Transform-Domain Fast Sum of the Squared Difference Computation for H.264/AVC Rate-Distortion Optimization

Lai-Man Po, *Member, IEEE*, and Kai Guo

***Abstract*—In H.264/AVC, the rate-distortion optimization for mode decision plays a significant role to achieve its outstanding performance in terms of both compression efficiency and video quality. However, this mode decision process also introduces extremely high complexity in the encoding process especially the computation of the sum of squared differences (SSD) between the original and reconstructed image blocks. In this paper, fast SSD (FSSD) algorithms are proposed to reduce the complexity of the rate-distortion cost function implementation. The proposed FSSD algorithm is based on the theoretical equivalent of the SSDs in spatial and transform domains and determines the distortion in integer cosine transform domain using an iterative table-lookup quantization process. This approach could avoid the inverse quantization/transform and pixel reconstructions processes with nearly no rate-distortion performance degradation. In addition, the FSSD can also be used with efficient bit rate estimation algorithms to further reduce the cost function complexity. Experimental results show that the new FSSD can save up to 15% of total encoding time with less than 0.1% coding performance degradation and it can save up to 30% with ignorable performance degradation when combining with conventional bit rate estimation algorithm.**

***Index Terms*—Distortion measure, H.264/AVC, mode decision, rate-distortion optimization (RDO).**

I. Introduction

THE newest video coding standard is known as H.264/AVC [1], [2], which greatly outperforms the previous MPEG-1/2/4 [3], [4] and H.261/263 [5], [6] standards in terms of both picture quality and compression efficiency. It can provide the same picture quality as DVD (or MPEG-2) video while only consuming about 25% of the storage space and its bit rate is about half of that of the MPEG-4 advanced simple profile [2]. To achieve this superior coding performance, H.264/AVC adopts many advanced techniques, such as directional spatial prediction for intra-frame coding, variable and hierarchical block transform, arithmetic entropy coding, multiple reference frame motion compensation, deblocking, etc. It also uses seven different block sizes for motion-compensation in the inter-mode, and three different block sizes with various spatial directional prediction modes in the intra-mode. The main critical process employed is the rate-distortion optimized (RDO) mode decision technique [7], [8] which provides H.264/AVC much better coding performance in terms of compression efficiency and visual quality. To select the best macroblock coding mode, H.264/AVC encoder needs to compute the rate-distortion cost of all possible modes, which involves computation of integer transform, quantization, variable length coding and pixel reconstruction processes. All of these processing explains the high computational complexity of rate-distortion cost calculation. Hence, the cost function computation makes H.264/AVC impossible to be realized in real-time applications without high computing hardware.

To accelerate the H.264/AVC coding process, the JVT reference software version JM 6.1d provides two fast cost functions [11] based on sum of absolute difference (SAD) and sum of absolute Hadamard-transformed difference (SATD). These two cost functions could avoid the quantization and transformation computation, but the coding performance degradation is non-ignorable in terms of both bit rate and reconstructed video visual quality. Recently, a new transform domain rate estimation and distortion measure was proposed to reduce the rate-distortion cost function computation with ignorable coding performance degradation [10]. Both distortion measure and rate estimation are obtained in the transform domain, but the computation reduction is limited within 10%. In this paper, a new fast sum of squared difference (FSSD) computation algorithm with use of an iterative table-lookup quantization process is proposed to reduce the complexity of the H.264 rate-distortion cost function calculation with nearly lossless coding efficiency as compared with the original RDO. The method is based on iterative table-lookup process to replace the conventional quantization, inverse quantization and inverse integer cosine transform. The proposed algorithm can also be combined with fast bit rate estimation algorithm to further speed up the computation with only ignorable performance degradation.

The organization of this paper is that a review of conventional rate-distortion costs for H.264/AVC is presented in Section II. In Section III, we analyze the fundamental reason which determines the distortion. In Section IV, the detail of FSSD algorithm is presented. The simulation results and conclusion are presented in Section V and VI, respectively.

II. Rate-Distortion Cost Functions for H.264/AVC

In H.264/AVC encoding process, the best macroblock coding mode is selected by computing the rate-distortion cost of all

Manuscript received August 22, 2006; revised November 23, 2006. This work was supported in part by a grant from the City University of Hong Kong, Kowloon, Hong Kong, under Project 7001796. This paper was recommended by Associate Editor E. Steinbach.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon Hong Kong, China (e-mail: eelmpo@cityu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2007.896663

possible modes. The best mode is the one with minimum rate-distortion cost and this cost is defined as

$$J_{\mathrm{RD}} = \mathrm{SSD}(\mathbf{S},\mathbf{C}) + \lambda \cdot R \tag{1}$$

where λ is the Lagrange multiplier [9], R is the number of coded bits for each macroblock and the $\mathrm{SSD}(\mathbf{S},\mathbf{C})$ is the sum of the squared difference (SSD) between the original blocks $\mathbf{S}$ and the reconstructed block $\mathbf{C}$, and it can be expressed as

$$\mathrm{SSD}(\mathbf{S},\mathbf{C}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(s_{ij}-c_{ij})^2 = \|\mathbf{S}-\mathbf{C}\|_F^2 \tag{2}$$

where s_{ij} and c_{ij} are the (i,j)th elements of the current original block $\mathbf{S}$ and the reconstructed block $\mathbf{C}$, respectively. Moreover, N is the image block size ($N=4$ in H.264/AVC standard) and $\|\ \|_F$ is the *Frobenius* norm. In this paper, we call the $\mathrm{SSD}(\mathbf{S},\mathbf{C})$ as spatial-domain SSD since the distortion computation is performed in spatial-domain pixel values. It is also found that the computation of spatial-domain SSD is very time-consuming, since we have to obtain the reconstructed block after Transformation—Quantization—Inverse Quantization—Inverse Transformation—Pixel Reconstruction for each possible prediction mode. To accelerate the coding process, the JVT reference software version JM 6.1d provides a fast SAD-based cost function [11]

$$J_{\mathrm{SAD}} = \begin{cases} \mathrm{SAD}(\mathbf{S},\mathbf{P}) + \lambda_1 \cdot 4K, & \text{if intra } 4\times 4 \text{ mode} \\ \mathrm{SAD}(\mathbf{S},\mathbf{P}), & \text{otherwise} \end{cases} \tag{3}$$

where $\mathrm{SAD}(\mathbf{S},\mathbf{P})$ is sum of absolute differences between the original block $\mathbf{S}$ and the predicted block $\mathbf{P}$. The λ_1 is also approximate exponential function of the quantization parameter (QP) which is almost the square of λ, and the K equal to 0 for the probable mode and 1 for the other modes. The $\mathrm{SAD}(\mathbf{S},\mathbf{P})$ is expressed by

$$\mathrm{SAD}(\mathbf{S},\mathbf{P}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|s_{ij}-p_{ij}| \tag{4}$$

where s_{ij} and p_{ij} are the (i,j)th elements of the current original block $\mathbf{S}$ and the predicted block $\mathbf{P}$, respectively. This SAD-based cost function could save a lot of computations as the distortion part is based on the differences between the original block and the predicted block instead of the reconstructed block. Thus, the processes of image block transformation, quantization, inverse quantization, inverse transformation and reconstruction can all be saved. In addition, the number of bits is estimated by constants either equal 4 or 0. Thus, the variable length coding using CAVLC or CABAC can also be saved. However, the expense of the computation reduction usually comes with quite significant degradation of coding efficiency. To achieve better rate-distortion performance, JM6.1d also provided an alternative SATD-based cost function [11]

$$J_{\mathrm{SATD}} = \begin{cases} \mathrm{SATD}(\mathbf{S},\mathbf{P}) + \lambda_1 \cdot 4K, & \text{if intra } 4\times 4 \text{ mode} \\ \mathrm{SATD}(\mathbf{S},\mathbf{P}), & \text{otherwise} \end{cases} \tag{5}$$

where $\mathrm{SATD}(\mathbf{S},\mathbf{P})$ is sum of absolute Hadamard-transformed [12], [13] difference between the original block $\mathbf{S}$ and the predicted block $\mathbf{P}$, which is given by

$$\mathrm{SATD}(\mathbf{S},\mathbf{P}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|h_{ij}| \tag{6}$$

where h_{ij} are the (i,j)th element of the Hadamard transformed image block $\mathbf{H}$ which is the difference between the original block $\mathbf{S}$ and the predicted block $\mathbf{P}$. The Hadamard transformed block $\mathbf{H}$ is defined as

$$\mathbf{H} = \mathbf{T}_H(\mathbf{S}-\mathbf{P})\mathbf{T}_H^T \tag{7}$$

with

$$\mathbf{T}_H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{8}$$

Experimental results show that the J_{SATD} could achieve better rate-distortion performance than the J_{SAD}, but it requires more computation due to the Hadamard transformation. If the fast Hadamard transform (FHT) is used, the computational requirement of the J_{SATD} can reduce half of the computation. However, the overall rate-distortion performance is still lower than the optimized J_{RD}. In addition, all these distortion expressions are based on the difference between original block and predicted block, so they avoid obtaining the reconstructed blocks. Nevertheless, they cannot well predict the real distortion, which will be explained in the next section. The rate-distortion performance comparison of H.264/AVC using RDO-based, SAD-based and SATD-based cost functions for different QPs and three well-known test sequences in terms of PSNR and bit rate is shown in Table I. Compared with RDO-based encoder, the performance degradation of both SAD-based and SATD-based cost functions are non-ignorable.

III. Transform-Domain Sum of Squared Difference

In this section, we derive the theoretical equivalent of the SSDs in spatial, differential and transform domains in order to present the major cause of the SSD between the original block and reconstruction block in the rate-distortion cost function. The computation of the MPEG-like or H.264/AVC rate-distortion cost function J_{RD} can be summarized as follows.

1) Compute the predicted block using inter- or intra-frame prediction: $\mathbf{P}$.
2) Using the original block $\mathbf{S}$ and the predicted block $\mathbf{P}$ to compute the residual (difference) block: $\mathbf{D} = \mathbf{S} - \mathbf{P}$.
3) Discrete Cosine Transform (DCT) the residual block:

$$\mathbf{F} = \mathrm{DCT}(\mathbf{D}) = \mathbf{T}_{\mathrm{DCT}}\mathbf{D}\mathbf{T}_{\mathrm{DCT}}^{\mathrm{T}} \tag{9}$$

where $\mathbf{T}_{\mathrm{DCT}}$ is the DCT matrix which is a unitary matrix and $\mathbf{T}_{\mathrm{DCT}}$ is the transported matrix of $\mathbf{T}_{\mathrm{DCT}}$.

4) Quantization of the transformed residual block: $\mathbf{Z} = \mathbf{Q}(\mathbf{F})$.

TABLE I
H.264 RATE-DISTORTION PERFORMANCE COMPARISON USING RDO-BASED, SAD-BASED, AND SATD-BASED COST FUNCTIONS IN TERMS OF PSNR (dB) AND RATE (kbps)

Sequence	QP	28			32			36			40		
		RDO	SAD	SATD	RDO	SAD	SATD	RDO	SAD	SATD	RDO	SAD	SATD
Akiyo	PSNR	39.94	39.76	39.81	36.95	36.86	36.88	34.30	34.04	34.12	31.43	31.36	31.40
	Rate	218.09	225.16	221.64	154.03	160.47	157.54	111.77	114.78	112.64	80.78	82.85	80.86
Foreman	PSNR	37.84	37.65	37.68	34.90	34.54	34.72	32.40	32.26	32.29	29.69	29.46	29.62
	Rate	421.58	432.09	427.49	251.64	258.36	260.21	163.58	168.57	166.71	110.09	112.21	114.62
Stefan	PSNR	37.16	36.85	36.94	33.49	33.20	33.29	30.22	30.01	30.15	26.91	26.58	26.88
	Rate	891.62	915.64	910.84	597.29	610.31	611.58	388.42	398.64	399.61	242.23	250.40	255.31

5) Entropy code of the $\mathbf{Z}$ to find the number of bits to encode the block: $R = \mathrm{VLC}(\mathbf{Z})$.
6) Inverse quantization: $\hat{\mathbf{F}} = \mathbf{Q}^{-1}(\mathbf{Z})$.
7) Inverse transform the inverse quantized block: $\hat{\mathbf{D}} = \mathrm{DCT}^{-1}(\hat{\mathbf{F}})$.
8) Compute the reconstructed image block: $\mathbf{C} = \hat{\mathbf{D}} + \mathbf{P}$.
9) Calculate the R-D cost: $J_{\mathrm{RD}} = \mathrm{SSD}(\mathbf{S}, \mathbf{C}) + \lambda \cdot R$.

A. Differential-Domain SSD

Mathematically, the original block $\mathbf{S}$ and reconstructed block $\mathbf{C}$ can be expressed as

$$\mathbf{S} = \mathbf{D} + \mathbf{P} \tag{10}$$
$$\mathbf{C} = \hat{\mathbf{D}} + \mathbf{P} \tag{11}$$

where the $\mathbf{P}$ is the predicted block, $\mathbf{D}$ is the residual block and $\hat{\mathbf{D}}$ is the reconstructed residual block. Based on this relationship, the spatial-domain $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ can be expressed as differential-domain $\mathrm{SSD}(\mathbf{D}, \hat{\mathbf{D}})$:

$$\begin{aligned}\mathrm{SSD}(\mathbf{S}, \mathbf{C}) &= \|\mathbf{S} - \mathbf{C}\|_F^2 = \|\mathbf{D} + \mathbf{P} - \hat{\mathbf{D}} - \mathbf{P}\|_F^2 \\ &= \|\mathbf{D} - \hat{\mathbf{D}}\|_F^2 = \mathrm{SSD}(\mathbf{D}, \hat{\mathbf{D}}).\end{aligned} \tag{12}$$

That means the spatial-domain $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ is equivalent to differential-domain $\mathrm{SSD}(\mathbf{D}, \hat{\mathbf{D}})$. Based on this relationship, we can calculate the R-D cost in differential-domain with $\mathrm{J_{RD}} = \mathrm{SSD}(\mathbf{D}, \hat{\mathbf{D}}) + \lambda R$ for saving the computation of the reconstructed image block $\mathbf{C} = \hat{\mathbf{D}} + \mathbf{P}$. However, this computational reduction is very limited.

B. Transform-Domain SSD

Before we define the transform-domain SSD, we need to emphasize that the DCT matrix $\mathbf{T}_{\mathrm{DCT}}$ used in MPEG-like or H.264/AVC video coding is a unitary matrix, which has the property [16] of

$$\|\mathbf{X}\|_{\mathrm{F}} = \left\|\mathbf{T}_{\mathrm{DCT}}\mathbf{X}\mathbf{T}_{\mathrm{DCT}}^{\mathrm{T}}\right\|_{\mathrm{F}} \tag{13}$$

where $\mathbf{X}$ is a square matrix. According to (12) and (13), we can also express the SSD in transform-domain as

$$\begin{aligned}\mathrm{SSD}(\mathbf{S}, \mathbf{C}) &= \|\mathbf{D} - \hat{\mathbf{D}}\|_{\mathrm{F}}^2 = \left\|\mathbf{T}_{\mathrm{DCT}}(\mathbf{D} - \hat{\mathbf{D}})\mathbf{T}_{\mathrm{DCT}}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 \\ &= \left\|\mathbf{T}_{\mathrm{DCT}}\mathbf{D}\mathbf{T}_{\mathrm{DCT}}^{\mathrm{T}} - \mathbf{T}_{\mathrm{DCT}}\hat{\mathbf{D}}\mathbf{T}_{\mathrm{DCT}}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 \\ &= \|\mathbf{F} - \hat{\mathbf{F}}\|_{\mathrm{F}}^2 = \mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})\end{aligned} \tag{14}$$

where $\mathbf{F}$ and $\hat{\mathbf{F}}$ are the transformed residual block and inverse quantized-transformed residual block. Equation (14) shows that the cause of the SSD is due to the quantization errors in the DCT transformed residual block $\hat{\mathbf{F}}$. The reason behind is that the quantization is applied in the transformed coefficients of $\mathbf{F}$. That is why $\mathrm{SAD}(\mathbf{S}, \mathbf{P})$ and $\mathrm{SATD}(\mathbf{S}, \mathbf{P})$ cannot well predict the $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$. Both $\mathrm{SAD}(\mathbf{S}, \mathbf{P})$ and $\mathrm{SATD}(\mathbf{S}, \mathbf{P})$ are determined by the original block and predicted block, without considering the influence of quantization which should be the real reason of SSD. For example, if $\mathrm{quantization\ step} = 18$ and all the transform coefficients are multiple of 18, then quantized coefficients could be the same as the inverse quantized coefficients without error as

$$\mathbf{F} = \begin{bmatrix} 180 & 72 & 54 & 18 \\ 72 & 36 & 18 & 18 \\ 36 & 18 & 18 & 18 \\ 18 & 18 & 18 & 0 \end{bmatrix} \text{ and } \hat{\mathbf{F}} = \begin{bmatrix} 180 & 72 & 54 & 18 \\ 72 & 36 & 18 & 18 \\ 36 & 18 & 18 & 18 \\ 18 & 18 & 18 & 0 \end{bmatrix}.$$

This demonstrates that even most of the coefficients of $\mathbf{F}$ are large, $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ can be zero since $\|\mathbf{F} - \hat{\mathbf{F}}\|_F^2 = 0$. The example indicates that the elements of $\mathbf{D}$ or $\mathbf{F}$ may not be directly related to the $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ that is determined by the quantization error. That is why the rate-distortion performance of SAD-based cost function and SATD-based cost function is lower than optimized cost function.

On the other hand, the transform-domain $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ can be used to save more computations for R-D cost calculation using $\mathrm{J_{RD}} = \mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}}) + \lambda R$, as both inverse DCT transform and image block reconstruction can be avoided. Another advantage of using this transform-domain $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ is that without consideration of clipping function applied in the practical computation there will not have any performance degradation in terms of both coding efficiency and reconstructed image distortion as $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ and $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ are theoretically equivalent. In H.264/AVC, however, DCT is implemented by an integer cosine transform (ICT) with scaling factors for reducing the computational complexity [14]. The reason behind that is the ICT can be realized by shift and addition operations only. In order to achieve higher computational reduction using transform-domain $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})$, a new iterative table-lookup quantization process for fast SSD computation is proposed in ICT domain that could further avoid the computations of inverse quantization and inverse ICT processes.

IV. FAST SUM OF THE SQUARED DIFFERENCE COMPUTATION

A. Review of Integer Cosine Transform in H.264

The practical implementation of the DCT and quantization process in H.264/AVC is a little bit different from (9) and its architecture is shown in Fig. 1. The DCT is implemented by

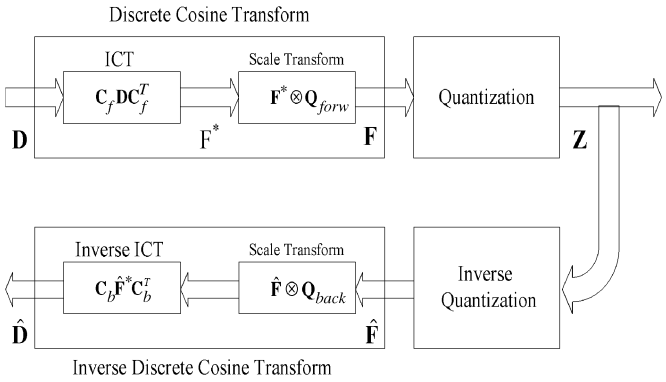

Fig. 1. Architecture of ICT and quantization.

ICT with scaling factors for complexity reduction [14], which can be expressed as

$$\begin{aligned}\mathbf{F} &= \mathrm{DCT}(\mathbf{D}) = \mathbf{C}_f\mathbf{D}\mathbf{C}_f^T \otimes \mathbf{Q}_{\mathrm{forw}} \\ &= \mathrm{ICT}(\mathbf{D}) \otimes \mathbf{Q}_{\mathrm{forw}} = \mathbf{F}^* \otimes \mathbf{Q}_{\mathrm{forw}}\end{aligned} \tag{15}$$

where, $\mathbf{C}_f$ is called ICT core matrices, $\mathbf{Q}_{\mathrm{forw}}$ is called scaling factors, and $\mathbf{F}^*$ is the ICT transformed block. The symbol $\otimes$ indicates the operator that each element of $\mathbf{C}_f\mathbf{D}\mathbf{C}_f^T$ (or $\mathbf{F}^*$) is multiplied by the scaling factor in the corresponding position. The forward core and scale transform matrices are defined as

$$\mathbf{C}_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad \mathbf{Q}_{\mathrm{forw}} = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

where $a = 1/2, b = \sqrt{2/5}$. The purpose is to reduce the computation complexity because the core transform can be realized by shift and addition operations only without multiplication. The quantization process of $\mathbf{Z} = \mathbf{Q}(\mathbf{F})$ for the transformed residual block $\mathbf{F}$ can be expressed as rounding operation on each coefficient of $\mathbf{F}$

$$z_{ij} = \mathrm{round}(f_{ij}/\Delta) \tag{16}$$

where z_{ij} and f_{ij} are coefficients of the quantized transform and unquantized transform blocks of $\mathbf{Z}$ and $\mathbf{F}$, respectively. Δ is the quantization step size, which is determined by the QP factor. On the other hand, the inverse quantization process of $\hat{\mathbf{F}} = \mathbf{Q}^{-1}(\mathbf{Z})$ can be expressed as scaling operation on each coefficients of $\mathbf{Z}$

$$\hat{f}_{ij} = z_{ij} \cdot \Delta \tag{17}$$

where $\hat{f}_{ij}$ are coefficients of the inverse quantized transformed block $\hat{\mathbf{F}}$. In the inverse transform, the core matrix and scale matrix is not the same as those in forward transform.

$$\hat{\mathbf{D}} = \mathbf{C}_b^T(\hat{\mathbf{F}} \otimes \mathbf{Q}_{\mathrm{back}})\mathbf{C}_b \tag{18}$$

TABLE II
Δ_{ij} IN DIFFERENT POSITIONS

Position Region	Position (i, j)	Δ_{ij}
I	(0,0),(0,2),(2,0),(2,2)	Δ / a^2
II	(1,1),(1,3),(3,1),(3,3)	$4\Delta / b^2$
III	Others	$2\Delta / ab$

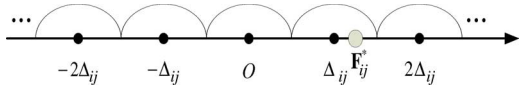

Fig. 2. Quantization diagram.

where $\mathbf{C}_{\mathrm{b}}$ and $\mathbf{Q}_{\mathrm{back}}$ are defined as

$$\mathbf{C}_b = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

$$\mathbf{Q}_{\mathrm{back}} = \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}.$$

In the H.264/AVC, scale transform and quantization are combined together to further reduce computational complexity.

$$z_{ij} = Q\left(f_{ij}^*\right) = \mathrm{round}\left(\frac{f_{ij}^* \cdot q_{ij}}{\Delta}\right) \tag{19}$$

where q_{ij} are the scale coefficients of the $\mathbf{Q}_{\mathrm{forw}}$ matrix and f_{ij}^* are coefficients of the ICT transformed block $\mathbf{F}^*$ using the ICT core matrix.

B. Fast Sum of Squared Difference Algorithm

As illustrated in the last section, the $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ could be determined in transform domain using $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}}) = \|\mathbf{F} - \hat{\mathbf{F}}\|_F^2$, so it is unnecessary to calculate the distortion using the reconstructed block $\mathbf{C}$. The computation of DCT-transformed $\mathbf{F}$ is, however, much more complex than the ICT-transformed $\mathbf{F}^*$ as the scale transform contains fractional coefficients. Thus, if we can build a bridge between $\mathrm{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ and $\mathbf{F}^*$, then we can skip the calculation of $\mathbf{F}$ and $\hat{\mathbf{F}}$. In order to achieve this purpose, we rearrange the quantization and inverse quantization processes of $\hat{\mathbf{F}}$ in terms of quantization and inverse quantization of $\mathbf{F}^*$. The coefficients of $\hat{\mathbf{F}}$ can be expressed as

$$\begin{aligned}\hat{f}_{ij} &= \Delta \cdot \mathrm{round}\left(\frac{f_{ij}}{\Delta}\right) = \Delta \cdot \mathrm{round}\left(\frac{f_{ij}^* \cdot q_{ij}}{\Delta}\right) \\ &= \Delta \cdot \mathrm{round}\left(\frac{f_{ij}^*}{\Delta / q_{ij}}\right) \\ &= \frac{\Delta}{q_{ij}} \cdot \mathrm{round}\left(\frac{f_{ij}^*}{\Delta / q_{ij}}\right) \cdot q_{ij} \\ &= \Delta_{ij} \cdot \mathrm{round}\left(\frac{f_{ij}^*}{\Delta_{ij}}\right) \cdot q_{ij} = \hat{f}_{ij}^* \cdot q_{ij}\end{aligned} \tag{20}$$

where $\Delta_{ij} = \Delta / q_{ij}$ represents the scaled quantization step for ICT-transformed coefficients f_{ij}^* of $\mathbf{F}^*$. The quantization steps

TABLE III
QUANTIZATION POINTS AND QUANTIZATION ZONES

Quantization point	…	$-2\Delta_{ij}$	$-\Delta_{ij}$	O	Δ_{ij}	$2\Delta_{ij}$	…
Quantization sub-zone	…	$(-2.5\Delta_{ij}, -1.5\Delta_{ij})$	$(-1.5\Delta_{ij}, -0.5\Delta_{ij})$	$(-0.5\Delta_{ij}, 0.5\Delta_{ij})$	$(0.5\Delta_{ij}, 1.5\Delta_{ij})$	$(1.5\Delta_{ij}, 2.5\Delta_{ij})$	…

TABLE IV
RELATION BETWEEN Δ_{ij}, QUANTIZATION SUBZONE, AND QUANTIZATION POINTS

Qstep	Position Regions	q_{ij}	Δ^*_{ij}	Quantization range				
Δ	I	a^2	Δ/a^2	Quantization point	…	O	Δ/a^2	…
				Quantization sub-zone	…	$(-\Delta/2a^2, \Delta/2a^2)$	$(\Delta/2a^2, 3\Delta/2a^2)$	…
	II	$b^2/4$	$4\Delta/b^2$	Quantization point	…	O	$4\Delta/b^2$	…
				Quantization sub-zone	…	$(-2\Delta/b^2, 2\Delta/b^2)$	$(2\Delta/b^2, 6\Delta/b^2)$	…
	III	$ab/2$	$2\Delta/ab$	Quantization point	…	O	$2\Delta/ab$	…
				Quantization sub-zone	…	$(-\Delta/ab, \Delta/ab)$	$(\Delta/ab, 3\Delta/ab)$	…

are not all equal for coefficients of $\mathbf{F}^*$ and the values of Δ_{ij} depend on both Δ and the coefficients q_{ij} of $\boldsymbol{Q}_{\text{forw}}$. The values of Δ_{ij} in different positions are shown in Table II. In addition, the inverse quantization of the ICT-transformed coefficients are defined as

$$\hat{f}^*_{ij} = Q^{-1}\left(Q\left(f^*_{ij}\right)\right) = \Delta_{ij} \cdot \text{round}\left(f^*_{ij}/\Delta_{ij}\right). \tag{21}$$

Based on this new relationship, we can reformulate the $\text{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ in terms of ICT transformed coefficients f^*_{ij} as

$$\begin{aligned}\text{SSD}(\mathbf{F}, \hat{\mathbf{F}}) &= \|\mathbf{F} - \hat{\mathbf{F}}\|_F^2 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[f_{ij} - \hat{f}_{ij}]^2 \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[f^*_{ij}\cdot q_{ij} - Q^{-1}\left(Q\left(f^*_{ij}\right)\right)\cdot q_{ij}\right]^2 \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} q_{ij}^2\left[f^*_{ij} - \hat{f}^*_{ij}\right]^2 \end{aligned} \tag{22}$$

This equation indicates that $\text{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ can be directly related to the quantization error of f^*_{ij}. Therefore, we can calculate $\text{SSD}(\mathbf{F}, \hat{\mathbf{F}})$ more easily as we do not need to obtain f_{ij} and $\hat{f}_{ij}$. However, division and multiplication operations are required to perform the quantization and inverse quantization processes of f^*_{ij}. They are normally performed in the following operations:

$$z_{ij} = Q\left(f^*_{ij}\right) = \text{round}\left(\frac{f^*_{ij}}{\Delta_{ij}}\right) \tag{23}$$

$$\hat{f}^*_{ij} = Q^{-1}\left(Q\left(f^*_{ij}\right)\right) = z_{ij}\cdot\Delta_{ij}. \tag{24}$$

In order to make use of the advantage given by (22), we propose to use an iterative table-lookup method for simplifying the quantization process of f^*_{ij} and computing SSD using $\hat{f}^*_{ij}$. Thus, the computational intensive division and multiplication operations can be avoided.

C. Iterative Table-Lookup Quantization Process

Basically, the quantization process is to find the nearest quantized value as shown in Fig. 2, in which $\pm k\Delta_{ij}$ for $k = 0, 1, 2, \ldots, M$ are all quantized values. From the figure we can observe that the quantized value of f^*_{ij} is determined by the subzone that f^*_{ij} is filled in. For instance, if f^*_{ij} is in Δ_{ij}'s corresponding subzone, f^*_{ij} is quantized as Δ_{ij}. In Table III, the quantization points (or quantized values) and their corresponding subzones are listed. The points $(\ldots, -2.5\Delta_{ij}, -1.5\Delta_{ij}, -0.5\Delta_{ij}, 0.5\Delta_{ij}, 1.5\Delta_{ij}, 2.5\Delta_{ij}, \ldots)$ are called boundary points. As long as we can find which subzone f^*_{ij} is located in, it is easy to obtain the quantized value z_{ij} and unquantized value $\hat{f}^*_{ij}$. The proposed iterative table-lookup quantization process is to find f^*_{ij}'s corresponding subzone by searching from the zero point towards positive axis direction by comparing the absolute value of f^*_{ij} with boundary points until f^*_{ij} is smaller than a certain boundary point. To use this approach, the quantization points and boundary points (quantization subzones) are generated in advance at the initial process of encoding process and their

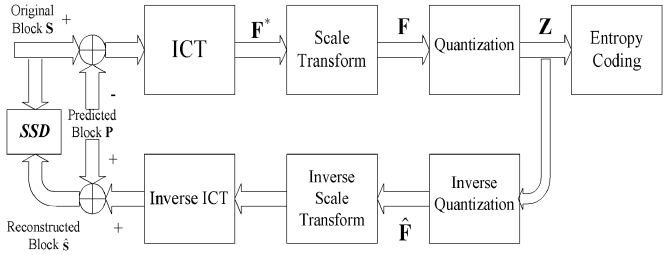

Fig. 3. Architecture of conventional SSD computation.

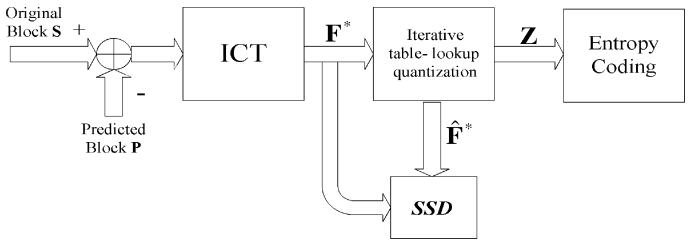

Fig. 4. Architecture of proposed FSSD computation.

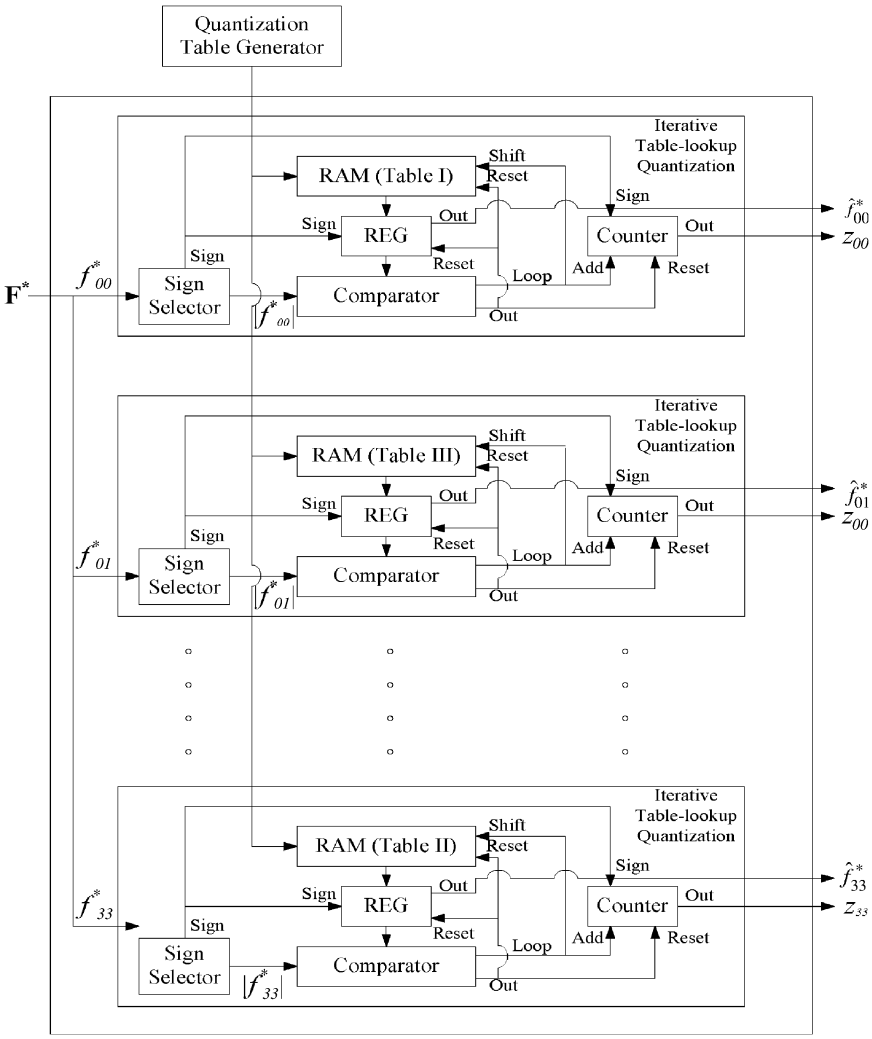

Fig. 5. Hardware structure for iterative table-lookup quantization process.

values are stored in the memory for table lookup. Thus, no extra computation is required in this quantization process for determining the quantization points and quantization subzones. The relation between Δ_{ij}, quantization subzones and quantization points is shown in Table IV. After this quantization process, we can obtain the quantization value z_{ij}, which is equal to the number of comparison times. Besides, we can also get $\hat{f}^*_{ij}$, which is equal to the value of quantization point of f^*_{ij}'s corresponding quantization subzone.

The overall rate-distortion cost computation process using this proposed iterative table-lookup quantization for FSSD computation can be summarized as follows.

Step 1) Compute the predicted block using inter- or intra-frame prediction: $\mathbf{P}$.

Step 2) Compute the residual (difference) block: $\mathbf{D} = \mathbf{S} - \mathbf{P}$.

Step 3) ICT transform the residual block:

$$\mathbf{F}^* = \mathrm{ICT}(\mathbf{D}) = \mathbf{C}_f \mathbf{D} \mathbf{C}_f^T.$$

Step 4) Set $\mathrm{SSD} = 0$ and for $i = 0$ to $N-1$ and $j = 0$ to $N-1$ determine the quantized coefficients z_{ij} and SSD by the following iterative table-lookup quantization process.

Step I: Set $k = 0$ and if $f^*_{ij} < 0$ then set $\mathrm{sign} = -1$, otherwise $\mathrm{sign} = 1$.

Step II: If $|f^*_{ij}| \geq (k + 0.5)\Delta_{ij}$, then $k = k + 1$ and goto Step II.

Step III: Set $z_{ij} = \mathrm{sign} \cdot k$ and $\hat{f}^*_{ij} = \mathrm{sign} \cdot k \cdot \Delta_{ij}$.

Step IV: $\mathrm{SSD} = \mathrm{SSD} + q^2_{ij}(f^*_{ij} - \hat{f}^*_{ij})^2$; If not the last f^*_{ij} coefficient, goto Step I.

Step 5) Entropy code of the $\mathbf{Z}$ to find the number of bits to encode the block: $R = \mathrm{VLC}(\mathbf{Z})$.

Step 6) Calculate the R-D cost: $J_{\mathrm{RD}} = \mathrm{SSD} + \lambda \cdot R$.

In the above procedures, we assume that the quantization points and subzones of different positions (i, j) according to the quantization step value (or QP value) are loaded in the encoder during the initial process. In addition, arithmetic operations are avoided in the proposed algorithm which only requires simple comparison operation between $|f^*_{ij}|$ and boundary points. Therefore, it is very suitable for hardware implementation.

The rate-distortion cost computation using conventional $\mathrm{SSD}(\mathbf{S}, \mathbf{C})$ calculation and the proposed FSSD calculation are compared in Figs. 3 and 4. The major difference is that the conventional quantization process is replaced by the iterative table-lookup quantization and the SSD calculation is preformed

TABLE V
COMPARISON OF THE ENCODING TIME BETWEEN STANDARD ALGORITHM (JM ENCODER) AND THE PROPOSED FSSD ALGORITHMS IN (a) AKIYO, (b) FOREMAN, (c) CONTAINER, and (d) STEFAN

QP	Conventional FSSD Algorithm (ms)	FSSD Algorithm (ms)	Time reduction by iterative table-lookup quantization (%)	Time reduction by inverse transform and pixel reconstruction (%)	Total reduction of encoding time (%)
28	89261	76512	6.61	7.67	14.28
32	85365	72374	6.85	8.37	15.22
36	83705	71731	6.74	7.56	14.30
40	83363	71468	6.68	7.79	14.47

(a)

QP	Conventional FSSD Algorithm (ms)	FSSD Algorithm (ms)	Time reduction by iterative table-lookup quantization (%)	Time reduction by inverse transform and pixel reconstruction (%)	Total reduction of encoding time (%)
28	103621	90812	5.65	6.71	12.36
32	95812	83721	6.15	6.47	12.62
36	89764	78267	6.31	6.50	12.81
40	84427	72891	6.68	6.98	13.66

(b)

QP	Conventional FSSD Algorithm (ms)	FSSD Algorithm (ms)	Time reduction by iterative table-lookup quantization (%)	Time reduction by inverse transform and pixel reconstruction (%)	Total reduction of encoding time (%)
28	101251	90139	4.48	6.49	10.97
32	93789	83394	4.64	6.44	11.08
36	87862	76931	5.12	7.32	12.44
40	82681	72158	5.50	7.23	12.73

(c)

QP	Conventional FSSD Algorithm (ms)	FSSD Algorithm (ms)	Time reduction by iterative table-lookup quantization (%)	Time reduction by inverse transform and pixel reconstruction (%)	Total reduction of encoding time (%)
28	126685	115892	4.06	4.46	8.52
32	114568	103625	4.25	5.30	9.55
36	102638	92258	4.71	5.40	10.11
40	95921	85784	5.08	5.49	10.57

(d)

in the ICT transform domain. The advantage of the proposed method is that we can obtain the SSD without computations of two scale transforms ($\mathbf{F}^* \otimes \mathbf{Q}_{\text{forw}}$ and $\hat{\mathbf{F}} \otimes \mathbf{Q}_{\text{back}}$), quantization $\mathbf{Q}(\mathbf{F})$, inverse quantization $\mathbf{Q}^{-1}(\mathbf{F})$ and inverse ICT transform $(\mathbf{C}_b^T \hat{\mathbf{F}}' \mathbf{C}_b)$. Such computational reduction is much higher than using differential-domain and transform-domain SSDs as expressed in (12) and (14).

Fig. 5 describes a possible hardware structure for the iterative table-lookup quantization implementation. As each coefficient does not depend on the others, the quantization process can be performed in parallel. In the other words, we can quantize all the coefficients of $\hat{f}_{ij}^*$ simultaneously. At first, the quantization points and boundary points (quantization subzones) are generated by quantization table generator at the beginning of encoding process and then stored in the RAM, in which there is a pointer that points to the current boundary point. Then the current boundary point and quantization point are loaded in a register, which will compare with $|f_{ij}^*|$ after the sign of f_{ij}^* is abstracted by a sign selector. If the comparator judges that $|f_{ij}^*|$ is larger, then "Loop" signal takes effect which tells counter to add one and the pointer to shift forward and point to the next boundary point. When $|f_{ij}^*|$ is smaller than the current boundary point, "Out" signal takes effect which resets the pointer to the initial position, asks the counter to output the accumulated value z_{ij} and asks the register to output the current quantization value, $\hat{f}_{ij}^*$. From the structure in Fig. 5, we can find that the proposed table-lookup quantization process is very suitable for hardware implementation, since it does not require complicated operation modules and can execute in the parallel mode.

V. SIMULATION RESULTS

The proposed iterative table-lookup quantization processing and FSSD computation were tested using the first 100 frames from four video sequences (Akiyo, Foreman, Stefan and Container) all in QCIF format 176 × 144. They present different kinds of video sequences respectively: Akiyo (slow motion), Foreman and Container (medium motion), Stefan (high motion). The experiments were carried out in the JVT JM 8.3 encoder. Test parameters are listed below.

Test condition:

1) RDO is enabled;
2) CAVLC is enabled;
3) GOP structure is IPPPIPPP...;
4) QP values are 28, 32, 36 and 40.

The percentage of reduced time of calculating distortion is defined as:

$$\Delta T_{\text{SSD}} = \frac{T_{\text{orgTOT}} - T_{\text{proposedTOT}}}{T_{\text{orgTOT}}} \times 100\% \qquad (25)$$

where T_{orgTOT} is the computation time of the original H.264/AVC encoder using conventional spatial-domain SSD(S,C) algorithm; while $T_{\text{proposedTOT}}$ is the computation time of the H.264/AVC encoder with proposed FSSD algorithm.

TABLE VI
AVERAGE NUMBER OF ITERATIVE TABLE-LOOKUP OPERATION FOR DIFFERENT VIDEOS

QP	Akiyo	Foreman	Container	Stefan
28	0.29	0.40	0.45	0.73
32	0.26	0.37	0.40	0.67
36	0.11	0.16	0.20	0.31
40	0.05	0.08	0.11	0.15

TABLE VII
COMPARISON OF THE RATE-DISTORTION PERFORMANCE BETWEEN STANDARD ALGORITHM (JM ENCODER) AND THE PROPOSED FSSD ALGORITHMS IN (a) AKIYO, (b) FOREMAN, (c) CONTAINER, AND (d) STEFAN

QP	PSNR – dB (Conventional SSD)	PSNR - dB (FSSD)	Bit-rate - Kbps (Conventional SSD)	Bit-rate – Kbps (FSSD)
28	39.94	39.94	218.09	218.09
32	36.95	36.95	154.03	154.18
36	34.30	34.30	111.77	111.84
40	31.43	31.42	80.78	80.71

(a)

QP	PSNR – dB (Conventional SSD)	PSNR - dB (FSSD)	Bit-rate - Kbps (Conventional SSD)	Bit-rate – Kbps (FSSD)
28	37.84	37.82	421.58	420.86
32	34.90	34.87	251.64	251.35
36	32.40	32.40	163.58	163.70
40	29.69	29.67	110.09	109.94

(b)

QP	PSNR – dB (Conventional SSD)	PSNR - dB (FSSD)	Bit-rate - Kbps (Conventional SSD)	Bit-rate – Kbps (FSSD)
28	38.09	38.09	331.82	331.92
32	35.28	35.27	212.09	211.53
36	32.61	32.60	145.46	145.18
40	29.85	29.84	100.13	100.10

(c)

QP	PSNR – dB (Conventional SSD)	PSNR - dB (FSSD)	Bit-rate - Kbps (Conventional SSD)	Bit-rate – Kbps (FSSD)
28	37.16	37.15	891.62	890.88
32	33.49	33.48	597.29	597.58
36	30.22	30.21	388.42	389.12
40	26.91	26.88	242.23	242.06

(d)

Table V(a)–(d) shows the comparison of computation complexity between original JM encoder and our proposed encoder with FSSD algorithm. From the simulation results, we can conclude that the proposed FSSD algorithm can reduce the encoding time by around 10%–15% in different QP values. In which about 4% to 7% reduction is achieved by the iterative table-lookup quantization process and 5% to 7% reduction is achieved by avoiding the inverse transform and pixel reconstruction. Besides, the reduction of low motion sequence (Akiyo) is more than that of high motion sequence (Stefan). The possible reason is that since the prediction accuracy of low motion sequence is better than that of high motion sequence, its residue is smaller and then less time is spent on the iterative table-lookup operation. Table VI shows the average number of table-lookup

TABLE VIII
PERFORMANCE OF FAST SSD ALGORITHMS WITH RATE ESTIMATION IN (a) AKIYO, (b) FOREMAN, (c) CONTAINER, AND (d) STEFAN

QP	Reduction of encoding time (%)	PSNR reduction	Bit rate increase (%)
28	40.85	-0.08	0.08
32	37.66	-0.12	2.84
36	34.91	-0.08	2.63
40	31.19	-0.02	0.47

(a)

QP	Reduction of encoding time (%)	PSNR reduction	Bit rate increase (%)
28	40.03	-0.03	1.01
32	36.43	-0.04	2.15
36	34.32	-0.02	0.62
40	30.45	-0.02	0.24

(b)

QP	Reduction of encoding time (%)	PSNR reduction	Bit rate increase (%)
28	39.84	-0.08	0.91
32	38.02	-0.10	0.02
36	34.50	-0.03	1.61
40	33.51	-0.06	1.59

(c)

QP	Reduction of encoding time (%)	PSNR reduction	Bit rate increase (%)
28	43.00	-0.11	0.51
32	38.57	-0.10	0.53
36	36.09	-0.06	0.52
40	29.75	-0.08	1.31

(d)

operations for different video sequences in various QP values. The result indicates that the iterative table-lookup quantization process does not cost much time so that our proposed quantization method is practical. In addition, these experimental results are obtained using integer rounding precision in the computation of the (22) in the JM encoder. The rate-distortion performance comparison of the conventional SSD and proposed FSSD in terms of PSNR and bit rate are shown in Table VII(a)–(d). These results show that the errors due to the clipping functions in computing the transform-domain SSD is very small and the rate-distortion performance differences between these two methods are always less than 0.1% and 0.03 dB in terms of bit rate and PSNR, respectively.

Since the proposed FSSD algorithm improves the computational efficiency of distortion measure with very little loss of rate-distortion performance, it can be combined with different types of H.264/AVC fast algorithms, such as fast inter/intra-mode selection algorithms and rate estimation algorithm. Here, FSSD algorithm is combined with a conventional rate estimation method to reduce more computation complexity [15]. The way to estimate the number of bits is:

$$\text{Total\_bits} = \alpha \cdot \text{total\_zeros} + \beta \cdot \text{total\_coeff} + \text{SAD} \quad (26)$$

where total_zeros and total_coeff present the number of zeros and the number of nonzero coefficients, respectively, after quantization. SAD is the sum of absolute value of quantized

transform coefficients. The experiences values of α and β are 1 and 3. Experimental results are listed in Table VIII(a)–(d). From the results, the reduction of encoding time is about 30%–40% with ignorable performance degradation. This indicates that the proposed FSSD algorithm can be easily combined with other fast H.264/AVC algorithms and achieves a good tradeoff between the computation complexity and the rate-distortion performance.

VI. Conclusions

In this paper, a new fast SSD computation algorithm is proposed, which can reduce the computation complexity with nearly the same rate-distortion performance of the original spatial-domain SSD method. In the algorithm, it is not necessary to calculate the reconstructed block for each possible mode, and quantization has been substituted by iterative table-lookup operation; therefore, scaling transform, quantization, inverse quantization, inverse scaling transform and inverse core transform can be all skipped. From the simulation results, the proposed algorithm can reduce the encoding time by around 10%–15% in different QP values and sequences. Besides, it can be combined with other fast H.264/AVC algorithms easily in order that better coding efficiency is achieved. Both fundamental analysis and experimental results show the efficiency of the new FSSD algorithm.

References

[1] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. Chichester, U.K.: Wiley, c2003.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] J. Watkinson, *MPEG-2*. Boston, MA: Focal Press, 1999.

[4] F. C. N. Pereira, *The MPEG-4 Book*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[5] V. Roden, "H.261 and MPEG1-a comparison," in *Proc. 1996 IEEE Int. Phoenix Conf.*, Mar. 1996, pp. 65–71.

[6] *Video Coding for Low Bitrate Communication*, ITU-T SG16, ITU-T Rec. H.263, 2000.

[7] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 182–190, Apr. 1996.

[8] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 11, pp. 74–90, Nov. 1998.

[9] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.

[10] Y.-K. Tu, J.-F. Yang, and M.-T. Sun, "Efficient rate-distortion estimation for H.264/AVC coders," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 16, no. 5, pp. 600–611, May 2006.

[11] Joint Video Team (JVT) Reference Software. ver. 6.1d.

[12] C.-P. Fan and J.-F. Yang, "Fast center weighted hadamard transform algorithms," *IEEE Trans. Circuits Syst. Part II, Analog Digit. Signal Process.*, vol. 45, no. 3, pp. 429–432, Mar. 1998.

[13] C.-P. Fan and J.-F. Yang, "Fixed-pipeline two dimensional hadamard transform algorithms," *IEEE Trans. Signal Process.*, vol. 45, no. 6, pp. 1669–1674, Jun. 1997.

[14] A. Hallapuro and M. Karczewicz, *Low Complexity Transform and Quantization* Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Doc. JVT-B038 and JVT-B039, Jan. 2002.

[15] Q. Chen and Y. He, "A fast bits estimation method for rate-distortion optimization in H.264/AVC," presented at the Picture Coding Syst. (PCS 2004), Dec. 2004, paper No. 35.

[16] *"The Matrix Reference Manual"* 2005 [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html, M. Brookes

Lai-Man Po (M'92) received the B.Sc. and Ph.D. degrees in electronic engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 1988 and 1991, respectively.

Since November 1991, he has been with the Department of Electronic Engineering, City University of Hong Kong, and is currently Associate Professor. His research interests are in vector quantization, motion estimation for video compression, and H.264/AVC fast mode decision algorithms.

Kai Guo received the B.Eng. degree in electronic engineering from Xidian University, Xidian, China.

He is currently working toward the M.Phil. degree in the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.