Fast Adaptive Multiple Transform for Versatile Video Coding

Zhaobin Zhang*, Xin Zhao†, Xiang Li†, Zhu Li*, Shan Liu†

*University of Missouri Kansas City
5100 Rockhill Rd
Kansas City, MO 64110
{zzktb@mail.,lizhu@}umkc.edu

†Tencent America
661 Bryant St
Palo Alto, CA 94301
{xinzzhao,xlxiangli,shanl}@tencent.com

Abstract

The Joint Video Exploration Team (JVET) recently launched the standardization of next-generation video coding named Versatile Video Coding (VVC) in which the Adaptive Multiple Transforms (AMT) is adopted as the primary residual coding transform solution. AMT introduces multiple transforms selected from the DST/DCT families and achieves noticeable coding gains. However, the set of transforms are calculated using direct matrix multiplication which induces higher run-time complexity and limits the application for practical video codec. In this paper, a fast DST-VII/DCT-VIII algorithm based on partial butterfly with dual implementation support is proposed, which aims at achieving reduced operation counts and run-time cost meanwhile yield almost the same coding performance. The proposed method has been implemented on top of the VTM-1.1 and experiments have been conducted using Common Test Conditions (CTC) to validate the efficacy. The experimental results show that the proposed methods, in the state-of-the-art codec, can provide an average of 7%, 5% and 8% overall decoding time savings under All Intra (AI), Random Access (RA) and Low Delay B (LDB) configuration, respectively yet still maintains coding performance.

1 Introduction

In block-based hybrid video coding codecs, such as H.264/AVC [1] and HEVC [2], prediction and transform residual coding are two vital techniques to achieve efficient coding performance. The typical prediction process only targets at removing the statistical redundancy between the reference and current image blocks, while further removing the inter-pixel redundancy between residual samples is usually done by linear transforms [3]. In October 2015, ISO/IEC JTC1 SC29/WG11 MPEG and ITU-T SG16/Q6 VCEG worked together as the Joint Video Exploration Team (JVET) to explore state-of-the-art algorithms and prepare for the next-generation video standards beyond HEVC. In the on-going working draft of the next-generation video coding standard Versatile Video Coding (VVC) [4], Adaptive Multiple Transforms (AMT) is adopted as the primary residual transform solution. Instead of only using DCT-II, AMT introduces multiple core transforms from DST/DCT families, including DCT-II, DST-VII, DCT-VIII, DST-I and DCT-V.

Transform selection is performed based on Intra Prediction Modes (IPM). For each intra coded transform unit (TU), given the intra prediction mode, one of the three transform sets is first identified according to Table 1. The encoder then tries every transform core in the identified transform set for horizontal and vertical transform, the best one is explicitly signaled. As there are two transform candidates in each

transform set [5], so the transform selection is signaled with two-bit index, with one bit indicating either horizontal or vertical [3].

Table 1: Selection of (H)orizontal and (V)ertical transform set indices for each intra prediction mode [5].

IPM	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
H	2	1	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1
V	2	1	0	1	0	1	0	1	2	2	2	2	2	1	0	1	0	1
IPM	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
H	0	1	0	1	0	1	2	2	2	2	2	1	0	1	0	1	0	
V	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	

Since there are totally five kinds of transforms need to be evaluated to select the optimal category, the encoder run-time complexity is expensive. To alleviate this hassle, several fast algorithms have been studied in literature. In [5], a two-pass transform selection method is proposed for encoder in which the encoding time saving is achieved by skipping the second pass in some cases. In [6], a sparse method is devised to approximate the transforms in JEM7 which requires fewer operation counts. In [7], a method for implementing N-point DST-VII using $2N + 1$ point Discrete Fourier Transform (DFT) is proposed. However, existing fast methods are not capable of supporting all the desirable features of a transform design in the video codec, including 16-bit intermediate operation, integer operation, identical results between fast method and matrix multiplication.

In this paper, a fast DST-VII/DCT-VIII algorithm is proposed to reduce AMT time complexity. We exploit the properties that are considered useful for compression efficiency and for efficient implementation for DST-VII and DCT-VIII core transform basis. To ensure the orthogonality property, a tuning process is performed on the transform matrices. The proposed fast method achieves significant time savings without unwelcome coding performance. Experiments and a thorough analysis of the complexity demonstrate the effectiveness of the proposed method. A proposal based on this work is being studied as core experiment (CE) in the 11th MPEG JVET meeting [8].

The remainder of this paper is organized as follows. Section II elaborates the proposed fast method in detail. The complexity analysis is provided in Section III, both theoretically and practically. Section VI presents and discusses the experimental results. The paper is summarized and concluded in Section V.

2 Proposed Fast Method

In VVC, non-square transform block partition scheme is introduced. Given a $M \times N$ residual block, the two-dimensional transform can be implemented as a separate hor-

izontal one-dimensional transform followed by a vertical one-dimensional transform. Typically, the forward and inverse transform matrices are transposes of each other. Unless otherwise specified, all the following notations and properties are based on the DST-VII forward transform matrix. It should be understood that the inverse transforms are corresponding transpose matrices thereby the same implementation could be achieved. In addition, the same design can be seamlessly deployed to DCT-VIII.

2.1 Fast Transform

The basis functions of N-point 2D DST-VII and DCT-VIII are tabulated in Table 2. The transform core, which is a matrix consists of the basis vectors, of 16-point DST-VII is shown in Eq. 1, and similar representations can be obtained for other sizes.

$$T = \begin{bmatrix}
a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p \\
c & f & i & l & o & o & l & i & f & c & 0 & -c & -f & -i & -l & -o \\
e & j & o & m & h & c & -b & -g & -l & -p & -k & -f & -a & d & i & n \\
g & n & l & e & -b & -i & -p & -j & -c & d & k & o & h & a & -f & -m \\
i & o & f & -c & -l & -l & -c & f & o & i & 0 & -i & -o & -f & c & l \\
k & k & 0 & -k & -k & 0 & k & k & 0 & -k & -k & 0 & k & k & 0 & -k \\
m & g & -f & -n & -a & l & h & -e & -o & -b & k & i & -d & -p & -c & j \\
o & c & -l & -f & i & i & -f & -l & c & o & 0 & -o & -c & l & f & -i \\
p & -a & -o & b & n & -c & -m & d & l & -e & -k & f & j & -g & -i & h \\
n & -e & -i & j & d & -o & a & m & -f & -h & k & c & -p & b & l & -g \\
l & -i & -c & o & -f & -f & o & -c & -i & l & 0 & -l & i & c & -o & f \\
j & -m & c & g & -p & f & d & -n & i & a & -k & l & -b & -h & o & -e \\
h & -p & i & -a & -g & o & -j & b & f & -n & k & -c & -e & m & -l & d \\
f & -l & o & -i & c & c & -i & o & -l & f & 0 & -f & l & -o & i & -c \\
d & -h & l & -p & m & -i & e & -a & -c & g & -k & o & -n & j & -f & b \\
b & -d & f & -h & j & -l & n & -p & o & -m & k & -i & g & -e & c & -a
\end{bmatrix} \tag{1}$$

The variables $a, b, \ldots, p$ can be derived from the formulations defined in Table 2, and their values might be different for different size DST-VII transform matrix. To avoid floating point operations, the transform core of DST-VII is scaled by a pre-defined factor and rounded to the nearest integer, or further tuned by an offset.

Table 2: Basis functions of DST-VII and DCT-VIII for N-point input.

Transform Type	Basis function $T_i(j)$, $i, j = 0, 1, \ldots, N-1$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1})$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \cos(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2})$

The DST-VII core transform matrix has several properties that are considered useful for efficient implementation. These useful features are summarized as follows.

1. **Feature #1:** The first row vector contains N unique values. A replicate phenomenon is that the summation of several elements is equal to that of another few or one element.

2. **Feature #2:** Except for the row vectors satisfied Feature 1, some of the other row vectors have replicate patterns.

3. **Feature #3:** Except for the row vectors satisfied Feature 1 and 2, the remainder row vector(s) only contain(s) one or two unique absolute values.

From the 16-point DST-VII matrix (Eq. 1), the first basis vector has the following characteristic which corresponds to Feature 1.

$$a+j=l, \quad b+i=m, \quad c+h=n, \quad d+g=o, \quad e+f=p \tag{2}$$

Denote the input vector for a 16-point transform as $x = \{x_0, x_1, x_2, \ldots, x_{15}\}$, and the output transform coefficient vector as $y = \{y_0, y_1, y_2, \ldots, y_{15}\}$. Based on relationship (2), to calculate y_0, instead of doing vector-by-vector multiplication which requires 16 multiplications. The following alternative implementation can be done to derive the same 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 only 11 multiplications. In addition, when calculating y_2, y_3, y_6, y_8, y_9, y_{11}, y_{12}, y_{14}, y_{15}, the similar implementation can be done, and 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.

To explain Feature 2, take the second basis vector as an example. It can be divided

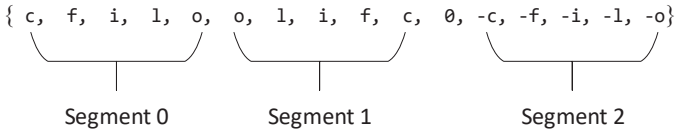

Figure 1: Repeating patterns are observed in Feature 2.

into 3 segments, and they are replicate with sign changes, or flipped version of each other. When calculating y_1, instead of doing the vector-by-vector multiplication which requires 15 multiplications, the following alternative implementation which requires only 5 multiplications can be done to derive the same results.

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

In addition, when calculating y_1, y_4, y_7, y_{10}, y_{13} the calculation can be done in the similar way, and intermediate results of $(x_0+x_9-x_{11})$, $(x_1+x_8-x_{12})$, $(x_2+x_7-x_{13})$, $(x_3+x_6-x_{14})$ and $(x_4+x_5-x_{15})$ can be re-used.

To explain Feature 3, take the 6th basis vector as an example. It is noticed that the 6th basis vector contains a single value k without considering the sign changes. Therefore, instead of calculating y_{10} using vector-by-vector multiplication which requires 11 multiplications, y_{10} can be alternatively derived by

$$y_{10} = k \cdot (x_0 - x_2 + x_3 - x_5 + x_6 - x_8 + x_9 - x_{11} + x_{12} - x_{14} + x_{15}) \tag{5}$$

which requires only 1 multiplication. For 16-point and 64-point forward and backward DST-VII, the above three features are also available, thus similar fast methods discussed above are also applicable. For the inverse transform, the transform core matrix is the transpose of the transform core matrix used for forward transforms, hence similar features could be utilized.

With the proposed method, identical results between fast method and matrix multiplication can be achieved, i.e., dual implementation support, which was considered as an important feature of the HEVC core transform design. With dual implementation support, the flexibility of selecting either fast method or matrix multiplication is provided to better optimize the complexity for a specific implementation scenario without any impact on the coded bitstream.

2.2 Transform Matrix Tuning

The core transform matrices of VVC are finite precision approximations of the DST/DCT matrix. This is beneficial for hardware implementation and avoids encoder-decoder mismatch as well as drift caused by manufacturers implementing the inverse transform with slightly different floating point representations. On the other hand, a disadvantage of using approximate matrix elements is that some of the properties discussed in Section 2.1 may not be satisfied anymore. There is a trade-off between the computational cost associated with using high bit-depth for the matrix elements and the degree to which some of the conditions are satisfied. In VVC test model VTM-2.0, transform matrix coefficients are represented in 10-bit. However, some of the vectors still cannot satisfy the above-mentioned properties due to the rounding error.

To make the matrix comply with these properties meanwhile minimize the side effects on compression efficiency, several principles have to be followed without any compromise.

1. The DST-VII/DCT-VIII matrix can be specified by using a small number of unique elements.

2. The orthogonality of DST-VII/DCT-VIII is optimized as much as possible.

3. The adjusted DST-VII and DCT-VIII basis is kept close to the original floating-point DST-VII and DCT-VIII basis.

In addition, several metrics are defined to measure the degree of approximation to the floating-point matrix, including the orthogonality measure, accuracy measure and norm measure. Given an integer N-point DST-VII/DCT-VIII approximation with scaled matrix elements equal to d_{ij} and basis vectors equal to $\mathbf{d}_i = [d_{i0}, \ldots, d_{i(N-1)}]^T$ where $i = 0, \ldots, N-1$, these measures are optimized using the following metric.

$$D = |\alpha^2 \cdot I - K \cdot K^T| \tag{6}$$

where $i, j = 0, \ldots, N-1$, and the scale factor $\alpha = 64 \cdot \sqrt{N}$. The tuning process is performed by traversing all possible integer values and measure the tuned matrix with the above three measures, the best one is selected as the final matrix. Meanwhile, during the optimization of the transform core K, it is restricted that per-element difference between K and K_0 has a magnitude being less than or equal to 1, such that the optimized transform basis is kept being close to the original DST-VII/DCT-VIII basis. Finally, by using 10-bit representation, all the three features are preserved which can be used to implement efficient fast methods.

3 Complexity Analysis

In this section, the complexity is analyzed in both arithmetic operations in theory and software execution time in practice.

3.1 Arithmetic Operations

With straightforward matrix multiplication, the number of operations for the 1D inverse transform is N^2 multiplications and $N(N-1)$ additions. Therefore, for a 2D transform, the number of multiplications required is $2N^3$ and the number of additions is $2N^2(N-1)$. Due to the adoption of the non-square transform structure, we will use the number of arithmetic operations for 1D inverse transform to compare the proposed method and the conventional matrix multiplication method.

Take the 16-point DST-VII inverse transform as an example, 256 multiplications and 240 additions are needed to generate a row vector. In the proposed fast method, there are 10 rows satisfying Feature 1, 5 rows satisfying Feature 2 and 1 row satisfying Feature 3. While performing Feature 1, there are 16 intermediate variables can be re-used which require 25 additions, another 1 intermediate variable that can be re-used which requires 1 multiplication. To leverage these intermediate variables generating final results, another 10×10 multiplications and 10×10 additions are required. Therefore, 101 multiplications and 125 additions are required in total to perform Feature 1. Similarly, 25 multiplications and 20 additions to finish the computation of rows satisfying Feature 2. To apply Feature 3, 1 multiplication and 10 additions are needed. Therefore, in total there are 127 multiplications and 155 additions using the proposed fast method, e.g., 50% and 35% arithmetic operations are saved for multiplication and addition, respectively.

The numbers of arithmetic operations required for the 1D N-point inverse transform of matrix multiplication and the proposed fast method are tabulated in Table

Table 3: The number of arithmetic operations for a 1D forward/inverse transform.

Transform Size	Matrix Multiplication			Partial Butterfly		
	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

3. 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.

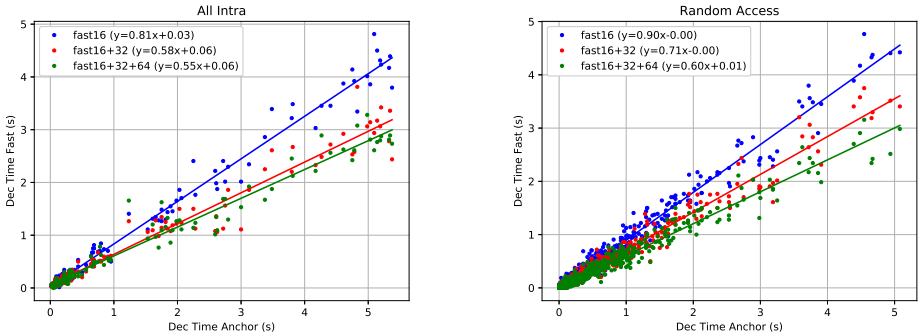

Figure 2: Software execution time of decoding transform under AI and RA configurations. The straight lines are linear regression approximation with L2 norm constraints.

3.2 Software Execution Time

In this subsection, the actual transform time of the proposed fast method is evaluated and compared with that of VTM-1.1 anchor. In this paper, 16-point, 32-point and 64-point DST-VII/DCT-VIII fast methods are implemented. Test sequences are VVC test sequences and the resolution ranging from 240P to 4K. In our implementation, transform time of both forward and inverse are collected. We encode and decode the sequences with different settings using VTM-1.1 and the proposed method, record the elapsed time for further analysis. Due to the limited space, we will only show the inverse transform time.

In order to see individual contribution of each block-level, we enable the proposed method gradually in block size. In Figure 2, *fast16* represents enabling fast method of block size 16, *fast16+32* refers to enabling fast method of block size 16 and 32, and *fast16+32+64* indicates enabling fast method of 16, 32 and 64. The vertical axis represents decoding transform time of fast method while horizontal axis represents that of anchor in seconds. The straight lines are the linear regression results with L2 norm metric. An average of 45% and 40% transform time savings have been achieved with all block-level enabled under AI and RA configurations, respectively. The time saving ratio increases from 19% to 45% when gradually enabling larger block size fast method under AI configuration. Under RA configuration, 10%, 29% and 40%

decoding time savings have been witnessed for *fast16*, *fast16+32* and *fast16+32+64*, respectively.

4 Experimental Results

4.1 Experiment settings

The proposed fast method has been integrated to VVC reference software VTM-1.1 [9]. The common test conditions [10], as defined by Joint Video Experts Team (JVET) for evaluating proposals during the VVC development, are used to perform the experiments. In those experiments, a set of 26 video sequences ranging from 416x240 to 3840x2160 are tested, including four artificial sequences with computer screen and mixed natural and screen content. It should be noted that Class D is not included into the overall average performance in the summary. The decoding time and encoding time are measured in seconds. The run-time ratio of the proposed method to anchor as defined in Eq. 7 is used to evaluate the time complexity, with 100% represents no run-time saving.

$$\Delta T = \frac{T_{proposed}}{T_{anchor}} \times 100\% \tag{7}$$

The quantization parameters (QP) are 22, 27, 32 and 37. AI, RA and LDB configurations have been tested. Under AI configuration, every picture is coded as Intra while under RA, intra period is set as one second, GOP size is set as 8. The codec is operating in 10-bit mode, RDOQ is enabled.

4.2 Run-time Performance

The run-time results are tabulated in Table 4. An average of 7%, 5% and 6% decoding time savings have been achieved for AI, RA and LDB, respectively. The perk decoding time saving occurs on Sequence *Johnny* of 17%, *FoodMarkert4* of 11% and *BasketballDrillText* of 12% for AI, RA and LDB, respectively. In terms of encoding time savings, 3%, 6% and 8% average time savings have been observed for AI, RA and LDB, respectively. The best encoding time saving is achieved on Sequence *RaceHorses* with 21%, *FoodMarket4* with 8% and *SlideShow* with 22% for AI, RA and LDB, respectively. Though some sequences happen to occupy more time than VTM-1.1 due to the execution time jitter of CPU, e.g., *BQMall* in AI settings, the overall time saving is apparently quite encouraging.

4.3 BD-Rate Performance

To validate the tuned transform cores can achieve comparable PSNR performance, we show the BD-Rate performance compared with VTM-1.1 in Table 5. The BD-Rate are -0.02%, 0.00% and 0.01% for AI, RA and LDB, respectively. Therefore, the tuned transform cores are able to yield almost the same coding performance on top of the time savings. The effects caused by the replacement of transform cores can be neglected.

Table 4: Run-time performance of proposed fast method compared with VTM-1.1 anchor.

Class	Sequence	All Intra		Random Access		Low Delay	
		ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}	ΔT_{Enc}	ΔT_{Dec}
A1	Tango2	96%	86%	94%	91%	-	-
	FoodMarket4	100%	88%	92%	89%	-	-
	CampfireParty2	95%	96%	93%	94%	-	-
A2	CatRobot1	100%	93%	95%	97%	-	-
	DaylightRoad2	96%	93%	93%	95%	-	-
	ParkRunning3	98%	92%	95%	97%	-	-
B	MarketPlace	98%	90%	94%	94%	90%	89%
	RitualDance	99%	98%	93%	96%	91%	90%
	Cactus	100%	94%	93%	94%	90%	92%
	BasketballDrive	96%	92%	95%	91%	92%	95%
	BQTerrace	98%	99%	97%	101%	92%	95%
C	BasketballDrill	96%	100%	94%	94%	91%	95%
	BQMall	100%	101%	96%	98%	97%	95%
	PartyScene	97%	102%	95%	100%	95%	100%
	RaceHorses	98%	99%	95%	94%	91%	93%
D	BasketballPass	93%	97%	95%	93%	93%	92%
	BQSquare	105%	98%	96%	98%	98%	97%
	BlowingBubbles	101%	88%	97%	97%	94%	91%
	RaceHorses	79%	83%	96%	98%	99%	98%
E	FourPeople	99%	94%	-	-	93%	95%
	Johnny	101%	83%	-	-	91%	96%
	KristenAndSara	98%	91%	-	-	90%	94%
Average		97%	93%	94%	95%	92%	94%

Table 5: BD-Rate performance of proposed fast method (tuned transform core) compared with VTM-1.1 anchor.

Class	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
A1	0.00%	-0.05%	0.04%	-0.01%	0.00%	-0.11%	-	-	-
A2	0.00%	0.01%	0.00%	0.01%	0.11%	-0.07%	-	-	-
B	0.00%	0.00%	0.00%	-0.03%	0.03%	0.16%	-0.01%	-0.13%	0.48%
C	0.00%	0.00%	0.00%	0.04%	-0.02%	0.18%	0.02%	-0.06%	-0.09%
D	0.01%	-0.02%	0.11%	0.03%	-0.25%	0.04%	0.06%	-0.06%	-0.43%
E	0.02%	-0.03%	-0.07%	-	-	-	0.03%	-0.68%	-0.46%
Average	0.00%	-0.01%	-0.01%	0.00%	0.03%	0.06%	0.01%	-0.24%	0.05%

5 Conclusion

This paper proposes a fast method aims at reducing the computational complexity for the primary residual coding transform solution AMT, which was recently adopted by

JVET as a core contribution of the next-generation video coding framework. Inherent features of DST/DCT families have been investigated comprehensively to reduce the number of arithmetic operations. A low-complexity transform scheme with dual implementation support has been proposed based on partial butterfly. Both theoretical and empirical analysis has been conducted to validate the effectiveness of the proposed method. The results show that the proposed method achieves better time efficiency than state-of-the-art codec VTM-1.1 without compromise on the coding performance.

6 References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[2] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

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

[4] B. Bross, J. Chen, and S. Liu, "Jvet common test conditions and software reference configurations," *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, vol. JVET-K1001, no. Ljubljana, SI, pp. 10–18, July 2018.

[5] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W. J. Chien, "Enhanced multiple transform for video coding," in *2016 Data Compression Conference (DCC)*, March 2016, pp. 73–82.

[6] A. Said, H. Egilmez, V. Seregin, and M. Karczewicz, "Complexity reduction for adaptive multiple transforms (amts) using adjustment stages," *Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, vol. JVET-J0066, no. San Diego, US, pp. 10–20, April 2018.

[7] M. Koo, J. Heo, J. Nam, N. Park, J. Lee, J. Choi, S. Yoo, H. Jang, L. Li, J. Lim, S. Paluri, M. Salehifar, and S. Kim, "Description of sdr video coding technology proposal by lg electronics," *JVET-J0017, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, vol. JVET-J0017, no. San Diego, US, pp. 10–20, April 2018.

[8] Z. Zhang, X. Zhao, X. Li, and S. Liu, "Ce6-related: Fast dst-7/dct-8 with dual implementation support," *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, vol. JVET-K0291, no. Ljubljana, SI, pp. 10–18, July 2018.

[9] JVET VVC, "Versatile video coding (vvc) reference software: Vvc test model (vtm)," `https://jvet.hhi.fraunhofer.de/svn/svn_VVCSoftware_VTM/tags/VTM-1.1/`, October 2018.

[10] J. Boyce, K. Suehring, X. Li, and V. Seregin, "Jvet common test conditions and software reference configurations," *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, vol. JVET-J1010, no. San Diego, US, pp. 10–20, April 2018.