THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF ELECTRONIC AND INFORMATION ENGINEERING

Adaptive Integer Kernels and Dyadic Approximation Error Analysis for State-of-the-Art Video Codecs

Wang Qiuwei

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

July 2010

DECLARATION

The author of this thesis is officially registered with the name *Wang Qiuwei* at the Hong Kong Polytechnic University, and he publishes research papers under the name *Wong Chau Wai*.

Wang Qiuwei and *Wong Chau Wai* are equivalent, and both of them are romanisated from the author's official Chinese name 王秋韡 with Hanyu Pinyin and the Hong Kong Government Cantonese Romanisation, respectively.

CERTIFICATION OF ORIGINALITY

WANG QIUWEI

DEDICATION

To my parents

ABSTRACT

In this thesis, new integer kernels are found and adaptive transform coding techniques are proposed to improve the coding efficiency of state-of-the-art video codecs with detailed analyses. The nonorthogonality error analysis is extended and improved. An error caused by the dyadic fraction approximation due to the integerization of transform coding is defined and followed by deep investigation.

The desire for the removal of the mismatch between encoder and decoder has been ever increasing. In the state-of-the-art video coding standard—the H.264/AVC—the transform coding stage was thus integerized to cope with this desire. One of our objectives is to improve the coding efficiency of video codec based on this "integer framework". We propose a new DCT-like integer kernel IK(5,7,3) and revitalize another DCT-like integer kernel IK(13,17,7) for the transform coding process of hybrid video coding. Making use one of these kernels together with the H.264/AVC Kernel IK(1,2,1), we are able to design new multiple-kernel schemes which give better coding performance over that of the conventional approaches. All these schemes make use of the Adaptive Kernel Mechanism (AKM) at macroblock level, which requires heavy computation during the encoding process. We subsequently discovered that a rate-distortion feature extracted from a pair of kernels gives an intrinsic property that can be used to select a better kernel for a two-kernel macroblock-level AKM system. This is a power tool with theoretical interest and practical uses. In order to reduce computation substantially, we make use of this tool to make an analysis and design of a frame-level AKM and come up with a simple solution that the kernel IK(1,2,1) be used for I- and P-Frames and the kernel IK(5,7,3) be used for B-Frames coding. This proposed frame-level AKM is similar, or even better, than the proposed macroblock-level AKM. Furthermore it substantially reduces computation and certainly gives a good improvement in terms of the PSNR and bitrate compared to those obtained from the H.264/AVC default arrangement and other macroblock-level AKM schemes available in the literature.

Nowadays, the demand for large-size (e.g. 16×16) integer transform kernels is increasing due to the explosive increase of resolution of videos. However, the orthogonality constraint for designing 16×16 integer kernels is much stronger than that for designing 4×4 kernels. Hence, several kernel designs violating the constraint in a controllable manner which roughly ensures the orthogonality have been proposed. An error analysis by Dong *et al.* showed that the well-controlled nonorthogonality noise is approximately negligible as compared to the quantization noise. In this thesis, we enhance the original analysis by pointing out three problems found in derivations and also giving two comments. Nevertheless, the problems are defects only, hence do not affect the overall justifications to the nonorthogonality analysis.

Although the integerization of transform coding process ensures no mismatch between encoder and decoder, it also introduces a by-product and we define it as the “dyadic approximation error” which can largely affect the visual quality of a reconstructed video sequence. We derive the analytical forms of the dyadic approximation error, and compare the significances among possible error terms (i.e. the quantization error, nonorthogonality error, and dyadic approximation error) using various transform kernels. We conclude that the dyadic approximation error is much larger than the nonorthogonality error, and it is comparable to the quantization error for fine quantization. We point out that the existence of this error is equivalent to scaling each frequency component by a position dependent scalar which is slightly larger or smaller than the unity, and also quantizing them with different stepsizes. Hence in the reconstruction process, many distorted frequency components are used, and eventually a reconstructed frame with frequency artifacts is generated. The conditions to eliminate the effect of dyadic approximation error for 16×16 transform kernels are found by experimental work.

On the whole, inspired by the establishment of the “integer framework” since the emergence of the H.264/AVC, we carry out a comprehensive investigation on the old problems under the new constraint, starting from the optimization of coding performance to the analyses of errors.

LIST OF PUBLICATIONS

Journal Papers

[1] **Chau-Wai Wong** and Wan-Chi Siu, "Comments on '2-D Order-16 Integer Transforms for HD Video Coding,'" *IEEE Transactions on Circuits and Systems for Video Technology*, accepted for publication.

[2] **Chau-Wai Wong** and Wan-Chi Siu, "Transform Kernel Selection Strategy for the H.264/AVC and Future Video Coding Standards," 2nd revised version submitted to the *IEEE Transactions on Circuits and Systems for Video Technology*.

[3] **Chau-Wai Wong** and Wan-Chi Siu, "Analysis on Dyadic Approximation Error for Hybrid Video Codecs with Integer Transforms," *IEEE Transactions on Image Processing* (1st review results received, a revised version is being prepared and to be submitted within 6 weeks).

International Conference Papers

[4] **Chau-Wai Wong** and Wan-Chi Siu, "Transform Kernel Selection Strategy for the H.264," *APSIPA Annual Summit and Conference (APSIPA09)*, pp. 64–70, Sapporo, Japan, 4–7 Oct 2009.

ACKNOWLEDGEMENT

I would like to express sincere gratitude to various peoples from the Hong Kong Polytechnic University, whom I have the opportunity to work with. My major appreciation goes to my supervisor Prof. W.C. Siu. Not only his professional advices concerning my study, hard-working style and willingness to devote to the advancement of science and technology, but also his view on the way of life, are a precious knowledge to me. This study could never reach its present level without his constant and kindly support.

Special thanks are given to Mr. W.L. Hui, Mr. W.H. Wong, Mr. Y.H. Kam, Dr. K.O. Cheng, Dr. C.H. Fu and Dr. H.K. Cheung for their invaluable advices when I was facing difficulties. I am greatly indebted to my colleagues, Ms. L.L. Wang, Mr. C.S. Wong, Mr. K.W. Hung, Mr. C.S. Sun, Mr. X. Zhang, Dr. X. Jing and Dr. Y. Zhang. Their expert knowledge and friendly encouragement have helped me overcome many difficulties in my study. It has been a wonderful time working with them for these two years.

I also want to thank all clerical staff in the general office of the Department of Electronic and Information Engineering. They have provided reliable support and created a good environment for me to work in.

Finally, I am deeply grateful to my family and friends for their support and encouragement. Without their understanding and patience, it is impossible for me to complete this research study.

STATEMENTS OF ORIGINALITY

The following contributions reported in this thesis are claimed to be original.

1. We have proposed a new DCT-like integer kernel IK(5,7,3) and revitalized another DCT-like integer kernel IK(13,17,7) for the transform coding process of hybrid video coding. Making use one of these kernels together with the H.264/AVC Kernel IK(1,2,1), we have designed new multiple-kernel schemes which give better coding performance over that of the conventional approaches.

2. We have discovered that a rate-distortion feature extracted from a pair of kernels gives an intrinsic property that can be used to select a better kernel for a two-kernel macroblock-level Adaptive Kernel Mechanism (AKM) system. It is a power tool with theoretical interest and practical uses. In order to reduce computation substantially, we have made use of this tool to make an analysis and design of a frame-level AKM and come up with a simple solution that the kernel IK(1,2,1) be used for I- and P-Frames and the kernel IK(5,7,3) be used for B-Frames coding.

3. We have enhanced the error analysis of nonorthogonality of transform coding by pointing out three problems found in derivations of a recent paper and also giving two comments.

4. A "dyadic approximation error" which can largely affect the visual quality of reconstructed video sequence has been defined by us. We have derived the analytical forms of the dyadic approximation error, and have compared the significances among possible error terms (i.e. the quantization error, nonorthogonality error, and dyadic approximation error) using various transform kernels. We have concluded that the dyadic approximation error is much larger than the nonorthogonality error, and it is comparable to the quantization error for fine quantization. We have pointed out that the existence of this error is equivalent to scaling each frequency component by a position dependent scalar which is slightly larger or smaller than the unity, and also quantizing them with different stepsizes. Hence in the reconstruction process, many distorted frequency components are used, and eventually a reconstructed frame with frequency artifacts is generated. The conditions to eliminate the effect of dyadic approximation error for 16×16 transform kernels have been found by experimental work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

1-d	One-Dimension / One-Dimensional
2-d	Two-Dimension / Two-Dimensional
3-d	Three-Dimension / Three-Dimensional
AKM	Adaptive Kernel Mechanism
AR	Autoregressive Model
BinDCT	Binary Arithmetic DCT
CBP	Coded Block Pattern
CIF	Common Intermediate Format
CMT	C-Matrix Transform
DC	Digital Camera
DCT	Discrete Cosine Transform
DPCM	Differential Pulse-Code Modulation
DST	Discrete Sine Transform
DVD	Digital Versatile Disc
FM-AKM	FraMe-level Adaptive Kernel Mechanism
GCT	Generalized Chen Transform
GFM-AKM	Generalized FraMe-level Adaptive Kernel Mechanism
GOP	Group of Pictures
HD	High Definition
HVC	High-Performance Video Coding
ICT	Integer Cosine Transform
IDCT	Inverse DCT
IK	Integer Kernel
IntDCT	Integer DCT
IST	Integer Sine Transform
KLT	Karhunen–Loève Transform
KPE	Kernel Percentage Error
KTA	Key Technical Area
LHS	Left-Hand Side
MB	MacroBlock
MB-AKM	MacroBlock-level Adaptive Kernel Mechanism
MDDT	Mode Dependent Directional Transform
ME	Motion Estimation
MF	Multiplication Factor

MV	Motion Vector
NGVC	Next-Generation Video Coding
NICT	Nonorthogonal Integer Cosine Transform
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
QStep	Quantization Step
RF	Rescaling Factor
RHS	Right-Hand Side
ROT	ROtational Transform
WHT	Walsh–Hadamard Transform

Chapter 1. Introduction

With the development of the information and communication technology, the processing of video on commercial products (i.e. the personal computers) has become available since 1990s. From then on, the digital video application has a more and more important role in our everyday work and life. For example, the videoconferencing allows people separated by the Atlantic Ocean to communicate effectively as if they were in one place. Or, people may watch high-quality films at home with the support of DVD (Digital Versatile Disc) players. The digital video application thrives after entering the new millennium, and it has significantly changed the way of our life in recent years. Shooting a video and uploading it to the Internet instantly has become a popular trend, thanks to the wide spread of the digital cameras (DC) and the mobiles phones with the DC function. It is also a brand new life experience to watch three-dimensional (3-d) movies and football matches in cinemas, at the time of writing. And in the promising future, 3-d television will be a new entertainment for families.

The sketch of the digital video application that we have described reveals its trace of evolution: from low-resolution to high-resolution, and from 2-d video to 3-d video. From the technological perspective, the main challenge that the researchers face is to compress the video data more severely. In the jargon of video professionals, the coding efficiency of videos has to be increased. This is supported by the fact that the increase of the size of the raw video contents is much faster than the increase of the channel capacity for transmission. For example, an upgrade of user experience of online video streaming from CIF (352×288) to 720p (1280×720) needs a bandwidth increase of 809% without the consideration of improvement of the compression technique. Hence it is essential to develop a new video codec with higher compression ability than the conventional ones by optimizing each functional block of the codec, and therefore a significant improvement in coding efficiency can be achieved.

Since the emergence of the first video coding standard twenty year ago, there have been many standards, i.e. H.261 [1], MPEG-1 [2], H.262/MPEG-2 [3], H.263 [4], MPEG-4 [5], and H.264/AVC [6], trying to reach higher coding efficiency based on previous ones. One thing in common is that all of them are hybrid video coding standard which makes use of the motion compensation, transform coding, quantization, entropy coding and so on to encode videos. In this thesis, we focus on the algorithms and techniques of the transform coding and quantization of the state-of-the-art video coding standard—the H.264/AVC, to improve the coding efficiency and investigate the theories behind.

1.1. Literature Review

1.1.1. Optimal Transforms for Image and Video Coding

A video is a three-dimensional (3-d) signal with one in the temporal domain (time domain) and the other two in the spatial domain. Generally, the motion compensation is used to remove the temporal redundancy in the time axis, and the transform coding is used to remove the spatial redundancy in the Cartesian coordinate system. The transform coding for videos deals with the two-dimensional (2-d) signals. There are two approaches to deal with the 2-d signals—one is to do a raster scan so that the 2-d signal is re-ordered into a 1-d signal hence 1-d transform coding techniques can be applied; the other is to apply the 1-d transform coding techniques to the horizontal and vertical directions of the 2-d signal independently. In practice, the second approach is more popular due to its low computational complexity. Up to now, all the video coding standards have adopted the second approach to code the 2-d residual signals. In this thesis, the second approach is assumed unless specified otherwise.

Starting from the first international video coding standard the H.261, the discrete cosine transform (DCT) [7-9] has been adopted for data compaction and decorrelation at the transform coding stage. Researchers have provided results on various studies of the DCT since its invention [8-9]. Some of them concentrated on different applications and fast algorithms of the DCT [10-12], whilst others made further investigation on various forms of DCTs. Until the most recent standard the H.264/AVC, it is replaced by the integer transform which also belongs to the family of DCT-like transforms. The DCT has been widely accepted as the most suitable transform due to its balance between coding efficiency and computational complexity. However, the limitations of the DCT have triggered numerous proposals, either to improve or replace it with a better algorithm.

1.1.1.1. Karhunen–Loève Transform (KLT)

To improve the coding efficiency of a video codec within the transform coding stage, the most intuitive idea is to adapt the transform kernel to the statistics of the input data fed for the transform coding process. The Karhunen–Loève (KL) Transform [13-14] can provide the optimal transform kernel which adapts to the statistics of input data, and thus the obtained kernel has the best decorrelation and energy compaction properties. Effros and Chou [15], Dony and Haykin [16], and Archer and Leen [17] proposed using the KL Transform (KLT) to replace the DCT in order to achieve higher coding efficiency with the cost of increasing the computational complexity. Dapena and Ahalt [18] tackled the problem by allowing the switch between the KLT and DCT, which can reduce the computational complexity to some extent. However, this family of algorithms by using the KLT requires rather high computational power hence it is seldom used for video coding.

1.1.1.2. Auto-Covariance Modeling

Another direction is to represent the collective behavior (up to 2nd order statistics) of the input data by the auto-covariance model, and then a transform kernel which has comparable computational complexity with respect to the DCT Kernel can be easily obtained. Chen [19-20] first analytically derived the auto-covariance model making use of the translational motion in the motion compensation. Niehsen [21] improves the model by taking the overlapped block motion compensation into account. Hui and Siu [22-23] further improve Chen's model using a deformation model by assuming both the deformation vectors of different pixels and the mean of deformation vectors are Gaussian distributed. The subsequently derived model can fit the real data better than the previous models. The auto-covariance model can provides better transforms which fit the data better than the DCT.

1.1.1.3. Directional Transforms

However, one may argue that the statistics of all blocks represented by the auto-covariance model still cannot well represent the detailed features of all blocks. One of these features is the directional edges, which cannot be fully characterized by the conventional auto-covariance model since it can only fully represent the horizontal and vertical statistics. Kamisli and Lim [24] had given an in-depth analysis by using the generalized separable auto-covariance model in which the information of edge direction has been incorporated, and subsequently shown that the correlation is strong along the edge directions and weak perpendicular to edge directions. This is a good justification to various directional schemes. Helsingius, Kuosmanen, and Astola [25] improved the coding efficiency by using a set of non-separable directional transforms which were trained from artificial edges of various orientations. Robert, Amonou and Pesquet-Popescu [26-28], and Kamisli and Lim [24, 29] proposed using the circular shifts which align the pixels of a slanted edge into a horizontal or vertical edge before taking the conventional horizontal or vertical DCT. Zeng and Fu [30-33] developed a framework that the first transform is performed along a direction rather than the horizontal or vertical direction and the second transform is applied to the resultant DCT coefficients of the first transform. Drémeau *et al.* [34] extends the idea from square blocks to rectangular blocks. Chang and Girod [35] refined Zeng's algorithm by modifying some transform directions, zigzag scanning pattern, and block partitioning. Xu, Xu and Wu [36] argued that Zeng and Fu's algorithm does not exploit the correlation among neighboring blocks with the similar direction, and does not consider the directions at fractional pixels. They therefore proposed a directional DCT-like transform based on the lifting structure of the DCT by incorporating the care for the directional edges into the so-called "primary operations" of the lifting steps. Xu, Wu, Liang and Zhang [37-38] further combined this idea with the lapped transforms and proposed the so called "directional lapped transforms."

1.1.1.4. Engineering Approach

While continuous efforts have been dedicated to the improvement of the directional approach based upon theoretical analyses, practitioners are seeking for schemes with higher coding efficiency yet reasonable computational complexity comparing to the H.264/AVC. Ye and Karczewicz [39-41] proposed a set of separable directional transforms (which is commonly referred to as the mode dependent directional transform (MDDT)) for each intra-prediction mode of the H.264/AVC, and certain improvement has been reported. However, the MDDT neither use the circular shifts nor perform a transform along a particular direction; instead, the 1-d transforms of the separable MDDT are applied horizontally and vertically in a fashion similar to the conventional 2-d DCT, and the transform kernels for each intra-mode are obtained empirically by the KLT of the residual data of that particular mode. To be frank, this method is not a truly directional approach, since it does not directly exploit the pixels in the most correlated directions but indirectly exploits the pixels in horizontal and vertical directions. Nevertheless, this method is easy to understand and implement, and requires low computation power; hence it has been adopted into the evaluation software of the next generation video codec, the Key Technical Area (KTA) [42], and may proposals (e.g. TI [43], Media Tek [44], LG [45], Huawei [46], SK telecom [47], Toshiba [48], ETRI [49], and etc.) for the new standards has employed it or its variation. Samsung and BBC [50] proposed a rotational transform (ROT) which rotates the basis of the DCT Kernel so that directional structure can be better caught by the rotated kernel. However, we believe that this algorithm also cannot fully exploit the directional information. At the time of writing, both MDDT and ROT were included in the document named "Test model under consideration" [51], which made them a closer step to the adoption in a test model for the development of the future video coding standard.

1.1.1.5. Multiple Candidate Kernels

Most algorithms mainly employ a single transform kernel which suits the feature of most input data, leaving the remaining data not well coded. To remedy this drawback, researchers propose multiple candidate kernels instead of a single kernel. Amongst the multiple-kernel scheme, three subclasses can be defined. The first subclass is the directional transforms we have mentioned before, since each orientation implies a different transform kernel. The second subclass uses self-organizing method to classify input data into different classes, and then find the KLT for each class. Wornell and Staelin [52] proposed an iterative Maximum Likelihood algorithm to find several kernels that are fit to input data sets with various characteristics. Dony and Haykin [16] investigated the best adaption criterion, and input data can be classified into appropriate classes by the subspace classifier. Recently, Zhao *et al.* [46, 53] proposed using multiple kernels for each intra-prediction mode to further improve the coding efficiency. The third subclass, which focuses more on the transform kernels, uses kernels with known characteristics (e.g. DCT, DST, etc.) to code input data. Rose, Heiman, and Dinstein [54], considered an segmented image as an chessboard pattern, and proposed the KLT/DCT for the "black" blocks and the DST for the residues of "white" blocks. The residue of a "white" block is obtained by subtracting the predicted signal using the information provided by the surrounding "black" blocks from its original value. The use of the DST to code the residues is justified by Jain [55] that a class of signal with boundary information is best coded with the sine transform. Recently, Han, Saxena, and Rose [56] theoretically analyzed the intra-prediction residue of H.264/AVC, and subsequently proved the optimality of the sine transform for the DC mode, horizontal mode, and vertical mode, which is not surprising since the residues of intra-prediction is a class of signal with boundary information. Lim *et al.* [57-59] proposed using DCT or DST to code intra- and inter-blocks, and some performance gain was achieved. In a recent response to CfP for the future standard, NHK [60] proposed using DST for residues of chroma components. Besides the popularity of the DST, Helsingius, Kuosmanen, and Astola [25] also tried the Walsh–Hadamard transform (WHT) as an alternative kernel in addition to the DCT. However, the performance is even worse since extra bits are needed to indicate the selected kernel.

A graph showing the various transform strategies is summarized in Fig. 1-1.

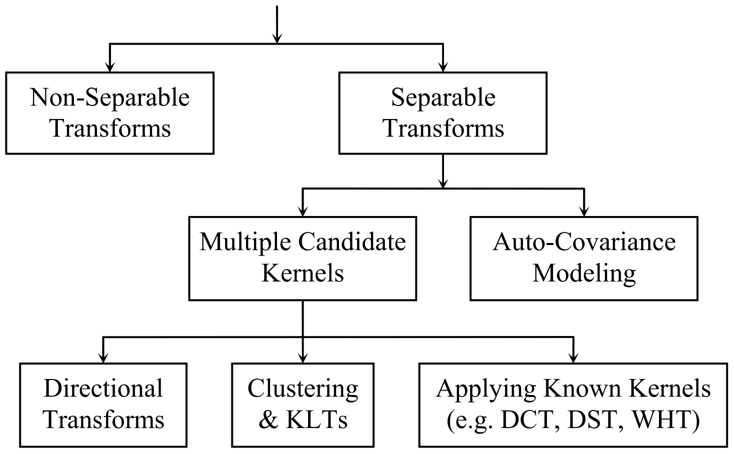


Fig. 1-1. Various transform strategies to improve the coding efficiency.

1.1.2. Drift-Free Transforms

1.1.2.1. Drift Controlling

For the video coding standards prior to the H.264/AVC, there exists a problem called the "drift" which leads to the reconstructed video deviates from the expected video that the encoder side generates. It is caused by the mismatch—the different implementations of the inverse DCT—between the encoder side and decoder side. In the early days, video people tends bound the mismatch error within an acceptable range. The IEEE standard 1180 [61] is a detailed description on this by defining the statistical upper bounds of the peak error, overall mean square error, mean error, and overall mean error. It was started and its amendments were evolved from the H.261, and they have subsequently served as the mismatch-controlling standards for the later video coding standards, i.e. the MPEG-1, H.262/MPEG-2, H.263, and MPEG-4. The various mismatch-controlling standards were later centralized by MPEG-C Part 1 and its first amendment [62-64] due to the withdrawal of IEEE standard 1180. However, the mismatch was not totally eliminated by the controlling algorithms such as the oddification [1-2, 4, 61] and LSB toggling [3, 6, 65]. Meanwhile, the high-quality video applications worsen the drift problem. Hence, the intrinsic drift-free transforms are demanding.

1.1.2.2. Drift Elimination via Integer Approximations of Transforms

The most intuitive way to avoid drift is to replace the floating-point arithmetic with the integer arithmetic, and many algorithms have been proposed accordingly. Jones, Hein, and Knauer [66] found the integer approximation of the conversion matrix (which can be viewed as the transform kernel) of the order-8 C-matrix transform (CMT) by trial and error. Srinivasan, Rao, and Kwak subsequently found the integer approximations of order-16 CMT [67] and order-32 CMT [68]. The CMT approach has two major drawbacks: i) the input signal needs to be preprocessed by the WHT before the transformation using the conversion matrix, and ii) the computational complexity increases significantly as the order of the transform goes larger due to the fact that the conversion matrix becomes less and less sparse. Cham and Chan [69-71] proposed another integer approximation of the DCT called the integer cosine transform (ICT). The ICT replaces the floating-point coefficients of a DCT Kernel with integers, while it maintains the orthogonality of the kernel and its relative ratios among the coefficients. The principle of dyadic symmetry (which is equivalent to the property of the recursive factorization of the DCT Kernel [72]) is utilized to simplify the formulations of the constraints. This approach is straightforward and can be implemented with fast algorithms [73-74]. However, the similarity of the ICT kernel to the DCT Kernel is mainly restricted by the number of bits representing the integer coefficients of the ICT kernel. Allen and Blonstein [75] proposed a similar algorithm named the generalized Chen transform (GCT), which also makes the use of the recursive factorization of the DCT Kernel but parameterizes the coefficients of the DCT Kernel differently. Liang and Tran [76-77] proposed a binary arithmetic DCT (binDCT), which fully makes the use of the recursive factorization that the transform kernel is completely decomposed into 2×2 rotation matrices, and is flexible that perfect reconstruction can be achieved with different levels of finite-bit precision. The perfect reconstruction is ensured by the fact that any rotation matrix and its inverse can be decomposed into a pair of similar lifting matrices. Chen, Oraintara, and Nguyen [78] proposed a similar design to the binDCT called the Integer DCT (IntDCT).

In the standardization community, Bjontegaard [79] proposed an integer cosine transform kernel by rounding the up-scaled DCT Kernel (which is similar to Cham's ICT) for the video codec aiming to totally remove the "IDCT mismatch". The design of the ICT was gradually improved and one of the ICTs was finally adopted [80-82] as the transform kernel of the H.264/AVC due to its simplicity and drift-free characteristic. Other video coding standards developed later, such as the Audio Video Standard (AVS) [83] and the SMPTE 421M (VC-1) [84], also make use of integer approximations of DCT instead of the conventional floating-point DCT.

1.1.3. Coping with High Resolution Video Applications with Large-Size Transforms

1.1.3.1. Need for Large-Size Kernels

As the video technology evolves, the demand for video contents of higher definition can be gradually fulfilled. The development of the state-of-the-art H.264/AVC was focused on increasing the compression ratio through using a combination of various coding tools, hence the H.264/AVC can exploit the inter-frame redundancy excellently by varying its motion estimation (ME) unit from block-size of 4×4 to 16×16. However, during the development of this standard, not much consideration on higher definition contents (e.g. 720p, 1080p, and etc.) were taken, hence the performance of the H.264/AVC for higher definition contents is not so good as for lower definition contents (e.g. QCIF, CIF, and etc.) This is because the size of the ME unit comparing to the size of the video objects becomes smaller as the definition of videos increases. Experimental work [85-86] proved that an enlargement of the ME unit can help to improve the coding efficiency. Simultaneously, larger-size (e.g. 16×16) transform kernels are also needed to cope with the enlargement of the ME unit.

1.1.3.2. Proposed Order-16 Kernels

Since the H.264/AVC uses 4×4 and 8×8 transform kernels, 16×16 (order-16) and even larger kernels are of interest. Cham and Chan [70] proposed a family of order-16 ICT kernels that has a higher efficiency than the Walsh–Hadamard transform (WHT) and C-matrix transform (CMT). Dong, Ngan, Fong, and Cham [73-74, 87] proposed a universal approach to develop fast algorithm for this family of ICT kernels so that these kernels can be incorporated into the video codec, and experimental results show that the order-16 kernels can improve the efficiency of video coding. Wien, Mayer, and Ohm [88] proposed a CMT kernel for the WHT domain implementation. Ma and Kuo [89] proposed a quasi–DCT-like kernel with non-smooth even-row basic vectors which trades for a low computational complexity.

1.1.3.3. Nonorthogonality Issue

Recently, Chen, Ye, and Karczewicz [90] proposed using an up-scaled order-16 integer kernel, and the kernel was adopted together with other proposed algorithms into the evaluation software KTA due to the overall performance improvement. However, this kernel is a nonorthogonal one (note that only order-4 DCT Kernel remains orthogonal after up-scaling and rounding) hence additional error will be introduced when it is used for video coding. Lee *et al.* [91] pointed out its nonorthogonality, and subsequently proposed an orthogonal one. Joshi, Reznik, and Karczewicz [92-93] also proposed one with a simple recursive factorization structure leading to fast implementation. Dong, Ngan, Fong, and Cham [74] analytically proved that the nonorthogonality error caused by up-scaling and rounding is insignificant in terms of the variance measurement, hence one of the kernel search criteria—the orthogonality—can be released a little bit so that much more nonorthogonal kernels that resembles the DCT Kernel well can also be used for video coding. They subsequently proposed a family of nonorthogonal ICT (NICT) kernels for video coding, and experimental results show that the NICT kernels perform better than the conventional ICT kernels.

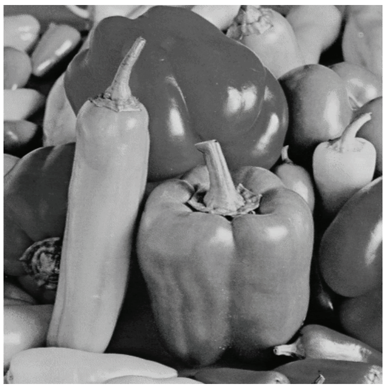
1.2. Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we make a technical review from the basic fundamentals on image modeling to the state-of-the-art video coding technology, the H.264/AVC. In Chapter 3, we proposed a set of new ICT kernels and developed a kernel selection strategy according to a newly-found rate-distortion feature in order to improve the coding efficiency. Chapter 4 extends and improves a recent error analysis on the nonorthogonality of the transform coding process. In Chapter 5, we propose a novel analysis on the dyadic approximation error introduced by the integerization of transform coding. We pointed out that the dyadic approximation error is much larger than the nonorthogonality error, and it is comparable to the quantization error for fine quantization. In Chapter 6, the whole thesis is concluded and future research directions are discussed.

Chapter 2. Technical Review

2.1. Basic Theories

The natural images have many geometric characteristics which not only help human "imagining" the parts that are covered or removed but also enable coding tools to achieve higher compression ratio by exploiting the characteristics. One of the well accepted characterizing methods is to describe regions of images by the following three categories, namely, the smooth region, edge, and texture (See Fig. 2-1). Amongst those characteristics, the smooth region is the easiest one to be "understood" by machines, and was indeed well exploited by the image codecs with the help of the discrete cosine transform (DCT).


(a)

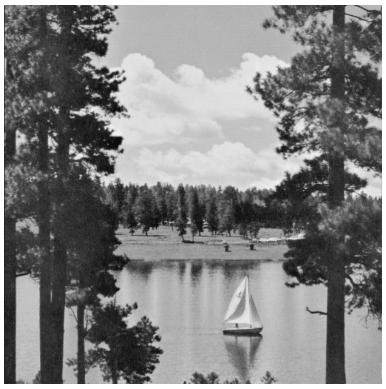

(b)

Fig. 2-1. Standard gray-level test images: (a) Peppers and (b) Lake.

To understand why the DCT is efficient for the smooth region, a careful observation and a statistical mindset are needed. Let us look at the smooth regions of various images carefully. It is not difficult to observe that the intensities of the images do not vary significantly, or at least the intensities of neighboring pixels do not differ significantly from one to another. Converting the above observation to a computer understandable form, researcher describe the "smoothness" using a statistical tool called the first-order autoregressive model (AR(1)) or first-order Markov process. The details of the AR(1) are described in the following.

2.1.1. First-Order Autoregressive Model (AR(1))

Although the image is a 2-d signal, it still can be segmented properly into a set of 1-d signals (e.g. segmenting horizontally or vertically). Let us model a collection of 1×N smooth regions using a random vector

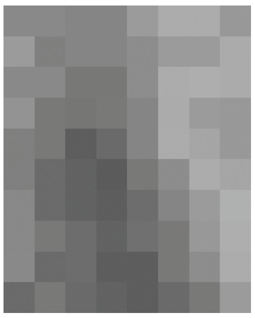
$$\mathbf{P} = [P(0)\ P(1)\ P(2)\ \dots\ P(N-1)], \tag{2-1}$$

where N can be any integer value (radix-2 integers are preferable) and $P(n)$'s are N successive random variables corresponding to N horizontal positions. Each observation of the random vector **P** represents an example of how the model describes one of the 1×N regions of the real images. Let us call an example by extracting ten 1×8 vectors from the bottom-left region of Fig. 2-1(a). As is shown in Fig. 2-2(b), each row can be viewed as an observation of the random vector **P**, and each column can be viewed as a random variable comprising **P**. In our example, there are total 10 observations.

The random process $\{P(n)\}$ fulfills a condition called the wide-sense stationary under which i) all variables have a same mean value, and ii) autocorrelation between any two variables only depends on their distance. Hence we can subtract the mean from **P** and construct a new random vector

$$\mathbf{X} = [X(0)\ X(1)\ X(2)\ \ldots\ X(N-1)], \tag{2-2}$$

where $X(n)$'s are called the centered random variables, and their observations have been shown in Fig. 2-2(c). Note that the removal of mean can simplify the subsequent analysis without affect the correctness.



(a)

137	137	133	133	155	172	172	152
155	128	133	133	141	155	155	170
137	137	120	120	141	172	174	170
147	121	118	121	133	170	155	152
128	121	93	108	133	172	166	152
128	109	98	91	121	141	172	166
136	109	98	93	111	133	155	176
136	121	111	98	114	121	152	174
137	109	111	93	93	121	141	172
109	121	108	98	93	108	121	155

One Observation

One Random Variable

(b)

3.5	3.5	-0.5	-0.5	21.6	38.6	38.6	18.6
21.6	-5.5	-0.5	-0.5	7.5	21.6	21.6	36.6
3.5	3.5	-13.5	-13.5	7.5	38.6	40.6	36.6
13.6	-12.5	-15.5	-12.5	-0.5	36.6	21.6	18.6
-5.5	-12.5	-40.5	-25.5	-0.5	38.6	32.6	18.6
-5.5	-24.5	-35.5	-42.5	-12.5	7.5	38.6	32.6
2.5	-24.5	-35.5	-40.5	-22.5	-0.5	21.6	42.6
2.5	-12.5	-22.5	-35.5	-19.5	-12.5	18.6	40.6
3.5	-24.5	-22.5	-40.5	-40.5	-12.5	7.5	38.6
-24.5	-12.5	-25.5	-35.5	-40.5	-25.5	-12.5	21.6

(c)

Fig. 2-2. (a) Bottom-left region of test image Peppers, (b) pixel values of (a), and (c) centered pixels values of (a).

Taking any of the centered vectors as an example, we observe that the adjacent values do not differ significantly, which is called the smoothness. In mathematics, the smoothness is modeled such that the random variables fulfills the wide-sense stationary AR(1) process as shown below

$$X(n) = \rho X(n-1) + \varepsilon(n) \text{ and } |\rho| \in [0,1], \tag{2-3}$$

where $X(n)$ and $X(n-1)$ are two adjacent random variables, ρ is the correlation coefficient with a large absolute value, and $\varepsilon(n)$ is a white noise process with zero mean. The ρ measures the correlation between two adjacent positions: for smooth images, the correlation coefficient is usually high hence the absolute value of ρ approaches 1; for non-smooth images, correlation between adjacent positions are low hence the absolute value of ρ can be as small as 0.

The autocorrelation of the AR(1) process can be thus calculated as follow

$$\begin{aligned} R(i,j) &= \mathrm{E}\left[x(i)x(j)\right], \quad for \quad i > j > 0 \\ &= \mathrm{E}\left\{x(i)\cdot\left[\rho\left[\rho\left[\cdots\left[\rho x(i)+\varepsilon(i+1)\right]\cdots\right]+\varepsilon(j-1)\right]+\varepsilon(j)\right]\right\}, \\ &= \mathrm{E}\begin{bmatrix} \rho^{i-j}\left[x(i)\right]^2+\rho^{i-j-1}\varepsilon(i+1)x(i)+\cdots+ \\ \rho^2\varepsilon(j-2)x(i)+\rho\varepsilon(j-1)x(i)+\varepsilon(j)x(i) \end{bmatrix} \end{aligned} \tag{2-4}$$

where $\varepsilon(i)$ and $x(j)$ are independent since $\{\varepsilon(n)\}$ is a white noise process. Hence the expectation operator can be applied separately on $\varepsilon(i)$ and $x(j)$ such that

$$\begin{aligned} R(i,j) &= \rho^{i-j}\mathrm{E}\left\{\left[x(i)\right]^2\right\}+\rho^{i-j-1}\mathrm{E}\left[\varepsilon(i+1)\right]\mathrm{E}\left[x(i)\right]+\cdots \\ &\quad +\rho^2\mathrm{E}\left[\varepsilon(j-2)\right]\mathrm{E}\left[x(i)\right]+\rho\mathrm{E}\left[\varepsilon(j-1)\right]\mathrm{E}\left[x(i)\right]+\mathrm{E}\left[\varepsilon(j)\right]\mathrm{E}\left[x(i)\right]. \\ &= \rho^{i-j}\sigma_x^2 \end{aligned} \tag{2-5}$$

For arbitrary i and j, the normalized autocorrelation is

$$r(i,j) = \rho^{|i-j|}. \tag{2-6}$$

Hence the autocorrelation matrix of the AR(1) process can be written as

$$\mathbf{R}_{\mathrm{AR}(1)} = \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \rho^2 & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \rho & \cdots & \rho^{N-3} \\ \rho^3 & \rho^2 & \rho & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & \cdots & 1 \end{bmatrix}. \tag{2-7}$$

Numerous experimental works have proved that using the data from smooth regions to fit this autocorrelation matrix, the values of ρ are always of high values, which justifies the effectiveness of modeling the smoothness using the wide-sense stationary AR(1) model with a large value of correlation coefficient ρ.

2.1.2. Karhunen–Loève Transform (KLT)

Modeling the smooth regions of images using AR(1) with a high correlation coefficient is one of the ways to describe how the human "imagines" the unknown smooth regions in a mathematical model that machine understands. The machines can use the "imagine" ability to remove the redundancies of images, hence image compression can be achieved.

One of the metrics for measuring the redundancies is the autocorrelation. If the positions separated by a distance of any value are not correlated, it means no estimation can be carried out hence the redundancies do not exist. Hence the objective becomes: alternating the input signals so that the autocorrelation values (except the center points) of resultant signals are zero or equivalently, the autocorrelation matrix is diagonal. If the AR(1) is used for modeling the input, the correlation coefficient ρ shall equals zero after the alternation of the input signals.

This problem has been tackled by the Karhunen–Loève transform [13-14], named after K. Karhunen and M. Loève, which can remove the redundancies of signals in terms of the decorrelation and energy compaction. The idea of the KLT is simple—rotate the coordinate of the input data set so that the input data concentrate along a single axis, so that the data represented by the rotated coordinate are weakly correlated. The rotation is carried out by multiplying the input random $N\times1$ vector by an $N\times N$ matrix $\mathbf{H}$ as shown below

$$\mathbf{y} = \mathbf{H}\,\mathbf{x}, \tag{2-8}$$

where $\mathbf{y}$ represents the transformed (rotated) random vector. The autocovariance matrix can be obtained by carrying out expectation on the autocovariance of the transformed random vector as shown below

$$\mathbf{C}_\mathbf{y} = \mathrm{E}[\mathbf{y}\mathbf{y}^\mathrm{T}], \tag{2-9}$$

where $\mathbf{C}_\mathbf{y}$ is the autocovariance matrix of transform signal, $\mathrm{E}[\cdot]$ is the expectation operation, and T is the transpose operation. The objective of the KLT is to diagonalize the $\mathbf{C}_\mathbf{y}$ matrix such that only the diagonal terms are non-zero. Let us substitute (2-8) into (2-9), and we obtain

$$\mathbf{C}_\mathbf{y} = \mathrm{E}[(\mathbf{Hx})(\mathbf{Hx})^\mathrm{T}] = \mathrm{E}[\mathbf{H}\,(\mathbf{xx}^\mathrm{T})\,\mathbf{H}^\mathrm{T}] = \mathbf{H}\,\mathrm{E}[\mathbf{xx}^\mathrm{T}]\,\mathbf{H}^\mathrm{T} = \mathbf{H}\,\mathbf{C}_\mathbf{x}\,\mathbf{H}^\mathrm{T}, \tag{2-10}$$

where $\mathbf{C}_\mathbf{x} = \mathrm{E}[\mathbf{xx}^\mathrm{T}]$ is the autocovariance matrix of input signal. The problem is further simplified to the eigenvalue problem—find a proper matrix (transform kernel, rotational matrix) $\mathbf{H}$ containing all eigenvectors such that $\mathbf{C}_\mathbf{x}$ is diagonalized.

It is clear that the KLT is data dependent—each class of data has its optimal transform kernel. In 0, we have reviewed a number of algorithms which adaptively generate the optimal KLTs for data of different characteristics, and we have pointed out the heavy computational load is the burden of its wide-spread applications. Hence it is logical to find a transform kernel that performs well most of time so that there is no need to carry out KLT each time, and fortunately the discrete cosine transform (DCT) fulfills such requirement.

2.1.3. Discrete Cosine Transform (DCT)

The AR(1) signal is a common signal that can be found in natural images and other applications. Its autocovariance matrix $\mathbf{C}_\mathbf{x}$ is represented as shown in (2-7). Hence the optimal kernel for AR(1) signal can be obtained by the diagonalization. By skipping the detailed mathematics involved, we obtain the representation of the n^{th} element of the k^{th} basis vector of the optimal kernel for AR(1) as shown below

$$h_k(n)=\sqrt{\frac{2}{N+\mu_k}}\sin\left[\omega_k\left(n+1-\frac{N+1}{2}\right)+\frac{\pi(k+1)}{2}\right], \tag{2-11}$$

where N is the dimensionality of input signal, μ_k's are the eigenvalues of (2-7) and

$$\mu_k=\frac{1-\rho^2}{1+\rho^2-2\cos\omega_k}, \tag{2-12}$$

where ω_k's are the real non-negative roots of the transcendental equation

$$\tan N\omega=\frac{(1-\rho^2)\sin\omega}{(1+\rho^2)\cos\omega-2\rho}. \tag{2-13}$$

As we have mentioned in 2.1.1, the AR(1) with a large value of ρ can effectively model the smooth regions. Hence let ρ approximate infinitively to 1, we obtain

$$\omega_k=\frac{k\pi}{N}\quad\text{for } k\geq 0\text{, and} \tag{2-14}$$

$$\mu_k=0. \tag{2-15}$$

By substituting (2-14) and (2-15) into (2-11), we obtain

$$h_k(n)=\sqrt{\frac{2}{N}}\cos\left[\frac{(2n+1)k\pi}{2N}\right], \tag{2-16}$$

which is the transform kernel of the DCT-II, and commonly known as the DCT [7]. This is why the DCT is a good kernel for image coding—it is the KLT of the AR(1) signal.

2.1.4. Integer Cosine Transform (ICT)

2.1.4.1. Importance of ICT

As we have mentioned in Subsection 1.1.2, the employment of floating-point DCT introduces the drift problem for video coding. The latest video coding standard the H.264/AVC solves the problem by employing one of the drift-free transforms—the order-4 integer cosine transform (ICT). As the reference software of the emerging video coding standard, the KTA, added an order-16 integer cosine transform to cope with high definition video contents, the details of the construction of the ICT will be explained using the order-16 ICT as an example.

The initial objective of the development of the ICT kernel is to remove the floating-point arithmetic so as to reduce the computational complexity. The removal of the floating-point arithmetic also has a by-product that different devices generates exact the same results if the algorithms are properly designed, which eventually solve the notorious mismatch problem for video coding.

2.1.4.2. Recursive Structure of DCT

Before a direct investigation into the details of the construction of the order-16 ICT, let us first examining the characteristic of its floating-point counterpart—the order-16 DCT Kernel, which can help to reduce the design complexity of integer kernels. One important characteristic is the relationship between the DCT Kernel and its half-length version, that is, the relationship between the DCT Kernels of order-N and order-(N/2), where N is a radix-2 value. According to the recursive spare matrix factorization algorithm [72], the order-N DCT-II kernel can be constructed by the order-(N/2) DCT-II kernel and order-(N/2) DCT-IV kernel as shown below

$$\mathbf{C}_N^{\mathrm{II}} = \frac{1}{\sqrt{2}}\mathbf{B}_N \begin{bmatrix} \hat{\mathbf{C}}_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & \hat{\mathbf{C}}_{\frac{N}{2}}^{\mathrm{IV}}\mathbf{J}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{J}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix}, \tag{2-17}$$

where $\mathbf{C}_N^{\mathrm{II}}$ is the order-N DCT-II kernel, $\mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}}$ is the order-(N/2) DCT-IV kernel, $\mathbf{I}_{\frac{N}{2}}$ is the N×N diagonal matrix with 1's along the southeast direction and 0's in other positions, $\mathbf{J}_{\frac{N}{2}}$ is the N/2×N/2 anti-diagonal matrix which is a horizontal flipped version of $\mathbf{I}_{\frac{N}{2}}$, "^" is the bit-reversal operator which permutes the row vectors into a bit-reversed order, and $\mathbf{B}_N$ is used to permute the bit-reversed ordered vectors back into natural order. The bit-reversal operation maps the indices of all vectors from $\{m(i) \mid m(i) = i_{\mathrm{b}}$ and $i = 0, 1, \ldots, N-1\}$ to $\{n(i) \mid n(i) = bitReverse(i_{\mathrm{b}})$ and $i = 0, 1, \ldots, N-1\}$, where the integer with subscript "b" is in the binary form and the function $bitReverse(\cdot)$ reverses the order of its input binary sequence. For example, when $N = 16$, the index of a vector is mapped from $m = 3_{10} = 0011_2$ to $n = 12 = 1100_2$, since 0011 and 1100 are opposite ordered.

Let us explain physical meaning of (2-17) by temporarily ignoring the minor effects caused by the bit-reversal–related operations (i.e. the $\mathbf{B}_N$ and operator "^") with the help of a few illustrations. The DCT-II kernel is mainly constructed by two matrices—the half-length kernel matrix

$$\begin{bmatrix} \hat{\mathbf{C}}_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & \hat{\mathbf{C}}_{\frac{N}{2}}^{\mathrm{IV}}\mathbf{J}_{\frac{N}{2}} \end{bmatrix}, \text{ and} \tag{2-18}$$

the hybrid-diagonal matrix

$$\begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{J}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix}. \tag{2-19}$$

For the half-length kernel matrix (2-18), its northwest region is filled by the order-(N/2) DCT-II kernel, whereas its southeast region is filled by the horizontally flipped DCT-IV kernel (the anti-diagonal matrix has the effect of horizontal flipping when it is multiplied to the RHS of a matrix). Let us illustrate (2-18) with a more intuitive way as shown in Fig. 2-3(a), i.e. directly flipping the symbol $\hat{\mathbf{C}}_{\frac{N}{2}}^{\mathrm{IV}}$ to indicate the flip operation. The hybrid-diagonal matrix (2-19) can be split into two parts are shown below

$$\begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{J}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ 0 & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{J}_{\frac{N}{2}} & 0 \end{bmatrix}. \tag{2-20}$$

When (2-19) is multiplied to the RHS of (2-18), it is equivalent to multiplying the above two parts to (2-18) independently and then adding the results together. The first part has the effect of changing the sign of all elements of the right-half matrix of (2-18), which is shown by the change from Fig. 2-3(a) to Fig. 2-3(b). Similarly, the second part has the effect of flipping all elements of (2-18) horizontally, which is shown by the change from Fig. 2-3(a) to Fig. 2-3(c). Hence, by adding up Fig. 2-3(b) and Fig. 2-3(c), the result of the multiplication between (2-18) and (2-19) is represented in Fig. 2-3(d). It has explained the physical meaning of (2-17) that the full-length DCT-II kernel is constructed by arranging the half-length DCT-II kernel in the even symmetric structure and the half-length DCT-IV kernel in the odd symmetric structure.

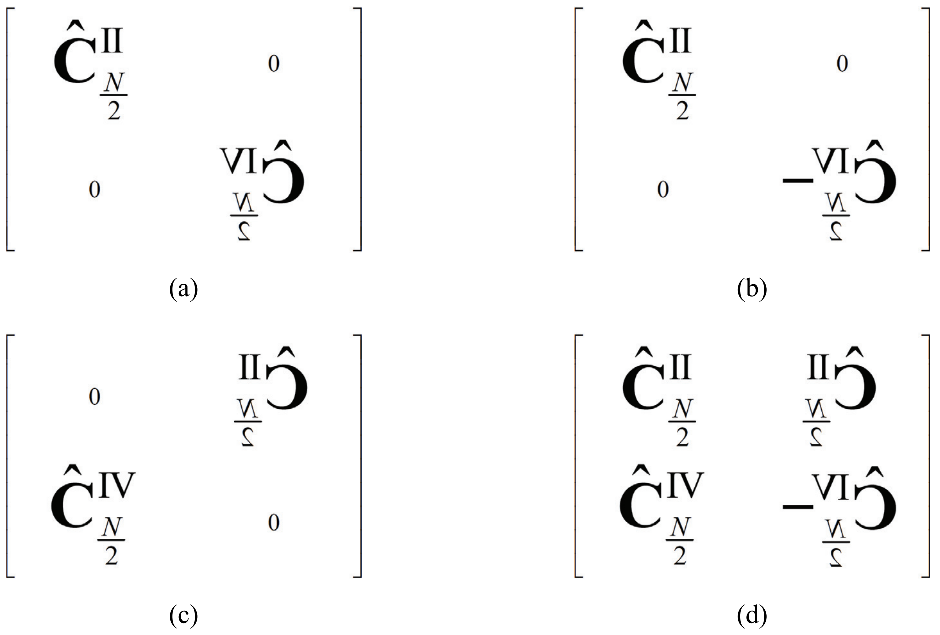


Fig. 2-3. Intuitive representation for flipped sub-matrices[a]: (a) the half-length kernel matrix, (b) the result obtained by multiplying the first part of (2-20) to the RHS of (2-18), (c) the result obtained by multiplying the second part of (2-20) to the RHS of (2-18), and (d) the result obtained by multiplying (2-20) to the RHS of (2-19). Note that the large symbols used in this figure are to imply that the size of the matrices is usually larger than 2×2.

[a] A horizontally flipped matrix symbol represents the result obtained by applying a horizontal flipping operation on the original matrix.

More generally, a radix-2 DCT-II kernel of the size $N\times N$ can be recursively decomposed into lower-order DCT-II and DCT-IV kernels. For example, the order-16 DCT-II kernel can be decomposed into an order-8 DCT-IV kernel, an order-4 DCT-IV kernel, and an order-4 DCT-II kernel. This can significantly simplify the design of ICT kernels. There is no need to start directly from the high-order ICT kernel; instead, with the recursive spare matrix factorization algorithm, one can decide the lowest level to which the recursive algorithm will decompose, and then design the lower-order ICT kernels one by one, so as to reduce the design complexities.

2.1.4.3. *Design of ICT Kernels*

Let us make use of the design of an order-16 ICT kernel as an example, and discuss the design criteria and technical details of designing integer kernels. We first decompose the matrix recursively (the lowest level is order 4) and insert the normalization factors, i.e. 2 and √2. The decomposed order-16 kernel is shown below

$$\mathbf{C}_{16}^{\mathrm{II}} = \mathbf{B}_N \begin{bmatrix} \begin{bmatrix} \hat{\mathbf{C}}_4^{\mathrm{II}}/2 & 0 \\ 0 & \hat{\mathbf{C}}_4^{\mathrm{IV}}\mathbf{J}_4/2 \end{bmatrix} \begin{bmatrix} \mathbf{I}_4 & \mathbf{J}_4 \\ \mathbf{J}_4 & -\mathbf{I}_4 \end{bmatrix} & 0 \\ 0 & \hat{\mathbf{C}}_8^{\mathrm{IV}}\mathbf{J}_8/\sqrt{2} \end{bmatrix} \begin{bmatrix} \mathbf{I}_8 & \mathbf{J}_8 \\ \mathbf{J}_8 & -\mathbf{I}_8 \end{bmatrix}. \qquad (2\text{-}21)$$

where $\mathbf{C}_8^{\mathrm{IV}}$ is the order-8 DCT-IV kernel, $\mathbf{C}_4^{\mathrm{IV}}$ is the order-4 DCT-IV kernel, and $\mathbf{C}_4^{\mathrm{II}}$ is the order-4 DCT-II kernel, and they are all orthonormal kernels. The design of the order-16 ICT is equivalent to designing integer versions of the floating-point kernels $\mathbf{C}_8^{\mathrm{IV}}$, $\mathbf{C}_4^{\mathrm{IV}}$, and $\mathbf{C}_4^{\mathrm{II}}$. The floating-point kernels are shown in (2-22)–(2-24).

The design criteria are:

a) all elements are represented with non-negative integer numbers;

b) the bases (row vectors) are orthogonal to each other;

c) all row vectors should be normalized, either with independent norms or with a same norm for the whole matrix, and the norms are preferred to be radix-2 integers; and

d) the integer matrix after normalization should be as close as the floating-point matrix, which indicates that the relative magnitude among the elements in each basis should be kept.

$$\mathbf{C}_8^{\mathrm{IV}} = \begin{bmatrix} 0.498 & 0.478 & 0.441 & 0.387 & 0.317 & 0.236 & 0.145 & 0.049 \\ 0.478 & 0.317 & 0.049 & -0.236 & -0.441 & -0.498 & -0.387 & -0.145 \\ 0.441 & 0.049 & -0.387 & -0.478 & -0.145 & 0.317 & 0.498 & 0.236 \\ 0.387 & -0.236 & -0.478 & 0.049 & 0.498 & 0.145 & -0.441 & -0.317 \\ 0.317 & -0.441 & -0.145 & 0.498 & -0.049 & -0.478 & 0.236 & 0.387 \\ 0.236 & -0.498 & 0.317 & 0.145 & -0.478 & 0.387 & 0.049 & -0.441 \\ 0.145 & -0.387 & 0.498 & -0.441 & 0.236 & 0.049 & -0.317 & 0.478 \\ 0.049 & -0.145 & 0.236 & -0.317 & 0.387 & -0.441 & 0.478 & -0.498 \end{bmatrix} \qquad (2\text{-}22)$$

$$\mathbf{C}_4^{\mathrm{IV}} = \begin{bmatrix} 0.694 & 0.588 & 0.393 & 0.138 \\ 0.588 & -0.138 & -0.694 & -0.393 \\ 0.393 & -0.694 & 0.138 & 0.588 \\ 0.138 & -0.393 & 0.588 & -0.694 \end{bmatrix} \tag{2-23}$$

$$\mathbf{C}_4^{\mathrm{II}} = \begin{bmatrix} 0.500 & 0.500 & 0.500 & 0.500 \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.500 & -0.500 & -0.500 & 0.500 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix} \tag{2-24}$$

Let us replace the floating-point numbers in (2-22)–(2-24) with variables as shown below

$$\mathbf{IK}_8^{\mathrm{IV}} = \begin{bmatrix} a & b & c & d & e & f & g & h \\ b & e & h & -f & -c & -a & -d & -g \\ c & h & -d & -b & -g & e & a & f \\ d & -f & -b & h & a & g & -c & -e \\ e & -c & -g & a & -h & -b & f & d \\ f & -a & e & g & -b & d & h & -c \\ g & -d & a & -c & f & h & -e & b \\ h & -g & f & -e & d & -c & b & -a \end{bmatrix}, \tag{2-25}$$

$$\mathbf{IK}_4^{\mathrm{IV}} = \begin{bmatrix} i & j & k & l \\ j & -l & -i & -k \\ k & -i & l & j \\ l & -k & j & -i \end{bmatrix}, \text{ and} \tag{2-26}$$

$$\mathbf{IK}_4^{\mathrm{II}} = \begin{bmatrix} \alpha & \alpha & \alpha & \alpha \\ \beta & \gamma & -\gamma & -\beta \\ \alpha & -\alpha & -\alpha & \alpha \\ \gamma & -\beta & \beta & -\gamma \end{bmatrix}, \tag{2-27}$$

where IK is the symbol for an integer cosine kernel with the subscript denoting its order and superscript denoting its family number, and variables a, b, c, d, e, f, g, h, i, j, k, l, α, β, and γ are non-negative integers.

For $\mathbf{IK}_8^{\mathrm{IV}}$, the orthogonality constraint yields the following set of equations

$$\begin{cases} a(b-f)+e(b-c)-d(f+g)+h(c-g)=0 \\ a(c+g)+e(f-g)-d(b+c)+h(b+f)=0\,. \\ a(d+e)-b(f+c)+g(f-c)+h(d-e)=0 \end{cases} \tag{2-28}$$

They should also fulfill inequity corresponding to the floating-point DCT-IV kernel so as to preserve the DCT-IV property

$$a > b > c > d > e > f > g > h > 0. \tag{2-29}$$

A computer search for integers that fulfills (2-28) and (2-29) can be carried out, and each kernel can be denoted as $\mathrm{IK}_8^{\mathrm{IV}}(a,b,c,d,e,f,g,h)$. Since each row has the same norm

$$s_1 = (a^2 + b^2 + c^2 + d^2 + e^2 + f^2 + g^2 + h^2)^{½}, \qquad (2\text{-}30)$$

the relationship between the floating-point and integer kernels can be denoted as

$$\mathbf{C}_8^{\mathrm{IV}} = {}^1\!\!/_{s_1}\,\mathbf{IK}_8^{\mathrm{IV}}. \qquad (2\text{-}31)$$

For $\mathbf{IK}_4^{\mathrm{IV}}$, the orthogonality constraint yields only one equations as shown below

$$j\,(\,i - l\,) - k\,(\,i + l\,) = 0, \qquad (2\text{-}32)$$

They should also fulfill inequity corresponding to the DCT-IV kernel so as to preserve the DCT-IV property

$$i > j > k > l > 0. \qquad (2\text{-}33)$$

Another computer search can be carried out so that proper integers can be found. Each kernel can be denoted as $\mathrm{IK}_4^{\mathrm{IV}}(i, j, k, l)$. Similar to the order-8 DCT-IV kernel, the order-4 DCT-IV kernel also has equal norms for all rows

$$s_2 = (i^2 + j^2 + k^2 + l^2)^{½}, \qquad (2\text{-}34)$$

hence the floating-point kernel can be obtained by normalizing the integer DCT-IV kernel by s_2:

$$\mathbf{C}_4^{\mathrm{IV}} = {}^1\!\!/_{s_2}\,\mathbf{IK}_4^{\mathrm{IV}}. \qquad (2\text{-}35)$$

For $\mathbf{IK}_4^{\mathrm{II}}$, it is a natively orthogonal kernel. However, its norms of row vectors are not necessarily equal. If we assume a single norm, the constraint is shown as below

$$s_3 = 2\alpha = [2(\beta^2 + \gamma^2)]^{½}. \qquad (2\text{-}36)$$

Similarly, the integer kernel can be normalized to

$$\mathbf{C}_4^{\mathrm{II}} = {}^1\!\!/_{s_3}\,\mathbf{IK}_4^{\mathrm{II}}. \qquad (2\text{-}37)$$

In addition, the inequity relationship should also be fulfilled as shown below

$$\beta > \alpha > \gamma > 0. \qquad (2\text{-}38)$$

A third computer search should be carried out to find the proper integers, and each kernel can be denoted as $\mathrm{IK}_4^{\mathrm{II}}(\alpha, \beta, \gamma)$.

With the above newly obtained kernels, the design of a set of order-16 ICT kernels has been finished, and each kernel can be subsequently denoted as $\mathrm{IK}_{16}^{\mathrm{II}}$ (a, b, c, d, e, f, g, h; i, j, k, l; α, β, γ).

Several points relating to the design of integer kernels are to be stressed:

a) the recursive decomposition is not a must, but it does simplify the design process;

b) the relative relationships among kernel elements have been taken care of by the inequalities (i.e. (2-29), (2-33), and (2-38)), however, to obtain better approximations toward the original DCT Kernel, a tighter constraint (e.g. the ratios among different elements being bounded in a small range) should be applied; and

c) in applications that are very sensible to computational complexity, the DCT-IV kernels can be further decomposed into sparse matrices [72-73] to lower the computational power.

2.1.4.4. Possible Design of Order-4 ICT Kernel for H.264/AVC

The H.264/AVC is the first video coding standard to employ the integer transform. Since the order of the required kernel is 4, the some design procedures described in 2.1.4.3 can be skipped, e.g., the recursive decomposition is not needed since kernel structure is already very simple, and the orthogonality condition is natively fulfilled, so that simple methods such as the up-scaling and rounding [94-95], which will be used in Chapter 3, can be applied to design new kernels.

The up-scaling and rounding method can be described using the following equation

$$\mathbf{IK}_4^{\mathrm{II}} = \left[u \cdot \mathbf{C}_4^{\mathrm{II}} \right], \tag{2-39}$$

where u is a non-negative floating-point number served as the up-scaling factor, and $[\cdot]$ is the round-to-nearest-integer operation.

However, as we have mentioned in point b) of 2.1.4.3, to obtain better approximations toward the DCT Kernel, a tighter constraint on relative relationships among kernel elements should be applied. In detail, we make use of the ratios among matrix elements of the normalized version of a candidate order-4 ICT kernel to measure how far it deviates from the DCT Kernel, and only candidate kernels with small deviations are retained.

This measurement is mathematically defined by the kernel percentage error (KPE). Let us normalize the kernel template of $\mathbf{IK}_4^{\mathrm{II}}$ as shown in (2-27), and the result is shown as below

$$\mathbf{H} = \begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix}, \tag{2-40}$$

where A = 0.5, B = β / $[2(\beta^2+\gamma^2)]^{½}$, C = γ / $[2(\beta^2+\gamma^2)]^{½}$. Coefficients β and γ are the non-negative integers. The two ratios A/B and C/B can be regarded as features extracted from the template of integer kernel. They can be represented in terms of β and γ as shown below

$$\begin{cases} ratio1 = \dfrac{A}{B} = \sqrt{\tfrac{1}{2}\left[1+\left(\tfrac{\gamma}{\beta}\right)^2\right]} \\ ratio2 = \dfrac{C}{B} = \dfrac{\gamma}{\beta} \end{cases}. \tag{2-41}$$

Similarly, we can also derive the ratios for the DCT Kernel

$$\begin{cases} ratio1_{\text{DCT}} = \dfrac{A_{\text{DCT}}}{B_{\text{DCT}}} = \sqrt{\tfrac{1}{2}\left[1+\left(\tfrac{\gamma_{\text{DCT}}}{\beta_{\text{DCT}}}\right)^2\right]} \\ ratio2_{\text{DCT}} = \dfrac{C_{\text{DCT}}}{B_{\text{DCT}}} = \dfrac{\gamma_{\text{DCT}}}{\beta_{\text{DCT}}} \end{cases}. \tag{2-42}$$

The *KPE* can thus be calculated as follow

$$KPE = \frac{\left|ratio1 - ratio1_{\text{DCT}}\right| + \left|ratio2 - ratio2_{\text{DCT}}\right|}{ratio1_{\text{DCT}} + ratio2_{\text{DCT}}}. \tag{2-43}$$

By substituting (2-41) and (2-42) into (2-43), we obtain the *KPE* as shown below

$$KPE = \left| \frac{\sqrt{\tfrac{1}{2}\left[\left(\tfrac{\gamma}{\beta}\right)^2+1\right]} + \tfrac{\gamma}{\beta}}{\sqrt{\tfrac{1}{2}\left[\left(\tfrac{\gamma_{\text{DCT}}}{\beta_{\text{DCT}}}\right)^2+1\right]} + \tfrac{\gamma_{\text{DCT}}}{\beta_{\text{DCT}}}} - 1 \right|, \tag{2-44}$$

where β_{DCT} and γ_{DCT} are the corresponding coefficients of the DCT Kernel. The smaller the *KPE* is, the closer an ICT kernel resembles the DCT Kernel. The *KPE* indirectly measures the similarity between a newly-found ICT kernel and the DCT Kernel, and it will be employed hereafter for designing order-4 ICT kernels.

2.2. Hybrid Video Coding

2.2.1. Overview

The objective of video coding is to encode a video into a small-size representation so that it can be transmitted through a bandwidth-limited channel, and reconstruct the video at the decoder side. The video is comprised of a series of images called frames, which can be denoted as $\{F_t\}$ where t is the time instance and $t \geq 0$. Rather than encoding the video frame by frame, a typical video encoder, which works in analogues to the differential pulse-code modulation (DPCM), mainly codes the differences between frames—called the "residues"—due to the high similarities among consecutive frames. The DPCM model is shown in Fig. 2-4 and codec of the H.264/AVC is shown in Fig. 2-5, and the major DPCM loops are shaded respectively. One of the advantages by employing the DPCM structure is that the encoder side also contains the decoder, which allows the encoder side to simulate the decoding process exactly and thus control the quality of the reconstructed video.

One of the major differences between video coding and image coding is: the former one deals with residual signals whereas the latter one deals with raw image signals. It is well known that the spatial correlation coefficient ρ of images is usually higher than that of the residues. However, the DCT or its integer variation which intends for highly correlated signals was still used as the only or major transform kernel for all the video coding standards, due to its good performance.

Since the resolutions of videos vary from one to another, the frames must be separated into a smaller unit called the "macroblock" which is the basic processing unit in Fig. 2-5, and all macroblocks are coded separately.

Let us explain how a typical video codec works by using the H.264/AVC as an example. A set of symbols and notations are defined for the sake of a clear presentation:

a) the consecutive frames of an original video are denoted by F_0, F_1, …, F_{t-1}, F_t, F_{t+1}, …;

b) each macroblock is denoted by $MB_t(n)$, where n is index of the macroblock;

c) any symbol with a superscript "p" indicates that it is a predicted signal, e.g. $MB_t^{\mathrm{p}}(n)$ is a predicted version of $MB_t(n)$; and

d) any symbol with a superscript "r" indicates that it is a reconstructed signal, e.g. $r_t^{\mathrm{r}}(n)$ is a reconstructed version of $r_t(n)$.

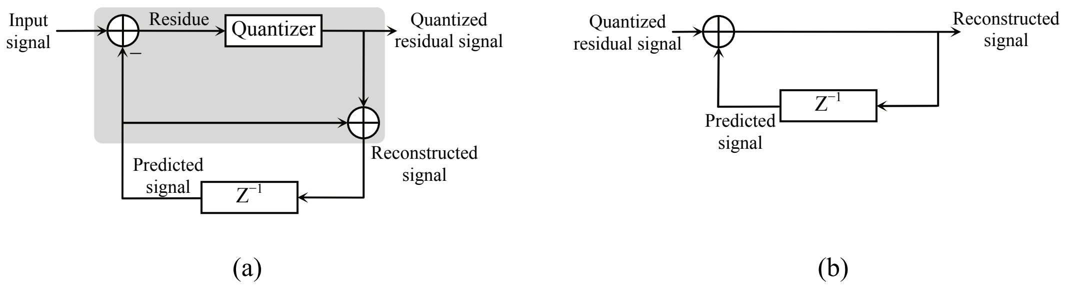
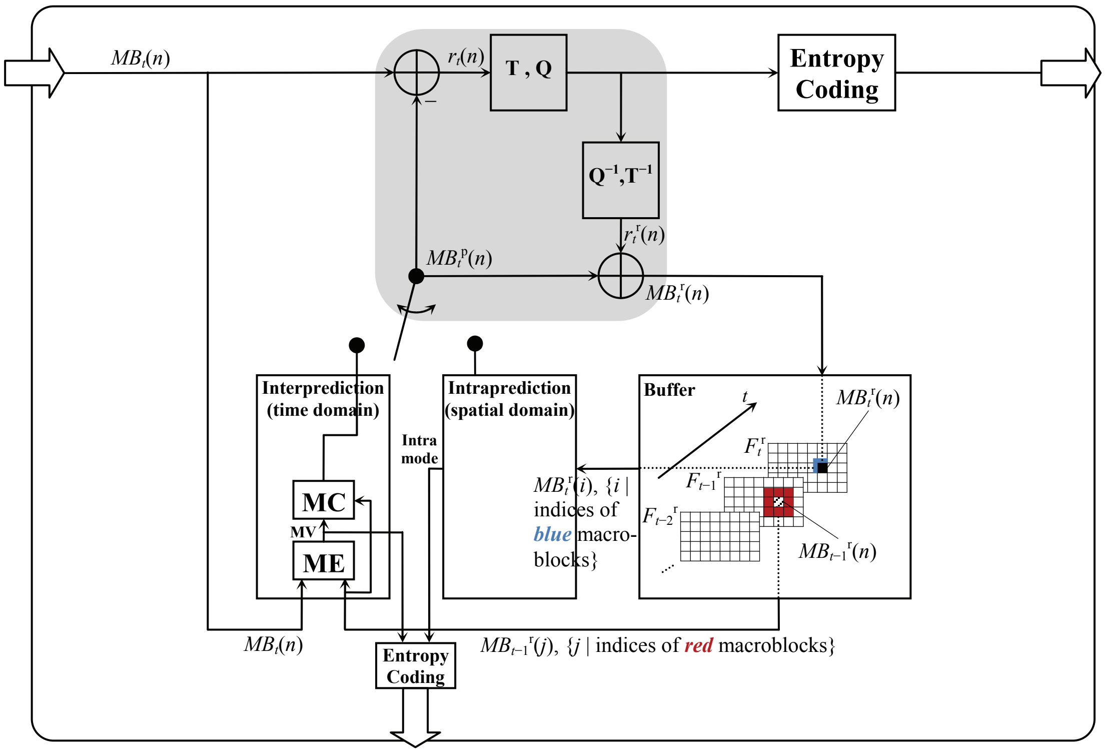
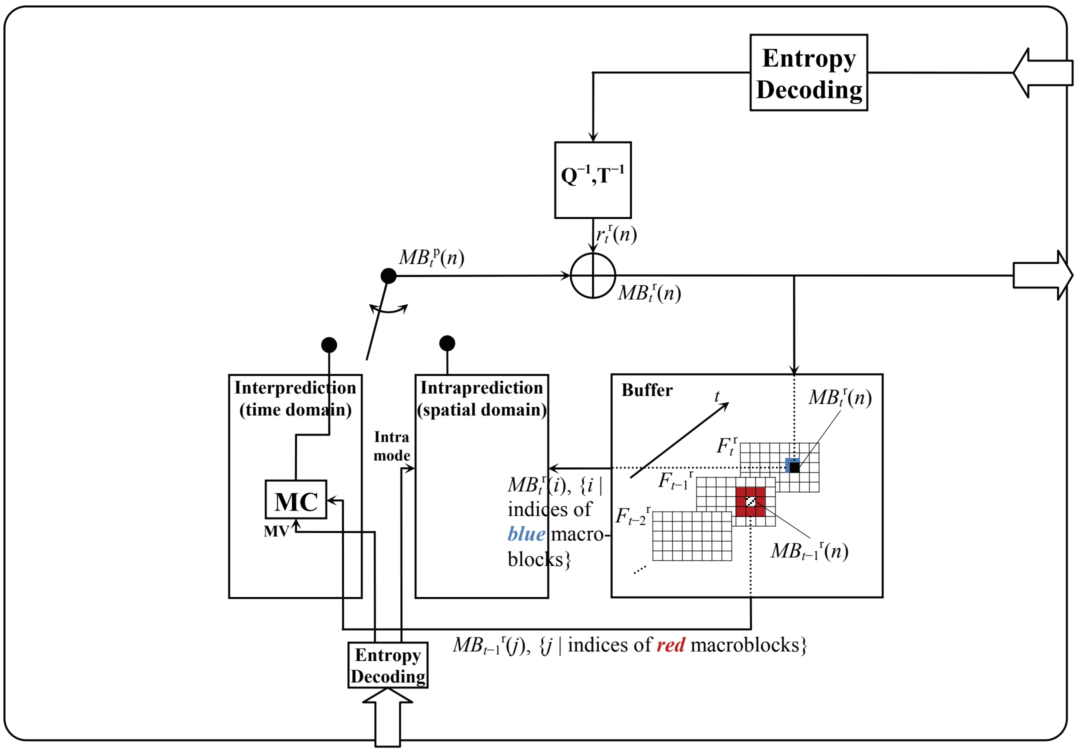


Fig. 2-4. Block diagram for DPCM codec: (a) encoder, and (b) decoder.


MBt(n)
rt(n)
T , Q
Entropy Coding
Q⁻¹,T⁻¹
rt^r(n)
MBt^p(n)
MBt^r(n)
Interprediction (time domain)
Intraprediction (spatial domain)
Buffer
MC
MV
ME
Intra mode
MBt^r(i), {i | indices of blue macro-blocks}
t
Ft^r
Ft−1^r
Ft−2^r
MBt^r(n)
MBt−1^r(n)
MBt(n)
MBt−1^r(j), {j | indices of red macroblocks}
Entropy Coding

(a)



Entropy Decoding
Q^{-1},T^{-1}
$r_t^r(n)$
$MB_t^p(n)$
$MB_t^r(n)$
Interprediction (time domain)
Intraprediction (spatial domain)
Buffer
Intra mode
MC
MV
t
F_t^r
F_{t-1}^r
F_{t-2}^r
$MB_t^r(n)$
$MB_{t-1}^r(n)$
$MB_t^r(i)$, {i | indices of *blue* macro-blocks}
$MB_{t-1}^r(j)$, {j | indices of *red* macroblocks}
Entropy Decoding

(b)

Fig. 2-5. Block diagram for H.264/AVC codec: (a) encoder, and (b) decoder.

When the encoding starts, the input signal $MB_t(n)$ subtract the predicted signal $MB_t^p(n)$, and then the residual signal $r_t(n)$ is obtained. Since the predicted signal resembles the input signal, the redundancy in the time domain has been partially removed. However, since the prediction cannot be ideal, the redundancy in the spatial domain still exists. Hence the transform coding "T" and the quantization "Q" need to be carried out to compressed the data. On one hand, the transformed-and-quantized data need to be losslessly coded with entropy encoder for output; on the other hand, they should be decoded via dequantization "Q^{-1}" and inverse transform coding "T^{-1}" instantly in order to simulate what the decoder side can obtain. The thus obtained reconstructed signals $MB_t^r(n)$ are stored in the buffer and will be used with other reconstructed signals to construct predict signals for later usage.

There are two kinds of prediction in the H.264/AVC, namely, the inter-prediction and the intra-prediction. Suppose the time instance is t, and the predicted signal $MB_t^p(n)$ of the current frame F_t^r is needed. For inter-prediction, inter-frame correlation is exploited to reduce the temporal redundancy. The inter-prediction will refer to the previous frame F_{t-1}^r, and firstly define a search range, e.g. the red-shaded region with the center macroblock $MB_{t-1}^r(n)$, for the construction of the $MB_t^p(n)$. Secondly, the motion estimation (ME) process together with the motion compensation process will search for a macroblock within the search range input that resembles $MB_t(n)$ the most as the predicted signal $MB_t^p(n)$. And finally, the location difference between the obtained macroblock and the $MB_{t-1}^r(n)$ will be store as a motion vector

(MV) for output. For intra-prediction, intra-frame correlation is exploited to reduce the spatial redundancy. The intra-prediction is carried out inside the frame in which the unknown signal $MB_t^{\mathrm{p}}(n)$ resides. Firstly, the reconstructed upper, upper-left and left macroblocks $MB_t^{\mathrm{r}}(\alpha)$, $MB_t^{\mathrm{r}}(\beta)$ and $MB_t^{\mathrm{r}}(\gamma)$ are needed to be available. Secondly, the pixels that are at the boundary of the $MB_t^{\mathrm{p}}(n)$ (blue-shaded pixels) are used for intra-prediction and all intra-modes are tried one by one to obtain the best predicted signal $MB_t^{\mathrm{p}}(n)$. And finally, the selected intra-mode will be coded for output.

For the decoder, it can be constructed by taking the decoder-in-encoder out and changing the signal flow direction of MV, intra-mode, and entropy-coded data from output to input. The flowchart of H.264/AVC decoder is shown in Fig. 2-5(b).

2.2.2. Drift Problem

In order to ensure that the reconstructed signals at the decoder side are exactly the same as those in the buffer of the encoder side, the corresponding functional blocks must be the same. However, the implementations of the transform coding process, i.e. the floating-point DCT, usually vary from one to another, which leads to the serious drift problem, as we have mentioned in Subsection 1.1.2. The drift problem can significantly lower the visual quality of reconstructed videos: the error accumulates as more and more inter-predictive frames are generated. The drift is caused by the mismatch of between the implementations of the inverse DCT (IDCT) at the encoder side and the decoder side. Intuitively, the IDCT shall be implemented as closely as the arithmetic that the formula of the IDCT specifies. However, in practical implementations, there are mainly two factors leading to the implementations of the IDCT deviate from the ideal one: first, every bit is expensive in some applications, e.g. the embedded systems running on batteries, only a small number of bits can be afforded; second, different fast IDCT algorithms produce slightly different results. To tackle this problem, various approaches were proposed, either partially controlling or totally eliminating the mismatch error. In the early days, the video people tended to follow the first approach. They argued that totally eliminating the mismatch error requires a stringent definition on the implementation of the IDCT and may also hinder creative ideas of future implementations [61], hence agreed to bound the mismatch error in a reasonable small range as defined in the IEEE standard 1180 and later the MPEG-C Part 1, and carry out constant intra-refresh to discard the accumulated error. However, the mismatch was not totally eliminated for all video coding standards. As the technology develops, the computational power grows significantly, which means that there is a higher degree of flexibility in choosing the implementation structure of the IDCT. Meanwhile, the demand for better visual quality implementation is ever increasing, so that finer quantization for video coding is more popular. It was observed that the drift is more severe at finer quantization [65, 96] even though measures suppressing the mismatch such as the oddification and LSB toggling are employed. Hence, the intrinsic drift-free transforms are demanding and therefore the H.264/AVC employs the integer transforms to tackle this problem.

2.2.3. Quantization Error

The quantization is a general operation, which is carried out by a quantizer, uses less number of bits to represent the quantizer input. It is widely used in data compression applications, e.g. image coding, video coding and etc., since it allows the control on bitrate of output by tuning its quantization step. From the information theory point of view, it is a lossy operation hence the signals bypass the quantizer will have a reduction in their information contained.

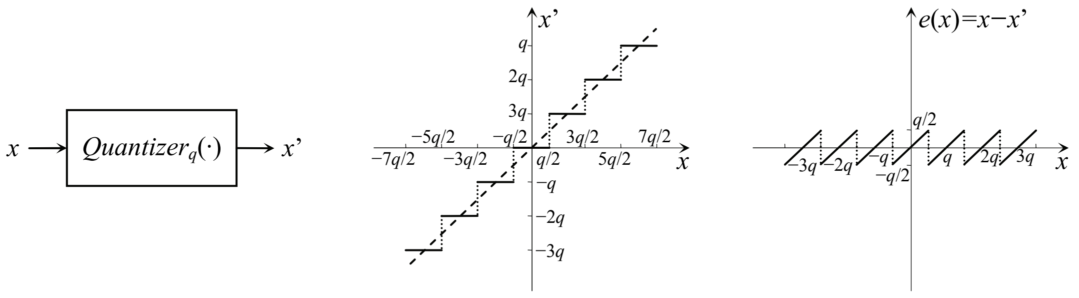
Let us explain how a uniform quantizer works. The quantization model is shown in Fig. 2-6(a), in which x denotes the input signal, x' denotes the quantized signal, and q denotes the quantization step. As the input signal bypass the quantizer, the output signal is rounded to nearest number that equals an integer multiples of q. In mathematics, it can be written as shown below

$$Quantizer_q(x) = x', \tag{2-45}$$

$$x = nq + e, \tag{2-46}$$

$$x' = nq, \text{ for } n \in \mathbb{Z}, \tag{2-47}$$

where $e \in ½[-q, q]$. The output-input relationship of quantization process has been illustrated in Fig. 2-6(b), in which the solid line represents the mapping caused by the quantizer whereas the dash line is the extreme mapping for $q = 0$.



(a) (b) (c)

Fig. 2-6. (a) Quantization model, (b) output-input relationship, and (c) error-input relationship.

Let us calculate the error caused by quantization. Let us define the quantization error as shown below

$$e(x) = x - x', \tag{2-48}$$

and the relationship between the error and the input signal has been depicted in Fig. 2-6(c). Hence the average power of error can be formulated as

$$\int p(x)\cdot e^2(x)\,dx, \tag{2-49}$$

where $p(x)$ is the probability density function (pdf) of the input signal. For the sake of the simplicity, we assume that the input signal is uniformly distributed. By skipping the calculation steps, we obtain the average power equals $q^2/12$, which is well-known in the field.

In video coding applications, the quantizer input is the transform coded signal which fulfills the Laplacian distribution hence the pdf has to be changed accordingly. However, our experimental works show that by using the pdf obtained from the real data, the value of the average power thus calculated is in the same order with respect to the value of the average power by assuming a uniform distribution. This implies that the uniformly distributed model is effective under certain circumstances.

2.3. Transform Coding and Quantization in H.264/AVC

2.3.1. Integer Transform and Scaling

As we have mentioned in Subsection 2.2.2, all video coding standards prior to the H.264/AVC suffer from the drift problem. Various algorithms that can totally eliminate the drift problem have been reviewed in Subsection 1.1.2. Amongst those algorithms, the integer cosine transform (ICT) was adopted into the H.264/AVC due to its low complexity, resource saving, ease for understanding, and etc.

For the H.264/AVC baseline profile, only order-4 ICT is employed, which is different for previous standards in which order-8 DCT is employed. This is mainly because the H.264/AVC uses much finer block segmentations comparing to the previous standards, which allows the motion estimation unit to be as small as 4×4 so that features within small blocks can be well captured. Let us review how the H.264/AVC designs the transform coding and quantization processes by employing the integer transform. The order-4 integer transform kernel is given as follow

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \tag{2-50}$$

which can be obtained by the up-scaling and rounding method as we have mentioned in 2.1.4.4. This is the simplest DCT-like kernel that can be obtained using this approach. Its normalization matrix can subsequently be obtained as shown below

$$\mathbf{S} = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/\sqrt{10} & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/\sqrt{10} \end{bmatrix}, \tag{2-51}$$

hence the orthonormal kernel can be represented as **T** = **SH**. By applying **T** to the horizontal and vertical directions of a 4×4 input matrix **x**, we obtain the transformed matrix as shown below

$$\mathbf{X} = \mathbf{T}\mathbf{x}\mathbf{T}^{\mathrm{T}} = (\mathbf{SH})\,\mathbf{x}\,(\mathbf{SH})^{\mathrm{T}} = \mathbf{S}(\mathbf{H}\mathbf{x}\mathbf{H}^{\mathrm{T}})\mathbf{S}^{\mathrm{T}} = \mathbf{S}\mathbf{C}\mathbf{S}^{\mathrm{T}}, \tag{2-52}$$

where

$$\mathbf{C} = \mathbf{H}\mathbf{x}\mathbf{H}^{\mathrm{T}} \tag{2-53}$$

is only transformed by the integer kernel hence we define it as the integer-transformed matrix. Since **S** is a diagonal matrix, (2-52) can be further simplified to

$$\mathbf{X} = \mathbf{C} \otimes \mathbf{S}_{\mathrm{f}}, \tag{2-54}$$

where ⊗ denotes the term-by-term multiplication operation between **C** and $\mathbf{S}_{\mathrm{f}}$, and

$$\mathbf{S}_{\mathrm{f}} = \begin{bmatrix} 1/2 \\ 1/\sqrt{10} \\ 1/2 \\ 1/\sqrt{10} \end{bmatrix} \begin{bmatrix} 1/2 & 1/\sqrt{10} & 1/2 & 1/\sqrt{10} \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \\ 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \end{bmatrix} \tag{2-55}$$

in which three different values of normalization factors exist.

The quantization can be subsequently carried out on the transformed matrix **X**. Suppose the quantization step is denoted as q, hence the quantized-and-transformed coefficient at position (i,j) can be represented as

$$X_{\mathrm{q}}(i,j) = X(i,j)\,/\,q = C(i,j) \times S_{\mathrm{f}}(i,j)\,/\,q. \tag{2-56}$$

Since the multiplication of $S_{\mathrm{f}}(i,j)$ and the division of q involve floating-point arithmetic that causes mismatch, they should not be carried out directly. Instead, terms $S_{\mathrm{f}}(i,j)$ and q should be grouped together, and implemented without floating-point arithmetic. The H.264/AVC uses the "dyadic rational (fraction) approximation" to tackle this problem, and let us explain how it works. Suppose an integer m is multiplied by a floating-point number s, and we want to remove the floating-point arithmetic. The floating-point number s can be approximated by $k/2^n$ as shown below

$$s \approx k\,/\,2^n, \tag{2-57}$$

where k is an integer and n is a natural number. Hence the floating-point multiplication can be changed to the integer multiplication of m and k followed by a division of 2^n

$$m \times s \rightarrow m \times k\,/\,2^n. \tag{2-58}$$

And the division by 2^n is easily implemented by an n-bit right-shifting operation in binary machines as shown below

$$m \times s \rightarrow (m \times k) >> n, \tag{2-59}$$

where ">>" denotes the right-shifting operation. Hence, we can approximate the floating-point number $S_{\mathrm{f}}(i,j)\,/\,q$ with the dyadic approximation as shown below

$$S_{\mathrm{f}}(i,j)\,/\,q \approx k(i,j)\,/\,2^n, \tag{2-60}$$

and we call $k(i,j)$ the multiplication factor (MF). Hence (2-56) can be modified into

$$X_q(i,j) \approx C(i,j) \times k(i,j) / 2^n = [C(i,j) \times k(i,j)] >> n. \quad (2\text{-}61)$$

The transform coding and quantization are thus equivalently regrouped into two machine-friendly operations:

a) the 2-d integer transform, as shown in (2-53), and

b) the scaling, as shown in (2-61).

As we have mentioned in Subsection 2.2.3, quantization can control the bitrate and the quality of reconstructed videos by tuning the quantization step (QStep). For the H.264/AVC, a uniform quantizer is employed and various QSteps are allowed for different requirements of visual qualities from finest to coarsest. A total number of 52 QSteps are allowed, which arises a problem that a large table has to be prepared for

$$52 \text{ QSteps} \times 3 \text{ MFs/QSteps} = 156 \text{ MFs}. \quad (2\text{-}62)$$

Note that for one particular quantization step q, 3 different values of MF exist by referring back (2-55). To solve this problem, the design of the H.264/AVC incorporates the idea of periodicity into selection of QSteps. TABLE 2-1 has shown the allowed values of QSteps and corresponding quantization parameters (QPs). As defined, for each increase of 6 in *QP*, the *QStep* doubles. Or equivalently, the increase between two successive *QP*s equals $2^{1/6}-1 \approx 12.5\%$. For example, when *QP* is 4, *QStep* is 1; when *QP* is increased to 10, *QStep* is doubled to $1 \times 2 = 2$.

TABLE 2-1
QUANTIZATION STEPS AND CORRESPONDING QUANTIZATION PARAMETERS

QP	QStep	QP	QStep		QP	QStep
0	0.6250	6	1.250		48	160
1	0.6875	7	1.375		49	176
2	0.8125	8	1.625	…	50	208
3	0.8750	9	1.750		51	224
4	1.0000	10	2.000			
5	1.1250	11	2.250			

With the above design, it is only needed to calculate and store the *MF*s for *QP*s range from 0 to 5, with a total number reduced to

$$6 \text{ QSteps} \times 3 \text{ MFs/QSteps} = 18 \text{ MFs}. \quad (2\text{-}63)$$

For *QP* larger than 5, the scaling using *corresponding QP* within the range 0 to 5 can be carried out first, and then followed by a right-shifting of n bits depending on the *distance between the real QP* and *corresponding QP*. In mathematics,

$$\textit{corresponding QP} = \textit{rem}(QP,6), \text{ and} \quad (2\text{-}64)$$

$$\textit{distance between the real QP} = \textit{floor}(QP/6), \quad (2\text{-}65)$$

where the former one finds the remainder of *QP*/6 and the latter one rounds to result of *QP*/6 to integer towards 0. To conclude, the scaling after incorporating the periodicity into the selection of QSteps is achieved as shown below

$$X_q(i,j,QP) = \left\{\left[C(i,j)\cdot k\left(i,j,rem(QP,6)\right)\right] >> n\right\} >> floor(QP/6), \tag{2-66}$$

where the reference software of the H.264/AVC recommends $n = 15$. Taking care of the sign of $C(i,j)$ and rounding issues, we obtain

$$\begin{cases} \left|X_q(i,j,QP)\right| = \left[\left|C(i,j)\right|\cdot k\left(i,j,rem(QP,6)\right) + f\right] >> qbits \\ sign\left(X_q(i,j,QP)\right) = sign\left(C(i,j)\right) \end{cases}, \tag{2-67}$$

where $qbits = 15 + floor(QP/6)$, and $f = 2^{qbits}/6$ for inter-prediction and $f = 2^{qbits}/3$ for intra-prediction. The values of MFs can be calculated by

$$k(i,j,QP) = rounding(\ 2^{15} \times S_f(i,j)\ /\ q(QP)\ ) \text{ for } QP \in [0,5], \tag{2-68}$$

where *rounding*(·) is the round-to-nearest-integer operation. The values of MFs for QP from 0 to 51 calculated from (2-68) are shown in TABLE 2-2. Note that the values in brackets are the modified values adapted to the decoder according to (2-71), since the standard only defines the decoder side.

TABLE 2-2
MULTIPLICATION FACTORS $k(i,j,rem(QP,6))$ FOR QP RANGES FROM 0 TO 51

***rem*(*QP*,6)**	**{(*i*,*j*) | 0 ≤ *i* ≤ 3 or 0 ≤ *j* ≤ 3}**		
	(0,0) (2,0) (2,2) (0,2)	**(1,1) (1,3) (3,1) (3,3)**	**Others**
0	13107	5243	8290 (8066)
1	11916	4766 (4660)	7536 (7490)
2	10082	4033 (4194)	6377 (6554)
3	9362	3745 (3647)	5921 (5825)
4	8192	3277 (3355)	5181 (5243)
5	7282	2913 (2893)	4605 (4559)

Similarly, the dequantization and inverse transform are also regrouped into two machine-friendly operations:

a) the inverse scaling (rescaling), as shown in (2-72), and

b) the 2-d integer inverse transform, as shown in (2-73).

The rescaling factor (RF) can be obtained by combining the dequantization [$\times q(QP)$] and the normalization of inverse transform [$\times S_i(i,j)$] together as shown below

$$k'(i,j,QP) = rounding(\ 2^{6} \times S_i(i,j) \times q(QP)\ ) \text{ for } QP \in [0,5]. \tag{2-69}$$

The values of RFs for QP from 0 to 51 calculated from (2-69) have been shown in TABLE 2-3, and the values in the brackets (if any) are the finalized values defined in the H.264/AVC standard.

TABLE 2-3

RESCALING FACTORS $k'(i, j, rem(QP,6))$ FOR QP RANGES FROM 0 TO 51

***rem*(*QP*,6)**	**{(*i,j*) | 0 ≤ *i* ≤ 3 or 0 ≤ *j* ≤ 3}**		
	(0,0) (2,0) (2,2) (0,2)	**(1,1) (1,3) (3,1) (3,3)**	**Others**
0	10	16	13
1	11	18	14
2	13	21 (20)	16
3	14	22 (23)	18
4	16	26 (25)	20
5	18	29	23

The relationship between MF and RF can be obtained by the multiplication of (2-68) and (2-69) as show below

$$k(i, j, QP) \times k'(i, j, QP) = 2^{21} \times S_{\mathrm{f}}(i,j) \times S_{\mathrm{i}}(i,j), \tag{2-70}$$

hence the modified values in TABLE 2-2 can be calculated by

$$k(i, j, QP) = rounding(\, 2^{21} \times S_{\mathrm{f}}(i,j) \times S_{\mathrm{i}}(i,j) \,/\, k'(i, j, QP)\,). \tag{2-71}$$

The rescaling can be carried out as shown below

$$\begin{cases} \left|C_{\mathrm{r}}(i,j)\right| = \left[\left|X_{\mathrm{q}}(i,j,QP)\right| \cdot k'\left(i,j,rem(QP,6)\right)\right] >> deqbits \\ sign\left(C_{\mathrm{r}}(i,j)\right) = sign\left(X_{\mathrm{q}}(i,j,QP)\right) \end{cases}. \tag{2-72}$$

Finally, the inverse integer transform is carried out as shown below

$$\mathbf{C} = \mathbf{H}_{\mathrm{i}}^{\mathrm{T}}\, \mathbf{x}\, \mathbf{H}_{\mathrm{i}}, \tag{2-73}$$

where

$$\mathbf{H}_{\mathrm{i}}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & ½ & -½ & -1 \\ 1 & -1 & -1 & 1 \\ ½ & -1 & 1 & -½ \end{bmatrix}. \tag{2-74}$$

is obtained by dividing the 2nd and 4th basic vectors of (2-50) by 2 to avoid the increase of dynamic range at the decoder side.

2.3.2. Nonorthogonality Error

The demand for large-size (e.g. 16×16) integer transform kernels is ever increasing due to the explosive increase of resolution of videos. In 2.1.4.3, we reviewed how order-16 ICT kernels can be found through a computer search, with the constraints of the orthogonality conditions (2-28), (2-30), (2-32), (2-34) and (2-36), and inequalities (2-29), (2-33) and (2-38). However, these constraints are too strong for those low-complexity kernels with small integer values. The kernel designs violating the constraints with varying degrees have been reviewed in Subsection 1.1.3. The basic idea is to compromise the similarity to the DCT Kernel with the easy-to-implement property, and is practically achieved by relaxing the condition of orthogonality of transform kernel in a controllable manner which roughly ensures the orthogonality. The error analysis [74] showed that the well-controlled nonorthogonality noise is approximately negligible as compared to the quantization noise.

Let us review the error analysis of nonorthogonality presented in [74]. Although there are some defects in the original analysis and we thus point them out in Chapter 4, the framework is correct and those defects do not affect the overall justifications to the results of the analysis. The derivation starts by assuming an input signal as a 1-d, zero-mean, unit-variance first-order Markov Process. It is denoted as an $N\times 1$ vector $\mathbf{x}$, and the correlation between two adjacent elements is ρ. The nonorthogonal transform coding process is applied to the input signal by using the nonorthogonal transform kernel $\mathbf{T}$. Note that the perfect reconstruction is no longer preserved after the forward transform (2-75) and inverse transform (2-76), and the reconstruction error is shown in (2-77).

$$\boldsymbol{\theta} = \mathbf{Tx} \tag{2-75}$$

$$\mathbf{y} = \mathbf{T}^{\mathrm{T}}\boldsymbol{\theta} \tag{2-76}$$

$$\mathbf{y} - \mathbf{x} = \mathbf{T}^{\mathrm{T}}\boldsymbol{\theta} - \mathbf{x} = (\mathbf{T}^{\mathrm{T}}\mathbf{T} - \mathbf{I})\,\mathbf{x} = \mathbf{E}_{\mathrm{r}}\,\mathbf{x} \tag{2-77}$$

where $\boldsymbol{\theta}$ is the vector of transform coefficients, superscript T is the matrix transpose operation, $\mathbf{y}$ is the reconstructed vector, $\mathbf{I}$ is the identity matrix and

$$\mathbf{E}_{\mathrm{r}} = \mathbf{T}^{\mathrm{T}}\mathbf{T} - \mathbf{I}. \tag{2-78}$$

Then the average variance of the reconstruction error without the consideration of quantization can be formulated as

$$\begin{aligned}\sigma_{\mathrm{r}0}^2 &= \mathrm{E}\left[\tfrac{1}{N}\sum_{k=0}^{N-1}\left[x(k)-y(k)\right]^2\right]\\ &= \tfrac{1}{N}\mathrm{E}\left[(\mathbf{y}-\mathbf{x})^{\mathrm{T}}(\mathbf{y}-\mathbf{x})\right]\\ &= \tfrac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}M(k,j)R(k,j)\end{aligned} \tag{2-79}$$

where E is the expectation operator, $M(k,j)$ is the element of $\mathbf{M} = \mathbf{E}_{\mathrm{r}}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}$ at the position (k,j), and $R(k,j)$ is the element of the autocorrelation matrix of $\mathbf{x}$ at the position (k,j). By assuming the input signal as wide-sense stationary, (2-79) can be further simplified to

$$\sigma_{\mathrm{r}0}^2 = \tfrac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}M(k,j)R(j-k). \tag{2-80}$$

The relationship that the autocorrelation is less than or equal to the variance was used by [74], hence [74] obtained an upper bound of $\sigma_{\mathrm{r}0}^2$ for a given NICT as shown below

$$\sigma_{\mathrm{r}0}^2 \le \tfrac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}M(k,j)\cdot\sigma_x^2 = \tfrac{1}{N}\sigma_x^2\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}M(k,j). \tag{2-81}$$

Note that in the real coding environment, the transform coding is always followed by the quantization. Hence, considering the noise introduced by the nonorthogonal transform coding alone is not sufficiently to model the real scenario with the existence of quantization. The relationship between the reconstruction error σ_{r}^2 and the quantization error σ_{q}^2 is also examined by the authors. Recall that the quantized-and-dequantized version of $\boldsymbol{\theta}$ is denoted as

u, the reconstructed vector with the consideration of quantization is denoted as $\mathbf{y}_\mathrm{r}$, and the quantization error $\boldsymbol{\theta} - \mathbf{u}$ is denoted as **q**. Hence the average variance of the reconstruction can be formulated as

$$\begin{aligned}\sigma_\mathrm{r}^2 &= \mathrm{E}\left[\tfrac{1}{N}\sum_{k=0}^{N-1}\left[x(k) - y_\mathrm{r}(k)\right]^2\right] \\ &= \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{T}^\mathrm{T}\mathbf{q} - \mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right)^\mathrm{T}\left(\mathbf{T}^\mathrm{T}\mathbf{q} - \mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right)\right] \\ &= \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{T}\mathbf{T}^\mathrm{T}\mathbf{q} - 2\mathbf{q}^\mathrm{T}\mathbf{T}\mathbf{T}_\mathrm{er}\boldsymbol{\theta} + \left(\mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right)^\mathrm{T}\mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right]\end{aligned} \qquad (2\text{-}82)$$

where $\mathbf{T}_\mathrm{er} = \mathbf{T}^\mathrm{T} - \mathbf{T}^{-1}$. By replacing $\mathbf{T}\mathbf{T}^\mathrm{T}$ with $\mathbf{E}_\mathrm{r} + \mathbf{I}$, [74] obtained

$$\begin{aligned}\sigma_\mathrm{r}^2 &= \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\left(\mathbf{E}_\mathrm{r} + \mathbf{I}\right)\mathbf{q} - 2\mathbf{q}^\mathrm{T}\mathbf{T}\mathbf{T}_\mathrm{er}\boldsymbol{\theta} + \left(\mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right)^\mathrm{T}\mathbf{T}_\mathrm{er}\boldsymbol{\theta}\right] \\ &= \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{q}\right] + \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{y} - \mathbf{x}\right)^\mathrm{T}\left(\mathbf{y} - \mathbf{x}\right)\right] + \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{E}_\mathrm{r}\mathbf{q}\right] - \tfrac{2}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{E}_\mathrm{r}\boldsymbol{\theta}\right]. \\ &= \sigma_\mathrm{q}^2 + \sigma_\mathrm{r0}^2 + \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{E}_\mathrm{r}\mathbf{q}\right] - \tfrac{2}{N}\mathrm{E}\left[\mathbf{q}^\mathrm{T}\mathbf{E}_\mathrm{r}\boldsymbol{\theta}\right]\end{aligned} \qquad (2\text{-}83)$$

Reference [74] argued that the third and fourth terms in (2-83) can be eliminated because "the autocorrelation of q and the crosscorrelation of q and θ are both zero as shown by Widrow *et al.* [97]". Hence the relationship is simplified to

$$\sigma_\mathrm{r}^2 = \sigma_\mathrm{q}^2 + \sigma_\mathrm{r0}^2 \qquad (2\text{-}84)$$

which means that the reconstruction error is simply the addition of the quantization noise and the nonorthogonal noise. The upper bounds of various NICT kernels were then evaluated by assuming that the variance of the input signal $\sigma_x^2 = 1$. Subsequently experimental works carried out for a set of nonorthogonal order-16 ICT kernels showed that the nonorthogonality error is negligible compared to the quantization error.

2.3.3. Dyadic Approximation Error

In an ideal hybrid video codec model, the only error source is the quantization process. However, practical implementations can introduce other errors, for example, the error introduced by the nonorthogonal transform kernels. Besides the quantization error and the nonorthogonality error, the H.264/AVC also suffers from a new kind of error called the "dyadic approximation error" due to the integerization of transform coding. Let us use Fig. 2-7 to explain the source of this error and the way to model it. Recall that the main idea of integerization of the transform coding is to split the transform into the integer transform and the normalization (see Fig. 2-7(a) and Fig. 2-7(b)), and then combine the normalization and quantization together with a new name "post-scaling" (see Fig. 2-7(b) and Fig. 2-7(c)). In spite of different forms the processes in Fig. 2-7(a), Fig. 2-7(b), and Fig. 2-7(c), they are totally equivalent, and do not suffer from any error except the nonorthogonality and the quantization. The multiplication between an integer m and a floating-point number s in the post-scaling of Fig. 2-7(c) is implemented by the integer multiplication between m and k

followed by an n-bit right-shifting operation, where the integer k and natural number n are defined in the dyadic fraction representation of s, $k/2^n$. The operations needed are so fundamental that the ALUs of different processors are able to produce exactly the same results if the number of shifting bits n (more specifically, n_1 for the post-scaling stage and n_2 for the pre-scaling stage) is properly specified hence the potential mismatch between the encoder side and decoder side can be avoided. Nevertheless, a dyadic fraction is only an approximation of the exact value of a real number (usually irrational), hence the dyadic approximation also introduces errors. Fig. 2-7(c) is a part of the implementation block diagram of the H.264/AVC for the ideal case by assuming the shifting bits n_1, $n_2 \rightarrow \infty$, whereas Fig. 2-7(d) shows the practical case when the number of shifting bits is not infinitely large. The error introduced by the dyadic approximation can be measured in terms of the ratio between the approximated value and the actual value, which is $(k/2^n)/s$. The dyadic approximation error is implicitly shown by the "Finite Precision" in blocks 2 and 3 of Fig. 2-7(d), which is not traced by any block and thus inconvenient for mathematical analysis. Hence it is beneficial to explicitly trace the errors in the post-scaling and the pre-scaling block by the 2nd and 5th blocks in Fig. 2-7(e), in which all aforementioned errors are explicitly modeled. The reason that the transform block and the quantization block are used instead of the integer transform block and the post-scaling block is due to their equivalence which has been shown in Fig. 2-7(a) to Fig. 2-7(c). We can thus derive the MSE between the reconstructed signal and the input signal taking the dyadic approximation as well as the quantization and the nonorthogonality into the consideration. Let us first denote

1) an input signal as an $N\times1$ vector $\mathbf{x}$,
2) the normalized nonorthogonal transform kernel as an $N\times N$ matrix $\mathbf{H}$,
3) the transformed vector as $\mathbf{z}$,
4) the matrix which models the error introduced by the dyadic approximation at the post-scaling stage as $\mathbf{S}_1$ (which is a diagonal matrix with diagonal elements slightly larger or smaller than 1),
5) the distorted vector due to the dyadic approximation as $\mathbf{v}$,
6) the quantized vector as $\mathbf{u}$,
7) the vector modeling the quantization error as $\mathbf{n}_q$,
8) the matrix which models the error introduced by the dyadic approximation at the pre-scaling stage as $\mathbf{S}_2$ (which is a diagonal matrix with diagonal elements slightly larger or smaller than 1),
9) the output vector as $\mathbf{y}_r$, and
10) the output obtained by omitting the quantization as $\mathbf{y}$.

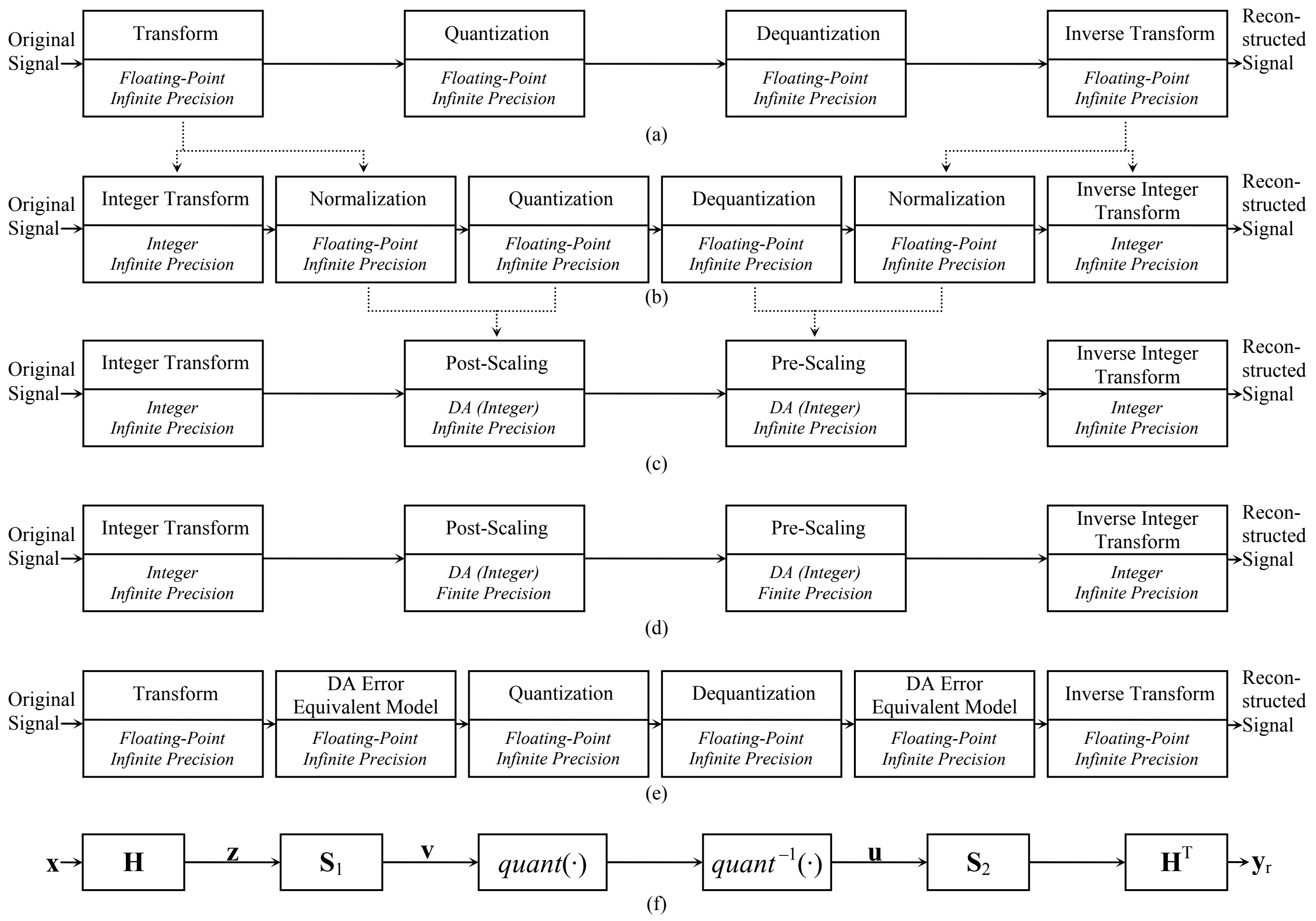


Fig. 2-7. Each block diagram describes the stages that convert the original signal into the reconstructed signal. Each row presents a variation: (a), (b), and (c) the ideal cases; (d) the practical case which includes the dyadic approximation error implicitly; (e) the practical case which models all error factors explicitly; and (f) the analytical expression of (e). Note that i) for each block in (a)–(e), the bottom part indicates whether this block causes error; and ii) "DA" stands for "Dyadic Approximation".

Note that throughout this thesis, we use a letter in bold to represent a matrix or a vector (for example **H** or **x**), whereas a letter in italic with index term(s) to represent one particular element in that matrix or vector (for example, $H(i,j)$ or $x(i)$). Let us also define the following symbols and equivalences for the clarity of presentation

$$\mathbf{N}_1 = \mathbf{N}_1^{\mathrm{T}} = \mathbf{S}_1^{-1} - \mathbf{I} \tag{2-85}$$

$$\mathbf{N}_2 = \mathbf{N}_2^{\mathrm{T}} = \mathbf{S}_2 \quad - \mathbf{I} \tag{2-86}$$

$$\mathbf{Er} = \mathbf{Er}^{\mathrm{T}} = \mathbf{H}\mathbf{H}^{\mathrm{T}} - \mathbf{I} \tag{2-87}$$

Note that the transpose relationship in (2-85) and (2-86) is obvious since the off-diagonal elements of $\mathbf{S}_i$ for i = 1 or 2 are all zero. Let us represent the operation from the forward transform to inverse transform in Fig. 2-7(f) by the following set of equations

$$\mathbf{z} = \mathbf{H}\mathbf{x} \tag{2-88}$$

$$\mathbf{v} = \mathbf{S}_1\mathbf{z} \tag{2-89}$$

$$\mathbf{u} = quant^{-1}[quant(\mathbf{v})] = \mathbf{v} - \mathbf{n}_q \tag{2-90}$$

$$\mathbf{y}_r = (\mathbf{S}_2\mathbf{H})^{\mathrm{T}}\mathbf{u} \tag{2-91}$$

The reconstruction error can be calculated by using the average MSE between the input $\mathbf{x}$ and output $\mathbf{y}_\mathrm{r}$ as shown below

$$\sigma_\mathrm{r}^2 = \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{x}-\mathbf{y}_\mathrm{r}\right)^\mathrm{T}\left(\mathbf{x}-\mathbf{y}_\mathrm{r}\right)\right] \tag{2-92}$$

The difference between $\mathbf{x}$ and $\mathbf{y}_\mathrm{r}$ can be expanded into

$$\begin{aligned}\mathbf{x}-\mathbf{y}_\mathrm{r} &= \mathbf{H}^{-1}\mathbf{z}-\left(\mathbf{S}_2\mathbf{H}\right)^\mathrm{T}\mathbf{u} \\ &= \left(\mathbf{H}^\mathrm{T}-\mathbf{H}_\mathrm{er}\right)\left(\mathbf{S}_1^{-1}\mathbf{v}\right)-\left(\mathbf{S}_2\mathbf{H}\right)^\mathrm{T}\left(\mathbf{v}-\mathbf{n}_\mathrm{q}\right) \\ &= \mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}-\mathbf{H}_\mathrm{er}\mathbf{S}_1^{-1}\mathbf{v}+\mathbf{H}^\mathrm{T}\left(\mathbf{S}_1^{-1}-\mathbf{S}_2\right)\mathbf{v}\end{aligned} \tag{2-93}$$

where $\mathbf{H}_\mathrm{er} = \mathbf{H}^\mathrm{T} - \mathbf{H}^{-1}$. Let us define

$$\mathbf{A} = \mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}-\mathbf{H}_\mathrm{er}\mathbf{S}_1^{-1}\mathbf{v} \tag{2-94}$$

$$\mathbf{B} = \mathbf{H}^\mathrm{T}\left(\mathbf{S}_1^{-1}-\mathbf{S}_2\right)\mathbf{v} \tag{2-95}$$

Hence (2-92) can be represented by

$$\begin{aligned}\sigma_\mathrm{r}^2 &= \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{A}+\mathbf{B}\right)^\mathrm{T}\left(\mathbf{A}+\mathbf{B}\right)\right] \\ &= \tfrac{1}{N}\left\{\mathrm{E}\left[\mathbf{A}^\mathrm{T}\mathbf{A}\right]+2\mathrm{E}\left[\mathbf{A}^\mathrm{T}\mathbf{B}\right]+\mathrm{E}\left[\mathbf{B}^\mathrm{T}\mathbf{B}\right]\right\}\end{aligned} \tag{2-96}$$

The first term of (2-96) can be written as

$$\begin{aligned}\tfrac{1}{N}\mathrm{E}\left[\mathbf{A}^\mathrm{T}\mathbf{A}\right] &= \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}-\mathbf{H}_\mathrm{er}\mathbf{S}_1^{-1}\mathbf{v}\right)^\mathrm{T}\left(\mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}-\mathbf{H}_\mathrm{er}\mathbf{S}_1^{-1}\mathbf{v}\right)\right] \\ &= \tfrac{1}{N}\mathrm{E}\left[\mathbf{n}_\mathrm{q}^\mathrm{T}\mathbf{S}_2\mathbf{H}\mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}+\mathbf{v}^\mathrm{T}\mathbf{S}_1^{-1}\mathbf{H}_\mathbf{er}^\mathrm{T}\mathbf{H}_\mathbf{er}\mathbf{S}_1^{-1}\mathbf{v}-2\mathbf{v}^\mathrm{T}\mathbf{S}_1^{-1}\mathbf{H}_\mathrm{er}^\mathrm{T}\mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}\right] \\ &= \tfrac{1}{N}\mathrm{E}\left[\mathbf{n}_\mathrm{q}^\mathrm{T}\mathbf{n}_\mathrm{q}+\mathbf{n}_\mathrm{q}^\mathrm{T}\left(\mathbf{N}_2^\mathrm{T}\mathbf{Er}\mathbf{N}_2+\mathbf{N}_2^\mathrm{T}\mathbf{N}_2+\mathbf{Er}+2\mathbf{Er}\mathbf{N}_2+2\mathbf{N}_2\right)\mathbf{n}_\mathrm{q}+\left(\mathbf{y}-\mathbf{x}\right)^\mathrm{T}\left(\mathbf{y}-\mathbf{x}\right)\right] \\ &= \sigma_\mathrm{q}^2+\tfrac{1}{N}\mathrm{E}\left[\mathbf{n}_\mathrm{q}^\mathrm{T}\left(\mathbf{N}_2^\mathrm{T}\mathbf{Er}\mathbf{N}_2+\mathbf{N}_2^\mathrm{T}\mathbf{N}_2+\mathbf{Er}+2\mathbf{Er}\mathbf{N}_2+2\mathbf{N}_2\right)\mathbf{n}_\mathrm{q}\right]+\sigma_\mathrm{r0}^2\end{aligned}$$

(2-97)

where $\sigma_\mathrm{r0}^2 = \tfrac{1}{N}\mathrm{E}\left[\left(\mathbf{y}-\mathbf{x}\right)^\mathrm{T}\left(\mathbf{y}-\mathbf{x}\right)\right]$. It can be observed that σ_r0^2 is the average MSE between the input and output without quantization. It has been shown in [74] that

$$\sigma_\mathrm{r0}^2 \le \sigma_\mathrm{x}^2\left[\tfrac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}M(k,j)\right], \tag{2-98}$$

where

$$\mathbf{M} = (\mathbf{H}^\mathrm{T}\mathbf{H} - \mathbf{I})^\mathrm{T}(\mathbf{H}^\mathrm{T}\mathbf{H} - \mathbf{I}). \tag{2-99}$$

For the second term of (2-96), we have

$$\begin{aligned}\tfrac{2}{N}\mathrm{E}\left[\mathbf{A}^\mathrm{T}\mathbf{B}\right] &= \tfrac{2}{N}\mathrm{E}\left[\left(\mathbf{H}^\mathrm{T}\mathbf{S}_2\mathbf{n}_\mathrm{q}-\mathbf{H}_\mathrm{er}\mathbf{S}_1^{-1}\mathbf{v}\right)^\mathrm{T}\left(\mathbf{H}^\mathrm{T}\left(\mathbf{S}_1^{-1}-\mathbf{S}_2\right)\mathbf{v}\right)\right] \\ &= \tfrac{2}{N}\mathrm{E}\left[\mathbf{n}_\mathrm{q}\mathbf{S}_2\mathbf{H}\mathbf{H}^\mathrm{T}\left(\mathbf{S}_1^{-1}-\mathbf{S}_2\right)\mathbf{v}+\mathbf{v}^\mathrm{T}\left(-\mathbf{S}_1^{-1}\left(\mathbf{H}^\mathrm{T}-\mathbf{H}^{-1}\right)\mathbf{H}^\mathrm{T}\left(\mathbf{S}_1^{-1}-\mathbf{S}_2\right)\right)\mathbf{v}\right]\end{aligned} \tag{2-100}$$

where $\mathrm{E}[\mathbf{n}_\mathrm{q}\mathbf{S}_2\mathbf{H}\mathbf{H}^\mathrm{T}(\mathbf{S}_1^{-1} - \mathbf{S}_2)\mathbf{v}] = 0$ since

$$\mathrm{E}[n_\mathrm{q}(i)v(j)] = 0 \text{ for all } i, j \in [0, N-1].$$

This can be proved by considering the cases for $i = j$ and $i \neq j$. For $i = j$, E[$n_q(i)v(j)$] measures the correlation between a quantizer input and its quantization noise. As is shown in [97], these two signals are uncorrelated provided that the quantizer input is "band-limited in the CF domain" of which general cases approximately fulfills the condition. For $i \neq j$, E[$n_q(i)v(j)$] measures the correlation between a quantizer input and the quantization noise of another quantizer. By assuming that the DCT coefficients are independent of each other (which means that each DCT coefficient can be regarded as an independent quantizer input), $n_q(i)$ and $v(j)$ are obviously uncorrelated. Hence

$$\tfrac{2}{N}\mathrm{E}\left[\mathbf{A}^{\mathrm{T}}\mathbf{B}\right] = \tfrac{2}{N}\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\mathbf{Er}\left(\mathbf{S}_2\mathbf{S}_1 - \mathbf{I}\right)\mathbf{H}\right)\mathbf{x}\right]. \tag{2-101}$$

For the third term of (2-96)

$$\begin{aligned}\tfrac{2}{N}\mathrm{E}\left[\mathbf{B}^{\mathrm{T}}\mathbf{B}\right] &= \tfrac{2}{N}\mathrm{E}\left[\left(\mathbf{H}^{\mathrm{T}}\left(\mathbf{S}_1^{-1} - \mathbf{S}_2\right)\mathbf{v}\right)^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\left(\mathbf{S}_1^{-1} - \mathbf{S}_2\right)\mathbf{v}\right)\right] \\ &= \tfrac{2}{N}\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\mathbf{S}_1^{\mathrm{T}}\left(\mathbf{S}_1^{-1} - \mathbf{S}_2\right)^{\mathrm{T}}\left(\mathbf{Er} + \mathbf{I}\right)\left(\mathbf{S}_1^{-1} - \mathbf{S}_2\right)\mathbf{S}_1\mathbf{H}\right)\mathbf{x}\right] \\ &= \tfrac{2}{N}\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\left(\mathbf{I} - \mathbf{S}_1\mathbf{S}_2\right)\left(\mathbf{Er} + \mathbf{I}\right)\left(\mathbf{I} - \mathbf{S}_2\mathbf{S}_1\right)\mathbf{H}\right)\mathbf{x}\right]\end{aligned} \tag{2-102}$$

Hence we obtain

$$\sigma_{\mathrm{r}}^2 = \sigma_{\mathrm{q}}^2 + \sigma_{\mathrm{r0}}^2 + \left\{\tfrac{1}{N}\mathrm{E}\left[\mathbf{n}_{\mathrm{q}}^{\mathrm{T}}\mathbf{\Omega}\mathbf{n}_{\mathrm{q}}\right] + \tfrac{1}{N}\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\mathbf{\Psi}\mathbf{x}\right]\right\} \tag{2-103}$$

where σ_{q}^2 is the average MSE of the quantization error, σ_{r0}^2 is the average MSE of the nonorthogonality error,

$$\mathbf{\Omega} = \mathbf{N}_2^{\mathrm{T}}\mathbf{Er}\mathbf{N}_2 + \mathbf{N}_2^{\mathrm{T}}\mathbf{N}_2 + \mathbf{Er} + 2\mathbf{Er}\mathbf{N}_2 + 2\mathbf{N}_2\text{, and} \tag{2-104}$$

$$\mathbf{\Psi} = \mathbf{H}^{\mathrm{T}}\mathbf{S}_1\left[\mathbf{N}_1 - \mathbf{N}_2 - \left(\mathbf{N}_1 + \mathbf{N}_2 + 2\mathbf{I}\right)\mathbf{Er}\right]\left(\mathbf{N}_1 - \mathbf{N}_2\right)\mathbf{S}_1\mathbf{H}. \tag{2-105}$$

Let us continue to simplify the third term of (2-103) as shown below

$$\begin{aligned}\mathrm{E}\left[\mathbf{n}_{\mathrm{q}}^{\mathrm{T}}\mathbf{\Omega}\mathbf{n}_{\mathrm{q}}\right] &= \mathrm{E}\left[\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Omega(i,j)n_{\mathrm{q}}(i)n_{\mathrm{q}}(j)\right] \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Omega(i,j)\mathrm{E}\left[n_{\mathrm{q}}(i)n_{\mathrm{q}}(j)\right]\end{aligned}. \tag{2-106}$$

For quantization noise $\mathbf{n}_q$, elements of its autocorrelation matrix are zero except the diagonal terms, hence

$$\mathrm{E}\left[\mathbf{n}_{\mathrm{q}}^{\mathrm{T}}\mathbf{\Omega}\mathbf{n}_{\mathrm{q}}\right] = \sum_{i=0}^{N-1}\Omega(i,i)\mathrm{E}\left[n_{\mathrm{q}}^2(i)\right] = \sigma_{\mathrm{q}}^2\sum_{i=0}^{N-1}\Omega(i,i) \tag{2-107}$$

Similarly, the fourth term of (2-103) can be simplified as shown below

$$\begin{aligned}\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\mathbf{\Psi}\mathbf{x}\right] &= \mathrm{E}\left[\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Psi(i,j)x(i)x(j)\right] \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Psi(i,j)\mathrm{E}\left[x(i)x(j)\right] \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Psi(i,j)R_x(i-j) \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\Psi(i,j)\sigma_x^2\rho(i,j)\end{aligned}. \tag{2-108}$$

The correlation coefficient $\rho(i,j)$ ranges from −1 to 1, hence the maximum value of $\mathrm{E}[\mathbf{x}^{\mathrm{T}}\mathbf{\Psi}\mathbf{x}]$ can be reached if a) $|\rho(i,j)| \to 1$, and b) $\rho(i,j)$ has the same sign with $\Psi(i,j)$. Condition a) is easily fulfilled by setting the absolute value of all correlation coefficients to 1. Condition b) can only be fulfilled for some positions. It is because the number of sign patterns of $\Psi(i,j)$ is $2^{(N-1)^2/2}$, whereas the number of sign patterns of $\rho(i,j)$ is only 2^{N-1}. Although condition b) cannot be really achieved, our experimental works show that b) can lead to a good approximation to the real upper bound. Hence the fourth term of (2-103) can be written as

$$\mathrm{E}\left[\mathbf{x}^{\mathrm{T}}\mathbf{\Psi}\mathbf{x}\right] \le \sigma_{\mathrm{x}}^2 \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \left|\Psi(i,j)\right| \tag{2-109}$$

Let us also modify (2-98) by using the above idea,

$$\sigma_{\mathrm{r0}}^2 \le \sigma_{\mathrm{x}}^2 \left[\tfrac{1}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \left|M(k,j)\right|\right] \tag{2-110}$$

Hence the upper bound of the average variance of reconstruction can be written as follow

$$\sigma_{\mathrm{q}}^2 + \sigma_{\mathrm{x}}^2 \left[\tfrac{1}{N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \left|M(i,j)\right|\right] + \left\{\sigma_{\mathrm{q}}^2 \left[\tfrac{1}{N} \sum_{i=0}^{N-1} \Omega(i,i)\right] + \sigma_{\mathrm{x}}^2 \left[\tfrac{1}{N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \left|\Psi(i,j)\right|\right]\right\} \tag{2-111}$$

where M, Ω and Ψ are defined by (2-99), (2-104) and (2-105) respectively. The first, second and third terms are caused by the quantization term, the nonorthogonality term and the dyadic approximation term, respectively. By examining (2-104) and (2-105) carefully, we observe that the dyadic approximation term depends also on the nonorthogonality of transform kernel, however, the nonorthogonality term only depends on itself.

Chapter 3. Transform Kernel Selection Strategy for H.264/AVC and Future Video Coding Standards

3.1. Introduction

As we have mentioned in Section 2.3, the H.264/AVC heavily makes use of the simplest order-4 integer cosine transform (ICT) kernel for the transform coding process[b].On a careful examination, it is not difficult to find that the normalized ICT kernel of the H.264/AVC is not too close to the discrete cosine transform (DCT) kernel. Hence the desirable properties measured such as decorrelation property, energy compaction property and compression ability which are available in DCT are limited. However, experimental results show that its performance is comparable to the case of employing the DCT Kernel [98]. This has arisen the curiosity of many of us on why the supposed-to-be-deteriorated kernel can have similar performance as the DCT Kernel. One explanation is that some signals are better compressed by the DCT Kernel while others are better compressed by the ICT kernel. In this chapter, we have designed a whole set of experiments and the theoretical works that possibly lead to some answers.

Under the framework of the H.264/AVC, the transform kernel must be integer, which is a mandatory requirement in order to avoid the drift problem. Hence for a practical realization, kernels must be provided with integer versions which may have different characteristics compared to the integerized versions of original floating-point kernels due to scaling and rounding. Some researchers focused on the existing kernels with various known characteristics and used various combinations of them to see if the coding efficiency can be improved and the reason behind. We also follow this approach. In this chapter, all new kernels are directly-derived integer kernels (they can be normalized to orthonormal kernels easily), so the characteristics between integer version and orthonormal version are exactly the same, which is an advantage. We have also tried to search for integer kernels with simple integers for the ease of decoding, while the orthonormal kernels (i.e. kernels derived directly from the Karhunen–Loève (KL) Transform) mainly have irrational floating numbers, hence it is often that their simple integer version cannot be found satisfactorily.

In Subsection 3.2.1, the notations used in this chapter are defined. In Subsection 3.2.2, we review the multiple-kernel scheme and suggest kernels that can be used. In Subsection 3.2.3, we propose one DCT-like integer kernel and revitalize another DCT-like integer kernel. Subsequently in Subsection 3.2.4, we show that a dual-kernel system comprising of the H.264/AVC Kernel and any one of the proposed kernels has a higher coding efficiency. In

[b] To some extent, Walsh–Hadamard Transform (WHT) has also been used.

Subsection 3.3.1, we discuss our discovery of the kernel selection tendencies when a dual-kernel system is employed. In Subsection 3.3.2, before exploring the kernel selection tendencies, we visualize the kernel selection process using a graphical approach. In Subsection 3.3.3, we propose the newly-found rate-distortion feature that has a crucial effect to the kernel selection tendencies. In Subsection 3.3.4, we analyze the theory behind this feature. In Subsection 3.3.5, we propose a fast kernel selection algorithm which uses one kernel for each type of frame in the H.264/AVC. In Subsection 3.3.6, we generalize the rate-distortion feature from two kernels to more kernels. In Subsection 3.3.7, we further generalize the algorithm to adapt different choices of compromises on quality and bitrate. In Section 3.4, we give how a video coding standard and the relative codec should change by employing our proposed algorithms. In Section 3.5, conclusions are drawn and further directions are discussed.

3.2. Macroblock-Level Adaptive Kernel Mechanism

3.2.1. Notations

Recall that the integer transform and scaling process used in the H.264/AVC can be denoted as

$$\mathbf{Y} = \mathbf{H} \cdot \mathbf{X} \cdot \mathbf{H}^{\mathrm{T}} \otimes \mathbf{E} \tag{3-1}$$

where **X** denotes the 4×4 input residual signal, **Y** denotes the transformed-and-quantized signal, **H** denotes that integer transform kernel, **E** denotes the multiplication matrix, "T" means the matrix transpose operation and the "$\otimes$" denotes term-by-term multiplication.

The integer version of a DCT-like kernel mentioned in this chapter is denoted as IK(a,b,c) and its complete form is

$$\mathbf{H}_{\text{DCT-like}} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \tag{3-2}$$

where a, b and c are all non-negative integer values. It can also be denoted as (a,b,c) for the sake of simplicity if no ambiguity is found (such as those in TABLE 3-5). For example, the H.264/AVC default integer kernel (H.264/AVC Kernel) is denoted as IK(1,2,1) , i.e. a = 1, b = 2 and c = 1 [6].

Recall also that the integer version of a DST-like kernel was employed as an alternative kernel for less correlated signals together with the H.264/AVC Kernel as the main kernel for the codec in [57-58]. This particular integer version of DST-like kernel is referred to as the IST Kernel as shown in (3-3).

$$\mathbf{H}_{\text{DST-like}} = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & -1 & -1 \\ 2 & -1 & -1 & 2 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{3-3}$$

Note that these kernels can be normalized to obtain their orthonormal kernels which allow perfect reconstruction for signals fed for a pure transform process. Also, the H.264/AVC combines the normalizations of the integer kernels and the quantization / dequantization process together to form the corresponding scaling matrices (called the “multiplication matrix” and the “rescaling matrix”) which can be easily deduced by following a set of standard formulas in Subsection 2.3.1.

3.2.2. Multiple-Kernel Scheme

Let us also simplify the representation of multiple-kernel scheme in the transform coding process, and name it as the Adaptive Kernel Mechanism (AKM). Conventionally, the AKM is applied at the macroblock level, hence we name it as “MacroBlock-level AKM”—MB-AKM. It is adaptive because the encoder adaptively chooses the best kernel for each macroblock. The MB-AKM aims at achieving higher coding efficiency by selecting the most suitable kernel from a group of candidates at the transform coding stage. It employs the coding cost calculated by the Rate-Distortion Optimization (RDO) [99-100] as the kernel selection criterion. After kernel selection, with the extra information given by two relative flags: the MB Mode (Macroblock Mode) flag and the CBP (Coded Block Pattern) flag [6], a sequence of signaling bits is generated by the encoder to indicate the selected kernel for every macroblock. At the decoder side, each macroblock can be inversely transformed using the kernel indicated by the signaling bit directly. In Section 3.3, we will apply the AKM to a higher level – the frame level. Hence we name the algorithm as a Frame-level AKM (FM-AKM).

In this chapter, we employ the AKM to improve the coding efficiency of the H.264/AVC, and make analyses on the questions raised at the end of the first paragraphs in Section 3.1. In order to improve the coding efficiency, kernels which lead to high compression ratio should be considered. It is thus intuitively to find kernels with good decorrelation and energy compaction properties, since these characteristics are closely related the compression ability of the encoder. Besides, the structure of a kernel should be simple, and this is an important factor for improving the computational efficiency.

In Ref. [57-58], it appears that the IST Kernel is suitable for the decorrelation of less correlated signals, and this leaves the H.264/AVC Kernel to decorrelate highly correlated signals. However, we would like to point out that the decorrelation property of the H.264/AVC Kernel is limited compared to the DCT Kernel. In other words, the DCT Kernel has better performance in decorrelating the highly correlated signals (such as residual signals found around edges of objects) than the H.264/AVC Kernel. Hence beyond the H.264/AVC and IST Kernels, it is good to employ a third kernel which is even closer to the DCT Kernel than the H.264/AVC Kernel for the highly correlated signals. For the sake of ease of presentation, we name the third kernel as the "DCT-like kernel". In the strict sense, the H.264/AVC Kernel is also DCT-like. However, since its similarity to the DCT Kernel is not high comparing to other DCT-like kernels and it has simple structure, we use the term "H.264/AVC Kernel" to imply its uniqueness.

When the AKM with these three kernels are used, each kernel has a role of itself: 1) The decorrelation and energy compaction properties of the DCT-like kernel is comparable to those of the DCT Kernel, hence the DCT-like kernel is used for highly correlated signal. 2) The H.264/AVC Kernel IK(1,2,1) preserves the DCT properties to some extent, and it often can be used as a substitution of the DCT-like kernel. The main reason of employing this kernel is its simplicity (for this transform can be realized by only add and shift operations). 3) The IST Kernel is be used for low correlated signal. However, some researchers have shown that the performance difference between the DST and the DCT for low correlation signals is small [20]. This is one of the reasons triggering us to look into this problem in details and to set up a new combination, which eliminates the IST Kernel and uses the IK(1,2,1) together with another DCT-like kernel for possible improvement.

3.2.3. Proposed DCT-Like Kernels

Since the AKM includes several kernels and the H.264/AVC and IST Kernels are available, we need to see if there are DCT-like kernels which have closer property with the DCT Kernel. One possible way is to scale up the floating-point DCT Kernel by a scalar *u* and then round the scaled result so as to preserve the characteristics of DCT, as we have reviewed in 2.1.4.4. This is a systematical approach to find DCT-like kernels. During the kernel searching process, the similarity between the newly-found kernel and the DCT Kernel was indirectly measured by the kernel percentage error (*KPE*) as shown below

$$KPE = \left| \frac{\sqrt{\frac{1}{2}\left[\left(\frac{\gamma}{\beta}\right)^2 + 1\right]} + \frac{\gamma}{\beta}}{\sqrt{\frac{1}{2}\left[\left(\frac{\gamma_{\mathrm{DCT}}}{\beta_{\mathrm{DCT}}}\right)^2 + 1\right]} + \frac{\gamma_{\mathrm{DCT}}}{\beta_{\mathrm{DCT}}}} - 1 \right| \tag{3-4}$$

where β_{DCT} and γ_{DCT} are the corresponding coefficients of the DCT Kernel. Detailed derivation of KPE can be found in 2.1.4.4. The smaller the KPE value is, the closer a newly-found DCT-like kernel resembles the DCT Kernel.

It should be emphasized that a kernel must be simple so that they lead to small dynamic ranges of intermediate results, in order to keep the computational complexity at a reasonable level. Through a computer search by setting *u* from 1.00 to 50.00 with step 0.01, we have found a number of integer kernels which have relatively small *KPE* as shown in TABLE 3-1. There are some new kernels as a result of our search, while others were proposed in the literature before. The first few small-scale kernels have the advantage of simplicity, while as the scale of kernel increases the similarity also increases. We have found that when *u* is around 26.00, the similarity measure of the scaled kernel IK(13,17,7) reaches the maximum. It is also the most famous DCT-like kernel once adopted during the H.264/AVC standardization process proposed by Bjontegaard [94]. The multiplication factors and rescaling factors are subsequently calculated according to Subsection 2.3.1 and the results are shown in TABLE 3-2.

TABLE 3-1
THE DCT-LIKE KERNELS FOUND BY THE UP-SCALING METHOD

Kernel	KPE	Description
IK(2,3,1)	8.55%	Adopted by AVS-M [83] [c]
IK(3,4,2)	9.41%	New Kernel
IK(4,5,2)	1.53%	Adopted by AVS-M
IK(5,6,2)	8.55%	New Kernel
IK(5,7,3)	1.55%	New Kernel
IK(6,8,3)	4.19%	New Kernel
IK(7,9,4)	3.28%	Adopted by AVS-M
…		
IK(13,17,7)	0.26%	Proposed by Bjontegaard [94], once adopted by the H.26L TML-1 [101]
…		

TABLE 3-2
MULTIPLICATION FACTORS AND RESCALING FACTORS OF IK(13,17,7) FOR H.264/AVC[d]

QP mod 6	Multiplication Factor (MF)			Rescaling Factor (RF)		
	Pos1	Pos2	Pos3	Pos1	Pos2	Pos3
0	4699	4699	4699	4	4	4
1	4699	4699	4699	4	4	4
2	3759	3759	3759	5	5	5
3	3759	3759	3759	5	5	5
4	3133	3133	3133	6	6	6
5	2685	2685	2685	7	7	7

However, the kernel IK(13,17,7) also poses a problem that the dynamic range of intermediate step variables increases significantly [98]. The additional bits of the dynamic

[c] The AVS stands for the Audio Video Standard. It is an audio-video coding standard initiated by the government of P. R. China. AVS-M is a part of the AVS which mainly targets at the application of mobile video.

[d] "Pos1" refers to positions (0,0), (0,2), (2,0) and (2,2); "Pos2" refers to positions (1,1), (1,3), (3,1) and (3,3) and "Pos3" refers to the remaining positions.

range required for a new integer kernel with respect to the H.264/AVC Kernel, $\Delta BITS$, can be calculated by

$$\Delta BITS = 2\log_2\left[\left(\sum_{j=0}^{3}\left|k_{ij}\right|\right)_{\max,\forall i\in[0,3]} \middle/ 6\right] \tag{3-5}$$

where k_{ij} is the element of integer kernel at i^{th} row and j^{th} column. For IK(13,17,7), it needs an addition of 6 bits for the integer transform coefficients compared to the H.264/AVC Kernel, which requires a 32-bit multiplication in general computer realization. There are two approaches to reduce this problem, one is to implement the codec by dedicated hardware, and the other is to choose a kernel with smaller $\Delta BITS$.

With the latter idea, we propose the newly-found kernel IK(5,7,3) to serve as an alternative kernel. This kernel has two main properties. 1) The ratios among the kernel elements *a*, *b* and *c* are similar to the DCT Kernel so as to preserve the decorrelation property, however, it is not so close as that of the IK(13,17,7). 2) The kernel IK(5,7,3) can be represented by only 3 bits ($7_{10}=111_2$), and the $\Delta BITS$ is also 3. Hence we can see that IK(5,7,3) is a compromise between the decorrelation property (represented by IK(13,17,7)) and the computational complexity (represented by IK(1,2,1)). The multiplication factors and rescaling factors are subsequently calculated according to Subsection 2.3.1 and the results are shown in TABLE 3-3.

TABLE 3-3
MULTIPLICATION FACTORS AND RESCALING FACTORS OF IK(5,7,3) FOR H.264/AVC

QP mod 6	Multiplication Factor (MF)			Rescaling Factor (RF)		
	Pos1	Pos2	Pos3	Pos1	Pos2	Pos3
0	4474	3325	3857	3	3	3
1	3355	3325	3857	4	3	3
2	3355	2494	2893	4	4	4
3	3355	2494	2893	4	4	4
4	2684	2494	2314	5	4	5
5	2237	1995	2314	6	5	5

3.2.4. Experimental Results of MB-AKM

In this subsection, we evaluate the MB-AKM with different alternative kernels. We implemented the MB-AKM in Joint Model (JM) 12.2 [102], and our evaluation is in-line with the "Recommended Simulation Common Conditions" suggested by VCEG [103], but we have made some slightly changes:

- For the QCIF and CIF sequences, we encoded the first 300 frames; for HD sequences, we encoded the first 100 frames.
- The prediction structure selected was IBBP.
- The video sequences were coded using High Profile.
- The frame rate is 15 Hz for the first 4 sequences and 30 Hz for other sequences.

- The quantization parameter (QP) values are 22, 27, 32 and 37. The QP values are incremented by 1 for each level of reference.
- The search range is 32 for QCIF/CIF and is 64 for HD sequences.
- The entropy coding scheme is CABAC.
- The RDO was on.
- The 8×8 transform was off.

The evaluation results are shown in the TABLE 3-4. We tested the performances of MB-AKM by employing IK(13,17,7), IK(5,7,3) and the IST Kernel, respectively. We used the Bjontegaard Metric [104] to measure both bitrate change ($\Delta Bitrate$) and PSNR change ($\Delta PSNR$). Note that all the results listed in the table are the changes of bitrates and PSNR of new schemes with respect to the H.264/AVC's default arrangement (using the single kernel IK(1,2,1)). Either a negative $\Delta Bitrate$ or a positive $\Delta PSNR$ means an improvement in coding efficiency over the H.264/AVC Kernel. The sub-averages are listed to show the impact of MB-AKM to videos of different sizes. The signaling bits which are the overhead introduced by the MB-AKM are not included in the TABLE 3-4 due to the limited space and also for the sake of simplicity, without affecting the comparison since all combinations require more or less the same among of signaling bits. However, we do include the signaling bits in the TABLE 3-5 for MB-AKM{(1,2,1)&(13,17,7)} and MB-AKM{(1,2,1)&(5,7,3)}. According to the test results, the signaling bits will increase the overall bitrate by around 0.4%.

The column "Complexity" representing the complexity of a video sequence is a loosely defined term which were resulted from the observers' subjective opinions within the range of frames set by the simulation condition. The higher the value is, the more complex the video sequence is regarded.

From TABLE 3-4, we can see that both MB-AKM{(1,2,1)&(13,17,7)} and MB-AKM{(1,2,1)&(5,7,3)} outperform the MB-AKM{(1,2,1)&IST} in almost all cases. The average bitrate saving of MB-AKM{(1,2,1)&(13,17,7)}, MB-AKM{(1,2,1)&(5,7,3)} and MB-AKM{(1,2,1)&IST} without the costs of signaling bits are 3.18%, 2.52% and 0.70%, respectively.

When only IK(13,17,7) or IK(5,7,3) is used instead of the H.264/AVC Kernel, the performance is even worse in most cases, which complies with the results in [98]. The result may give some of us a surprise but it is reasonable, and will be explained in details in Subsection 3.3.4.

If we add the IST Kernel as the third candidate kernel in parallel with IK(13,17,7) or IK(5,7,3), the improvement is very small, indicating that the DCT-like kernels are superior to the IST Kernel serving as alternative kernel. It is also indicated by our further experimental work that the MB-AKM with more than two kernels cannot improve the coding efficiency significantly. (Results are not included here since the improvements are not significant.)

TABLE 3-4
EVALUATION RESULTS—THE COMPARISON AGAINST THE H.264/AVC'S DEFAULT SCHEME USING IK(1,2,1)

Sequence			Results Relate to IK(13,17,7)						Results Relate to IK(5,7,3)						IST	
			MB-AKM{(1,2,1)&(13,17,7)}		MB-AKM{(1,2,1)&(13,17,7)&IST}		IK(13,17,7)		MB-AKM{(1,2,1)&(5,7,3)}		MB-AKM{(1,2,1)&(5,7,3)&IST}		IK(5,7,3)		MB-AKM{(1,2,1)&IST}	
Size	Name	Complexity	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)	ΔBitrate (%)	ΔPSNR (dB)
QCIF	Container	1	-1.30	0.0624	-0.84	0.0382	14.72	-0.6379	-1.02	0.0467	-1.10	0.0527	12.79	-0.5600	-0.24	0.0160
	Foreman	3	-3.49	0.1713	-3.89	0.1913	1.92	-0.0946	-2.82	0.1378	-3.42	0.1677	2.49	-0.1212	-1.19	0.0579
	Silent	2	-4.52	0.2488	-4.69	0.2591	14.90	-0.7460	-3.45	0.1940	-3.89	0.2148	13.95	-0.6990	-0.84	0.0443
QCIF Average			*-3.10*	*0.1608*	*-3.14*	*0.1629*	*10.51*	*-0.4929*	*-2.43*	*0.1262*	*-2.80*	*0.1451*	*9.74*	*-0.4601*	*-0.76*	*0.0394*
CIF	Paris	1	-4.57	0.2510	-4.87	0.2686	5.40	-0.3136	-3.68	0.2038	-3.68	0.2044	6.21	-0.3503	-0.72	0.0398
	Foreman	2	-2.83	0.1243	-3.25	0.1426	3.89	-0.1669	-2.18	0.0956	-2.64	0.1154	3.93	-0.1668	-0.87	0.0383
	Mobile	4	-5.23	0.2384	-5.48	0.2492	-2.82	0.1217	-4.25	0.1934	-4.33	0.1969	-1.80	0.0748	-0.75	0.0360
	Tempete	3	-6.13	0.2532	-6.26	0.2576	-3.08	0.1178	-5.23	0.2159	-5.26	0.2169	-2.43	0.0916	-0.66	0.0279
CIF Average			*-4.69*	*0.2167*	*-4.96*	*0.2295*	*0.85*	*-0.0603*	*-3.84*	*0.1772*	*-3.98*	*0.1834*	*1.48*	*-0.0877*	*-0.75*	*0.0355*
HD (720p)	BigShips	2	-0.67	0.0177	-0.51	0.0134	10.47	-0.2702	-0.35	0.0092	-0.60	0.0157	9.00	-0.2331	-0.28	0.0073
	City	4	-1.57	0.0480	-1.35	0.0407	9.28	-0.2712	-1.23	0.0376	-1.38	0.0420	8.26	-0.2449	-0.44	0.0136
	Crew	3	-3.68	0.0984	-3.95	0.1049	4.06	-0.1050	-3.38	0.0882	-3.93	0.1023	2.96	-0.0769	-1.21	0.0298
	Night	5	-2.53	0.0928	-2.71	0.0982	6.32	-0.2200	-2.24	0.0791	-2.33	0.0820	6.06	-0.2093	-0.55	0.0200
	ShuttleStart	1	-0.30	0.0042	-0.01	-0.0046	13.51	-0.3522	0.71	-0.0188	0.58	-0.0133	12.57	-0.3254	-0.48	0.0089
HD Average			*-1.75*	*0.0522*	*-1.71*	*0.0505*	*8.73*	*-0.2437*	*-1.30*	*0.0391*	*-1.53*	*0.0457*	*7.77*	*-0.2179*	*-0.59*	*0.0159*
Total Average			***-3.18***	***0.1433***	***-3.27***	***0.1476***	***6.70***	***-0.2656***	***-2.52***	***0.1141***	***-2.77***	***0.1247***	***6.33***	***-0.2552***	***-0.70***	***0.0303***

TABLE 3-5
COMPARISON BETWEEN MB-AKM{(1,2,1)&(13,17,7)} AND MB-AKM{(1,2,1)&(5,7,3)} (SIGNALING BITS INCLUDED)

Sequence		Improvement over the H.264/AVC Default Scheme with Signaling Bits							
		MB-AKM{(1,2,1)&(13,17,7)}				MB-AKM{(1,2,1)&(5,7,3)}			
Size	Name	Actual Bits Saving (%)	Signaling Bits (%)	ΔBitrate (%)	ΔPSNR (dB)	Actual Bits Saving (%)	Signaling Bits (%)	ΔBitrate (%)	ΔPSNR (dB)
QCIF	Container	-0.83	0.47	-1.30	0.0624	-0.56	0.46	-1.02	0.0467
	Foreman	-3.07	0.42	-3.49	0.1713	-2.39	0.43	-2.82	0.1378
	Silent	-3.97	0.55	-4.52	0.2488	-2.88	0.57	-3.45	0.1940
QCIF Average		*-2.62*	*0.48*	*-3.10*	*0.1608*	*-1.94*	*0.49*	*-2.43*	*0.1262*
CIF	Paris	-4.35	0.22	-4.57	0.2510	-3.45	0.23	-3.68	0.2038
	Foreman	-2.41	0.42	-2.83	0.1243	-1.77	0.41	-2.18	0.0956
	Mobile	-5.03	0.20	-5.23	0.2384	-4.05	0.20	-4.25	0.1934
	Tempete	-5.89	0.24	-6.13	0.2532	-5.00	*0.23*	-5.23	0.2159
CIF Average		*-4.42*	*0.27*	*-4.69*	*0.2167*	*-3.57*	*0.27*	*-3.84*	*0.1772*
HD (720p)	BigShips	-0.29	0.38	-0.67	0.0177	0.04	0.39	-0.35	0.0092
	City	-1.27	0.30	-1.57	0.0480	-0.94	0.29	-1.23	0.0376
	Crew	-3.04	*0.64*	-3.68	0.0984	-2.76	*0.62*	-3.38	0.0882
	Night	-2.10	0.43	-2.53	0.0928	-1.81	0.43	-2.24	0.0791
	ShuttleStart	0.11	0.41	-0.30	0.0042	1.14	0.43	0.71	-0.0188
HD Average		*-1.32*	*0.43*	*-1.75*	*0.0522*	*-0.87*	*0.43*	*-1.30*	*0.0391*
Total Average		***-2.79***	***0.39***	***-3.18***	***0.1433***	***-2.13***	***0.40***	***-2.52***	***0.1141***

We also find that the employment of MB-AKM has the most impact to the CIF sequences but the least impact to the HD sequences. (The HD sequences may need the help of larger-size transforms which are outside of the scope of this thesis, to achieve higher coding efficiency improvement.) We also note that for high complexity videos, the bitrate reduction caused by employing the MB-AKM is higher in most cases.

For each sequence, we can draw a Rate-Distortion graph for various coding schemes (such as MB-AKM{(1,2,1)&(5,7,3)}, MB-AKM{(1,2,1)&IST}, etc.) to visualize the performance differences. Fig. 3-1 shows the case for the CIF Sequence Tempete which has an average bitrate reduction of 5.23% and PSNR gain of 0.22 dB. Specifically, at point (38.84 dB, 1937.63 kbits/s), the MB-AKM has a bitrate reduction of 4.7% and a PSNR gain of 0.31 dB. From this graph, we can see that the MB-AKM has a much better performance in the high bitrate region (the RHS of the RD graph) covered roughly by the right most two points generated by the QP values equal 27 and 22 respectively. Other RD graphs also have similar situations. This means that the MB-AKM is more useful when the quantization is not so coarse. It is reasonable because a fine quantization requires more accurate quantized signals of which the MSE and the number of bits required by using various kernels can be very different. The above finding indicates that the MB-AKM is suitable for high fidelity video coding applications (with small values of QP).

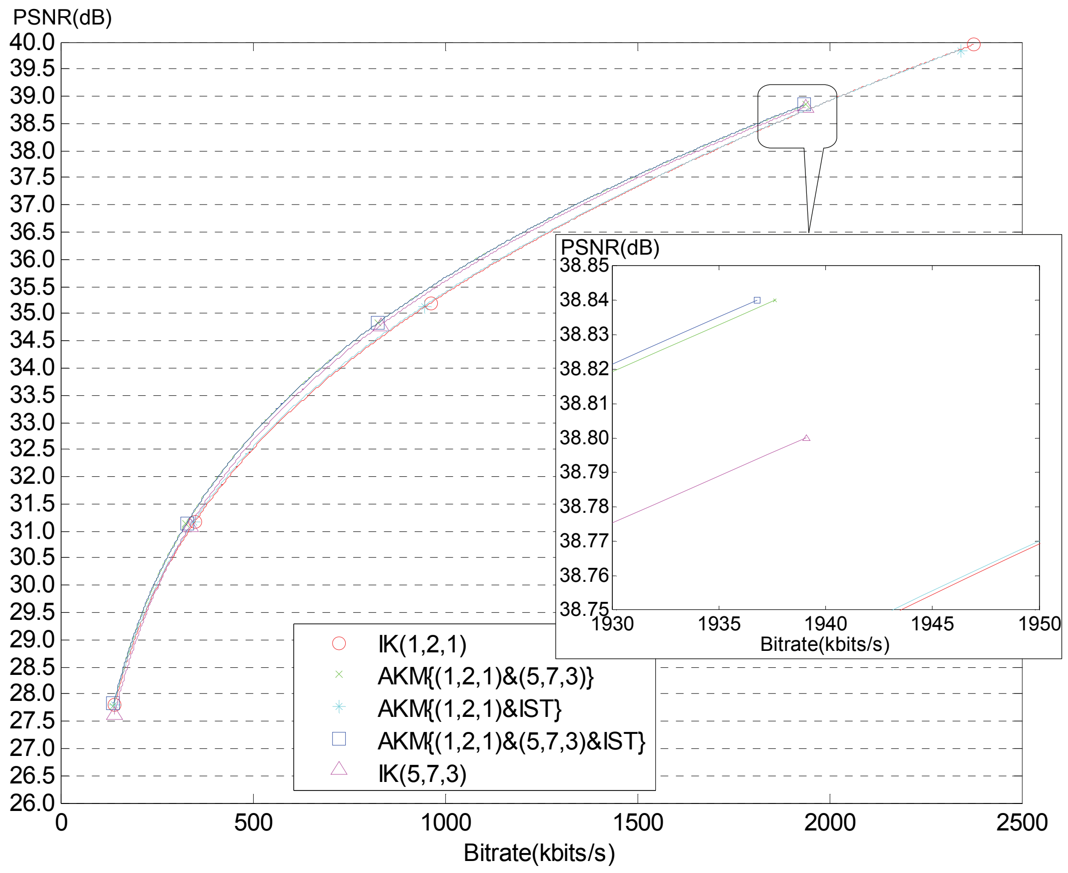


Fig. 3-1. Rate-distortion graph for CIF sequence *Tempete* relating to IK(5,7,3).

Note that the costs of signaling bits can be easily covered by the improvement made by the MB-AKM with IK(13,17,7) and IK(5,7,3). However, the improvement of MB-

AKM{(1,2,1)&IST} with the consideration of signaling bits is just marginal (with a bitrate reduction of just 0.70%).

The coding performance of MB-AKM{(1,2,1)&(5,7,3)} is close to the MB-AKM{(1,2,1)&(13,17,7)}. However, since the former one is a compromise between the complexity and the coding efficiency, its coding efficiency is slightly less efficient. With these two schemes proposed by us, one can improve the coding efficiency with the cost of raising the computational complexity. A detailed comparison of the schemes is shown in TABLE 3-6.

Roughly speaking the IK(5,7,3) requires word-lengths of 3 more bits as compared to IK(1,2,1) and the IK(13,17,7) also requires word-lengths of 3 more bits as compared to IK(5,7,3) for a practical realization, even though the coding efficiency is improved by 2.52% for employing the dual-kernel MB-AKM{(1,2,1)&(5,7,3)} and improved by 3.18% for employing the dual-kernel MB-AKM{(1,2,1)&(13,17,7)}. Computational complexity is not an issue at all if we use software with sufficiently large word-lengths and/or registers in the CPU. However, for dedicated hardware realization, every bit is expensive. This will make MB-AKM{(1,2,1)&(5,7,3)} be a better choice compared with the MB-AKM{(1,2,1)&(13,17,7)}. This point is very much application specific, and we would leave it to readers/users for making their appropriate choice.

TABLE 3-6
COMPARISONS OF SCHEMES—Δ*BITS* IS DEFINED IN (3-5) AND AVE-Δ*BITRATE* IS EXTRACTED FROM TABLE 3-4

Scheme	**Complexity (Δ*BITS* in (3-5))**	**Change of Coding Efficiency (Ave-Δ*Bitrate*)**
IK(1,2,1) only	0	0
IK(5,7,3) only	3	6.33%
IK(13,17,7) only	6	6.70%
MB-AKM{(1,2,1)&(5,7,3)}	0 and 3	-2.52%
MB-AKM{(1,2,1)&(13,17,7)}	0 and 6	-3.18%
MB-AKM{(1,2,1)&IST}	0 and 0	-0.70%

3.3. Fast Kernel Selection Strategy / Frame-Level Adaptive Kernel Mechanism

3.3.1. Kernel Selection Tendencies for Different Types of Frames

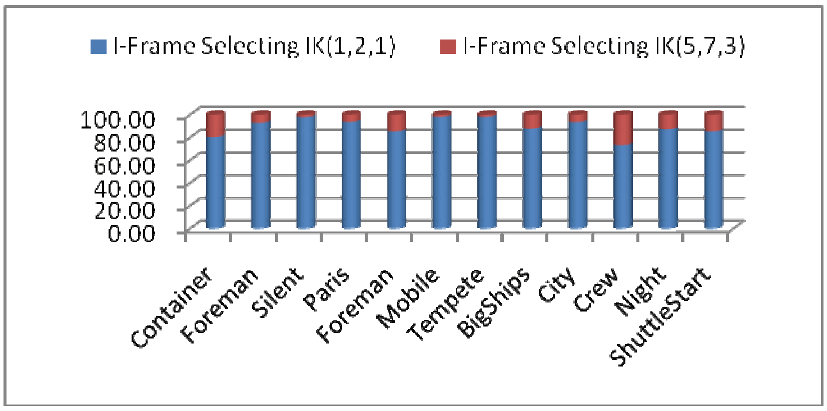
After employing the MB-AKM, we further examined the results by the types of frames, and found that there exist different kernel selection tendencies for different types of frames. As it is shown in TABLE 3-7, most macroblocks of I- and P-Frames are selected to be coded with the IK(1,2,1), while most macroblocks of B-Frames are selected to be coded with the IK(5,7,3). The statistics of the overwhelming kernels are shown in TABLE 3-7 and their counterparts are omitted since they can obviously be deduced. We can observe that the tendency for using the IK(1,2,1) is not so strong for P-Frames as the *QP* goes larger. It can be explained that as *QP* goes larger, the difference between the transformed-and-quantized signals by using different kernels is small, hence other random factors in the encoder can lower the tendency towards the IK(1,2,1). The drop of the tendency towards IK(5,7,3) is even faster for the B-Frames since the smaller residuals of B-Frame will give smaller differences between the results of IK(1,2,1) and IK(5,7,3). However, the tendency towards the IK(1,2,1) is not affected much for the I-Frame, since the residuals are the largest amongst all three types of frames. Fig. 3-2 further demonstrates that, for all sequences, the results are essentially consistent.

The above observation cannot be explained using our "common understanding" on the properties of a transform kernel. The "common understanding" refers to the general belief of most researchers in the area that the decorrelation and energy compaction properties of transform kernels are the key factors affecting the compression ratio. The following paragraph gives an explanation on the contradiction between the observation of our experimental results and the "common understanding".

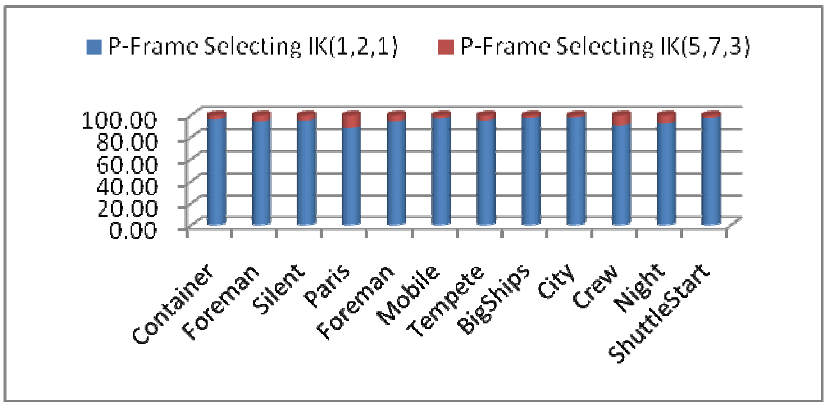
TABLE 3-7
KERNEL SELECTION TENDENCIES FOR I-, P- AND B-FRAMES

Sequence			Percentage (%)		
Size	**Name**	**QP**	**I-Frame coded with IK(1,2,1)**	**P-Frame coded with IK(1,2,1)**	**B-Frame coded with IK(5,7,3)**
QCIF	**Container**	**27**	80.5	96.1	85.0
		32	98.5	94.0	78.9
	Foreman	**27**	92.7	94.3	97.2
		32	94.5	93.4	90.1
	Silent	**27**	97.9	95.0	96.3
		32	95.8	93.2	92.6
CIF	**Paris**	**27**	93.6	88.2	99.4
		32	95.8	93.2	92.6
	Foreman	**27**	85.3	94.3	95.5
		32	89.0	92.0	82.1
	Mobile	**27**	98.2	97.0	99.8
		32	100.0	91.9	99.3
	Tempete	**27**	98.2	95.1	99.5
		32	95.8	92.7	95.1
HD (720p)	**BigShips**	**27**	87.8	97.3	73.4
		32	86.5	93.0	25.9
	City	**27**	93.5	97.8	85.0
		32	91.5	94.3	69.1
	Crew	**27**	73.0	90.5	89.1
		32	81.6	87.4	70.0
	Night	**27**	87.3	92.3	98.8
		32	86.7	89.0	84.7
	ShuttleStart	**27**	85.4	97.3	74.1
		32	82.5	95.0	47.8

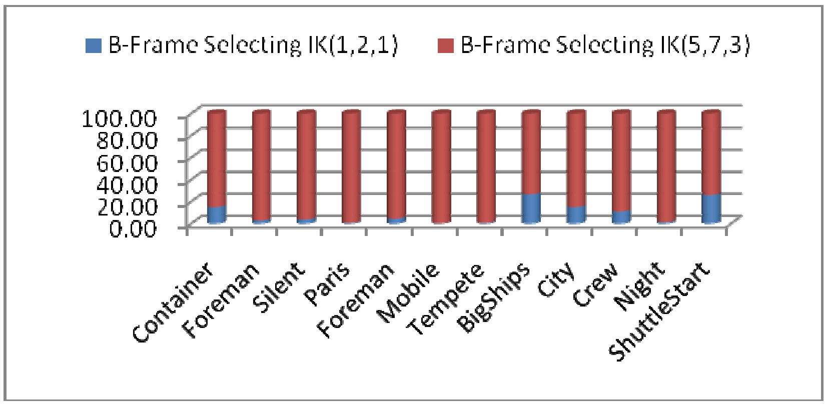
As the prediction level increases (from I to P then B), the input data fed to the DCT process become less and less correlated. The DCT-like kernel has a strong decorrelation property compared to that of the H.264/AVC Kernel. Hence, for input data with strong correlation (I-Frame data) the IK(5,7,3) is preferred (i.e. with a popularity of p_I%), while for input data with less correlation (B-Frame data) either IK(5,7,3) or IK(1,2,1) may be chosen. It is because no matter how the less correlated data are compressed with a kernel of high or low decorrelation property, the compressed results tend to be as random as the original data. Hence by considering other random factors in the encoder, either kernel may be selected. Let us denote the popularity of IK(5,7,3) for B-Frame data as p_B%, and it easy to reach the conclusion that $p_I\% > p_B\%$ which means IK(5,7,3) has a higher popularity in I-Frames.



(a)



(b)



(c)

Fig. 3-2. Stacked bar charts showing the kernel selection tendencies of (a) I-Frame, (b) P-Frame and (c) B-Frame at $QP = 27$.

3.3.2. Kernel Selection Process Using a Graphical Approach

The "common understanding" is not sufficient to explain the observations in TABLE 3-7, so let us be less prescriptive on our belief that the decorrelation and energy compaction properties be the key factors affecting the compression ratio, and make an investigation on the criteria of kernel selection. In order to explain why only one of the two kernels is overwhelmingly selected, we must investigate the kernel selection process – the Rate-Distortion Optimization by minimizing the Lagrangian Cost Function as shown below,

$$J = D + \lambda \cdot R \qquad (3\text{-}6)$$

where J is the "cost", D denotes the "distortion", R denotes the "rate" and the scalar λ is the Lagrange Multiplier. As we can see from (3-6), a large λ means a more emphasis on the issue of "rate", while a small λ means a more emphasis on the issue of "distortion".

The kernel selection process using RDO can be explained by an example as shown in Fig. 3-3. Labels P_1, P_2 and P_3 are three possible operation points that can be selected. When the criterion of selection is determined (which means that λ is a fixed value), we draw a family of parallel lines with slope $-\lambda$ passing through these operation points. The operation point, through which the line passes with the smallest intersection (J_1) on the D-axis, is the best one that we are searching for. Hence in Fig. 3-3, P_1 is the best operation point.

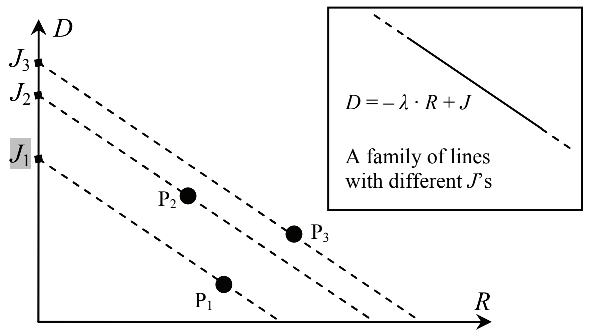


Fig. 3-3. Visualizing the kernel selection process.

By employing the MB-AKM for the H.264/AVC, each macroblock will be coded using IK(1,2,1) and IK(5,7,3), and then the best one is selected by the RDO. The operation points for a macroblock using IK(1,2,1) and IK(5,7,3) are usually at different locations on the RD plane. In order to avoid confusion when more than one macroblock are simultaneously shown on the same RD plane, we deliberately connect the operation points of the same macroblock coded by two kernels by a line segment. As it will be shown later, for two kernels with not so large performance gap, the slope of the line segment is usually a negative value. For convenience, we denote the negation of the slope of the line segment by k. For different kernel selection criteria, the relationship between λ and k differs hence the optimal operation point differs (the shaded J represents the optimal cost), which is shown in Fig. 3-4(a) and Fig.

3-4(b). More specifically, the relationship between λ and k and the subsequent selection of optimal point is concluded in TABLE 3-8.

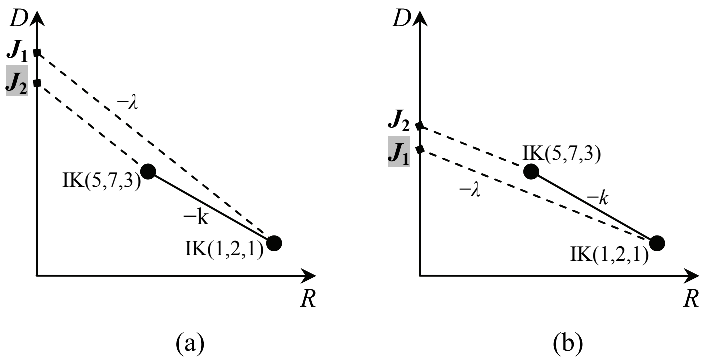


Fig. 3-4. The selection of the optimal operation point: (a) when $\lambda > k$, the optimal operation point is IK(5,7,3); and (b) when $\lambda < k$, the optimal operation point is IK(1,2,1).

TABLE 3-8
CRITERION OF OPTIMAL OPERATION POINT SELECTION

Relationship	Smallest Intersection	Optimal Operation Point
$\lambda > k$	J_2	Upper-left point – IK(5,7,3)
$\lambda < k$	J_1	Lower-right point – IK(1,2,1)

3.3.3. Rate-Distortion Feature Extracted from a Pair of Kernels

In the H.264/AVC reference software JM, the RDO for kernel selection is employed in macroblock level, and the MSE of a macroblock is the “distortion” and the number of bits of the macroblock is the “rate”. The value of λ in the H.264/AVC reference software JM is defined as

$$\lambda = s \cdot 2^{(QP-12)/3} \tag{3-7}$$

where s is the value of lambda weight and QP is the value of quantization parameter.

For the H.264/AVC, the prediction structure mainly contains three levels – I (no prediction), P (1st order prediction) and B (2nd order prediction). According to [105], in order to obtain an encoded video sequence with higher coding efficiency, under the same QP, the I-Frame should focus more on quality, while the B-Frame should focus more on using less number of bits. The actual implementation is in accordance with the above idea in the sense that the value of s usually differs from one frame type to another. A common selection suggested by the reference software JM[e] may be $s = s_{\text{I-Frame}} = 0.65$, $s = s_{\text{P-Frame}} = 0.68$ and $s = s_{\text{B-Frame}} = 2.00$, which shows the trend of a more emphasis on bitrate saving as the prediction

[e] Note the choice of λ and even of RDO is only an encoder issue and has nothing to do with the H.264/AVC Standard. However, the coding efficiency does rely intensively on the RDO and a precise model/formulation on the Lagrange multiplier which balance the quality PSNR and the bitrate. We have carried out experiments with different values of lambda weight and found the suggested values of lambda weight can roughly give optimized performances for all sequences.

level increases. Even if macroblocks of all types of frames have exactly the same contents, different λ's may select different kernels for each type of frame.

Let us visualize the performances of kernels of all macroblocks for I-, P- and B-Frames in Fig. 3-5(a), Fig. 3-5(c) and Fig. 3-5(e), respectively. In Fig. 3-5(a), each line segment represents a pair of operation points of a macroblock. In each line segment, the top-left one is the operation point of IK(5,7,3), and the bottom-right one is the operation point of IK(1,2,1). Since the contents of macroblocks differ from one to another, the line segments may span across the whole RD plane. As we can see from the graph, the line segments are almost pointing to the same direction. Note that the operation points of kernel IK(1,2,1) sit at the bottom-right and the operation points of kernel IK(5,7,3) sit at the top-left of the graphs.

Let us use Fig. 3-5(a) as an example. A red line segment with a slope of $-\lambda_{\text{I-Frame}}$ showing at the up-right region is used to determine the optimal kernel. We can do this by moving the line segment in parallel within the RD plane, to each pair of operation points. It is obvious to conclude that the bottom-right point in each pair is optimal, since a red line with a slope of $-\lambda_{\text{I-Frame}}$ passing through the bottom-right point can give a smaller intersection J on the y-axis (hence smaller RD cost). The optimality of the bottom-right point can also be verified in Fig. 3-5(b) which shows the distribution of the negation of slopes concentrating at $k = 31.61$. Note that 96.95% of the negation of slopes of the line segments of the I-Frame are larger than $\lambda_{\text{I-Frame}}$, which implies that 96.95% macroblocks be better coded with the IK(1,2,1) for the I-Frame. The steps of interpretation are also similar and valid for P- and B-Frames, and we can conclude that most macroblocks of the P- and B-Frames are better coded with the IK(1,2,1) and IK(5,7,3), respectively.

Fig. 3-6 shows the results of the MB-AKM{(1,2,1)&(13,17,7)}. We can observe that the results are similar – the default IK(1,2,1) gives better results for I- and P-Frames, and IK(13,17,7) gives better results for B-Frames. Note that the lengths of line segments in Fig. 3-6 are longer than that those in Fig. 3-5, mostly, which supports the finding in TABLE 3-4 that the MB-AKM{(1,2,1)&(13,17,7)} outperforms the MB-AKM{(1,2,1)&(5,7,3)} to some extent. Hence, with respect to TABLE 3-7, a comparison between Fig. 3-5 and Fig. 3-6 further confirm the existence of kernel selection tendency in the AKM with a pair of homogeneous DCT-like kernels.

However, Fig. 3-7 shows that the line segments formed by the operation points of the IK(1,2,1) and the IST Kernel are not pointing to one particular direction. Many operation points of the IST Kernel can be found sitting far away from those of the IK(1,2,1). Some of them have huge MSE values and there is one order magnitude higher than the MSE values of operation points of the IK(1,2,1). Hence Fig. 3-7 reveals that the kernel selection tendency does not exist in the MB-AKM{(1,2,1)&IST}.

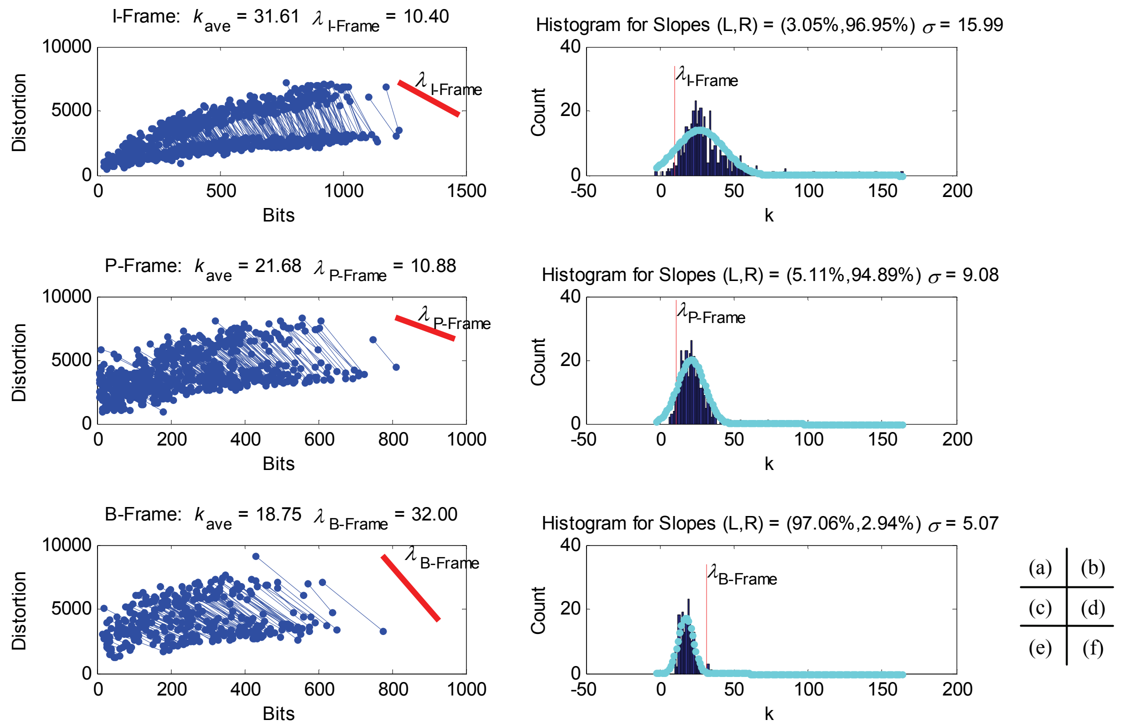


Fig. 3-5. Visualizing the performances of MB-AKM{(1,2,1)&(5,7,3)} for Sequence *Tempete* at *QP* = 24. Note that the λ's for kernel selections are shown using red color. (a) Line segments of all macroblock of I-Frame, (b) the distribution of the slopes of line segments of I-Frame, (c) line segments of all macroblock of P-Frame, (d) the distribution of the slopes of line segments of P-Frame, (e) line segments of all macroblock of B-Frame, and (f) the distribution of the slopes of line segments of B-Frame.

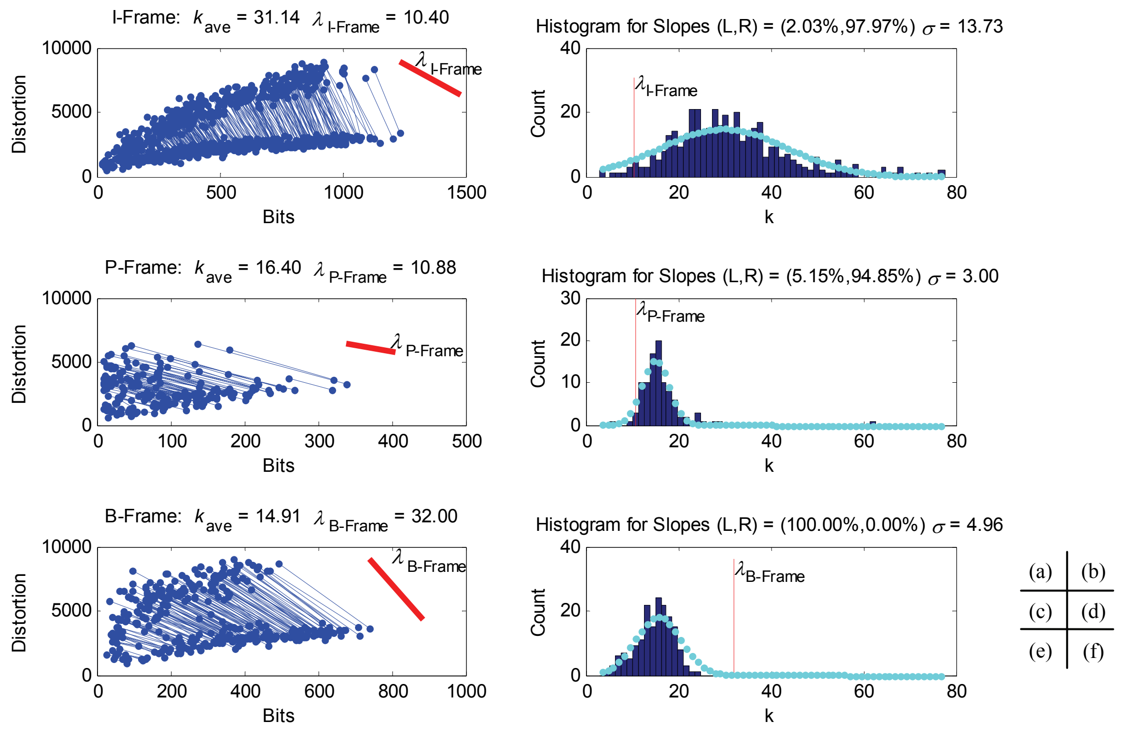


Fig. 3-6. Visualizing the performances of MB-AKM{(1,2,1)&(13,17,7)} for Sequence *Tempete* at *QP* = 24. Note that the λ's for kernel selections are shown using red color. (a) Line segments of all macroblock of I-Frame, (b) the distribution of the slopes of line segments of I-Frame, (c) line segments of all macroblock of P-Frame, (d) the distribution of the slopes of line segments of P-Frame, (e) line segments of all macroblock of B-Frame, and (f) the distribution of the slopes of line segments of B-Frame.

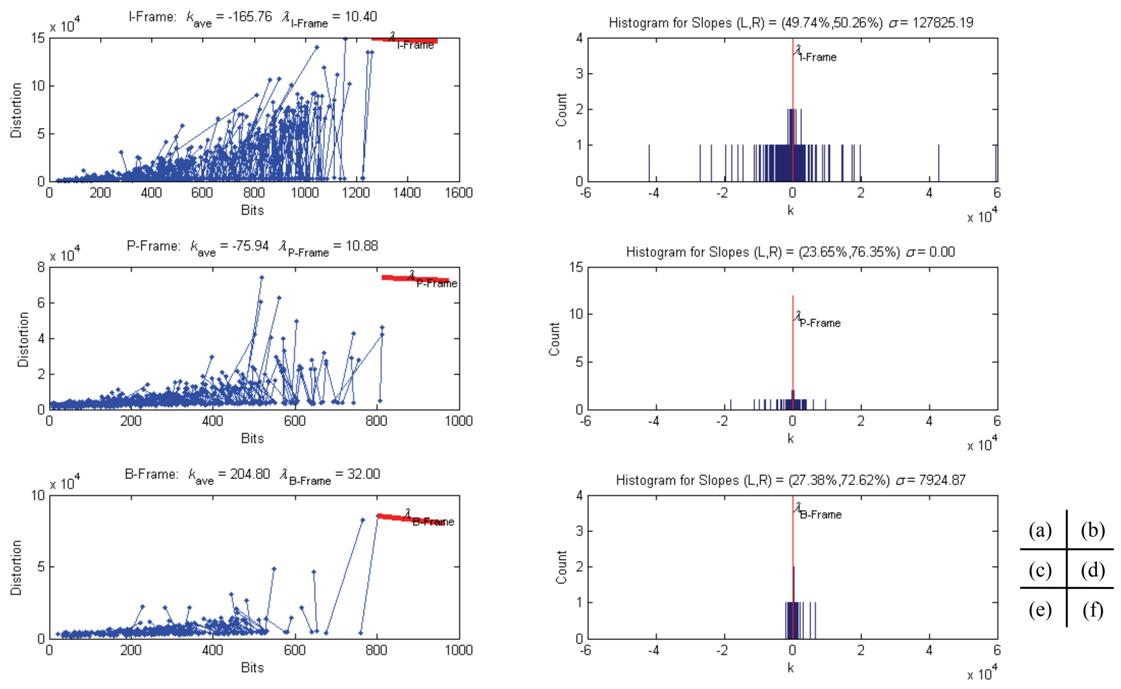


Fig. 3-7. Visualizing the performances of MB-AKM{(1,2,1)&IST} for Sequence *Tempete* at QP = 24. Note that the λ's for kernel selections are shown using red color and there is no particular direction which line segments are pointing to. (a) Line segments of all macroblock of I-Frame, (b) the distribution of the slopes of line segments of I-Frame, (c) line segments of all macroblock of P-Frame, (d) the distribution of the slopes of line segments of P-Frame, (e) line segments of all macroblock of B-Frame, and (f) the distribution of the slopes of line segments of B-Frame.

According to our testing, as QP increases (resulted in an increase in λ), k also increases, which continues to preserve the kernel selection tendencies. Using a third-order curve fitting algorithm for different video sequences and QPs, we have found the relationship between k and QP as shown below

$$k = k_0 \cdot 2^{QP/3} \tag{3-8}$$

where k_0 is a scalar. Subtracting (3-8) from (3-7), we obtain

$$\lambda - k = 2^{QP/3} \cdot \left(2^{-4} s - k_0\right). \tag{3-9}$$

We thus arrive at a crucial conclusion that whether $\lambda > k$ or $\lambda < k$ is independent of the value of QP. Hence we propose to adopt k_0 as the feature that can be extracted from a pair of kernels, only if (3-8) is fulfilled. According to a large amount of experiments we have carried out, we found that the rate-distortion feature exists when both kernels are homogeneous DCT-like kernels.

This rate-distortion feature k_0 is important because 1) it exists widely in pairs of DCT-like kernels; and 2) it is independent of QP. It can serve as a theoretical support for us to propose a scheme better than the MB-AKM for the H.264/AVC, and suggest that only one appropriate kernel be used for each type of frame, which has a unique criterion described by the value of λ.

3.3.4. Theory Behind RD Feature k_0

In this subsection, we analyze quantitatively why the rate-distortion feature k_0 exists. We also make use of Fig. 3-5 and Fig. 3-6 to justify some important observations in TABLE 3-4.

The flowchart in Fig. 3-8 illustrates the integer transform and the scaling stages of the H.264/AVC encoder. The maximum increase/decrease in the number of bits for each block has been shown using round brackets, while the number of additional bits (q bits) (see (3-5)) calculated with respect to IK(1,2,1) is shown in the square brackets. One may argue that the additional bits shift can be combined with the up scaling process so as to reduce the dynamic range during for the ease of implementation. However, we have to point out that combining q-bits shift with the up scaling stage will lead to a dyadic fraction approximation differed from its actual value as compared to our approach. This eventually leads to larger reconstruction errors, and thus lower the coding efficiency.

It is obvious to see that, at the final stage of scaling, a $(15+\lfloor QP/6\rfloor)$-bit right-shift [102] has to be done[f] to obtain the transformed-and-quantized DCT coefficients for the IK(1,2,1), and in our modification, a $(15+\lfloor QP/6\rfloor+q)$-bit right-shift has to be done to obtain the transformed-and-quantized DCT coefficients for a kernel with larger integers. Kernels with larger integers suffer from longer bit-shift (the additional q bits) operations since they embrace more non-information (larger ICT coefficients which actually bear no information) at the integer transform stage, and this arrangement implies more information will be lost for using kernels with larger integers. Hence the IK(5,7,3) coded macroblocks may have larger MSE values, which leads it to operate in a region with smaller PSNR and thus requires less bits. In other words, by choosing any line segment in Fig. 3-5(a) or Fig. 3-6(a) as an example, the operation point of IK(5,7,3) or IK(13,17,7) is situated at to the upper-left side of operation point of IK(1,2,1). It is also reasonable that the length of line segment formed by IK(1,2,1) and IK(13,17,7) is longer than that formed by IK(1,2,1) and IK(5,7,3), since a macroblock coded by a kernel with larger integers suffers more from information loss.

With the help of Fig. 3-5 and Fig. 3-6, we can also explain why the performance of a single kernel IK(5,7,3) or IK(13,17,7) performs worse than the performance of the default kernel IK(1,2,1). As we can see from Fig. 3-5, IK(5,7,3) is a good kernel for B-Frames, but is a bad kernel for I- and P-Frames. We also know that for the IBBPBBP… coding structure, the bits required for P-Frames dominants the size of the bitstream. (It is because that residuals of P-Frames are much larger than that of the B-Frames.) Hence, by using IK(5,7,3) only, although the B-Frames are efficiently encoded, the P-Frames require a large amount of bits

[f] The $(15+\lfloor QP/6\rfloor)$-bit right-shift is suggested by the evaluation software JM and has been widely used by the community, but it is not specified in the standard since it defines the items for decoding only.

(P-Frames are not efficiently encoded with the IK(5,7,3)), which means that the overall bitstream is not efficiently encoded. In other words, the drop of coding efficiency is mainly caused by the wrong selection of kernel for P-Frames. Hence, for using a single kernel, IK(1,2,1) outperforms IK(5,7,3) and IK(13,17,7) respectively, which is not a surprising result.

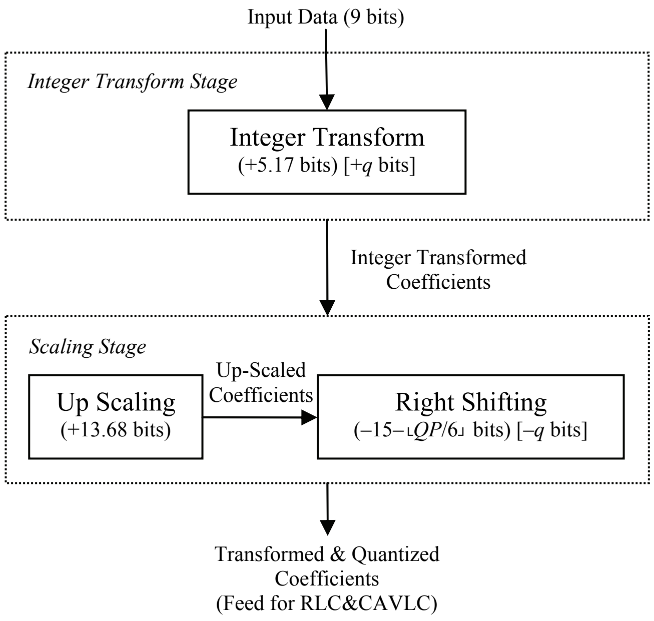


Fig. 3-8. Flowchart for transform coding and quantization. Note that $\lfloor \cdot \rfloor$ is the round-towards-zero operation.

According to the observations in Fig. 3-5 and Fig. 3-6 and the analysis in the third paragraph of this subsection, it is reasonable that the MB-AKM{(1,2,1)&(13,17,7)} outperforms MB-AKM{(1,2,1)&(5,7,3)}. However, under the single kernel arrangement, the performance of IK(5,7,3) (with a bitrate expansion of 6.33%) is better than IK(13,17,7) (with a bitrate expansion of 6.70%). Recall that the operation points of IK(13,17,7) on the RD plane are further away from the operation points of the IK(1,2,1) compared with those of the IK(5,7,3), which can be observed from a comparison between Fig. 3-5 and Fig. 3-6, it implies that the potential gain/loss over the IK(1,2,1) by using the IK(13,17,7) is larger than that of the IK(5,7,3). Hence, given the fact that both IK(5,7,3) and IK(13,17,7) cause loss in coding efficiency, we can deduce that the drop of coding efficiency by using the IK(13,17,7) is larger. We may also conclude that a kernel which can make a large improvement under the MB-AKM scheme with another alternative kernel (such as IK(1,2,1)), can possibly have a huge performance drop under the single kernel arrangement.

Our finding also implies that, when we are using homogeneous DCT-like kernels, the dominant factor that affects the coding efficiency is the Lagrange Multiplier λ.

3.3.5. Fast Kernel Selection Algorithm for H.264/AVC

Let us now turn to examine a special case – the H.264/AVC. We can now have a clear conclusion on its kernel selection. Because most kernels of the MB-AKM we used before are DCT-like kernels, one pair of kernels often has a feature which leads the encoder to select one particular kernel for most macroblocks for that type of frames as indicated in Fig. 3-5. Given that for most macroblocks the best kernel is predetermined by its frame type, it is reasonable to propose using only one particular kernel for all macroblocks of each type of frame since the computational complexity is much lowered and the overhead is removed by using a single kernel for each type of frames when the coding efficiency can mainly be maintained.

Hence, we propose a Frame-level AKM (FM-AKM) using IK(1,2,1) for I- and P-Frames, and IK(5,7,3) for B-Frames. This can be considered as a fast algorithm for the MB-AKM. We also evaluate the performance of MB-AKM{(1,2,1)&(5,7,3)} as a comparison, since this arrangement has a better performance than the MB-AKM{(1,2,1)&IST}. We again implemented our proposed FM-AKM and the MB-AKM (for reference) in JM 12.2, and our evaluation conditions were the same as those in Subsection 3.2.4 with Bjontegaard Metric employed for the measurement of both changes of Bitrate and PSNR. The evaluation results are shown in TABLE 3-9.

As we can see from the table, our MB-AKM already outperforms the H.264/AVC Default Scheme, and our proposed FM-AKM is even better for most cases. When the overheads (the signaling bits) is not taken into the consideration, the proposed method has an average PSNR increase of 0.1092 dB, while the MB-AKM has an average PSNR increase of 0.1141 dB. The bitrate saving for the proposed FM-AKM can achieve 2.40% on average, comparing to 2.52% for the MB-AKM, which shows that the performance improvement of the FM-AKM is slightly less than that of the MB-AKM[g]. However, the employment of MB-AKM introduces overheads (although it takes only about 0.40% of the overall bits of the encoded video sequence), hence the actual improvement of coding efficiency is reduced to 2.13%. So we can observe that the overall increase of coding efficiency by employing the FM-AKM is higher than the increase of coding efficiency by employing the MB-AKM, with reduced computational complexity. The MB-AKM uses IK(1,2,1) and IK(5,7,3) for each macroblock

[g] Although the overall tendency shows that the performance of the MB-AKM is better than that of the FM-AKM if signaling bits are not counted, some sequences (i.e. Foreman, Silent and ShuttleStart) are not so consistent with this tendency. In theory, the PSNR improvement of the FM-AKM cannot exceed that of the MB-AKM. However, we used the RDO as the kernel selection criterion for the MB-AKM, which may not accurately optimize the coding efficiency. Due to this uncertainty, and in a straight sense, we can only conclude that the FM-AKM outperforms the MB-AKM for most of the sequences. However, the FM-AKM still has the advantage of not requiring the signaling bits.

iteratively, while the proposed FM-AKM uses IK(1,2,1) or IK(5,7,3) for each macroblock depending on the frame type. An overall comparison of the schemes is shown in TABLE 3-10.

TABLE 3-9
COMPARISON BETWEEN PROPOSED FM-AKM AND MB-AKM

Sequence		**Improvement over the H.264/AVC Default Scheme**							
		FM-AKM{(1,2,1)&(5,7,3)} (Proposed)				**MB-AKM{(1,2,1)&(5,7,3)}**			
Size	**Name**	**Actual Bits Saving (%)**	**Signaling Bits (%)**	**ΔBitrate (%)**	**ΔPSNR (dB)**	**Actual Bits Saving (%)**	**Signaling Bits (%)**	**ΔBitrate (%)**	**ΔPSNR (dB)**
QCIF	Container	-0.63	0.00	-0.63	0.0307	-0.56	0.46	-1.02	0.0467
	Foreman	-2.93	0.00	-2.93	0.1430	-2.39	0.43	-2.82	0.1378
	Silent	-3.65	0.00	-3.65	0.2002	-2.88	0.57	-3.45	0.1940
	QCIF Average	*-2.40*	*0.00*	*-2.40*	*0.1246*	*-1.94*	*0.49*	*-2.43*	*0.1262*
CIF	Paris	-3.57	0.00	-3.57	0.1984	-3.45	0.23	-3.68	0.2038
	Foreman	-1.81	0.00	-1.81	0.0782	-1.77	0.41	-2.18	0.0956
	Mobile	-3.92	0.00	-3.92	0.1779	-4.05	0.20	-4.25	0.1934
	Tempete	-5.08	0.00	-5.08	0.2092	-5.00	*0.23*	-5.23	0.2159
	CIF Average	*-3.60*	*0.00*	*-3.60*	*0.1659*	*-3.57*	*0.27*	*-3.84*	*0.1772*
HD (720p)	BigShips	-0.34	0.00	-0.34	0.0088	0.04	0.39	-0.35	0.0092
	City	-0.89	0.00	-0.89	0.0309	-0.94	0.29	-1.23	0.0376
	Crew	-2.16	0.00	-2.16	0.0561	-2.76	*0.62*	-3.38	0.0882
	Night	-2.02	0.00	-2.02	0.0722	-1.81	0.43	-2.24	0.0791
	ShuttleStart	-0.60	*0.00*	-0.60	0.0170	1.14	0.43	0.71	-0.0188
	HD Average	*-1.20*	*0.00*	*-1.20*	*0.0370*	*-0.87*	*0.43*	*-1.30*	*0.0391*
	Total Average	**-2.40**	***0.00***	***-2.40***	***0.1092***	**-2.13**	***0.40***	***-2.52***	***0.1141***

TABLE 3-10
COMPARISONS OF SCHEMES—Δ*BITS* IS DEFINED IN (3-5) AND AVE-Δ*BITRATE* IS EXTRACTED FROM TABLE 3-4

Scheme	**Complexity (Δ*BITS* in (3-5))**	**Change of Coding Efficiency (Ave-Δ*Bitrate*)**
IK(1,2,1) only	0	0
MB-AKM{(1,2,1)&(5,7,3)}	0 or 3 for each macroblock	-2.13%
FM-AKM{(1,2,1)&(5,7,3)} (Proposed)	0 or 3 for each frame	-2.40%

3.3.6. Generalizing Rate-Distortion Feature to More Kernels

In Subsection 3.3.5, we successfully enhanced the coding efficiency by the FM-AKM with two candidate kernels. The success of raising the level of the algorithm from macroblock level to frame level is justified by the kernel selection tendencies and ultimately by the newly found rate-distortion feature k_0 between two kernels. Hence it is intuitively to make an investigation on the possibility of extracting features from even more kernels. We thus carried out experiments by plotting and examining the operation points that belong to different candidate kernels for each macroblock. Fig. 3-9 contains nine plots of operation points for different macroblocks of the I-Frame of the Sequence Tempete, at QP = 24. For each plot, it contains four operation points representing the performances of four kernels when encoding one particular macroblock using the MB-AKM. The result is very attractive as is shown in Fig. 3-9. It is observed that the set of operation points can form an essentially stable topology. For example, the encircled operation points are more likely to sit lower than other points and between the left two and right one. Hence, we may regard the stable topology formed by a set of operation points as a new feature which can be extracted from a set of DCT-like kernels. This feature is a generalization of the rate-distortion feature from a pair of kernels to a group of kernels. The feature is now generalized. It means that the kernel selection tendencies also exist when the MB-AKM with more than two candidate kernels is used. Hence the FM-AKM with *n* candidate kernels is a possible scheme provided that the kernels are DCT-like.

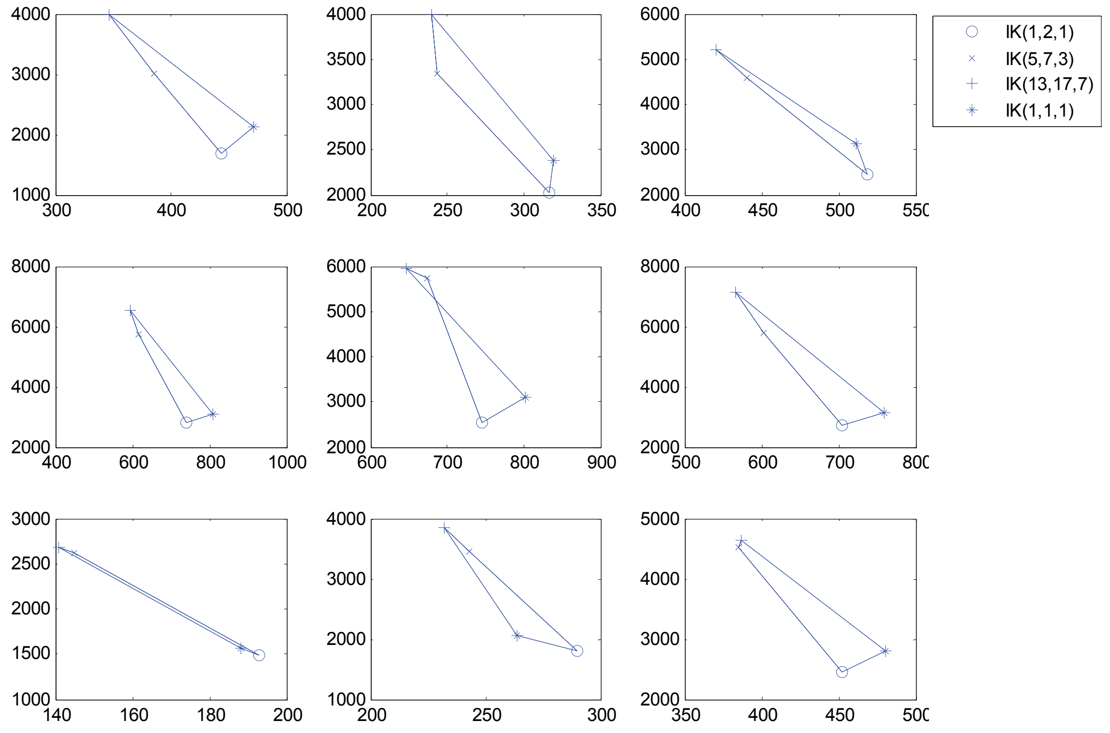


Fig. 3-9. Plots of operation points for nine different macroblocks of the I-Frame of Sequence *Tempete* at *QP* = 24.

3.3.7. Generalizing FM-AKM for Any Value of Lambda Weight

In the above subsection, we have proposed using the FM-AKM – IK(1,2,1) for I- and P-Frames and IK(5,7,3) for B-Frames. We have to stress that the proposed method is effective if users follow the suggested value of lambda weight. Although we have carried out experiments to show that the optimal value of lambda weight is in the range of 0.45–0.65 for the best overall coding efficiency, users can have the freedom to choose other values of lambda weight to cope with specific coding requirement. The FM-AKM can be generalized to the Generalized FM-AKM (GFM-AKM), so that it can adapt freely any value of the lambda weight. The feature (stable topology) is a good justification for the determination of the best kernel for each type of frames. The steps are shown as follow:

1. encode the first frame of each type using *n* candidate kernels by the RDO, and record the most popular kernel for each frame type, and
2. encode the remaining frames of each type by the most popular kernel of that frame type.

In the proposed algorithm, the preferred kernel for each frame type will only be determined by the statistics of the first frame of that frame type. This is due to the existence of the kernel selection tendency among all possible kernels, or to be justified by the feature (stable topology) of a group of operation points.

3.4. Further Remarks

We have proposed an AKM with four different alternatives. In order to apply various alternatives, we would like to make two suggestions.

3.4.1. Changes on Standard

Since the standard gives no provision on using the AKM, we suggest it allowing more kernels by reserving flag bits and adding data structures for this purpose.

The possible bits are as follows:

AKM_FLAG (1-bit): to be defined in the header of a video sequence. 0 means AKM is not used but default kernel is used. 1 means AKM is used and the decoder has to fetch the candidate kernel definitions from the header.

CANDIDATE_KERNEL_NO (8-bit): to be defined in the header of a video sequence. This is to signal how many candidate kernels have been included in the header.

CANDIDATE_KERNEL_DEFI (at most 456 bits): to be defined in the header of a video sequence. All information describes a transform coding-quantization process, including three integer kernel coefficients (one byte each), the eighteen multiplication factors (two bytes each) and eighteen rescaling factors (one byte each).

LEVEL_OF_ADAPTATION (1-bit): to be defined in the header of video sequence. 1 means the AKM has been applied to macroblock-level, so each macroblock has a SINGALING_BIT. 0 means the AKM has been applied to the frame-level, so each frame has a SINGALING_BIT.

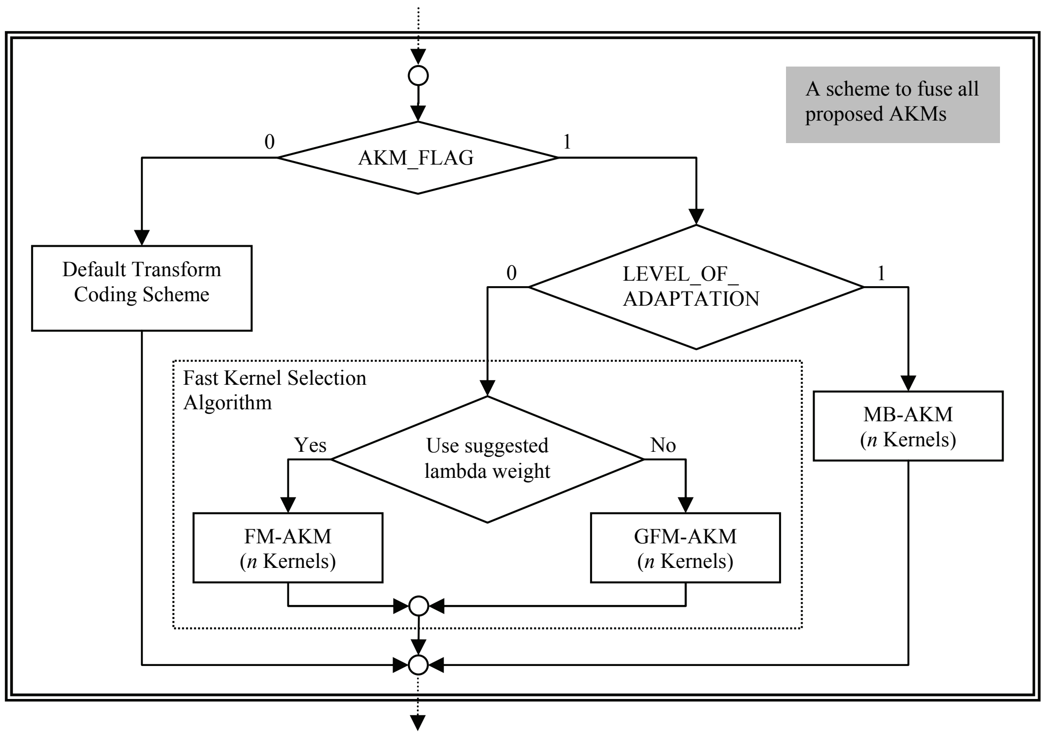
SINGALING_BIT: to be defined in the header of each GOP/frame. This is to indicate which kernel has to be used for a frame/macroblock. The signaling bits have been entropy-coded before inserting them into the header.

TABLE 3-11 shows the values of flag bits for various AKMs we have proposed and used in this chapter.

TABLE 3-11
PROPOSED FLAG BITS FOR VARIOUS AKMS

Coding Scheme	**Flags to be added to video coding standard**		
	AKM_FLAG	**CANDIDATE_ KERNEL_NO**	**LEVEL_OF_ ADAPTATION**
Default Coding Scheme	0	-	-
MB-AKM (2 Kernels)	1	2	1
FM-AKM (2 Kernels)	1	2	0
FM-AKM (*n* Kernels)	1	*n*	0
GFM-AKM (*n* Kernels)	1	*n*	0

3.4.2. Encoder Changes



A scheme to fuse all proposed AKMs
AKM_FLAG
0
1
Default Transform Coding Scheme
LEVEL_OF_ ADAPTATION
0
1
Fast Kernel Selection Algorithm
Use suggested lambda weight
Yes
No
MB-AKM (*n* Kernels)
FM-AKM (*n* Kernels)
GFM-AKM (*n* Kernels)

Fig. 3-10. Flowchart by fusing all proposed AKMs.

This is to change the encoder to fuse four AKMs with the help of the flag bits and the data structures defined in the last subsection. Details of the possible changes are shown in the Fig. 3-10. At a start a user has to initiate whether the AKM should be used. If it is not used, the default transform coding scheme which employs the IK(1,2,1) will be used. If the AKM is used, the user should specify whether the Fast Kernel Selection Algorithm (FM-AKM/GFM-AKM) or the MB-AKM should be used. If the MB-AKM is used, the encoder just encodes each macroblock *n* times (depending upon the number of candidate kernels available) and select the kernel with the smallest cost *J*. Note that the MB-AKM is a time consuming process since no prior knowledge has been applied to speed up the kernel selection process. If the Fast Kernel Selection Algorithm is used, the user has to decide whether to use the default values of lambda weight or other values for a specific coding requirement. If the user decides to use the default values of lambda weight, the default arrangement is used, i.e. to use the IK(1,2,1) for I- and P-Frames and to use the IK(5,7,3) for B-Frames. If this default is not selected, the user has to provide a list of DCT-like kernels and to let the encoder perform RD selection in the first frame of each type. After performing the kernel selection of the first frames, the preferred kernel for each type of the frames can be determined and the encoder can then code each type of the frames by a preferred kernel.

3.5. Conclusions

In this chapter, we have proposed a new DCT-like kernel IK(5,7,3) and revitalized another DCT-like kernel IK(13,17,7) to effect adaptive kernel mechanism in hybrid video coding. We have found that the macroblock-level AKM using either one of these two kernels with the H.264/AVC default kernel IK(1,2,1) outperforms the H.264/AVC's default arrangement and the MB-AKM{(1,2,1)&IST}. This is especially true for high fidelity video coding applications, i.e. with small values of QP, and the MB-AKM{(1,2,1)&(5,7,3)} which is a dual-kernel gives the best performance as compared with other single or dual-kernel approaches in the literature. However, due to the trial-and-error characteristic of the MB-AKM, the computational complexity is high. Hence, further investigations were carried out to exploit the kernel selection tendencies, aiming at the development of a fast approach for the AKM. We have found that the slope of a pair of DCT-like kernels forms an important rate-distortion feature for kernel selection. This feature is due to the differences in length of shifting bits of kernels. With the help of the rate-distortion feature analysis, we are able to propose a fast frame-level kernel selection algorithm (FM-AKM) using the IK(1,2,1) for I- and P-Frames and the IK(5,7,3) for B-Frames for the H.264/AVC. This fast approach gives comparable or even better results in terms of PSNR and bitrate compared with the new dual kernel MB-AKM{(1,2,1)&(5,7,3)}. We have also generalized the rate-distortion feature extracted from a pair of kernels to a feature extracted from a group of many kernels, which is justified by an essentially stable topology formed by operation points. We further generalize the FM-AKM to let the algorithm adapt to any value of the lambda weight to cope with specific coding requirement.

A possible future work is to expand our research into the quantitative analyses on how the lengths of shifting bits affect the rate-distortion feature (i.e. to see how the information be distributed in each word, to study more exactly the information lost due to bit-shift, etc.). Recently, researchers have shown that it is beneficial to include larger block-size transforms (such as 8×8 or 16×16 transforms) in the future standards to cope with the High Definition video contents. It is a good idea to expand our research into larger block-size transforms, and it appears to have some positive results for the 16×16 block-size transforms from our preliminary investigation. Hopefully the newly-found features can also be proved more quantitatively, which can thus benefit the kernel selection strategy for video encoding and kernel determination for future standards.

Chapter 4. Comments on “2-D Order-16 Integer Transforms for HD Video Coding”

4.1. Introduction

In Dong *et al.*’s paper [74], the authors proposed the order-16 Nonorthogonal Integer Cosine Transform (NICT) which makes a trade-off between the preservation of the characteristics of the DCT and the orthogonality property of the transform. It was proved that the reconstruction error caused by the nonorthogonality is negligible as compared to the error caused by the quantization process through a set of enlightening derivations as we have reviewed in Subsection 2.3.2. However, some problems in the derivations need to be pointed out. We also give comments on the assumption of the variance of input signals and the worse-case scenario of the upper bound of the error caused by the nonorthogonality.

4.2. Corrections

A brief review on the error analysis of the nonorthogonality of transform coding appeared in [74] is given in Subsection 2.3.2. Let us point out three problems. First, $\mathbf{TT}^{\mathrm{T}}$ should not be replaced with $\mathbf{E}_{\mathrm{r}} + \mathbf{I}$ in (2-83) since $\mathbf{TT}^{\mathrm{T}}$ does not equal $\mathbf{T}^{\mathrm{T}}\mathbf{T}$. Instead, we define

$$\mathbf{E}_{\mathrm{r}}' = \mathbf{TT}^{\mathrm{T}} - \mathbf{I}, \tag{4-1}$$

hence the results in (2-83) can be revised to (equivalent to Eqn. 14 in [74])

$$\sigma_{\mathrm{r}}^2 = \sigma_{\mathrm{q}}^2 + \sigma_{\mathrm{r0}}^2 + \tfrac{1}{N}\mathrm{E}\left[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\,\mathbf{q}\right] - \tfrac{2}{N}\mathrm{E}\left[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\,\boldsymbol{\theta}\right]. \tag{4-2}$$

This correction does not affect the final result of derivations shown in (2-84), but it does affect the intermediate steps as shown as below.

The second problem is about the derivations related to $\mathrm{E}[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\mathbf{q}] = 0$ and $\mathrm{E}[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\boldsymbol{\theta}] = 0$. Let us quote a few words from the original paper [74], “because the autocorrelation of q and the crosscorrelation of q and θ are both zero as shown by Widrow *et al.*” Recall that in Widrow *et al.*’s work [97], they only worked on the 1-d case and proved that the crosscorrelation between the quantizer input θ and its quantization noise q (a uniform independent noise) equals 0 provided that the quantizer input is “band-limited”. This result is important, yet it only exploits the crosscorrelation in 1-d case. The idea that “the autocorrelation of q equals zero” stated by the authors is not fully correct since

a) in 1-d case (a random variable q) the autocorrelation equals σ_{q}^2 at the origin and 0 elsewhere, or

b) in N-d case (a vector $\mathbf{q}$ composed of N random variables) the diagonal elements of autocorrelation matrix are the variances of random variables and the offdiagonal elements are all 0.

Ref. [74] deals with the *N*-d case, hence the result from Widrow *et al.* is not sufficient. To begin with, we convert the third and fourth terms of (4-2) into (4-3) and (4-4) respectively, because the original form is not an algebraically convenient form for analysis.

$$\mathrm{E}\left[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\,\mathbf{q}\right]=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}E_{\mathrm{r}}'(m,n)\mathrm{E}\left[q(m)q(n)\right]=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}E_{\mathrm{r}}'(m,n)R_{qq}(m,n)\quad\text{(4-3)}$$

$$\mathrm{E}\left[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\,\boldsymbol{\theta}\right]=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}E_{\mathrm{r}}'(m,n)\mathrm{E}\left[q(m)\theta(n)\right]=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}E_{\mathrm{r}}'(m,n)R_{q\theta}(m,n)\quad\text{(4-4)}$$

Note that $R_{qq}(m,n)$ is the (m,n)th element of the autocorrelation matrix of vector $\mathbf{q}$ which is composed of N random variables. The nth diagonal element is the variance of the nth random variable, whereas elements are zero elsewhere since each of them is the crosscorrelation of two different random variables (for example, $q(1)$ and $q(4)$ are two uniform independent noise processes). Hence (4-3) can be reduced to

$$\mathrm{E}[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\mathbf{q}] = \sigma_{\mathrm{q}}^{2}\textstyle\sum_{i} E_{\mathrm{r}}'(i,i). \quad\text{(4-5)}$$

By a careful examination on (4-1), it is observed that all diagonal terms ($E_{\mathrm{r}}'(i,i)$ for all i) are strictly zero, since $\mathbf{T}$ is an orthogonal matrix. Hence (4-3) is reduced to $\mathrm{E}[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\mathbf{q}] = 0$. Similarly, $R_{q\theta}(m,n)$ is the (m,n)th element of the matrix of the crosscorrelation between $\mathbf{q}$ and $\boldsymbol{\theta}$. The nth diagonal element is the crosscorrelation between quantization noise $q(n)$ and relative quantizer input $\theta(n)$. As we have mentioned before, by Widrow *et al.*'s work, $\mathrm{E}[q(n)\theta(n)] = 0$ in 1-d case. Hence we can approximate the diagonal elements to zero. The offdiagonal elements can also be approximated to zero. Each of them measures the crosscorrelation between the quantization noise of one quantizer $q(m)$ and the input of another quantizer $\theta(n)$. It is widely accepted that the DCT coefficients can be treated as uncorrelated random variables. Now that the inputs of two quantizers are not correlated, the input of one quantizer is also uncorrelated with the quantization noise of another quantizer. Hence the offdiagonal entries are also zero, and (4-4) can be further reduced to

$$\mathrm{E}[\mathbf{q}^{\mathrm{T}}\mathbf{E}_{\mathrm{r}}'\boldsymbol{\theta}] = 0. \quad\text{(4-6)}$$

We can thus conclude that the average variance of the reconstruction (4-2) is reduced to

$$\sigma_{\mathrm{r}}^{2} = \sigma_{\mathrm{q}}^{2} + \sigma_{\mathrm{r0}}^{2}, \quad\text{(4-7)}$$

as stated in (2-84).

Third, it is inappropriate to conclude that the nonorthogonal order-16 kernels listed in TABLE III of [74] has negligible nonorthogonality error by simply evaluating the upper bound of σ_{r0}^{2} for their submatrices ($\mathbf{T}_{8\mathrm{o}}$'s), instead, the upper bound for the order-16 kernels ($\mathbf{T}_{16}$'s) themselves should to be directly measured and then compared with the quantization error to show whether the former one can be neglected. By using the same settings as in [74], i.e. $\sigma_{x}^{2} = 1$, we have obtained the results for $\mathbf{T}_{16}$'s as shown in column "Corrected" of TABLE I. It is observed that the errors measured for the order-16 kernels ($\mathbf{T}_{16}$'s) are much smaller than that for their submatrices ($\mathbf{T}_{8\mathrm{o}}$'s). It is reasonable since the other submatrix ($\mathbf{T}_{8\mathrm{e}}$) is orthogonal comparing to the nonorthogonal submatrix ($\mathbf{T}_{8o}$) hence the error can be diluted significantly.

TABLE 4-1
UPPER BOUND OF AVERAGE VARIANCE OF RECONSTRUCTION ERROR WITHOUT THE CONSIDERATION OF QUANTIZATION

NICT Matrix Elements								Upper Bound of a) σ_{r0}^2 by setting $\sigma_x^2 = 1$, or b) $\sigma_{r0}^2 / \sigma_x^2$		
								For $\mathbf{T}_{8o}$	For $\mathbf{T}_{16}$	
x_1	x_3	x_5	x_7	x_9	x_{11}	x_{13}	x_{15}	Dong *et al.*	Corrected[h]	Proposed_wc
28	27	23	21	17	14	8	2	1.0849×10^{-6}	$0\ (-2.7967\times10^{-23})$	2.3868×10^{-6}
29	28	26	22	20	13	10	2	8.3628×10^{-7}	$0\ (-3.1019\times10^{-23})$	9.1990×10^{-7}
38	36	35	29	25	18	9	3	7.6103×10^{-7}	$0\ (-1.1410\times10^{-23})$	3.3368×10^{-6}
39	37	35	29	26	18	11	2	7.0311×10^{-7}	2.0913×10^{-23}	1.4603×10^{-6}
40	39	33	31	24	19	14	4	1.8527×10^{-6}	8.3438×10^{-23}	2.6789×10^{-6}
40	38	35	31	24	19	11	4	6.5425×10^{-7}	1.9439×10^{-23}	1.0820×10^{-6}

4.3. Comments

Besides, let us give two more comments. First, the assumption of the input signal as a unit-variance Markov Process is inappropriate. It is worth to recall that residual signals of the H.264/AVC encoder are being modeled, and these real data can hardly have a unit-variance. Actually, even if the variance of the input signal is not unit, the representation of the upper bound in (2-81) derived in [74] is still valid. We carried out a set of experiments to find the variances of input signals. It is found that the variances are of the order 10^2 to 10^3. By substituting these variances into (2-81) making the use of column "Corrected" in TABLE 4-1, we found that the errors caused by the nonorthogonal transform are in the order of 10^{-21} to 10^{-20} which are still much smaller than that caused by the quantization. This conclusion can be made safely by using the real variance of input signals.

Second, let us propose a better method compared to the method as shown in (2-81) to find the upper bound. In (2-81), one contributing factor of the upper bound is obtained by summing up all $M(k,j)$'s which may be a positive or negative number. Hence, the cancellation effect occurs and one may wonder whether the assumption can reach the real worse-case scenario. What is the result if $R(k,j)$ approximates infinitely close to σ_x^2 and has the same sign with $M(k,j)$? This leads to the largest upper bound. To exploit this possibility, we do not make the assumption of wide-sense stationary (i.e. $M(k,j) \neq M(k-j, 0)$), which can give a flexible selection of the values of $M(k,j)$, and make the analysis using the Markov Model.

It is well-known that a zero-mean AR(1)-process can be denoted as $x(n) = \rho(n)x(n-1) + \varepsilon(n)$ ($|\rho(n)| < 1$) where $x(n)$ and $x(n-1)$ are two adjacent random variables, $\rho(n)$ is the correlation coefficient between $x(n)$ and $x(n-1)$ and $\varepsilon(n)$ is a white noise process with zero

[h] The upper bounds of variance are set to zero for any negative value generated by the oversimplified and imprecise model as shown in (2-81).

mean. By removing the assumption of wide-sense stationary, the correlation between two variables $x(i)$ and $x(j)$ $(i<j)$ can be calculated as

$$\begin{aligned}
R(i,j) &= \mathrm{E}\left[x(i)x(j)\right] \\
&= \mathrm{E}\left\{x(i)\cdot\left[\rho(j)\left[\rho(j-1)\left[\cdots\left[\rho(i+1)x(i)+\varepsilon(i+1)\right]\cdots\right]+\varepsilon(j-1)\right]+\varepsilon(j)\right]\right\} \\
&= \mathrm{E}\begin{bmatrix}\left[\rho(j)\rho(j-1)\cdots\rho(i+1)\right]\left[x(i)\right]^2+\left[\rho(j)\rho(j-1)\cdots\rho(i+2)\right]\varepsilon(i+1)x(i) \\ +\cdots+\left[\rho(j)\rho(j-1)\right]\varepsilon(j-2)x(i)+\rho(j)\varepsilon(j-1)x(i)+\varepsilon(j)x(i)\end{bmatrix} \\
&= \left[\rho(j)\rho(j-1)\cdots\rho(i+1)\right]\mathrm{E}\left\{\left[x(i)\right]^2\right\}+\left[\rho(j)\rho(j-1)\cdots\rho(i+2)\right]\mathrm{E}\left[\varepsilon(i+1)\right]\mathrm{E}\left[x(i)\right] \\
&\quad +\cdots+\left[\rho(j)\rho(j-1)\right]\mathrm{E}\left[\varepsilon(j-2)\right]\mathrm{E}\left[x(i)\right]+\rho(j)\mathrm{E}\left[\varepsilon(j-1)\right]\mathrm{E}\left[x(i)\right]+\mathrm{E}\left[\varepsilon(j)\right]\mathrm{E}\left[x(i)\right] \\
&= \left[\rho(j)\rho(j-1)\cdots\rho(i+1)\right]\sigma_i^2
\end{aligned} \tag{4-8}$$

where σ_i^2 is the variance of $x(i)$. It is seen that $R(i,j)$ is determined by a set of correlation coefficients. Let us denote $\rho(j)\cdot\rho(j-1)\cdot\ldots\cdot\rho(i+1)$ by $\rho(i,j)$. The worse-case scenario can be reached when 1) $|\rho(i,j)|\to 1$, and 2) $sign[\rho(i,j)] = sign[M(i,j)]$. The first condition is easily fulfilled by setting the absolute value of all correlation coefficients to 1. The second condition can only be fulfilled for some positions. It is because the number of sign patterns of $M(i,j)$ is $2^{(N-1)^2/2}$, whereas the number of sign patterns of $\rho(i,j)$ is only 2^{N-1}. Hence a computer search for all sign patterns of $\rho(i,j)$ can be carried out to find the best one which leads to the worst-case scenario. When the variance of noise approximates zero, we can use σ_x^2 instead of σ_i^2 for $i\in[1,n]$. The proof is shown below

$$\begin{aligned}
\mathrm{E}\left\{\left[x(n)\right]^2\right\} &= \mathrm{E}\left\{\left[\rho(n)x(n-1)+\varepsilon(n)\right]^2\right\} \\
&= \mathrm{E}\left\{\left[\rho(n)x(n-1)\right]^2+2\rho(n)\varepsilon(n)x(n-1)+\left[\varepsilon(n)\right]^2\right\} \\
&= \left[\rho(n)\right]^2\mathrm{E}\left\{\left[x(n-1)\right]^2\right\}+2\rho(n)\mathrm{E}\left[\varepsilon(n)\right]\mathrm{E}\left[x(n-1)\right]+\mathrm{E}\left\{\left[\varepsilon(n)\right]^2\right\}. \\
&\Rightarrow \sigma_n^2=\left[\rho(n)\right]^2\sigma_{n-1}^2+\sigma_\varepsilon^2\approx\sigma_{n-1}^2 \\
&\Rightarrow \sigma_n^2=\sigma_{n-1}^2=\cdots=\sigma_1^2=\sigma_x^2
\end{aligned} \tag{4-9}$$

Hence the autocovariance matrix can be written as

$$\mathbf{R}=\sigma_x^2\begin{bmatrix}
1 & \rho(1) & \rho(2)\rho(1) & \rho(3)\rho(2)\rho(1) & \cdots & \rho(N-1)\cdot\cdot\rho(3)\rho(2)\rho(1) \\
\rho(1) & 1 & \rho(2) & \rho(3)\rho(2) & \cdots & \rho(N-1)\cdot\cdot\rho(3)\rho(2) \\
\rho(2)\rho(1) & \rho(2) & 1 & \rho(3) & \cdots & \rho(N-1)\cdot\cdot\rho(3) \\
\rho(3)\rho(2)\rho(1) & \rho(3)\rho(2) & \rho(3) & 1 & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\rho(N-1)\cdot\cdot\rho(3)\rho(2)\rho(1) & \rho(N-1)\cdot\cdot\rho(3)\rho(2) & \rho(N-1)\cdot\cdot\rho(3) & \cdots & \cdots & 1
\end{bmatrix} \tag{4-10}$$

Actually the worst-case scenario is reached when any two of random variable $x(n)$'s have exactly the same observation (ρ = 1) or neighboring observations have exactly the opposite signs (ρ = −1). This can be infinitively approximated, but cannot be reached. Through computer searches for various kernels, we obtained new results as shown in the column

"Proposed_wc" of TABLE 4-1 in this chapter. The best sign patterns of $\rho(i,j)$'s considered are not shown here for the sake of simplicity. Although the number of computer search is small compared to the possible number of sign patterns, it still needs 2^{15} different sign patterns for a length-16 input signal.

4.4. Conclusion

Dong *et al.*'s paper is a comprehensive and instructive work on integer transform kernel design for the HD video coding. In spite of some minor (but yet useful) problems pointed out by us, the paper is a piece of excellent work that can give readers inspiration, and for reference of future work. We have also proposed an improved method for the evaluation of the upper bound of the reconstruction error. Hence, this makes the original more complete and a more real worst-case scenario can be attained.

Chapter 5. Analysis on Dyadic Approximation Error for Hybrid Video Codecs with Integer Transforms

5.1. Introduction

In the latest video coding standard, the H.264/AVC [6], one of the integer variations of the DCT constructed using the up-scaling and rounding method is adopted to solve the possible mismatch between encoder and decoder by removing the floating-point arithmetic, as we have mentioned in Subsection 2.3.1. However, when using this method to design an integer kernel of the size 16×16 (or an order-16 integer kernel), the similarity to the original DCT Kernel and the easy-to-implement property need always to be compromised—one way to achieve both merits is to relax the condition of orthogonality of transform kernel in a controllable manner which roughly ensures the orthogonality, and the error analysis showed that the well-controlled nonorthogonality noise is approximately negligible as compared to the quantization noise, as we have mentioned in Subsection 2.3.2.

However, the adoption of integer transform kernel introduces another problem. The removed floating-point arithmetic is actually incorporated into the quantization / dequantization, and the combined process is called the post-scaling / pre-scaling [106]. The floating-point numbers in the scaling stages are approximated by dyadic fractions, and the floating-point multiplications are replaced with integer multiplications followed by bit-wise right-shifting operations. The dyadic approximation thus introduces a new kind of error beyond the quantization error [97, 107] and nonorthogonality error [74, 91], which has not been quantitatively investigated. Let us call it the "dyadic approximation error". It exists at both the encoder side and the decoder side and has a value of several orders higher than the nonorthogonality error in terms of variance measurement. The impact of this error is position dependent in the compressed domain (DCT domain) which eventually affects the visual quality in the spatial domain. The dyadic approximation also introduces a system error between raw residual signal and the reconstructed residual signal regardless of the quantization effect. Even for some cases when the system error is compensated due to a perfect match between the dyadic fractions of the encoder side and the decoder side, our finding suggests that it is just quasi–error-free. It is equivalent to tune the quantizer step for DCT coefficients of each position up/down slightly, which results in an undesired effect that the DCT coefficients are quantized by a non-uniform quantization matrix. Fortunately, the dyadic approximation error can be avoided by using more shifting bits.

Before this chapter in Subsection 2.3.3, the analysis of dyadic approximation error in one-dimension is carried out, and analytical representations of different kinds of errors are also given. In Section 5.2, we extent the analysis into two-dimension, evaluate and compare the significances of different error terms using a set of available kernels. In Section 5.3, the characteristics of the dyadic approximation error are explored and the lower bound of number of shifting bits to avoid this error is obtained. Finally, Section 5.4 concludes this chapter.

5.2. 2-D Analysis on Dyadic Approximation Error

In Subsection 2.3.3, we have derived the analytical representation of reconstruction error by considering quantization, nonorthogonality and dyadic approximation in one-dimension (1-d). In this section, we extend the analysis into two-dimensional (2-d) case such as image and video coding, and we will give examples on the latter one. We use different order-16 nonorthogonal integer transform kernels ([74, 90]) as well as orthogonal integer transform kernels ([74, 88-89, 108]) for evaluation. Due to the limitation of space, only the experimental results obtained by using the following order-16 nonorthogonal kernel [74] are presented

$$
\mathbf{H} = \begin{bmatrix}
32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 \\
40 & 38 & 35 & 31 & 24 & 19 & 11 & 4 & -4 & -11 & -19 & -24 & -31 & -35 & -38 & -40 \\
40 & 36 & 24 & 8 & -8 & -24 & -36 & -40 & -40 & -36 & -24 & -8 & 8 & 24 & 36 & 40 \\
38 & 24 & 4 & -19 & -35 & -40 & -31 & -11 & 11 & 31 & 40 & 35 & 19 & -4 & -24 & -38 \\
40 & 16 & -16 & -40 & -40 & -16 & 16 & 40 & 40 & 16 & -16 & -40 & -40 & -16 & 16 & 40 \\
35 & 4 & -31 & -38 & -11 & 24 & 40 & 19 & -19 & -40 & -24 & 11 & 38 & 31 & -4 & -35 \\
36 & -8 & -40 & -24 & 24 & 40 & 8 & -36 & -36 & 8 & 40 & 24 & -24 & -40 & -8 & 36 \\
31 & -19 & -38 & 4 & 40 & 11 & -35 & -24 & 24 & 35 & -11 & -40 & -4 & 38 & 19 & -31 \\
32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 \\
24 & -35 & -11 & 40 & -4 & -38 & 19 & 31 & -31 & -19 & 38 & 4 & -40 & 11 & 35 & -24 \\
24 & -40 & 8 & 36 & -36 & -8 & 40 & -24 & -24 & 40 & -8 & -36 & 36 & 8 & -40 & 24 \\
19 & -40 & 24 & 11 & -38 & 31 & 4 & -35 & 35 & -4 & -31 & 38 & -11 & -24 & 40 & -19 \\
16 & -40 & 40 & -16 & -16 & 40 & -40 & 16 & 16 & -40 & 40 & -16 & -16 & 40 & -40 & 16 \\
11 & -31 & 40 & -35 & 19 & 4 & -24 & 38 & -38 & 24 & -4 & -19 & 35 & -40 & 31 & -11 \\
8 & -24 & 36 & -40 & 40 & -36 & 24 & -8 & -8 & 24 & -36 & 40 & -40 & 36 & -24 & 8 \\
4 & -11 & 19 & -24 & 31 & -35 & 38 & -40 & 40 & -38 & 35 & -31 & 24 & -19 & 11 & -4
\end{bmatrix}
$$

As we will discover soon, the experimental results for different kernels are similar. Note that $\mathbf{H}_{\text{int}}$'s normalized form $\mathbf{H}$, which is needed for the following analysis, can be obtained by normalizing each row vector.

To begin with, let us define the input signal as an $N \times N$ matrix $\mathbf{X}$, hence the transformed matrix is $\mathbf{Z} = \mathbf{H}\mathbf{X}\mathbf{H}^{\mathrm{T}}$. In order to simplify the derivation, matrices $\mathbf{X}$ and $\mathbf{Z}$ can be lexicographically ordered into column vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ of length N^2, where $\boldsymbol{x}(i + jN) = X(i,j)$ and $\boldsymbol{z}(i + jN) = Z(i,j)$ for $i, j \in [0, N-1]$. We convert the 2-d transform into the lexicographical form $\boldsymbol{z} = \boldsymbol{\mathcal{H}}\boldsymbol{x}$ where $\boldsymbol{\mathcal{H}}$ is the Kronecker product of $\mathbf{H}$ with itself, i.e. $\boldsymbol{\mathcal{H}} = \mathbf{H} \otimes \mathbf{H}$ [109]. By considering the dyadic approximation error at the forward transform stage, we multiply the transform kernel $\mathbf{H}$ by the scalar matrix $\mathbf{S}_1$. Hence, the distorted transformed signal $\boldsymbol{u}$ in lexicographical form can be formulated as $\boldsymbol{u} = \left[(\mathbf{S}_1\mathbf{H}) \otimes (\mathbf{S}_1\mathbf{H})\right]\boldsymbol{x} = (\mathbf{S}_1 \otimes \mathbf{S}_1)\ (\mathbf{H} \otimes \mathbf{H})\ \boldsymbol{x}$

$= \mathcal{S}_1\mathcal{H}\boldsymbol{x}$ where $\mathcal{S}_1 = \mathbf{S}_1 \otimes \mathbf{S}_1$. It can be seen that $\boldsymbol{v} = \mathcal{S}_1\mathcal{H}\boldsymbol{x}$ shares exactly the same form with the combination of (2-88) and (2-89) except that the order has been raised from N to N^2. Hence, we rewrite (2-85)–(2-91) for the 2-d case by simply changing the variables from the typewritten form into the script form, as shown below

$$\boldsymbol{z} = \mathcal{H}\boldsymbol{x} \tag{5-1}$$

$$\boldsymbol{v} = \mathcal{S}_1\boldsymbol{z} \tag{5-2}$$

$$\boldsymbol{u} = quant^{-1}[quant(\boldsymbol{v})] = \boldsymbol{v} - \boldsymbol{q} \tag{5-3}$$

$$\boldsymbol{y}_{\mathrm{T}} = (\mathcal{S}_2\mathcal{H})^{\mathrm{T}}\boldsymbol{u} \tag{5-4}$$

$$\mathcal{N}_1 = \mathcal{N}_1^{\mathrm{T}} = \mathcal{S}_1^{-1} - \mathrm{I} \tag{5-5}$$

$$\mathcal{N}_2 = \mathcal{N}_2^{\mathrm{T}} = \mathcal{S}_2 \ - \mathrm{I} \tag{5-6}$$

$$\mathcal{E}_{\mathcal{H}} = \mathcal{E}_{\mathcal{H}}^{\mathrm{T}} = \mathcal{H}\mathcal{H}^{\mathrm{T}} - \mathrm{I} \tag{5-7}$$

where the definitions of symbols are exactly the same w.r.t. their 1-d counterparts, hence we can omit the definitions for the sake of simplicity. By going through the derivations in Subsection 2.3.3 once again, we obtain similar representation of the average variance of reconstructed signal, as shown below

$$\sigma_{\boldsymbol{q}}^2 + \sigma_{\boldsymbol{x}}^2\left[\frac{1}{N^2}\sum_{j=0}^{N^2-1}\sum_{i=0}^{N^2-1}\left|\mathcal{M}(i,j)\right|\right] + \left\{\sigma_{\boldsymbol{q}}^2\left[\frac{1}{N^2}\sum_{i=0}^{N^2-1}\mathcal{W}(i,i)\right] + \sigma_{\boldsymbol{x}}^2\left[\frac{1}{N^2}\sum_{j=0}^{N^2-1}\sum_{i=0}^{N^2-1}\left|\mathcal{Y}(i,j)\right|\right]\right\} \tag{5-8}$$

where

$$\mathcal{M} = (\mathcal{H}^{\mathrm{T}}\mathcal{H} - \mathrm{I})^{\mathrm{T}}(\mathcal{H}^{\mathrm{T}}\mathcal{H} - \mathrm{I}), \tag{5-9}$$

$$\mathcal{W} = \mathcal{N}_2^{\mathrm{T}}\mathcal{E}_{\mathcal{H}}\mathcal{N}_2 + \mathcal{N}_2^{\mathrm{T}}\mathcal{N}_2 + \mathcal{E}_{\mathcal{H}} + 2\mathcal{E}_{\mathcal{H}}\mathcal{N}_2 + 2\mathcal{N}_2\text{, and} \tag{5-10}$$

$$\mathcal{Y} = \mathcal{H}^{\mathrm{T}}\mathcal{S}_1\left[\mathcal{N}_1 - \mathcal{N}_2 - \left(\mathcal{N}_1 + \mathcal{N}_2 + 2\mathbf{I}\right)\mathcal{E}_{\mathcal{H}}\right]\left(\mathcal{N}_1 - \mathcal{N}_2\right)\mathcal{S}_1\mathcal{H}. \tag{5-11}$$

Equation (5-8) is composed of the quantization term, the nonorthogonality term, and the dyadic approximation term. To examine the significance of each term (i.e. the importance of each term in percentage), we need to find out the variables affecting each term for one particular transform kernel. For example, the quantizer step q directly affects $\sigma_{\boldsymbol{q}}^2$ where $\sigma_{\boldsymbol{q}}^2 = q^2/12$ for uniformly distributed quantizer input[i]. The average variances $\sigma_{\boldsymbol{x}}^2$'s are measured at different q's and from sequences of different resolutions. The value of $\sigma_{\boldsymbol{x}}^2$ is in the order of 10^1 to 10^2 which closely depends on the motion complexity of a video sequence. Let us use the measurement results of a 720p (1280×720) video sequence *City* as shown in TABLE 5-1. Note that the input signal $\boldsymbol{x}$ is actually the residual signal rather than the original

[i] Although the quantizer input (i.e. the DCT coefficients) obeys the Laplacian Distribution, we have verified that the quantization noise power in variance form can be approximated by using a uniformly distributed noise process. The true and approximated values are basically in the same order.

signal of a frame, hence its value depends on the feedback loop of the hybrid video coding and thus is effectively affected by the quantizer step q. Hence quantizer step q is the variable of both $\sigma_{\boldsymbol{q}}^2$ and $\sigma_{\boldsymbol{x}}^2$. Matrix $\mathcal{M}$ is independent of all variables since we use one particular transform kernel. $\mathcal{W}$ and $\mathcal{Y}$ depend very much on the dyadic approximation error matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ which are the models created to reflect the dyadic approximation errors at the encoder side and decoder side. Recall that the number of shifting bits n and the quantizer step q are closely related to the dyadic approximation error equivalent model, hence n and q are the variables of $\mathcal{W}$ and $\mathcal{Y}$. The significances of the three terms in (5-8) are affected by the quantizer step and the number of shifting bits, and the latter one relates to the question on the number of shifting bits should be used for a 16×16 integer transform kernel so that the impact of the dyadic approximation error can be omitted. We thus calculated the value of three terms by varying the quantizer step and the number of shifting bits. The smallest possible number of shifting bits at the encoder side and decoder side are n_1 = 19 and n_2 = 13 respectively, according to our experiments, and they were increased to n_1 = 25 and n_2 = 19 with stepsize 1 to achieve more accurate dyadic approximations. The results are shown in TABLE 5-2. We can see that the nonorthogonality error has the lowest weight (0–0.1%) in all cases, hence it needs no further consideration in the following analysis. The remaining two kinds of errors—the quantization error and the dyadic approximation error—are comparable when the quantizer step is small or the number of shifting bits is small. Fig. 5-1 shows the dyadic approximation error over the reconstruction error in percentage w.r.t. the quantizer step q and the number of additional shifting bits Δn (by assuming that the number of additional shifting bits for the encoder and decoder sides are the same). It is observed that when Δn reaches a certain level (i.e. 5), the impact of the dyadic approximation error can be omitted. Similar results were obtained for other order-16 orthogonal/nonorthogonal integer kernels, however, they are not shown here due to the limitation of space.

TABLE 5-1
AVERAGE VARIANCES OF INPUT SIGNAL $\boldsymbol{x}$

Quantization Parameter in the H.264/AVC Reference Software	q	σ_x^2
4	1	16.0
18	5	16.3
24	10	19.1
36	40	52.1

TABLE 5-2

THE SIGNIFICANCES OF DIFFERENT KINDS OF ERRORS

Quantizer Step	Average Variances		No of Shifting Bits		Quantization Error		Nonorthogonality Error		Dyadic Approximation Error		Reconstruction Error (Total Error)
Q	σ_x^2	σ_q^2	Encoder Side n_1	Decoder Side n_2	Value ($\times 10^{-2}$)	Weight	Value ($\times 10^{-5}$)	Weight	Value ($\times 10^{-4}$)	Weight	Value ($\times 10^{-2}$)
1	16.0	0.08	19	13	8.3	33.1%	12.9	0.1%	1680.0	66.8%	25.1
			20	14	8.3	139.0%	12.9	0.2%	-234.9 [j]	39.2%	6.0
			21	15	8.3	111.4%	12.9	0.2%	-86.6	11.6%	7.5
			22	16	8.3	95.9%	12.9	0.1%	33.9	3.9%	8.7
			23	17	8.3	99.9%	12.9	0.2%	-0.3	0.0%	8.3
			24	18	8.3	98.6%	12.9	0.2%	10.4	1.2%	8.5
			25	19	8.3	100.4%	12.9	0.2%	-4.3	0.5%	8.3
5	16.3	2.08	19	13	208.3	93.1%	13.1	0.0%	1537.6	6.9%	223.7
			20	14	208.3	101.9%	13.1	0.0%	-388.4	1.9%	204.5
			21	15	208.3	100.7%	13.1	0.0%	-145.6	0.7%	206.9
			22	16	208.3	98.2%	13.1	0.0%	371.0	1.7%	212.1
			23	17	208.3	99.9%	13.1	0.0%	16.0	0.1%	208.5
			24	18	208.3	100.1%	13.1	0.0%	-18.7	0.1%	208.2
			25	19	208.3	100.1%	13.1	0.0%	-16.0	0.1%	208.2
10	19.1	8.33	19	13	833.3	92.0%	15.4	0.0%	7195.7	7.9%	905.3
			20	14	833.3	98.9%	15.4	0.0%	921.9	1.1%	842.6
			21	15	833.3	97.7%	15.4	0.0%	1950.6	2.3%	852.9
			22	16	833.3	99.8%	15.4	0.0%	186.9	0.2%	835.2
			23	17	833.3	100.0%	15.4	0.0%	-37.6	0.0%	833.0
			24	18	833.3	100.1%	15.4	0.0%	-61.5	0.1%	832.7
			25	19	833.3	99.9%	15.4	0.0%	62.7	0.1%	834.0
40	52.1	133.33	19	13	13333.3	95.9%	42.0	0.0%	56771.1	4.1%	13901.1
			20	14	13333.3	97.6%	42.0	0.0%	33451.2	2.4%	13667.9
			21	15	13333.3	98.8%	42.0	0.0%	16009.6	1.2%	13493.5
			22	16	13333.3	99.8%	42.0	0.0%	2468.5	0.2%	13358.1
			23	17	13333.3	99.8%	42.0	0.0%	2272.8	0.2%	13356.1
			24	18	13333.3	100.0%	42.0	0.0%	522.8	0.0%	13338.6
			25	19	13333.3	100.0%	42.0	0.0%	-125.4	0.0%	13332.1

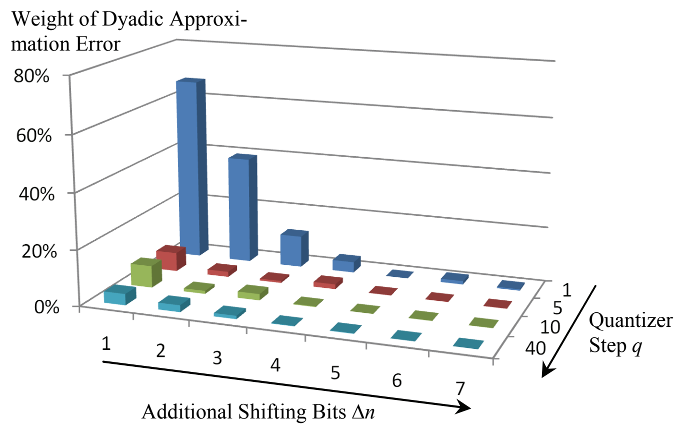


Fig. 5-1. Dyadic approximation error over the reconstruction error in percentage w.r.t. the quantizer step q and number of additional shifting bits Δn.

[j] The negative sign appears confusing. By recalling (5-8), we find that the only term which can introduce negative value is $\sigma_{\mathrm{q}}^2\left[\frac{1}{N^2}\sum_{i=0}^{N^2-1} w(i,i)\right]$. In some cases, the dyadic approximation even helps to compensate the error caused by the quantization, which are not common but somehow reasonable.

5.3. Characteristics of Dyadic Approximation Error

5.3.1. Position Dependency in Compressed Domain

Let us investigate the characteristics of the dyadic approximation error. First of all, let us recall (5-1)–(5-4) by focusing on the dyadic approximation error equivalent model–the $N^2\times 1$ vectors $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. For a dyadic approximation error scalar $\boldsymbol{S}_1(i)$ (or $\boldsymbol{S}_2(i)$, we use one of them for explanation) at position i, it can be calculated by dividing the real floating-point number by its dyadic approximation. Since the real floating-point number depends on the norm of the basic vector of the integer transform kernel at position i, the values of $\boldsymbol{S}_1(i)$'s for different i's are obviously different. In other words, the value of dyadic approximation error scalar $\boldsymbol{S}_1(i)$ is position dependent. Since $\boldsymbol{S}_1(i)$'s measure the error of dyadic approximation, their values are around unity. TABLE 5-3 gives an example of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ when the quantizer step q = 5, the number of forward shifting bits n_1 = 21, and the number of inverse shifting bits n_2 = 15.

TABLE 5-3
DYADIC APPROXIMATION ERROR SCALARS

Pos Idx	S_1	S_2	$S_1\times S_2$
0	1.0156	1.0000	1.0156
1	1.0280	0.9650	0.9920
2	0.9799	1.0220	1.0015
3	1.0280	0.9650	0.9920
4	0.9668	1.0472	1.0125
5	1.0280	0.9650	0.9920
6	0.9799	1.0220	1.0015
7	1.0280	0.9650	0.9920
:	:	:	:
252	1.0113	1.0021	1.0134
253	0.9920	1.0004	0.9924
254	1.0188	0.9781	0.9965
255	0.9920	1.0004	0.9924

Note that the position dependency property is only valid in the compressed domain (the DCT domain) and does not exist in the spatial domain of the reconstructed signal, since the error term of an element in the spatial domain is a weighted sum of all error terms in the compressed domain.

5.3.2. System Error and Equivalent Quantizer Step

Video coding standards only defines the decoder, hence only the pre-scaling factors are defined for the H.264/AVC. As a result, their counterparts, the post-scaling factors, have to be found by the designer of the encoder to best suit the values of pre-scaling factors. A clever recommendation was made by VCEG [80]

$$scalar(i,j) = 2^{total\ number\ of\ shifting\ bits} \times norm[basis(i)] \times norm[basis(j)] \,/\, rescalar(i,j) \quad (5\text{-}12)$$

which implies that an increase in $scalar(i,j)$ means an decrease in $rescalar(i,j)$, or vice versa. It is also inheritably true for the dyadic approximation error terms $\boldsymbol{S}_1(i)$ and $\boldsymbol{S}_2(i)$. Their product $[\boldsymbol{S}_1(i)\times\boldsymbol{S}_2(i)]$, which measures the dyadic approximation error of the whole system, is definitely closer to unity compared to any of them. If the dyadic approximation error of the whole system is not unity, the perfect reconstruction property of the transform cannot be preserved even though the quantization is bypassed. This causes a drift between the raw residual signal and the reconstructed residual signal, which lowers the coding efficiency and the visual quality. If the dyadic approximation error of the whole system is unity, another problem also exists. Let us name it as "quasi–error-free". Recall in Fig. 2-7(e), the blocks "DA Error Equivalent Model" and "Quantization" are next to each other. We can create an equivalent scenario by deliberately setting the dyadic approximation error to zero and moving the errors into the quantization stage. That is, setting quantizer step $q = 5$, $\boldsymbol{S}_1(k) = 1.1$ and $\boldsymbol{S}_2(k) = 0.9091$ is equivalent to setting $q = 5/1.1 = 4.545$ and $\boldsymbol{S}_1(k) = \boldsymbol{S}_2(k) = 1$. For the general case, even though the dyadic approximation error of the whole system is unity, transformed coefficients of different positions are quantized with different quantizer steps, equivalently. This means that components of certain "frequencies" levels are attenuated whereas others are amplified, which causes artifacts for the reconstructed images.

The "quasi–error-free" phenomenon can also be explained in a more theoretical fashion. Let us assume that perfect match between encoder side and decoder side $\boldsymbol{S}_1(i)\times\boldsymbol{S}_2(i) = 1$ has been achieved. By a careful examination on (5-8), we find that the one of the two terms contributing to the dyadic approximation error, $\sigma_{\boldsymbol{x}}^2\left[\frac{1}{N^2}\sum_{j=0}^{N^2-1}\sum_{i=0}^{N^2-1}\left|\boldsymbol{\mathcal{Y}}(i,j)\right|\right]$, vanishes since $(\boldsymbol{\mathcal{N}}_1 - \boldsymbol{\mathcal{N}}_2)$ in (5-11) equals to zero. Hence the dyadic approximation error is simplified to $\sigma_{\boldsymbol{q}}^2\left[\frac{1}{N^2}\sum_{i=0}^{N^2-1}\boldsymbol{\mathcal{W}}(i,i)\right]$ which is non-zero whenever the decoder side has approximation errors.

5.3.3. Effect of Dyadic Approximation Error

In this part, we are going to see exactly the effect of dyadic approximation error. This is achieved by comparing the dyadic approximation error contained output and the dyadic approximation error compensated output. The latter one can be obtained by adding two blocks $\boldsymbol{S}_1^{-1}$ and $\boldsymbol{S}_2^{-1}$ onto Fig. 2-7(f), as shown in Fig. 5-2. We synthesized input signal according to the empirical data (we assume an AR(1)-process with $\rho = 0.6$ and $\sigma_{\boldsymbol{x}}^2 = 52.1$), selected the order-16 transform kernel that we have mentioned in Section 5.2, and then obtained the output by varying the number of additional shifting bits Δn and quantizer step q. The MSE values are calculated as the results are shown in Fig. 5-3. It is observed that with the dyadic approximation error compensation, the effect of dyadic approximation error has been fully removed and the MSE values are close to the theoretical value $q^2/12$ (see the horizontal solid blue line in each of the subplots). On the other hand, the MSE caused by the dyadic approximation error is huge when the number of additional shifting bits is small (see the left part of the dotted black line in each of the subplots). Unless 5 or more additional bits are used, the effect of the dyadic approximation error cannot be eliminated—which matches well with the results we obtained in Section 5.2. Let us examine one practical example in which the dyadic approximation error gives noticeable artifacts, or equivalently the cases fulfilling the condition that $\Delta n < 5$. Let us examine the case when $\Delta n = 1$. Fig. 5-4 shows the difference images of the reconstructed *Lena* images with and without dyadic approximation error compensation at different quantizer steps. It is observed that the error caused by the dyadic approximation is more and more noticeable as q goes larger, the observation of which is supported by Fig. 5-3 that the absolute MSE value grows larger as q increases for a certain Δn. The experiments were repeated for other order-16 kernels and the results are similar.

$$\begin{tikzcd}
\boldsymbol{x} \arrow[r] & \boxed{\boldsymbol{\mathcal{H}}} \arrow[r] & \boxed{\boldsymbol{S}_1^{-1}} \arrow[r] & \boxed{\boldsymbol{S}_1} \arrow[r] & \boxed{quant(\cdot)} \arrow[r] & \boxed{quant^{-1}(\cdot)} \arrow[r] & \boxed{\boldsymbol{S}_2} \arrow[r] & \boxed{\boldsymbol{S}_2^{-1}} \arrow[r] & \boxed{\boldsymbol{\mathcal{H}}^{\mathrm{T}}} \arrow[r] & \boldsymbol{y}_r
\end{tikzcd}$$

Fig. 5-2. Block diagram of the case that the dyadic approximation error is compensated.

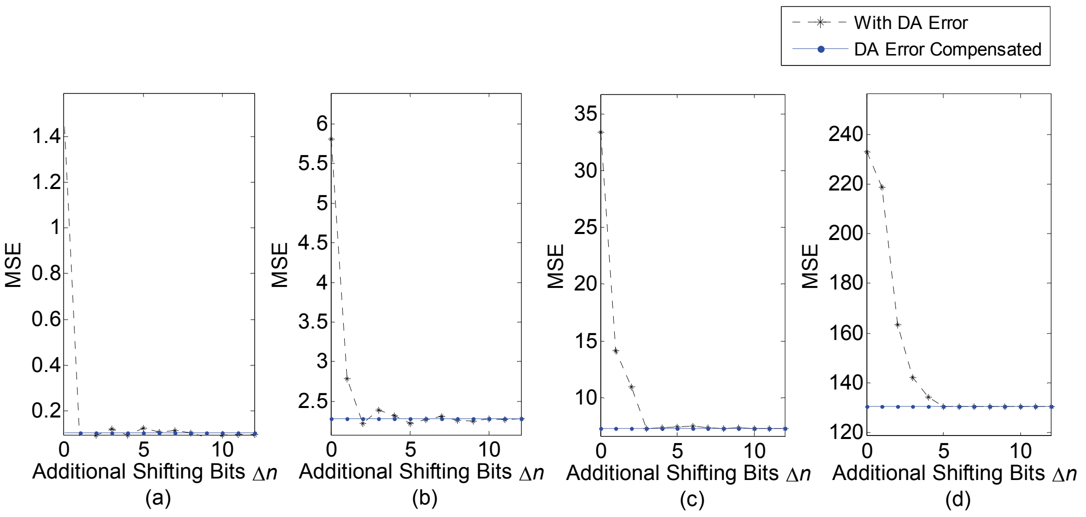


Fig. 5-3. Comparison of MSE values between output results with and without dyadic approximation error compensation for quantizer steps: (a) $q = 1$, (b) $q = 5$, (c) $q = 10$, and (d) $q = 40$.

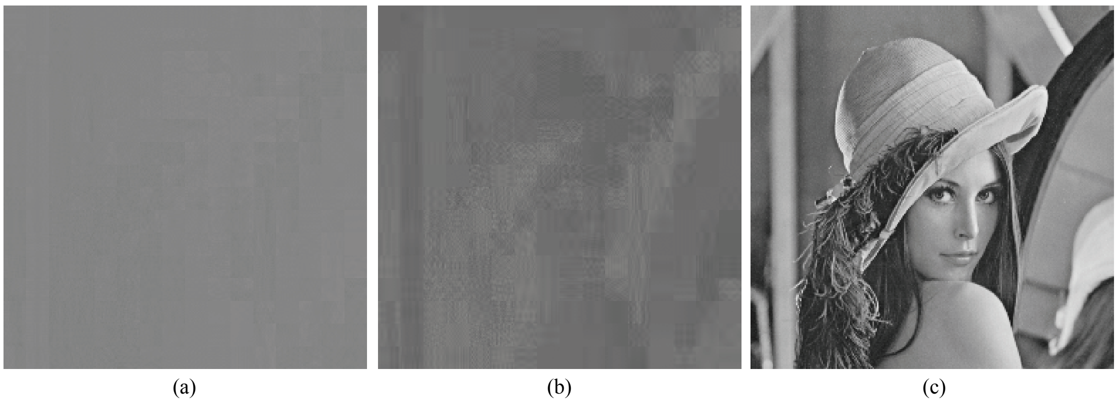


Fig. 5-4. Difference images of the reconstructed images with and without dyadic approximation error compensation for quantizer steps: (a) $q = 10$, and (b) $q = 40$. Results for $q = 1$ and $q = 5$ are not shown here since the details can hardly be noticed. The original image is shown in (c).

5.4. Conclusions

In this chapter, we have derived the analytical representation of the reconstruction error with the consideration of the dyadic approximation, as well as the quantization and the nonorthogonal transform. The dyadic approximation can cause two effects: the DCT coefficients are equivalently quantized with different stepsizes, and different “frequency” components are attenuated or amplified differently, which introduce visual artifacts. This problem is more serious as comparing to those caused by other factors, such as the nonorthogonality error. We have also suggested the least number of shifting bits for a set of order-16 transform kernels by both theoretical derivation and experimental results. We recommend that similar tests should be carried out to ensure the transform kernels being free from the dyadic approximation error for the future standards, such as the H.NGVC/HVC [110].

Chapter 6. Conclusions and Future Work

6.1. Conclusions

In this thesis, we have focused on the following topics inherited from the integerization of the transform coding process: the integer kernel design problem, the kernel selection strategies, the nonorthogonality error analysis, and the dyadic approximation error analysis.

In Chapter 3, we have proposed a new DCT-like kernel IK(5,7,3) and revitalized another DCT-like kernel IK(13,17,7) to effect adaptive kernel mechanism in hybrid video coding. We have found that the macroblock-level AKM using either one of these two kernels with the H.264/AVC default kernel IK(1,2,1) outperforms the H.264/AVC's default arrangement and the MB-AKM{(1,2,1)&IST}. This is especially true for high fidelity video coding applications, i.e. with small values of QP, and the MB-AKM{(1,2,1)&(5,7,3)} which is a dual-kernel gives the best performance as compared with other single or dual-kernel approaches in the literature. However, due to the trial-and-error characteristic of the MB-AKM, the computational complexity is high. Hence, further investigations were carried out to exploit the kernel selection tendencies, aiming at the development of a fast approach for the AKM. We have found that the slope of a pair of DCT-like kernels forms an important rate-distortion feature for kernel selection. This feature is due to the differences in length of shifting bits of kernels. With the help of the rate-distortion feature analysis, we are able to propose a fast frame-level kernel selection algorithm (FM-AKM) using the IK(1,2,1) for I- and P-Frames and the IK(5,7,3) for B-Frames for the H.264/AVC. This fast approach gives comparable or even better results in terms of PSNR and bitrate compared with the new dual kernel MB-AKM{(1,2,1)&(5,7,3)}. We have also generalized the rate-distortion feature extracted from a pair of kernels to a feature extracted from a group of many kernels, which is justified by an essentially stable topology formed by operation points. We further generalize the FM-AKM to let the algorithm adapt to any value of the lambda weight to cope with specific coding requirement.

In a recent paper [74], a comprehensive and instructive work on integer transform kernel design for the HD video coding was presented. However, we pointed out some minor (but yet useful) problems in Chapter 4, but the paper is really a piece of excellent work that can give readers inspiration, and for reference of future work. We have also proposed an improved method for the evaluation of the upper bound of the reconstruction error. Hence, this makes the original more complete and a more real worst-case scenario can be attained.

In Chapter 5, we have derived the analytical representation of the reconstruction error with the consideration of the dyadic approximation, as well as the quantization and the nonorthogonal transform. The dyadic approximation can cause two effects: the DCT coefficients are equivalently quantized with different stepsizes, and different "frequency" components are attenuated or amplified differently, which introduce visual artifacts. This problem is more serious as comparing to those caused by other factors, such as the nonorthogonality error. We have also suggested the least number of shifting bits for a set of order-16 transform kernels by both theoretical derivation and experimental results. We recommend that similar tests should be carried out to ensure the transform kernels being free from the dyadic approximation error for the future standards, such as the H.NGVC/HVC.

6.2. Future Research Directions

A possible future work is to expand the research of Chapter 3 into the quantitative analyses on how the lengths of shifting bits affect the rate-distortion feature (i.e. to see how the information be distributed in each word, to study more exactly the information lost due to bit-shift, etc.). Recently, researchers have shown that it is beneficial to include larger block-size transforms (such as 8×8 or 16×16 transforms) in the future standards to cope with the High Definition video contents. It is a good idea to expand integer kernel design and the investigation of kernel selection strategies into larger block-size transforms, and it appears to have some positive results for the 16×16 block-size transforms from our preliminary investigation. Hopefully the rate-distortion features in larger blocks can also be proved quantitatively, which can thus benefit the kernel selection strategy for video encoding and kernel determination for future standards.

REFERENCES

[1] "Video Codec for Audiovisual Services at p×64 kbits," ITU-T Recommendation H.261.

[2] "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 11172-2 (MPEG-1 Part 2).

[3] "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video," ITU-T Recommendation H.262 | ISO/IEC 13818-2 (MPEG-2 Part 2).

[4] "Video Coding for Low Bit Rate Communication," ITU-T Recommendation H.263.

[5] "Information Technology—Coding of Audio-Visual Objects—Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2).

[6] "AVC for Generic Audiovisual Services," ITU-T Recommendation H.264 | ISO/IEC 14496-10 (MPEG-4 Part 10).

[7] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transfom," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90-93, 1974.

[8] K. R. Rao and P. C. Yip, *Discrete Cosine Transform: Algorithms, Advantages and Applications*: Academic Press, 1990.

[9] V. Britanak, P. C. Yip and K. R. Rao, *Discrete Cosine and Sine Transforms: General properties, Fast algorithms and Integer Approximations*: Academic Press, 2007.

[10] Y. L. Chan and W. C. Siu, "Variable temporal-length 3-D discrete cosine transform coding," *IEEE Transactions on Image Processing*, vol. 6, no. 5, pp. 758-763, 1997.

[11] L. P. Chau and W. C. Siu, "Direct formulation for the realization of discrete cosine transform using recursive structure," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 1, pp. 50-52, 1995.

[12] Y. H. Chan and W. C. Siu, "On the realization of discrete cosine transform using the distributed arithmetic," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, no. 9, pp. 705-712, 1992.

[13] K. Karhunen, "Über lineare Methoden in der Wahrscheinlichkeitsrechnung (Over linear methods in probability theory)," *Annales Academiae Scientiarum Fennicae,* vol. 37, pp. 79, 1947.

[14] M. Loève, “Fonctions aléatoires du second ordre (Random functions of second order),” *Processus stochastiques et mouvement brownien*, 1948.

[15] M. Effros and P. A. Chou, “Weighted universal transform coding: universal image compression with the Karhunen–Loève transform,” *International Conference on Image Processing (ICIP 1995)*, pp. 61-64, 23-26 Oct 1995.

[16] R. D. Dony and S. Haykin, “Optimally adaptive transform coding,” *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1358-1370, 1995.

[17] C. Archer and T. K. Leen, “A generalized Lloyd-type algorithm for adaptive transform coder design,” *IEEE Transactions on Signal Processing*, vol. 52, no. 1, pp. 255-264, 2004.

[18] A. Dapena and S. Ahalt, “A hybrid DCT-SVD image-coding algorithm,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 2, pp. 114-121, 2002.

[19] C. F. Chen and K. K. Pang, “Hybrid coders with motion compensation,” *Multidimensional Systems and Signal Processing*, vol. 3, no. 2, pp. 241-266, 1992.

[20] C. F. Chen and K. K. Pang, “The optimal transform of motion-compensated frame difference images in a hybrid coder,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 6, pp. 393-397, 1993.

[21] W. Niehsen and M. Brunig, “Covariance analysis of motion-compensated frame differences,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 536-539, 1999.

[22] K. C. Hui and W. C. Siu, “Extended analysis of motion-compensated frame difference for block-based motion prediction error,” *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1232-1245, 2007.

[23] W. C. Siu and K. C. Hui, “On modelling the hybrid video coding for analysis and future development,” *6th International Conference on Information, Communications & Signal Processing*, pp. 1-5, 10-13 Dec 2007.

[24] F. Kamisli and J. S. Lim, “Transforms for the motion compensation residual,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP09)*, pp. 789-792, 19-24 Apr 2009.

[25] M. Helsingius, P. Kuosmanen and J. Astola, “Image compression using multiple transforms,” *Signal Processing: Image Communication*, vol. 15, no. 6, pp. 513-529, 2000.

[26] A. Robert, I. Amonou and B. Pesquet-Popescu, “Improving DCT-based coders through block oriented transforms,” *Advanced Concepts for Intelligent Vision Systems*, Video Processing and Coding, pp. 375-383: Springer Berlin / Heidelberg, 2006.

[27] A. Robert, I. Amonou and B. Pesquet-Popescu, “Improving intra mode coding in H.264/AVC through block oriented transforms,” *IEEE 8th Workshop on Multimedia Signal Processing*, pp. 382-386, 3-6 Oct 2006.

[28] A. Robert, I. Amonou and B. Pesquet-Popescu, “Improving H.264 video coding through block oriented transforms,” *IEEE International Conference on Multimedia and Expo*, pp. 705-708, 23-26 Jun 2008.

[29] F. Kamisli and J. S. Lim, “Video compression with 1-d directional transforms in H.264/AVC,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP10)*, pp. 738-741, 14-19 Mar 2010.

[30] J. J. Fu and B. Zeng, “Diagonal discrete cosine transforms for image coding,” *Advances in Multimedia Information Processing - PCM06*, pp. 150-158, 2006.

[31] B. Zeng and J. J. Fu, “Directional discrete cosine transforms for image coding,” *IEEE International Conference on Multimedia and Expo*, pp. 721-724, 9-12 Jul 2006.

[32] J. J. Fu and B. Zeng, “Directional discrete cosine transforms: a theoretical analysis,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP07)*, pp. 1105-1108, 15-20 Apr 2007.

[33] B. Zeng and J. J. Fu, “Directional discrete cosine transforms—a new framework for image coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 305-313, 2008.

[34] A. Drémeau, C. Herzet, C. Guillemot and J. J. Fuchs, “Sparse optimization with directional DCT bases for image compression,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP10)*, pp. 1290-1293, Dallas, Texas, 14-19 Mar 2010.

[35] C. L. Chang and B. Girod, “Direction-adaptive partitioned block transform for image coding,” *15th IEEE International Conference on Image Processing (ICIP08)*, pp. 145-148, 12-15 Oct 2008.

[36] H. Xu, J. Z. Xu and F. Wu, “Lifting-based directional DCT-like transform for image coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 10, pp. 1325-1335, 2007.

[37] J. Z. Xu, F. Wu, J. Liang and W. J. Zhang, "Directional lapped transforms for image coding," *Data Compression Conference (DCC08)*, pp. 142-151, 25-27 Mar 2008.

[38] J. Z. Xu, F. Wu, J. Liang and W. J. Zhang, "Directional lapped transforms for image coding," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 85-97, 2010.

[39] M. Karczewicz, "Improved intra coding," ITU-T SG16, Doc. VCEG-AF15, San Jose, CA, 20-21 Apr 2007.

[40] Y. Ye and M. Karczewicz, "Improved intra coding," ITU-T SG16, Doc. VCEG-AG11, Shenzhen, China, 20 Oct 2007.

[41] Y. Ye and M. Karczewicz, "Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," *15th IEEE International Conference on Image Processing (ICIP08)*, pp. 2116-2119, 12-15 Oct 2008.

[42] "Key Technical Area (KTA)," http://iphome.hhi.de/suehring/tml/download/KTA/.

[43] M. Budagavi *et al.*, "Description of video coding technology proposal by Texas Instruments Inc.," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A101, Dresden, DE, 15-23 Apr 2010.

[44] Y. W. Huang *et al.*, "A Technical Description of MediaTek's Proposal to the JCT-VC CfP," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A109r2, Dresden, DE, 15-23 Apr 2010.

[45] B. M. Jeon, S. W. Park, J. S. Kim and J. Y. Park, "Description of video coding technology proposal by LG Electronics," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A110, Dresden, DE, 15-23 Apr 2010.

[46] H. T. Yang *et al.*, "Description of video coding technology proposal by Huawei Technologies & Hisilicon Technologies," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A111, Dresden, DE, 15-23 Apr 2010.

[47] J. Y. Lim *et al.*, "Description of video coding technology proposal by SK telecom, Sejong Univ. and Sungkyunkwan Univ.," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A113, Dresden, DE, 15-23 Apr 2010.

[48] T. Chujoh, A. Tanizawa and T. Yamakage, "Description of video coding technology proposal by TOSHIBA," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A117r1, Dresden, DE, 15-23 Apr 2010.

[49] H. Y. Kim *et al.*, "Description of video coding technology proposal by ETRI," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A127, Dresden, DE, 15-23 Apr 2010.

[50] K. McCann, W. J. Han and I. K. Kim, "Samsung's Response to the Call for Proposals on Video Compression Technology," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A124, Dresden, DE, 15-23 April 2010.

[51] "Test model under consideration," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A205, Dresden, DE, 15-23 Apr 2010.

[52] G. W. Wornell and D. H. Staelin, "Transform image coding with a new family of models," *1988 International Conference on Acoustics, Speech, and Signal Processing (ICASSP88)*, pp. 777-780, 11-14 Apr 1988.

[53] X. Zhao, L. Zhang, S. W. Ma and W. Gao, "Rate-distortion optimized transform," ISO/IEC JTC1/SC29/WG11, Doc. M16926, Xi'an, China, Oct 2009.

[54] K. Rose, A. Heiman and I. Dinstein, "DCT/DST alternate-transform image coding," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 94-101, 1990.

[55] A. K. Jain, "A fast Karhunen–Loève transform for a class of random processes," *IEEE Transactions on Communications*, vol. 24, no. 9, pp. 1023-1029, 1976.

[56] J. N. Han, A. Saxena and K. Rose, "Towards jointly optimal spatial prediction and adaptive transform in video image coding," *35th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP10)*, pp. 726-729, Dallas, Texas, 14-19 Mar 2010.

[57] S. C. Lim, H. C. Choi, S. Y. Jeong and J. S. Choi, "Integer sine transform for inter frame," ITU-T Q6/SG16, Doc. VCEG-AJ12, San Diego, CA, 8-10 Oct 2008.

[58] S. C. Lim, D. Y. Kim and Y. L. Lee, "Alternative transform based on the correlation of the residual signal," *Congress on Image and Signal Processing (CISP08)*, pp. 389-394, 27-30 May 2008.

[59] S. C. Lim *et al.*, "Rate-distortion optimized adaptive transform coding," *Optical Engineering*, vol. 48, no. 8, pp. 087004, 2009.

[60] A. Ichigaya *et al.*, "Description of video coding technology proposal by NHK and Mitsubishi," JCTVC of ITU-T and ISO/IEC, Doc. JCTVC-A122, Dresden, DE, 15-23 Apr 2010.

[61] "IEEE Standard Specifications for the Implementations of 8×8 Inverse Discrete Cosine Transform (IEEE Std 1180-1990)," CAS Standards Committee of the IEEE Circuits and Systems Society, 1990.

[62] "Information technology—MPEG Video Technologies—Part 1: Accuracy Requirements for Implementation of Integer-Output 8×8 Inverse Discrete Cosine Transform," ISO/IEC 23002-1 (MPEG-C Part 1).

[63] "Information technology—MPEG Video Technologies—Part 1: Accuracy Requirements for Implementation of Integer-Output 8×8 Inverse Discrete Cosine Transform AMENDMENT 1: Software for Integer IDCT Accuracy Testing," ISO/IEC 23002-1 AMENDMENT 1(MPEG-C Part 1 AMD1).

[64] G. J. Sullivan, "Standardization of IDCT approximation behavior for video compression: the history and the new MPEG-C parts 1 and 2 standards," *Applications of Digital Image Processing XXX*, pp. 669611, San Diego, CA, USA, 2007.

[65] Y. Yagasaki, "Oddification problem for IDCT mismatch," ISO/IEC JTC1/SC2/WG11, Mar 1993.

[66] H. W. Jones, D. H. Hein and S. C. Knauer, "The Karhunen–Loève, discrete cosine and related transforms obtained via the Hadamard transform," *International Telemetring Conference*, pp. 87-98, Los Angeles, CA, US, Nov 1978.

[67] R. Srinivasan and K. R. Rao, "An approximation to the discrete cosine transform for N=16," *Signal Processing*, vol. 5, pp. 81-85, Jan, 1983.

[68] H. S. Kwak, R. Srinivasan and K. R. Rao, "C-matrix transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, no. 5, pp. 1304-1307, 1983.

[69] W. K. Cham, "Development of integer cosine transforms by the principle of dyadic symmetry," *IEE Proceedings I Communications, Speech and Vision*, vol. 136, no. 4, pp. 276-282, 1989.

[70] W. K. Cham and Y. T. Chan, "An order-16 integer cosine transform," *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1205-1208, 1991.

[71] W. K. Cham, "Family of order-4 four-level orthogonal transforms," *Electronics Letters*, vol. 19, no. 21, pp. 869-871, 1983.

[72] W. H. Chen, C. Smith and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004-1009, 1977.

[73] J. Dong, K. N. Ngan, C. K. Fong and W. K. Cham, "A universal approach to developing fast algorithm for simplified order-16 ICT," *IEEE International Symposium on Circuits and Systems (ISCAS07)*, pp. 281-284, 27-30 May 2007.

[74] J. Dong, K. N. Ngan, C. K. Fong and W. K. Cham, "2-D order-16 integer transforms for HD video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 10, pp. 1462-1474, 2009.

[75] J. D. Allen and S. M. Blonstein, "The multiply-free Chen transform—a rational approach to JPEG," *Picture Coding Symposium (PCS91)*, pp. 237-240, Tokyo, Japan, Sep 1991.

[76] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3032-3044, 2001.

[77] J. Liang, T. D. Tran, W. Dai and P. Topiwala, "FastVDO's Unified 16-Bit Transform/Quantization Approach," Doc. JVT-B103d1, Geneva, CH, 29 Jan-1 Feb 2002.

[78] Y. J. Chen, S. Oraintara and T. Nguyen, "Video compression using integer DCT," *International Conference on Image Processing (ICIP00)*, pp. 844-845, 10-13 Sep 2000.

[79] G. Bjontegaard, "Response to call for proposals for H.26L," ITU-T SG16/Q15, Doc. Q15-F-11, Seol, Korea, 3-6 Nov 1998.

[80] A. Hallapuro, M. Karczewicz and H. S. Malvar, "Low complexity transform and quantization – Part I: Basic implementation," Doc. JVT-B038, Geneva, CH, 29 Jan-1 Feb 2002.

[81] G. Sullivan, T. Wiegand and A. Luthra, "JVT (of ISO/IEC MPEG and ITU-T Q.6/16 VCEG) 2nd Meeting Report, Jan 29 – Feb 1 2002," JVT of ITU-T and ISO/IEC, Doc. JVT-B001_draft_5, Geneva, CH, 29 Jan-1 Feb 2002.

[82] H. S. Malvar, A. Hallapuro, M. Karczewicz and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598-603, 2003.

[83] "Information Technology—Advanced Coding of Audio and Video—Part2: Video," GB/T 20090.2-2006 (AVS).

[84] "Television—VC-1 Compressed Video Bitstream Format and Decoding Process," SMPTE 421M (VC-1).

[85] S. Naito, A. Matsumura and A. Koike, "Efficient coding scheme for super high definition video based on extending H.264 high profile," *Conference on Visual Communications and Image Processing*, pp. 607727, 2006.

[86] K. H. Lee *et al.*, "Technical considerations for ad hoc group on new challenges in video coding standardization," ISO/IEC JTC1/SC29/WG11, Doc. M15580, Hannover, Germany, Jul 2008.

[87] C. K. Fong and W. K. Cham, "Simple order-16 integer transform for video coding," *International Conference on Image Processing (ICIP10)*, pp. 161-164, 26-29 Sep 2010.

[88] M. Wien, C. Mayer and J. R. Ohm, "Integer transforms for H.26L using adaptive block transforms," ITU-T SG16/Q15, Doc. Q15-K-24, Portland, Oregon, USA, 22-25 Aug 2000.

[89] S. W. Ma and C. C. J. Kuo, "High-definition video coding with super-macroblocks," *Conference on Visual Communications and Image Processing*, pp. 650816, San Jose, CA, USA, 2007.

[90] P. S. Chen, Y. Ye and M. Karczewicz, "Video coding using extended block sizes," ITU-T SG16/Q6, Doc. COM16-C123-E, Jan 2009.

[91] B. S. Lee *et al.*, "A 16x16 transform kernel with quantization for (ultra) high definition video coding," ITU-T SG16/Q6, Doc. VCEG-AK13, Yokohama, Japan, 15-18 Apr 2009.

[92] R. Joshi, Y. Reznik and M. Karczewicz, "Simplified transforms for extended block sizes," ITU-T SG16/Q6, Doc. VCEG-AL30, Geneva, CH, 6-10 Jul 2009.

[93] R. Joshi, Y. A. Reznik and M. Karczewicz, "Efficient large size transforms for high-performance video coding," *Applications of Digital Image Processing XXXIII*, pp. 77980W, San Diego, California, USA, 2010.

[94] G. Bjontegaard, "Coding improvement by using 4x4 blocks for motion vectors and transform," ITU-T Q.15/SG16, Doc. Q15-C-23, 2-5 Dec 1997.

[95] A. T. Hinds, "Design of high-performance fixed-point transforms using the common factor method," *Applications of Digital Image Processing XXXIII*, pp. 77980U, San Diego, California, USA, 2010.

[96] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg and D. J. LeGall, *MPEG Video Compression Standard*, New York, NY: Chapman & Hall, 1997.

[97] B. Widrow, I. Kollár and M. C. Liu, "Statistical theory of quantization," *IEEE Transactions on Instrumentation and Measurement*, vol. 45, no. 2, pp. 353-361, 1996.

[98] H. S. Malvar, "Low-Complexity length-4 transform and quantization with 16-bit arithmetic," Doc. VCEG-N44, Sep 2001.

[99] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74-90, 1998.

[100] T. Wiegand *et al.*, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688-703, 2003.

[101] G. Bjontegaard, "H.26L test model long term number 1 (TML-1) draft 2," ITU-T SG16/Q15, Doc. Q15-H-36d2, 3-6 Aug 1999.

[102] "H.264/AVC Reference Software, Joint Model (JM) Ver 12.2," http://iphome.hhi.de/suehring/tml/.

[103] T. K. Tan, G. Sullivan and T. Wedi, "Recommended simulation common conditions for coding efficiency experiments revision 1," ITU-T SG16, Doc. VCEG-AE010, 13-19 Jan 2007.

[104] G. Bjontegaard, "Calculation of average PSNR differences between RD curves," JVT of ISO/IEC MPEG and ITU-T VCEG, Doc. VCEG-M33, Mar 2001.

[105] M. Flierl and B. Girod, "Generalized B pictures and the draft H.264/AVC video-compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 587-597, 2003.

[106] I. E. G. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia," pp. 187-198: Wiley, 2003.

[107] M. A. Robertson and R. L. Stevenson, "DCT quantization noise in compressed images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 27-38, 2005.

[108] M. Wien and S. J. Sun, "ICT comparison for adaptive block transforms," ITU-T QG16/Q6, Doc. VCEG-L12, Eibsee, Germany, 9-12 Jan 2001.

[109] A. K. Jain, "Fundamentals of Digital Image Processing," pp. 28-31: Prentice-Hall, 1989.

[110] "Joint call for proposals on video compression technology," ITU-T Q6/16 and ISO/IEC JTC1/SC29/WG11, Doc. VCEG-AM91 | N11113, Jan 2010.