Transform-Exempted Calculation of Sum of Absolute Hadamard Transformed Differences

Ce Zhu, *Senior Member, IEEE*, and Bing Xiong

***Abstract*— Sum of absolute Hadamard transformed differences (SATD) is an important distortion metric applied in the latest video coding standard H.264/AVC, which is an alternative to the sum of absolute differences (SAD) to improve coding efficiency. However, the SATD requires more computation load due to the Hadamard transform involved. Inspired by a geometric interpretation of the simplest two-point SATD calculation, an efficient way to compute the SATD is proposed, which considers the joint effect of the Hadamard transform and the following SAD processing in the SATD calculation and thus enables the computation of SATD without performing the Hadamard transform separately. We further extend the two-point transform-exempted SATD (TE-SATD) computation scheme to four-point and 4 × 4 block SATD calculation. With the same coding performance, the proposed TE-SATD-based fast algorithms can save 38% and 17% operations in computing a 4 × 4 block SATD compared with the conventional SATD calculation and the fast Hadamard transform-based calculation, respectively.**

***Index Terms*— Computation complexity, fast algorithm, H.264/AVC, Hadamard transform, sum of absolute difference (SAD), sum of absolute transformed difference (SATD).**

I. Introduction

THE LATEST video coding standard H.264/AVC coder [1] utilizes several advanced coding techniques to attain significantly higher compression ratio than the previous video coding standards. Among these, the rate-distortion optimization (RDO), which is the procedure conducted to select the best coding mode from all possible modes in both intra- and inter-prediction, is considered to be one of the most important factors contributing to the success of H.264/AVC in terms of compression ratio and visual quality. Nevertheless, this technique increases computational complexity remarkably. To lower the computation burden, the H.264/AVC reference software [2] provides a simplified way to estimate the rate-distortion cost with the prediction error and a simple bit cost estimate for a prediction mode, instead of obtaining the exact value by going through the whole encoding/decoding processes. Two distortion metrics are suggested to measure the prediction error; one being sum of absolute differences (SAD), and the other sum of absolute Hadamard-transformed differences (SATD).

Manuscript received May 19, 2008; revised September 29, 2008. First version published April 7, 2009; current version published August 14, 2009. This paper was recommended by Associate Editor J. Boyce.

The authors are with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore 639798 (e-mail: eczhu@ntu.edu.sg; xiongbing@pmail.ntu.edu.sg).

Digital Object Identifier 10.1109/TCSVT.2009.2020264

Consider a prediction error block of size 4×4, $\boldsymbol{D}_{4\times4}$, which is obtained by subtracting a reference block from the current block in the context of video coding. The SAD and SATD for the block are defined respectively as

$$SAD(\boldsymbol{D}_{4\times4}) = \sum_{i=1}^{4}\sum_{j=1}^{4}\left|d_{ij}\right| \tag{1}$$

$$SATD(\boldsymbol{D}_{4\times4}) = \sum_{i=1}^{4}\sum_{j=1}^{4}\left|t_{ij}\right| \tag{2}$$

where d_{ij} denotes the (i, j)th element of $\boldsymbol{D}_{4\times4}$ and t_{ij} are the (i, j)th element of the following 2-D Hadamard transformed block

$$\boldsymbol{T}_{4\times4} = T(\boldsymbol{D}_{4\times4}) = \boldsymbol{H}_{4\times4}\cdot\boldsymbol{D}_{4\times4}\cdot\boldsymbol{H}_{4\times4}^{T} \tag{3}$$

with $\boldsymbol{H}_{4\times4}$ being a 4×4 Hadamard matrix. For any integer $n > 1$, $2n \times 2n$ Hadamard matrix is given as $\boldsymbol{H}_{2n\times2n} = \boldsymbol{H}_{2\times2} \otimes \boldsymbol{H}_{n\times n}$, where $\otimes$ denotes the Kronecker product and

$$\boldsymbol{H}_{2\times2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4}$$

It can be seen that the Hadamard matrices are symmetric, i.e., $\boldsymbol{H}_{2n\times2n}^{T} = \boldsymbol{H}_{2n\times2n}$.

In the SATD measurement, the Hadamard transform closely imitates the integer cosine transform (ICT) applied to the error block in the coding process of H.264/AVC. Two out of four row vectors (base functions) in the 4×4 Hadamard matrix are equal to those of 4×4 ICT matrix in the H.264/AVC encoder, while the other two row vectors have the similar characteristics as the ICT ones. Therefore, when applying the SATD to measure the prediction distortion, the frequency characteristics of the prediction error are nicely emulated, providing a better estimation of rate-distortion cost in mode selection.

The SATD computation comprises two steps of transforming the block difference and then performing the SAD operation on the transformed elements. Although it usually achieves better coding performance than SAD [3], the sophisticated SATD requires more computation operations due to the Hadamard transform involved. For example, the SATD requires 96 more additions/subtractions in transforming a 4×4 block. Therefore, with the two separate steps involved in the SATD calculation, fast algorithms mainly reduce the computation by individually speeding up either one step, such as the algorithm applying fast Hadamard transform (FHT) to accelerate the SATD calculation [4], [5]. However, the FHT still requires 64 additions [4]. In this letter, we examine the joint effect of both

steps involved in the two-point SATD calculation in the first instance, based on which we derive a simpler way to directly calculate the SATD in the original domain without performing the Hadamard transform. It is thus termed as transform-exempted SATD (TE-SATD) calculation. We then develop the four-point and 4 × 4 block fast SATD calculation based on the two-point TE-SATD scheme. For 4 × 4 block SATD calculation, the proposed methods can save 38% and 17% computation operations compared with the conventional SATD and the FHT-based SATD calculations, respectively, while producing exactly the same coding performance.

The remainder of the letter is organized as follows. Section II presents the proposed TE-SATD fast scheme and discusses computation savings for two-point, four-point, and 4 × 4 block SATD calculations. Comparative simulation results and conclusion are given in Sections III and IV, respectively.

II. Transform-Exempted SATD Calculation

For notational simplicity, we define an operator $SA(\bullet)$ as summing up all the absolute elements in a matrix. Thus, the SAD and the SATD defined in (1) and (2) can be expressed as

$$SAD(\boldsymbol{D}_{4\times4}) = SA(\boldsymbol{D}_{4\times4}) \tag{5}$$

$$SATD(\boldsymbol{D}_{4\times4}) = SA(T(\boldsymbol{D}_{4\times4})). \tag{6}$$

We can see that SAD and SATD are calculated with the same operator but in different domains, one in the original domain and the other in the transform domain. It is easy to obtain the following properties for the operator $SA(\bullet)$.

Property 1: For a matrix $\boldsymbol{D}$ comprising N partitioned submatrices $\boldsymbol{D}_i$, $i = 1, 2, \ldots, N$, the following holds

$$SA(\boldsymbol{D}) = \sum_{i=1}^{N} SA(\boldsymbol{D}_i). \tag{7}$$

Property 2: Defining a vector stacking operator for a matrix $\boldsymbol{D}_{M\times N}$ as $\overrightarrow{\boldsymbol{D}_{M\times N}} = \left[d_{11}\ d_{12}\ d_{13}\ \ldots d_{1N}\ d_{21}\ d_{22}\ldots d_{MN}\right]^T$, we have

$$SA(\overrightarrow{\boldsymbol{D}_{M\times N}}) = SA(\boldsymbol{D}_{M\times N}). \tag{8}$$

In the following, we will develop fast algorithms for the SATD calculation in steps, which features the exemption of Hadamard transform, thereby denoted as transform-exempted SATD (TE-SATD) calculation.

A. TE-SATD Calculation for Two-Point SATD

For a two-point signal $\boldsymbol{D}_{2\times1} = [d_1\ d_2]^T$, the conventional SATD calculation first converts the signal into the transformed domain using the 1-D Hadamard transform as

$$\boldsymbol{T}_{2\times1} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = T(\boldsymbol{D}_{2\times1}) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} d_1 + d_2 \\ d_2 - d_2 \end{bmatrix}. \tag{9}$$

Then the second step is to perform the $SA(\bullet)$ operation over the transformed signal $\boldsymbol{T}_{2\times1}$ to obtain the SATD for $\boldsymbol{D}_{2\times1}$, that is

$$SATD(\boldsymbol{D}_{2\times1}) = SA(\boldsymbol{T}_{2\times1}) = |t_1| + |t_2|. \tag{10}$$

Motivated by the following geometrical observation, we find a new and simplified way to calculate the SATD by combining the two steps in the SATD calculation.

We consider constant distance trajectories of SAD and SATD shown in Fig. 1. The constant distance trajectory consists of the points with a constant value of c measured in SAD or SATD metric, which is denoted as the set of points $\{(d_1,\ d_2) \mid \mathrm{SAD}([d_1\ d_2]^T) = c\}$ or $\{(d_1,\ d_2) \mid \mathrm{SATD}([d_1\ d_2]^T) = c\}$. In the coordinate basis $(d_1,\ d_2)$, the constant SAD trajectory is a square rotated by 45 degrees (shown in the solid line), of which all the points on the square have the constant SAD value of $|d_1| + |d_2| = c$. For the SATD, we rewrite (9) by decomposing the two-point Hadamard transform as

$$\begin{aligned}\boldsymbol{T}_{2\times1} &= \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \\ &= \sqrt{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \cos(-45^\circ) & \sin(-45^\circ) \\ -\sin(-45^\circ) & \cos(-45^\circ) \end{bmatrix}\begin{bmatrix} d_1 \\ d_2 \end{bmatrix}.\end{aligned} \tag{11}$$

It shows that the effect of the two-point Hadamard transform is to rotate the original coordinate by -45° followed by a permutation of the two axes (without considering the scaling effect), resulting in the transformed coordinate basis shown as (t_1, t_2) in the figure. Since SATD is the SAD in the transformed domain, constant SATD trajectory is a rotated square (indicated as the dashed square) with respect to the transformed coordinate basis (t_1, t_2). Interestingly, this trajectory turns out to be an upright square if it is viewed in the original basis (d_1, d_2), which just corresponds to a constant trajectory of the function $max(|d_1|, |d_2|)$. This suggests that for any $[d_1\ d_2]^T$, its SATD value can be directly calculated from $max(|d_1|, |d_2|)$ (ignoring the scale factor) in the original domain without performing the Hadamard transform, as the following theorem states.

Theorem 1: The SATD for $\boldsymbol{D}_{2\times1} = [d_1\ \ d_2]^T$ can be calculated as $2 \cdot max(|d_1|, |d_2|)$.

Proof: The SATD for $\boldsymbol{D}_{2\times1}$ is

$$SATD(\boldsymbol{D}_{2\times1}) = |d_1 + d_2| + |d_1 - d_2| \tag{12}$$

if $|d_1| > |d_2|$, $d_1 + d_2$ and $d_1 - d_2$ have identical signs. Otherwise, they have opposite signs. Thus

$$SATD(\boldsymbol{D}_{2\times1}) = \begin{cases} 2 \times |d_1| & |d_1| > |d_2| \\ 2 \times |d_2| & |d_1| \le |d_2| \end{cases} \tag{13}$$

which is equivalent to $2 \times max(|d_1|, |d_2|)$. Proof completed. ■

This theorem gives a fast algorithm, i.e., transform-exempted SATD or TE-SATD, to compute the SATD for a 2 × 1 block without performing the Hadamard transform. Compared with the conventional two-point SATD calculation scheme, the proposed TE-SATD can save a substantial amount of computation as summarized in the following.

1) The TE-SATD requires only two absolute operations and one comparison for taking the maximum. The weight

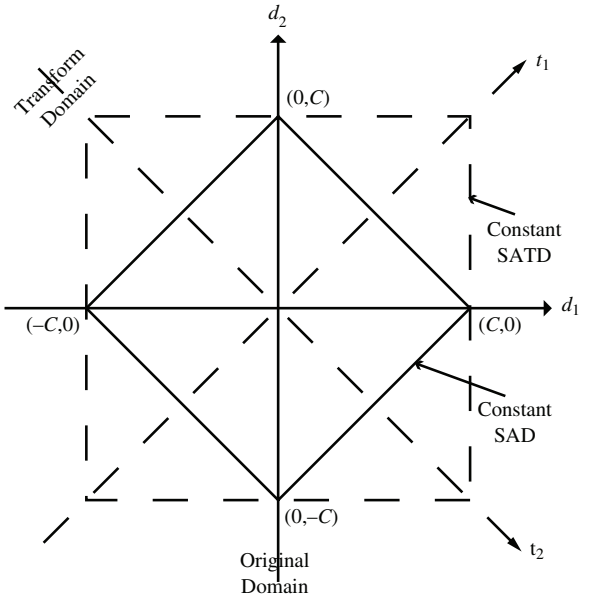

Fig. 1. Geometric comparison of constant SAD and SATD trajectories.

of two may be implemented by a shift operation and the scaling constant can be actually ignored without affecting the SATD-based comparison result.

2) In the conventional SATD calculation, the 1-D Hadamard transform in (9) requires two addition/subtraction operations (no fast algorithm in this case), and the $SA(\bullet)$ operation in (10) needs another two absolute operations and one addition. Therefore, by performing the two steps separately, the conventional SATD calculation requires totally two absolute and three addition/subtraction operations for a two-point signal.

In order to facilitate comparison of computation complexity, we conducted the following simulation. We performed the absolute, addition, subtraction, and comparison operations for 100 million times respectively on a laptop of Intel Pentium Mobile 1.7-GHz CPU and 965-MB memory running Microsoft Windows XP, which took 390, 380, 371, and 371 ms, respectively. Note that in the experiment, the absolute operation was implemented using a lookup table (the same way as in H.264/AVC reference software JM 8.5). As a reviewer pointed out, on older architectures, *max* operation would get translated to a branch instruction; however, in modern CPUs with SSE support, *max* usually takes similar time as other single-cycle instructions. Therefore, we can simply treat these operations as addition-equivalent operations. From the above analysis for the two-point SATD computation, we can see that the TE-SATD calculation scheme needs totally three addition-equivalent operations versus five operations in the conventional SATD calculation, thus saving $(5-3)/5(=40\%)$ computation.

B. TE-SATD Calculation for Four-Point SATD

Theorem 2: The SATD for a four-point signal $\boldsymbol{D}_{4\times1} = [d_1\ d_2\ d_3\ d_4]^T$ can be calculated as

$$SATD(\boldsymbol{D}_{4\times1}) = 2\cdot[max(|d_1+d_3|, |d_2+d_4|) + max(|d_1-d_3|, |d_2-d_4|)]. \quad (14)$$

Proof: Applying matrix partition and *Property 1* of the $SA(\bullet)$ operator, we have

$$\begin{aligned} SATD(\boldsymbol{D}_{4\times1}) &= SA(T(\boldsymbol{D}_{4\times1})) = SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{D}_{4\times1}) \\ &= SA\left(\begin{bmatrix}\boldsymbol{H}_{2\times2} & \boldsymbol{H}_{2\times2}\\ \boldsymbol{H}_{2\times2} & -\boldsymbol{H}_{2\times2}\end{bmatrix}\begin{bmatrix}\boldsymbol{D}_{1,2}\\ \boldsymbol{D}_{3,4}\end{bmatrix}\right) \\ &= SA(\boldsymbol{H}_{2\times2}(\boldsymbol{D}_{1,2}+\boldsymbol{D}_{3,4})) \\ &\quad + SA(\boldsymbol{H}_{2\times2}(\boldsymbol{D}_{1,2}-\boldsymbol{D}_{3,4})) \end{aligned} \quad (15)$$

where $\boldsymbol{D}_{1,2} = [d_1\ d_2]^T$ and $\boldsymbol{D}_{3,4} = [d_3\ d_4]^T$ are the submatrices of $D_{4\times1}$. In the final step of the above equation, the SATD for the 4×1 block has been converted to two individual 2×1 SATD calculations for $(\boldsymbol{D}_{1,2}+\boldsymbol{D}_{3,4})$ and $(\boldsymbol{D}_{1,2}-\boldsymbol{D}_{3,4})$, respectively. Therefore, applying Theorem 1 to the two 2×1 SATD calculations, we obtain $SATD(\boldsymbol{D}_{4\times1}) = 2\cdot[max(|d_1+d_3|, |d_2+d_4|)+max(|d_1-d_3|, |d_2-d_4|)]$. Proof completed. ■

The computation complexity for 4×1 block SATD calculation is discussed comparatively in the following.

1) With the transform-exempted SATD calculation shown in (14), the total number of operations required is 11 (not counting the weighting of 2).
2) With the conventional SATD calculation, 12 additions/subtractions are needed for Hadamard transform and 7 operations (four absolute operation and three additions) for the following $SA(\bullet)$ operator, totaling 19 operations. We can see that 42% computation saving can be obtained with the TE-SATD for a 4×1 signal.
3) For the 4×1 SATD calculation, fast Hadamard transform (FHT) is applicable. It is known that the FHT for an N-point signal requires $N\log_2 N$ subtraction/addition operations (in the 1-D case) with the butterfly structure. Therefore, the FHT for a four-point signal demands eight operations, plus seven operations for the $SA(\bullet)$ operator totaling 15 operations for the SATD calculation. The proposed TE-SATD can still save $(15-11)/15 = 27\%$ computation when compared to the FHT-based SATD calculation.

C. TE-SATD Calculation for 4×4 Block SATD

In the H.264/AVC reference software, 2-D 4×4 block Hadamard transform-based SATD is employed. The SATD for such a 4×4 block $\boldsymbol{D}_{4\times4}$ is

$$SATD(\boldsymbol{D}_{4\times4}) = SA(T(\boldsymbol{D}_{4\times4})) = SA\left(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{D}_{4\times4}\cdot\boldsymbol{H}_{4\times4}^T\right). \quad (16)$$

Two fast methods to calculate the 4×4 block SATD are developed based on the 4×1 and 2×1 TE-SATD, respectively, which are elaborated as follows.

Method 1: 4×1 TE-SATD-based 4×4 SATD Calculation. This method is developed based on the TE-SATD calculation for 4×1 blocks introduced in the preceding section. First, we perform a 1-D Hadamard transform for $\boldsymbol{D}_{4\times4}$ to obtain $\boldsymbol{P}_{4\times4} = \boldsymbol{D}_{4\times4}\cdot\boldsymbol{H}_{4\times4}^T$. Denote $\boldsymbol{P}_j = [p_{1j}\ p_{2j}\ p_{3j}\ p_{4j}]^T$, $j = 1, 2, 3, 4$, as the jth column vector in $\boldsymbol{P}_{4\times4}$, where p_{ij} is the element of the ith row and jth column in $\boldsymbol{P}_{4\times4}$. Applying matrix partition

and Property 1 of the $SA(\bullet)$ operator, we can obtain

$$\begin{aligned} SATD(\boldsymbol{D}_{4\times4}) &= SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_{4\times4}) \\ &= SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_1)+SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_2) \\ &\quad + SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_3)+SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_4). \end{aligned} \quad (17)$$

In this equation, each $SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_j)$ can be computed using the 4×1 fast TE-SATD in Theorem 2. Therefore we can achieve the fast 4×4 SATD calculation based on the 4×1 TE-SATD, which can save a considerable amount of computation as discussed in the following.

1) We first study the computation complexity for the conventional 4×4 SATD calculation, in which the 2-D 4×4 Hadamard transform requires 96 operations and the following $SA(\bullet)$ operation over the 4×4 transformed matrix, which needs additional 31 operations (16 absolute operations and 15 additions). Thus, the conventional SATD calculation for a 4×4 block requires $96+31=127$ operations in total.
2) If using the FHT, the transform step costs 64 operations, thus totaling $64+31=95$ operations for the SATD calculation based on the FHT.
3) In the proposed method, the 1-D Hadamard transform still needs 32 operations with the FHT. Then the 4×1 TE-SATD based calculation of $SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_j)$, $j=1,2,3,4$, will cost totally 11×4 operations. Finally, three additions are used to sum up the four $SA(\boldsymbol{H}_{4\times4}\cdot\boldsymbol{P}_j)$. Therefore, the proposed fast method demands 79 operations in total.

Compared to the conventional SATD and the FHT-based SATD calculations, the proposed Method 1 can save 48 operations (or equivalently around 38% computation) and 16 operations (about 17% computation) for 4×4 SATD calculation, respectively.

Method 2: 2×1 TE-SATD-based 4×4 SATD Calculation.

Theorem 3: The SATD for a 4×4 block $\boldsymbol{D}_{4\times4}$ can be computed as $SATD(D_{4\times4})=SA(H_{16\times16}\overrightarrow{D_{4\times4}})$.

TABLE I
COMPUTATIONAL COMPARISON OF CONVENTIONAL, FHT-BASED AND TE-SATD-BASED SATD CALCULATIONS: "ADD," "ABS" AND "CMP" REFER TO ADDITION/SUBTRACTION, ABSOLUTE AND COMPARISON OPERATIONS, RESPECTIVELY

Method \ Block	2×1	4×1	4×4
Convent.	5 ($3add+2abs$)	19 ($15add+4abs$)	127 ($111add+16abs$)
FHT	–	15 ($11add+4abs$)	95 ($79add+16abs$)
TE-SATD	3 ($2abs+1cmp$)	11 ($5add+4abs+2cmp$)	79 ($55add+16abs+8cmp$)

Proof: It is known [6] that the vector stacking operator has the following property:

$$\overrightarrow{\boldsymbol{A}_{N\times N}\cdot\boldsymbol{B}_{N\times N}\cdot\boldsymbol{C}_{N\times N}}=\left(\boldsymbol{C}^T_{N\times N}\otimes\boldsymbol{A}_{N\times N}\right)\cdot\overrightarrow{\boldsymbol{B}_{N\times N}}. \quad (18)$$

Then we have (also applying Property 2 of the $SA(\bullet)$ operator)

$$\begin{aligned} SATD(\boldsymbol{D}_{4\times4}) &= SA\left(\overrightarrow{\boldsymbol{H}_{4\times4}\cdot\boldsymbol{D}_{4\times4}\cdot\boldsymbol{H}^T_{4\times4}}\right) \\ &= SA((\boldsymbol{H}_{4\times4}\otimes\boldsymbol{H}_{4\times4})\overrightarrow{\boldsymbol{D}_{4\times4}}) \\ &= SA(\boldsymbol{H}_{16\times16}\overrightarrow{\boldsymbol{D}_{4\times4}}). \end{aligned} \quad (19)$$

Proof completed. ■

Based on the theorem, we can achieve fast 4×4 SATD calculation using the 2×1 TE-SATD scheme as follows. We first arrange $\boldsymbol{H}_{16\times16}$ and $\overrightarrow{\boldsymbol{D}_{4\times4}}$ into 2×2 partitioned sub-matrices and 2×1 partitioned sub-vectors, respectively. Performing the partitioned sub-matrices multiplication and applying Property 1 of the $SA(\bullet)$ operator, we can get (20), where $\boldsymbol{H}=\boldsymbol{H}_{2\times2}$, and $\boldsymbol{D}_{ij,mk}=[d_{ij}\ d_{mk}]^T$ with d_{ij} being the ith row, jth column element of $\boldsymbol{D}_{4\times4}$. It can be found that each $SA(\bullet)$ in (20) is actually a 2×1 SATD calculation. Therefore, the computation of 4×4 SATD is translated into a sum of eight 2×1 SATD calculations. Then we can apply

$$\begin{aligned} SATD(\boldsymbol{D}_{4\times4}) &= \boldsymbol{SA}\left(\begin{bmatrix} \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} \\ \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} \\ \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} \\ \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} \\ \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} \\ \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} \\ \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} \\ \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} \end{bmatrix}_{16\times16} \begin{bmatrix} \boldsymbol{D}_{11,12} \\ \boldsymbol{D}_{13,14} \\ \boldsymbol{D}_{21,22} \\ \boldsymbol{D}_{23,24} \\ \boldsymbol{D}_{31,32} \\ \boldsymbol{D}_{33,34} \\ \boldsymbol{D}_{41,42} \\ \boldsymbol{D}_{43,44} \end{bmatrix}_{16\times1}\right) \\ &= \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}+\boldsymbol{D}_{13,14}+\boldsymbol{D}_{21,22}+\boldsymbol{D}_{23,24}+\boldsymbol{D}_{31,32}+\boldsymbol{D}_{33,34}+\boldsymbol{D}_{41,42}+\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}-\boldsymbol{D}_{13,14}+\boldsymbol{D}_{21,22}-\boldsymbol{D}_{23,24}+\boldsymbol{D}_{31,32}-\boldsymbol{D}_{33,34}+\boldsymbol{D}_{41,42}-\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}+\boldsymbol{D}_{13,14}-\boldsymbol{D}_{21,22}-\boldsymbol{D}_{23,24}+\boldsymbol{D}_{31,32}+\boldsymbol{D}_{33,34}-\boldsymbol{D}_{41,42}-\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}-\boldsymbol{D}_{13,14}-\boldsymbol{D}_{21,22}+\boldsymbol{D}_{23,24}+\boldsymbol{D}_{31,32}-\boldsymbol{D}_{33,34}-\boldsymbol{D}_{41,42}+\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}+\boldsymbol{D}_{13,14}+\boldsymbol{D}_{21,22}+\boldsymbol{D}_{23,24}-\boldsymbol{D}_{31,32}-\boldsymbol{D}_{33,34}-\boldsymbol{D}_{41,42}-\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}-\boldsymbol{D}_{13,14}+\boldsymbol{D}_{21,22}-\boldsymbol{D}_{23,24}-\boldsymbol{D}_{31,32}+\boldsymbol{D}_{33,34}-\boldsymbol{D}_{41,42}+\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}+\boldsymbol{D}_{13,14}-\boldsymbol{D}_{21,22}-\boldsymbol{D}_{23,24}-\boldsymbol{D}_{31,32}-\boldsymbol{D}_{33,34}+\boldsymbol{D}_{41,42}+\boldsymbol{D}_{43,44})) \\ &\quad + \boldsymbol{SA}(\boldsymbol{H}(\boldsymbol{D}_{11,12}-\boldsymbol{D}_{13,14}-\boldsymbol{D}_{21,22}+\boldsymbol{D}_{23,24}-\boldsymbol{D}_{31,32}+\boldsymbol{D}_{33,34}+\boldsymbol{D}_{41,42}-\boldsymbol{D}_{43,44})) \end{aligned} \quad (20)$$

TABLE II
COMPARISON OF SAD, CONVENTIONAL SATD, FHT-BASED SATD AND TE-SATD IN H.264/AVC
(TIME: IN SECONDS, Y-PSNR: IN DB, BIT RATE: KB/S@15-HZ)

Metric / Sequence	SAD			Conventional SATD			FHT Based SATD			TE-SATD		
	Time	Y-PSNR	Bit rate	Time	Y-PSNR	Bit rate	Time	Y-PSNR	Bit rate	Time	Y-PSNR	Bit rate
Foreman.cif	36.61	35.70	300.24	106.73	36.09	300.59	86.43	36.09	300.59	61.11	36.09	300.59
Coastguard.cif	38.27	32.99	674.83	112.34	33.20	686.01	90.10	33.20	686.01	62.53	33.20	686.01
Dancer.cif	36.98	38.47	334.89	107.11	38.83	331.82	85.78	38.83	331.82	61.72	38.83	331.82
Goldfish.cif	24.95	38.10	392.69	73.84	38.41	388.99	59.52	38.41	388.99	41.60	38.41	388.99
BBC.cif	37.18	33.69	757.82	108.81	33.95	752.05	86.27	33.95	752.05	61.45	33.95	752.05
Akio.cif	12.80	42.00	106.90	41.67	42.26	106.24	34.08	42.26	106.24	23.66	42.26	106.24
Stefan.qcif	8.68	32.60	242.72	25.87	32.76	242.96	20.31	32.76	242.96	14.71	32.76	242.96

the fast 2×1 TE-SATD algorithm to accelerate the SATD calculation for 4×4 blocks.

Now we discuss the computational complexity of this method. First we need to perform the processing (addition/subtraction) of the eight partitioned 2×1 vectors of $\boldsymbol{D}_{4\times4}$, shown in the inner parenthesis of (20). It can be found that the processing of the eight 2×1 vectors is an eight point vector-based Hadamard transform, where one point refers to a 2×1 vector. Therefore, we can still apply the same FHT butterfly structure to obtain vector-based Hadamard transform, which requires totally 48 operations (24×2, 24 is the operation number for eight point FHT) considering that one point here is a 2×1 vector containing two elements. Then eight 2×1 TE-SATD calculations are needed, each of which requires three operations. Another seven additions are used to sum up the eight 2×1 TE-SATD results to obtain the final 4×4 SATD value. Thus, this TE-SATD method for 4×4 blocks requires totally 79 $(48 + 3 \times 8 + 7)$ operations, same as that by the proposed Method 1.

To summarize the computational complexity of the different SATD calculation methods for blocks of 2×1, 4×1, and 4×4, Table I tabulates the required operation numbers for SATD computation using the conventional SATD, FHT-based SATD, and TE-SATD, respectively. It can be seen that the proposed TE-SATD can save considerable computation operations for these different sizes of block SATD calculation.

Remarks: After we completed the writing of this letter and were ready to submit for the first review, we came across the patent [7] on fast SATD estimation which was publicized recently. In that patent [7], the inventor described a fast SATD estimation method that can achieve the same computation saving as ours. The invention is to eliminate a recursion operation (a butterfly step) in the FHT butterfly architecture for the SATD calculation. More specifically, the patented method is to perform butterfly steps in the FHT structure except the final step that can be saved by applying the equation $|a+b|+|a-b| = 2\cdot max(|a|, |b|)$ to every two neighboring units for obtaining the SATD directly. We would like to highlight that this letter is the result of our independent work without having any idea of the patent until we completed the work and the writing. We started the research originally by comparing a few distortion metrics including SAD and SATD, and investigating fundamental reasons why SATD normally outperforms SAD. We then found that the geometric relationship between SAD and SATD in the constant distance trajectory shown in Fig. 1, which clearly shows that the two-point SATD can be obtained as the maximum of two elements in the original domain without performing the transform. Inspired by this, we obtained the simplified two-point (2×1 block) transform-exempted SATD (TE-SATD) calculation shown in Theorem 1. We then extended to the simplified four-point (4×1 block) TE-SATD calculation in Theorem 2. With the two theorems, we have derived mathematically two fast 4×4 block SATD calculation methods based on 4×1 and 2×1 TE-SATD schemes, respectively. In short, our progressive path was to develop a transform-exempted two-point SATD calculation scheme first, then extending to 1-D Hadamard transform-based four-point SATD calculation and finally to 2-D Hadamard transform-based 4×4 block SATD calculation. It can be seen that although equivalent results are obtained, our analytical approach and development methods are quite different from the patented one.

III. EXPERIMENTAL RESULTS

Comparative experiments were conducted by using SAD and SATD with the different calculation schemes in H.264 video coding [1], [2]. Considering that the proposed two TE-SATD based methods for 4×4 blocks are equivalent in computation complexity, we only tested the first method in this experiment (the first algorithm is straightforward for software programming, while the second method can facilitate the generalization to other $N \times N$ block fast SATD calculation based on the 2×1 TE-SATD). Seven video sequences (six CIF and one QCIF) were considered, and the first 80 frames of each sequence were tested. The coding structure of IPPPPIPPPP was used in the experiment. The search range was a square region of $[-8, +8]$ and the quantization parameter (QP) was set to 30. All the block partition sizes recommended in H.264 for motion compensation were enabled, the fast motion estimation was disabled, and the number of reference frames was set to five. The RD-optimized mode decision was not used. The experiments were conducted on a desktop PC of Intel(R) Core(TM)2 CPU 6400 @ 2.13 GHz and 0.98 GB RAM running Microsoft Windows XP operating system. The coding results using SAD, conventional SATD (without FHT), FHT-based SATD, and TE-SATD-based SATD are compared in Table II, in terms of average Y-PSNR,

the associated bit rate, and the total computation time for SAD/SATD calculation. Note that the time counting may exhibit some deviations and may not be accurate especially for the counting of a short time interval of SAD and SATD calculation in the program. Similar program code structures were used in the different SATD implementations, to make a (relatively) fair comparison of computation time. The table reports averaged time over two runs.

From the table, it can be seen that the coding performance (PSNR and bit rate) for the conventional SATD, the FHT-based SATD, and the proposed TE-SATD-based algorithm are exactly the same, which is better than that of SAD. The TE-SATD-based calculation is faster than both the conventional SATD and the FHT-based SATD for all the seven testing sequences.

IV. Conclusion

We have investigated a fast SATD calculation in this letter. Inspired by the geometric interpretation of the two-point SATD calculation, we have developed the simplified TE-SATD calculation scheme by considering the joint result of the Hadamard transform and the following SAD processing. We have then extended the two-point TE-SATD scheme to the four-point and 4×4 block fast SATD calculation, which have exhibited considerable speedup gains over the conventional and FHT based SATD calculation methods.

References

[1] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264\ISO/IEC 14496-10 AVC)*, Joint Video Team (JVT) ITU-T and ISO/IEC JTC 1, ISO/IEC MPEG and ITU-T VCEG, JVT-G050, Mar. 2003.

[2] *Joint Video Team (JVT) Reference Software (Version 8.5)* [Online]. Available: http://iphome.hhi.de/suehring/tml/

[3] L. M. Po and K. Guo, "Transform-domain fast sum of the squared difference computation for H.264/AVC rate-distortion optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 6, pp. 765–773, Jun. 2007.

[4] C. H. Tseng *et al*., "Enhanced intra-4 × 4 mode decision for H.264/AVC coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 1027–1032, Aug. 2006.

[5] C. P. Fan and J. F. Yang, "Fixed-pipeline 2-D Hadamard transform algorithms," *IEEE Trans. Signal Process.*, vol. 45, no. 6, pp. 1669–1674, Jun. 1997.

[6] A. Graham, *Kronecker Products and Matrix Calculus with Applications*. New York: Wiley, 1981.

[7] F.-D. Jou, "Method for fast SATD estimation," U.S. Patent 2007/019862 A1, Aug. 23, 2007.