary. The filter varies with BS values, pixel positions, and color components. More details can be found in the standard itself [2].

Designs for 4:4:4 Color Format

As the 4:4:4 color format is usually used in the most demanding video applications, special requirements, such as very high or even lossless fidelity, should be fulfilled. H.264/AVC supports two lossless coding modes, the PCM mode and the transform-bypass mode. The former is developed only to enable the functionality of lossless coding, regardless of coding efficiency, whereas the latter uses intra and inter prediction to remove the spatial and temporal redundancy in video signal and the prediction error is entropy coded with transform and quantization bypassed.

Furthermore, in contrast to 4:2:0 color format, chrominance components are represented in full spatial resolution and thus should be coded using the techniques as powerful as those for luminance components in order to improve the overall performance. The three color planes in 4:4:4 videos can share the common slice structure and MB coding parameters as for 4:2:0 videos. Alternatively, each color plane can be treated as a independent monochrome video picture and coded individually with its own slice segmentation and mode selections using the powerful coding tools for luminance components. Furthermore, such a color plane separated coding scheme provides an enhanced parallel processing solution for the encoder.

Features for Network Transportation

Robustness to data errors and losses and flexibility for operation over a variety of network environments are enabled by a number of new design aspects in H.264/AVC, including parameter sets, flexible MB ordering (FMO), arbitrary slice ordering (ASO), redundant pictures, data partitioning, and switching P- (SP-) and switching I- (SI-) pictures. Details of these features are not introduced here, because they have no direct relevance to this work. Interested readers are referred to [36] and [37] for more in-depth exposition.

R-D Optimized Mode Decision

Besides efficient coding tools, another factor directly influencing the performance of an encoder is coder control, which determines a combination of coding tools and the corresponding set of coding parameters such that a certain R-D performance subject to the the coding constraint is achieved. Coder control is not defined by the standard, so can be developed freely.

In JM, the coder control algorithm is based on Lagrangian optimization techniques. Let $\mathbf{S} = \{S_k | k = 0, \cdots, K-1\}$ denote a frame constituted by a set of K MBs. Let $\mathbf{I} = \{I_k | k = 0, \cdots, K-1\}$ be the set of the coding modes for the MBs in $\mathbf{S}$. The task for coder control is to minimize the overall distortion subject to a rate constraint R_c,

$$\min_{\mathbf{I}}[D(\mathbf{S}, \mathbf{I})], \quad s.\ t.\ R(\mathbf{S}, \mathbf{I}) \leq R_c \tag{3.12}$$

where $D(\mathbf{S}, \mathbf{I})$ and $R(\mathbf{S}, \mathbf{I})$ represent the total distortion and rate, respectively, resulting from the quantization of $\mathbf{S}$ with a combination of coding options $\mathbf{I}$. The constrained problem is converted to an unconstrained one using Lagrangian weighting factor λ.

$$\mathbf{I}_{opt} = \arg\min_{\mathbf{I}}[J(\mathbf{S}, \mathbf{I}|\lambda)], \quad \text{with } J(\mathbf{S}, \mathbf{I}|\lambda) = D(\mathbf{S}, \mathbf{I}) + \lambda R(\mathbf{S}, \mathbf{I}) \tag{3.13}$$

Under the assumption of additive cost measures, the joint R-D cost for the frame is determined by finding the minimum R-D cost for each MB as below.

$$\min_{\mathbf{I}}[J(\mathbf{S}, \mathbf{I}|\lambda)] = \min_{\mathbf{I}} \sum_{k=0}^{K-1} J(S_k, \mathbf{I}|\lambda) = \sum_{k=0}^{K-1} \min_{I_k} J(S_k, I_k|\lambda) \tag{3.14}$$

The assumption of additive cost measures is reasonable, especially for inter MBs. For intra MBs, where spatial prediction is employed, the assumption of independent additive R-D cost is violated, because the decision of a prediction mode for an encoded block/MB influences the decision in the succeeding blocks. Nevertheless, such dependence is neglected in this R-D optimized mode decision compared with the computationally complexity of the mode search including the preceding blocks.

The distortion is measured by (3.15),

$$D(S_k, I_k) = \sum_{x_k \in S_k} |x_k - \hat{x}_k|^p \tag{3.15}$$

where x_k and $\hat{x}_k$ denote the values of the pixels in S_k and their reconstructions, respectively. The distortion is measured by summation of absolute difference (SAD) with the exponent p equal to 1 or by summation of squared difference (SSD) with p equal to 2. The rate $R(S_k, I_k)$ is obtained by provisionally encoding S_k.

The value of λ depends on the distortion measure. For mode decision, SSD is used and λ_{Mode} is calculated by (3.16) [38],

$$\lambda_{Mode} = 0.85 \times 2^{\lfloor QP/3 \rfloor} \tag{3.16}$$

where $\lfloor x \rfloor$ means the largest integer value less than or equal to x. For ME, SAD is used and λ_{ME} is the square root of λ_{Mode}, i.e., $\lambda_{ME} = \sqrt{\lambda_{Mode}}$.

3.3.3 Network Abstraction Layer (NAL)

Rather than forcing a specific bitstream interface to the system as in previous video coding standards, H.264/AVC designs NAL, which enables simple and effective customization of the method of carrying the video content in a manner appropriate for a broad variety of networks.

NAL Unit (NALU)

The coded video signal is organized into logical data packets, called NAL units (NALU). Each NALU contains an integer number of bytes, where the first byte is a header and the remaining bytes contain the payload data. The header byte consists of three SEs, forbidden_zero_bit (1 bit), nal_ref_idc (2 bits), and nal_unit_type (5 bits). The nal_unit_type indicates the type of the payload in the NALU. The main NALU types and their corresponding content are shown in Table 3.2. As can be seen, NALUs are classified into two categories: VCL and non-VCL. The former comprises the data that represent the samples in the video frames, whilst the latter comprises any associated additional information, such as parameter sets and supplemental enhancement information (SEI) that enhances the usability of the decoded video signal but does not affect the normative decoding process. The nal_ref_idc indicates the reference capability of the payload. If the payload is referenced for decoding other NALUs, the value of nal_ref_idc is not equal to 0, but no specific value is assigned in the standard. Otherwise, the value of nal_ref_idc equals 0.

nal_unit_type	Content of NALU	NALU type class
1	Coded slice of a non-IDR picture	VCL
2	Coded slice data partition A	VCL
3	Coded slice data partition B	VCL
4	Coded slice data partition C	VCL
5	Coded slice of an IDR picture	VCL
6	Supplemental enhancement information (SEI)	non-VCL
7	Sequence parameter set (SPS)	non-VCL
8	Picture parameter set (PPS)	non-VCL
9	Access unit delimiter	non-VCL
10	End of sequence	non-VCL
11	End of stream	non-VCL

Table 3.2: *Types of NAL units*

NALU Stream

A series of NALUs generated by an encoder is referred to as a NALU stream, which may contain one or more coded video sequences. Each coded video sequence can be decoded independently, given the necessary parameter sets, which are packeted by NALUs and conveyed "in-band" or "out-of-band". A coded video sequence consists of a series of access units. Decoding of each access unit results in one decoded picture. At the beginning of a coded video sequence is an instantaneous decoding refresh (IDR) access unit that represents an I-picture. The presence of an IDR access unit indicates all following coded pictures in decoding order can be decoded without inter prediction from any picture decoded prior to the IDR picture. Fig. 3.15 shows the format of an access unit. Each access unit contains a primary coded picture consisting of a set of VCL NALUs. It may also be prefixed with an access unit delimiter to aid in locating the start of the access unit. Some SEI may also precede the primary coded picture. Following the primary coded picture may be some additional VCL NALUs that contain redundant representations of areas of the same video picture, i.e., the aforementioned redundant pictures. Finally, if the access unit is the last one of a coded video sequence and/or the entire NALU stream, the NALUs with nal_unit_type equal to 10 and 11 are presented for indication.

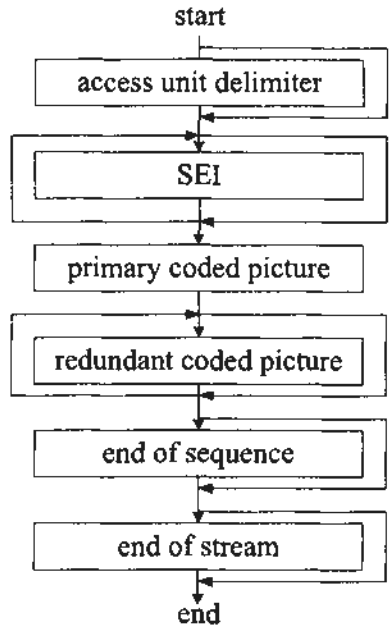


Figure 3.15: *The structure of an access unit*

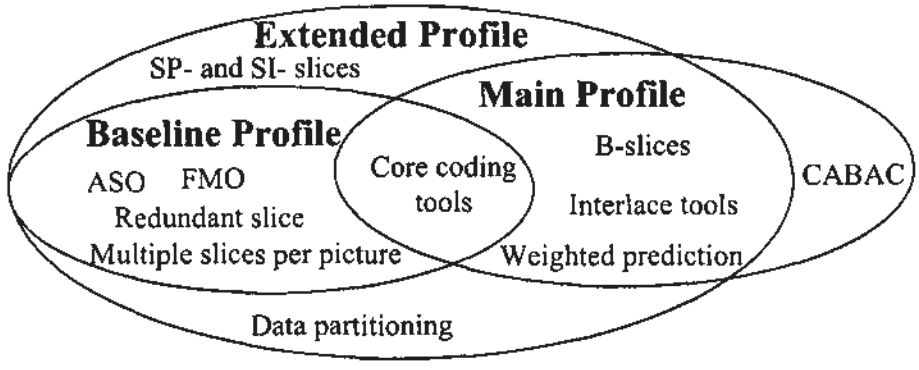


Figure 3.16: *The coding tools in H.264/AVC profiles*

3.3.4 Profiles and Applications

As introduced in Section 3.1, a profile defines a set of coding tools used in generating a conforming bitstream. Well defined profiles and levels facilitate interoperability between various applications that have similar functional requirements. In the initial version of H.264/AVC finalized in 2003, only three profiles were supported: Baseline, Main, and Extended Profiles. Fig. 3.16 shows the coding tools included in these profiles, where the core coding tools cover I-slices, P-slices, intra prediction, variable MC block sizes, 1/4-pixel MCP, MV prediction, multipicture reference, 4×4 ICT as well as the hierarchical procedure, in-loop deblocking filter, and CAVLC.

The Baseline Profile includes the core coding tools and some error resilience tools. It aims at video conversational services operating typically below 1 mbits/s with low latency requirements, such as video telephony, video conferencing, and wireless communications. The Main Profile includes more efficiency-oriented tools to favor entertainment-quality consumer applications operating from 1 to 8 mbits/s with moderate latency, such as broadcast, video-on-command (VOD) via various channels, and high-density storage media. The Extended Profile is a superset of the Baseline Profile, providing more efficient switching between coded bitstreams and improved error resilience. It may be particularly useful for streaming services over the Internet, which typically operates at 50-1500 kbits/s and tolerates higher latency.

The union of these three profiles covers all the coding tools in the initial version of H.264/AVC, but excludes those in the FRExt, developed from 2003 to 2007. The FRExt produces totally eight profiles, collectively named high profiles. Table 3.3 compares four high profiles, which build upon the Main Profile. As can be seen, the difference in capability among high profiles is primarily in terms of supported bit depths and chrominance formats. New coding tools further enhancing the coding efficiency are also included, of which some are specially designed for videos with 4:4:4 color format.

The other four profiles not shown in Table 3.3 are defined for all-intra coding. Among them, High 10 Intra, High 4:2:2 Intra, and High 4:4:4 Intra Profiles provide the same features and capabilities as High 10, High 4:2:2, and High 4:4:4 Profiles, respectively, except disabling all inter prediction modes. The last profile, CAVLC 4:4:4 Intra Profile is similar to High 4:4:4 Intra, but the entropy coding is restricted to CAVLC to reduce the complexity.

High profiles are intended to support high quality applications, such as studio camcorders, professional digital video recording and editing systems, digital cinema/large-screen digital imagery, and high fidelity display systems. The four all-intra coding profiles are suitable for easy editing of bitstreams. Furthermore, it is anticipated that the High Profile (see the second column in Table 3.3) will overtake the Main Profile in the near future as the primary choice for entertainment-quality applications, since the former significantly outperforms the latter without adding much implementation complexity.

Features	High	High 10	High 4:2:2	High 4:4:4 Predictive
Main Profile Tools	x	x	x	x
8×8/4×4 adaptive transform	x	x	x	x
Perceptual-based quantization	x	x	x	x
Separate Cb/Cr QP control	x	x	x	x
Color plane separated coding				x
Residual color transform				x
Intra residual lossless coding				x
4:2:0 color format	x	x	x	x
4:2:2 color format			x	x
4:4:4 color format				x
8-bit bit depth	x	x	x	x
10-bit bit depth		x	x	x
14-bit bit depth				x

Table 3.3: *Comparison of the features in four high profiles of H.264/AVC*

3.4 Audio and Video Coding Standard of China (AVS)-Video

AVS is a suite of standards, including system (Part 1), video (Part 2 and Part 7), audio (Part 3), digital copyright management (DRM) (Part 6), and other supporting standards. The target applications lie in the pivotal fields of Chinese information industry, such as HDTV broadcast, high-density storage media, wireless broad-band multimedia communication, and multimedia streaming. AVS is developed by AVS workgroup, a Chinese standard body established by National Information Industry Ministry in June 2002. This section introduces the technical designs in AVS-Video in Section 3.4.1, and then the profiles and applications in Section 3.4.2.

3.4.1 Technical Highlights

AVS-Video is essentially similar to H.264/AVC, but designed to reduce the implementation cost. Compared with H.264/AVC, some coding tools providing functionalities rather than improving coding efficiency, such as FMO, data partitioning, and SP-/SI-pictures, are not included. To make the design more succinct, the flexibilities of the necessary coding tools, such as intra and inter predictions, reference picture management, and deblocking filter, are greatly reduced. A summary of the differences between H.264/AVC and AVS-Video is given in Table 3.4. As shown in the table, AVS-Video includes two parts, Part 2 [39; 40; 41] and Part 7 [42]. This sub-section highlights the innovations in Part-2 only, which is more relevant to HD video coding. Detailed

Features	AVS Part 2	AVS Part 7	H.264/AVC
Number of profiles	3	1	11
Color format	4:2:0,4:2:2,4:4:4	4:2:0	4:2:0,4:2:2,4:4:4
Bit depth	8	8	up to 14
Picture type	I,P,B	I,P	I,P,B,SP,SI
Interlace handling	PAFF	N/A	PAFF,MBAFF
Luminance intra prediction	8×8 5 modes	4×4 9 modes	4×4,8×8,16×16 totally 22 modes
No. of reference frame	up to 2	up to 2	up to 16
Reference picture management	background- and core-picture, non-reference P	adaptive sliding window, non-reference P	MMCO-based commands
Prediction in B-picture	forward,backward, Symmetric,Direct	N/A	list0,list1,Direct, bi-predictive
Direct mode	temporal	N/A	spatial, temporal
MV prediction	geometry median	geometry median	median
Interpolation	1/2-pel:[-1,5,5,-1] 1/4-pel:[1,7,7,1]	1/2-pel(Hor.): [-1,4,-12,41,41,-12,4,-1] 1/2-pel(Ver.):[-1,5,5,-1] 1/4-pel: bilinear	1/2-pel: [1,-5,20,20,-5,1] 1/4-pel: bilinear
Minimum MC size	8×8	4×4	4×4
Transform	8×8 PIT	4×4 PIT	4×4/8×8 ICTs
Hierarchical transform	No	No	Yes
Default quantization	64 QPs stepsize doubles for each 8 inc.	64 QPs stepsize doubles for each 8 inc.	52 QPs stepsize doubles for each 6 inc.
Adaptive quantization	perceptual-based	No	perceptual-based
Separate CbCr QP	Yes	No	Yes
Entropy coding	8×8 CA-2D-VLC 8×8 CBAC	4×4 CA-2D-VLC	CAVLC CABAC
Deblocking filter	3 BS levels	2 BS levels	4 BS levels
Lossless coding	No	No	Yes
Weighted prediction	Yes	No	Yes
NAL	No	Yes	Yes
Parameter set	No	Yes	Yes
SEI	No	Yes	Yes
Flexible slice structure	Yes	Yes	Yes
FMO	No	No	Yes
Redundant picture	No	No	Yes
Data partitioning	No	No	Yes

Table 3.4: *Comparison of the features in AVS and H.264/AVC*

illustrations of the different applications of Part 2 and Part 7 are given in Section 3.4.2.

Special Reference Pictures Up to two reference frames are allowed for MCP, which can be used as four fields for field-mode. By default, P-pictures reference the nearest preceding reference pictures, and B-pictures reference the nearest preceding and succeeding reference pictures, both in the display order. To increase the flexibility, three special types of reference pictures are introduced.

- Core picture. Core pictures form the coarsest temporal layer in a bitstream, which reference the previous core pictures only.
- Background picture. It is an I-picture, which is expected to be coded at higher quality with noise suppressed for better background prediction. It is stored in the buffer as one of the two reference frames and replaced only upon the reception of the next background frame. This feature is beneficial to video surveillance, where the background is seldom changed, and can also be regarded as a special case of long-term picture in H.264/AVC.
- Non-reference P-picture. P-picture can be marked as non-reference at the picture header, such that temporal scalability is enabled without decoding delay.

Symmetric mode in B-pictures Four prediction modes are used in B-pictures: forward, backward, Direct and Symmetric modes, where the latter two are bi-directional. The Symmetric mode transmits only the forward MV; the backward MV is derived by scaling the forward MV according to the temporal distance. The underlying assumption is that most of the motions in a video sequence are translation and occur within a few neighboring pictures.

Pre-scaled Integer Transform (PIT) 2-D order-8 integer transform is used to match the smallest MC block size in AVS. The transform matrix in (3.17) resembles DCT more than that in H.246/AVC and has higher transform coding gain.

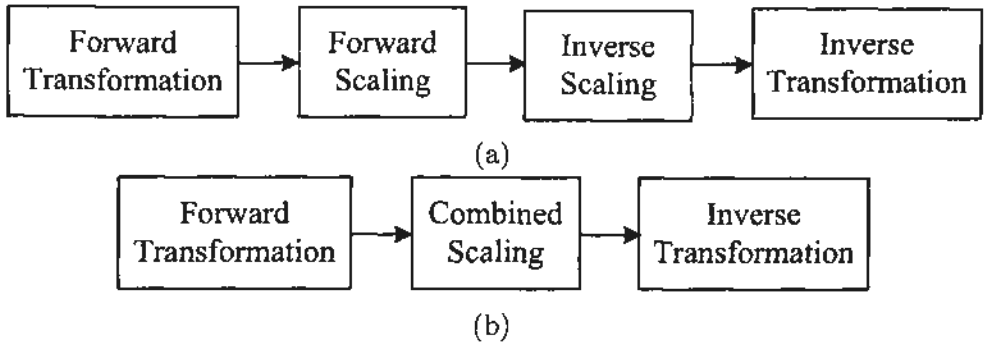


Figure 3.17: *The flow diagrams of (a) ICT and (b) PIT*

$$T_8 = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\ 10 & 4 & -4 & -10 & -10 & -4 & 4 & 10 \\ 9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\ 4 & -10 & 10 & -4 & -4 & 10 & -10 & 4 \\ 2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \end{bmatrix} \tag{3.17}$$

Compared to ICT, of which the flow diagram is shown in Fig. 3.17 (a), the transform procedure of PIT is further simplified. As shown in Fig. 3.17 (b), the inverse scaling is moved from the decoder to the encoder and combined with the forward scaling as one single process, named combined scaling. By this means, the inverse scaling is saved and thus no scaling matrix needs to be stored, whilst the complexity of encoders remains unchanged. The simplified transform is named Pre-scaled Integer Transform (PIT), and interested readers are referred to [43] for more details.

Adaptive Quantization AVS Part 2 supports perceptual-based quantization but the customization of the quantization matrix is restricted by three quantization patterns, as shown in Fig. 3.18. Each quantization pattern divides the transform block into six specific subbands and the coefficients in one subband use the common quantization weights. There are two default sets of the weights. One set {pl, pa, pb, pc, pd, ph} equal to [135, 143, 143, 160, 160, 213]>>7 is for detail protection, while the other set [128, 98, 106, 116, 116, 128]>>7 is for detail reduction. Therefore, to determine a quantization matrix, three types of information are transmitted in the picture header,

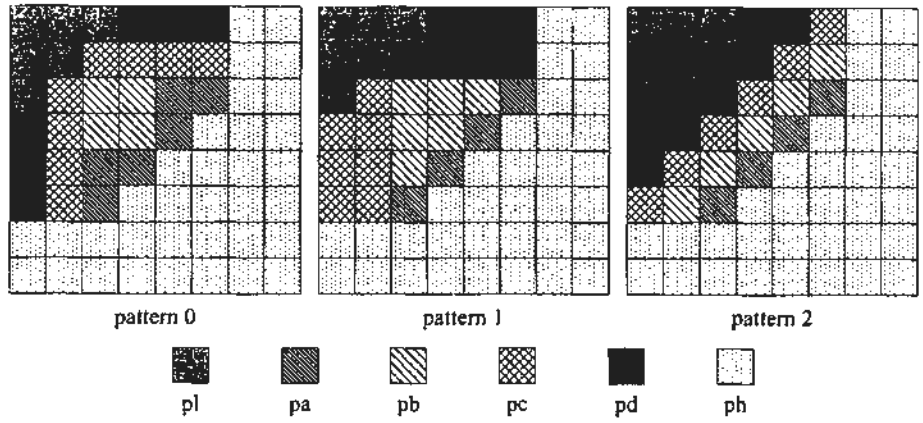


Figure 3.18: *Three AVS quantization patterns in AVS Part 2*

the choice of the pattern, the choice of the default set, and the differences between the actual weights and the selected default set if any.

Entropy Coding Like H.264/AVC, AVS Part 2 supports two entropy coding schemes, Huffman-like coding and arithmetic coding. The former also employs Exp-Golomb codes to code the SEs except residual data.

In the Huffman-like coding scheme, the residual data are coded using context-adaptive 2-D VLC (CA-2D-VLC), where the *levels* and *runs* are jointly coded as (*level*, *run*) pairs along the reverse zig-zag scan order and are ended with "EOB". Two main features distinguish CA-2D-VLC from the previous (*level*, *run*) coding schemes as in MPEG-2. Firstly, a plurality of look-up tables is defined to code one transform block. When coding a sequence of (*level*, *run*) pairs in one transform block, CA-2D-VLC uses the table indexed 0 for the first pair and switches the tables upon the detection of context changing. The context changing herein means the maximum magnitude of the levels that have been coded in the current block exceeds one of the pre-defined thresholds. There are seven thresholds defined for inter-coded luminance blocks, seven for intra-coded luminance blocks, and five for chrominance blocks, and therefore, totally 19 tables are designed for these three types of blocks. Secondly, all the codewords in tables are constructed using order-k Exp-Golomb codes, of which the aforementioned Exp-Golomb codes in Section 3.3.2 are the special cases with k=0 (see Table 3.5). Exp-Golomb codes with larger orders favor flatter-shaped p.d.f. and vice versa, but the mapping from a (*level*, *run*) pair to a codeNum only reflects a relative probability.

The arithmetic coding is named context-based arithmetic coding (CBAC), which

order-k	Codeword	codeNum Range
k=0	1	0
	$01x_0$	1-2
	$001x_1x_0$	3-6
	$0001x_2x_1x_0$	7-14
	...	...
k=1	$1x_0$	0-1
	$01x_1x_0$	2-5
	$001x_2x_1x_0$	6-13
	$0001x_3x_2x_1x_0$	14-29
	...	...
k=2	$1x_1x_0$	0-3
	$01x_2x_1x_0$	4-11
	$001x_3x_2x_1x_0$	12-27
	...	...
k=3	$1x_2x_1x_0$	0-7
	$01x_3x_2x_1x_0$	8-23
	$001x_4x_3x_2x_1x_0$	24-55
	...	...

Table 3.5: *Order-k Exp-Golomb codes*

basically has the same diagram as CABAC in H.264/AVC (see Fig. 3.12). The main difference between CBAC and CABAC lies in two aspects. Firstly, the arithmetic coding engine performs the interval subdivisions, i.e., probability update, in the logarithmic domain, where multiplication in the physical domain is equivalent to addition. Therefore, the probability update of the arithmetic coding is approximated by additions and shifts. Secondly, the transform block is coded as a sequence of (*level, run*) pairs as in CA-2D-VLC and the context selection is jointly determined by the maximum magnitude of the levels that have been coded in the current block and the position of the current level in the zig-zag order.

3.4.2 Profiles and Applications

AVS Part 2 contains three profiles: Jizhun (Base) [39], Jiaqiang (Enhanced) [40], and Shenzhan (Extended) [41] Profiles. Fig. 3.19 shows the coding tools included in different profiles, where the core coding tools are I-, P-, and B-pictures, 8×8 intra prediction, variable MC block sizes, 1/4-pixel MCP, MV prediction, two reference frames, 2-D order-8 PIT, deblocking filter, and CA-2D-VLC.

Jizhun Profile mainly focuses on digital TV applications and storage media. As

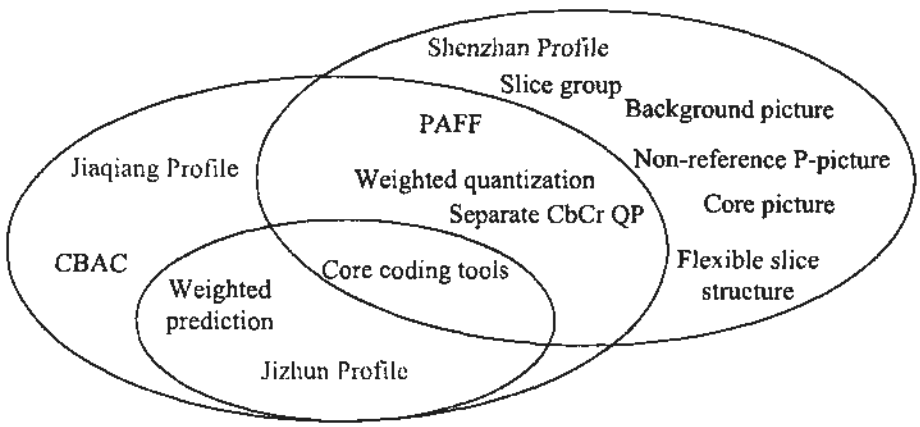


Figure 3.19: *The coding tools in AVS profiles*

the enhancement of Jizhun Profile, Jiaqiang Profile adds CBAC and adaptive quantization, in order to address the requirements of high quality video applications, such as digital cinema. Shenzhan Profile is developed specially for video surveillance applications. Features of flexible slice structure, core picture, and background picture are incorporated, which improve the capability of error resilience, temporal scalability, and friendliness to the surveillance environment.

AVS Part 7 has been developed to meet the needs of mobile video applications, such as interactive storage media, video services on wireless broad-band and packet networks. All the coding tools in AVS Part 7 constitute only one profile, named Jiben (Basic) Profile.

3.5 Summary

This chapter mainly introduces the latest video coding standards, H.264/AVC and AVS, which represent the state-of-the-art technology of hybrid video coding. H.264/AVC and AVS adopt many new technical developments, including intra prediction, integer transform, elaborate MCP modes, context-based entropy coding, and deblocking filter, and achieve a key breakthrough on R-D performance. In this work, H.264/AVC and AVS are the platforms where the proposed techniques are integrated, and their performances are the benchmarks the proposed techniques compare to.

This chapter is part of the chapter, we have submitted for publication [20], entitled "Present and future video coding standards", in the book *Intelligent Multimedia Communication: Techniques and Applications.*

Chapter 4

Analysis of HD Video Sources

4.1 Analysis of Correlation

Spatial correlation is useful for estimating the energy of the coefficients in the case of unitary transform coding, whilst temporal correlation can help to estimate the prediction error and designing the predictor for inter-frame coder [44]. The research work starts from considering the video signal as a WSS random field and comparing the spatial correlation between HD and CIF videos in Section 4.1.1. In Section 4.1.2, the video source is modeled as a separable random field as introduced in Section 2.2.2, and then the correlation comparison is made.

4.1.1 Correlation Coefficient for WSS Random Field

Given the WSS random field, the correlation coefficient of two pixels with the horizontal and vertical distances τ_h and τ_v, respectively, is denoted as $r(\tau_h, \tau_v)$, which has been defined in (2.40) and is used as the correlation measure here. Progressive video sequences with three resolutions: CIF, $720p$, and $1080p$ (see Table 1.1), are used as the source for correlation comparison. Only the luminance component of the first frame is considered. τ_h and τ_v take the integer values between -16 to 16, such that the function $r(\tau_h, \tau_v)$ reflects the reduction of correlation inside one MB. The experimental results for CIF, $720p$, and $1080p$ sequences are shown in Figs. 4.1, 4.2, and 4.3, respectively.

In general, higher spatial resolution results in higher correlation for any (τ_h, τ_v), $-16 \leq \tau_h, \tau_v \leq 16$. For each CIF sequence, the correlation coefficient increases rapidly with the decreasing distance to $(\tau_h, \tau_v) = (0, 0)$, leading to a center-sharped shape of $r(\tau_h, \tau_v)$. When $|\tau_h|$ and $|\tau_v|$ tend to 16, the correlation coefficients reduce to 0.1 to 0.4. For most $720p$ sequences, $r(\tau_h, \tau_v)$ with the (τ_h, τ_v) around $(0, 0)$ approaches 1.0 and flattens out in the whole range $[-16, 16]$. As the further evidence in Fig. 4.3, the

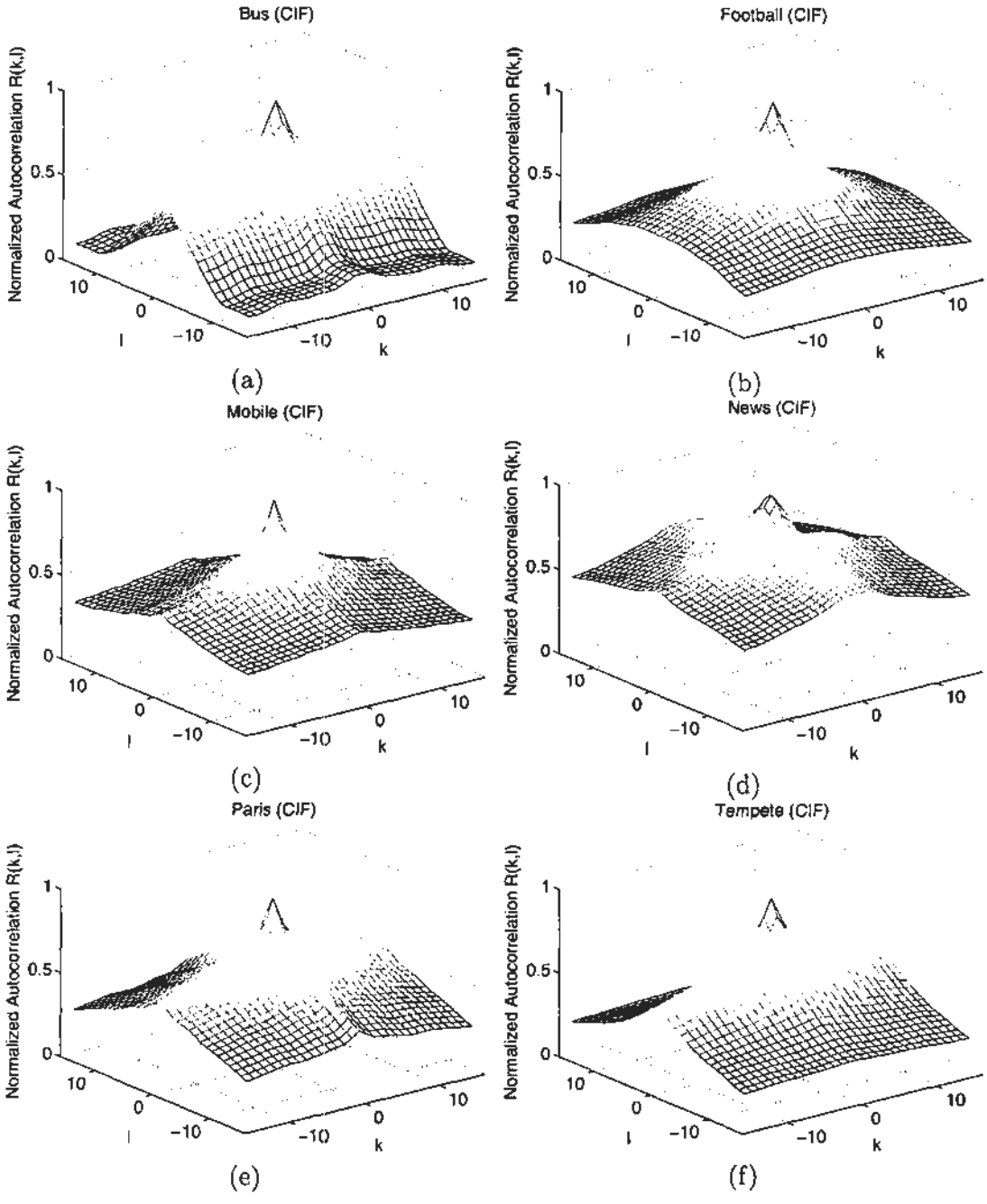


Figure 4.1: *Correlation coefficient* $r(\tau_h, \tau_v)$, $-16 \leq \tau_h, \tau_h \leq 16$, *calculated for six CIF sequences.*

correlation coefficient decreases little with the distance to $(\tau_h, \tau_v) = (0, 0)$, regardless of the video content.

On the other hand, it cannot be ignored that some $720p$ sequences, such as *City* and *Harbour*, appear less correlated than some CIF sequences, such as *News*. This phenomenon can be explained by two reasons. Firstly, HD sequences contain much

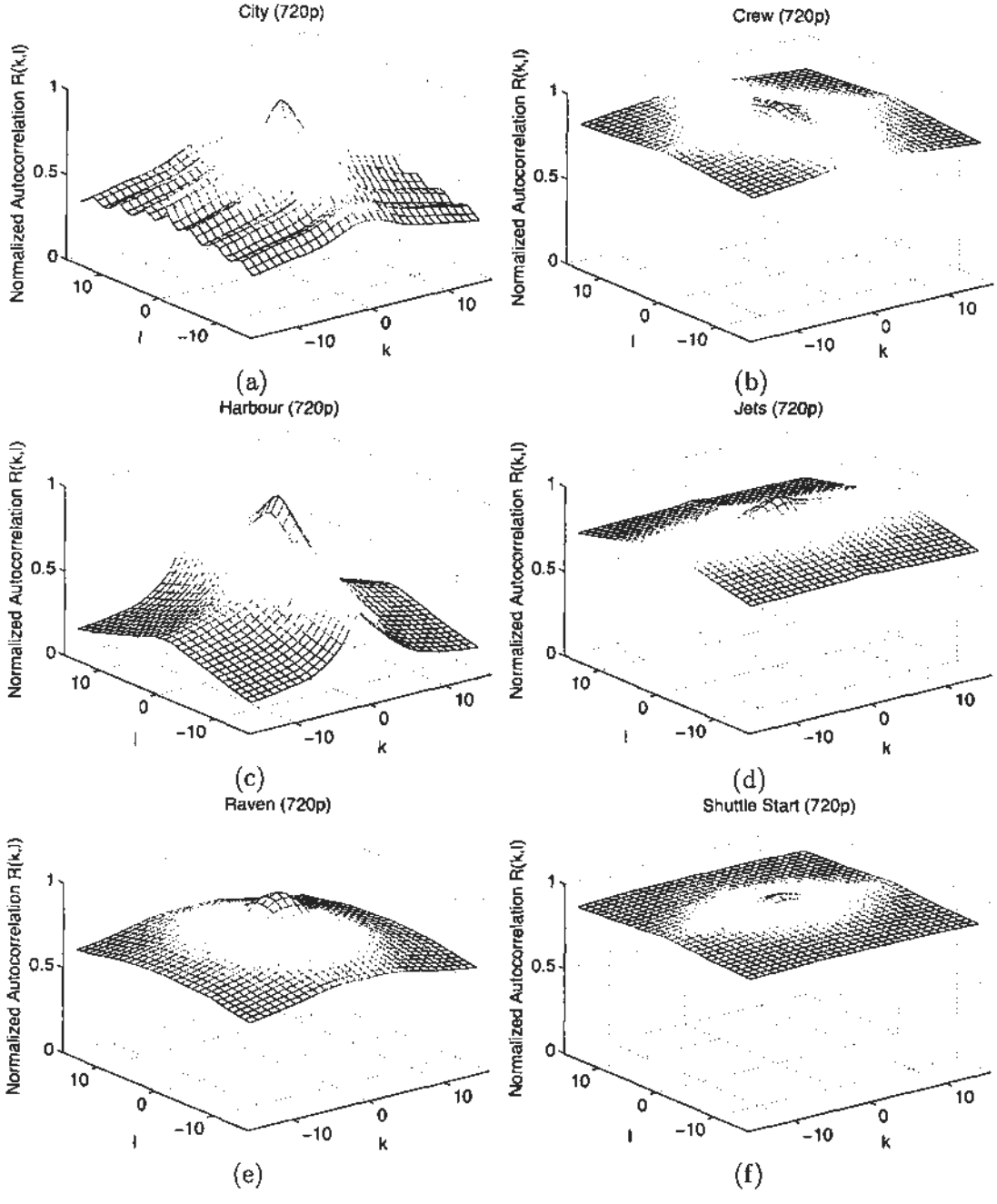


Figure 4.2: *Correlation coefficient* $r(\tau_h, \tau_v)$, $-16 \leq \tau_h, \tau_h \leq 16$, *calculated for six 720p sequences.*

more content and details than CIF ones, which are intended to reduce the correlation among pixels. However, it is usually well compensated by physically closer distance of pixels and eventually the correlation among pixels in HD videos exceeds that in CIF ones. In some extreme cases, the details and textures in HD videos are so rich that the correlation--even though compensated by higher sampling rate--is still lower than that

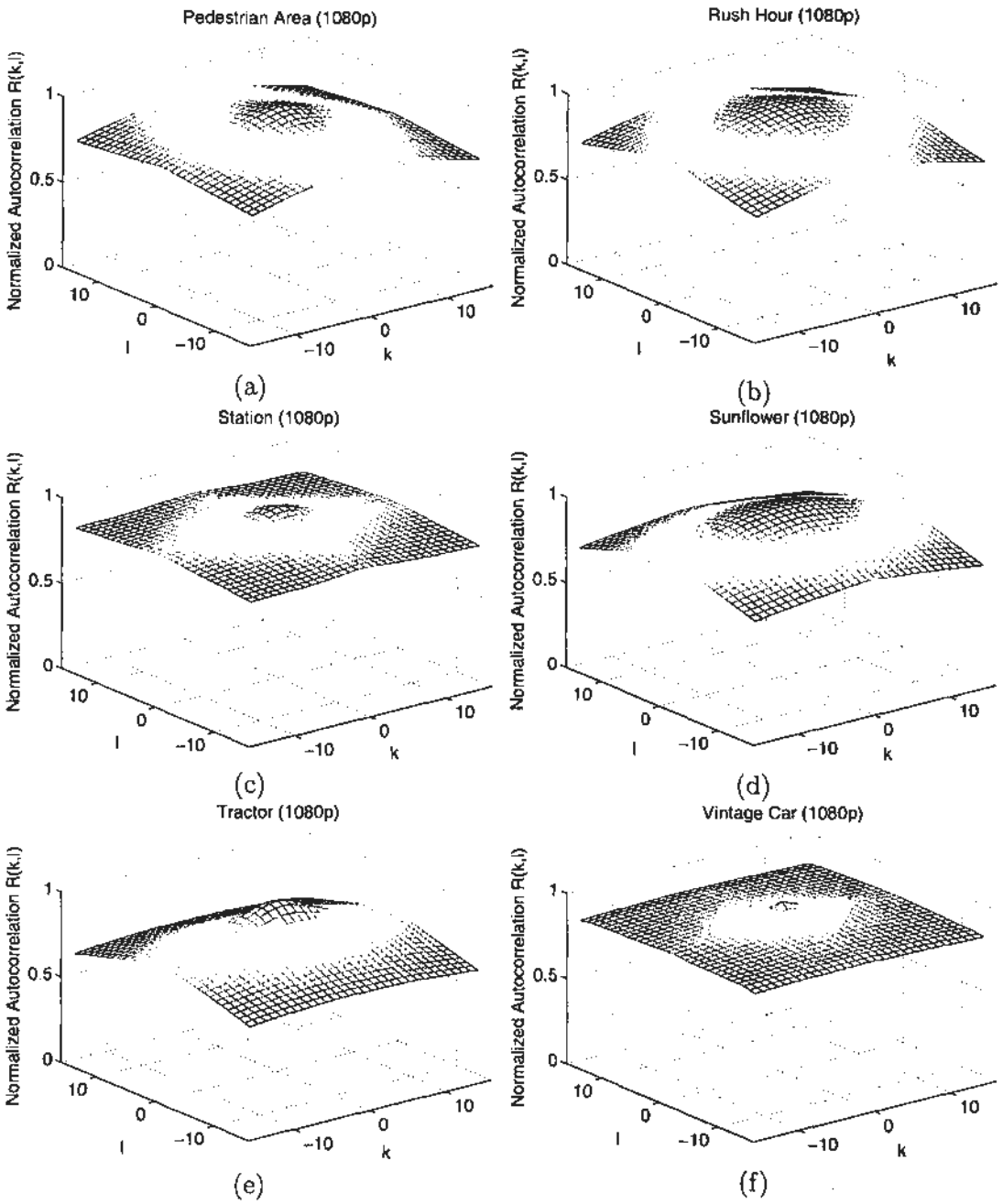


Figure 4.3: *Correlation coefficient* $r(\tau_h, \tau_v)$, $-16 \leq \tau_h, \tau_h \leq 16$, *calculated for six 1080p sequences.*

of some CIF sequences containing less details. Secondly, the intensities captured on the sampling grid are not directly stored as video source; instead, they are pre-processed by an anti-aliasing filter to ensure the cut-off frequency of the stored video signal does not exceed half of the sampling rate. The pre-filter in HD cameras and other low resolution ones are different, where the latter has more lowpass characteristics and smoothens the

Test sequence	Resolution	ρ	σ_ρ
Riverbed	HD 1920×1080	0.9894	0.0042
Flamingo		0.9882	0.0063
Kayak		0.9864	0.0065
Fireworks		0.8159	0.0441
Raven	HD 1280×720	0.9770	0.0129
Night		0.9548	0.0511
Crew		0.9758	0.0460
City		0.9513	0.0456
FlowerGarden	SD 720×576	0.9417	0.1154
F1		0.8923	0.1439
Basketball		0.9391	0.1066
Mobile		0.8836	0.1154
Foreman	CIF 352×288	0.9722	0.0712
News		0.9199	0.1259
Tempete		0.8815	0.0478
Paris		0.8656	0.1053

Table 4.1: *The correlation coefficient of two adjacent pixels in raw video sequences*

captured frames more, thus increasing the correlation among pixels.

4.1.2 Correlation Coefficient for Separable Random Field

Video signal is usually modeled as a separable AR(1) field, as introduced in Section 2.2.2, and therefore the spatial statistics of columns and rows are considered equivalent. In this sub-section, we focus on the horizontal correlation. The correlation coefficient ρ of two adjacent pixels, an important parameter for separable AR(1) field to derive the autocorrelation matrix (see Equation (2.32)), is calculated for video sequences with resolutions from CIF up to $1080p$.

In a video frame with size $M \times N$, each row is considered as a realization $x[n]$, $n = 0, 1, \cdots, N-1$, of an AR(1) random process $\mathbf{x}[n]$, $n = 0, 1, \cdots, N-1$, and one frame contains M realizations. The correlation coefficient ρ is calculated by (2.19), where τ equals one. The values of ρ for different sequences are shown in the third column of Table 4.1, which verifies the the conclusion in Section 2.2.2 that when a video signal is modeled by a separable AR(1) field, the correlation coefficient ρ is around 0.95, no matter what is the spatial resolution. Though, overall, the values of ρ in HD videos are slightly higher than those in other videos, the difference is not substantial.

In addition to studying the correlation in original video sources, we also pay attention to the prediction error, where transform coding applies, no matter in intra or

Test sequence	Resolution	ρ	σ_ρ
Riverbed	HD 1920×1080	0.8909	0.1658
Flamingo		0.8606	0.1596
Kayak		0.8596	0.1320
Tractor		0.7804	0.1504
Pedestrians		0.7772	0.1862
Station		0.6814	0.1793
Fireworks		0.5031	0.1285
Raven	HD 1280×720	0.7328	0.1174
Optis		0.6966	0.0698
Night		0.6807	0.1660
Crew		0.6712	0.1476
Harbour		0.5738	0.1859
City		0.5618	0.1080
FlowerGarden	SD 720×576	0.5774	0.1423
F1		0.5629	0.1809
Basketball		0.4912	0.2401
Mobile SD		0.3729	0.1704
Foreman	CIF 352×288	0.5445	0.2035
News		0.3543	0.1870
Tempete		0.3176	0.2353
Mobile		0.2323	0.1686
Paris		0.1770	0.2223

Table 4.2: *The correlation coefficient of two adjacent pixels in residual video sequences*

inter mode. Even for intra mode, only prediction errors are transformed thanks to the intra predictions in the recent video coding standards, such as H.264/AVC and AVS. Studying the correlation of prediction errors will influence coding strategies directly. In this experiment, the prediction errors are obtained using the criterion of R-D cost [38] and only the horizontal correlation is analyzed, as the correlation equivalence of two spatial directions still holds for prediction errors.

The third column of Table 4.2 shows the values of ρ for prediction errors. Generally, the correlation of prediction errors increases with the resolution, and the difference of ρ between HD and low resolution videos is substantial. Therefore, as evident in Table 4.2, the prediction errors of HD videos are distinguished from those of the low resolution ones by higher spatial correlation. Note that the conclusion does not hold in some extreme cases, of which the reasons have been explained in Section 4.1.1. Nevertheless, based on a large set of HD videos, the conclusion is useful and quite instructive for studying the tools specially for coding HD videos.

As one of the techniques exploiting the above property, 2-D order-16 transform is

proposed in this work. For a given bit-rate, the MSE produced by transform coding improves with the block size, as illustrated in Section 2.4.1, although order-8 transform is a good trade-off between performance and complexity. Compared with order-8 transform, order-16 transform compacts energy to a smaller proportion of coefficients, which are more likely to survive the quantization. In practice, for image or video coding, the performance of 2D order-16 transform suffers non-optimal entropy coding (typically run-length coding), as it is difficult to accurately model the distribution of a *run*, which has much larger dynamic range than that produced by an order-8 transform and dramatically varies throughout video sequences. However, based on the aforementioned observation that the prediction errors of HD videos have higher correlation, 2D order-16 transforms deserve more study as a special coding tool, as the transform coefficients have high probabilities to be distributed in the low frequency domain and thus *run* can be modeled more accurately.

In the above studies, the video signal or the prediction error signal is modeled as a separable AR(1) field, a special case of WSS random field, and therefore ρ is uniform throughout the random field. Actually, such a modeling has not been proved accurate for residual video signals, although well-suited to video sources. Intuitively, the statistics of a residual video signal should vary more than that of the video source, because in edge areas, where the prediction is sensitive to occlusion and geometric distortion, the residue has strong energy along the edge directions, whereas in smooth areas, the prediction errors are almost reduced to zero. In other words, for a residual video signal, it is inaccurate to represent the correlation by a global parameter ρ; instead, the correlation should be calculated by the definition given in (2.10) and for two adjacent pixels, it is shown in (4.1),

$$\rho(n) = r_{xx}[n, n+1] = \frac{C_{xx}[n, n+1]}{\sigma_n \sigma_{n+1}} = \frac{E\left[\mathbf{x}[n]\mathbf{x}[n+1]\right] - \eta_n \eta_{n+1}}{\sigma_n \sigma_{n+1}} \tag{4.1}$$

where $\mathbf{x}[n]$ and $\mathbf{x}[n+1]$ are the two random variables at the two adjacent positions n and $n+1$, respectively, and the correlation coefficient of $\mathbf{x}[n]$ and $\mathbf{x}[n+1]$, denoted as $\rho(n)$, replaces the single parameter ρ. As $\rho(n)$ varies with n, its standard deviations, denoted as σ_ρ, for original and residual videos are shown in the fourth columns of Tables. 4.1 and 4.2, respectively. The variance of $\rho(n)$ is small for most original videos, which verifies again that a separable AR(1) field is accurate for original videos, especially HD

videos. On the other hand, for residual video signals, the variance of $\rho(n)$ becomes larger, which shows a separable AR(1) field is no longer accurate to model residual video signals, for the correlation varies in local areas. It is also noted that in Table 4.2, $\rho(n)$ for HD and low resolution sequences have not much difference. To exploit such a property, in addition to the 2D order-16 transform for HD video coding, a richer library of transforms with smaller sizes should also be employed, such that smaller transforms are the complements to the 2D order-16 one, in order to suit more detailed areas and avoid the ringing artifacts.

4.2 Analysis of Power Spectral Density

Studying the energy distribution of HD videos in the frequency domain helps to design appropriate filters used for coding and processing. This section compares HD and other video signals in the frequency domain by power spectral density (PSD), which has been introduced in Section 2.2.1.

Given a WSS random process $\mathbf{x}$, the PSD $S_{xx}(\omega)$ is by definition the DTFT of the corresponding autocorrelation $R(\tau)$ as shown in (2.33). Extending it to a WSS random field with autocorrelation $R(\tau_h, \tau_v)$, one calculates the PSD $S_{xx}(\omega_1, \omega_2)$ by 2-D DTFT, as in (4.2).

$$S_{xx}(\omega_1, \omega_2) = \sum_{\tau_h=-\infty}^{\infty} \sum_{\tau_v=-\infty}^{\infty} R(\tau_h, \tau_v) e^{-j\omega_1\tau_h - j\omega_2\tau_v} \tag{4.2}$$

In practice, $R(\tau_h, \tau_v)$ is an estimate based on a set of video signals; the estimating procedure has been introduced in Section 4.1.1. Applying DTFT to the estimated $R(\tau_h, \tau_v)$ produces an estimated PSD, which may no longer be consistent. In this work, PSD is estimated by the periodgram of the random field, as given in (4.3),

$$\hat{S}_{xx}(\omega_1, \omega_2) = \frac{1}{MN} |X(\omega_1, \omega_2)|^2 = \frac{1}{MN} \left| \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m, n] e^{-j\omega_1 m - j\omega_2 n} \right|^2 \tag{4.3}$$

which is extended from the definition for the 1-D case, as given in (2.35). To achieve a more accurate result, a video signal, taken as a realization of a random field, is divided into frames, whose periodgrams are averaged to produce the estimated PSD.

In the experiments, progressive video sequences with three types of resolutions: CIF, 720p, and 1080p as in Section 4.1.1, are used as the source for PSD comparison

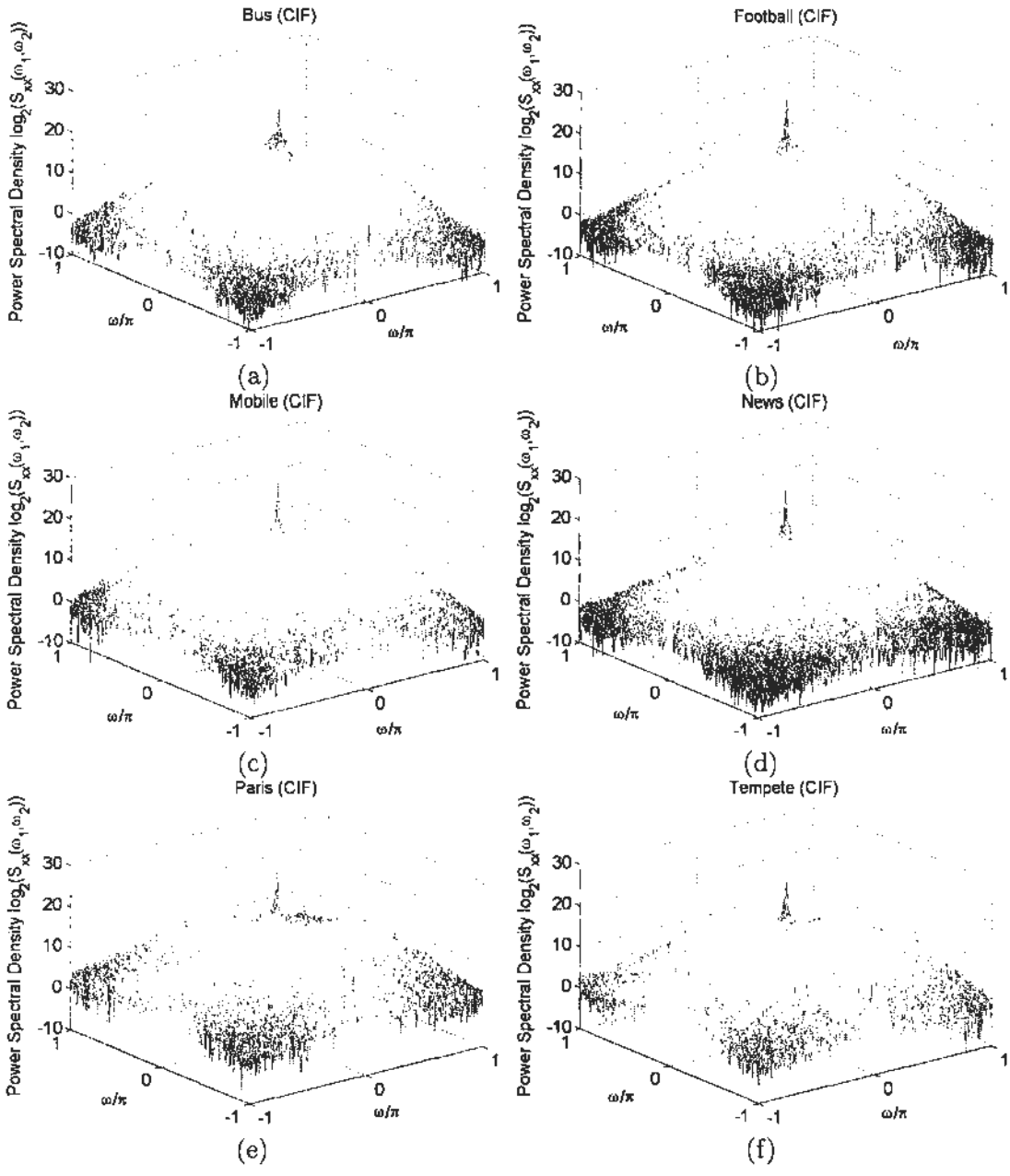


Figure 4.4: *PSD of CIF sequences*

and the results are shown in Tables 4.4, 4.5, and 4.6, respectively. For a clearer view, the magnitudes of PSD are plotted in the logarithmic domain. In each sequence, the first 30 frames are involved and only the luminance component is considered. Two differences on PSD between HD and CIF sequences can be observed. Firstly, the low frequency energy in CIF sequences is smaller than that in HD sequences. Secondly, the high frequency energies in HD sequences are mainly distributed in the horizontal and vertical directions, whereas the high frequency energies in CIF sequences are distributed

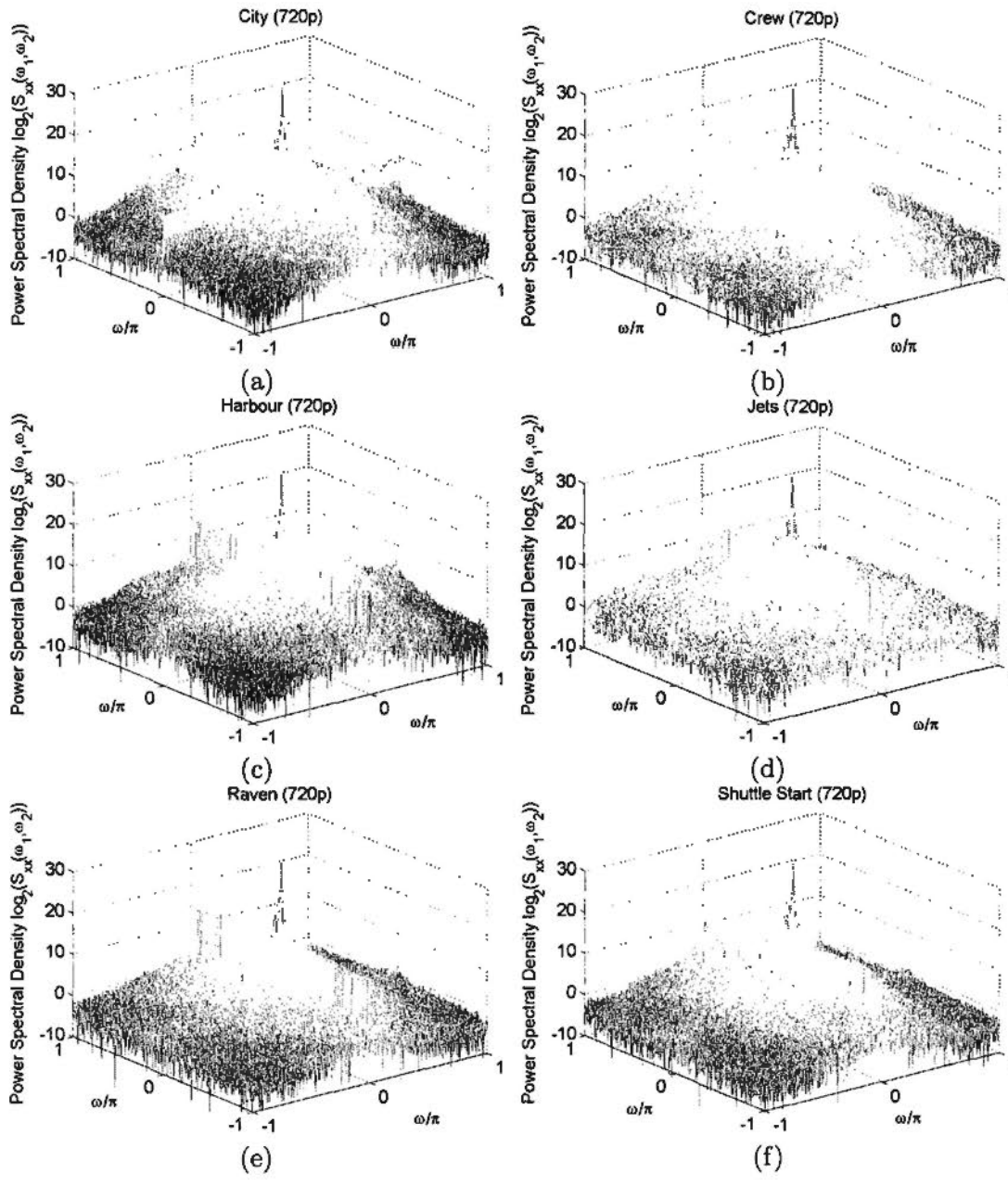


Figure 4.5: *PSD of 720p sequences*

in arbitrary directions. The unique property of energy distribution of HD sequences can be exploited to design adaptive interpolation filter (AIF) for MCP, which will be introduced in Chapter 6.

4.3 Summary

This chapter studies the unique statistical properties of HD videos. Considering the video source as a WSS random field, one can find that HD videos have higher spatial

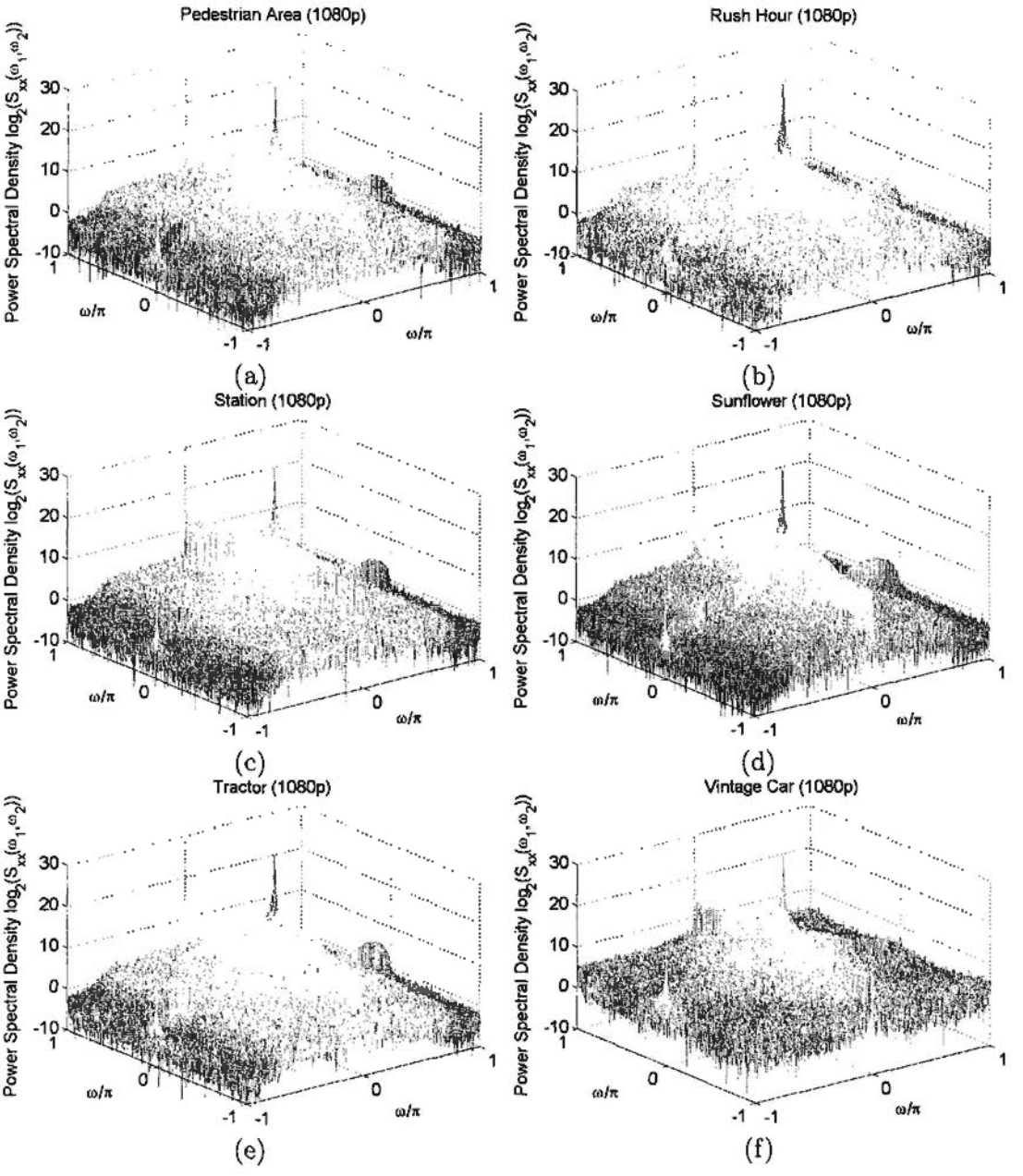


Figure 4.6: *PSD of 1080p sequences*

correlation than other low resolution videos; considering video source as a separable 2-D AR(1) field, one can find the parameter ρ, representing the correlation coefficient of two adjacent pixels in the horizontal or vertical direction, of the residual HD frames is substantially larger than that of the residual frames with lower resolution, except for some extreme cases. These findings imply that 2-D order-16 transform is an efficient coding tool especially for HD video coding. It is also pointed out that a separable 2-D AR(1) field is not well-suited to residual frames, no matter what the resolution

is, because the correlation depends on the accuracy of MCP and varies in local areas. Therefore, in HD video coding, adaptive block-size transform (ABT) should also be employed such that smaller transforms can compensate the 2D order-16 one to avoid the ringing artifacts especially in low correlated areas. Finally, the energy distribution of HD videos is compared with that of other videos by PSD. The results show that HD videos distinguish from others by higher low frequency energy and special PSD, mainly distributed along the vertical and horizontal directions. The unique property of energy distribution of HD sequences can be exploited to design AIF for MCP.

In this chapter, Section 4.1.2 is part of the regular paper published in *IEEE transactions on Circuits and System for Video Technology*, entitled "2D order-16 integer transforms for HD video coding" [45]. Section 4.2 is part of the regular paper submitted to *IEEE transactions on Circuits and System for Video Technology*, entitled "Parametric interpolation filter for high-definition video coding" [46].

Chapter 5

2-D Order-16 Integer Transforms

In this chapter, 2-D order-16 transforms are proposed for HD video coding, based on one of the conclusions in Chapter 4 that the prediction errors of HD videos have higher correlation. After a review of the work related to 2-D order-16 transform and adaptive block-size transform (ABT) in Section 5.1, two new types of 2-D order-16 integer transforms, non-orthogonal Integer Cosine Transform (NICT) and modified ICT (MICT), are introduced in Section 5.2. Section 5.3 introduces the efforts we have made to reduce the complexity of the proposed transforms. Section 5.4 presents the solutions of the related problems, such as transform size selection and entropy coding, when the 2-D order-16 transforms are integrated into H.264/AVC High Profile (HP) [29] and AVS Enhanced Profile (EP) [40]. Section 5.5 reports the experimental results.

5.1 Related Work

In the existing video coding standards, the techniques focusing on improving coding efficiency are all originally designed as general coding tools for videos of various resolutions. Although efficient when applied to HD videos, they do not fully exploit HD videos' properties. Wien and Sun [47] once proposed an adaptive block-size transform (ABT) scheme to H.264/AVC, with seven transform block sizes of 4×4 up to 16×16. However, this scheme was not proposed for HD video coding and the reported experimental results were based on a set of CIF sequences. The maximum transform block size was finally reduced to 8×8 due to the visible artifacts and the overall complexity [48]. Recently, Naito and Koike [49] and Ma and Kuo [50] have investigated the efficiency of using super-MB for HD video coding within the framework of H.264/AVC. Naito and Koike [49] allow the MB size to be adaptively selected from 16×16 to $2^N \times 2^N$ ($N \geq 4$), where N may be configured at the picture level. The block sizes of intra prediction

and MCP are scaled, according to the MB size. The experimental results based on two 4K×2K sequences are reported and significant bit saving can be observed, when the sequences are coded using the fixed QP equal to 45. In [50], the MB size is fixed to be 32×32 and a 2-D order-16 integer transform is also proposed and adaptively used in addition to the 2-D order-4 and order-8 ICTs in H.264/AVC HP. Modest PSNR gain is shown based on two 1080p sequences, but subjective improvement is invisible.

For 2-D order-16 integer transforms, some work similar to MICT has been published [47; 50; 51; 52]. MICT is developed by modifying the dyadic symmetries of the DCT matrix and is naturally orthogonal, no matter what the values of the elements are. The one proposed by Wien and Sun [47] has one-norm transform matrix, which means the norms of all the basis vectors are the same and the memory for storing the scaling matrix can be saved. However, this 2-D order-16 integer transform has much lower coding gain [12] than DCT and introduces ringing artifacts, when integrated into H.264/AVC. The one proposed by Ma and Kuo [50] is simple and compatible with the 2-D order-8 ICT in H.264/AVC. However, the proposed transform modifies the structure of the DCT matrix so significantly that it is actually equivalent to the combined processes of 2-D order-8 ICT and 2-D order-2 Walsh-Hadamard Transforms (WHT). In other words, at first, the four 8×8 partitions of the input 16×16 block are transformed by 2-D order-8 ICT, respectively, and then the coefficients at the corresponding positions of the four transformed blocks are transformed by 2-D order-2 WHT and thus further de-correlated. Liang *et al.* [51] [52] proposed a family of approximations of DCT, named binDCT, and the lifting scheme for implementation. The lifting parameters, which are theoretically rational, are replaced by proper dyadic approximations and thus enable the multiplication-free fast algorithm. The trade-off between coding gain and computational complexity can be made by tuning the resolution of the approximation. However, the binDCT is not a unitary transform and thus uses the exact inversion, although invertibility is guaranteed by the lifting scheme. As a drawback, matrix multiplication cannot provide exactly the same results as the lifting scheme, unless additional shifts are used.

5.2 The Proposed 2-D Order-16 Integer Transforms

Two new types of 2-D order-16 integer transforms are proposed in this section. One type, called non-orthogonal ICT (NICT), allows the basis vectors to be non-orthogonal, in order to use matrix elements with small magnitudes and maintain the structure of the DCT matrix as well, e.g., dyadic symmetries and relative magnitudes. The transform error, which makes the average variance of the reconstruction error larger than that of the quantization error due to the non-orthogonality, is analyzed quantitatively. The specific NICT proposed in this work achieves the balance among the performance, the complexity, and the transform error. The latter modifies the structure of the DCT matrix by the principle of dyadic symmetry and is inherently orthogonal, no matter what the values of the matrix elements are. Both types allow selecting matrix elements more freely by releasing the orthogonality constraint and can provide comparable performance with that of DCT.

5.2.1 Review of ICT

ICT is generated from DCT by replacing the real-numbered elements of the DCT matrix with integers while maintaining the structure, such as relative magnitudes and signs, dyadic symmetries, and orthogonality, among the matrix elements [53]. ICT can be implemented using integer arithmetic without mismatch between the encoder and decoder and if well-designed, can provide almost the same compression efficiency as DCT. Furthermore, ICT has fast algorithms, which can be developed in the similar way for DCT.

The flow diagram of ICT from the input data to the reconstructed data is shown in Fig. 3.17 (a). As the transform matrix of ICT contains only integers and the norms of basis vectors are much larger than unity, the transformation process shown as (5.1) is not normalized. Therefore, a normalization process in (5.2), known as scaling, is required after transformation to ensure the conservation of energy and the reversibility of ICT.

$$\mathbf{F}_n = \mathbf{T}_n \times \mathbf{f}_n \times \mathbf{T}_n^T \tag{5.1}$$

$$\mathbf{S}_n = \langle \mathbf{F}_n / \mathbf{R}_n \rangle \tag{5.2}$$

In (5.1) and (5.2), $\mathbf{f}_n$ is the input $n \times n$ block, $\mathbf{T}_n$ and $\mathbf{R}_n$ are the $n \times n$ transform

matrix and scaling matrix, respectively, $\mathbf{S}_n$ is the output of ICT, and the notation $\langle \mathbf{X}/\mathbf{Y} \rangle$ means that each element in matrix $\mathbf{X}$ is divided by the element at the same position in matrix $\mathbf{Y}$. The elements in the scaling matrix $\mathbf{R}_n$ are derived from the norms of basis vectors of $\mathbf{T}_n$ as below, where column vector $\mathbf{m}$ comprises the norms of all basis vectors.

$$\mathbf{R}_n = \mathbf{m} \times \mathbf{m}^T \tag{5.3}$$

$$m_i = \sqrt{\sum_{j=0}^{n-1} \mathbf{T}_n^2(i,j)}, \quad 0 \leq i < n \tag{5.4}$$

The divisions in (5.2) are usually approximated by integer multiplication and shift as shown in (5.5),

$$\mathbf{S}_n = \langle \mathbf{F}_n \cdot \mathbf{P}_n \rangle >> N \tag{5.5}$$

where the notation $\langle \mathbf{X} \cdot \mathbf{Y} \rangle$ means inner product, i.e., each element in matrix $\mathbf{X}$ is multiplied by the element at the same position in matrix $\mathbf{Y}$ and $>> N$ means N-bit right shift.

Similarly, for the inverse ICT, the whole process including inverse scaling and inverse transformation can be represented by (5.6) as below.

$$\mathbf{f}_n = (\mathbf{T}_n^T \times \langle \mathbf{S}_n \cdot \mathbf{P}_n \rangle \times \mathbf{T}_n) >> N \tag{5.6}$$

The block diagram shown in Fig. 3.17 (a) is further simplified by the Pre-scaled Integer Transform (PIT) (see Fig. 3.17 (b)), which moves the inverse scaling from the decoder to the encoder and combines it with the forward scaling as one single process, named combined scaling. With PIT, the inverse scaling is saved and thus no scaling matrix is needed to be stored, whilst the complexity of encoders remains unchanged. However, the inverse scaling moved to the encoder side can be considered as a frequency weighting matrix to the transformed signals. In order not to alter the energy distribution of the transformed signals significantly, the elements' magnitudes in the scaling matrix should be almost the same. In other words, the norms of the basis vectors should be very close to each other according to (5.3) and (5.4), which is a key for the PIT design. Interested readers are referred to [54] for more details.

5.2.2 Order-16 Non-orthogonal ICT

The general transform matrix of an order-16 ICT is shown in (5.7) [53], which has alternating even and odd symmetries with respect to the vertical line. The basis vectors with even symmetry form the even part of the transform matrix, $\mathbf{T}_{8e}$, whereas those with odd symmetry form the odd part, $\mathbf{T}_{8o}$. Designing an order-16 ICT is equivalent to designing $\mathbf{T}_{8e}$ and $\mathbf{T}_{8o}$, and the orthogonality of each part is necessary and sufficient for the orthogonality of the order-16 ICT.

$$\mathbf{T}_{16} = \left[\begin{array}{cccccccc|cc} x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & \cdots \\ x_1 & x_3 & x_5 & x_7 & x_9 & x_{11} & x_{13} & x_{15} & -x_{15} & -x_{13} & \cdots \\ x_2 & x_6 & x_{10} & x_{14} & -x_{14} & -x_{10} & -x_6 & -x_2 & -x_2 & -x_6 & \cdots \\ x_3 & x_9 & x_{15} & -x_{11} & -x_5 & -x_1 & -x_7 & -x_{13} & x_{13} & x_7 & \cdots \\ x_4 & x_{12} & -x_{12} & -x_4 & -x_4 & -x_{12} & x_{12} & x_4 & x_4 & x_{12} & \cdots \\ x_5 & x_{15} & -x_7 & -x_3 & -x_{13} & x_9 & x_1 & x_{11} & -x_{11} & -x_1 & \cdots \\ x_6 & -x_{14} & -x_2 & -x_{10} & x_{10} & x_2 & x_{14} & -x_6 & -x_6 & x_{14} & \cdots \\ x_7 & -x_{11} & -x_3 & x_{15} & x_1 & x_{13} & -x_5 & -x_9 & x_9 & x_5 & \cdots \\ x_8 & -x_8 & -x_8 & x_8 & x_8 & -x_8 & -x_8 & x_8 & x_8 & -x_8 & \cdots \\ x_9 & -x_5 & -x_{13} & x_1 & -x_{15} & -x_3 & x_{11} & x_7 & -x_7 & -x_{11} & \cdots \\ x_{10} & -x_2 & x_{14} & x_6 & -x_6 & -x_{14} & x_2 & -x_{10} & -x_{10} & x_2 & \cdots \\ x_{11} & -x_1 & x_9 & x_{13} & -x_3 & x_7 & x_{15} & -x_5 & x_5 & -x_{15} & \cdots \\ x_{12} & -x_4 & x_4 & -x_{12} & -x_{12} & x_4 & -x_4 & x_{12} & x_{12} & -x_4 & \cdots \\ x_{13} & -x_7 & x_1 & -x_5 & x_{11} & x_{15} & -x_9 & x_3 & -x_3 & x_9 & \cdots \\ x_{14} & -x_{10} & x_6 & -x_2 & x_2 & -x_6 & x_{10} & -x_{14} & -x_{14} & x_{10} & \cdots \\ x_{15} & -x_{13} & x_{11} & -x_9 & x_7 & -x_5 & x_3 & -x_1 & x_1 & -x_3 & \cdots \end{array}\right] \tag{5.7}$$

$$\mathbf{T}_{8e} = \begin{bmatrix} x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 & x_0 \\ x_2 & x_6 & x_{10} & x_{14} & -x_{14} & -x_{10} & -x_6 & -x_2 \\ x_4 & x_{12} & -x_{12} & -x_4 & -x_4 & -x_{12} & x_{12} & x_4 \\ x_6 & -x_{14} & -x_2 & -x_{10} & x_{10} & x_2 & x_{14} & -x_6 \\ x_8 & -x_8 & -x_8 & x_8 & x_8 & -x_8 & -x_8 & x_8 \\ x_{10} & -x_2 & x_{14} & x_6 & -x_6 & -x_{14} & x_2 & -x_{10} \\ x_{12} & -x_4 & x_4 & -x_{12} & -x_{12} & x_4 & -x_4 & x_{12} \\ x_{14} & -x_{10} & x_6 & -x_2 & x_2 & -x_6 & x_{10} & -x_{14} \end{bmatrix} \tag{5.8}$$

$$\mathbf{T}_{8o} = \begin{bmatrix} x_1 & x_3 & x_5 & x_7 & x_9 & x_{11} & x_{13} & x_{15} \\ x_3 & x_9 & x_{15} & -x_{11} & -x_5 & -x_1 & -x_7 & -x_{13} \\ x_5 & x_{15} & -x_7 & -x_3 & -x_{13} & x_9 & x_1 & x_{11} \\ x_7 & -x_{11} & -x_3 & x_{15} & x_1 & x_{13} & -x_5 & -x_9 \\ x_9 & -x_5 & -x_{13} & x_1 & -x_{15} & -x_3 & x_{11} & x_7 \\ x_{11} & -x_1 & x_9 & x_{13} & -x_3 & x_7 & x_{15} & -x_5 \\ x_{13} & -x_7 & x_1 & -x_5 & x_{11} & x_{15} & -x_9 & x_3 \\ x_{15} & -x_{13} & x_{11} & -x_9 & x_7 & -x_5 & x_3 & -x_1 \end{bmatrix} \tag{5.9}$$

$\mathbf{T}_{8e}$ is an order-8 ICT. For compatibility, it is designed to be the scaled version of the transform matrix of the order-8 ICT in H.264/AVC or AVS, according to whichever

standard it is integrated into. The scaling factor is 4. In other words, the element set of the even part, $\{x_0, x_2, \ldots x_{12}, x_{14}\}$, is equal to $\{32, 40, 40, 36, 32, 24, 16, 8\}$ and $\{32, 48, 32, 40, 32, 24, 16, 12\}$, as proposed to AVS and H.264/AVC, respectively.

The possible element sets of an orthogonal $\mathbf{T}_{8o}$, i.e., x_1, x_3, $\ldots$ x_{13}, x_{15}, are the solutions of a set of three quadratic equations [55]. The solutions have relatively large magnitudes, i.e., represented by at least 6 bits, and those with small DCT distortion [47] have even larger magnitudes that increase the computational complexity significantly. In this work, NICT is proposed for the freedom of making trade-off between complexity and performance at the expense of orthogonality. Except orthogonality, NICT preserves all the advantages of ICT, such as bit-exact implementation and the fast algorithm.

Based on the property of orthogonal transforms, if there is no quantization in the frequency domain, signals can be reconstructed perfectly, otherwise, the average variance of the reconstruction error equals that of the quantization error [12]. For a non-orthogonal transform, the average variance of the reconstruction error is larger than that of the quantization error, and even though there is no quantization, the reconstruction is imperfect. Hence, we analyze the transform error introduced by the non-orthogonality.

First of all, we discuss the reconstruction error without quantization. Let $\mathbf{x}$ be a 1-D vector with length N. $\mathbf{x}$ is transformed by an order-N non-orthogonal transform $\mathbf{T}$, and the vector $\boldsymbol{\theta}$ is obtained in the transform domain, i.e., $\boldsymbol{\theta} = \mathbf{T}\mathbf{x}$. Let the reconstructed vector be $\mathbf{y}$, where $\mathbf{y} = \mathbf{T}^T\boldsymbol{\theta} = \mathbf{T}^T\mathbf{T}\mathbf{x}$. Obviously, $\mathbf{T}^T\mathbf{T}$ is not equal to the identity matrix $\mathbf{I}$, and the difference between $\mathbf{T}^T\mathbf{T}$ and $\mathbf{I}$ is denoted as $\mathbf{E}_r$, i.e., $\mathbf{E}_r = \mathbf{I} - \mathbf{T}^T\mathbf{T}$. The average variance of the reconstruction error σ_{r0}^2 can be expressed as below.

$$\begin{aligned}
\sigma_{r0}^2 &= \frac{1}{N}\sum_{k=0}^{N-1} E[(x_k - y_k)^2] = \frac{1}{N}E[(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})] \\
&= \frac{1}{N}E[(\mathbf{x}-\mathbf{T}^T\mathbf{T}\mathbf{x})^T(\mathbf{x}-\mathbf{T}^T\mathbf{T}\mathbf{x})] = \frac{1}{N}E[((\mathbf{I}-\mathbf{T}^T\mathbf{T})\mathbf{x})^T((\mathbf{I}-\mathbf{T}^T\mathbf{T})\mathbf{x})] \\
&= \frac{1}{N}E[(\mathbf{E}_r\mathbf{x})^T(\mathbf{E}_r\mathbf{x})] = \frac{1}{N}E[\mathbf{x}^T\mathbf{E}_r^T\mathbf{E}_r\mathbf{x}] \qquad (5.10)
\end{aligned}$$

Then, we denote $\mathbf{M} = \mathbf{E}_r^T\mathbf{E}_r$, and the deduction continues,

$$\begin{aligned}\sigma_{r0}^2 &= \frac{1}{N}E[\mathbf{x}^T\mathbf{M}\mathbf{x}] = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\mathbf{M}(k,j)E[x_k x_j] \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\mathbf{M}(k,j)R_{xx}[k,j] \le \frac{1}{N}\sigma_x^2\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}|\mathbf{M}(k,j)| \end{aligned} \tag{5.11}$$

where $R_{xx}[.]$, defined in (2.8), and σ_x^2 are the autocorrelation and the variance of the source, respectively. As the autocorrelation is less than or equal to the variance, the upper bound of σ_{r0}^2 for a given NICT is shown in (5.11). Hence, minimizing σ_{r0}^2 is equivalent to minimizing the term $\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}|\mathbf{M}(k,j)|$ in (5.11). Table 5.1 gives some examples of σ_{r0}^2 in the design of NICT with σ_x^2 equal to 1.0, and it will be explained later.

Then, we take the quantization into consideration and show the relationship of the average variances of the reconstruction error σ_r^2 and the quantization error σ_q^2. Let $\mathbf{u}$ be the quantized version of $\boldsymbol{\theta}$ and $\tilde{\mathbf{y}}$ be the reconstructed vector, i.e., $\tilde{\mathbf{y}} = \mathbf{T}^T\mathbf{u}$. The quantization error is denoted as $\mathbf{q}$, i.e., $\mathbf{q} = \boldsymbol{\theta} - \mathbf{u}$. The average variance of σ_r^2 can be expressed by (5.12).

$$\begin{aligned}\sigma_r^2 &= \frac{1}{N}\sum_{k=0}^{N-1}E[(x_k - \tilde{y}_k)^2] = \frac{1}{N}E[(\mathbf{x}-\tilde{\mathbf{y}})^T(\mathbf{x}-\tilde{\mathbf{y}})] \\ &= \frac{1}{N}E[(\mathbf{T}^{-1}\boldsymbol{\theta} - \mathbf{T}^T\mathbf{u})^T(\mathbf{T}^{-1}\boldsymbol{\theta} - \mathbf{T}^T\mathbf{u})] \end{aligned} \tag{5.12}$$

Since $\mathbf{T}^{-1}$ is not equal to $\mathbf{T}^T$ because of the non-orthogonality of $\mathbf{T}$, we use $(\mathbf{T}^T - \mathbf{T}_{er})$ to represent $\mathbf{T}^{-1}$, and substitute $\mathbf{T}^{-1}$ in (5.12) with $(\mathbf{T}^T - \mathbf{T}_{er})$. Then we get (5.13).

$$\begin{aligned}\sigma_r^2 &= \frac{1}{N}E[((\mathbf{T}^T - \mathbf{T}_{er})\boldsymbol{\theta} - \mathbf{T}^T\mathbf{u})^T((\mathbf{T}^T - \mathbf{T}_{er})\boldsymbol{\theta} - \mathbf{T}^T\mathbf{u})] \\ &= \frac{1}{N}E[(\mathbf{T}^T(\boldsymbol{\theta}-\mathbf{u}) - \mathbf{T}_{er}\boldsymbol{\theta}))^T(\mathbf{T}^T(\boldsymbol{\theta}-\mathbf{u}) - \mathbf{T}_{er}\boldsymbol{\theta}))] \\ &= \frac{1}{N}E[(\mathbf{T}^T\mathbf{q} - \mathbf{T}_{er}\boldsymbol{\theta})^T(\mathbf{T}^T\mathbf{q} - \mathbf{T}_{er}\boldsymbol{\theta})] \\ &= \frac{1}{N}E[\mathbf{q}^T\mathbf{T}\mathbf{T}^T\mathbf{q} - 2\mathbf{q}^T\mathbf{T}\mathbf{T}_{er}\boldsymbol{\theta} + (\mathbf{T}_{er}\boldsymbol{\theta})^T\mathbf{T}_{er}\boldsymbol{\theta}] \\ &= \frac{1}{N}E[\mathbf{q}^T(\mathbf{I} - \mathbf{E}_r)\mathbf{q} - 2\mathbf{q}^T\mathbf{T}(\mathbf{T}^T - \mathbf{T}^{-1})\boldsymbol{\theta} + (\mathbf{T}^T\boldsymbol{\theta} - \mathbf{T}^{-1}\boldsymbol{\theta})^T(\mathbf{T}^T\boldsymbol{\theta} - \mathbf{T}^{-1}\boldsymbol{\theta})] \\ &= \frac{1}{N}E[\mathbf{q}^T\mathbf{q} - \mathbf{q}^T\mathbf{E}_r\mathbf{q} + 2\mathbf{q}^T\mathbf{E}_r\boldsymbol{\theta} + (\mathbf{y}-\mathbf{x})^T(\mathbf{y}-\mathbf{x})] \\ &= \sigma_q^2 + \sigma_{r0}^2 - \frac{1}{N}E[\mathbf{q}^T\mathbf{E}_r\mathbf{q}] + \frac{2}{N}E[\mathbf{q}^T\mathbf{E}_r\boldsymbol{\theta}] \end{aligned} \tag{5.13}$$

x_1	x_3	x_5	x_7	x_9	x_{11}	x_{13}	x_{15}	DCT Distortion (%)	Upper bound of σ_{r0}^2 (10^{-6})
42	38	37	32	22	19	10	4	12.90	0
62	61	49	47	37	31	21	5	10.38	0
94	93	73	70	58	51	26	6	15.71	0
120	114	103	94	68	57	34	14	8.77	0
120	108	104	85	69	52	32	2	11.64	0
28	27	23	21	17	14	8	2	9.04	3.69
29	28	26	22	20	13	10	2	6.92	1.00
38	36	35	29	25	18	9	3	8.53	5.21
39	37	35	29	26	18	11	2	7.36	2.22
40	39	33	31	24	19	14	4	8.70	4.11
40	38	35	31	24	19	11	4	3.74	1.51

Table 5.1: *Comparison of sets of matrix elements*

The third and forth terms in (5.13) are all equal to zero, because the autocorrelation of $\mathbf{q}$ and the cross-correlation of $\mathbf{q}$ and $\boldsymbol{\theta}$ are all zero as shown by Widrow *et al.* [56]. Therefore, the relationship between σ_r^2 and σ_q^2 is shown in (5.14).

$$\sigma_r^2 = \sigma_q^2 + \sigma_{r0}^2 \tag{5.14}$$

Clearly, the average variance of the reconstruction error σ_r^2 is always σ_{r0}^2 larger than the quantization error variance σ_q^2, no matter the transform coefficients are quantized or not, whereas with an orthogonal transform, σ_r^2 is equal to σ_q^2.

The key of designing an NICT is the balance among σ_{r0}^2, the approximation of DCT, and the magnitudes of the matrix elements. Table 5.1 gives 11 element sets of $\mathbf{T}_{8o}$ in (5.9). The DCT distortions [47] and upper bounds of σ_{r0}^2 in (5.11) are of $\mathbf{T}_{8o}$ in (5.9) instead of $\mathbf{T}_{16}$ in (5.7). The upper five sets are given in [55], which lead to orthogonal $\mathbf{T}_{8o}$, but the magnitudes and the DCT distortions of these sets are larger than the rest six sets leading to non-orthogonal $\mathbf{T}_{8o}$.

Although non-orthogonal, the bottom six element sets are all with negligible σ_{r0}^2. Particularly, the one in the bottom row has the minimum DCT distortion and relatively small σ_{r0}^2 and thus is used for the element set of the odd part. The performance will be verified in Section 5.5.

5.2.3 Order-16 Modified ICT

The MICT transform matrix was first proposed by Cham [57]. It is obtained by modifying the structure of the order-16 DCT matrix using the principle of dyadic symmetry. The basis vectors of the order-16 MICT are inherently orthogonal no matter what the elements are, so compared with the elements in NICT transform matrix, even smaller magnitudes can be selected. The even part of the general transform matrix remains the same as the $\mathbf{T}_{8e}$ in (5.8), while the odd part is changed, shown as (5.15).

$$\mathbf{M}_{8o} = \begin{bmatrix} x_1 & x_3 & x_5 & x_7 & x_9 & x_{11} & x_{13} & x_{15} \\ x_9 & x_{11} & x_{13} & x_{15} & -x_1 & -x_3 & -x_5 & -x_7 \\ x_5 & x_7 & -x_1 & -x_3 & -x_{13} & -x_{15} & x_9 & x_{11} \\ x_{15} & x_{13} & -x_{11} & -x_9 & x_7 & x_5 & -x_3 & -x_1 \\ x_{13} & -x_{15} & -x_9 & x_{11} & x_5 & -x_7 & -x_1 & x_3 \\ x_3 & -x_1 & -x_7 & x_5 & -x_{11} & x_9 & x_{15} & -x_{13} \\ x_7 & -x_5 & x_3 & -x_1 & -x_{15} & x_{13} & -x_{11} & x_9 \\ x_{11} & -x_9 & x_{15} & -x_{13} & x_3 & -x_1 & x_7 & -x_5 \end{bmatrix} \tag{5.15}$$

With the even part $\mathbf{T}_{8e}$ in (5.8) and the odd part $\mathbf{M}_{8o}$ in (5.15), the first 3 basis vectors of the order-16 MICT transform matrix resemble those of DCT and represent low frequency components. So the MICT shows good energy compaction capability especially for the regions in HD videos, where pixels are higher correlated. Though the dyadic symmetries of most basis vectors in the odd part of the transform matrix are different from those in the DCT matrix, the MICT will not bring significant performance penalty, if the elements are well-selected, as evident in Section 5.5.

Since the elements in the transform matrices are selected without orthogonality constraint, it is free to make a trade-off between the performance and the magnitudes of the elements, i.e., the computational complexity. In this work, the magnitudes of elements are represented by only 3-4 bits and 16-bit integer implementation can be developed easily. In detail, the even part is selected to be the same as the order-8 ICT in AVS or H.264/AVC, according to whichever it is integrated into. For the odd part, there are three considerations for elements' values. Firstly, the magnitudes should be comparable to those of the even part. Secondly, the waveform of the second basis of the 16×16 MICT transform matrix should resemble that of DCT. Thirdly, it should be suitable for the design of fast algorithm, which will be illustrated in Section 5.3.2. Taking all these constraints into consideration, Fong proposed the value of $\{x_1, x_3, \ldots x_{13}, x_{15}\}$

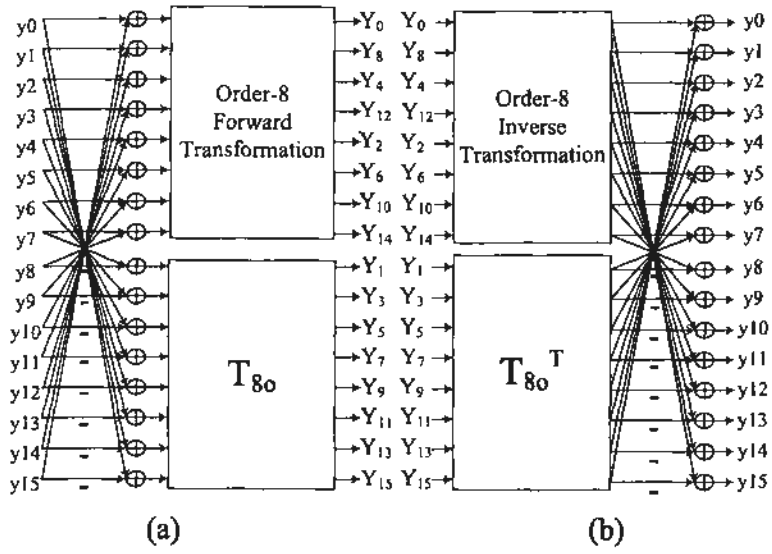


Figure 5.1: *The compatible structures for order-8 and order-16 (a) forward transformation and (b) inverse transformation*

to be {11, 11, 11, 9, 8, 6, 4, 1} [57].

5.3 Complexity Reduction

This section introduces the research efforts we have made for complexity reduction. Section 5.3.1 introduces that the 2-D order-16 integer transforms are compatible with the 2-D order-8 ICT in AVS EP or H.264/AVC HP, according to whichever standard they are integrated into. In Section 5.3.2, the fast algorithm for the 2-D order-16 MICT proposed in Section 5.2.3 is developed and extended to a general approach.

5.3.1 Compatibility with the Order-8 ICT

It is natural that the order-16 DCT is compatible with the order-8 one. However, it is not always true for integer transforms. Bossen [58] points out that the compatibility of order-8 and order-4 ICTs can be achieved by taking the order-4 ICT as the even part of the order-8 one. In this work, the order-16 integer transforms use the the order-8 ICTs in the standards or their scaled version as the even part for the compatibility.

Firstly, the compatible transformation is presented. Fig. 5.1 (a) shows the flow diagram of the forward transformation, applicable for both types of order-16 integer transforms. The bottom-right and upper-right blocks complete the function of matrix

multiplication with the odd and even parts of $\mathbf{T}_{16}$, respectively. In case of the order-16 MICT, where $\mathbf{T}_{8e}$ is exactly the same as the order-8 ICT, the upper-right block may be reused directly for the order-16 forward transformation. In the other case of the order-16 NICT, where $\mathbf{T}_{8e}$ is equal to the order-8 ICT scaled by a factor 4, the output of upper-right block should be left shifted 2 bits before they are reused for the forward transformation of the order-16 NICT. Therefore, a module specially for the order-8 forward transformation is saved, and the output data from the order-8 forward transformation can be reused by the order-16 one. For the inverse transformation, the compatibility can be achieved in the similar fashion, as shown in Fig. 5.1 (b).

Based on the compatibility of transform matrix, the inherent relationship between the scaling matrices, $\mathbf{R}_8$ and $\mathbf{R}_{16}$, is derived as (5.16), according to (5.3) and (5.4),

$$\mathbf{R}_8(i,j) \times 2^M = \mathbf{R}_{16}(2i,2j),\ 0 \le i,j < 8 \tag{5.16}$$

where M equals 5 and 1 for the case of the NICT and the MICT, respectively. When the divisions in (5.2) are approximated by integer multiplication and right shift as shown in (5.5), the relationship between the scaling matrix $\mathbf{P}_8$ and $\mathbf{P}_{16}$ for the scaling is

$$\mathbf{P}_8(i,j) = \mathbf{P}_{16}(2i,2j) \times 2^K,\ 0 \le i,j < 8, \tag{5.17}$$

where K equals 5 and 1 for the case of the NICT and the MICT, respectively. Equation (5.17) indicates that $\mathbf{P}_8(i,j)$ is 2^K times larger than $\mathbf{P}_{16}(2i,2j)$ if N, the number of bits for right shift in (5.5), is the same for the order-16 and the order-8 transforms. The difference between $\mathbf{P}_8(i,j)$ and $\mathbf{P}_{16}(2i,2j)$ can be easily sorted out by shift operations. In other words, $\mathbf{P}_{16}(2i,2j)$, $0 \le i,j < 8$, may be used for the scaling of the order-8 ICT but N in (5.5) for 8×8 scaling is K bits less than that for the 16×16 scaling. Hence, the memory for storing an 8×8 scaling matrix is saved, thus achieving the compatibility of scaling matrix. When the order-16 integer transforms are implemented as PIT to be compatible with the order-8 PIT in AVS, (5.17) is also true for the combined scaling with K doubled. Interested readers are referred to [59] for more details.

5.3.2 Fast Algorithm for the Order-16 MICT

ICT and NICT have fast algorithms, which can be developed in the similar way for DCT. However, fast algorithms for the MICTs are not guaranteed, and little research effort has been devoted to it. We propose an approach to developing fast algorithms for the integer transforms with different structures from the DCT matrix, including the MICT. Due to the separability of a 2-D transform and the similarity of the forward and inverse transformation flows, we focus on the fast algorithm for 1-D order-16 transformation in this section.

The eight butterflies in the left part of Fig. 5.1 (a) can be easily developed as the first stage, which exploits the symmetries with respect to the line in (5.7). The upper-right block completes the function of matrix multiplication with the even part, which is the same as the order-8 ICTs in the standards in this work. Fortunately, both order-8 ICTs in AVS and H.264/AVC have efficient fast algorithms, which are borrowed here. For more general case that the even part of the MICT has different structure with $\mathbf{T}_{8e}$ in (5.8), the fast algorithms can be developed in the same way for $\mathbf{M}_{8o}$ as introduced below.

The bottom-right block in Fig. 5.1 (a), completing the function of matrix multiplication with the odd part, has quite different structure from the odd part of the order-16 DCT matrix. So we cannot decompose it to butterfly operations just as we do for $\mathbf{T}_{8e}$. Instead, we represent $\mathbf{M}_{8o}$ as the product of three 8×8 matrices, as shown in (5.18).

$$\mathbf{M}_{8o} = \mathbf{M}_1 \times \mathbf{M}_2 \times \mathbf{M}_3 \tag{5.18}$$

There are some considerations for the three matrices. Firstly, they should contain integers only. Secondly, the integers should be very small such that only additions and shifts are involved and multiplications are avoided in fast algorithms. Last but not least, the matrices should be sparse, which means that many zeros appear in the matrices.

Obviously, row vectors in $\mathbf{M}_{8o}$ are orthogonal and have the same L^2 norm, where the L^2 norm of a row vector $\mathbf{a}$ is defined as $\mathbf{a} \times \mathbf{a}^T$. We denote the L^2 norm of every row vector of $\mathbf{M}_{8o}$ as n_O, and clearly $|\det(\mathbf{M}_{8o})|$ is equal to n_O^4, where $\det(.)$ and $|.|$ mean determinant and absolute value, respectively. To simplify the problem, we assume $\mathbf{M}_1$,

$\mathbf{M}_2$, and $\mathbf{M}_3$ in (5.18) are also row-orthogonal and the row vectors have the same L^2 norm in each matrix, denoted as n_1, n_2, and n_3, respectively. Similarly, $|\det(\mathbf{M}_1)|$, $|\det(\mathbf{M}_2)|$, and $|\det(\mathbf{M}_3)|$ are equal to n_1^4, n_2^4, and n_3^4, respectively. If (5.18) is satisfied, (5.19) is necessarily true.

$$\begin{aligned} |\det(\mathbf{M}_{8o})| = n_O^4 &= |\det(\mathbf{M}_1)| \times |\det(\mathbf{M}_2)| \times |\det(\mathbf{M}_3)| \\ &= n_1^4 \times n_2^4 \times n_3^4 \end{aligned} \tag{5.19}$$

Therefore,

$$n_O = n_1 \times n_2 \times n_3. \tag{5.20}$$

So when designing an order-16 MICT with a fast algorithm, we should make sure that n_O of $\mathbf{M}_{8o}$ is the product of three integers. Otherwise, (5.18) has no solution under this assumption. With the constraint of (5.20), the L^2 norms of row vectors in $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ are much smaller than that in $\mathbf{M}_{8o}$, which means the elements in the three matrices have very small magnitudes and may also contain many zeros.

Among the three matrices, $\mathbf{M}_3$ will be found first. Both sides of (5.18) are post-multiplied by the inverse of $\mathbf{M}_3$, which is equal to $\mathbf{M}_3^T/n_3$, so (5.18) changes to (5.21) as below.

$$\mathbf{M}_{8o} \times \mathbf{M}_3^T/n_3 = \mathbf{M}_1 \times \mathbf{M}_2 \tag{5.21}$$

It is noticed that $\mathbf{M}_3^T$ may be regarded as a set of eight column vectors and $\mathbf{M}_{8o} \times \mathbf{M}_3^T/n_3$ contains only integers. So we first collect a set of column vectors by exhaustive search, in which each column vector, b_i, satisfies two conditions. One is that the L^2 norm of b_i is n_3, and the other is the elements in $(\mathbf{M}_{8o} b_i/n_3)$ are all integers. If (5.18) has solutions, we can pick out eight orthogonal column vectors from the set to form $\mathbf{M}_3^T$ and thus get $\mathbf{M}_3$.To find $\mathbf{M}_2$, both sides of (5.21) are postmultiplied by the inverse of $\mathbf{M}_2$, i.e., $\mathbf{M}_2^T/n_2$, so (5.21) becomes (5.22).

$$(\mathbf{M}_{8o} \times \mathbf{M}_3^T/n_3) \times \mathbf{M}_2^T/n_2 = \mathbf{M}_1 \tag{5.22}$$

$\mathbf{M}_2$ can be obtained by the similar method of searching $\mathbf{M}_3$. Finally, with $\mathbf{M}_2$ and $\mathbf{M}_3$ known, $\mathbf{M}_1$ is computed by (5.22).

For the case of $\mathbf{M}_{8o}$ in this work, its determinant is 9.9049×10^{10} which is equaled

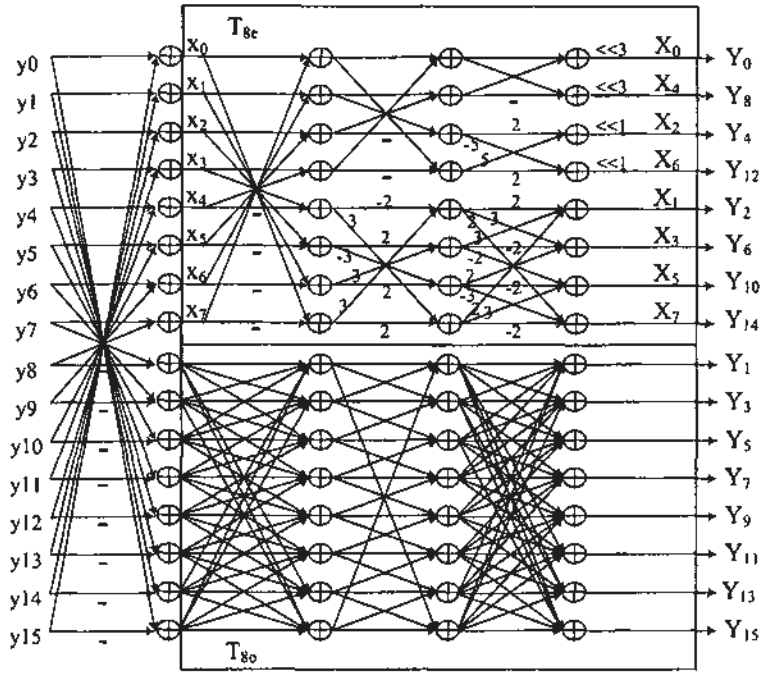


Figure 5.2: *The flow diagram of the order-16 forward transformation*

to 561[4] and the corresponding n_O is 561. To show that n_O is a product of three integers as in (5.20), we let n_1, n_2, and n_3 be 3, 11 and 17 with the order changeable. Then the method introduced above is used to search for $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$, and (5.23) shows one of the solutions of (5.18). The magnitudes of elements in (5.23) are so small that the fast algorithm can be implemented by using only additions and shifts. Fig. 5.2 shows the flow diagram of the forward transformation of the order-16 MICT. The shifts are not indicated in the bottom-right block for a clearer view.

$$\mathbf{M}_{8o} = \mathbf{M}_1 \times \mathbf{M}_2 \times \mathbf{M}_3 \tag{5.23}$$

$$= \begin{bmatrix} -2 & 0 & 1 & -1 & -1 & 3 & -1 & 0 \\ 3 & -1 & 1 & 1 & 0 & 2 & 0 & 1 \\ -1 & -3 & 1 & 0 & 1 & 0 & 2 & -1 \\ 0 & 1 & 0 & 1 & 3 & 1 & -1 & -2 \\ 1 & -1 & -3 & -2 & 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 0 & -2 & 0 & 1 & -3 \\ 0 & -2 & 0 & 1 & -1 & -1 & -3 & -1 \\ -1 & 0 & -2 & 3 & -1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -2 & 0 & -1 & 0 & 2 & -1 & 1 \\ 0 & -1 & 0 & 2 & 1 & -1 & 0 & 2 \\ 0 & -2 & 1 & 1 & 0 & 0 & 1 & -2 \\ -2 & 0 & -1 & 0 & 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & 0 & -1 & -1 & -2 & 0 \\ -2 & 0 & 1 & -1 & 0 & 0 & 2 & 1 \\ -1 & 1 & 0 & 2 & -1 & 2 & 0 & 0 \\ -1 & -1 & -2 & 0 & -2 & -1 & 0 & 0 \end{bmatrix}$$

5.4 Integration of the 2D Order-16 Integer Transforms into the Standards

In this work, the proposed two types of 2D order-16 integer transforms are integrated into AVS EP [40] and H.264/AVC HP [29], respectively. For AVS, there is an order-8 ICT already existing, so the 2D order-16 integer transform is used adaptively as an alternative to the 2D order-8 one according to local activities. For H.264/AVC, an ABT scheme is used in the High Profile, including the 2D order-8 and order-4 ICTs, but the 2D order-4 ICT does not contribute much to the efficiency of HD video coding, as verified in Section 5.5. Hence, the proposed ABT scheme for H.264/AVC uses only 2D order-16 and order-8 transforms with 2D order-4 ICT removed.

Transform Size Selection

The luminance component in an MB can be transformed by one 2-D order-16 transform, or, alternatively, by four 2D order-8 transforms. The selection of the optimum transform size is performed using the criterion of minimum R-D cost as introduced in (3.14) with a proper Lagrange multiplier λ [60]. The two block-size transforms are tried one by one and their R-D costs J are calculated, respectively. The one with the lower cost is selected. With this selection criterion, the best R-D performance can be achieved, but a 1-bit binary signal should be transmitted in every MB header to indicate which transform is used.

16×16 Intra Prediction

For AVS EP, the largest block size for intra prediction is 8×8, and to apply the 2D order-16 integer transforms to intra-coded MB, the 16×16 intra prediction must be in place. Five prediction directions are used, including DC, horizontal, vertical, bottom-right and bottom-left, all of which are extended from the modes for 8×8 blocks. However, intra prediction with large block size is more likely to introduce visible artifacts, since more pixels are predicted from the same source [61] [62]. Therefore, instead of using the reference pixels directly, a 3-tap lowpass filter [1 2 1]/4 is applied to the reference pixels before they are used, as shown in Fig. 5.3. As for the DC mode, the predicted value is the average of all the available top and left neighboring pixels.

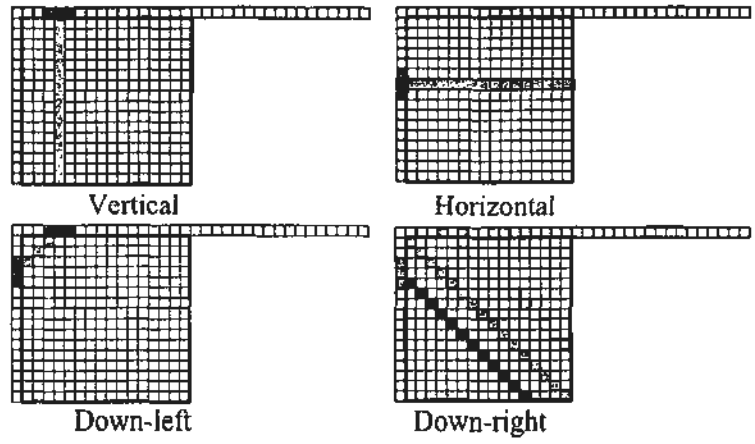


Figure 5.3: *16×16 intra prediction modes*

16×16 Block Entropy Coding

In AVS EP and H.264/AVC HP, the 8×8 residual blocks are coded by context-based arithmetic coding [29] [40]. Their arithmetic coding engines can code sources with various statistics by using different probability models and can also keep track of the varying statistics experienced in the coding process by updating the p.d.f. of the probability models. Therefore, instead of designing new codebooks, two additional sets of probability models specially for the 16×16 residual blocks are designed for AVS and H.264/AVC, respectively, to drive the arithmetic coding engines which remain unchanged in this work. All the probability models can be updated adaptively according to the changes of the source statistics.

Coded Block Pattern (CBP)

CBP, a 6-bit integer number ranging from 0 to 63, is an SE included in both AVS and H.264/AVC. The two most significant bits (MSB) indicate whether the residual blocks of two chrominance components, Cb and Cr, contain non-zero coefficients or not, respectively, while the four least significant bits (LSB) are used for the four 8×8 residual blocks of luminance component for the same purpose. After the 2D order-16 transforms are integrated into the standards, CBP becomes a 3-bit integer number with the LSB indicating whether the 16×16 luminance residual block contains non-zero coefficients, if an MB is coded by the 2D order-16 transform.

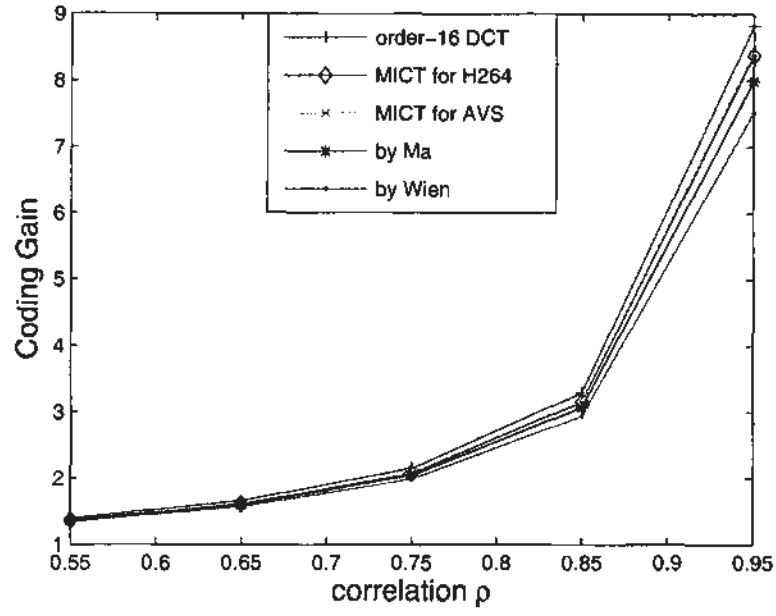


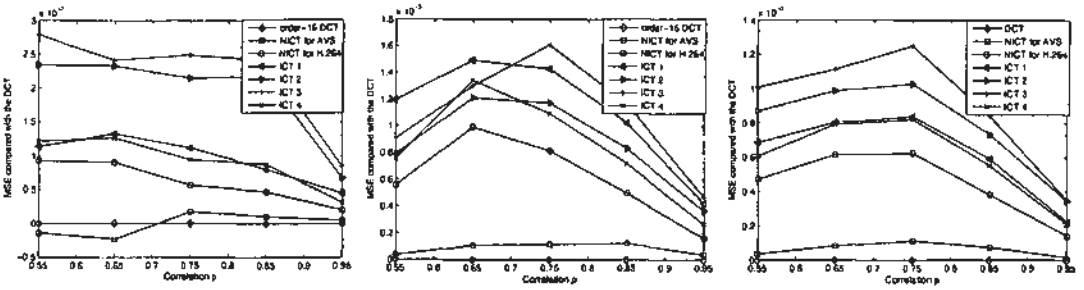
Figure 5.4: *Comparison of coding gains of orthogonal transforms*

5.5 Experimental Results

5.5.1 Transform Coding Gain

As introduced in Section 2.4.2, transform coding gain G_{TC} [12] is an important measure to evaluate the performance of a transform coder. G_{TC} weights the reconstruction error variance of PCM coding D_{PCM} and the corresponding reconstruction error variance of quantized transform coding D_{TC}, as defined in (2.80). Under the assumptions of optimum quantization and bit allocation at the same overall rate, G_{TC} is the ratio of arithmetic to geometric means of the variances of transform coefficients, as shown in (2.81).

Suppose the source is 1-D zero-mean, unit-variance first-order Markov processes with adjacent element correlation ρ ranging from 0.55 to 0.95, which is typical for the prediction errors of HD videos (see Table 4.2). Figure 5.4 shows the coding gains of the order-16 DCT, the proposed MICTs for AVS and H.264/AVC, respectively, and the two MICTs proposed by Wien [47] and Ma [50], respectively. DCT has the highest coding gain, no matter what the value of ρ is. The coding gains of the proposed two MICTs - too close to be distinguished from each other - are higher than those of the MICTs proposed by Ma and Wien, especially when ρ approaches 1.0.



(a) (b) (c)

Figure 5.5: *The reconstruction error with (a) 4:1, (b) 4:2, and (c) 4:3 zonal filtering*

5.5.2 Reconstruction Error of Non-orthogonal Transforms

In the case of non-orthogonal transforms, even with optimal quantization and bit allocation, (2.81) cannot be derived to substitute (2.80), because D_{PCM} depends on quantization only, whereas D_{TC} depends not only on quantization but also on the variance of the source. Therefore, we cannot evaluate the performance of the proposed NICTs by (2.81); instead, we have to implement the transforms and quantization and compare the reconstruction errors thus introduced.

Seven order-16 transforms are tested, including the order-16 DCT, the proposed two NICTs and the four ICTs, of which the even part is the order-8 ICT in H.264/AVC and the element sets of the odd part are shown in the first four rows of Table 5.1. These transforms are applied to the 1-D zero-mean, unit-variance first-order Markov processes with adjacent element correlation ρ ranging from 0.55 to 0.95. Before inverse transform, the transform coefficients are zonal filtered [12], which means the bit-rate is reduced by simply retaining the first n ($n < 16$) transform coefficients. The MSE between the source and the reconstructed value are calculated.

Fig. 5.5 compares the MSE introduced by 4:1, 4:2, and 4:3 zonal filters. In detail, the three zonal filters retain the first 4, 8, and 12 transform coefficients, respectively, and set all the others to zero. For a clear view, the curves in Fig. 5.5 present the difference MSE compared with that of the order-16 DCT. We notice that the four ICTs, although orthogonal, produce larger MSE than the proposed two NICT. Therefore, the performance of an integer transform is greatly determined by the approximation of the exact DCT rather than the orthogonality, if σ_{r0}^2 is negligible compared with σ_q^2 in (5.14).

Platform	RM62b (AVS), JM11 (H.264/AVC)
Sequence structure	IBBPBBP...
Intra frame period	0.5 second
Entropy coding	Arithmetic coding
Fast motion estimation	on
Deblocking filter	on
R-D optimization	on
QP	fixed (27, 30, 35, 40, AVS)
	fixed (20, 24, 28, 32, H.264/AVC)
Rate control	off
Reference frame	2 (AVS), 5 (H.264/AVC)
Search range	±32
Frame number	60

Table 5.2: *Test conditions*

	Proposed NICT		Proposed MICT	
HD Sequence	PSNR gain (dB)	Bit saving (%)	PSNR gain (dB)	Bit saving (%)
Bigship	0.167	-5.98	0.123	-4.65
City	0.170	-5.69	0.123	-4.39
Crew	0.214	-9.31	0.219	-9.81
Optis	0.209	-7.65	0.136	-5.21
Raven	0.132	-4.14	0.087	-2.83
Sheriff	0.162	-5.66	0.086	-3.19
Pedestrian	0.227	-6.90	0.223	-7.12
Riverbed	0.471	-8.86	0.235	-4.77
RushHour	0.198	-7.07	0.190	-7.45
Station	0.195	-8.74	0.116	-4.02
Sunflower	0.158	-4.41	0.151	-4.41
Tractor	0.192	-5.49	0.087	-2.57
Average	**0.21**	**-6.66**	**0.15**	**-5.04**

Table 5.3: *Performances of NICT and MICT on the AVS platform*

5.5.3 Objective Evaluation

The proposed two types of 2-D order-16 integer transforms are implemented using 16-bit arithmetic and integrated into the RM62b platform for AVS EP and the JM11 platform for H.264/AVC HP, respectively. The integration problems of transform size selection, intra prediction, entropy coding, and CBP, are solved in Section 5.4. The test conditions are listed in Table 5.2. Tables 5.3, 5.4, and Table 5.5 show performance gains, compared to AVS EP using a 2-D order-8 ICT and H.264/AVC HP using both 2-D order-4 and order-8 ICTs, respectively. The improvement is measured by PSNR

HD Sequence	order-4, order-8, order-16		order-8, order-16	
	PSNR gain (dB)	Bit saving (%)	PSNR gain (dB)	Bit saving (%)
BigShip	0.14	-5.09	0.10	-3.76
City	0.17	-5.29	0.13	-4.18
Crew	0.18	-7.71	0.19	-8.26
Optis	0.28	-9.47	0.28	-9.46
Raven	0.33	-10.61	0.30	-9.87
Sheriff	0.16	-5.08	0.14	-4.56
Pedestrian	0.18	-6.77	0.20	-7.59
Riverbed	0.40	-8.55	0.48	-10.20
RushHour	0.15	-7.41	0.14	-6.95
Station	0.23	-9.48	0.23	-10.09
Sunflower	0.45	-15.16	0.40	-13.92
Tractor	0.30	-8.35	0.27	-7.75
Average	**0.25**	**-8.25**	**0.24**	**-8.05**

Table 5.4: *Performance of NICT on the H.264/AVC platform*

HD Sequence	order-4, order-8, order-16		order-8, order-16	
	PSNR gain (dB)	Bit saving (%)	PSNR gain (dB)	Bit saving (%)
BigShip	0.12	-4.51	0.08	-3.13
City	0.13	-4.27	0.10	-3.21
Crew	0.17	-7.49	0.18	-7.98
Optis	0.24	-8.33	0.23	-7.96
Raven	0.27	-8.70	0.23	-7.48
Sheriff	0.12	-3.81	0.09	-3.02
Pedestrian	0.14	-5.59	0.15	-5.92
Riverbed	0.10	-2.28	0.14	-3.16
RushHour	0.11	-5.90	0.09	-4.82
Station	0.17	-7.04	0.17	-7.55
Sunflower	0.44	-14.96	0.39	-13.50
Tractor	0.22	-6.28	0.16	-4.80
Average	**0.19**	**-6.60**	**0.17**	**-6.05**

Table 5.5: *Performance of MICT on the H.264/AVC platform*

gain at the same bit-rate or by bit-rate saving at the same PSNR, using the method in [63].

With the NICT, the improvements are all greater than 0.1 dB and more than 0.2 dB on average, whilst for the best case of *Riverbed*, the gain is up to 0.47 dB on the AVS platform and 0.48 dB on the H.264/AVC platform. The sequence *Riverbed* has smooth textures and global motions, so the energy of prediction errors is quite small, no

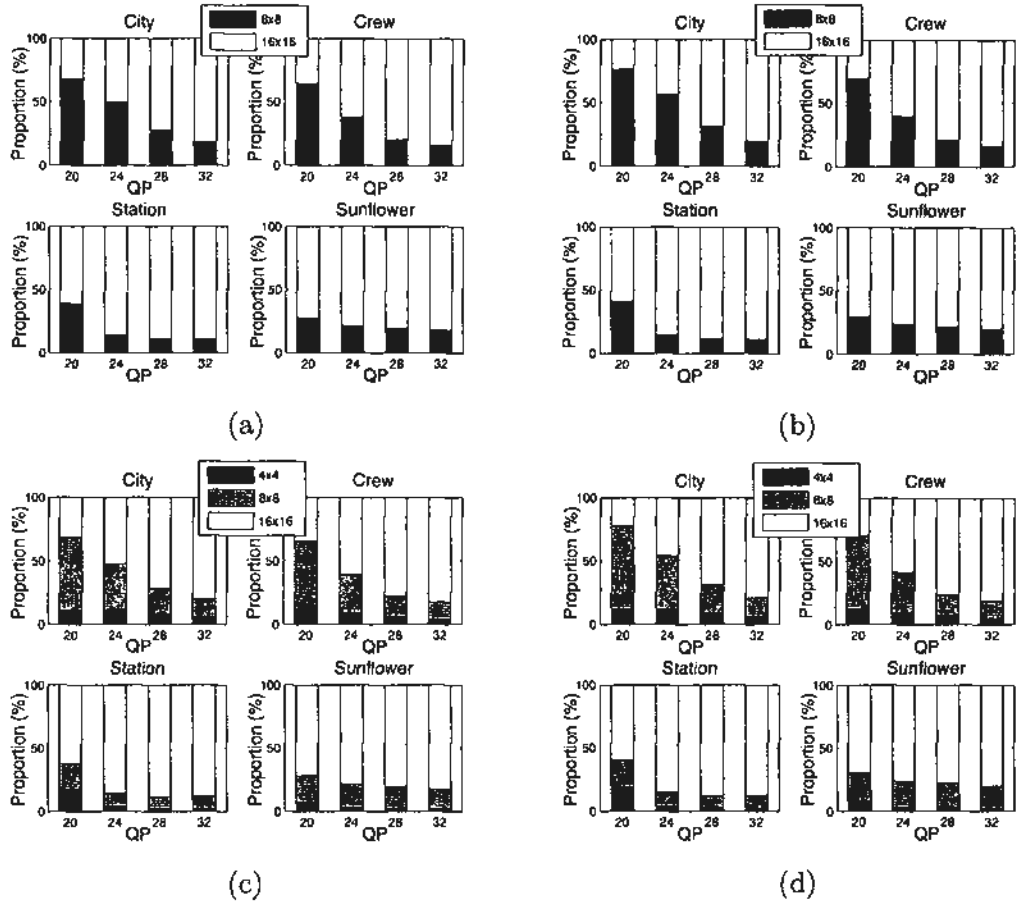


Figure 5.6: *The proportion of different block size transforms used in H.264/AVC HP. (a) 2-D order-8 ICT and order-16 NICT, (b) 2-D order-8 ICT and order-16 MICT, (c) 2-D order-8 and order-4 ICTs and 2-D order-16 NICT, (d) 2-D order-8 and order-4 ICTs and 2-D order-16 MICT*

matter using inter or intra prediction. In the transform domain, the prediction errors can be represented by only a few coefficients of the 2-D order-16 NICT, most of which are distributed in the low frequency area.

With the MICT, the coding efficiency is improved on an average of around 0.06 dB less than that by NICT on both platforms, but for the sequences with large homogeneous regions or smooth motions, such as *Crew*, *Pedestrian*, *RushHour*, and *Sunflower*, the performance of the MICT is comparable to or even better than the NICT, due to the efficiency of the first three basis vectors. However, for the sequences full of details, such as *Sheriff*, *Station*, and *Tractor*, the gain of the MICT becomes trivial because of the relative inefficiency to compress high frequency components. Overall, the performance gap between the proposed NICT and MICT is small.

Fig. 5.6 (a) and (b) show the percentage of MBs coded by the 2-D order-16 integer transforms. On average, more than half of the MBs are coded by 2-D order-16 transforms, and for some sequences, such as *Sunflower* and *Station*, the percentage of 2-D order-16 transforms is up to 80%. This implies that 2-D order-16 integer transforms are very useful in HD video coding. There is a tendency that the higher the bit-rate, the less the 2-D order-16 integer transforms are used. That is because at high bit-rates, many high frequency coefficients survive the quantization and high cost coefficients, i.e., a (*level, run*) pair with large *run*, have to be coded due to the large block size of 16×16. Hence, 2-D order-8 ICT is more likely to be selected.

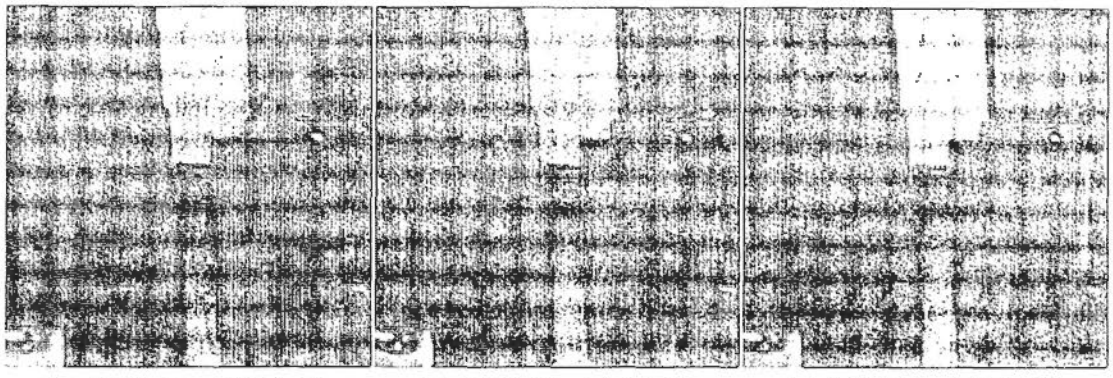
We also study the ABT scheme using three transforms, 2-D order-4, order-8, and order-16 transforms, on the H.264/AVC platform. As shown in Tables 5.4 and 5.5, the performance gap of the two ABT schemes with and without the 2-D order-4 ICT is marginal. The same conclusion can also be drawn by Fig. 5.6 (c) and (d), where the percentages of MBs coded by 2-D order-4, order-8, and order-16 transforms are compared on the H.264/AVC platform. The percentage of using 2-D order-4 ICT is very small as we expected. Therefore, in the proposed ABT scheme, the 2-D order-4 ICT in H.264/AVC is removed.

5.5.4 Subjective Evaluation

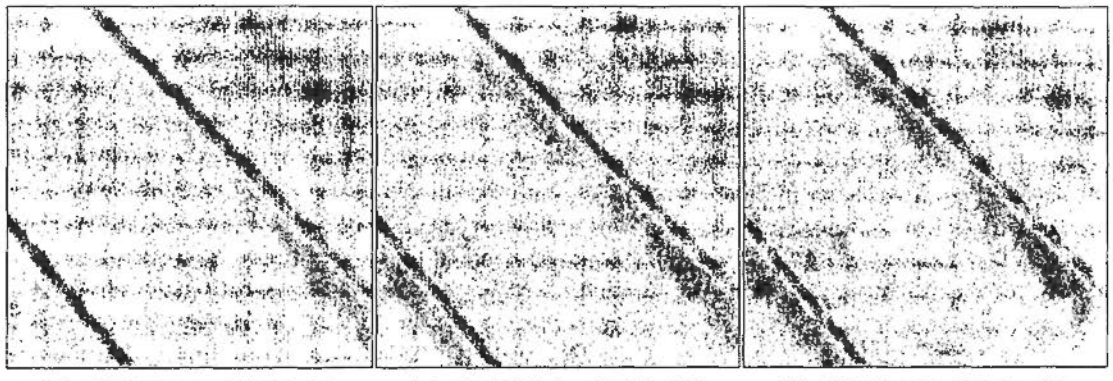
Using 2-D order-16 transform can preserve more details and provide better visual quality, especially at low bit-rate. Fig. 5.7 gives two examples for the subjective comparison. The images are all cropped from H.264 coded HD videos with size of 150×150 pixels. The indicated PSNR and number of bits are the R-D information of the associated frame. Obviously, the vertical edges of the buildings in *City* and the horizontal edges of the railway sleepers in *Station* are better preserved by the additional 2-D order-16 transforms. However, at high bit-rate, the visual quality without using 2-D order-16 transform is good enough and therefore the improved fidelity by using 2-D order-16 transforms is hard to observe visually.

5.5.5 Efficiency of the Fast Algorithm

If the transform is implemented by matrix multiplication, the complexity is high, due to many additions and multiplications introduced by the dot products. The fast algorithm



(a) 9400 bit, 34.14 dB (b) 7584 bit, 34.67 dB (c) 7600 bit, 34.52 dB



(d) 60208 bit, 37.36 dB (e) 64544 bit, 37.78 dB (f) 64072 bit, 37.77 dB

Figure 5.7: *Images cropped from City (720p) and Station (1080p) both coded with QP=32. (a) and (d) are coded using H.264/AVC HP. (b) and (e) are coded using H.264/AVC HP with additional 2-D order-16 NICT. (c) and (f) are coded using H.264/AVC HP with additional 2-D order-16 MICT.*

Operation	Fast MICT	Matrix multiplication
Addition	150	240
Multiplication	0	256
Shift	32	0

Table 5.6: *Operations of fast algorithm and matrix multiplication*

developed in Section 5.3.2 can implement the transformation process of the MICT by shifts and additions only. Table 5.6 compares the numbers of additions, shifts and multiplications required by transforming a 16-point vector using the fast algorithm and matrix multiplication, respectively. Without multiplication, the computational complexity is reduced significantly. However, the number of additions required for the fast algorithm is considerable, because multiplications with small magnitude integers are replaced by combinations of additions and shifts.

	Fast algorithm (s)	Matrix multiplication (s)	Saving time(%)
DCT	0.040	0.568	93%
MICT	0.031	0.282	89%
M_{8o}	0.005	0.026	77%

Table 5.7: *Execution time of fast algorithms and matrix multiplication*

The 2-D order-16 MICT was implemented using both fast algorithm and matrix multiplication by the C language. Ten thousand 16×16 random blocks were generated, in which the numbers were uniformly distributed from -256 to 255. The total execution time for transforming these blocks by the two methods was recoded respectively. The PC platform has a 3.2 GHz Intel Pentium 4 CPU and 1.0 GB RAM. Two Windows API functions QueryPerformanceFrequency and QueryPerformanceCounter are used for timing, which measure the execution time to nanosecond accuracy. The time comparison is shown in the third row of Table 5.7, where using fast algorithm can save about 89% of the computational time.

The efficiency of the fast algorithm for the MICT is compared with that of the fast DCT (FDCT) [17], because there is no publications related to fast algorithms for 2-D order-16 ICT, NICT or MICT. We did the experiment as described above for DCT, and the time for computing the 2-D order-16 DCT using FDCT and matrix multiplication are recorded, respectively. The second row of Table 5.7 shows FDCT can save 93% of the computational time compared with using matrix multiplication. The proposed fast algorithm for the 2-D order-16 MICT is 4% less efficient than the FDCT for DCT because of the different dyadic symmetries in $\mathbf{M}_{8o}$.

At the same time, we are also concerned with the efficiency of the fast algorithm used for the bottom-right module in Fig. 5.1 (a), which is specially designed for the multiplication with matrix $\mathbf{M}_{8o}$. Similarly, ten thousand 8×8 random blocks were generated, in which the numbers were uniformly distributed from -256 to 255. As shown in the fourth row of Table 5.7, the fast algorithm can save almost 77% computational time compared with matrix multiplication. This proves that the fast algorithm for $\mathbf{M}_{8o}$ has lower efficiency than that of the whole 2-D order-16 MICT. If the structure of the even part is different from $\mathbf{T}_{8e}$ in (5.8) and some existing fast algorithms of 2-D order-8 ICTs cannot be used, the method for decomposing $\mathbf{M}_{8o}$ has to be applied to the even part and the fast algorithm will be about 10% less efficient, but overall, the

computational complexity is reduced significantly.

5.6 Summary

In this chapter, we propose using 2D order-16 transforms for HD video coding, which are expected to be more efficient to exploit the higher spatial correlation. Specifically, two series of 2D order-16 transforms, NICT and MICT, are introduced. The former resembles DCT and is approximately orthogonal, of which the transform error introduced by the non-orthogonality is proven to be negligible. The latter modifies the structure of the DCT matrix and is inherently orthogonal, no matter what the values of the matrix elements are. Both types allow selecting matrix elements more freely by releasing the orthogonality constraint and can provide comparable performance with that of DCT. Each type is integrated into AVS EP and the H.264 HP, respectively, and used adaptively as an alternative to the 2D order-8 transform according to local activities. At the same time, many efforts have been devoted to further reducing the complexity of the 2D order-16 transforms and specially for MICT, a fast algorithm is developed and extended to a universal approach. Experimental results show that 2D order-16 transforms provide significant performance improvement for both AVS EP and H.264 HP, which means they can be efficient coding tools especially for HD video coding.

The fast algorithm of MICT was presented at *ISCAS2007* as an oral paper, entitled "A universal approach to developing fast algorithm for simplified order-16 ICT" [64]. All the work in this chapter has been published in *IEEE Transactions on Circuits and System for Video Technology* as a regular paper, entitled "2-D order-16 integer transforms for HD video coding" [45].

Chapter 6

Parametric Interpolation Filter for MCP

6.1 Introduction

Motion compensated prediction (MCP), a technique of predictive coding introduced in Section 2.3, is the key to the success of the modern video coding standards, as it removes the temporal redundancy in video sequences and reduces the size of bitstreams significantly. With MCP, the pixels to be coded are predicted from the temporally neighboring ones, and only the prediction errors and the MVs are transmitted. However, due to the finite sampling rate, the actual position of the prediction in the neighboring frames may be out of the sampling grid, where the intensity is unknown, so the intensities of the positions in between the integer pixels, called sub-positions, must be interpolated and the resolution of MV is increased accordingly. Fig. 6.1 shows an example of fractional pixel MCP, where the shaded and white circles represent integer- and half-pixels, respectively. The 4×4 block with bold boundary in Fig. 6.1 (a) is the current block to be coded, of which the target block, located at the half-pixel position in the reference frame (see the bold block in Fig. 6.1 (b)), has to be interpolated for MCP. Consequently, the resolution of the MV is increased to half-pixel. In the existing video coding standards, the interpolation filter is designed to fit the general statistics of various video sources, so the filter coefficients are fixed. Considering the time-varying statistics of video sources, some researchers propose using adaptive interpolation filter (AIF), of which the coefficients are analytically calculated for each frame using the linear minimum mean squared error (LMMSE) estimator (see Section 2.3), and then quantized and transmitted at the frame header.

The optimal interpolation filter for each frame based on the LMMSE criterion is derived in Section 6.2, by which the prediction error produced is used as the benchmark to evaluate the prediction error produced by other fixed and adaptive interpolation

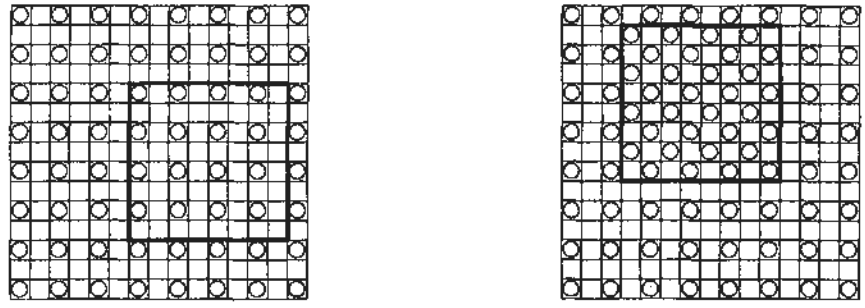


Figure 6.1: *Illustration of half-pixel MCP. (a) The block to be predicted in the current frame. (b) The target block located at the half-pixel position in the reference frame.*

techniques. Then, Section 6.3 reviews the existing AIF techniques, all of which are the approximations of the optimal interpolation filter with reduced support region, imposed symmetry constraints, and coarsely quantized filter coefficients, and result in larger prediction error compared with the benchmark. The formula to calculate the increased prediction error is also given in Section 6.3; according to the formula, it is inferred that the conflict of the coefficients' precision and the size of side information is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually. Then, in Section 6.4, parametric interpolation filter (PIF) is proposed, which not only tracks the non-stationary statistics of video sources as the existing AIF techniques do, but also represents each updated interpolation filter by parameters instead of individual coefficients, thus solving the conflict of the accuracy of coefficients and the size of side information. The experimental results are shown in Section 6.5.

6.2 Optimal Adaptive Interpolation Filter (AIF) Designing

Supposing the accuracy of MCP is in units of 1/4 distance between pixels, one should interpolate all the half- and 1/4-pixel samples in the reference frame for prediction, and therefore the reference frame is interpolated to be 16 times the spatial resolution, i.e., 4 times both sides. As shown in Fig. 6.2, such an interpolation process consists of two steps: upsampling the original reference frame to 16 times the spatial resolution by inserting zero-valued samples in the half- and 1/4-pixel sampling grids, which produces undesired spectra in the frequency domain, and then removing the undesired spectra

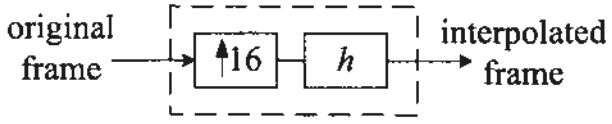


Figure 6.2: *The interpolation process of the optimal AIF*

by a lowpass filter h.

Using LMMSE estimator, the optimal h, denoted as h_{opt}, is achieved by minimizing the prediction error σ_e^2 as given in (2.57), where a_k, $k = 1, \cdots, K$, are the coefficients of h_{opt} and the set $\mathbf{s}_p$ represents the support region for generating s_0. To be consistent with the interpolation in H.264/AVC, the support region for each sub-position is the surrounding 6×6 integer pixels, and therefore the size of h in Fig. 6.2 is 23×23.

6.2.1 Optimal AIF for P-frames

Let P and S be the reference frame and the current frame, respectively. Upsampling P and S by a factor 16 using zero-insertion and zero-order holding, respectivley, we get P_{16} and S_{16}. Note P_{16} is where h in Fig. 6.2 applies. Then, the prediction error σ_e^2 can be expressed by (6.1)

$$\sigma_e^2 = E\left[\left(\sum_{i=-11}^{11}\sum_{j=-11}^{11} h(i,j)P_{16}(x-i+d_x, y-j+d_y) - S_{16}(x,y)\right)^2\right] \tag{6.1}$$

where (x,y) indicates the spatial coordinate and (d_x, d_y) includes the two components of MV, which has 1/4-pixel resolution. Letting $\partial\sigma_e^2/\partial h(m,n)$ equal to 0, one can derive the minimum σ_e^2 and the optimal interpolation filter h_{opt}, by the solution of the Wiener-Hopf equations in (6.2),

$$\sum_{i=-11}^{11}\sum_{j=-11}^{11} h(i,j)R_{pp}(i-m, j-n) = R_{ps}(m,n),\ \forall m,n,\ -11 \le m,n \le 11 \tag{6.2}$$

where R_{pp} and R_{ps} represent the autocorrelation of P_{16} and the motion-compensated cross-correlation of P_{16} and S_{16}, respectively. R_{pp} and R_{ps} are calculated with all the MVs for the current frame known; therefore, ME is performed before starting coding the current frame. The detailed reasoning steps, skipped here, are similar to those in Section 2.3. Different from the solution (2.52) in Section 2.3, the autocorrelation R_{pp} here is a four-dimensional function instead of a matrix and R_{ps} is a matrix, not a vector.

In order to use matrix operation to solve the problem as in (2.55), the dimensions of R_{pp}, R_{ps}, and h in (6.2) are all reduced as in (6.3), (6.4), and (6.5), respectively,

$$\mathbf{R}_{pp}[(i+11)\times 23+(j+11),(m+11)\times 23+(n+11)] = R_{pp}(i-m,j-n) \tag{6.3}$$

$$\mathbf{R}_{ps}[(m+11)\times 23+(n+11)] = R_{ps}(m,n) \tag{6.4}$$

$$\mathbf{h}[(i+11)\times 23+(j+11)] = h(i,j) \tag{6.5}$$

such that

$$\mathbf{h} = \mathbf{R}_{pp}^{-1}\mathbf{R}_{ps}. \tag{6.6}$$

With $\mathbf{h}$ known, h_{opt} is finally obtained based on (6.5).

6.2.2 Optimal AIF for B-frames

For B-frames, forward, backward, and bi-directional MCPs are allowed, where the former two can follow the solution introduced in Section 6.2.1 to derive the optimal AIF. For bi-directional MCP, the solution of the optimal AIF is modified. Denoting $P_{16,f}$ and $(d_{x,f}, d_{y,f})$ as the upsampled reference frame and MV for forward MCP, respectively, and $P_{16,b}$ and $(d_{x,b}, d_{y,b})$ for the backward case, one should first re-write (6.1) to (6.7).

$$\begin{aligned}\sigma_e^2 = E\Bigg[\Bigg(&\frac{1}{2}\sum_{i=-11}^{11}\sum_{j=-11}^{11} h(i,j)\,(P_{16,f}(x-i+d_{x,f}, y-j+d_{y,f}) \\ &+ P_{16,b}(x-i+d_{x,b}, y-j+d_{y,b})) - S_{16}(x,y)\Bigg)^2\Bigg]\end{aligned} \tag{6.7}$$

Similarly, letting $\partial\sigma_e^2/\partial h(m,n)$ equal to 0, one finally derives h_{opt} by the equations in (6.8) and then obtains the minimum σ_e^2.

$$\begin{aligned}0 &= \frac{\partial\sigma_e^2}{\partial h(m,n)} \\ &\Rightarrow \sum_{i=-11}^{11}\sum_{j=-11}^{11} h(i,j)\left[\frac{1}{2}R_{ff}(i-m,j-n) + \frac{1}{2}R_{bb}(i-m,j-n) + R_{fb}(i-m,j-n)\right] \\ &= R_{fs}(m,n) + R_{bs}(m,n), \qquad \forall m,n,\ -11 \le m,n \le 11\end{aligned} \tag{6.8}$$

In (6.8), R_{ff} and R_{bb} represent the autocorrelations of the forward and backward upsampled reference frames, $P_{16,f}$ and $P_{16,b}$, respectively, and R_{fb}, R_{fs}, and R_{bs} are

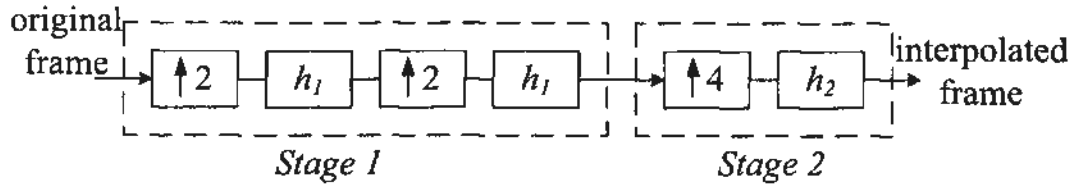


Figure 6.3: *The normative interpolation in H.264/AVC*

the motion-compensated cross-correlations of $P_{16,f}$ and $P_{16,b}$, $P_{16,f}$ and S_{16}, and $P_{16,b}$ and S_{16}, respectively. According to (6.7), deriving h_{opt} for B-frames relies not only on the autocorrelation of pixels in each reference frames, i.e., R_{ff} and R_{bb}, and the cross-correlation of pixels in the reference and current original frames, i.e., R_{fs} and R_{bs}, but also on the cross-correlation of the pixels in forward and backward reference frames, i.e., R_{fb}.

6.3 Approximation to Optimal AIF and the Effects

If no symmetry constraint and quantization are imposed, the optimal AIF h_{opt} obtained in Section 6.2 has 23^2 different real-valued coefficients, which are too expensive to be coded for each frame. To reduce the side information representing AIF, the related work restricts the support region, imposes the symmetry constraints, and quantizes the coefficients, leading to an approximation of h_{opt} and larger prediction error. Section 6.3.1 reviews the related work, and the increased prediction error caused by using an approximation of h_{opt} is analyzed in Section 6.3.2.

6.3.1 Related Work

The normative interpolation defined in H.264/AVC, having been introduced in Section 3.3.2, includes two stages, which interpolate the half-pixel and 1/4-pixel sub-positions, respectively, as shown in Fig. 6.3. The interpolation in the first stage is separable, which means the sampling rate in one direction is doubled by inserting zero-valued samples followed by filtering using a 1-D filter h_1, [1, 0, -5, 0, 20, 32, 20, 0, -5, 0, 1]/32, and then the process repeats for the other direction. The second stage is non-separable and uses bilinear filtering supported by the integer pixels and the interpolated half-pixel values. The filter coefficients are fixed for all video sequences. The impulse response and the frequency response of the normative interpolation filter in

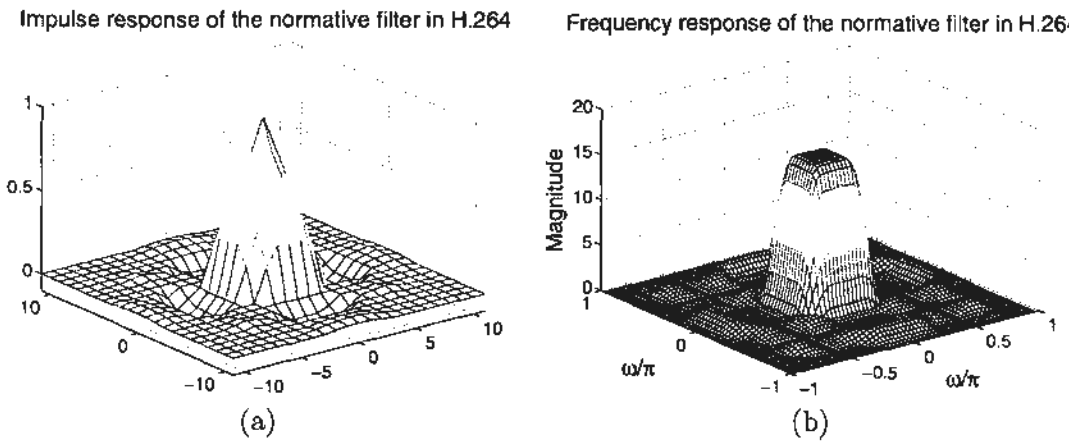


Figure 6.4: *The impulse and frequency responses of the normative interpolation filter in H.264/AVC*

H.264/AVC are shown in Fig. 6.4 (a) and (b), respectively. As can be seen, this interpolation filter is almost ideal, as it has a square passband with the cutoff frequencies $\pi/4$ in both horizontal and vertical directions and very small ripples in the stopband.

Recently, much research effort has been devoted to the design of AIF, which considers the non-stationary statistics of video signals and allows filter coefficients to be adjusted for every frame. All the AIF techniques design the interpolation filter for each frame using the aforementioned LMMSE estimator, but use different ways to reduce the side information, such as reducing the support region, imposing the symmetry constraints, and coarsely quantizing the filter coefficients.

AIF in [65] follows the process in Fig. 6.3. Only h_1 is modified for every frame. Owing to the even symmetry of h_1, only three coefficients are analytically calculated and coded.

Vatis *et al.* [66; 67; 68] developed a 2-D non-separable interpolation filter, of which the interpolation process is shown in Fig. 6.2. The spatial sampling rate is increased 16 times at one time and each sub-position can be interpolated directly by filtering the surrounding 6×6 integer pixels. h is circularly symmetric, because the spatial statistics are assumed to be isotropic.

To reduce the complexity of 2-D non-separable interpolation filter, 2-D separable AIF [69] is proposed, in which the spatial statistics of horizontal and vertical directions are considered different and the interpolation filters for the two directions are separately designed. As shown in Fig. 6.5, the horizontal sampling rate is increased four times by

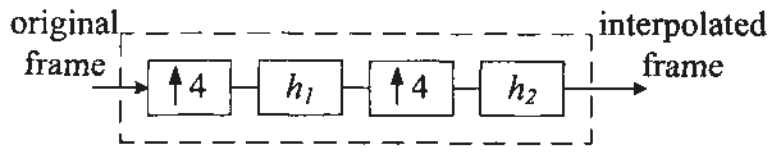


Figure 6.5: *The interpolation process in Separable AIF [69]*

zero-insertion and a 1-D filter h_1 is specially designed for the current frame. Then, the process repeats for the vertical direction using h_2.

A directional AIF (DAIF) is proposed in [70; 71; 72; 73], which also follows the process in Fig. 6.2 (b), but is much simpler than [66]. Furthermore, it is not even a 2-D interpolation filter; the support region for each sub-position is restricted to 1-D aligning integer pixels only. By doing this, the correlation along neither the horizontal nor the vertical direction can also be exploited for interpolation.

Instead of adaptively calculating the filter coefficients for each frame, switched interpolation filter with offset (SIFO) [74; 75] provides the capability of switching between fixed filters and sending DC offsets at the sub-position level. The three fixed filters include the one in H.264/AVC and the other two non-separable filters with size 4×4. When not using any of the filters, the pixels in different sub-positions and the integer position will be added by their relevant DC offsets, which are transmitted at the frame level, in order to compensate the illumination changes.

In some AIF techniques, SIFO is used to enhance the performance of DAIF. Fixed directional interpolation filter (FDIF) is proposed in [76; 77], where the support region for each sub-position is the same as that in DAIF, but the filter coefficients are fixed. The regular FDIF is further improved in two ways. First, one of the 15 sub-positions is defined as a strong filter position (SFP), filtered using a dedicated filter for this particular position. Second, fixed DC offsets are defined for the rest sub-positions. The enhanced DAIF proposed in [78; 79] adds a 5×5 filter for integer pixels and a DC offset to each integer and sub-position.

Apart from reducing the support region and imposing symmetry constraints, all coefficients in the existing AIF techniques are quantized to 512 levels. However, 9-bit representation is not enough to represent the filter coefficients and introduces significantly larger prediction error. That is why the performances of the aforementioned AIF techniques are very close to each other, no matter the size of the support region

and whether symmetry constraints are imposed. Although 9-bit representation is not precise enough, the required bits for coding these coefficients are still significant especially at low bit-rates. Therefore, the conflict of the coefficients' precision and the size of side information is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually.

6.3.2 Approximation Effects

The related work [65]- [79] makes some effort to reduce the side information as introduced in Section 6.3.1, leading to an approximation of h_{opt}, denoted as $\tilde{h}$. The difference between h_{opt} and $\tilde{h}$ is denoted as h_Δ, i.e., $h_\Delta = h_{opt} - \tilde{h}$, and the increased energy of prediction error, Δerr, is given in (6.9).

$$
\begin{aligned}
\Delta err &= \sigma_e^2|_{\tilde{h}} - \sigma_e^2|_{h_{opt}} \\
&= E\left[(\sum_{i,j} \tilde{h}(i,j) P_{16}(x-i+d_x, y-j+d_y) - S_{16}(x,y))^2 \right. \\
&\quad \left. - (\sum_{i,j} h_{opt}(i,j) P_{16}(x-i+d_x, y-j+d_y) - S_{16}(x,y))^2 \right] \\
&= E\left[(\sum_{i,j} (h_{opt}(i,j) - h_\Delta(i,j)) P_{16}(x-i+d_x, y-j+d_y) - S_{16}(x,y))^2 \right. \\
&\quad \left. - (\sum_{i,j} h_{opt}(i,j) P_{16}(x-i+d_x, y-j+d_y) - S_{16}(x,y))^2 \right] \\
&= \sum_{i,j} \sum_{m,n} h_\Delta(i,j) h_\Delta(m,n) R_{pp}(i-m, j-n) \\
&\quad -2 \sum_{m,n} h_\Delta(m,n) \left(\sum_{i,j} h_{opt}(i,j) R_{pp}(i-m, j-n) - R_{ps}(m,n) \right)
\end{aligned}
\tag{6.9}
$$

Since h_{opt} is the solution of (6.2), the latter part in (6.9) is equal to zero and Δerr is finally expressed by (6.10).

$$
\Delta err = \sum_{i,j} \sum_{m,n} h_\Delta(i,j) h_\Delta(m,n) R_{pp}(i-m, j-n) \tag{6.10}
$$

Δerr introduced by different interpolation techniques is studied, based on the first ten frames (except I-frame) of a set of ten HD sequences. Fig. 6.6 gives the typical

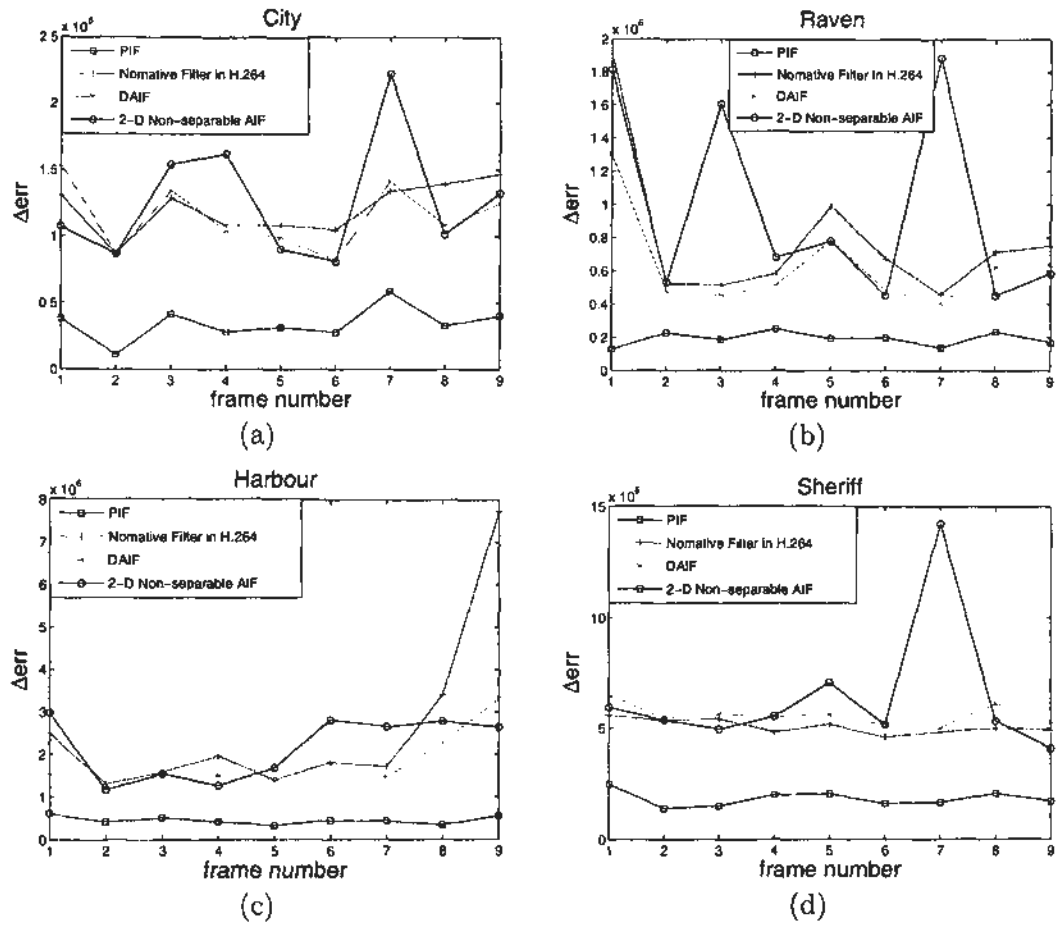


Figure 6.6: *Δerr introduced by different AIF techniques (QP_P=28, QP_B=29)*

results on four out of the ten sequences. Generally, Δerr introduced by 2-D non-separable AIF and DAIF is close to or even larger than that introduced by the standard filter in H.264/AVC. Especially, 2-D non-separable AIF, which has larger support region than DAIF and is expected to perform better, is more likely to have particularly large Δerr. That is because $\mathbf{R}_{pp}$ in (6.3) is ill-conditioned, so any slight change in h_{opt}, having been denoted as h_Δ, will influence Δerr significantly. Furthermore, 2-D non-separable AIF has much more coefficients involved in quantization; quantization error influences the value of Δerr more significantly than the support region and the imposed symmetry constraint. Therefore, the 9-bit uniform quantization used in the existing AIF techniques cannot produce precise enough filter coefficients, which actually causes the value of Δerr to increase dramatically and frequently disables the AIF mode at the frame level during the encoding process, based on the R-D criterion [68].

6.4 Proposed Parametric Interpolation Filter (PIF)

In the related work, the accuracy of the coefficients and the size of the side information are conflicting, because the coefficients are coded individually. In this work, the impulse response of an interpolation filter is represented by a function h_f, determined by five parameters, and h_{opt} is approximated by tuning the parameters to minimize Δerr in (6.10). When the parameters are determined, the filter coefficients are calculated as the function values. Obviously, the side information for coding five parameters is very small. Yet the accuracy of the coefficients can also be guaranteed, if the parameters are quantized in enough precision. How to find the form of h_f is presented in Section 6.4.1 and how to determine the parameters' values for a certain filter and then code them is introduced in Section 6.4.2.

6.4.1 Function Representation of an Interpolation Filter

Let $F(e^{j\omega_x}, e^{j\omega_y})$ be the Fourier transform of the reference frame P. After upsampling P using zero-insertion, shown in Fig. 6.2, the Fourier transform of the upsampled frame P_{16}, denoted as $F_{16}(e^{j\omega_x}, e^{j\omega_y})$, is given by (6.11),

$$\begin{aligned} F_{16}(e^{j\omega_x}, e^{j\omega_y}) &= \sum_{x=0}^{4w-1}\sum_{y=0}^{4h-1} P_{16}(x,y)e^{-j\omega_x x}e^{-j\omega_y y} \\ &= \sum_{x=0}^{4w-1}\sum_{y=0}^{4h-1}\left(\sum_{k=0}^{w-1}\sum_{l=0}^{h-1} P(k,l)\delta(x-4k, y-4l)\right)e^{-j\omega_x x}e^{-j\omega_y y} \\ &= \sum_{k=0}^{w-1}\sum_{l=0}^{h-1} P(k,l)e^{-j4\omega_x k}e^{-j4\omega_y l} = F(e^{j4\omega_x}, e^{j4\omega_y}) \end{aligned} \tag{6.11}$$

where w and h mean the width and height of the frame in the unit of pixel, respectively. According to (6.11), F_{16} is a frequency-scaled version of F. Fig. 6.7 (a) and (b) give examples of F and its corresponding F_{16}, respectively. In Fig. 6.7 (b), the undesired spectra centering at integer multiples of $(\pi/2, \pi/2)$, i.e., the original sampling rate, are introduced by the zero-insertion upsampling and should be removed. This requires a lowpass filter h (see Fig. 6.2 (b)) with a gain of 16 and a cutoff frequency $\pi/4$, of which the ideal frequency response is shown in Fig. 6.7 (c). It is similar to the frequency response of the normative interpolation filter in H.264/AVC (see Fig. 6.4), by which the information in F is completely preserved.

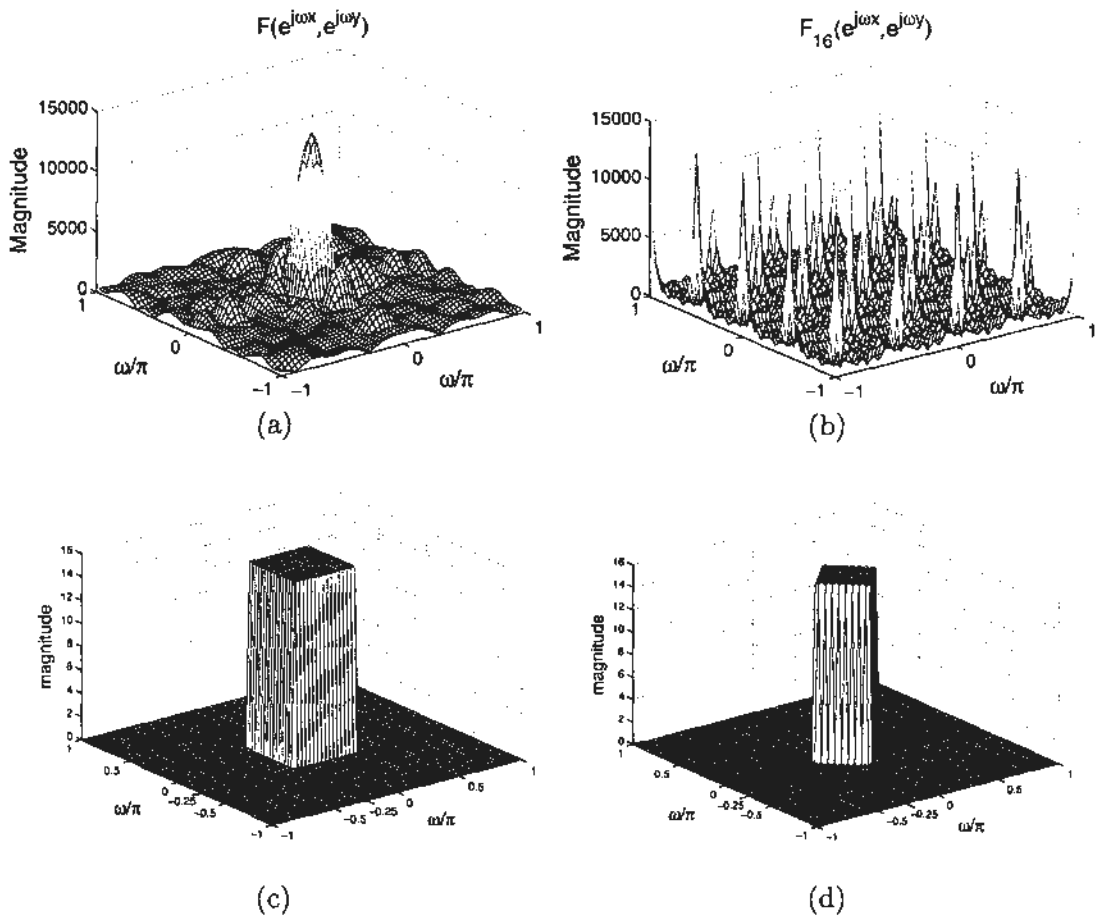


Figure 6.7: *The Fourier transform of original and upsampled frames and the frequency responses of ideal interpolation filters*

Considering the special PSD of HD videos as shown in Section 4.2, where the high frequency energies are mainly distributed in the horizontal and vertical directions, we propose the desired filter with a diamond-shaped passband (see Fig. 6.7 (d)), such that high frequency components neither in the horizontal direction nor in the vertical direction are filtered out. There are two reasons for proposing the filter with such a passband. Firstly, interpolation in the context of video coding is for a better MCP, and the high frequency components are more likely to introduce large prediction error, thus exerting negative influence on MCP. Secondly, high frequency components outside the diamond-shaped passband have very low energy and contribute little to the content of the frame. This passband shape also agrees with our observation on a number of optimal filters.

Theoretically, the cutoff frequency should be $\pi/4$, such as the one in Fig. 6.7 (d),

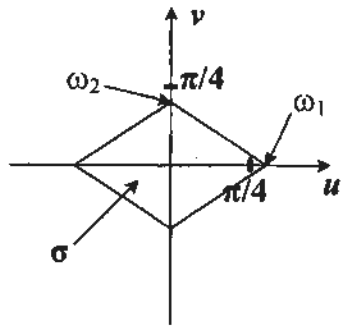


Figure 6.8: *The passband of the proposed ideal interpolation filter*

although, in practice, they may vary around $\pi/4$. As shown in Fig. 6.8, two parameters, ω_1 and ω_2, are used to denote the cutoff frequencies at two axes and the shaded diamond-shaped area, σ, represents the passband. The corresponding impulse response, h_d, can be obtained by inverse Fourier transform, as expressed in (6.12),

$$h_d(m,n) = \frac{1}{4\pi^2} \int_{[-\pi,\pi]^2} H_d(e^{ju}, e^{jv}) e^{jmu+jnv} du dv \tag{6.12}$$

where H_d is given as below.

$$H_d(e^{ju}, e^{jv}) = \begin{cases} 16, \text{if } (u,v) \in \sigma \\ 0, \quad \text{otherwise} \end{cases} \tag{6.13}$$

Substituting (6.13) into (6.12), one can re-write (6.12) as in (6.14).

$$\begin{aligned} h_d(m,n) &= \frac{1}{4\pi^2} \left(\int_{-\omega_1}^{0} \int_{-\frac{\omega_2}{\omega_1}u-\omega_2}^{\frac{\omega_2}{\omega_1}u+\omega_2} 16 e^{jmu+jnv} du dv + \int_{0}^{\omega_1} \int_{\frac{\omega_2}{\omega_1}u-\omega_2}^{-\frac{\omega_2}{\omega_1}u+\omega_2} 16 e^{jmu+jnv} du dv \right) \\ &= \frac{4}{\pi^2} \left(\int_{-\omega_1}^{0} e^{jmu} du \int_{-\frac{\omega_2}{\omega_1}u-\omega_2}^{\frac{\omega_2}{\omega_1}u+\omega_2} e^{jnv} dv + \int_{0}^{\omega_1} e^{jmu} du \int_{\frac{\omega_2}{\omega_1}u-\omega_2}^{-\frac{\omega_2}{\omega_1}u+\omega_2} e^{jnv} dv \right) \end{aligned} \tag{6.14}$$

After some reasoning steps, one can finally find the formula of h_d as shown in (6.15),

$$h_d(m,n) = \frac{8\omega_1\omega_2}{\pi^2} \text{Sinc}\left(\frac{\omega_1 m + \omega_2 n}{2}\right) \text{Sinc}\left(\frac{\omega_1 m - \omega_2 n}{2}\right), \quad -\infty < m,n < \infty \tag{6.15}$$

where function Sinc(.) is defined as follows.

$$\text{Sinc}(x) = \begin{cases} \frac{\sin(x)}{x}, & \text{if } x \neq 0 \\ 1, & \text{otherwise} \end{cases} \tag{6.16}$$

h_d has to be truncated by an appropriate window function w before being used for

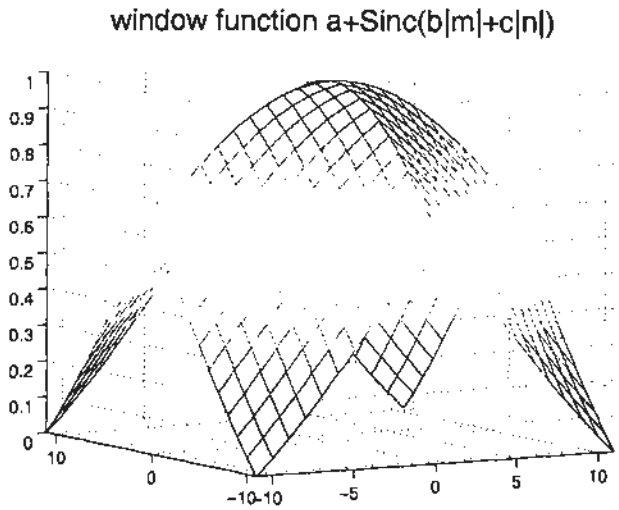


Figure 6.9: *The proposed window function for PIF*

interpolation. In this work, w, defined as in (6.17), is proposed, which is empirically better than other widely used ones, such as rectangular, triangular, Hanning, and Blackman window functions.

$$w(m,n) = \begin{cases} a + \text{Sinc}(b|m| + c|n|), & \text{if } -11 \leq m,n \leq 11 \\ 0, & \text{otherwise} \end{cases} \tag{6.17}$$

Then, the approximation of the ideal desired filter, denoted as h_f, is obtained as the product of h_d and w, as in (6.18).

$$\begin{aligned} h_f(m,n) &= h_d(m,n)w(m,n) \\ &= \begin{cases} N\text{Sinc}(\frac{\omega_1 m+\omega_2 n}{2})\text{Sinc}(\frac{\omega_1 m-\omega_2 n}{2})\,(a + \text{Sinc}(b|m| + c|n|)), & \text{if } -11 \leq m,n \leq 11 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.18}$$

Therefore, h_f is determined by a parameter set, $\mathbf{x} = \{\omega_1, \omega_2, a, b, c\}$, and the factor N guarantees the filter gain is 16. h_f is the interpolation filter proposed in this work, which represents filters by parameters rather than individual coefficients and is named as parametric interpolation filter (PIF).

6.4.2 Parameter Determination and Coding

As the form of h_f has been given in (6.18), the parameter set $\mathbf{x}$ has to be determined for each frame and coded. The optimal $\mathbf{x}$ should make h_f reduce as much prediction

error as h_{opt} can, by achieving the minimum of Δerr in (6.10), i.e.,

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \Delta err \tag{6.19}$$

where h_Δ in (6.10) can be expressed as in (6.20)

$$h_\Delta(i,j) = h_{opt}(i,j) - h_f(i,j|\mathbf{x}), \quad -11 \leq i,j \leq 11 \tag{6.20}$$

The minimum is achieved by using the BFGS Quasi-Newton method. The family of Quasi-Newton methods, well-known for finding local maximum and minimum of functions, are based on Newton's method to find the stationary point of a function, where the gradient is zero. Newton's method assumes that the function can be locally approximated as a quadratic Taylor expansion in the region around the optimum, and use the first and second derivatives, i.e., gradient and Hessian, to find the stationary point. In Quasi-Newton methods, the Hessian matrix of the objective function does not need to be computed. Instead, it is updated by analyzing successive gradient vectors instead. Different Quasi-Newton methods have different Hessian matrix updating algorithms. In this work, BFGS algorithm is used for the Quasi-Newton method and the procedure is briefly introduced as below.

From an initial guess of $\mathbf{x}_0$ and an approximate Hessian matrix $\mathbf{B}_0$, the following steps are repeated until $\mathbf{x}$ converges to the solution of the objective function $f(\mathbf{x})$, which is Δerr here.

1. Obtain a search direction $\mathbf{s}_k$ by solving $\mathbf{s}_k = -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}_k)$.

2. Perform a line search to find the optimal stepsize α_k, such that $f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)$ achieves the minimum in the search direction $\mathbf{s}_k$ found in the first step. The line search algorithm used finds a stepsize satisfying the strong Wolfe conditions, introduced in [80].

3. Update $\mathbf{x}_{k+1}$ by (6.21).

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k \tag{6.21}$$

4. Calculate the intermediate vector $\mathbf{y}_k$ by (6.22), which will be used to update $\mathbf{B}_k$.

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \tag{6.22}$$

5. $\mathbf{B}_k$ is updated by (6.23) for the next iteration.

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \tag{6.23}$$

As only the inverse of $\mathbf{B}_k$ is used in this algorithm, it is more efficient to update $\mathbf{B}_k^{-1}$ directly by (6.24).

$$\mathbf{B}_{k+1}^{-1} = (I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) \mathbf{B}_k^{-1} (I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{6.24}$$

The detailed description is omitted here, as this method is not part of the contribution in this work. Interested readers are referred to [80] to more in-depth illustrations.

Since this numerical method is used to solve the local minimum problem, the initial estimates of $\mathbf{x}$ and $\mathbf{B}_0$ become critical. At the beginning of coding a video sequence, the initial estimates of ω_1 and ω_2 are both 0.25π, and empirical initial values 0.1, 0.15, and 0.15 are assigned to a, b, and c, respectively. During the coding process, the initial estimate keeps updated by the value of $\hat{\mathbf{x}}$ of the latest P-frame using AIF mode. $\mathbf{B}_0$ is initialized by the identity matrix $\mathbf{I}$, so that the first step is equivalent to a gradient descent, but further steps are refined more and more by $\mathbf{B}_k$.

As the BFGS method is for unconstrained optimization, the solution $\mathbf{x}$ will theoretically have any real numbers. However, the values of ω_1 and ω_2 are expected to be around $\pi/4$ and the valid range is $[0, \pi]$. The values of a, b, and c are around zero. Therefore, in case any of ω_1 and ω_2 is larger than π or any of a, b, and c has the absolute value larger than one, the solution is discarded and consequently the PIF mode in the current frame is disabled. However, based on our experiments, such case has never happened. The magnitude of each parameter is uniformly quantized to 8196 steps, coded by 13-bit FLC. To indicate the signs of a, b, and c, three additional bits are also coded. Therefore, the side information of interpolation filter is exactly 68 bits for each frame, which is about 20% of AIF in [66].

6.5 Experimental Results

6.5.1 Comparison with the Optimal AIF

Since PIF, like other existing AIF techniques, is the approximation of the optimal AIF h_{opt}, this sub-section uses the energy of the prediction error produced by the optimal

HD sequences	Filter in H.264	2D Non-sep. AIF	DAIF	PIF
BigShip	5.446	10.308	6.068	2. 033
City	12.088	12.644	11.485	3.428
Crew	24.187	8.581	6.879	1.728
Harbour	25.986	21.690	17.575	4.527
Jet	1.627	4.122	2.072	0.960
Optis	5.448	6.520	5.029	1.996
Raven	7.916	9.765	6.325	1.914
Sailormen	11.857	12.031	13.534	5.716
Sheriff	5.082	6.418	5.604	1.831
ShuttleStart	1.977	4.083	1.847	0.918

Table 6.1: *Average Δerr produced by different interpolation filters (10^5)*

AIF denoted by $\sigma_e^2|_{h_{opt}}$ as the benchmark to evaluate the performance of the proposed PIF and other AIF techniques, measured by the increased error energy denoted by Δerr. How to calculate the value of Δerr for a certain frame has been introduced in detail in Section 6.3.2, where part of the experimental results has also been given. Ten HD sequences are used in the experiment; for each sequence, Δerr produced by different AIF techniques, including PIF, are calculated for the first ten frames (except I-frame). As shown in Fig. 6.6, Δerr produced by PIF is always much less than those provided by 2-D non-separable AIF [66], DAIF [70], and the normative one used in H.264/AVC, no matter the sequences contain rich details, such as *City* and *Harbour*, or contain large smooth areas, such as *Raven* and *Sheriff*. Fig. 6.6 also implies that replacing the normative interpolation filter by PIF in H.264/AVC can significant reduce the energy of the prediction error and improve the overall R-D performance, which will be verified in Section 6.5.2. For all the test sequences, the average Δerr on the first ten frames of each sequence is given in Table 6.1, in the unit of 10^5. As can be seen, values of Δerr produced by 2-D non-separable AIF and DAIF are larger than those produced by the interpolation filter in H.264/AVC on average, owing to the extremely large values for some frames. However, large prediction errors produced by the AIF mode will not cause performance loss during the encoding, because the AIF mode can be disabled at the frame level, if necessary, based on the R-D criterion [68].

Fig. 6.10 (a)-(d) show the impulse responses of four interpolation filters: DAIF, 2-D non-separable AIF, PIF and h_{opt}, respectively, for coding the first P-frame of *City* with QP equal to 28. Their corresponding frequency responses are given in Fig. 6.10

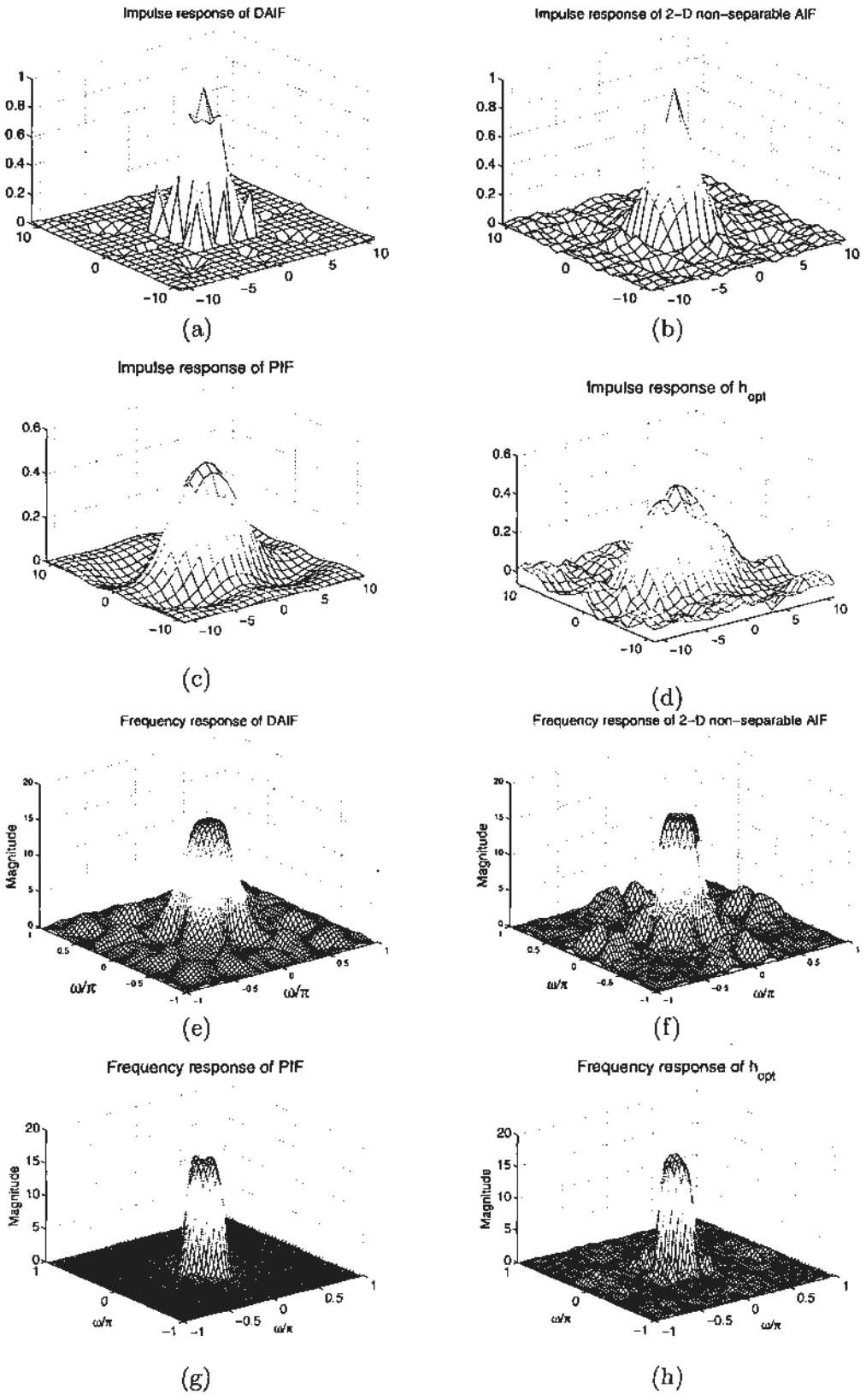


Figure 6.10: *Impulse and frequency response by different interpolation filters on City*

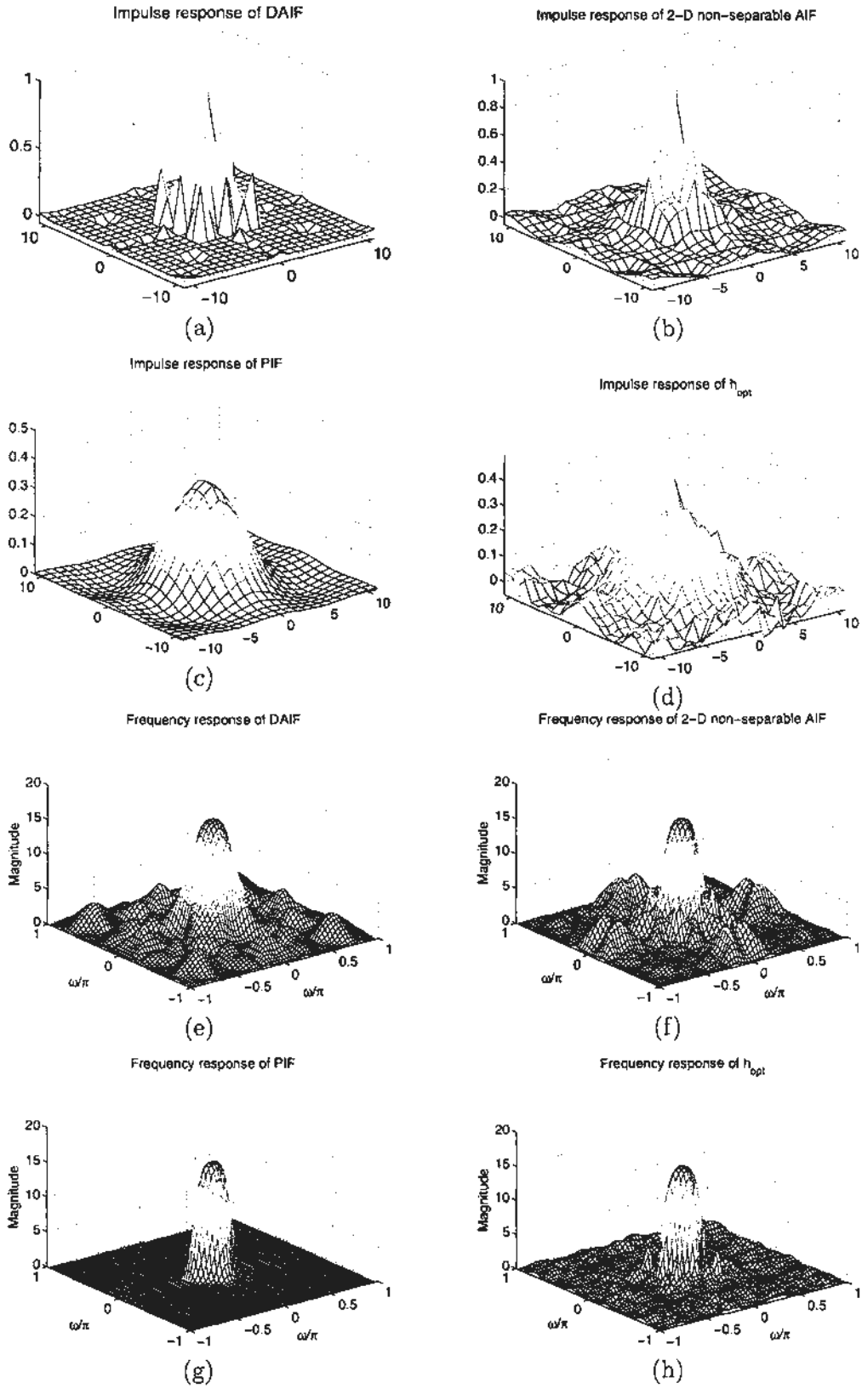


Figure 6.11: *Impulse and frequency response by different interpolation filters on Harbour*

Test sequence	1280×720 progressive
Sequence structure	IBBPBBP...
Intra frame period	Only the first frame
Entropy coding	CABAC
FME	on
R-D optimization	on
Adaptive rounding	off
QP	I(22, 27, 32, 37) P(23, 28, 33, 38) B(24, 29, 34, 39)
Reference frame	4
Search range	±64
Frame number	60

Table 6.2: *Test conditions*

(e)-(h), respectively. The DAIF (see (a) and (e)) and 2-D non-separable AIF (see (b) and (f)) are quite different from h_{opt} (see (d) and (h)) and introduce undesired high frequency components, whereas PIF (see (c) and (g)) resembles h_{opt} with high frequency components filtered out. The parameter set $\mathbf{x}$ of the PIF in Fig. 6.10 is $\{0.202\pi, 0.302\pi,$ -0.038, 0.191, 0.065$\}$. Another example of the impulse and frequency responses of these four interpolation filters under the same test conditions but on the sequence *Harbour* is given in Fig. 6.11, where the parameter set $\mathbf{x}$ of the PIF is determined to be $\{0.242\pi,$ 0.179π, 0.430, 0.059, 0.355$\}$. Similar observations can be obtained.

6.5.2 Improvement of R-D Performance

The proposed PIF is integrated into the VCEG's reference software KTA2.0 and Table 6.2 shows the test conditions. In the encoder, the MVs estimated in the first pass are reused in the second pass in order to show PIF's performance with one-pass encoding strategy. All the filters in comparison, including the optimal filter, PIF, DAIF, and 2-D non-separable AIF, are implemented using 32-bit integer arithmetic. The AIF mode can be disabled based on the criterion of frame level R-D distortion [68].

Table 6.3 compares the R-D performances provided by the four interpolation techniques, which are measured by the average percentage of bit-rate reduction at the same PSNR or, equivalently, by the average PSNR improvement at the same bit-rate, using the method proposed in [63]. The benchmark is H.264/AVC High Profile.

The bit-rate reduction of 2-D non-separable AIF and DAIF is about 2.5% on average, because they cannot efficiently reduce the prediction error energy, as introduced in

HD Sequence	2D Non-sep. AIF		DAIF		PIF		h_{opt}	
	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)
BigShip	-1.07	0.03	-1.53	0.04	-8.24	0.23	-9.23	0.26
City	-4.63	0.15	-4.30	0.13	-15.27	0.52	-17.06	0.57
Crew	-4.77	0.12	-4.65	0.12	-9.59	0.25	-14.08	0.38
Harbour	-4.65	0.17	-3.38	0.12	-7.87	0.30	-10.50	0.40
Jet	-0.38	0.03	-1.63	0.04	-8.84	0.26	-11.26	0.31
Optis	-1.12	0.03	-1.16	0.03	-6.65	0.16	-7.52	0.19
Raven	-2.03	0.07	-4.72	0.18	-11.37	0.46	-14.67	0.60
Sailormen	-0.80	0.02	-0.49	0.01	-7.43	0.20	-8.82	0.23
Sheriff	-1.97	0.05	-1.53	0.04	-5.95	0.18	-7.71	0.21
ShuttleStart	-3.13	0.11	-3.21	0.11	-11.15	0.42	-13.06	0.46
Average	**-2.46**	**0.08**	**-2.66**	**0.08**	**-9.24**	**0.30**	**-11.39**	**0.36**

Table 6.3: *The R-D performances of DAIF, 2-D non-separable AIF, PIF, and h_{opt}*

HD sequences	2D Non-sep. AIF	DAIF	PIF	h_{opt}
BigShip	25.0	55.3	84.6	94.3
City	41.7	63.2	99.1	100
Crew	28.5	48.2	81.6	91.2
Harbour	53.1	74.1	99.1	100
Jet	15.4	51.3	78.5	96.1
Optis	39.0	63.6	82.5	95.6
Raven	33.8	78.9	99.1	100
Sailormen	36.0	50.9	86.8	99.6
Sheriff	29.8	50.4	83.3	96.9
ShuttleStart	2.7	44.3	87.7	100
Average	**30.5**	**58.0**	**88.2**	**97.4**

Table 6.4: *The frequencies of occurrence of DAIF, 2-D non-separable AIF, PIF, and h_{opt} (%)*

Section 6.3.2 and verified in Section 6.5.1. The optimal AIF h_{opt} is also used for MCP, but the bitstream is non-decodable, because the coefficients of h_{opt} are not quantized or transmitted. As shown in the eighth and ninth columns in Table 6.3, the bit-rate reduction provided by h_{opt} is about 11.39%, equivalent to 0.36 dB PSNR improvement; the performance can be taken as the upper bound of the R-D performance. The performance of the proposed PIF is always much better than DAIF and 2-D non-separable AIF for all the test sequences. On average, the bit-rate reduction provided by PIF is 9.24%, about 7% more than the two AIFs and approaches the upper bound.

In Table 6.4, the frequencies of occurrence of different interpolation techniques give further evidence for the efficiency of the proposed PIF. The percentage that the coded

frames choose the PIF mode instead of the normative interpolation in H.264/AVC varies from 78.5% to 99.1%, depending on different test sequences, which is close to the frequency of occurrence of h_{opt}. However, DAIF and 2-D non-separable AIF are much less used. Especially for the latter, the average frequency of occurrence is only 30.3%.

6.6 Summary

This chapter first reviews the existing AIF techniques, and then points out that the conflict of the accuracy of coefficients and the size of side information is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually. To resolve this conflict, PIF is proposed for MCP, which represents interpolation filters by a function determined by five parameters instead of by individual coefficients. The function is designed based on the fact that high frequency energy of HD video source is mainly distributed along the vertical and horizontal directions, and the parameters are tuned to minimize the energy of prediction error, thus approaching the optimal interpolation filter. The experimental results show that PIF significantly outperforms the existing AIF techniques and approaches the efficiency of the optimal filter.

The work proposed in this chapter will be presented at *ICIP2009* as a poster paper, entitled "Parametric interpolation filter for motion compensated prediction" [81]. At the same time, this work is submitted to *IEEE transactions on Circuits and System for Video Technology*, entitled "Parametric interpolation filter for high-definition video coding" [46], and also filed as a US provisional patent application, entitled "Parametric interpolation filter for motion-compensated prediction" [82].

Chapter 7

Real-time De-interlacing for HD Videos

Among the HD video formats (see Table 1.1), 1080i is the most widely used and has been adopted by many countries for HDTV broadcasting, such as Australia, U.S.A., Canada, China, and some European countries. 1080i is directly compatible with CRT-based HDTV sets, which, however, never entered the market in large volume. 1080i is also compatible with 1080p-based HDTV, if de-interlacing is performed immediately before display. In other words, every field in 1080i videos should be converted to a frame in real time.

It is challenging to de-interlace HD videos in real time, especially by using software. The algorithm should be simple, to process such a large amount of data within the limited time, and be flexible as well, to adapt to spatially and temporally local activities and thus achieve good visual quality. The latter requirement inevitably employs implicit or explicit motion and texture detectors, which makes the algorithm more complicated and conflicts with the former requirement. In this chapter, two de-interlacing techniques are proposed to resolve the conflict, such that H.264 coded HD videos can be de-interlaced in real time at a good visual quality.

7.1 Related Work

Most of the existing de-interlacing techniques fulfill only one of the two conflicting requirements: low complexity and good visual quality. Some basic methods are simple enough to work in real time at the expense of visual quality, such as fixed filters. Spatial filters, including line averaging (LA), edge-dependent linear averaging (ELA) [83], and median filter (MF) [84], interpolate the missing pixels by using the neighborhoods in the same field and usually have lowpass characteristics. Vertical-temporal filter (VTF) [85] is supported by both spatially and temporally neighboring pixels, where the temporal

filtering usually has highpass characteristics in order to limit the contribution from the temporally neighboring pixels to high vertical frequencies. Motion compensation (MC) is another basic but more efficient idea for de-interlacing [86], which performs interpolation along the motion trajectory. MC can be very simple by copying the pixels from neighboring fields, if the MVs are given. Otherwise, motion estimation (ME) slows down the de-interlacing process significantly.

When time constraint is not imposed, e.g., in the film studio, more advanced techniques can provide better visual quality. Some of them are the enhancements of the aforementioned basic methods [87; 88; 89; 90; 91]. Byun *et al.* [87] improve ELA by introducing horizontal and vertical edge patterns and modeling them as a weighting function to estimate edge direction. Wang *et al.* [88] assume that the edge direction in local area is stationary, which can be detected by minimizing a cost function. Hwang *et al.* [89] improve MF by applying smoothening prior to the filtering, which makes the MF more robust against noise. In some hybrid methods, the aperture of MF is extended to the temporal domain, and the motion predicted pixels become part of the MF's candidates [90] [91]. Some de-interlacing algorithms focus on improving the accuracy of MC [92; 93; 94; 95]. Braspenning *et al.* [92] propose an ME algorithm, named 3D recursive searching (3DRS). 3DRS calculates spatial and temporal prediction vectors from a 3D neighborhood, and yields coherent vector fields that closely correspond to the true motion of objects. It is improved in [93] by introducing bi-directional ME with variable block sizes. Li *et al.* [94] propose a phase-correction filter, which enables accurate MC between opposite parity fields. Chang *et al.* [95] propose a hybrid approach, comprising edge-based interpolation and local/global ME and MC. More advanced MC methods are based on the generalized sampling theorem (GST) [96], which says any signal limited to a frequency of $0.5f_s$ can be exactly reconstructed from N independent sets of samples, representing the same signal with a sampling frequency f_s/N. The theorem is used to solve the problem of interpolation on a subsampled signal, as first represented by Delogne [97] and Vandendorpe [98]. However, GST de-interlacers are very sensitive to MV errors and the performance degrades significantly as MV errors increase, although they provide excellent results with accurate MVs [99]. Wang *et al.* [100] propose measuring the reliability of MV using the *a posteriori* probability based on the Bayes theorem and the GST algorithm is only applied to where the MVs

are reliable.

Recently, some research work takes both requirements into consideration and tries to resolve the conflict [101; 102; 103]. Van De Ville *et al.* [101] propose reconstructing a frame in an iterative way, based on MC method. Huang *et al.* [102] use the combination of motion detector and extended MF. Both methods in [101] and [102] are still too complicated to achieve real-time implementation for SD videos, let alone HD videos. Chen *et al.* [103] propose a de-interlacer as a trade-off between flexibility and simplicity, which classifies the field dynamically to background or foreground and uses an extended MF. It can de-interlace HD videos in real time, but introduces obvious serrations.

7.2 Real-time De-interlacing for H.264 Coded HD Videos

The proposed de-interlacers are designed as post-processing techniques for decoded interlaced videos, assuming that the values of the syntax elements (SE) in the bitstreams, which have been used in the decoding process, are also accessible to the de-interlacer. SE values, such as MVs and MB type, give many hints of the motions and edges in video sequences, as if they were the outputs of complicated motion or edge detectors. Hence, according to the SE values, de-interlacing methods can be appropriately selected and adapt to spatially and temporally local activities. At the same time, the time for motion and edge detection is significantly reduced, which makes it possible to de-interlacing HD videos in real time.

Simsic *et al.* [104] have proposed a de-interlacer specially for MPEG-2 coded videos, which uses the MVs and MB type to indicate the presence of motion or to give a level of confidence of motion. Considering the trend that H.264/AVC will overtake MPEG-2 in the near future, we propose two de-interlacers for H.264 coded videos, of which the syntax is more complicated and informative. For example, MV indicates the direction and speed of motion, MB partition reveals the smoothness of motion or texture, direction of intra prediction suggests the presence of texture, and transform coefficients imply edges inside a block. Such a wealth of information should be verified and summarized, in order to make sure only reliable SE values are used to select the appropriate de-interlacing algorithms. Hence, accuracy analysis is also introduced to deal with the disparity between the SE values and the real motions and textures, which is mainly caused by the encoding strategies and quantization. Furthermore,

the proposed de-interlacers are designed for parallel implementation, so one can take advantage of the multi-core systems, which are very common for PC and DSP platforms nowadays.

The proposed two de-interlacers provide efficient solutions for the prevalent scenario that H.264 coded interlaced HD video needs to be de-interlaced in real time after decoding, and thus resolve the conflict between high visual quality and low complexity as introduced above. The de-interlacers can be extended for other newly developed standards whose syntax is similar to H.264/AVC, such as AVS [39; 42; 40; 41].

Let f_n denote the n^{th} $(n \geq 0)$ field to be de-interlaced in an interlaced video sequence. The vertical sampling rate of f_n is doubled by zero insertion to form a frame F_n, as shown in (7.1), where (x, y) designates the spatial position.

$$F_n(x,y) = \begin{cases} f_n(x, \frac{y}{2}), & \text{if } (y\%2) = (n\%2) = 0 \\ f_n(x, \frac{y-1}{2}), & \text{if } (y\%2) = (n\%2) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{7.1}$$

In F_n, the inserted zero-valued pixels are to be interpolated. According to the H.264/AVC syntax, f_n is partitioned into blocks; each block is a field area associated with an MB, i.e., 16×16 for field-coded MBs and 16×8 for frame-coded MBs. The k^{th} block in f_n, denoted as M_k, corresponds to an upsampled version P_k in F_n. P_k is called a processing block, in which the zero-valued pixels are to be interpolated using the algorithm determined by M_k's SE values.

In the following sub-sections, two de-interlacers are proposed, which distinguish from each other by the way to use M_k's SE values and the complexity. In the statistics-based de-interlacer [105], the encoder is assumed not to produce erroneous SE values, which means the encoder will not intentionally provide bad SEs and the coded SEs are the best according to the encoder's strategy. With this assumption, the reliability of SEs is measured by the degree of confidence based on the statistical analysis, and then the de-interlacer is designed accordingly. Statistics-based de-interlacer performs very fast, as it uses the SE values without any verification. Nevertheless, the assumption on the encoder imposes the limitation to the application of the statistics-based de-interlacer. In the robust de-interlacer [106], the limitation is eliminated by verifying the reliability of MVs, and the usage of other SEs is improved as well. At the same time,

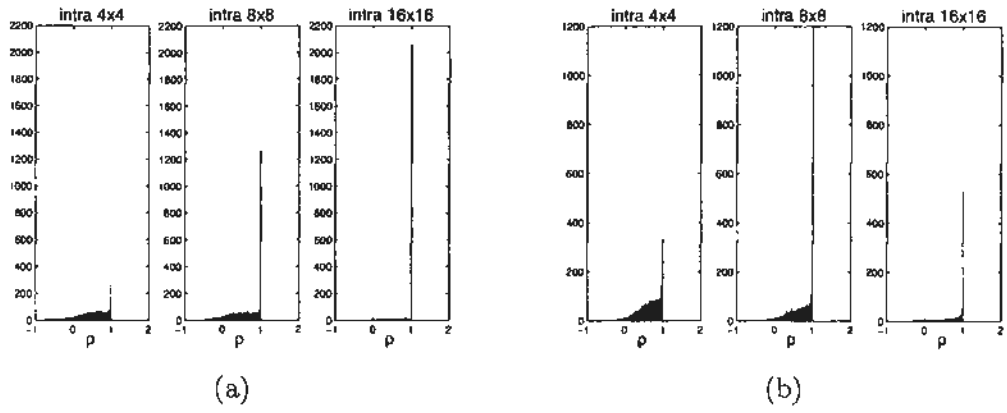


Figure 7.1: *The histograms of ρ of vertically neighboring pixels in intra-coded MBs based on sequences (a) StockholmPan and (b) Shields.*

the computational complexity is increased, which slightly slows down the de-interlacing process. Yet the robust de-interlacer also performs in real time.

7.2.1 Statistics-based De-interlacer

The process of de-interlacing one field consists of two stages. The first stage, named decision stage, decides the interpolation method for each processing block. The output of the decision stage is a 2-D mode map, which records the interpolation methods of all the processing blocks in one field. The second stage, named interpolation stage, applies the interpolation methods indicated in the 2-D mode map to the processing blocks. The two stages are introduced in detail below.

Decision Stage

At first, the spatial correlation in intra-coded MBs and the proportion of different MB partitions are investigated, and then the flow charts for mode decision are designed as described below.

If the MB M_k is intra-coded, the correlation coefficient ρ, defined in (2.19), of the vertically neighboring pixels inside M_k is calculated, where each column is regarded as a realization of a discrete random process and each row is a random variable. As typical examples of the results, Fig. 7.1 shows the histograms of ρ of the vertically neighboring pixels in different intra-coded cases. These two histograms are obtained by investigating 1080i video sequences *StockholmPan* and *Shields*, both of which are coded at bit-rate 15 mbits/s. The texture, as the *a priori* knowledge, does not provide

strong indication which block size is used, because of various encoding strategies. For example, Fig. 7.1 (b) shows smooth areas with ρ approaching 1.0 are more likely to be intra8×8-coded than being intra16×16-coded. However, the coded block size does provide indications for textures, when used as the *a posteriori* knowledge. Intra16×16-coded M_k always indicates very smooth areas with vertical correlation approaching 1.0, whereas the pixels in intra4×4-coded M_k have smaller correlation, representing fine textures. For Intra16×16-coded and Intra4×4-coded M_k, LA, which is simple and efficient for areas without vertical aliasing, and ELA [83], which has outstanding performance for sharp and consistent edges, are applied to the associated processing block P_k, respectively. For intra8×8-coded M_k, there is no strong indication for the textures, which may be complicated or smooth.

When determining the interpolation method for the processing block P_k associated with intra8×8-coded M_k, we first check the condition that all the 8×8 blocks in M_k are vertically smooth. In details, for each 8×8 partition in M_k, denoted as $M_{k,l}$ $(0 \leq l \leq 3)$, let $M_{k,l}(i,j)$, $0 \leq i,j < 8$, be the transform coefficient in the position of the i^{th} row and the j^{th} column. The 8×8 partition $M_{k,l}$ is considered vertically smooth, if the following condition is satisfied.

$$M_{k,l}(i,j) = 0, \qquad \forall i,\ 4 \leq i < 8 \tag{7.2}$$

For any vertically smooth $M_{k,l}$, only the first four bases of the 8×8 ICT in H.264/AVC are involved in the vertical transformation to represent the columns. The spectra of the first four bases are plotted in Fig. 7.2 (a). At the same time, the frequency response of the LA filter is also plotted in Fig. 7.2 (a) by the solid line. As can be seen, the main lobes of the first four bases are all in the passband of the LA filter, and therefore the LA is sufficient for the vertically smooth 8×8 areas. On the other hand, the signals using at least one of the other four higher frequency bases for vertically expansion will be blurred after the LA filtering, as the main lobes of the other four bases are in the LA filter's stopband (see Fig. 7.2 (b)). For the blocks not vertically smooth, the non-linear filter ELA will be applied. In summary, the proposed decision flow for the processing block P_k associated with intra-coded M_k is shown in Fig. 7.3.

While de-interlacing intra-coded fields, the interpolation method switches between LA and ELA, which, however, is at the mercy of aliasing. Aliasing, such as unstable motion and crawling lines, cannot be observed in a still de-interlaced frame, but

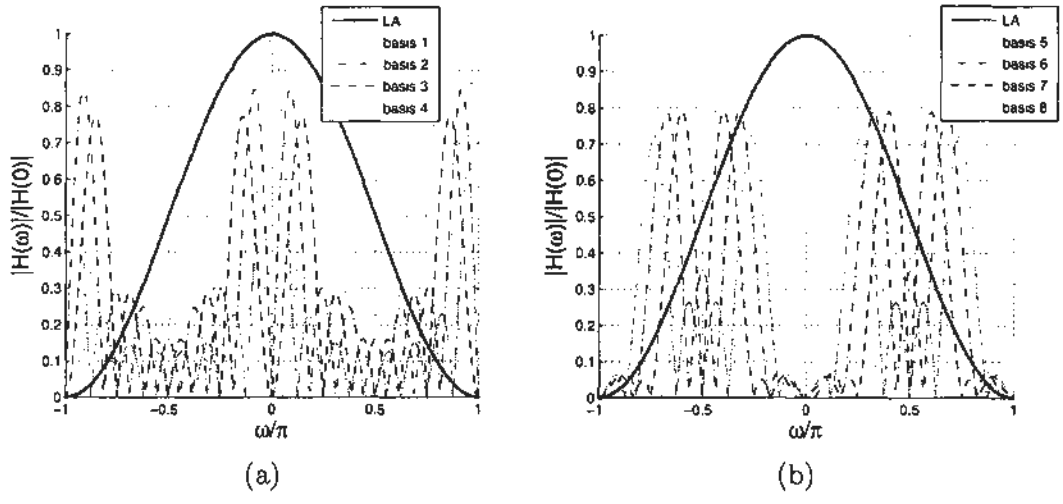


Figure 7.2: *Frequency response of LA and the spectra of the bases of the 8×8 ICT in H.264/AVC. (a) The first four bases. (b) The other four bases.*

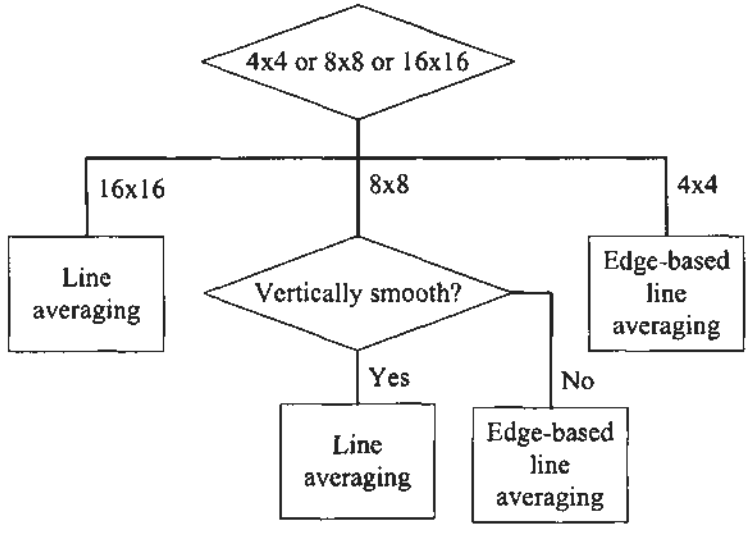


Figure 7.3: *The mode decision for processing blocks associated with intra-coded MBs*

appears when the de-interlaced frames are displayed successively. Therefore, if the video sequence is all-intra-coded, the defect caused by the proposed decision flow (see Fig. 7.3) becomes obvious. Fortunately, for most HD applications, this drawback will not influence the visual quality much, since the interval of I-frames is no smaller than 0.5 second. The drawback can be overcome by MC methods, which are used by the processing blocks associated with inter-coded MBs as described below.

For inter-coded MBs, the proportion of different MB types is investigated. Table 7.1 shows the results of B-frames at the bit-rate about 15 mbits/s, i.e., PSNR around

Test sequence	Skip/Direct	16×16	16×8	8×16	8×8	Intra
Flamingo	26.91	40.89	12.93	7.63	8.63	3.02
Kayak	37.74	29.58	12.61	4.14	4.57	11.35
newMobileCalendar	57.33	24.92	6.01	4.12	6.41	1.22
Parkrun	69.09	19.11	3.07	1.95	6.74	0.04
Shields	68.81	16.49	3.35	2.62	8.57	0.21
StockholmPan	71.22	13.70	3.76	2.85	8.43	0.09

Table 7.1: *The percentages of different MB types in B-frames*

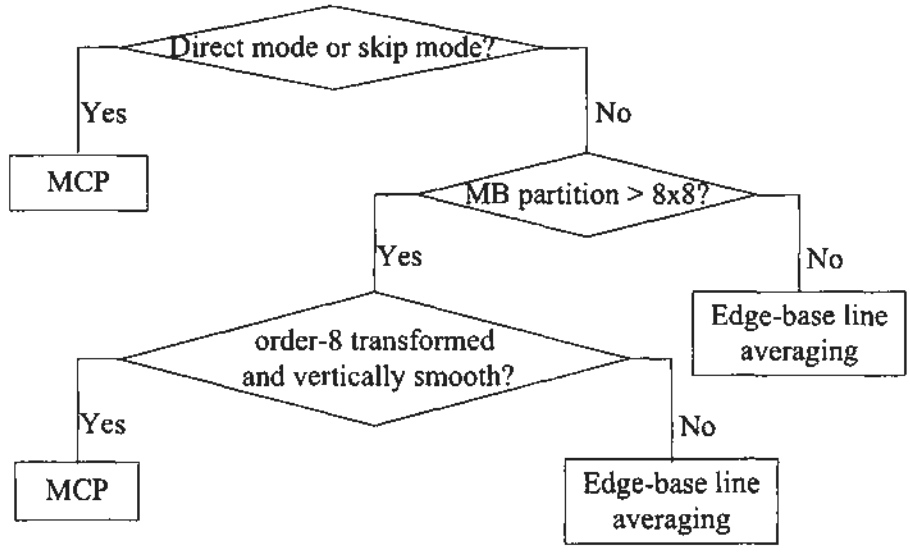


Figure 7.4: *The mode decision for processing blocks associated with inter-coded MBs*

38dB. Obviously, the proportion of *Skip* and *Direct* modes is often larger than 50% for HD videos, except for some extreme cases, e.g., *Flamingo* and *Kayak*. With *Skip* and *Direct* modes, the MVs, derived from those of spatially or temporally neighbouring blocks, imply uniform translation, and therefore MC is a safe method for the associated processing block P_k. Otherwise, the partition of M_k is checked as follows.

If the partition of M_k is not larger than 8×8, which implies rich details inside and the MVs are very likely to be inaccurate, intra-field method, i.e., ELA, is applied. Otherwise, the partition of M_k is larger than 8×8, where the possibilities of smooth and complicated motions and textures all exist. In this case, MC method will be applied, if the residues are transformed by the 8×8 ICT and all vertically smooth. Otherwise, ELA will be applied. In summary, the proposed decision flow for the processing block P_k associated with inter-coded M_k is shown in Fig. 7.4.

Test sequence	Skip/Direct	16×16	16×8	8×16	8×8	Intra
Flamingo	5.03	16.17	11.66	10.04	10.03	47.06
Kayak	10.14	8.18	3.73	2.37	2.03	73.56
newMobileCalendar	9.90	25.90	14.39	10.63	26.10	13.07
Parkrun	11.36	30.82	15.80	11.04	17.76	13.3
Shields	4.15	31.09	16.21	13.21	23.47	12.1
StockholmPan	7.07	22.53	14.46	11.41	24.73	19.92

Table 7.2: *The percentages of different MB types in P-frames*

Besides the mode decision processes illustrated by Fig. 7.3 and Fig. 7.4, a complementary MC (CMC) method is introduced for the frames that are far from their reference frames, e.g., hierarchical B-frames and P-frames in the IBBPBBP sequence structure. Take the IBBPBBP sequence structure as an example. It improves the coding efficiency significantly, but the percentage of *Skip* mode in P-frames is greatly reduced (see Table 7.2), as uniform translation is not likely to continue over such a long interval of at least six fields. Actually, most MBs in the P-frames have accurate matching blocks in the temporally neighboring (in the display order) non-referred frames. The CMC allows the processing blocks in P-frames to find the best matching blocks in the neighboring non-referred frames. The details are presented later in the introduction to the interpolation stage.

In summary, the interpolation method is mainly decided in the decision stage, and can be updated in the interpolation stage by using CMC, which is a type of accuracy re-check. However, the de-interlacer cannot afford many rechecking steps due to the time constraint of real-time HD applications.

Interpolation Stage

In this stage, all the processing blocks in a field are interpolated using the corresponding methods indicated in the 2-D mode map, which is the output of the decision stage. If the de-interlacer is implemented in a multi-core system, the decision stage and the interpolation stage can be processed in parallel so as to maximize the CPU usage. As shown in Fig. 7.5 (a), the decision stage is implemented together with the reconstruction process in order to collect and analyze the SE values conveniently, and therefore should be in the decoding order. Given the 2-D mode map, the interpolation stage is independent of the decision stage (see Fig. 7.5 (b)), and can be performed several fields

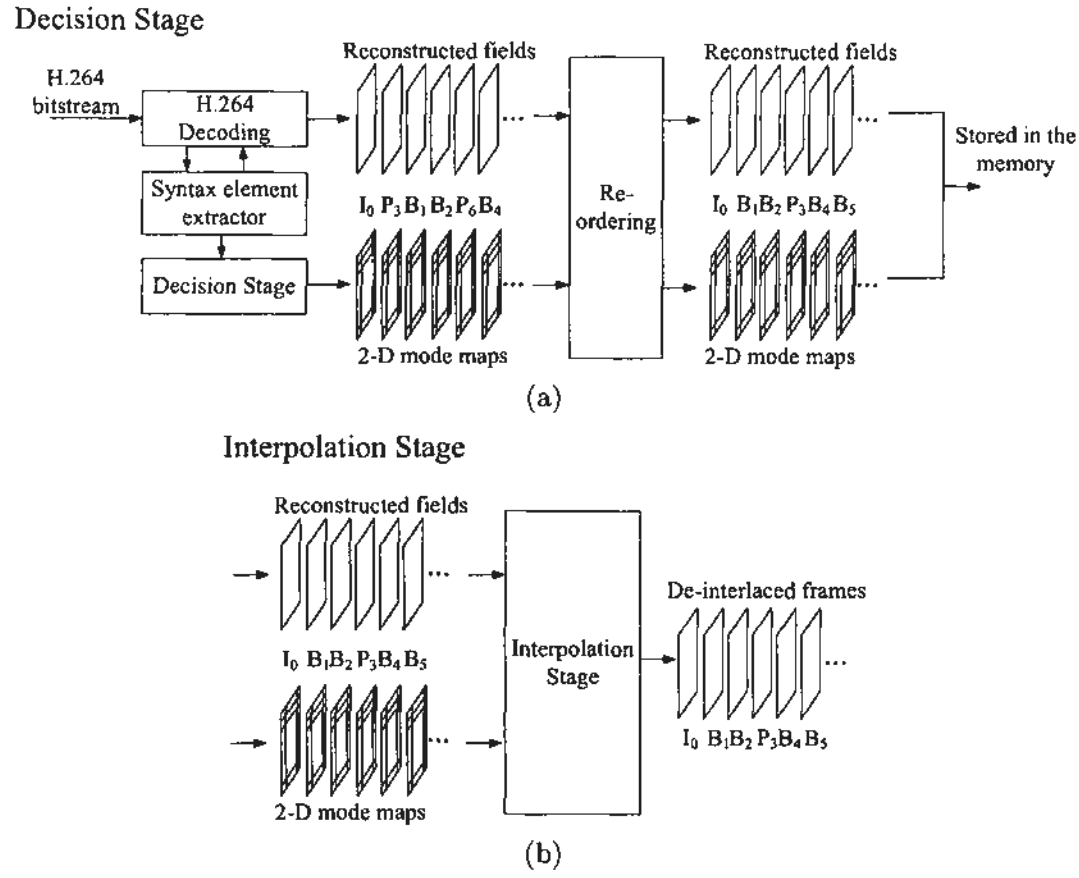


Figure 7.5: *Two stages of the statistics-based de-interlacing process. (a) The decision stage. (b) The interpolation stage.*

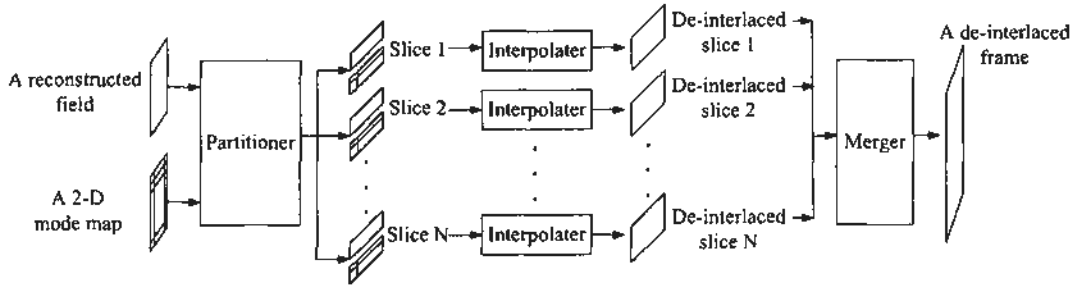


Figure 7.6: *The parallel computation in the interpolation stage*

behind the decision stage, e.g., in the display order. Furthermore, in the interpolation stage, a field can be divided into parts according to the number of CPU cores, and be interpolated in parallel by multi-threading mechanism, as illustrated in Fig. 7.6

If interpolated by the MC method, the processing block P_k has the same partition and direction of prediction as that of the associated MB M_k. However, no matter

how many frames are allowed by the H.264 coded bitstream, only the immediately previous and next fields are used as reference for the MC method, which have the same parity as the current missing field. The MVs pointing to other reference fields are first linearly scaled to the neighboring fields. If the pixels in a neighboring field, to which the MV points, are not on the interlaced sampling grid, temporal backward projection (TBP) [107] is employed to solve the problem. TBP uses the existing pixels for MC instead of using the interpolated sub-position pixels, so it avoids error propagation and is very simple to fulfill the real-time requirement. As the drawback, it assumes correct MVs. However, with the aforementioned mode decision process (see Fig. 7.4), the MC methods are mostly selected, when M_k is in the *Skip* or *Direct* mode, which fulfills the assumption. Further protection is also introduced to improve the robustness against incorrect MVs. The pixels used for MC are passed through a 3-tap MF as shown in (7.3) instead of being used directly,

$$F_n(x,y) = \text{median}(F_n(x,y-1), F_n(x,y+1), r) \tag{7.3}$$

where $F_n(x,y)$ is the pixel to be interpolated, $F_n(x,y-1)$ and $F_n(x,y+1)$ are the upper and bottom existing pixels, and r is the prediction value. In case the MCP is bi-directional, r in is the mean of the forward and backward pixels the MVs point to.

As mentioned above, the SE values sometimes do not reflect the real motion, due to the encoding strategies and the restriction of the standards. Take the common scenario shown in Fig 7.7 as an example. The right P-frame is far from its reference, i.e., the left P-frame. The shaded MB cannot find the perfectly matching block in the reference frame, as the side of the object (the triangle) has been distorted over such a long interval, so it has no choice but to use intra mode. Consequently, the associated processing block selects one of the intra-field interpolation methods based on the flow chart in Fig. 7.3. However, it is noticed that the shaded MB actually contains motion, as it has a perfectly matching block in the previous B-frame that cannot be referenced due to the restriction of the standards. The complementary MC (CMC) method is proposed in this work specially for such a scenario; it allows the intra-field interpolation method that has been determined in the decision stage to be replaced to a more appropriate MC method in the interpolation stage.

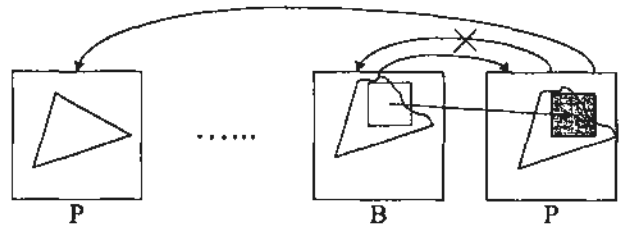


Figure 7.7: *The scenario for CMC*

The CMC method is illustrated in Fig. 7.8, where the horizontal and vertical coordinates represent the time and the line, respectively. The frame at time instance n, denoted as F_n, is called the target CMC frame, where CMC is applied. If an existing pixel in F_n, denoted as $F_n(x, y)$, is used by the MC method for a missing pixel $F_{n-1}(x - dx_1, y - dy_1)$ in the processing block $P_{n-1,k}$ in F_{n-1} with the MV $MV1 = (dx_1, dy_1)$ and the reference and target blocks are perfectly matched, the inverse MV1, $(-dx_1, -dy_1)$, pointing to F_{n-1}, will be used to interpolate the missing pixel $F_n(x, y - 1)$ with the value $F_{n-1}(x - dx_1, y - dy_1 - 1)$. In this work, two blocks are taken as perfectly matched if the transform coefficients of $M_{n-1,k}$ are all zero as indicated by CBP in the bitstream. Similarly, CMC can also perform backward, e.g., with $MV2 = (dx_2, dy_2)$. As the missing field of F_n and the existing fields of F_{n-1} and F_{n+1} have the same parity, i.e., odd or even, the inverse MV always points to the existing pixels in F_{n-1} and F_{n+1}. If the reference pixels from F_{n-1} and F_{n+1} are used to predict the same pixel in F_n, e.g., x_0 in Fig. 7.8, the mean will be used as the interpolated value, as shown in (7.4).

$$F_n(x, y-1) = [F_{n-1}(x - dx_1, y - dy_1 - 1) + F_{n+1}(x - dx_2, y - dy_2 - 1) + 1] >> 1 \quad (7.4)$$

CMC can increase the usage frequency of MC method, which is superior to intra-field method in reducing flickers. As shown in Fig. 7.9, for P-frames, the usage frequency of the MC method, determined by the decision stage, ranges from 10% to 30%. After CMC is applied, the proportion increases to 50% to 60%, which is more consistent with the proportion of real motions in a video sequence.

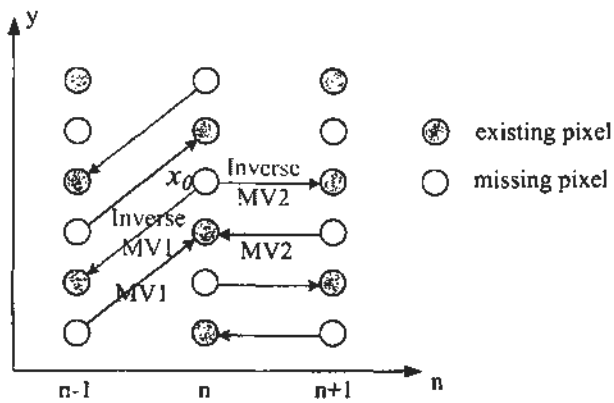


Figure 7.8: *The complementary MC method*

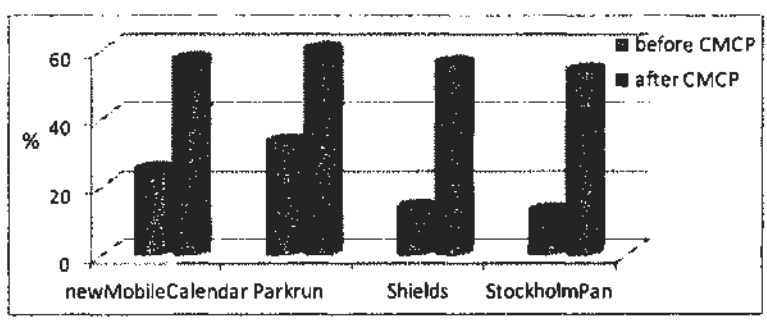


Figure 7.9: *The percentage of pixels interpolated by the MC method before and after CMC*

7.2.2 Robust De-interlacer

De-interlacing algorithm for intra16×16

Intra16×16-coded regions are smoother than other regions, which has been verified in Section 7.2.1. Therefore, for any P_k whose associated M_k is intra16×16-coded, a simple lowpass filter given in (7.5) is applied, of which the frequency response is shown in Fig. 7.10, and the filtering process is given in (7.6).

$$f_L = [-1 \ 5 \ 5 \ -1]/8 \tag{7.5}$$

$$\begin{aligned} F_n(x,y) = (&-F_n(x,y-3) + 5 \times F_n(x,y-1) \\ &+5 \times F_n(x,y+1) - F_n(x,y+3))/8 \end{aligned} \tag{7.6}$$

Compared with the LA filter, which has been used in the statistics-based de-interlacer for the intra16×16-coded M_k, the employed lowpass filter is more complex, but has narrower transition band so that low frequency components are better preserved.

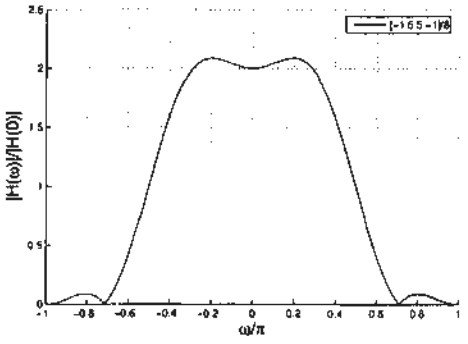


Figure 7.10: *The frequency response of the lowpass filter f_L in (7.5)*

De-interlacing algorithm for intra8×8 and intra4×4

Intra8×8- and intra4×4-coded MBs are where the edges occur. In H.264/AVC, nine intra prediction modes, representing nine prediction directions, are employed for intra8×8 and intra4×4 coding, as shown in Fig. 7.11 (a). MB coded by a certain intra mode may contain edges along the indicated direction. As shown in Fig. 7.11 (b), missing pixel $F_n(x, y)$, the central square labeled "d", is interpolated along the direction consistent with the intra prediction direction for $F_n(x, y-1)$.

At first, the initial value of $F_n(x, y)$, denoted as $\tilde{F}_n(x, y)$, is obtained by applying the lowpass filter as in (7.6). Then, if the intra prediction for $F_n(x, y-1)$ is not "horizontal", i.e., Mode 1, the MF given in (7.7) is applied,

$$F_n(x, y) = \text{median}(\tilde{F}_n(x, y), c_1, c_2) \tag{7.7}$$

where c_1 and c_2 are the two neighboring pixels along the appropriate direction. For example, if the intra prediction for $F_n(x, y-1)$ is Mode 4, c_1 and c_2 refer to a and f, respectively. In Fig. 7.11 (b), the pixels located at non-integer positions, such as a and aa, are interpolated using bilinear filter supported by the nearest two integer pixels. MF in (7.7) implicitly detects whether $F_n(x, y)$ really belongs to an edge and interpolates it accordingly. If edge occurs, $F_n(x, y)$ is more highly correlated with c_1 and c_2 along the edge direction than with $\tilde{F}_n(x, y)$, resulting in the interpolated value equal to c_1 or c_2. Otherwise, the region is smooth, so $F_n(x, y)$ is more likely to have the initial value $\tilde{F}_n(x, y)$, the output of a lowpass filter. Such implicitness is efficient for the scenario that edges are indicated by the intra prediction modes, but conversely, not all the intra prediction modes imply the presence of edges because of various encoding

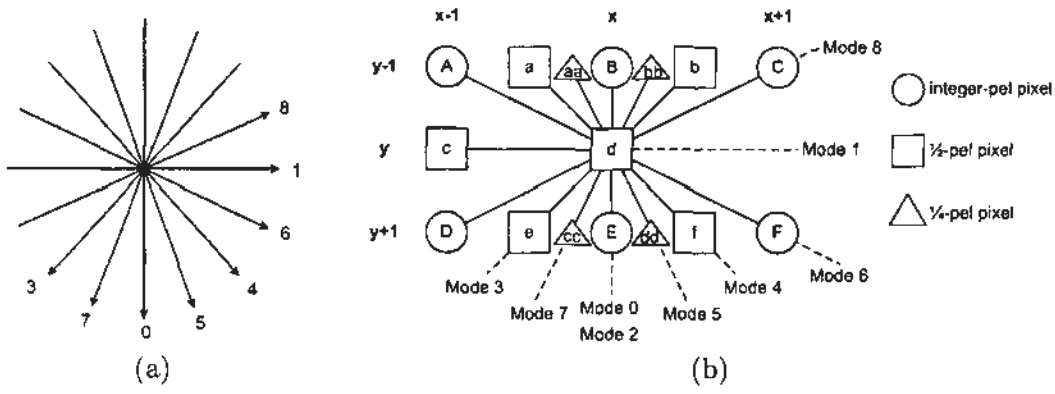


Figure 7.11: *4×4 and 8×8 intra prediction modes. (a) The prediction directions and (b) the corresponding interpolation directions for de-interlacing*

strategies. Similarly, if the intra prediction is "horizontal", MF in (7.8) is applied.

$$F_n(x,y) = median(F_n(x-1,y), F_n(x,y-1), F_n(x,y+1)) \tag{7.8}$$

De-interlacing algorithm for INTER

The MVs of M_k, decoded from the bitstream, may point to any picture in the reference picture list and the resolution is up to 1/4-pixel in terms of either field or frame sampling grid. These MVs cannot be directly used for the MC method for de-interlacing, because the proposed de-interlacer only uses integer-pixel MC in terms of the frame sampling grid and the reference picture is limited to the direct previous de-interlaced frame, in order to reduce the complexity. Therefore, if not equal to zero, a decoded MV in M_k, denoted as (dx^M, dy^M) should be converted to a new MV (dx^P, dy^P) that can be used for de-interlacing P_k.

Firstly, if dy^M represents the vertical displacement in a field sampling grid, dy^M is mapped to the equivalent $\widetilde{dy}^M$ in a frame sampling grid as shown in (7.9),

$$\widetilde{dy}^M = \begin{cases} dy^M \times 2, & \text{if } f_n \text{ and } f_M \text{ have the same parity} \\ (dy^M - 2) \times 2, & \text{if } f_n \text{ is bottom and } f_M \text{ is top} \\ (dy^M + 2) \times 2, & \text{if } f_n \text{ is top and } f_M \text{ is bottom} \end{cases} \tag{7.9}$$

where f_M with the field number M is the field that dy^M points to. Otherwise, if dy^M represents the vertical displacement in a frame sampling grid, $\widetilde{dy}^M$ equals dy^M.

Then, $(dx^M, \widetilde{dy}^M)$ is linearly scaled and approximated to integer displacements,

resulting a new MV (dx^P, dy^P) used for de-interlacing P_k.

$$\begin{aligned} dx^P &= \text{round}(dx^M/(n-M)/4) \\ dy^P &= \text{round}(\widetilde{dy}^M/(n-M)/4) \end{aligned} \tag{7.10}$$

Note that the divisions in (7.10) are usually implemented by multiplications and bit shifts.

After the MV for the MC method is obtained, its reliability will be verified. Although H.264/AVC allows MBs to have different number of MC blocks, ranging from 1 to 16, P_k is alway partitioned into 16 blocks, denoted as $B^P_{k,i}$ ($0 \leq i \leq 15$); each has a forward MV, denoted as $(dx^P_{k,i}, dy^P_{k,i})$.

The following verification procedure is for P_k with the size 16×32, which means the associated MB M_k is field-coded and the size of M_k is 16×16. Each $B^P_{k,i}$ is a unit for MC. To verify the accuracy of its MV, two factors are taken into consideration. One is the MC error measured by SAD (see (7.11)).

$$SAD^P_{k,i} = \sum_{(x,y)\in B^P_{k,i}} |F_n(x,y) - F_{n-1}(x + dx^P_{k,i}, y + dy^P_{k,i})| \tag{7.11}$$

The other factor is the MC error introduced by the corresponding MV decoded from the H.264/AVC bitstream, denoted as $(dx^M_{k,i}, dy^M_{k,i})$; this factor is measured by the summation of the absolute values of the dequantized transform coefficients, as shown in (7.12),

$$E^M_{k,i} = \max(LB, \sum_{l=0}^{3}\sum_{j=0}^{3} |C^M_{k,i}(l,j)|) \tag{7.12}$$

where $C^M_{k,i}$, corresponding to $B^P_{k,i}$, is a 4×4 block in the transform domain comprising dequantized coefficients and LB is set as the minimum value of $E^M_{k,i}$. In case the value of $\sum_{l=0}^{3}\sum_{j=0}^{3} |C^M_{k,i}(l,j)|$ equals zero, $(dx^M_{k,i}, dy^M_{k,i})$ cannot be considered accurate. Rather, the actual accuracy is greatly influenced by the quantization stepsize and deadzone. Therefore, the minimum value of $E^M_{k,i}$, LB, should be non-zero. In this work, LB is set to 30.

Fig. 7.12 shows the flow chart for verifying the MV for $B^P_{k,i}$, given $SAD^P_{k,i}$ and $E^M_{k,i}$. Firstly, $SAD^P_{k,i}$ is compared with $E^M_{k,i}$. Typically, $SAD^P_{k,i}$ is larger than $E^M_{k,i}$, because $(dx^P_{k,i}, dy^P_{k,i})$ is an approximation of $(dx^M_{k,i}, dy^M_{k,i})$. If $SAD^P_{k,i}$ is larger than $1.5E^M_{k,i}$,

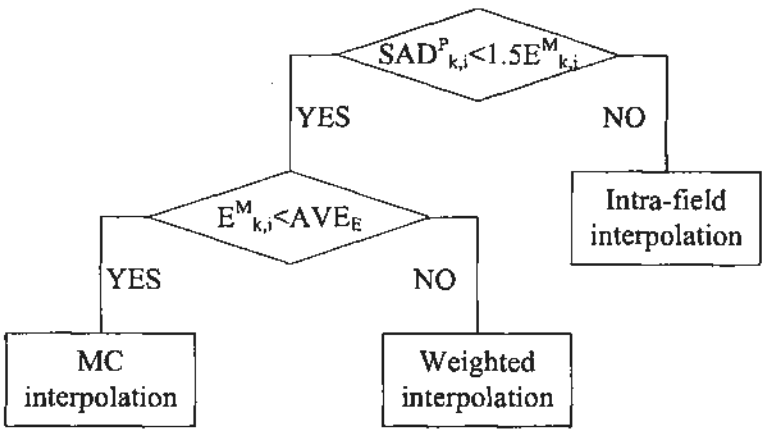


Figure 7.12: *The flow chart for MV verification*

which means the scaling and approximation as shown in (7.9) and (7.10) introduce significantly larger MC error compared with that introduced by $(dx^M_{k,i}, dy^M_{k,i})$, then intra-field interpolation as shown in (7.6) is used. Otherwise, $SAD^P_{k,i}$ is smaller than or equal to $1.5E^M_{k,i}$, which means the scaling and approximation do not enlarge the MC error severely, so the MC method with the MV $(dx^P_{k,i}, dy^P_{k,i})$ can be a possible de-interlacing algorithm. Then, $E^M_{k,i}$ is examined. If $E^M_{k,i}$ is smaller than its average, denoted as AVE_E, $(dx^P_{k,i}, dy^P_{k,i})$ is considered reliable and MC de-interlacing is applied, as in (7.13).

$$F_n(x,y) = F_{n-1}(x + dx^P_{k,i}, y + dy^P_{k,i}) \tag{7.13}$$

Otherwise, the missing pixel is calculated as the weighting average of inter- and intra-field interpolation results, as in (7.14),

$$F_n(x,y) = \alpha \tilde{F}_n(x,y) + (1-\alpha) F_{n-1}(x + dx^P_{k,i}, y + dy^P_{k,i}) \tag{7.14}$$

where $\tilde{F}_n(x,y)$ is calculated by (7.6) and the weighting factor (1-α) represents a level of confidence of using the MC de-interlacing, calculated as below:

$$\alpha = \min(1, \frac{E^M_{k,i} - AVE_E}{AVE_E}). \tag{7.15}$$

Parallel Implementation

If used in a multi-core system, the de-interlacer can be implemented in two stages: *extraction stage* and *interpolation stage*, in order to favor the parallel computing. These two stages are similar to the aforementioned decision stage and the interpolation stage in

the statistics-based de-interlacer. The difference is that here the de-interlacing method for each processing block is not all determined in the extraction stage, but is left to be determined after the MV verification step in the interpolation stage. Therefore, no mode map is output in the extraction stage. In the extraction stage, the SE values of each MB is preliminarily processed, converted to a compact representation of the motions and edges, and stored for each P_k.

- If M_k is intra16×16-coded, P_k is marked as "smooth".
- If M_k is intra8×8- or intra4×4-coded, the intra prediction modes are stored for each partition of P_k.
- If M_k is inter-coded, $(dx^P_{k,i}, dy^P_{k,i})$ and $E^M_{k,i}$ for each $B^P_{k,i}$ in P_k are calculated and stored.
- AVE_E is stored once per field.

In the interpolation stage, a field is de-interlaced in a block-by-block manner and individual processing block can be de-interlaced independently of others, given the motion or edge information, i.e., the output of the extraction stage.

The extraction stage can be performed together with the reconstruction process in order to collect and process the SE values conveniently, so it should be in the decoding order. The interpolation stage can be performed several fields behind the extraction stage, e.g., in the display order. In the interpolation stage, a field can be further divided into parts according to the number of cores, and all the parts are interpolated in parallel by multi-threading mechanism to maximize the CPU usage.

7.3 Experimental Results

7.3.1 Subjective Quality

The proposed two de-interlacers outperform many other commonly used real-time ones, such as the basic methods, like ELA [83] and VTF [85], and the method by Chen *et al.* [103]. It also provides comparable visual quality, compared with MC method, such as the one proposed in [93], which is slow and often used without the real-time constraint imposed. Figs. 7.13, 7.14, and 7.15 shows the visual qualities of the de-interlaced

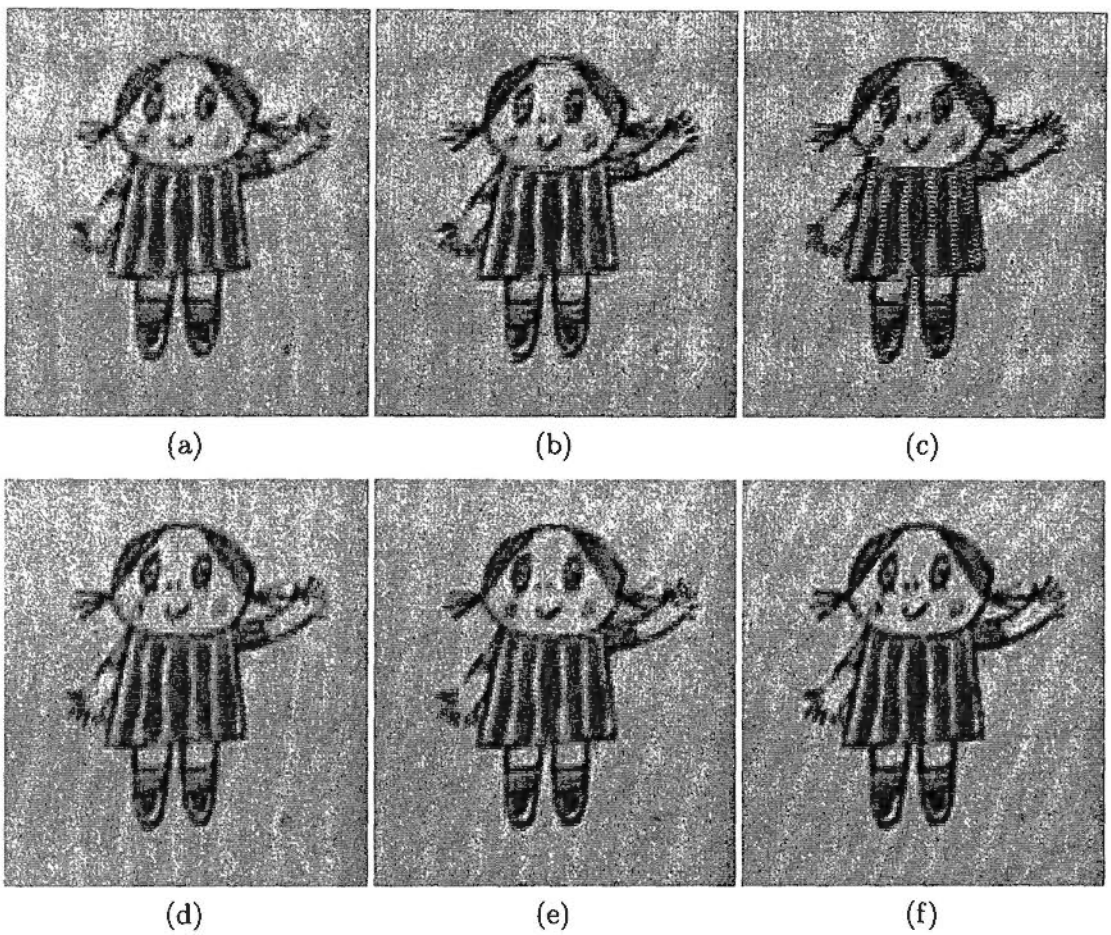


Figure 7.13: *Comparing the visual quality of the de-interlaced sequence new_MobileCalendar coded by H.264/AVC at 14 mbits/s. (a) is provided by ELA [83], (b) is provided by VTF [85], (c) is provided by Chen's method [103], (d) is provided by Li's method [93], (e) is provided by the proposed statistic-based de-interlacer, and (f) is provided by the proposed robust de-interlacer.*

frames provided by ELA, VTF, Chen, Li, and the proposed two de-interlacers. The pictures are cropped from the 1080i de-interlaced sequences, *new_MobileCalendar*, *Shields*, *Stockholmpan*, respectively, which are reconstructed from 14 mbits/s H.264/AVC bitstreams.

As can be seen, serrations are easily observed in the pictures de-interlaced by ELA, VTF, and Chen's methods. For ELA, the serrations are caused by the noise, which ELA is sensitive to. Observing the hair of the girl, the collar, and the edge of the roof in Figs. 7.13(a), 7.14(a), and 7.15(a), respectively, one can find that the interpolation are mistakenly performed along the 135° direction instead of the correct 45° direction. For VTF and Chen's method, the serrations are caused by inaccurate temporal compensation; the artifacts are more severely for the latter, where inaccurate pixels in

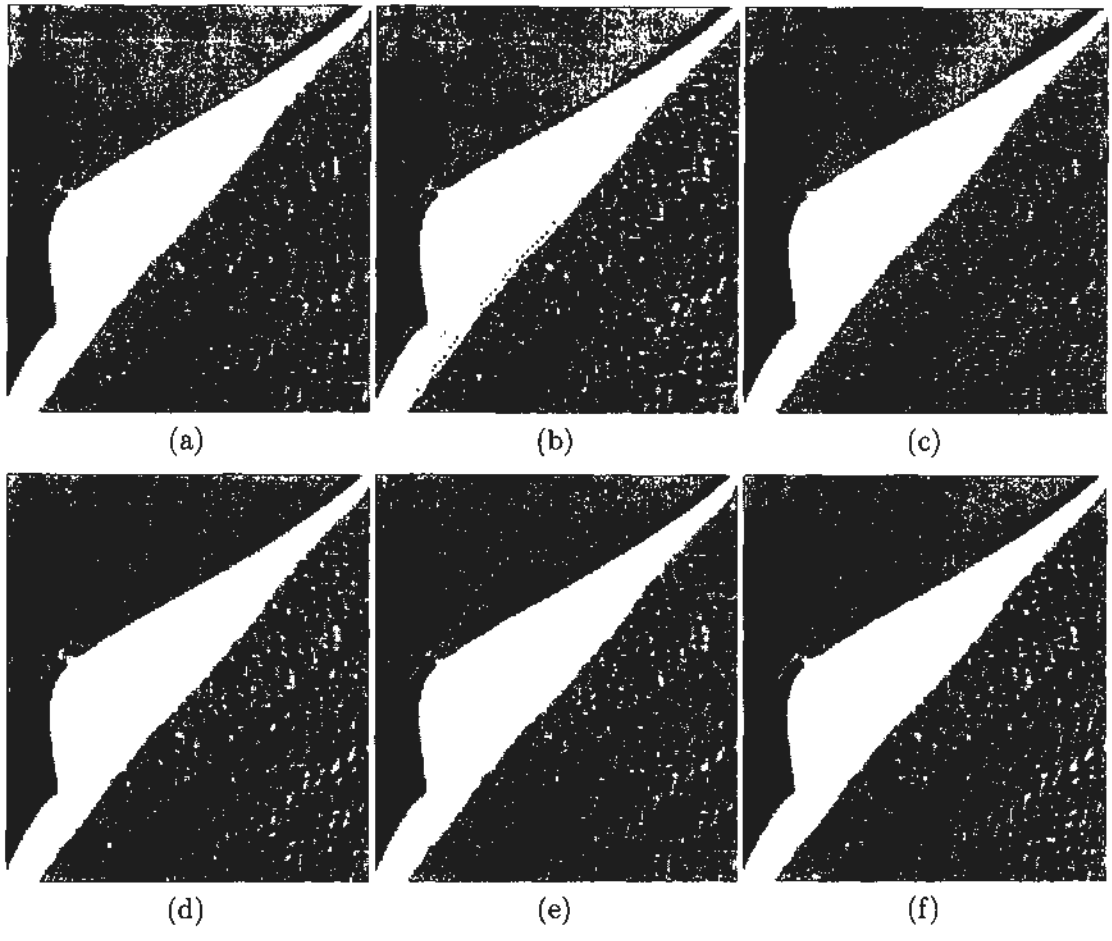


Figure 7.14: *Comparing the visual quality of the de-interlaced sequence Shields coded by H.264/AVC at 14 mbits/s. (a) is provided by ELA [83], (b) is provided by VTF [85], (c) is provided by Chen's method [103], (d) is provided by Li's method [93], (e) is provided by the proposed statistic-based de-interlacer, and (f) is provided by the proposed robust de-interlacer.*

neighboring fields are directly copied. In the frames de-interlaced by Li's method and the proposed de-interlacers, serrations are almost invisible, owing to the improved MC. However, in Figs. 7.13(d), slight serrations (see the left hand and the vertical edges in the clothes) are still possible to be observed by using the statistics-based de-interlacer, because the MVs used for the MC method are not verified and no accurate MC is guaranteed.

Besides serration, the other common artifact in de-interlaced videos is unstable motion, a.k.a. flickering, caused by different aliasing components in opposite parity fields. Unstable motion can only be observed, when the de-interlaced frames are displayed successively. The proposed two de-interlacers provide smooth de-interlaced videos by

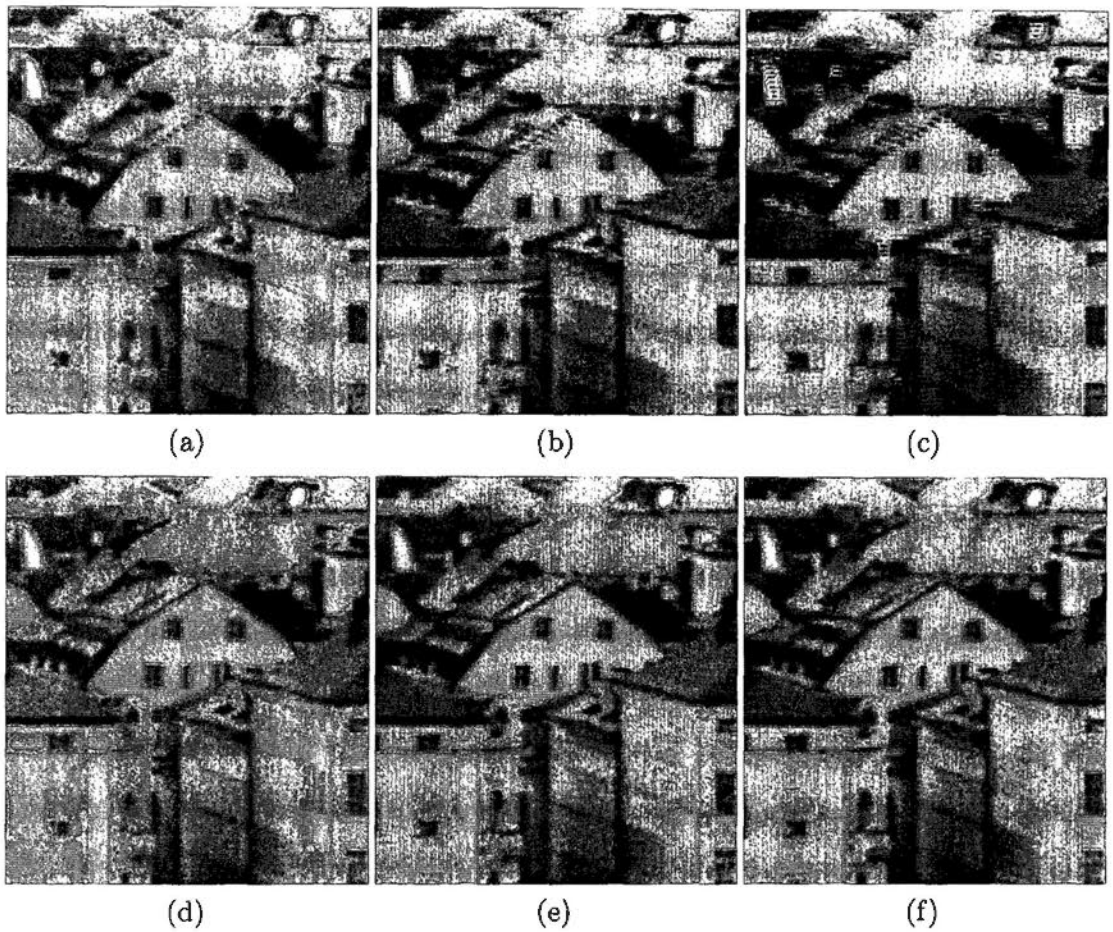


Figure 7.15: *Comparing the visual quality of the de-interlaced sequence StockholmPan coded by H.264/AVC at 14 mbits/s. (a) is provided by ELA [83], (b) is provided by VTF [85], (c) is provided by Chen's method [103], (d) is provided by Li's method [93], (e) is provided by the proposed statistic-based de-interlacer, and (f) is provided by the proposed robust de-interlacer.*

using the MC method, whereas the intra-field method ELA introduces obvious flickering. More HD sequences de-interlaced from 14 mbits/s H.264/AVC bitstreams, using these six algorithms, are available at http://ivp.ee.cuhk.edu.hk/~deinterlacing.

7.3.2 Objective Quality

It is difficult to evaluate the objective quality of the de-interlaced videos, because the captured scenes are in the field format and the original frames, i.e., the ground truth of the de-interlaced frames, are not available. In this sub-section, HD interlaced videos are generated from 1080p videos by alternatingly dropping the top and bottom fields in successive frames, so that the PSNR of the de-interlaced videos can be calculated by using the original 1080p videos as the ground truth. Before de-interlacing is performed,

HD sequences	ELA [83]	VTF [85]	Chen's [103]	Li's [93]	Proposed 1	Proposed 2
Bluesky	16.99	32.57	27.67	37.32	36.38	38.01
PedestrianArea	20.99	38.81	34.48	38.85	39.40	39.72
RushHour	24.49	40.04	36.53	40.24	41.13	40.88
Sunflower	21.12	39.89	34.17	41.89	42.16	41.32
ToysCalendar	20.48	33.51	31.25	36.52	35.60	36.21
Tractor	20.34	33.28	29.35	34.41	34.37	35.15
WalkingCouple	21.33	31.60	30.41	34.45	34.13	34.54

Table 7.3: *The objective quality of de-interlaced frames measured by PSNR in dB*

Test sequence	IPP dec.	Statistics-based De-interlacer		Robust De-interlacer	
		IPP dec.+ De-interlacing	Execution time	IPP dec.+ de-interlacing	Execution time
Flamingo	4.354	4.478	0.124	4.684	0.330
Kayak	4.208	4.423	0.215	4.711	0.503
Mountain	4.426	4.639	0.213	4.754	0.328
Parkrun	4.335	4.375	0.040	4.768	0.433
Shields	4.264	4.697	0.433	4.948	0.684
Stockholmpan	4.687	4.929	0.242	4.843	0.156
Average	4.379	4.590	0.211	4.785	0.406

Table 7.4: *The time (second) for de-interlacing 116 fields*

the artificial interlaced videos are first coded by H.264/AVC. Then the decompressed videos are de-interlaced.

The experimental results are shown in Table 7.3. The objective qualities in PSNR provided by six de-interlacers are compared, where "Proposed 1" and "Proposed 2" refer to the statistics-based de-interlacer and robust de-interlacer, respectively. The interlaced videos for de-interlacing are reconstructed from H.264 coded bitstreams at the bit-rate 14 mbits/s. As shown in Table 7.3, for all tested sequences, the proposed two de-interlacers significantly outperforms the commonly used real-time de-interlacers [83; 85; 103] and also achieves slightly better PSNR, compared with the MC method [93], which cannot work in real time. Considering that PSNR is not precisely correlated with the subjective quality, but only reflects a rough estimate of it, one can conclude that the quality provided by the proposed two de-interlacers is comparable to what is provided by those non-real-time de-interlacers.

7.3.3 Computational Time

The PC used to evaluate the computational time has two 3.0 GHz Dual Core Intel Xeon 5160 CPUs and 1 GB RAM. The proposed de-interlacer has been implemented using multi-threading to maximize the CPU usage of the four-core system, and integrated into the Integrated Performance Primitives (IPP) decoder version 5.3, which decodes the H.264/AVC bitstreams by multi-threading as well. Two Windows API functions QueryPerformanceFrequency and QueryPerformanceCounter are used for timing, which measure the execution time to nanosecond accuracy. Each sequence is decoded five times, and the reported execution time is the average of the middle three records.

Table 7.4 compares the average execution time of decoding 116 fields by the IPP decoder and decoding with de-interlacing the 116 fields by the IPP decoder with the proposed de-interlacers integrated in it. For statistics-based de-interlacer, the average execution time is only 0.211s for 116 fields, i.e., 2×10^{-3}s for de-interlacing one field, which means the de-interlacer occupies only 10% of the decoding time, if the speed for decoding 1080*i* videos is 50 fps, i.e., 2×10^{-2}s for decoding one field. Therefore, real-time de-interlacing is achieved for HD videos. However, for robust de-interlacer, the average execution time is almost doubled compared with the statisitics-based de-interlacer, because of the additional MV verification steps and more complicated FIR interpolation filter. Given the decoding speed 50 fps, simultaneously performing decoding and de-interlacing using the robust de-interlacer slows down the decoding about 20% and the speed is reduced to 40 fps.

7.4 Summary

In this chapter, two real-time de-interlacers, statistics-based de-interlacer and robust de-interlacer, are proposed for H.264/AVC coded HD videos. The two de-interlacers distinguish from each other by the way the SE values are used and the complexity. Assuming the encoder does not produce erroneous SE values, the statistics-based de-interlacer measures the reliability of SEs by the degree of confidence based on the statistical analysis, and designs the de-interlacer algorithm accordingly. Statistics-based de-interlacer performs very fast, as it uses the SE values without any verification. Nevertheless, the assumption on the encoder imposes the limitation to the application. In the robust de-interlacer, the limitation is eliminated by verifying the reliability of MVs,

and the usage of other SEs is improved as well. At the same time, the computational complexity is increased, which slightly slows down the de-interlacing process. Yet the robust de-interlacer also performs in real time. The visual qualities provided by both proposed de-interlacers are better than those by commonly used real-time de-interlacers and comparable to those by advanced non-real-time ones.

The statistics-based de-interlacer was presented at *ICME2008* as an oral paper, entitled "An adaptive and parallel scheme for HD video de-interlacing" [105], and has also been filed as a US non-provisional patent application, entitled "Method and apparatus of de-interlacing video" [108]. The robust de-interlacer has been submitted to *IEEE Transactions on Circuits and System for Video Technology* as a transaction brief paper, entitled "Real-time de-interlacing for H.264 coded HD videos" [106].

Chapter 8

Conclusions

Digital video representation has been experiencing a tremendous growth in the past two decades. The typical video formats, such as CIF, QCIF, and SD, are gradually superseded by HD videos. Today, HD video formats have been adopted for various applications, including broadcast, high-density storage media, filmmaking, video gaming, and surveillance, all over the world, but how to design techniques especially efficient for HD video coding based on their unique statistical characterization is still an open research issue. In this thesis, the unique statistical characterization of HD videos and the corresponding coding and processing techniques have been presented and discussed, which are summarized in Section 8.1. In addition, directions for future research are also pointed out in Section 8.2.

8.1 Contributions of the Thesis

First of all, the unique statistical properties of HD videos are studied quantitatively in Chapter 4. It is found that HD videos have higher spatial correlation than other low resolution videos, when considered as a WSS random field. For residual HD frames, the parameter ρ, representing the correlation coefficient of two adjacent pixels in the horizontal or vertical direction, is substantially larger [45]. It is also pointed out the spatial correlation in residual frames depends on the accuracy of MCP and varies in local areas, and therefore a separable 2-D AR(1) field is not well-suited, no matter what the resolution is. The energy distribution of HD videos is also studied and compared with that of other videos by PSD. The results show that HD videos distinguish from other ones by higher low frequency energy and special PSD, mainly distributed along the vertical and horizontal directions.

Secondly, based on the fact that the prediction errors of HD videos have higher correlation, two new types of 2-D order-16 integer transforms, MICT and NICT, are studied and developed for HD video coding [109; 64; 45] in Chapter 5. The former modifies the structure of the DCT matrix by the principle of dyadic symmetry and is inherently orthogonal, no matter what the values of the matrix elements are. The latter is non-orthogonal in order to use matrix elements with small magnitudes and maintain the structure of the DCT matrix as well, e.g., dyadic symmetries and relative magnitudes. It preserves all the advantages of ICT at the expense of non-orthogonality, by which the transform error introduced is proven to be negligible. The specific NICT proposed in this work achieves the balance among the performance, the complexity, and the transform error. Both types allow selecting matrix elements more freely and can provide comparable performance with that of DCT. At the same time, many efforts have been devoted to further reducing the complexity of the 2D order-16 transforms and specially for MICT, a fast algorithm is developed and extended to a universal approach. Experimental results show that the proposed 2D order-16 integer transforms provide significant performance improvement for both AVS EP and H.264 HP, which means they can be efficient coding tools especially for HD video coding.

Thirdly, based on the fact that high frequency energy of HD video source is mainly distributed along the vertical and horizontal directions, PIF is proposed for MCP [81] in Chapter 6, which is an improvement of existing AIF techniques. Before proposing PIF, the effect of trade-off between the coefficient precision of AIF and size of side information is studied and it is pointed out that the conflict of these two aspects is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually. The proposed PIF represents filters by five parameters instead of individual coefficients and approximates the optimal filter by tuning the parameters, thus solving the conflict of the accuracy of coefficients and the size of side information. At the same time, PIF tracks the non-stationary statistics of video signals as the existing AIF techniques do. The experimental results show that PIF significantly outperforms the existing AIF techniques and approaches the efficiency of the optimal filter.

In the near future, video materials with even higher definitions, such as UHD (see Table 1.1), will be captured and distributed, but their bit-rates produced by the latest

coding technology will go up faster than the increased capacity of the wireless or wired network infrastructure [8]. Therefore, high-performance video coding techniques specially for UHD videos are required. The contributions on coding technology in this work are quite instructive to indicate the trend of coding strategy with increasing definition.

Finally, interlaced HD videos are studied in Chapter 7. Real-time de-interlacers are proposed specially for H.264/AVC coded HD videos [105; 108; 106]. They adapt to local activities, according to the SE values, and thus resolve the conflict between high visual quality and low complexity. Accuracy analysis is also introduced to deal with the disparity between the SE values and the real motions and textures. To satisfy different delay constraints, two real-time de-interlacers, statistical-based de-interlacer and robust de-interlacer, are proposed, which distinguish from each other by the way to use SE values and the complexity. Assuming the encoder does not produce erroneous SE values, the statistics-based de-interlacer measures the reliability of SEs by the degree of confidence based on the statistical analysis, and designs the de-interlacer algorithm accordingly without any verification. Statistics-based de-interlacer performs very fast, but the assumption on the encoder imposes the limitation to its applications. In the robust de-interlacer, the limitation is eliminated by verifying the reliability of MVs, and the usage of other SEs is improved as well. Although the computational complexity is increased, which slightly slows down the de-interlacing process, the robust de-interlacer also performs in real time. The visual qualities provided by both proposed de-interlacers are better than those by commonly used real-time de-interlacers and comparable to those by advanced non-real-time ones. The proposed two de-interlacers provide efficient solutions for the prevalent scenario that H.264 coded interlaced HD video needs de-interlacing in real time after decoding. The de-interlacers can be extended for other newly developed standards whose syntax is similar to H.264, such as AVS [39; 42; 40; 41].

8.2 Future Research Directions

Directional Transform

According to the study in Chapter 5, for residual frames, a separable 2-D AR(1) field is not well-suited, since the correlation coefficient of two adjacent pixels, denoted as ρ, varies in different areas and the average ranges from 0.1 to 0.9, depending on the

spatial resolution. The reason is that edges become dominant in a residual frame. In a frame to be coded, the energy in smooth areas is greatly reduced by MCP, but in edge areas, where the prediction is sensitive to occlusion and geometric distortion, residual energy along the edge direction is still strong. When the edges are neither horizontal nor vertical, conventional transforms including DCT and its simplified variants result in large-magnitude high frequency coefficients that not only need more bits to code, but also introduce large quantization noise at low bit-rates. This phenomenon will still exist with the evolution of MCP techniques, because the reduction of prediction error energy in smooth area is always more significant than that in edge area.

Directional transform for the residue of intra and inter prediction can be a useful coding tool to compact the energy along the edge direction into a few coefficients. Actually, it has been a research topic in image coding for many years, and will have a more significant impact on video coding, where efficiently dealing with edges becomes more critical according to the above rationale. Apart from the high efficiency in dealing with edges, a successful directional transform should also have acceptable complexity. The related processing techniques, including quantization, entropy coding, and transform mode decision, also need optimizing to maximize the benefit of directional transform.

Block-based PIF

All the existing adaptive interpolation techniques, including the proposed PIF, adjust the filter coefficients once per frame, which means they can track the temporal non-stationary statistics of video sources at the frame level. Nevertheless, the spatial statistics inside a frame is ignored, which also varies in different regions, such as smooth and edge areas. Therefore, designing appropriate interpolation filters for different areas can further reduce the prediction error and improve the R-D performance that has been gained by the frame-based AIF techniques. To incorporate such an idea into the framework of block-based hybrid coding, it is quite natural to adapt the interpolation filters to the MB or block level. With block-based adaptive interpolation, more than one set of interpolation filters for different sub-positions are needed to be developed and coded for each frame, which generates too large side information to be carried in the bitstream, if PIF is not used. PIF, which produces very small overhead, is the only interpolation technique that can adapt to the MB or block level.

Implementing block-based PIF requires solving some theoretical and practical problems. First, MBs in a frame are classified into several categories, such as smooth, texture, and edge; the autocorrelations and cross-correlations are calculated and subsequently used to derive the optimal interpolation filters for each category (see (6.2)), respectively. Second, the appropriate PIF for each MB is determined by the criterion of minimum R-D cost (see (3.14)). Using the PIF for the category that the MB belongs to is not always an optimal choice, although reasonable to some extent. Last but not least, the signal indicating which PIF is selected is coded for each MB, where context-based coding should be designed, as the selection of PIF is highly correlated with those in the neighboring MBs.

At present, MPEG and VCEG, are seeking the evidence regarding the possibility of a major performance gain over H.264/AVC, and will launch new standardization activities in due course. The directional transform and block-based PIF, offering new degrees of freedom to improve the overall R-D performance, are exactly what the next generation video coding requires.

Bibliography

[1] *Video codec for audiovisual services at p×64 kbits*, ITU-T Recommendation H.261, 1993.

[2] *Advanced video coding for generic audiovisual services*, ITU-T Recommendation H.264, Rev. 3, Apr. 2007.

[3] *Basic parameter values for the HDTV standard, for the studio and for international program exchange*, ITU-R Recommendation BT.709-3, 1998.

[4] *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 2: Video*, ISO/IEC IS 11 172-2, 1993.

[5] *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, ITU-R Recommendation BT.601-5, 1998.

[6] *Digital cinema system specification Version 1.2*, Digital cinema initiatives, LLC, 2008.

[7] *Parameter values for an expanded hierarchy of LSDI image formats for production and international programme exchange*, ITU-R Recommendation BT. 1769, 2006.

[8] "Vision and requirements for high-performance video coding (HVC)," ISO/IEC JTC1/SC29/WG11 document N10361, 2009.

[9] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York: McGraw-Hill, 2002.

[10] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. New Jersey: Prentice-Hall, 2002.

[11] T. K. Tan, G. J. Sullivan, and T. Wedi, "Recommended simulation common conditions for coding efficiency experiments revision 3," ITU-T Q.6/SG16 (VCEG) document VCEG-AI10, 2008.

[12] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs: Prentice-Hall, 1984.

[13] W. K. Pratt, *Digital Image Processing*. New York: John Wiley & Sons, Inc., 2001.

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. New Jersey: Prentice-Hall, 2002.

[15] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Boston: Academic Press, 1990.

[16] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Transactions of the Institute of Electronics and Communication Engineers*, vol. 71, no. 11, pp. 1095–1097, 1988.

[17] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1D DCT algorithm with eleven multiplications," in *Proc. of 1989 IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP1989)*, vol. 2, May 1989, pp. 988–991.

[18] H. F. Harmuth, *Transmission of Information by Orthogonal Functions*. New York: Springer-Verlag, 1969.

[19] C. Reader, "History of MPEG video compression," JVT document JVT-E066, 2002.

[20] J. Dong and K. N. Ngan, *Intelligent Multimedia Communication: Techniques and Applications.* Springer-Verlag Publisher, to be published in 2009, ch. Present and future video coding standards.

[21] *Video coding for low bitrate communication*, ITU-T Recommendation H.263, 1998.

[22] *Information technology - Generic coding of moving pictures and associated audio information: Video*, ISO/IEC IS 13818-2, 1996.

[23] *Information technology - Coding of audio-visual objects - Part 2: Visual*, ISO/IEC IS 14496-2, 1998.

[24] R. Koenen, "MPEG-4 overview," ISO/IEC JTC1/SC29/WG11 document N4668.

[25] M. van der Schaar, D. S. Turaga, and T. Stockhammer, *MPEG-4 beyond Conventional Video Coding : Object Coding, Resilience, and Scalability.* San Rafael: Morgan & Claypool Publishers, 2006.

[26] *Advanced video coding for generic audiovisual services*, ITU-T Recommendation H.264, Rev. 1, May 2003.

[27] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, Jul. 2003.

[28] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression Video Coding for Next-Generation Multimedia.* Chichester: Johl Wiley & Sons Ltd., 2003.

[29] *Advanced video coding for generic audiovisual services*, ITU-T Recommendation H.264, Rev. 2, Mar. 2005.

[30] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions." in *SPIE Conference on Applications of Digital Image Processing XXVII*, 2004.

[31] G. J. Sullivan, H. Yu, S. Sekiguchi, H. Sun, T. Wedi, S. Wittmann, Y.-L. Lee, A. Segall, and T. Suzuki, "New standardized extensions of MPEG4-AVC/H.264 for professional-quality video applications," in *Proc. of 2007 IEEE International Conference on Image Processing (ICIP2007)*, San Antonio, USA, Sept. 2007.

[32] *Advanced video coding for generic audiovisual services*, ITU-T Recommendation H.264, Rev. 4, Nov. 2007.

[33] A. Vetro, P. Pandit, H. Kimata, A. Smolic, and Y.-K. Wang, "Joint draft 8.0 on multiview video coding," JVT document JVT-AB204, 2008.

[34] *IEEE standard specifications for the implementations of 8×8 inverse cosine transform*, IEEE Std. 1180-1990, 1990.

[35] *Information technology - MPEG video technologies - Part 1: Accuracy requirements for implementation of integer-output 8×8 inverse discrete cosine transform*, ISO/IEC IS 23003-1, 2006.

[36] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 657–673, Jul. 2003.

[37] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 645–656, Jul. 2003.

[38] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.

[39] *Information technology - advanced coding of audio and video - Part2: Video*, GB T20 090.2, 2006.

[40] "Information technology - advanced coding of audio and video - part2: Video," AVS document N1572, Dec. 2008.

[41] "Information technology - advanced coding of audio and video - part2: Video," AVS document N1540, Sept. 2008.

[42] "Information technology - advanced coding of audio and video - part7: Mobility video," AVS document N1151, Dec. 2004.

[43] C. Zhang, L. Yu, J. Lou, W. K. Cham, and J. Dong, "The technique of prescaled integer transform: concept, design and applications," *IEEE Transactions on Circuits and System for Video Technology*, vol. 18, pp. 84–97, Jan. 2008.

[44] A. Habibi, "Comparison of n^{th}-order DPCM encoder with linear transformation and block quantization techniques," *IEEE Trans. Commun.*, vol. 19, pp. 948–956, Dec. 1971.

[45] J. Dong, K. N. Ngan, C. K. Fong, and W. K. Cham, "2D order-16 integer transforms for HD video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 10, pp. 1462–1474, Oct. 2009.

[46] J. Dong and K. N. Ngan, "Parametric interpolation filter for high-definition video coding," *submitted to IEEE Trans. Circuits Syst. Video Technol.*

[47] M. Wien and S. Sun, "ICT comparison for adaptive block transforms," ITU-T Q.6/SG16 (VCEG) document VCEG-L12, Jan. 2001.

[48] M. Wien and J. R. Ohm, "Simplified adaptive block transforms," ITU-T Q.6/SG16 (VCEG) document VCEG-O30r1, Dec. 2001.

[49] S. Naito and A. Koike, "Efficient coding scheme for super high definition video based on extending H.264 high profile," in *Proc. of SPIE Visual Commun. Image Process.*, vol. 6077, Jan. 2006, pp. 607 727–1–607 727 -8.

[50] S. Ma and C.-C. Kuo, "High-definition video coding with super-macroblocks," in *Proc. of SPIE Visual Commun. Image Process.*, vol. 6508, Jan. 2007, pp. 650 816–1–650 816–12.

[51] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Trans. Signal Process.*, vol. 49, no. 12, pp. 3032–3044, Dec. 2001.

[52] J. Liang, T. D. Tran, and P. Topiwala, "A 16-bit architecture for H. 26L, treating DCT transform and quantization," ITU-T Q.6/SG16 (VCEG) document VCEG-M16, Apr. 2001.

[53] W. K. Cham, "Development of integer cosine transforms by the principle of dyadic symmetry," *Proc. Inst. Electr. Eng. I: Commun. Speech Vis.*, vol. 136, no. 4, pp. 276–282, 1989.

[54] C.-X. Zhang, J. Lou, L. Yu, J. Dong, and W. K. Cham, "The technique of pre-scaled integer transform in hybrid video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, May 2005, pp. 316–319.

[55] W. K. Cham and Y. T. Chan, "An order-16 integer cosine transform," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1205–1208, May 1991.

[56] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *IEEE Trans. Instrum. Meas.*, vol. 45, pp. 353–361, Apr. 1996.

[57] W. K. Cham, Private communication.

[58] F. Bossen, "ABT cleanup and complexity reduction," JVT document JVT-E087, Oct. 2002.

[59] J. Dong, J. Lou, C.-X. Zhang, and L. Yu, "A new approach to compatible adaptive block-size transforms," in *Proc. of SPIE Visual Commun. Image Process.*, vol. 5960, Jul. 2005, pp. 38–47.

[60] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *Proc. of IEEE Int. Conf. Image Process.*, vol. 3, Oct. 2001, pp. 542 – 545.

[61] M. Wien, "Variable block-size transforms for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, Jul. 2003.

[62] T. Suzuki, K. Sato, and Y. Yagasaki, "Study of ABT for HDTV coding," JVT document JVT-E073r2, Oct. 2002.

[63] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU-T Q.6/SG16 (VCEG) document VCEG-M33, Apr. 2001.

[64] J. Dong, K. N. Ngan, C. K. Fong, and W. K. Cham, "A universal approach to developing fast algorithm for simplified order-16 ICT," in *Proc. 2007 IEEE Int. Symp. Circuits Syst. (ISCAS2007)*, New Orleans, USA, May 2007.

[65] T. Wedi, "Adaptive interpolation filter for motion compensated prediction," in *Proc. of IEEE Int. Conf. Image Process.*, vol. 2, Sept. 2002, pp. 509–512.

[66] Y. Vatis, B. Edler, D. T. Nguyen, and J. Ostermann, "Two-dimensional non-separable adaptive wiener interpolation filter for H.264/AVC," ITU-T Q.6/SG16 (VCEG) document VCEG-Z17, Apr. 2005.

[67] Y. Vatis and J. Ostermann, "P and B pictures with 2-D non-separable Wiener interpolation," ITU-T Q.6/SG16 (VCEG) document VCEG-AD08, Oct. 2006.

[68] ——, "Rate-distortion optimized coder control for adaptive interpolation filter in the KTA reference model," ITU-T Q.6/SG16 (VCEG) document VCEG-AE16, Jan. 2007.

[69] S. Wittmann and T. Wedi, "Simulation results with separable adaptive interpolation filter," ITU-T Q.6/SG16 (VCEG) document VCEG-AG10, Oct. 2007.

[70] D. Rusanovksyy, K. Ugur, and J. Lainema, "Adaptive interpolation with directional filters," ITU-T Q.6/SG16 (VCEG) document VCEG-AG21, Oct. 2007.

[71] K. Ugur, A. Hallapuro, D. Rusanovksyy, and J. Lainema, "Implementation of adaptive filters with 16-bit arithmetic," ITU-T Q.6/SG16 (VCEG) document VCEG-AG22, Oct. 2007.

[72] D. Rusanovskyy, K. Ugur, and J. Lainema, "Simulation results with directional interpolation filter (DIF)," ITU-T Q.6/SG16 (VCEG) document VCEG-AH17, Jan. 2008.

[73] D. Rusanovskyy, K. Ugur, A. Hallapuro, and J. Lainema, "Implementation of directional interpolation filter with 16-bit integer arithmetic," ITU-T Q.6/SG16 (VCEG) document VCEG-AH18, Jan. 2008.

[74] M. Karczewicz, P. Chen, and Y. Ye, "Switched interpolation filter with offset," ITU-T Q.6/SG16 (VCEG) document VCEG-AI35, Jul. 2008.

[75] M. Karczewicz, Y. Ye, P. Chen, and G. Motta, "Single pass encoding using switched interpolation filters with offset," ITU-T Q.6/SG16 (VCEG) document VCEG-AJ29, Oct. 2008.

[76] A. Fuldseth, G. Bjontegaard, D. Rusanovskyy, K. Ugur, and J. Lainema, "Low complexity directional interpolation filter," ITU-T Q.6/SG16 (VCEG) document VCEG-AI12, Jul. 2008.

[77] "Improvements on enhanced directional adaptive filtering (EDAIF-2)," Nokia and Qualcomm Inc., ITU-T Q.6/SG16 (VCEG) document COM16-C125-E, Jan. 2009.

[78] G. Motta, Y. Ye, and M. Karczewicz, "Improved filter selection for B-slices in E-AIF," ITU-T Q.6/SG16 (VCEG) document VCEG-AI38, Jul. 2008.

[79] M. Karczewicz, Y. Ye, P. Chen, and G. Motta, "Experimental results of interpolation filters on High-Definition sequences," ITU-T Q.6/SG16 (VCEG) document VCEG-AJ30, Oct. 2008.

[80] J. Nocedal and S. J. Wright, *Numerical optimization*. New York: Springer, 2006.

[81] J. Dong and K. N. Ngan, "Parametric interpolation filter for motion compensated prediction," in *Proceedings of 2009 IEEE International Conference on Image Processing (ICIP2009)*, Cairo, Egypt, Nov. 2009.

[82] K. N. Ngan and J. Dong, "Parametric interpolation filter for motion-compensated prediction," US Provisional Patent Application, Oct. 2009.

[83] T. Doyle and M. Looymans, "Progressive scan conversion using edge information," *Signal Processing of HDTV II*, pp. 711–721, 1990.

[84] M. J. J. C. Annegarn, T. Doyle, P. H. Frencken, and D. A. van Hees, "Video signal processing circuit for processing an interlaced video signal," U.S. Patent 4 740 842, Apr., 1988.

[85] G. de Haan and E. B. Bellers, "De-interlacing – an overview," *Proc. of IEEE*, vol. 86, no. 9, pp. 1839–1857, Sept. 1998.

[86] ——, "De-interlacing of video data," *IEEE Trans. Consum. Electron.*, vol. 43, pp. 819–825, 1997.

[87] M. Byun, M. K. Park, and M. G. Kang, "EDI-based deinterlacing using edge patterns," in *Proc. of IEEE Int. Conf. Image Process. 2005*, vol. 2, Sept. 2005, pp. 1018–1021.

[88] C. Wang, H. Han, and S. Peng, "A spatial deinterlacing algorithm based on edge orientation optimized in local area," in *Proc. of Congress on Image and Signal Process. 2008*, vol. 4, May 2008, pp. 87–91.

[89] H. Hwang, M. h. Lee, and D. I. Song, "Interlaced to progressive scan conversion with double smoothing," *IEEE Trans. Consum. Electron.*, vol. 39, pp. 241–246, Aug. 1993.

[90] P. Haavisto, J. Juhola, and Y. Neuvo, "Scan rate up-conversion using adaptive weighted median filtering," *Signal Process. of HDTV II*, pp. 703–710, 1990.

[91] Q. Huang, W. Gao, D. Zhao, and Q. Huang, "An edge-based median filtering algorithm with consideration of motion vector reliability for adaptive video deinterlacing," in *Proc. of IEEE Int. Conf. on Multimedia and Expo 2006*, Jul. 2006, pp. 837–840.

[92] R. A. Braspenning and G. de Haan, "True-motion estimation using feature correspondence," in *Proc. of SPIE Visual Commun and Signal Process. 2004*, Jan. 2004, pp. 229–239.

[93] S. Li, J. Du, D. Zhao, Q. Huang, and W. Gao, "An improved 3DRS algorithm for video de-interlacing," in *Proc. of 2006 Picture Coding Symposium (PCS2006)*, Apr. 2006.

[94] R. Li, B. Zeng, and M. L. Liou, "Reliable motion detection/compensation for interlaced sequences and its applications to deinterlacing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 23–29, Feb. 2000.

[95] Y.-L. Chang, S.-F. Lin, C.-Y. Chen, and L.-G. Chen, "Video de-interlacing by adaptive 4-field global/local motion compensation approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 1569–1582, Dec. 2005.

[96] J. L. Yen, "On nonuniform sampling of bandwidth-limited singals," *IRE Trans. Circuit Theory*, vol. CT-3, pp. 251–257, Dec. 1956.

[97] P. Delogne, L. Cuvelier, B. Maison, B. van Caillie, and L. Vandendorpe, "Improved interpolation, motion estimation and compensation for interlaced pictures," *IEEE Trans. Image Process.*, vol. 3, pp. 482–491, Sept. 1994.

[98] L. Vandendorpe, L. Cuvilier, B. Maison, P. Queluz, and P. Delogne, "Motion-compensated conversion from interlaced to progressive formats," *Signal Process.: Image Commun.*, vol. 6, pp. 193–211, Jun. 1994.

[99] G. Thomas, "A comparison of motion-compensated interlaced-to-progressive conversion methods," *Signal Process. Image Commun.*, vol. 12, pp. 209–229, 1998.

[100] D. Wang, A. Vincent, and P. Blanchfield, "Hybrid de-interlacing algorithm based on motion vector reliability," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 1019–1025, Aug. 2005.

[101] D. van de Ville, W. Philips, and I. Lemahieu, "Motion compensated de-interlacing for both real time video and still images," in *Proc. of IEEE Int. Conf. Image Process. 2000*, vol. 2, Sept. 2000, pp. 680–683.

[102] Q. Huang, W. Gao, D. Zhao, and H. Sun, "Adaptive deinteralcing for real-time applications," in *Pacific-Rim Conf. on Multimedia*. LNCS 3768, 2005, pp. 550–560.

[103] M. J. Chen, C. H. Huang, and C. T. Hsu, "Efficient de-interlacing technique by inter-field information," *IEEE Trans. Consum. Electron.*, vol. 50, pp. 1202–1208, Nov. 2004.

[104] B. D. Simsic and P. L. Swan, "Method and apparatus for de-interlacing interlaced content using motion vectors in compressed video streams," U.S. Patent 6 269 484, Jul., 2001.

[105] J. Dong and K. N. Ngan, "An adaptive and parallel scheme for HD video de-interlacing," in *Proc. of IEEE Int. Conf. on Multimedia and Expo 2008*, Jun. 2008, pp. 1137 - 1140.

[106] ——, "Real-time de-interlacing for H.264 coded HD videos," *submitted to IEEE Trans. Circuits Syst. Video Technol.*

[107] J. W. Woods and S.-C. Han, "Hierarchical motion compensated de-interlacing," *Proc. SPIE*, vol. 1605, pp. 805–810, 1991.

[108] K. N. Ngan, J. Dong, and Y. Huo, "Method and apparatus of de-interlacing video," US Non-Provisional Patent Application No. 12/129891, May 2008.

[109] J. Dong and K. N. Ngan, "16×16 integer cosine transform for HD video coding," in *Proceedings of the Seventh IEEE Pacific-Rim Conference on Multimedia (PCM2006)*, Hangzhou, China, Nov. 2006.