Fast DST-VII/DCT-VIII With Dual Implementation Support for Versatile Video Coding

Zhaobin Zhang, Xin Zhao, Xiang Li, *Senior Member, IEEE*, Li Li, *Member, IEEE*, Yi Luo, Shan Liu, and Zhu Li, *Senior Member, IEEE*

***Abstract*— The Joint Video Exploration Team (JVET) recently launched the standardization of the next-generation video coding named Versatile Video Coding (VVC) with the inherited technical framework from its predecessor High-Efficiency Video Coding (HEVC). The simplified Enhanced Multiple Transform (EMT) has been adopted as the primary residual coding transform solution, termed Multiple Transform Selection (MTS). In MTS, only the transform set consisting of DST-VII and DCT-VIII remains, excluding the other transform sets and the dependency on intra prediction modes. Significant coding gains are achieved by introducing new DST/DCT transforms, but the full matrix implementation is relatively costly compared to partial butterfly in terms of both software run-time and operation counts. In this work, we exploit the inherent features existing in DST-VII and DCT-VIII. Instead of repeating the element-wise additions and multiplications in full matrix operation, these features can be leveraged to achieve more efficient implementations which only use partial elements to derive the identical results. Existing transform matrices are further tuned to utilize these (anti-)symmetric features. A partial butterfly-type fast algorithm with dual-implementation support is proposed for DST-VII/DCT-VIII transform in VVC. Complexity analysis including operation counts and software run-time are conducted to validate the effectiveness. In addition, we prove the features are perfectly supported by theory. The proposed fast methods achieve noticeable software run-time savings without compromising on coding performance by comparing with the VVC Test Model VTM-3.0. It is shown that under Common Test Condition (CTC) with inter MTS enabled, an average of 9%, 0%, and 3% decoding time savings are achieved for All Intra (AI), Random Access (RA) and Low Delay B (LDB), respectively. Under low QP test condition with inter MTS enabled, the proposed fast methods achieve 1%, 2% and 4% decoding time savings on average for AI, RA, and LDB, respectively.**

***Index Terms*— Versatile video coding (VVC), fast DST-VII/DCT-VIII, multiple transform selection (MTS), Enhanced multiple transform (EMT), adaptive multiple transform (AMT), VVC test model (VTM).**

Manuscript received January 24, 2020; accepted February 15, 2020. Date of publication February 28, 2020; date of current version January 7, 2021. This work was supported in part by a grant from NSF I/UCRC under Award 1747751 and in part by the Tencent Media Lab. This article was recommended by Associate Editor Dr. B. Jeon. *(Corresponding author: Zhu Li.)*

Zhaobin Zhang, Li Li, and Zhu Li are with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City, Kansas, MO 64110 USA (e-mail: zzktb@mailumkc.edu; lil1@umkc.edu; lizhu@umkc.edu).

Xin Zhao, Xiang Li, Yi Luo, and Shan Liu are with Tencent America, LLC., Palo Alto, CA 94306 USA (e-mail: xinzzhao@tencent.com; xlxiangli@tencent.com; juvenalluo@tencent.com; shanl@tencent.com).

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

Digital Object Identifier 10.1109/TCSVT.2020.2977118

I. Introduction

ALTHOUGH deep learning-based methods for image/video coding [1]–[6] have achieved remarkable progress, there are still many issues need to be solved to be widely used in real applications. Conventional codecs are still playing an indispensable role in industrial application scenarios. Since October 2015, ISO/IEC MPEG and ITU-T VCEG have been working together as the Joint Video Exploration Team (JVET) to explore the state-of-the-art techniques and prepare for the next-generation video coding standards [7] with capability beyond HEVC, termed Versatile Video Coding (VVC). VVC inherits the block-based hybrid video coding framework from its predecessors H.265/HEVC [8] and H.264/AVC [9], but introduces new block partitioning schemes. It supports up to 128 × 128 Coding Tree Units (CTU) with recursive quadtree (QT) and nested recursive multi-type tree (MTT) partitioning. Various techniques have been incorporated to improve the compression efficiency in the under-development VVC Test Model (VTM). The primary intra prediction tools include up to 65 prediction angles, prediction filtering using neighboring reference samples and cross-component linear model (CCLM) prediction (Y to Cb, Cr). The primary inter prediction tools include affine motion compensation (4 × 4 subblocks), improved temporal merge motion vector (MV) predictors (8 × 8 subblocks) and Adaptive Motion Vector Resolution switches between 1/4, 1, 4 sample accuracy for MVD.

In hybrid video coding frameworks, two techniques of crucial importance to achieve efficient compression efficiency are prediction and residual transform coding. The prediction process focuses on removing the statistical redundancy between the current block and the reference block and transform coding deals with the inter-pixel correlations which is typically done with linear transforms. A variety of transform schemes have been developed in the literature among which the DCT type II (DCT-II) [10] becomes the most popular solution due to its superior capability of balancing the coding efficiency and the time complexity [11].

In regards to energy compaction performance, it is theoretically proven that DCT-II can efficiently approximate the optimal signal-dependent Karhunen-Loève transform (KLT) [12]–[15] under the first-order stationary Markov assumption. However, the drastic dynamics existing in natural image content are not always following the first-order Markov condition [16].

To better capture the dynamic characteristics of image data content, numerous of transform schemes [17]–[25] have been proposed in the past decades. But those methods suffer from either limited coding efficiency or impractical complexity that hinders the application on video coding codecs. The noteworthy milestone comes when the Enhanced Multiple Transform (EMT), *i.e.,* Adaptive Multiple Transform [12] is proposed. In EMT, four additional transforms including DST-VII, DST-I, DCT-V, and DCT-VIII are introduced. It is reported to achieve -3.1% BD-Rate reduction for AI configuration, and up to -3.6% and -4.0% BD-Rate reduction on 2K and 4K content, respectively [11] which makes it one of the rarest techniques that can achieve more than 2% coding gains since HEVC.

EMT serves as the foundation and prototype of developing the VVC transform solutions due to its superior capability. Since there are five kinds of transforms need to be evaluated to select the optimal category, the encoder run-time complexity is very expensive. In the recent VVC working draft [26], the simplified EMT, named Multiple Transform Selection (MTS) is adopted as the primary residual transform solution after comprehensive consideration about relative merits. In MTS, only the transform set consisting of DST-VII and DCT-VIII remains, excluding the other transform sets and the dependency on intra prediction modes [26], [27].

In VVC, there are two types of MTS, including explicit MTS (with signaling) and implicit MTS (without signaling). The implicit MTS applies DST-VII/DCT-VIII based on block size information (small blocks always use DST-VII for intra), and there is no transform type signaling. The switching between explicit and implicit MTS is done by high-level syntax flags, such that the encoder could choose either explicit MTS or implicit MTS, based on the different trade-off between coding performance and encoder complexity. More detailed design regarding implicit MTS can be found in [28]. It should be noted that our fast method applies to both explicit MTS and implicit MTS, as long as DST-VII or DCT-VIII is used.

This paper is an extended version of our previous work [7]. The major changes and additional contributions include the updated content according to current VVC development status, finer description of the fast methods, theoretical analysis and proof, evaluation results with a newer test model under low QP test condition with and without inter MTS enabled. The proposed scheme was adopted in the 13th JVET Meeting of ITU-T VCEG and ISO/IEC as the primary transform solution [29]. The contributions can be summarized as follows.

- The inherent characteristics of DST-VII and DCT-VIII transforms are exploited to derive the partial butterfly-type features. Based on the features, a fast DST-VII/DCT-VIII algorithm is devised for VVC with finer description.
- The proposed fast method supports both full matrix multiplication and fast implementation with an approximate of 50% operation counts reduction.
- To compensate the rounding error deviation, the transform matrices are tuned to satisfy the (anti-)symmetric features without apparent side impacts on coding efficiency.

TABLE I
BASIS FUNCTIONS OF DCT-II, DST-VII AND DCT-VIII FOR N-POINT INPUT

Transform Type	Basis Function $T_i(j)$, $i, j = 0, 1, \ldots, N-1$
DCT-II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos(\frac{\pi \cdot i \cdot (2j+1)}{2N})$, where $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1})$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \cos(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2})$

- We prove the (anti-)symmetric features leveraged to devise the fast algorithm in theory.
- Extensive experiments are conducted under varied test conditions. The proposed scheme achieves significant coding time reduction by comparing with VTM-3.0, yet still maintains almost the same coding efficiency performance.

The remainder of this paper is organized as follows. Section II introduces the sinusoidal transforms involved in this paper and reveals corresponding characteristics. Section III elaborates the technical details of the proposed fast method. We prove the features in Section IV. Section V performs the complexity analysis, in terms of both the number of arithmetic operations and software execution time. Section VI shows the experimental results. The paper is summarized and concluded in Section VII.

II. DISCRETE SINUSOIDAL TRANSFORM FAMILY

The discrete sinusoidal transform family [30] covers the well-known discrete Fourier transform, cosine transform, sine transform and the Karhunen-Loève transform. Among all the members, there are eight kinds of transforms based on cosine functions and another eight kinds of transforms based on sine functions, namely DCT-I, DCT-II, …, DCT-VIII and DST-I, DST-II, …, DST-VIII, respectively. Variants of discrete cosine and sine transforms are derived from different symmetry of their symmetric-periodic sequences [31]. The transform basis functions of selected types of DCT and DST as used in this paper, *i.e.,* DCT-II, DST-VII, and DCT-VIII are formulated in Table I.

To better understand the sinusoidal families involved in this paper, the first four principal basis functions of DCT-II, DST-VII, and DCT-VIII are visualized in Figure 1 for a 64-point input, *i.e.,* $N = 64$. It can be seen that the most principal DCT-II basis, *i.e.,* T_0 shows a constant magnitude distribution, while the DST-VII and DCT-VIII characterize a gradually increasing and gradually decreasing magnitude distribution of the data samples, respectively. Across the plots, we can observe that non-overlapping distributions can be characterized by DST-VII and DCT-VIII under different phases and periods.

The characteristics of the transform basis functions intuitively reveal that the advantages by applying DST-VII/ DCT-VIII for intra prediction residuals along the intra

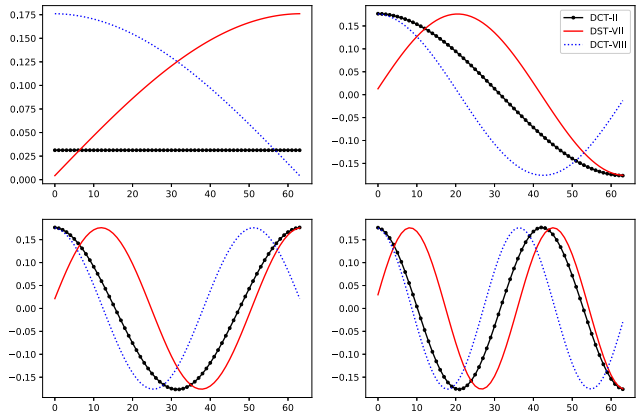


Fig. 1. The first four basis functions of DCT-II, DST-VII and DCT-VIII.

prediction direction since the residual magnitude generally increases along the intra prediction directions. As is observed in Figure 1, inter-prediction residual generally shows a larger residual magnitude for residues closer to PU boundary, thus joint utilization of DST-VII and DCT-VIII would be beneficial to better de-correlate the residual blocks.

III. Fast Multiple Transform Selection

In this section, the proposed fast method for DST-VII and DCT-VIII is presented in detail. First, the proposed fast transform scheme is described, including the (anti-)symmetric properties which are considered beneficial for reducing arithmetic operations. Second, an example is provided to showcase how these features are leveraged to devise the fast algorithm. Finally, the transform matrix tuning is presented to compensate for the deviation induced by the rounding error. Multiple measures have been taken to achieve the orthogonality and efficiency of the tuned transform matrices.

In the block-based hybrid video coding scheme, linear transforms are typically applied to the residual blocks obtained from inter- or intra-frame prediction. VVC supports up to 64×64 transform block and introduces the non-square transform block partition scheme. To achieve efficient implementation, existing HEVC/VVC reference software deploys a two-dimensional transform as a consecutive combination of two one-dimensional transforms with each for horizontal and vertical, respectively. The resulting coefficients are further processed by quantization and entropy coding. Typically, the forward and inverse transform matrices are transposed matrix of each other.

In the remaining sections, we present the proposed method by using DST-VII forward transform as an example unless otherwise specified. Unless otherwise stated, description to the algorithm is based on the 8-bit transform matrix representation. The proposed scheme is applicable to both 8-bit and 10-bit transform matrix representation. This paper only focuses on 16-, 32- and 64-point transform sizes for the following reasons: 1) There is already a similar fast algorithm implemented for 4-point transform. 2) No similar (anti-)symmetric features are observed to the best of our knowledge. 3) The benefits are quite limited even 4-point and 8-point fast implementation is achieved due to their relative small transform sizes. It should be noted that the inverse transforms are the transposed matrices hence the similar deployment can be realized. In addition, the same philosophy can be seamlessly migrated to DCT-VIII.

A. Fast Transform

In the following chapters, unless other stated, we denote the residual coefficients vector $\{x_0, \ldots, x_{15}\}$ as *input vector*, the transformed output vector $\{y_0, \cdots, y_{15}\}$ as *output vector*, the transform matrix derived from Table I as *transform matrix*, a vector from the transform matrix as *transform vector*, each element in the transform matrix as *element*. The first feature that is noteworthy to mention is that there exist only N distinct possible values except for 0 in an N-point transform matrix. Typically, the first basis vector contains all the values while others only contain partial values with or without sign changes and a possible 0. We denote the transform vector which contains N unique values (usually the first transform vector) as *basis transform vector*, and each element in the basis transform vector as a *member*.

For example, we use $\{a, b, \ldots, p\}$ to represent the *basis transform vector* of a 16-point DST-VII transform matrix that is derived from the equations defined in Table I. To make it simple, we use T to represent the whole transform matrix. It should be understood that the actual values might be different for a transform matrix of different sizes. For instance, a might be a different number in a 32-point transform matrix as it is in a 16-point transform matrix. However, it is assured that the same *member* represents the identical number within a transform matrix. Similar to the preprocessing applied on DCT-II in HEVC, the transform matrix elements in VVC are also scaled by a scale factor *e.g.,* $64 \cdot \sqrt{N}$ or $256 \cdot \sqrt{N}$, and then rounded to the closest integer. Further tuning by an offset might also be applied in the application.

Figure 2 sketches the framework of the proposed fast method on a 16-point DST-VII forward transform. The input vector $\{x_0, \cdots, x_{15}\}$ multiplies the transform matrix T to obtain the output vector $\{y_0, \cdots, y_{15}\}$. The filled red circle and green circle represent the subtraction and addition operation, respectively. In the conventional full matrix multiplication, the transformed results are calculated by repeating multiplying each input with the transform matrix element and adding them together. By leveraging the innate partial butterfly-type features, the proposed fast method accelerates this process through the simplification process and the intermediate results re-use mechanism. The proposed scheme supports both direct matrix multiplication and partial butterfly-type fast method. Parallel processing is supported when selecting direct matrix multiplication.

As shown with different colors in Figure 2, the DST-VII transform matrix consists of three features that are considered useful for a more efficient implementation. These features are summarized as follows. It should be noted that these features are non-overlapping, *i.e.,* only one feature is applicable for a given transform vector.

1) **Feature #1:** *N members* are included without considering the sign changes. These elements can be grouped into several groups with a fixed number of elements.

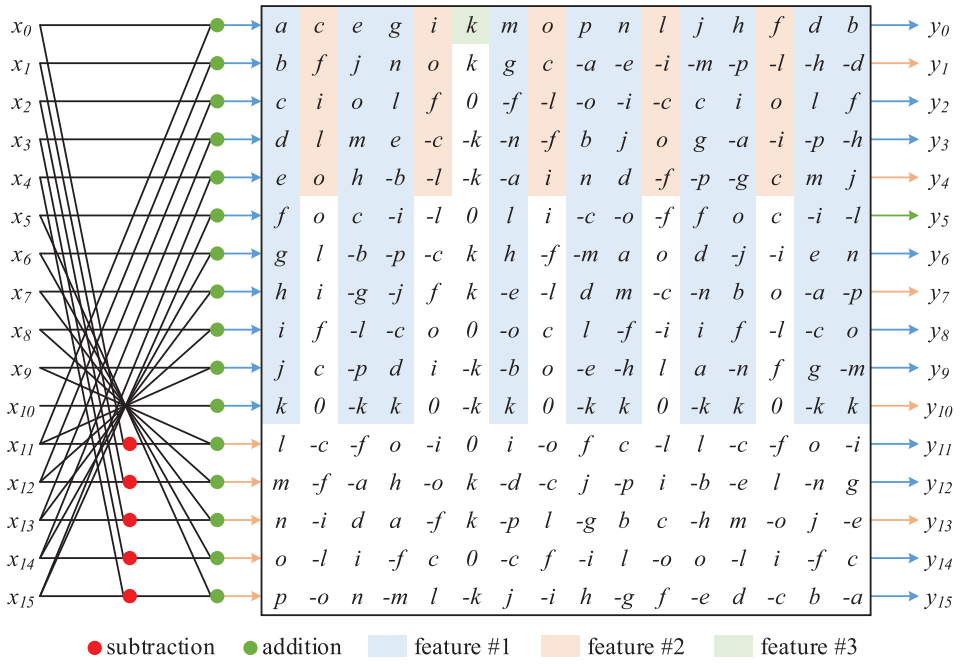


Fig. 2. Fast 16-point DST-VII forward transform algorithm. The input vector $\{x_0, \cdots, x_{15}\}$ contains the residual coefficients, the output vector $\{y_0, \cdots, y_{15}\}$ contains the transformed outputs and the middle part is the transform matrix denoted as T. Instead of using element-wise matrix multiplication, the fast algorithm leverages the (anti-)symmetric characteristics in the transform vectors to derive the identical results. The colored entries are involved in the fast method to derive the results whereas the entries with the white background are not used anymore. The benefits come from the reduced number of arithmetic operations. Since Feature #3 mainly involves additions and subtractions, we describe it in (7) without illustrating in this figure.

An equation exists by manipulating additions in each group.

2) **Feature #2:** Only a subset of the N *members* are included without considering the sign changes. They can be divided into several groups with a fixed number of consecutive elements such that every two consecutive groups are spatially symmetric or anti-symmetric, *i.e.,* symmetric by applying a negative sign.
3) **Feature #3:** Except for zero, some transform vector(s) only contain(s) a single *member* when neglecting the sign changes.

As mentioned in Feature #1, there exists an equation in each group. We observe that the relationships can be expressed using the following equations, *i.e.,* three elements form a group and five groups are derived in each two added elements equals to another element.

$$\begin{aligned} a + j &= l \\ b + i &= m \\ c + h &= n \\ d + g &= o \\ e + f &= p \end{aligned} \tag{1}$$

Take deriving y_0 using the first transform vector (colored in blue) as an example, conventional matrix multiplication would directly calculate the following equation which requires 16 multiplications and 15 additions.

$$\begin{aligned} y_0 = {} & a \cdot x_0 + b \cdot x_1 + c \cdot x_2 + d \cdot x_3 + e \cdot x_4 + f \cdot x_5 \\ & + g \cdot x_6 + h \cdot x_7 + i \cdot x_8 + j \cdot x_9 + k \cdot x_{10} \\ & + l \cdot x_{11} + m \cdot x_{12} + n \cdot x_{13} + o \cdot x_{14} + p \cdot x_{15} \end{aligned} \tag{2}$$

Benefit from Feature #1 (1), when calculating $(a \cdot x_0 + j \cdot x_9 + l \cdot x_{11})$ which requires three multiplications and two additions, we can calculate its equivalent form $a \cdot (x_0 + x_{11}) + j \cdot (x_9 + x_{11})$ by combing the common items thereby only two multiplications are needed. Although the addition operations are increased but some intermediate results can be re-used to eliminate the additional cost. We will introduce this in the following chapters. Similar simplification can be applied to $(b \cdot x_1 + i \cdot x_8 + m \cdot x_{12})$, $(c \cdot x_2 + h \cdot x_7 + n \cdot x_{13})$, $(d \cdot x_3 + g \cdot x_6 + o \cdot x_{14})$ and $(e \cdot x_4 + f \cdot x_5 + p \cdot x_{15})$. Therefore, to calculate y_0, instead of doing the element-wise multiplication which requires 16 multiplications and 15 additions, the following equivalent formulation can be utilized to derive the identical results.

$$\begin{aligned} y_0 = {} & a \cdot (x_0 + x_{11}) + b \cdot (x_1 + x_{12}) + c \cdot (x_2 + x_{13}) \\ & + d \cdot (x_3 + x_{14}) + e \cdot (x_4 + x_{15}) + f \cdot (x_5 + x_{15}) \\ & + g \cdot (x_6 + x_{14}) + h \cdot (x_7 + x_{13}) + i \cdot (x_8 + x_{12}) \\ & + j \cdot (x_9 + x_{11}) + k \cdot x_{10} \end{aligned} \tag{3}$$

which requires 11 multiplications and 20 additions. So we can save 5 multiplications but need 5 additional additions. Typically, it is faster performing addition instruction than multiplication instruction on modern CPUs. Thus time saving can be achieved by the simplification process. In addition, when calculating y_2, y_3, y_6, y_8, y_9, y_{11}, y_{12}, y_{14}, y_{15}, similar simplification can be achieved and what really matters is the intermediate results of $(x_0 + x_{11})$, $(x_1 + x_{12})$, $(x_2 + x_{13})$, $(x_3 + x_{14})$, $(x_4 + x_{15})$, $(x_5 + x_{15})$, $(x_6 + x_{14})$, $(x_7 + x_{13})$, $(x_8 + x_{12})$, $(x_9 + x_{11})$ and $k \cdot x_{10}$ can be re-used. This could save a lot more on top of that. In summary, the time complexity

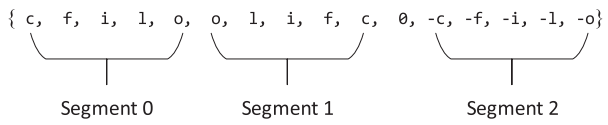


Fig. 3. Repeating patterns are observed in Feature #2, in which every five consecutive elements except zero constitute a group in a mirror-symmetric or mirror-antisymmetric manner.

reduction from Feature #1 comes from the simplification process and the intermediate results re-use mechanism.

We take the second transform vector (colored in orange) from T to showcase how Feature #2 is leveraged to achieve a better implementation.

As shown in Figure 3, the second transform vector can be classified into three segments. Each segment consists of the same five consecutive elements when neglecting the sign changes. They are replicate with sign changes, or flipped version of each other. Therefore, benefit from Feature #2, when computing the transformed result y_1, instead of doing the following element-wise operation

$$\begin{aligned} y_1 = {} & c \cdot x_0 + f \cdot x_1 + i \cdot x_2 + l \cdot x_3 + o \cdot x_4 \\ & + o \cdot x_5 + l \cdot x_6 + i \cdot x_7 + f \cdot x_8 + c \cdot x_9 \\ & - c \cdot x_{11} - f \cdot x_{12} - i \cdot x_{13} - l \cdot x_{14} - o \cdot x_{15} \end{aligned} \quad (4)$$

which requires 15 multiplications and 14 additions, the following simplified form can be utilized to derive the identical outcome with only 5 multiplications and 14 additions. The reduced number of multiplications can lead to time complexity reduction.

$$\begin{aligned} y_1 = {} & c \cdot (x_0 + x_9 - x_{11}) + f \cdot (x_1 + x_8 - x_{12}) \\ & + i \cdot (x_2 + x_7 - x_{13}) + l \cdot (x_3 + x_6 - x_{14}) \\ & + o \cdot (x_4 + x_5 - x_{15}) \end{aligned} \quad (5)$$

On top of that, when deriving y_1, y_4, y_7, y_{10}, y_{13}, the intermediate results $(x_0+x_9-x_{11})$, $(x_1+x_8-x_{12})$, $(x_2+x_7-x_{13})$, $(x_3 + x_6 - x_{14})$ and $(x_4 + x_5 - x_{15})$ can be re-used to further reduce the operation counts. Therefore, the benefits from Feature #2 come from both the simplification process and the intermediate results re-use mechanism.

We take the sixth transform vector (colored in green) to explain how Feature #3 is utilized to reduce the computational complexity. This transform vector has very distinct characteristics, *i.e.,* consists of only a single *member* k when neglecting the sign changes. To calculate y_5, the conventional element-wise matrix multiplication calculates in the following manner, which requires 11 multiplications and 10 additions.

$$\begin{aligned} y_5 = {} & k \cdot x_0 + k \cdot x_1 - k \cdot x_3 - k \cdot x_4 + k \cdot x_6 + k \cdot x_7 \\ & - k \cdot x_9 - k \cdot x_{10} + k \cdot x_{12} + k \cdot x_{13} - k \cdot x_{15} \end{aligned} \quad (6)$$

Benefit from Feature #3, we can alternatively derive the identical results through the simplified formulation (7) by combining the common terms which requires only 1 multiplication and 10 additions. Therefore, the time reduction achieved from Feature #3 comes from the simplification process.

$$\begin{aligned} y_5 = {} & k \cdot (x_0 - x_2 + x_3 - x_5 + x_6 - x_8 \\ & + x_9 - x_{11} + x_{12} - x_{14} + x_{15}) \end{aligned} \quad (7)$$

It should be noted that in a given transform matrix, the combination pattern is fixed to implement Feature #1, *i.e.,* the pattern of addition of two elements equals to another element only exists in the 16-point or 64-point transform matrices, the pattern of addition of three elements equals to addition of another two elements only exists in a 32-point transform matrix. We summarize the applicable transform vectors for the mentioned three features of 16, 32, and 64-point DST-VII transform matrices as follows.

- 16-point transform
  - Feature #1: T_{3n} and T_{2+3m}, where $n = 0, \cdots, 5$, $m = 2, 3, 4$
  - Feature #2: T_{1+3n}, where $n = 0, \cdots, 4$
  - Feature #3: T_5
- 32-point transform
  - Feature #1: T_5 and T_{8+m+5n}, where $m = 0, \cdots, 3$, $n = 0, \cdots, 4$ but $n \neq 2$ when $m = 1$
  - Feature #2: T_{2+5n}, where $n = 0, \cdots, 5$
  - Feature #3: T_6 and T_{19}
- 64-point transform
  - Feature #1: T_{2+3n} and T_{3m}, where $n = 0, \cdots, 20$, $m = 0, \cdots, 21$ but $m \neq 7$
  - Feature #2: T_{1+3n}, where $n = 0, \cdots, 20$
  - Feature #3: T_{21}

For the DST-VII inverse transform, the transform matrix is the transposed version of the forward transform matrix thereby similar deployment can be achieved. The above-mentioned features also exist in 32-point and 64-point transform matrices with a slight difference of how the groups are identified and the number of elements in each group. Therefore, a similar implementation can be applied to 32-point and 64-point transform matrices.

Since DST-VII and DCT-VIII share the same implementation logic, we omit full description to DCT-VIII to avoid redundancy. It would be easier to understand by examining the basis transform vector of DCT-VIII. The basis transform vector of the 16-point DST-VII is

$$\begin{aligned} T_0^{DST-VII} = {} & \{8, 17, 25, 33, 40, 48, 55, \\ & 62, 68, 73, 77, 81, 85, 87, 88, 88\} \end{aligned} \quad (8)$$

and the basis transform vector of the 16-point DCT-VIII is

$$\begin{aligned} T_0^{DCT-VIII} = {} & \{88, 88, 87, 85, 81, 77, 73, \\ & 68, 62, 55, 48, 40, 33, 25, 17, 8\} \end{aligned} \quad (9)$$

To leverage the proposed rules, the fast method Feature #1 for 16-point DCT-III can be formulated as

$$T_0(15 - j) + T_0(6 + j) = T_0(4 - j), \quad j = 0, \cdots, 4 \quad (10)$$

which corresponds to (16) of the 16-point DST-VII formulation.

TABLE II
COMPARISON OF THE TUNED TRANSFORM MATRICES AND THE ORIGINAL TRANSFORM MATRICES

Bit depth	Metric	16 point		32 point		64 point	
		Tuned	Original	Tuned	Original	Tuned	Original
8-bit	Orthogonality	$o_{ij} \leq 0.0017$	$o_{ij} \leq 0.0020$	$o_{ij} \leq 0.0026$	$o_{ij} \leq 0.0026$	$o_{ij} \leq 0.0018$	$o_{ij} \leq 0.0018$
	Closeness	$m_{ij} \leq 0.1910$	$m_{ij} \leq 0.1509$	$m_{ij} \leq 0.1956$	$m_{ij} \leq 0.1956$	$m_{ij} \leq 0.9037$	$m_{ij} \leq 0.7902$
	Norm measure	$n_i \leq 0.0047$	$n_i \leq 0.0066$	$n_i \leq 0.0039$	$n_i \leq 0.0045$	$n_i \leq 0.0039$	$n_i \leq 0.0045$
10-bit	Orthogonality	$o_{ij} \leq 0.0009$	$o_{ij} \leq 0.0010$	$o_{ij} \leq 0.0005$	$o_{ij} \leq 0.0007$	$o_{ij} \leq 0.0010$	$o_{ij} \leq 0.0010$
	Closeness	$m_{ij} \leq 0.0189$	$m_{ij} \leq 0.0156$	$m_{ij} \leq 0.0437$	$m_{ij} \leq 0.0382$	$m_{ij} \leq 0.0833$	$m_{ij} \leq 0.0751$
	Norm measure	$n_i \leq 0.0019$	$n_i \leq 0.0024$	$n_i \leq 0.0004$	$n_i \leq 0.0006$	$n_i \leq 0.0012$	$n_i \leq 0.0017$

B. Transform Matrix Tuning

Similar to HEVC, VVC implements the transform matrices in a finite-precision approximation in the reference software. Using integer operations is friendly to hardware implementation. In addition, it might also avoid potential mismatch caused by the platforms from different manufactures. One of the drawbacks is the finite-precision representation is not as accurate as the floating-point representation. This might lead to inferior coding efficiency. Another disadvantage is that the useful features as mentioned in Section III-A are not valid in the rounded transform matrices.

To better explain this point, we take the 32-point DST-VII forward transform matrix as an example. We use $\{a, b, \ldots, z, A, B, \ldots, F\}$ to denote the *basis transform vector* which is also the first transform vector. In floating-point 32-point DST-VII forward transform matrix, Feature #1 can be expressed with the following equations.

$$\begin{aligned} a + l + A &= n + y \\ b + k + B &= o + x \\ c + j + C &= p + w \\ d + i + D &= q + v \\ e + h + E &= r + u \\ f + g + F &= s + t \end{aligned} \tag{11}$$

In VVC test model, the transform matrix elements are multiplied and scaled to the closest integer resulting in the following actual values.

$$\begin{aligned} \{a, b, \ldots, F\} = \{&4, 9, 13, 17, 21, 26, 30, 34, 38, 42, \\ &46, 49, 53, 56, 60, 63, 66, 69, 71, 74, 76, \\ &78, 81, 82, 84, 85, 87, 88, 89, 89, 90, 90\} \end{aligned} \tag{12}$$

Therefore, for quintuple #1 in (11), the left side and the right side can be calculated as

$$\begin{aligned} b + k + B &= 9 + 46 + 88 = 143 \\ o + x &= 60 + 82 = 142 \end{aligned} \tag{13}$$

which are not equal any more. To bring this rounded transform matrix back to validity for Feature #1, the basis transform vector has to be tuned. In this example, we can either subtract 1 from b, k or B, or add 1 to o or x.

To make the adjusted transform matrices well-adapted in video coding scenario, we define the following principles that should be strictly followed during the tuning process. First, the N-point transform matrix can be represented using N distinct basis transform vector *members* without considering the sign changes. Second, the orthogonality between any two transform vectors should be optimized as much as possible. Finally, the adjusted basis transform vector *members* should be kept as close as possible to the floating-point basis transform vector *members*.

The following metrics are defined to evaluate the quality of the tuned transform matrices, including orthogonality, accuracy and norm measurement.

1) Orthogonality measure: $o_{ij} = \mathbf{d}_i^T \mathbf{d}_j / \mathbf{d}_0^T \mathbf{d}_0$, $i \neq j$
2) Closeness measure: $m_{ij} = |\alpha c_{ij} - d_{ij}| / d_{00}$
3) Norm measure: $n_i = |\mathbf{1} - \mathbf{d}_i^T \mathbf{d}_i / \mathbf{d}_0^T \mathbf{d}_0|$

Given the original N-point transform matrix element $c_{ij} = T_i(j)$ as defined in Table I, the scaled approximated transform matrix element of d_{ij}, which constitutes the scaled approximated transform vector $\mathbf{d}_i = [d_{i0}, \ldots, d_{i(N-1)}]^T$, where $i = 0, \ldots, N-1$, the global optimization objective is defined as follows.

$$D = |\alpha^2 \cdot I - T \cdot T^T| \tag{14}$$

where $i, j = 0, \ldots, N-1$, and the scale factor $\alpha = 64 \cdot \sqrt{N}$. The tuning process is deployed by trying all possible integer values and evaluate the tuned transform matrix using the above-mentioned three measurements and the optimal setting will be adopted.

In addition, the per-element magnitude difference between the tuned transform matrix T and the floating-point transform matrix T_0 is restrained to be no larger than 1. In such a way, the adjusted transform matrices are kept as close as possible to the original floating-point transform matrices to avoid severe performance deviation. The tuned transform matrices in both 8-bit and 10-bit can be found in [29], [32].

The comparison between the tuned transform matrices and the original transform matrices is tabulated in Table II. The worst value of o_{ij}, m_{ij} and n_i are used to measure the level of approximation. As can be seen from the table, in most cases, although the tuned transform matrices are farther from the orthonormal transform matrices, they provide better orthogonality and norm properties.

The sinusoidal graphs are also provided in Figure 4 to show the closeness of the original transform basis and the

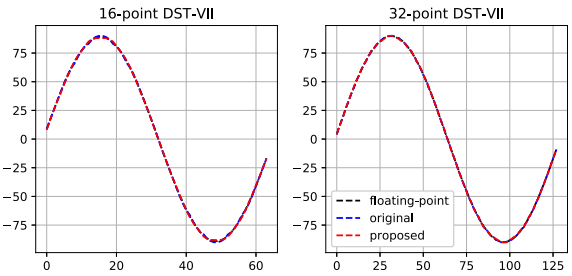


Fig. 4. Sinusoidal graphs of 16-point and 32-point DST-VII among the floating-point transform basis, the original transform basis and the proposed transform basis.

proposed transform basis to the floating-point transform basis. Here, the floating-point transform basis is directly derived from the formula listed in Table I and multiplied by a scale factor, the original transform basis refers to the existing one in the reference software prior to the proposed scheme was adopted, the proposed transform is the proposed transform basis presented in this paper. We can observe from the graph that, there are only some minor differences on the basis at the peak of wave and the bases look close among the three curves.

IV. Theoretical Derivation

To demonstrate the validity of the proposed features utilized to design the fast methods, we prove them from theory. Since Feature #2 and Feature #3 are very straightforward, we omit the theoretical proof process.

A. 16-Point Transform

In the 16-point DST-VII example, Feature #1 (1) can be expressed in the form of (15). Therefore, Feature #1 can be summarized in a more compact form as in (16).

$$\begin{aligned} T_0(0)+T_0(9) &= T_0(11)\\ T_0(1)+T_0(8) &= T_0(12)\\ T_0(2)+T_0(7) &= T_0(13)\\ T_0(3)+T_0(6) &= T_0(14)\\ T_0(4)+T_0(5) &= T_0(15) \end{aligned} \tag{15}$$

$$T_0(j)+T_0(9-j) = T_0(11+j), \quad j=0,\cdots,4 \tag{16}$$

According to the basis function defined in Table I, they can be re-written in the following format.

$$\begin{aligned} T_0(j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(j+1)}{2N+1}\\ T_0(9-j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(10-j)}{2N+1}\\ T_0(11+j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(12+j)}{2N+1} \end{aligned} \tag{17}$$

Therefore, proving (16) is equivalent to proving (18) by removing the non-zero element $\sqrt{\frac{4}{2N+1}}$.

$$\sin\frac{\pi(j+1)}{2N+1}+\sin\frac{\pi(10-j)}{2N+1}=\sin\frac{\pi(12+j)}{2N+1} \tag{18}$$

The proof process is provided as follows.

$$\begin{aligned} &\sin\frac{\pi(12+j)}{2N+1}-\sin\frac{\pi(10-j)}{2N+1}\\ &=2\cos\frac{11\pi}{2N+1}\sin\frac{\pi(1+j)}{2N+1}\\ &=2\cos\frac{\pi}{3}\sin\frac{\pi(1+j)}{2N+1}\\ &=\sin\frac{\pi(j+1)}{2N+1} \end{aligned} \tag{19}$$

Therefore, (16) has been proven.

B. 32-Point Transform

In the 32-point DST-VII transform matrix, the relationships defined in (11) can be expressed in the following form.

$$\begin{aligned} T_0(0)+T_0(11)+T_0(26) &= T_0(13)+T_0(24)\\ T_0(1)+T_0(10)+T_0(27) &= T_0(14)+T_0(23)\\ T_0(2)+T_0(8)+T_0(28) &= T_0(15)+T_0(22)\\ T_0(3)+T_0(8)+T_0(29) &= T_0(16)+T_0(21)\\ T_0(4)+T_0(7)+T_0(30) &= T_0(17)+T_0(20)\\ T_0(5)+T_0(6)+T_0(31) &= T_0(18)+T_0(19) \end{aligned} \tag{20}$$

Therefore, we can describe Feature #1 in 32-point transform in a compact manner as defined in (21).

$$\begin{aligned} &T_0(j)+T_0(11-j)+T_0(26+j)\\ &\quad = T_0(13+j)+T_0(24-j), \quad j=0,\cdots,5 \end{aligned} \tag{21}$$

Based on definitions in Table I, we have

$$\begin{aligned} T_0(j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(j+1)}{2N+1}\\ T_0(11-j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(12-j)}{2N+1}\\ T_0(26+j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(27+j)}{2N+1}\\ T_0(13+j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(14+j)}{2N+1}\\ T_0(24-j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(25-j)}{2N+1} \end{aligned} \tag{22}$$

After removing the non-zero element $\sqrt{\frac{4}{2N+1}}$, we can prove the equivalent objective as described in (23).

$$\begin{aligned} &\sin\frac{\pi(j+1)}{2N+1}+\sin\frac{\pi(12-j)}{2N+1}+\sin\frac{\pi(27+j)}{2N+1}\\ &\qquad =\sin\frac{\pi(14+j)}{2N+1}+\sin\frac{\pi(25-j)}{2N+1} \end{aligned} \tag{23}$$

On the left side of the equation, when we combine the second and the third term, we achieve the following expression:

$$\begin{aligned} &\sin\frac{\pi(j+1)}{2N+1}+\sin\frac{\pi(12-j)}{2N+1}+\sin\frac{\pi(27+j)}{2N+1}\\ &=\sin\frac{\pi(j+1)}{2N+1}+2\sin\frac{39\pi}{2(2N+1)}\cos\frac{\pi(-15-2j)}{2(2N+1)} \end{aligned} \tag{24}$$

TABLE III
FEATURE #1 APPLICABLE VECTORS IN 16-POINT DST-VII TRANSFORM MATRIX

y_2	y_3	y_6	y_8	y_9	y_{11}	y_{12}	y_{14}	y_{15}
$e-p=-f$	$g+d=o$	$m-b=i$	$p-e=f$	$n-h=c$	$j+a=l$	$h-n=-c$	$d+g=o$	$b-m=-i$
$j-l=-a$	$n-c=h$	$g-o=-d$	$-a+l=j$	$-e-f=-p$	$-m+i=-b$	$-p+f=-e$	$-h-c=-n$	$-d+o=g$
$o-g=d$	$l-j=a$	$-f-e=-p$	$-o+d=-g$	$-i+m=b$	$c-n=-h$	$i+b=m$	$l-a=j$	$f-p=-e$
$m-b=i$	$e-p=-f$	$-n+h=-c$	$b-m=-i$	$j+a=l$	$g+d=o$	$-a-j=-l$	$-p+e=-f$	$-h+n=c$
$h+c=n$	$-b-i=-m$	$-a+l=j$	$n-c=h$	$d-o=-g$	$-p+f=-e$	$-g+o=d$	$m-i=b$	$j-l=-a$

Similar processing can be performed on the right terms, then the right side is transformed to (25).

$$\sin\frac{\pi(14+j)}{2N+1}+\sin\frac{\pi(25-j)}{2N+1} = 2\sin\frac{\pi 39}{2(2N+1)}\cos\frac{\pi(-11+2j)}{2(2N+1)} \quad (25)$$

Equation (23) can be re-written as (26) by substituting corresponding items with (24) and (25).

$$\sin\frac{\pi(j+1)}{2N+1}+2\sin\frac{39\pi}{2(2N+1)}\cos\frac{\pi(-15-2j)}{2(2N+1)} = 2\sin\frac{39\pi}{2(2N+1)}\cos\frac{\pi(-11+2j)}{2(2N+1)} \quad (26)$$

After merging similar items, the objective becomes

$$\sin\frac{\pi(j+1)}{2N+1} = 2\sin\frac{39\pi}{2(2N+1)}\left[\cos\frac{\pi(-11+2j)}{2(2N+1)} - \cos\frac{\pi(-15-2j)}{2(2N+1)}\right] \quad (27)$$

The right-most 2 items can be further simplified as follows.

$$\cos\frac{\pi(-11+2j)}{2(2N+1)}-\cos\frac{\pi(-15-2j)}{2(2N+1)} = -2\sin\frac{-26\pi}{4(2N+1)}\sin\frac{\pi(4+4j)}{4(2N+1)} \quad (28)$$

Then we substitute corresponding items in (27), the objective turns to this form

$$\sin\frac{\pi(j+1)}{2N+1} = -4\sin\frac{39\pi}{2(2N+1)}\sin\frac{-26\pi}{4(2N+1)}\sin\frac{\pi(4+4j)}{4(2N+1)} \quad (29)$$

Since $\sin\frac{\pi(j+1)}{2N+1}$ is a non-zero element, we obtain the final simplified objective equation (30) which is equivalent to (21).

$$\sin\frac{39\pi}{2(2N+1)}\cdot\sin\frac{13\pi}{2(2N+1)}=\frac{1}{4} \quad (30)$$

This equation turns to be an identity relation since N is 32. Therefore, the Feature #1 of 32-point DST-VII transform matrix as described in (21) has been proved.

C. 64-Point Transform

Similar to 16-point and 32-point transform matrices, the 64-point Feature #1 also can be expressed in a compact manner.

$$T_0(j)+T_0(41-j)=T_0(43+j),\quad j=0,\cdots,20 \quad (31)$$

Based on the definitions in Table I, each item in (31) can be expanded to the following form.

$$\begin{aligned} T_0(j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(j+1)}{2N+1}\\ T_0(41-j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(42-j)}{2N+1}\\ T_0(43+j) &= \sqrt{\frac{4}{2N+1}}\cdot\sin\frac{\pi(44+j)}{2N+1} \end{aligned} \quad (32)$$

Replacing corresponding items in (31) with those in (32) and removing the non-zero item $\sqrt{\frac{4}{2N+1}}$, we obtain an equivalent objective equation.

$$\sin\frac{\pi(j+1)}{2N+1}+\sin\frac{\pi(42-j)}{2N+1}=\sin\frac{\pi(44+j)}{2N+1} \quad (33)$$

Start from the left side, each term can be expressed as the following form.

$$\sin\frac{\pi(j+1)}{2N+1}=\sin\left[\frac{\pi(44-j)}{2N+1}-\frac{43\pi}{2N+1}\right] \quad (34)$$

$$\sin\frac{\pi(42-j)}{2N+1}=\sin\left[\frac{\pi(-44-j)}{2N+1}+\frac{86\pi}{2N+1}\right] \quad (35)$$

The terms on the right side can be further expanded to achieve this description.

$$\sin\left[\frac{\pi(44-j)}{2N+1}-\frac{43\pi}{2N+1}\right] = \sin\frac{\pi(44+j)}{2N+1}\cos\frac{43\pi}{2N+1}-\cos\frac{\pi(44+j)}{2N+1}\sin\frac{43\pi}{2N+1} \quad (36)$$

$$\sin\left[\frac{\pi(-44-j)}{2N+1}+\frac{86\pi}{2N+1}\right] = \sin\frac{\pi(-44-j)}{2N+1}\cos\frac{86\pi}{2N+1}+\cos\frac{\pi(-44-j)}{2N+1}\sin\frac{86\pi}{2N+1} \quad (37)$$

When we add (36) and (37) and merge the common terms, we obtain the following equivalent objective.

$$\sin\frac{\pi(j+1)}{2N+1}+\sin\frac{\pi(42-j)}{2N+1} = \sin\frac{\pi(44+j)}{2N+1}\left[\cos\frac{43\pi}{2N+1}-\cos\frac{86\pi}{2N+1}\right] + \cos\frac{\pi(44+j)}{2N+1}\left[\sin\frac{86\pi}{2N+1}-\sin\frac{43\pi}{2N+1}\right] \quad (38)$$

TABLE IV
THE NUMBER OF ARITHMETIC OPERATIONS FOR A 1D FORWARD/INVERSE TRANSFORM

Transform Size	Matrix Multiplication			Fast DST-VII/DCT-VIII		
	Mult	Add	Shift	Mult	Add	Shift
16	256	240	16	127	155	16
32	1024	992	32	620	718	32
64	4096	4032	64	2207	2331	64

Since $N = 64$, (38) is equivalent to (39) as follows.

$$\begin{aligned}\sin\frac{\pi(j+1)}{2N+1} + \sin\frac{\pi(42-j)}{2N+1} &= \sin\frac{\pi(44+j)}{2N+1}\left[\cos\frac{\pi}{3} - \cos\frac{2\pi}{3}\right] \\ &\quad + \cos\frac{\pi(44+j)}{2N+1}\left[\sin\frac{2\pi}{3} - \sin\frac{\pi}{3}\right] \end{aligned} \tag{39}$$

It is obvious that

$$\cos\frac{\pi}{3} - \cos\frac{2\pi}{3} = 1 \tag{40}$$

$$\sin\frac{2\pi}{3} - \sin\frac{\pi}{3} = 0 \tag{41}$$

Finally, (39) can be expressed in the following format.

$$\sin\frac{\pi(j+1)}{2N+1} + \sin\frac{\pi(42-j)}{2N+1} = \sin\frac{\pi(44+j)}{2N+1} \tag{42}$$

Therefore, (31) has been proven.

In summary, these features utilized to design the fast DST-VII/DCT-VIII method are tenable in theory. These inherent properties are perfectly supported and considered useful for a more efficient implementation. It is the deviation caused by the rounding operation in the finite-precision expression that breaks the (anti-)symmetric properties.

V. COMPLEXITY ANALYSIS

We provide complexity analysis in this section, including both the number of arithmetic operation counts and the actual software execution time. In the software execution time section, two experiments are devised 1) using a separate test program to execute the transform operation by a large number of repetitions and calculate the average transform time; 2) collecting the actual transform time from the VTM.

A. Arithmetic Operations

In the element-wise matrix multiplication, N^2 multiplications and $N(N-1)$ additions are needed to derive a 1D inverse transformed results. Therefore, the doubled number of operations are needed for calculating a 2D transform results. We calculate the number of operation counts involved in a 1D transform process to conduct the comparison.

According to the three features as mentioned above, a reduced number of operation counts can be achieved. To check the practical effects, we tabulate the numbers of arithmetic operations required for a 1D N-point transform of the full-matrix multiplication and the fast method in Table IV. Overall, 41.8%, 33.1% and 43.8% total number of arithmetic operations are saved for 16-point, 32-point and 64-point DST-VII/DCT-VIII transform, respectively.

To showcase how these numbers of operation counts are obtained, we take the 16-point DST-VII inverse transform as an example to introduce the details. To derive a single output element, 16 multiplications and 15 additions are needed using element-wise matrix multiplication. Therefore, to obtain an output vector, 16 times operation counts are required, *i.e.,* 256 multiplications and 240 additions.

However, if using the proposed fast methods, Feature #1 can be applied to 10 transform vectors, Feature #2 can be applied to 5 transform vectors and Feature #3 can be applied to the remaining transform vector. In Feature #1, there are 16 shared values to be re-used which occupy 25 additions and another shared value which requires 1 multiplication. To use these shared values derive the transformed results for the 10 transform vectors, an additional of 10×10 multiplications and 10×10 additions are needed. Thus 101 multiplication operations and 125 addition operations are required by applying Feature #1. The number of operation counts can be calculated in a similar way for performing Feature #2, resulting in 25 multiplications and 20 additions. When applying Feature #3, there are 1 multiplication and 10 addition operations. By summing them up, the total numbers of multiplications and additions needed to derive a 1D 16-point DST-VII transformed vector are 127 and 155, respectively. Therefore, 50.4% and 35.4% operation counts reduction are achieved for multiplication and addition, respectively.

For the 1D 32-point and 64-point transforms, the numbers of operation counts can be calculated in a similar way. As tabulated in Table IV, 39.5% and 27.6% total number of multiplications and additions are reduced to derive a 1D 32-point transformed results. In a 1D 64-point transform case, 46.1% and 42.2% operation counts reduction are achieved for multiplication and addition, respectively.

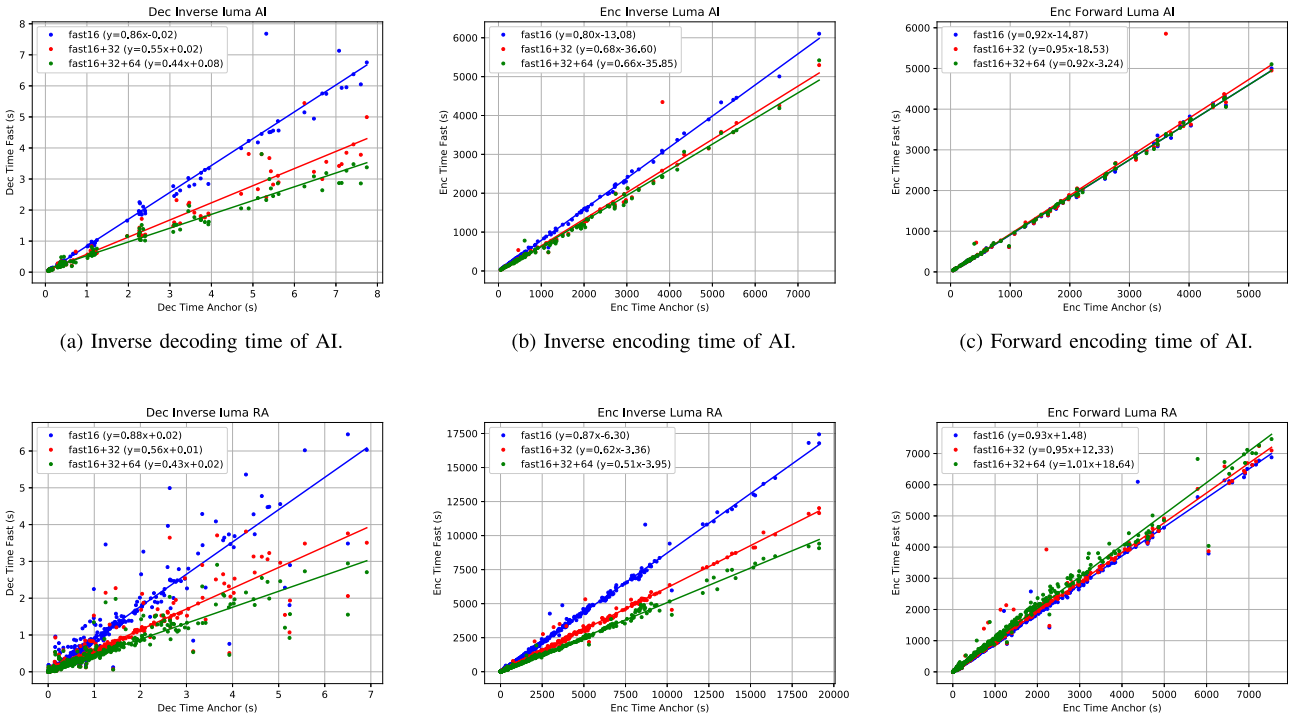
B. Software Execution Time

In this subsection, we devise two experiments to compare the software execution time between the proposed fast method and the direct matrix multiplication. Firstly, a standalone test program is used to measure the run-time (in seconds) of individual transform functions by repeating a large number of times. Secondly, the proposed fast method is integrated into VTM-3.0 and executed over all the CTC test sequences, the transform time is collected to provide a statistical summary showing the ratio of the run-time between the proposed fast transform and the anchor. It should be noted that the matrix multiplication implementation is the same no matter which VTM version is used.

The standalone program to measure the run-time of individual functions is written in C, and auto-vectorization is disabled (`#pragma loop(no_vector)`) for compiling to provide a fair comparison. The total number of repeating iterations are set empirically so that the total execution time is within a reasonable range. The detailed configurations of the test environment are available below.

TABLE V
COMPARISONS ON SOFTWARE EXECUTION SPEED (SECONDS)

Transform Size	No. of Iterations	Matrix Multiplication		Proposed Fast Method	
		Forward	Inverse	Forward	Inverse
16-point DCT-VIII	2^{23}	11.3	15.2	4.8	4.9
32-point DCT-VIII	2^{20}	11.1	14.8	5.7	5.5
64-point DCT-VIII	2^{17}	11.0	14.2	5.2	5.1



(a) Inverse decoding time of AI. (b) Inverse encoding time of AI. (c) Forward encoding time of AI.

(d) Inverse decoding time of RA. (e) Inverse encoding of RA. (f) Forward encoding of RA.

Fig. 5. Encoding and decoding software execution time of the Luma component under AI and RA configurations. The straight lines are linear regression approximation with L2 norm constraints.

– CPU: i7-6600U CPU @ 2.6 GHz
– Windows 10 Pro, 64-bit
– Memory (RAM): 16 GB
– Compiler: Visual Studio 2017

The results of All Intra (AI) and Random Access (RA) are tabulated in Table V. By repeating the 16-point DCT-VIII transform function by 2^{23} times, the proposed fast method can save 57.52% and 67.76% software execution time for forward and inverse transform, respectively. In the 32-point DCT-VIII transform, an average of 48.65% and 62.84% time savings are achieved for forward and inverse transform, respectively. Similarly, an average time saving of 52.73% and 64.08% is observed for forward and inverse transform, respectively.

In the second experiment, the 16-point, 32-point, and 64-point DST-VII/DCT-VIII fast methods are integrated into VTM-3.0 reference software. All the test sequences from VVC CTC are utilized for the testing. The software execution time of both forward and inverse transform are collected for the Luma component. Encoding process involves both forward and inverse transforms since the RDO process, while the decoding process only involves the inverse transform. To validate the individual contribution of each block-level, we enable the fast method from the smallest block size and increase gradually to the largest block size.

The statistical results are illustrated in Figure 5. *fast16* denotes enabling fast methods of block size 16, *fast16+32* represents enabling fast methods of both block size 16 and 32, and *fast16+32+64* means enabling fast methods of all block levels. The vertical axis is the encoding/decoding time of the proposed fast methods and the horizontal axis is the encoding/decoding time of the anchor. The solid lines are the linear regression approximations with the mean squared loss. The average percentage of time reduction can be approximately represented by $(1-l)\times 100\%$ where l is the *slope* of the solid line.

Overall, the superiority is quite remarkable, especially during inverse transform process. It is observed in Figure 5 that an average of 56%, 57% inverse decoding time savings are achieved under AI and RA configuration, respectively. In inverse transform of the encoding process, an average

TABLE VI
ADDITIONAL METRICS RELATED TO THE PROPOSED FAST METHOD

Metrics	Reported results
Additional memory requirements for storing transform matrices	No
Minimum bit-precision	Unchanged
Specify if a transform requires multiple iterations where transform output is fed back as input to the transform logic, and multiple iterations are required to produce final transform output.	No
Other operations and memory requirements relevant to the proposed tool.	No
List of all combinations of transforms block size and transform type used for secondary transformation.	Same with VTM-3.0
Provide analysis of implementation and arithmetic commonalities of proposed transforms.	Proposed method supports both partial butterfly and matrix multiplications.
If the proposal requires additional computations at the encoder or decoder.	No

of 34% and 49% time savings have been achieved for AI and RA configuration, respectively. In addition, superiority is increased when involving the 32-point and 64-point fast method implementation. Take the decoding time of inverse transform under AI configuration *i.e.,* Figure 5a as an example, 14% average time saving is achieved when only the 16-point fast implementation is enabled. 31% more time saving is achieved when the 32-point fast method is enabled on top of that. An additional 11% time saving is observed when the 64-point fast transform is enabled. Therefore, each size of fast transform has its own contributions to the final performance. The decreased superiority during the forward transform process is caused by the fact that the encoder needs to traverse all possible transforms to perform rate-distortion optimization thereby dilutes the superiority of the fast methods.

C. Other Metrics

The additional metrics that might help better understanding the proposed method are tabulated in Table VI. The proposed method shares the following merits, no additional memory requirements for storing transform matrices, no additional computations required at the encoder/decoder side, the combinations of transform block size and transform type used for secondary transform are consistent with VTM reference software, the minimum bit-precision is unchanged, etc.

VI. EXPERIMENTS

A. Experiment Settings

The proposed fast methods are integrated to VVC Test Model VTM-3.0 [33]. Three sets of experiments are conducted, including the CTC Set, the Low QP Set and comparing with relevant methods. The CTC Set uses the common test condition [34] as defined by Joint Video Experts Team (JVET) for evaluating proposals during the VVC standards development. The quantization parameters (QP) are set to 22, 27, 32, 37. The Low QP set uses low QP values of 2, 7, 12, 17 to test the high-quality compression performance. In each set, we provide the results of both inter MTS is disabled and enabled. By default, the intra MTS is enabled in VVC common test condition. We also compare with similar schemes [35]–[37] proposed in MPEG meeting and analyze relative merits.

In those experiments, a set of 26 video sequences ranging from 416×240 to 3840×2160 are tested, including four artificial sequences with the computer screen and mixed natural and screen content. It should be noted that Class D and Class F (screen content) are excluded in the overall average performance. The Bjøntegaard delta bitrates (BD-Rate) [38], [39] is used to evaluate the relative coding improvement. The run-time ratio of the proposed method to anchor as defined in (43) is used to evaluate the time complexity, with 100% represents no run-time saving. All Intra (AI), Random Access (RA) and Low Delay B (LDB) configurations are covered. Under AI configuration, every picture is coded as Intra while under RA, intra period is set as one second, GOP size is set to 8. The codec is operating in 10-bit mode, RDOQ is enabled. The decoding time and encoding time are measured in seconds.

$$\Delta T = \frac{T_{Proposed}}{T_{Anchor}} \times 100\% \tag{43}$$

B. Common Test Condition

The run-time results under common test condition are shown in Table VII. An average of 9%, 0%, and −1% decoding time savings are achieved for AI, RA and LDB configurations, respectively and 4%, 0%, and 1% overall encoding time savings are achieved for AI, RA, and LDB configurations, respectively. Among AI, RA and LDB configurations, the proposed fast methods achieve the most significant average decoding time reduction under AI configuration with up to 9% decoding time saving. In addition, the proposed fast methods behave consistently comparing with the anchor across all the resolutions. In high-resolution contents, the proposed fast method is slightly better than in low-resolution contents which could probably be caused by more sizes of fast transform matrices are involved. In the encoding process, the most time saving is with Sequence *Tango2* and *FoodMarket4* of 7%; *CampfireParty2*, *RitualDance* and *RaceHorses* of 2%; and *PartyScene* of 6% for AI, RA and LDB configuration, respectively. In the decoding process, the most time saving is with Sequence *Tango2* and *FoodMarket4* of 14%; *RitualDance* of 4%; and *PartyScene* of 7% for AI, RA and LDB, respectively.

An increased superiority is observed when inter MTS is enabled for RA and LDB which is also as expected. Overall, 0% and 3% decoding time savings are achieved for RA and LDB, respectively; an average of 2% and 3% encoding time savings are achieved for RA and LDB, respectively. The overall decoding time saving increases from −1% to 3% for LDB, and the overall encoding time saving increases from 0% to 2% for RA, from 1% to 3% for LDB. This demonstrates the fast method is of significant benefits for inter coding process,

TABLE VII
RUN-TIME PERFORMANCE OF THE PROPOSED FAST METHOD COMPARED WITH VTM-3.0 UNDER COMMON TEST CONDITION

Class	Sequence	All Intra		Random Access				Low Delay B			
				InterMTS off		InterMTS on		InterMTS off		InterMTS on	
		ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}
Class A1 4k	Tango2	93%	86%	102%	101%	98%	96%	-	-	-	-
	FoodMarket4	93%	86%	99%	99%	97%	101%	-	-	-	-
	CampfireParty2	97%	92%	98%	96%	98%	99%	-	-	-	-
Class A2 4k	CatRobot1	95%	91%	101%	101%	99%	101%	-	-	-	-
	DaylightRoad2	98%	96%	101%	101%	99%	99%	-	-	-	-
	ParkRunning3	96%	89%	101%	100%	96%	98%	-	-	-	-
Class B 1080p	MarketPlace	98%	88%	99%	97%	97%	98%	100%	101%	96%	83%
	RitualDance	94%	88%	98%	96%	95%	96%	99%	102%	97%	96%
	Cactus	97%	91%	100%	99%	98%	99%	100%	105%	97%	96%
	BasketballDrive	96%	91%	99%	98%	98%	96%	99%	104%	98%	92%
	BQTerrace	96%	95%	99%	99%	99%	101%	100%	104%	98%	92%
Class C WVGA	BasketballDrill	99%	96%	101%	102%	101%	104%	99%	100%	98%	95%
	BQMall	100%	95%	100%	101%	101%	102%	100%	101%	98%	80%
	PartyScene	100%	99%	101%	102%	98%	100%	94%	93%	98%	98%
	RaceHorses	96%	92%	100%	101%	99%	97%	99%	98%	97%	94%
Class D WQVGA	BasketballPass	97%	90%	99%	99%	99%	102%	104%	103%	97%	99%
	BQSquare	98%	99%	100%	101%	99%	101%	99%	97%	98%	98%
	BlowingBubbles	98%	101%	99%	98%	98%	100%	99%	98%	98%	99%
	RaceHorses	97%	96%	98%	99%	98%	107%	100%	103%	97%	99%
Class E 720p	FourPeople	96%	92%	-	-	-	-	100%	100%	85%	94%
	Johnny	100%	95%	-	-	-	-	99%	100%	97%	93%
	KristenAndSara	96%	92%	-	-	-	-	99%	100%	97%	94%
Class F	BaskettallDrillText	98%	95%	99%	101%	98%	99%	99%	99%	116%	113%
	AOV5	96%	94%	101%	100%	96%	98%	100%	103%	99%	98%
	SlideEditing	97%	99%	99%	101%	99%	107%	101%	103%	100%	101%
	SlideShow	96%	96%	100%	101%	97%	101%	105%	100%	97%	101%
Average		**96%**	**91%**	**100%**	**100%**	**98%**	**100%**	**99%**	**101%**	**97%**	**97%**

TABLE VIII
BD-RATE PERFORMANCE OF PROPOSED FAST METHOD (TUNED TRANSFORM MATRIX) COMPARED WITH VTM-3.0 UNDER COMMON TEST CONDITION WITH INTER MTS DISABLED

Class	All Intra			Random Access			Low Delay B		
	Y	U	V	Y	U	V	Y	U	V
Class A1	0.02%	0.00%	0.06%	0.00%	-0.12%	0.05%	-	-	-
Class A2	0.01%	0.01%	0.03%	-0.04%	0.05%	0.01%	-	-	-
Class B	0.00%	-0.01%	-0.01%	-0.02%	0.04%	-0.08%	0.00%	-0.26%	-0.19%
Class C	0.01%	-0.01%	-0.15%	0.01%	-0.09%	-0.01%	-0.01%	0.09%	0.12%
Class D	0.01%	0.15%	-0.05%	-0.03%	-0.26%	-0.17%	0.01%	0.25%	0.50%
Class E	0.02%	-0.03%	0.17%	-	-	-	0.02%	1.16%	-0.73%
Class F	0.00%	-0.06%	0.12%	-0.04%	-0.01%	-0.06%	0.05%	-0.04%	0.07%
Average	**0.01%**	**-0.01%**	**0.01%**	**-0.01%**	**-0.03%**	**-0.02%**	**0.00%**	**0.22%**	**-0.22%**

especially when inter MTS is enabled when more transform types are involved in the RDO process.

Although some sequences happen to occupy more time than the anchor, *e.g.,* decoding time of *BlowingBubbles* under AI, encoding time of *BasketballDrill* under RA with inter MTS off, the overall time saving is quite encouraging. This phenomenon is probably caused by the CPU perturbation while executing since the decoding time is too short on these sequences.

To validate the coding efficiency by using the tuned transform matrices, we collect the BD-Rate results as shown in Table VIII. Overall, the proposed fast method achieves 0.01%, −0.01% and 0.01% Luma component BD-Rate reduction under AI, RA, and LDB respectively compared with

TABLE IX
RUN-TIME PERFORMANCE OF PROPOSED FAST METHOD COMPARED WITH VTM-3.0 ANCHOR UNDER LOW QP TEST CONDITION

Class	Sequence	All Intra		Random Access				Low Delay B			
				InterMTS off		InterMTS on		InterMTS off		InterMTS on	
		ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}
Class A1 4k	Tango2	100%	98%	98%	99%	98%	99%	-	-	-	-
	FoodMarket4	98%	100%	100%	100%	97%	97%	-	-	-	-
	CampfireParty2	101%	101%	99%	99%	108%	103%	-	-	-	-
Class A2 4k	CatRobot1	99%	101%	98%	99%	97%	98%	-	-	-	-
	DaylightRoad2	85%	88%	98%	97%	96%	96%	-	-	-	-
	ParkRunning3	96%	95%	98%	96%	95%	94%	-	-	-	-
Class B 1080p	MarketPlace	98%	98%	100%	101%	98%	98%	100%	96%	98%	94%
	RitualDance	98%	100%	99%	101%	98%	96%	101%	98%	97%	96%
	Cactus	98%	99%	99%	102%	99%	100%	100%	101%	96%	96%
	BasketballDrive	95%	101%	101%	101%	100%	100%	101%	102%	98%	96%
	BQTerrace	101%	108%	100%	101%	99%	99%	100%	99%	100%	99%
Class C WVGA	BasketballDrill	97%	106%	100%	101%	97%	96%	100%	100%	98%	102%
	BQMall	99%	103%	101%	100%	97%	100%	119%	114%	98%	99%
	PartyScene	99%	100%	99%	100%	98%	101%	101%	98%	97%	101%
	RaceHorses	100%	101%	99%	99%	98%	95%	100%	99%	97%	96%
Class D WQVGA	BasketballPass	100%	100%	100%	102%	98%	98%	99%	102%	99%	99%
	BQSquare	100%	113%	98%	100%	98%	106%	101%	104%	98%	102%
	BlowingBubbles	101%	102%	101%	103%	100%	103%	102%	106%	100%	102%
	RaceHorses	101%	100%	102%	103%	105%	108%	101%	110%	98%	99%
Class E 720p	FourPeople	99%	102%	-	-	-	-	98%	101%	99%	99%
	Johnny	97%	99%	-	-	-	-	99%	100%	98%	98%
	KristenAndSara	97%	98%	-	-	-	-	103%	99%	98%	98%
Class F	BaskettallDrillText	100%	102%	99%	100%	99%	100%	99%	98%	97%	94%
	AOV5	100%	98%	100%	100%	99%	105%	101%	103%	99%	103%
	SlideEditing	99%	101%	106%	107%	99%	101%	101%	106%	100%	103%
	SlideShow	97%	103%	102%	99%	100%	102%	100%	100%	101%	104%
Average		**98%**	**99%**	**99%**	**99%**	**98%**	**98%**	**100%**	**100%**	**98%**	**96%**

the anchor. The Luma component achieves very similar BD-Rate performance in most cases and only trivial difference has been observed compared with anchor. There exist some differences, but they are tolerable by considering the time saving it can bring. In summary, no noticeable side effects have been introduced by replacing with the tuned transform matrices. The BD-Rate results when inter MTS is enabled are similar to Table VIII thus are omitted here.

C. Low QP Test Condition

The run-time results using low QP values are tabulated in Table IX. In contrast to common test condition, the proposed fast method achieves decreased superiority for AI and increased superiority for RA and LDB. Under the AI test condition, the decoding time saving decreases from 9% to 1% and the encoding time saving decreases from 4% to 2%. In contrast, both the decoding and encoding saving increases from 0% to 1% under RA test condition. Under the LDB configuration, the decoding time saving decreases from 1% to 0%, and the encoding time increases from -1% to 0%. The changes under the AI test condition are caused by the fact that more smaller block transforms involved in low QP encoding process thus the time saving has been diluted. The essence of the proposed fast method is to accelerate the transform process by reducing the number of operation counts. Therefore, more time saving can be achieved in larger block transform sizes. The changes under RA and LDB test conditions are mostly caused by the fluctuations in terms of both the transform size and the transform counts which heavily depends on the coding parameter settings.

When inter MTS is enabled, an improved time saving performance has also been observed in RA and LDB test conditions. The encoding time saving increases by 1% and 2% for RA and LDB, respectively. The decoding time increases by 1% and 4% for RA and LDB, respectively. This phenomenon is consistent with that of using normal QP values which reveals that more transform block sizes are involved thereby more time saving is obtained.

To check the effects on the coding efficiency by introducing the new transform matrices under low QP test conditions, we collect the BD-Rate results in Table X. An average of 0.02%, 0.01%, and 0.00% changes are observed for AI, RA, and LDB, respectively. In the Luma component, a large portion of classes are not affected by introducing the tuned transform matrices, *i.e.,* identical coding efficiency has been achieved.

TABLE X
BD-RATE PERFORMANCE OF PROPOSED FAST METHOD (TUNED TRANSFORM MATRIX) COMPARED WITH VTM-3.0 ANCHOR UNDER LOW QP TEST CONDITION WITH INTER MTS DISABLED

Class	All Intra			Random Access			Low Delay B		
	Y	U	V	Y	U	V	Y	U	V
Class A1	0.00%	-0.02%	0.01%	0.00%	0.03%	0.01%	-	-	-
Class A2	0.07%	-0.02%	-0.01%	0.02%	-0.01%	-0.01%	-	-	-
Class B	0.00%	0.01%	0.00%	0.01%	0.00%	0.01%	0.00%	-0.01%	0.02%
Class C	0.01%	0.00%	0.01%	0.00%	0.00%	0.02%	0.00%	-0.01%	-0.02%
Class D	0.01%	-0.05%	-0.01%	0.00%	0.03%	0.00%	0.00%	0.01%	-0.02%
Class E	0.00%	0.00%	0.00%	-	-	-	0.00%	-0.01%	-0.01%
Class F	0.00%	0.01%	-0.02%	-0.11%	-0.11%	-0.05%	-0.06%	0.02%	-0.04%
Average	**0.02%**	**0.00%**	**0.00%**	**0.01%**	**0.00%**	**0.01%**	**0.00%**	**-0.01%**	**0.00%**

TABLE XI
BD-RATE AND ENCODING/DECODING TIME COMPARISON WITH RELATED METHODS

Methods	Configuration	Over VTM-3.0			Over VTM-3.0, InterMTS on		
		Y BD-Rate	ΔT_{Enc}	ΔT_{Dec}	Y BD-Rate	ΔT_{Enc}	ΔT_{Dec}
JVET-M0288 [35]	All Intra	0.00%	96%	90%	-	-	-
	Random Access	-0.01%	99%	98%	-0.01%	97%	98%
	Low Delay B	0.02%	100%	99%	0.01%	97%	96%
JVET-M0538 [36]	All Intra	0.00%	98%	94%	-	-	-
	Random Access	-0.37%	119%	100%	0.00%	98%	98%
	Low Delay B	-0.49%	125%	100%	0.05%	97%	96%
JVET-M0080 [37]	All Intra	0.09%	96%	85%	-	-	-
	Random Access	0.01%	101%	98%	-0.04%	103%	96%
	Low Delay B	0.07%	101%	100%	-0.09%	106%	102%
Proposed	All Intra	0.01%	96%	91%	-	-	-
	Random Access	-0.01%	100%	100%	0.00%	98%	100%
	Low Delay B	0.00%	99%	101%	0.00%	97%	97%

Marginal fluctuations are observed in the other classes which are tolerable since the overall changes are very trivial that can be neglected.

D. Comparison With Relevant Methods

We compare the proposed methods with relevant schemes that were discussed in the core experiments on complexity reduction of MTS. The DFT-based scheme JVET-M0288 [35], transform adjustment-based method JVET-M0538 [36] and Transform Adjustment Filters (TAF)-based solution JVET-M0080 [37] are included in this comparison.

In JVET-M0288 [35], a DFT-based scheme is proposed by replacing DST-VII and DCT-VIII with corresponding DFT transforms. The existent symmetries can be utilized to devise efficient fast transform implementation. In JVET-M0538 [36], the authors perform a transform "adjustment" in which the DST-VII and DCT-VIII transform matrices are processed vector by vector. The transform vector is decomposed to a combination of an "adjusted" part and a DCT-II coefficients part. The "adjusted" part is achieved by multiplying the 8 lowest frequency coefficients with a pre-defined 8×8 matrix. The remaining part is copied from the DCT-II transform matrix coefficients. The primary benefits come from the regular patterns in DCT-II transforms which enable faster parallel computation, and the "zero-out" technique used in DCT-II of dimension 64 which reduces the worst-case number of multiplications. In JVET-M0080 [37], another adjustment-based method is proposed called TAF, in which the transform matrices are approximated using a low complexity adjustment stage. TAF is implemented as a sparse matrix, used as a preprocessor towards the partial butterfly DCT-II algorithm.

The results are shown in Table XI, including Luma component BD-Rate reduction, encoding and decoding time ratio over VTM-3.0 with and without inter MTS enabled. It can be observed that though JVET-M0288 achieves better performance in terms of decoding time savings, it requires a two-stage implementation which is not friendly in parallel computation required scenarios. Another set of DFT transform sets need to be stored in the reference software which requires additional memory space. The JVET-M0538 achieves better decoding time savings when inter MTS is on compared with the proposed method, but leads to larger coding degradation. When tested over VTM-3.0 with inter MTS off, it achieves inferior decoding time saving with noticeable encoding time increase. Over VTM-3.0 with inter MTS off, JVET-M0080 performs slightly better in terms of decoding time saving, but with much significant coding performance

degradation by comparing with the proposed method. Over VTM-3.0 with inter MTS on, the proposed method performs better than JVET-M0080 in terms of both software run-time savings and coding performance. In addition, none of the counterparts supports dual implementation, *i.e.,* direct matrix multiplication and fast transform deployment.

In summary, the compared methods fail to support the following features simultaneously.

- Noticeable software run-time saving.
- Negligible coding performance degradation.
- Parallel computation supported.
- Dual-implementation supported.

The proposed scheme achieves a superior overall performance by considering the run-time saving, side effects on coding performance and dual-implementation with capability of parallel computation supported.

VII. Conclusion

This paper presents a fast DST-VII/DCT-VIII method with dual-implementation support for Multiple Transform Selection (MTS) which is adopted by the under-development next-generation video coding standard VVC. The inherent partial butterfly-type features are exploited to reduce the number of arithmetic operations. A fast DST-VII/DCT-VIII method is devised by using these features which achieves an approximate of 50% arithmetic operations with dual-implementation support, *i.e.,* either full matrix multiplication or fast partial butterfly-type implementation. Complexity analysis is performed to validate the efficiency in terms of theoretical operation counts and software run-time. In addition, the theoretical proof is provided to demonstrate that these beneficial features are tenable in theory. Comprehensive experiments are conducted by comparing with VTM-3.0 as well as relevant methods. The proposed method shares superior merits in varied evaluation conditions. The proposed scheme was adopted in the 13th JVET Meeting of ITU-T VCEG and ISO/IEC MPEG in January 2019.

References

[1] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," 2016, *arXiv:1611.01704*. [Online]. Available: http://arxiv.org/abs/1611.01704

[2] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–23.

[3] N. Yan, D. Liu, H. Li, T. Xu, F. Wu, and B. Li, "Convolutional neural network-based invertible half-pixel interpolation filter for video coding," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 201–205.

[4] N. Yan, D. Liu, H. Li, B. Li, L. Li, and F. Wu, "Convolutional neural network-based fractional-pixel motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 840–853, Mar. 2019.

[5] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An End-To-End deep video compression framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, p. 11.

[6] N. Yan, D. Liu, H. Li, and F. Wu, "A convolutional neural network approach for half-pel interpolation in video coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[7] Z. Zhang, X. Zhao, X. Li, Z. Li, and S. Liu, "Fast adaptive multiple transform for versatile video coding," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 63–72.

[8] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[9] 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.

[10] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vols. C–23, no. 1, pp. 90–93, Jan. 1974, doi: 10.1109/T-C.1974.223784.

[11] X. Zhao, J. Chen, M. Karczewicz, A. Said, and V. Seregin, "Joint separable and non-separable transforms for next-generation video coding," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2514–2525, May 2018.

[12] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W.-J. Chien, "Enhanced multiple transform for video coding," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 73–82.

[13] K. Karhunen, *Über Lineare Methoden in der Wahrscheinlichkeitsrechnung* (Suomalaisen Tiedeakatemian toimituksia). Helsinki, Finland: Suomalainen Tiedeakatemia, 1947, p. 79.

[14] M. Loeve, *Probability Theory II*, vol. 46. New York, NY, USA: Springer-Verlag, 1978.

[15] R. Clarke, "Relation between the Karhunen Loève and cosine transforms," *IEE Proc. F. Commun., Radar Signal Process.*, vol. 128, no. 1, pp. 359–360, Nov. 1981. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-f-1.1981.0061

[16] X. Zhao, L. Li, Z. Li, X. Li, and S. Liu, "Coupled primary and secondary transform for next generation video coding," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2018, pp. 1–4.

[17] R. H. Bamberger and M. J. T. Smith, "A filter bank for the directional decomposition of images: Theory and design," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 882–893, Apr. 1992.

[18] E. J. Candès and D. L. Donoho, "Ridgelets: A key to higher-dimensional intermittency?" *Phil. Trans. Roy. Soc. London. A, Math., Phys. Eng. Sci.*, vol. 357, no. 1760, pp. 2495–2509, Sep. 1999.

[19] J.-L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 670–684, Jun. 2002.

[20] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005, doi: 10.1109/TIP.2005.859376.

[21] B. Zeng and J. Fu, "Directional discrete cosine transforms—A new framework for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 305–313, Mar. 2008.

[22] X. Zhao, L. Zhang, S. Ma, and W. Gao, "Video coding with rate-distortion optimized transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 1, pp. 138–151, Jan. 2012.

[23] X. Cao and Y. He, "Singular vector decomposition based adaptive transform for motion compensation residuals," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 4127–4131.

[24] Y. Ye and M. Karczewicz, "Improved h.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," in *Proc. 15th IEEE Int. Conf. Image Process.*, 2008, pp. 2116–2119.

[25] X. Zhao, L. Zhang, S. Ma, and W. Gao, "Rate-distortion optimized transform for intra-frame coding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Dallas, TX, USA, Mar. 2010, pp. 1414–1417, doi: 10.1109/ICASSP.2010.5495468.

[26] B. Bross, J. Chen, and S. Liu, *Versatile Video Coding (Draft 2)*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11 and JVET–K1001, Joint Video Experts Team (JVET), Ljubljana, SI, USA, p. 10–18, Jul. 2018.

[27] J. Chen, Y. Ye, and S. Kim, *Algorithm description for versatile video coding and test model 2 (vtm 2)*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11 and vols. JVET–K1002, Joint Video Experts Team (JVET), Ljubljana, SI, USA, p. 10–18, Jul. 2018.

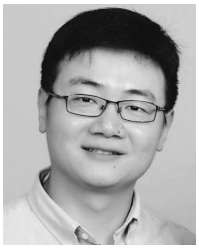
[28] H. Gao *et al.*, *Non-Ce6: Combined Test of Jvet-n0172/jvet-n0375/jvet-n0419/jvet-n0420 on Unification of Implicit Transform Selection*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11 and JVET–N0866, Joint Video Experts Team (JVET), Geneva, Switzerland, p. 10–18, Mar. 2019.

[29] X. Zhao, X. Li, Y. Luo, and S. Liu, *Ce6: Fast dst-7/dct-8 With Dual Implementation Support (Test 6.2.3)*, docuement ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 13 and JVET–M0497, Joint Video Experts Team (JVET), Marrakech, MA, USA, p. 10–18, Jan. 2019.

[30] A. K. Jain, "A sinusoidal family of unitary transforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vols. PAMI–1, no. 4, pp. 356–365, Oct. 1979.

[31] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Trans. Signal Process.*, vol. 42, no. 5, pp. 1038–1051, May 1994.
[32] Z. Zhang, X. Zhao, X. Li, and S. Liu, *Ce6-Related: Fast dst-7/dct-8 With Dual Implementation Support*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11 and VET–K0291, Joint Video Experts Team (JVET), Ljubljana, SI, USA, pp. 10–18, Jul. 2018.
[33] J. VVC. (May 2019). *Versatile Video Coding (VVC) Reference Software: VVC Test Model (VTM)*. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-3.0
[34] J. Boyce, K. Suehring, X. Li, and V. Seregin, *Jvet common test conditions and software reference configurations*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 11 and JVET–J1010, Joint Video Experts Team (JVET), San Diego, CA, USA, pp. 10–20, Apr. 2018.
[35] M. Koo, M. Salehifar, J. Lim, and S. Kim, *Ce6: Fast dst-7/dct-8 Based on DFT (test 6.2.1)*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 13 and JVET–M0288, Joint Video Experts Team (JVET), Marrakech, MA, USA, pp. 10–18, Jan. 2019.
[36] A. Said, H. Egilmez, Y.-H. Chao, V. Seregin, and M. Karczewicz, *Ce6: Efficient Implementations of MTS With Transform Adjustments (tests 1.4a-d)*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 13, vols. JVET–M0538, Joint Video Experts Team (JVET), Marrakech, MA, USA, pp. 10–18, Jan. 2019.
[37] P. Philippe, *Ce6: Mts simplification with transform adjustment (TAF) (tests 1.5a-d)*, document ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29/WG 13 and JVET–M0080, Joint Video Experts Team (JVET), Marrakech, MA, USA, pp. 10–18, Jan. 2019.
[38] G. Bjontegaard, *Calculation of Average PSNR Differences Between Rd-Curves*, document ITU-T SG16/Q6 and VCEG-M33, Austin, TX, USA, pp. 10–18, Apr. 2001. [Online]. Available: http://wftp3.itu.int/av-arch/video-site/0104_Aus/
[39] G. Bjontegaard, *Improvement of Bd-PSNR Model*, document ITU-T SG16/Q6 and VCEG-AI11, Berlin, Germany, pp. 10–18, Jul. 2008. [Online]. Available: https://www.itu.int/wftp3/av-arch/video-site/0807_Ber/

Zhaobin Zhang received the B.S. and M.S. degrees in mechanical electronic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City. His research interests include machine learning, and image/video compression and processing.



Xin Zhao received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in computer applications from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

In 2017, he joined Tencent America, LLC, Palo Alto, CA, USA, where he is currently a Principal Researcher. From 2012 to 2017, he was a Staff Engineer with Qualcomm, San Diego, CA, USA. Since 2012, he has been actively contributing to the development of 3D extensions to H.264/AVC and HEVC standards with Joint Collaborative Team on 3D Video Coding Extensions (JCT-3V), the Development of Versatile Video Coding (VVC) Standard with Joint Video Exploration Team (JVET) and explorations of next-generation video coding technologies with Alliance for Open Media (AOM). His research interests include image and video coding, video processing, and transmission.

Xiang Li (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from Tsinghua University, Beijing, China, and the Dr.-Ing. degree in electrical, electronic and communication engineering from the University of Erlangen-Nuremberg, Germany. He was with Qualcomm and Siemens. He has been working in the field of video compression for years and is an active contributor to international video coding standards. He is currently a Senior Principal Researcher and the Head of video coding standards in the Tencent Media Lab, Tencent America, LLC. He has published more than 40 journals and conference papers, 300 standard contributions, and holds 120 U.S. granted and pending patents. He served as a chair and co-chair in a number of Ad Hoc groups, core experiments, including the Co-Chair of JEM Reference Software, VVC Reference Software, and a Co-Editor of MPEG-5 EVC.

Li Li (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2011 and 2016, respectively. He is currently a Visiting Assistant Professor with the University of Missouri–Kansas City. His research interest includes image/video coding and processing. He received the Best 10% Paper Award at the 2016 IEEE Visual Communications and Image Processing (VCIP) and the 2019 IEEE International Conference on Image Processing (ICIP).

Yi Luo received the M.S.E.E. degree from the California Institute of Technology, Pasadena, CA, USA. He has worked in speech, audio, and video processing industry for more than 20 years. He is currently a Senior Research with Tencent America, LLC, Palo Alto, CA, USA. His research interest includes the efficient software implementation of video codec and its algorithm development.

Shan Liu received the B.Eng. degree in electronics engineering from Tsinghua University and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California. She was the Chief Scientist and the Head of the America Media Lab, Futurewei Technologies. She was formerly the Director of the Multimedia Technology Division, MediaTek USA. She was also formerly with MERL, Sony, and IBM. She is currently a Tencent Distinguished Scientist and a General Manager of Tencent Media Lab, Tencent America, LLC. She has been actively contributing to international standards since the last decade and has numerous proposed technologies adopted into various standards. She holds more than 100 granted U.S. and global patents, some of which have been productized to serve millions of users daily. She served the Industrial Relationship Committee for the IEEE Signal Processing Society from 2014 to 2015 and the Vice President of the Industrial Relations and Development of Asia–Pacific Signal and Information Processing Association (APSIPA) from 2016 to 2017. She was named as APSIPA Industrial Distinguished Leader in 2018.

Zhu Li (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Northwestern University, in 2004. He was the AFRL Summer Faculty, U.S. Air Force Academy, UAV Research Center, in 2016, 2017, and 2018. He was a Sr. Staff Researcher/Sr. Manager with Samsung Research America's Multimedia Core Standards Research Lab, Dallas, from 2012 to 2015, a Senior Staff Researcher at FutureWei, from 2010 to 2012, an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, from 2008 to 2010, and a Principal Staff Research Engineer with the Multimedia Research Lab (MRL), Motorola Labs, Schaumburg, Illinois, from 2000 to 2008. He is currently an Associate Professor with the Department of CSEE, University of Missouri–Kansas City, Kansas City, USA, directs the NSF I/UCRC Center for Big Learning, UMKC. He has 46 issued or pending patents, more than 100 publications in book chapters, journals, conference proceedings, and standards contributions in these areas. His research interests include image/video analysis, compression, and communication and associated optimization and machine learning problems.

He received the Best Paper Award from the IEEE International Conference on Multimedia & Expo (ICME), Toronto, in 2006, and the Best Paper Award from the IEEE International Conference on Image Processing (ICIP), San Antonio, in 2007. He was an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA from 2015 to 2019, and the IEEE TRANSACTIONS ON CIRCUITS & SYSTEM FOR VIDEO TECHNOLOGY from 2016 to 2019. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING since 2019 and has also been an Associate Editor-in-Chief (AEiC) of the IEEE TRANSACTIONS ON CIRCUITS & SYSTEM FOR VIDEO TECHNOLOGY, since 2020.