A Butterfly Structured Design of The Hybrid Transform Coding Scheme

Jingning Han, Yaowu Xu, and Debargha Mukherjee
Google Inc., 1950 Charleston Road, Mountain View, CA 94043
Emails: {jingning,yaowu,debargha}@google.com

***Abstract*—The hybrid transform coding scheme that alternates amongst the asymmetric discrete sine transform (ADST) and the discrete cosine transform (DCT) depending on the boundary prediction conditions, is an efficient tool for video and image compression. It optimally exploits the statistical characteristics of prediction residual, thereby achieving significant coding performance gains over the conventional DCT-based approach. A practical concern lies in the intrinsic conflict between transform kernels of ADST and DCT, which prevents a butterfly structured implementation for parallel computing. Hence the hybrid transform coding scheme has to rely on matrix multiplication, which presents a speed-up barrier due to under-utilization of the hardware, especially for larger block sizes. In this work, we devise a novel ADST-like transform whose kernel is consistent with that of DCT, thereby enabling butterfly structured computation flow, while largely retaining the performance advantages of hybrid transform coding scheme in terms of compression efficiency. A prototype implementation of the proposed butterfly structured hybrid transform coding scheme is available in the VP9 codec repository.**

I. Introduction

Transform coding is a central component in video and image compression. Many research efforts have been devoted to optimize the transform kernel to fully exploit signal correlation for compression gains. A recent approach that jointly optimized spatial prediction and the choice of the subsequent transform for video and image compression was developed in [1] and [2], where it was shown that the optimal Karhunen-Loeve transform (KLT) given available, partial boundary information is well approximated by a close relative of the discrete sine transform (DST), with basis vectors that tend to vanish at the known boundary and maximize energy at the unknown boundary. The overall intra coding scheme thus switches between this variant of DST named asymmetric DST (ADST), and the conventional DCT, depending on the prediction direction and boundary information. This adaptive prediction-transform approach, namely hybrid transform coding scheme, was experimentally shown to significantly outperform the DCT-based intra-frame prediction-transform coding.

On the hardware design side, the transform module typically contributes a large portion of codec computational complexity, and hence a butterfly structured implementation that allows parallel computing via single instruction multiple data (SIMD) operations [3] is highly desirable. The design of butterfly structured discrete cosine transform can trace back to 1970's (e.g., [4].) A recent development on fast transform using integer transform was proposed in [5], where it approximates the DCT transform element-wisely using a matrix whose entries are all small integers. The computation hence only involves additions and shifts. Similar principle was also applied to the ADST in [2], both targetting at 4×4 block size. In fact, methods along this line are typically limited to smaller transform dimensions. When it comes to transform dimension of 8×8 or above, it is in general difficult to find an orthogonal matrix whose elements are small integers and approximates the DCT closely. Hence, its advantages in simple computation will gradually disappear as transform size grows. It is noteworthy that larger block size transforms provides higher transform coding gains for stationary signal and are experimentally proved to contribute compression efficiency in various video codecs.

Challenges arise in the design of fast ADST, and hence hybrid transform coding scheme, of any block sizes. The original ADST kernel was derived as $sin(\frac{n\ (2k-1)\pi}{2N+1})$ in [1], where N is the block dimension, n and k denote the time and frequency indexes, respectively, both ranging from 1 to N. The DCT kernel, on the other hand, is of form $cos(\frac{(2n-1)(k-1)\pi}{2N})$. The butterfly structured implementations of sinusoidal transforms exist if and only if the denominator of the kernel argument, i.e., $(2N+1)$ for the ADST and $2N$ for DCT, is a composite number (and ideally can be decomposed into product of small integers). For this reason, most block-based video (and image) codecs are designed to make the block size power of two, e.g., N = 4, 8, 16, etc, for efficient computation of DCT transformation. It, however, makes the original ADST not capable of fast implementation. For example, when $N = 8$, $(2N + 1)$ turns out to be 17, which is a prime number that precludes the possibility of butterfly structure.

This work resolves this intrinsic conflict between DCT and the original ADST by designing a new variant of ADST whose kernel is of the form $sin(\frac{(2n-1)(2k-1)\pi}{4N})$. Clearly the denominator of the kernel argument, $4N$, is consistent with that of DCT, in that if $2N$ is a power of two, so is $4N$. Therefore, it can be implemented in a butterfly structure, and is henceforth referred to as btf-ADST. A prototype butterfly structured implementation of btf-ADST was initially provided in [6]. The btf-ADST is a basis-wise approximation to the original ADST, and well preserves the its superior coding performance given the boundary condition. We hence use this btf-ADST to replace the original ADST in the hybrid transform coding scheme. The overall scheme thus selects the appropriate 1-D transforms amongst the btf-ADST and DCT depending on the prediction direction to form the 2-D transformation. Further note that while there are many variants of butterfly design for the btf-ADST and DCT, most of them targeted only on reducing the number of multiplications without consideration on the rounding effects of intermediate steps[1], which potentially affects the accuracy of the computation hence incurring round-trip error. In our implementation, we use the structure that has more rotation (which involves more multiplication) steps in the first few stages, so as to reduce the accuracy impact due to rounding error. It is experimentally demonstrated that our proposed approach, in conjunction with SIMD, significantly reduces the runtime of hybrid transform coding scheme in terms of CPU cycles, while largely retaining its compression gains.

II. Spatial Prediction and Transform Coding

We revisit the mathematical theory that derived the original ADST, in the context 1-D first-order Gauss-Markov model, given partial

[1]In practice, all the computations are performed in the integer format for speed reasons.

prediction boundary [1], which leads to our btf-ADST proposed in this work.

Consider a zero-mean, unit variance, first-order Gauss-Markov sequence

$$x_k = \rho x_{k-1} + e_k, \tag{1}$$

where ρ is the correlation coefficient, and e_k is a white Gaussian noise process with variance $1-\rho^2$. Let $\underline{x} = [x_1, x_2, \cdots, x_N]^T$ denote the random vector to be encoded given x_0 as the available (one-sided) boundary. The superscript T denotes matrix transposition. The recursion (1) translates into the following set of equations:

$$\begin{aligned} x_1 &= \rho x_0 + e_1 \\ x_2 - \rho x_1 &= e_2 \\ &\vdots \\ x_N - \rho x_{(N-1)} &= e_N, \end{aligned} \tag{2}$$

or in compact notation:

$$Q\underline{x} = \underline{b} + \underline{e} \tag{3}$$

where

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ -\rho & 1 & 0 & 0 & \dots \\ 0 & -\rho & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -\rho & 1 \end{pmatrix}, \tag{4}$$

and $\underline{b} = [\rho x_0, 0, \cdots, 0]^T$ and $\underline{e} = [e_1, e_2, \cdots, e_N]^T$ capture the boundary information and innovation process, respectively. It can be shown that Q is invertible, and thus:

$$\underline{x} = Q^{-1}\underline{b} + Q^{-1}\underline{e}\ , \tag{5}$$

where the superscript -1 indicates matrix inversion. As expected, the "boundary response" or prediction, $Q^{-1}\underline{b}$, in (5) satisfies

$$Q^{-1}\underline{b} = [\rho x_0, \rho^2 x_0, \cdots, \rho^N x_0]^T\ . \tag{6}$$

The prediction residual

$$\underline{y} = Q^{-1}\underline{e} \tag{7}$$

is to be compressed and transmitted, which motivates the derivation of its KLT. The autocorrelation matrix of $\underline{y}$ is given by:

$$R_{\underline{y}\underline{y}} = E\{\underline{y}\underline{y}^T\} = Q^{-1}E\{\underline{e}\underline{e}^T\}(Q^T)^{-1} = (1-\rho^2)Q^{-1}(Q^T)^{-1}. \tag{8}$$

Thus, the KLT for $\underline{y}$ is a unitary matrix that diagonalizes $Q^{-1}(Q^T)^{-1}$, and hence also the more convenient:

$$P_1 = Q^T Q = \begin{pmatrix} 1+\rho^2 & -\rho & 0 & 0 & \dots \\ -\rho & 1+\rho^2 & -\rho & 0 & \dots \\ 0 & -\rho & 1+\rho^2 & -\rho & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & -\rho & 1+\rho^2 & -\rho \\ 0 & \dots & 0 & -\rho & 1 \end{pmatrix}. \tag{9}$$

Although P_1 is Toeplitz, note that the element at the bottom right corner is different from all the other elements on the principal diagonal, i.e., it is not $1+\rho^2$. This irregularity complicates an analytic derivation of the eigenvalues and eigenvectors of P_1. As a subterfuge, we approximate P_1 with

$$\hat{P}_1 = \begin{pmatrix} 1+\rho^2 & -\rho & 0 & 0 & \dots \\ -\rho & 1+\rho^2 & -\rho & 0 & \dots \\ 0 & -\rho & 1+\rho^2 & -\rho & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & -\rho & 1+\rho^2 & -\rho \\ 0 & \dots & 0 & -\rho & 1+\rho^2-\rho \end{pmatrix} \tag{10}$$

which is obtained by replacing the bottom-right corner element with $1+\rho^2-\rho$. The approximation clearly holds for $\rho \to 1$, which is indeed a common approximation that describes the spatial correlation of video/image signals. The unitary matrix T_S that diagonalizes $\hat{P}_1$, and hence an approximation for the required KLT of $\underline{y}$, can be shown as the following relative of the common DST:

$$[T_S]_{j,i} = \left(\frac{2}{\sqrt{2N+1}} \sin\frac{(2j-1)i\pi}{2N+1}\right) \tag{11}$$

where $j, i \in \{1, 2, \cdots, N\}$ are the frequency and time indexes of the transform kernel respectively. Needless to say, the constant matrix T_S is independent of the statistics of the innovation e_k, and can be used as an approximation for KLT when boundary information x_0 is available.

III. Butterfly Structured Variant of ADST

A key observation of the above derived ADST is that the rows of T_S (i.e., basis functions of the transform) possess smaller values in the beginning (closer to the known boundary), and larger values towards the other end. For instance, consider the row with $j=1$ (i.e., the basis function with the lowest frequency). In the case where $N \gg 1$, the first sample ($i=1$) is $\frac{2}{\sqrt{2N+1}} sin\frac{\pi}{2N+1} \approx 0$, while the last sample ($i=N$) takes the maximum value $\frac{2}{\sqrt{2N+1}} sin\frac{N\pi}{2N+1} \approx \frac{2}{\sqrt{2N+1}}$. This effectively exploits the fact that pixels closer to the known boundary are better predicted and hence have statistically smaller variance than those at far end. It inspires our search for a unitary sinusoidal transform that resembles the compression performance of the ADST, to overcome the intricacy of butterfly design of ADST and hence hybrid transform coding for parallel computing. We hence devise a new variant of DST as an alternative:

$$[T_{btf}]_{j,i} = \left(\sqrt{\frac{2}{N}} sin\frac{(2j-1)(2i-1)\pi}{4N}\right), \tag{12}$$

where $j, i \in \{1, 2, \cdots, N\}$ denote the frequency and time indexes respectively. Clearly, it also possesses the property of asymmetric basis function, but has the denominator of kernel argument, $4N$, consistent with that of DCT, thereby allowing the butterfly structured implementation. We refer to it as btf-ADST henceforth.

A butterfly structure of btf-ADST was initially provided in [6], where it assumed no precision loss in the intermediate steps that involve multiplications by irrational numbers. In practice, all these computations are performed in the integer format, which inevitably incurs rounding effects accumulated through every stage. To minimize the round-trip error, we modify the structure to make the initial stages consist more multiplications, so that the rounding errors are less magnified. In keeping the conventions used in [6], let $\overline{I}$ and D_N

denote the re-ordering operations:

$$\overline{I} = \begin{bmatrix} & & & 1 \\ & & \cdots & \\ & \cdots & & \\ 1 & & & \end{bmatrix}$$

$$D_N = \begin{bmatrix} 1 & & & & \\ & -1 & & & \\ & & 1 & & \\ & & \cdots & & \\ & & & -1 \end{bmatrix}. \quad (13)$$

Let P_J be the permutation matrix that move the first half of the vector entries to the even-numbered position, and the second half entries to the odd-numbered position but in a reversed order:

$$P_J = \begin{bmatrix} 1 & 0 & \cdots & & \\ 0 & \cdots & & 0 & 1 \\ 0 & 1 & \cdots & & \\ 0 & \cdots & & 1 & 0 \\ \cdots & \cdots & & & \cdots \end{bmatrix}, \quad (14)$$

where J is the height of the matrix. It formulates a second permutation:

$$H_N = P_N \begin{bmatrix} P_{N/2} & \\ & P_{N/2} \end{bmatrix} \cdots \begin{bmatrix} P_4 & & & \\ & \overline{I}_4 P_4 \overline{I}_4 & & \\ & & \cdots & \\ & & P_4 & \\ & & & \overline{I}_4 P_4 \overline{I}_4 \end{bmatrix}. \quad (15)$$

Similarly the permutation operator Q_J moves the odd-numbered entries to be in reversed order:

$$Q_J = \begin{bmatrix} 1 & 0 & \cdots & & & \\ 0 & \cdots & & 0 & 1 & \\ 0 & 0 & 1 & \cdots & & \\ 0 & \cdots & & 1 & 0 & \\ \cdots & \cdots & & & \cdots & \\ 0 & 1 & 0 & \cdots & 0 & \end{bmatrix}. \quad (16)$$

Let $J = log_2 N$, we define the following building blocks that formulate the butterfly structure.

Type 1 Translational Operators: matrices $U_N(j)$, $j = 1, 2, \cdots, J-1$ are defined as

$$U_N(j) = \begin{bmatrix} B(j) & & & \\ & B(j) & & \\ & & \cdots & \\ & & & B(j) \end{bmatrix}, \quad (17)$$

where

$$B(j) = \begin{bmatrix} I_{2^j} & I_{2^j} \\ I_{2^j} & -I_{2^j} \end{bmatrix}. \quad (18)$$

Type 2 Rotational Operators: $V_N(j)$, $j = 1, 2, \cdots, J-1$, are block diagonal matrices:

$$V_N(j) = \begin{bmatrix} I_{2^j} & & & \\ & E(j) & & \\ & & I_{2^j} & \\ & & \cdots & \\ & & & E(j) \end{bmatrix}, \quad (19)$$

where $E(j) = diag\{T_{1/2^{j+1}}, T_{5/2^{j+1}}, \cdots, T_{(2^{j+1}-3)/2^{j+1}}\}$ and

$$T_r = \begin{bmatrix} cos\ r\pi & sin\ r\pi \\ sin\ r\pi & -cos\ r\pi \end{bmatrix}. \quad (20)$$

As a special case,

$$V_N(J) = \begin{bmatrix} T_{1/4N} & & & \\ & T_{5/4N} & & \\ & & \cdots & \\ & & & T_{(2N-3)/4N} \end{bmatrix}. \quad (21)$$

Given the above established building blocks, the btf-ADST can be decomposed as:

$$T_{btf} = D_N \cdot H_N^T \cdot V_N(1) \cdot U_N(1) \cdot V_N(2) \cdots U_N(J-1) \cdot V_N(J) \cdot Q_N \cdot \overline{I}_N, \quad (22)$$

which directly translates into a butterfly graph. The theoretical coding performance of the two ADST variants and DCT compared against the KLT will be provided next, followed by the experimental codec evaluation.

IV. Quantitative Analysis

We quantitatively evaluate the performance of the btf-ADST, original ADST, and DCT, against the KLT (of $\underline{y}$ in Sec. II) in terms of coding gains [7] under the assumed signal model, at different correlation coefficient values.

Let the prediction residual, $\underline{y}$, be transformed to

$$\underline{z} = A\underline{y} = [z_1, z_2, ..., z_N]^T \quad (23)$$

with an $N \times N$ unitary matrix A. The objective of the encoder is to distribute a fixed number of bits to the different elements of $\underline{z}$ such that the average distortion is minimized. This bit-allocation problem is addressed by water filling algorithm of [7]. Under assumptions such as a Gaussian source, high-quantizer resolution, negligible quantizer overload, and with non-integer bit-allocation allowed, it can be shown that the minimum distortion (mean squared error) obtainable is proportional to the geometric mean of the transform domain sample variances $\sigma_{z_i}^2$, i.e.,

$$D_A \propto (\prod_{i=1}^{N} \sigma_{z_i}^2)^{1/N}, \quad (24)$$

where for Gaussian source the proportionality coefficient is independent of the transform A. These variances can be obtained as the diagonal elements of the autocorrelation matrix of $\underline{z}$:

$$R_{\underline{z}\underline{z}} = E[\underline{z}\underline{z}^T] = AE[\underline{y}\underline{y}^T]A^T = (1-\rho^2)AP_1^{-1}A^T \quad (25)$$

where we have used (8) and (9). The coding gain in dB of any transform A is now defined as:

$$\mathcal{G}_A = 10\log_{10}(D_{\mathbf{I}}/D_A). \quad (26)$$

Here $\mathbf{I}$ is the $N \times N$ identity matrix, and hence $D_{\mathbf{I}}$ is the distortion resulting from direct quantization of the untransformed vector $\underline{y}$. The coding gain, $\mathcal{G}_A$ thus provides a comparison of the average distortion incurred with and without the transformation A. Note that for any given A (including the btf-ADST, ADST, DCT, and KLT of $\underline{y}$), computing $R_{\underline{z}\underline{z}}$, and hence $\sigma_{z_i}^2$, does not require making any approximations for P_1. When A is the KLT of $\underline{y}$, $R_{\underline{z}\underline{z}}$ is a diagonal matrix (with diagonal elements equal to the eigen values of P_1^{-1}), the transform coefficients z_i are uncorrelated, and the coding gain reaches its maximum.

Fig. 1 compares btf-ADST, ADST and DCT in terms of their coding gains, relative to KLT, specifically it depicts $\mathcal{G}_{T_{btf}} - \mathcal{G}_{\mathbf{KLT}}$, $\mathcal{G}_{T_s} - \mathcal{G}_{\mathbf{KLT}}$, and $\mathcal{G}_{T_c} - \mathcal{G}_{\mathbf{KLT}}$, versus the correlation coefficient ρ. Clearly the original ADST well approximates KLT at various values of the correlation coefficient ρ. The maximum gap between

ADST and KLT, or the maximum loss of optimality, is less than 0.05 dB. The proposed btf-ADST closely resembles the performance of ADST and KLT, at a maximum loss of 0.15 dB. In comparison, DCT performs poorly (by about 0.55 dB loss) for the practically relevant case of high correlation ($\rho \approx 0.95$). At low correlation ($\rho \to 0$), the autocorrelation matrix of the prediction residual, $R_{\underline{yy}} \approx \mathbf{I}$, and hence any unitary matrix, including ADST and DCT, will function as a KLT. The block length used in obtaining the results of Fig. 1 is $N = 8$.

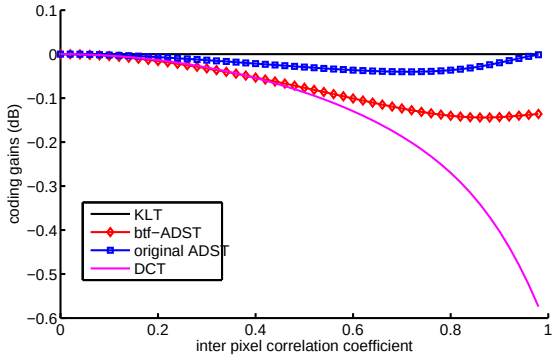

Fig. 1. Theoretical coding gains of btf-ADST, ADST, and DCT, relative to KLT, all applied to the prediction residuals, plotted versus the inter pixel correlation coefficient; the block dimension is 8×8.

V. Experimental Results

The proposed btf-ADST was employed to replace the original ADST in the hybrid transform coding scheme. This btf-ADST/DCT hybrid transform coding scheme was implemented in the VP9 codec [8]. We first evaluated its compression efficiency as compared to the original ADST/DCT hybrid transform and the conventional 2D-DCT coding scheme. Fig. 2 demonstrates the rate-distortion performance comparison for sequence $harbour$ at CIF resolution. Clearly, btf-ADST/DCT closely tracks the original ADST/DCT, while both significantly outperform the conventional 2D-DCT approach. Similar results were observed over a wide varieties of sequences and resolutions.

We next consider the computational efficiency of the proposed btf-ADST/DCT hybrid transform. The btf-ADST enables the butterfly structured implementation, which in conjunction with SIMD operations provides significant speed-up as compared to the original ADST operating via matrix multiplication. We compare the runtime of the btf-ADST/DCT and the original ADST/DCT hybrid transform schemes, in terms of the average CPU cycles, as shown in Fig. 3. The implementation was using streaming SIMD extension 2 (SSE2) and the experiments were running on a 64-bit platform. Evidently the proposed btf-ADST allows efficient hardware utilization and thereby substantial codec speed-up, while closely resembling the compression gains of the original ADST.

VI. Conclusions

This work devised a novel variant of ADST transform whose kernel approximates the original ADST basis-wisely and is consistent with the DCT kernel, thereby enabling the butterfly structured implementation. The proposed scheme allows efficient hardware utilization for significant codec speed-up, while largely retaining the advantageous compression performance of hybrid transform coding scheme.

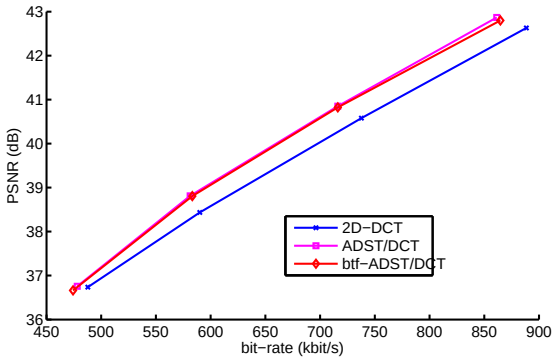

Fig. 2. Coding performance comparison of 2D-DCT, ADST/DCT hybrid transform, and btf-ADST/DCT hybrid transform for sequence $harbour$ at CIF resolution.

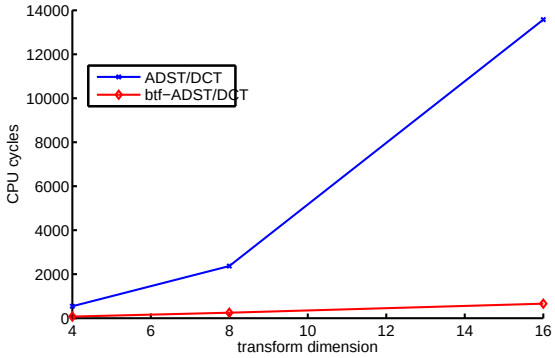

Fig. 3. Computational complexity in terms of CPU cycles. The btf-ADST was implemented using streaming SIMD extension 2 and the experiments were running on a 64-bit platform.

References

[1] J. Han, A. Saxena, and K. Rose, "Towards jointly optimal spatial prediction and adaptive transform in video/image coding," *IEEE Proc. ICASSP*, pp. 726–729, Mar. 2010.

[2] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Trans. on Image Processing*, vol. 21, pp. 1874–1884, April 2012.

[3] J. L. Hennessy and D. A. Patterson, *Computer Architecture-a quantitative approach, 4th Ed.*, Mogan Kaufmann, 2007.

[4] W.-H. Chen, C. Smith, and S. Fraclick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. on Communications*, vol. 25, pp. 1004–1009, Sep. 1977.

[5] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.

[6] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, vol. 32, no. 4, pp. 803–816, Aug. 1984.

[7] N. S. Jaynat and P. Noll, "Digital coding of waveforms," *Engle-wood Cliffs, NJ: Prentice-Hall*, 1984.

[8] open source, *http://www.webmproject.org/code/*.