

## Converting DCT Coefficients to H.264/AVC

Jun Xin, Anthony Vetro, Huifang Sun

TR2004-058 June 2004

### Abstract

Many video coding schemes, including MPEG-2, use a Discrete Cosine Transform (DCT). The recently completed video coding standard, H.264/AVC, uses an integer transform, which will be referred to as HT in this paper. We propose an efficient method to convert DCT coefficients to HT coefficients entirely in the transform domain. We show that the conversion is essentially a 2D transform. We derive the transform kernel matrix; provide a fast algorithm and an integer approximation of the transform. We show that the proposed transform domain conversion outperforms the conventional pixel domain approach. It is expected to have applications in transform domain video transcoding.

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.



**Publication History:**

1. First printing, TR-2004-058, June 2004



# Converting DCT Coefficients to H.264/AVC Transform Coefficients

Jun Xin, Anthony Vetro, and Huifang Sun

Mitsubishi Electric Research Laboratories, Cambridge MA 02139, USA  
jxin, avetro, hsun@merl.com

**Abstract.** Many video coding schemes, including MPEG-2, use a Discrete Cosine Transform (DCT). The recently completed video coding standard, H.264/AVC, uses an integer transform, which will be referred to as HT in this paper. We propose an efficient method to convert DCT coefficients to HT coefficients entirely in the transform domain. We show that the conversion is essentially a 2D transform. We derive the transform kernel matrix, provide a fast algorithm and an integer approximation of the transform. We show that the proposed transform domain conversion outperforms the conventional pixel domain approach. It is expected to have applications in transform domain video transcoding.

**Keywords** — Transform domain video transcoding, video transcoding, H.264/AVC, MPEG-2.

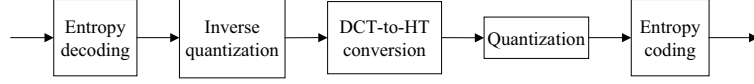
## 1 Introduction

The transform used in many video coding schemes, including MPEG-2 [1], MPEG-1, and H.263 etc., is a Discrete Cosine Transform (DCT). The recently completed video coding standard, H.264/AVC [2], uses a low-complexity integer transform, hereinafter referred to as HT.

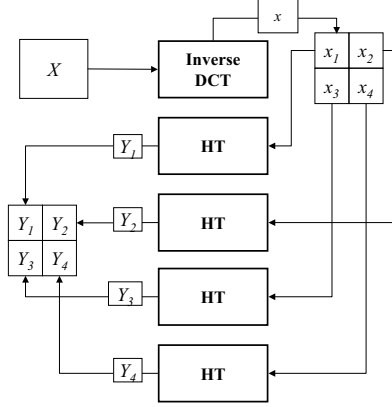
One important step in the transcoding of video from MPEG-2 format to H.264/AVC format is to convert the coefficients from the DCT domain to the HT domain, i.e. DCT-to-HT conversion. Fig. 1 shows the DCT-to-HT conversion in the context of intra-frame video transcoding.

Fig. 2 shows a pixel domain implementation of the DCT-to-HT conversion. The input is an  $8 \times 8$  block ( $X$ ) of DCT coefficients. An inverse DCT (IDCT) is applied to  $X$  to recover an  $8 \times 8$  pixel block ( $x$ ). The  $8 \times 8$  pixel block is divided evenly into four  $4 \times 4$  blocks ( $x_1, x_2, x_3, x_4$ ). Each of the four blocks is passed to a corresponding HT to generate four  $4 \times 4$  blocks of transform coefficients ( $Y_1, Y_2, Y_3, Y_4$ ). The four blocks of transform coefficients are combined to form a single  $8 \times 8$  block ( $Y$ ). This is repeated for all blocks of the video.

It is desired to perform the transcoding entirely in the compressed or transform domain, then reconstructing the image pixels is avoided. Transform domain transcoding could be more efficient than the pixel domain transcoding because complete decoding and reencoding are not required [3]. Therefore, there is a



**Fig. 1.** Intra transcoding



**Fig. 2.** Pixel domain HT-to-DCT conversion

need to have an efficient method to perform the DCT-to-HT conversion in the transform domain.

The paper is organized as follows. The proposed transform domain DCT-to-HT conversion is presented in Section 2. Section 3 and Section 4 discuss the fast algorithm and the integer approximation for the conversion respectively. Simulation results are given in Section 5. Section 6 concludes this paper.

## 2 DCT-to-HT Conversion

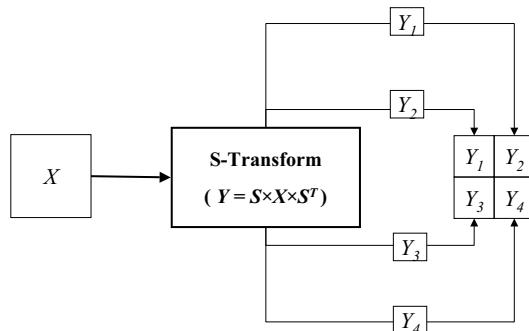
Fig. 3 shows our proposed transform domain DCT-to-HT conversion. In this paper, this conversion shall be called *S-transform*. It may be applied to the input DCT coefficients ( $X$ ) of an input video in the MPEG-2 format to produce output HT coefficients ( $Y$ ) of an output video in the AVC format. The S-transform is characterized by a transform kernel matrix,  $S$ , which is an  $8 \times 8$  matrix:

$$Y = SX S^T \quad (1)$$

where  $S^T$  is the transpose of  $S$ . We shall derive  $S$  in the following.

The HT of  $x_1, x_2, x_3$  and  $x_4$  are  $Y_1, Y_2, Y_3$ , and  $Y_4$  respectively (Fig. 2), i.e.

$$\begin{aligned} Y_1 &= H x_1 H^T \\ Y_2 &= H x_2 H^T \\ Y_3 &= H x_3 H^T \\ Y_4 &= H x_4 H^T \end{aligned} \quad (2)$$



**Fig. 3.** Transform domain DCT-to-HT conversion

where  $H$  is the transform kernel matrix of HT:  $H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$

If  $K = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix}$ , then we can rewrite (2) into a single equation

$$Y = K \times x \times K^T \quad (3)$$

where  $x$  is the IDCT of  $X$ . Let  $T_8$  be the transform kernel matrix of DCT, we have  $x = T_8^T X T_8$ . It then follows that

$$Y = K \times T_8^T \times X \times T_8 \times K^T \quad (4)$$

Comparing (4) with (1), we have

$$S = K \times T_8^T \quad (5)$$

Therefore, the direct DCT-to-HT transform is given by (1) and its transform kernel matrix  $S$ , is

$$S = \begin{pmatrix} a & b & 0 & -c & 0 & d & 0 & -e \\ 0 & f & g & h & 0 & -i & -j & k \\ 0 & -l & 0 & m & a & n & 0 & -o \\ 0 & p & j & -q & 0 & r & g & s \\ a & -b & 0 & c & 0 & -d & 0 & e \\ 0 & f & -g & h & 0 & -i & j & k \\ 0 & l & 0 & -m & a & -n & 0 & o \\ 0 & p & -j & -q & 0 & r & -g & s \end{pmatrix} \quad (6)$$

where the values  $a \dots s$  are (rounded off to four decimal places)

$$\begin{aligned} a &= 1.4142, & b &= 1.2815, & c &= 0.45, & d &= 0.3007, & e &= 0.2549, \\ f &= 0.9236, & g &= 2.2304, & h &= 1.7799, & i &= 0.8638, & j &= 0.1585, \\ k &= 0.4824, & l &= 0.1056, & m &= 0.7259, & n &= 1.0864, & o &= 0.5308, \\ p &= 0.1169, & q &= 0.0922, & r &= 1.0379, & s &= 1.975. \end{aligned}$$

### 3 Fast DCT-to-HT Conversion

The sparseness and symmetry of the S-transform kernel matrix  $S$  can be exploited to perform fast computation of the transform.

As suggested by (1), the 2D S-transform is separable. Therefore, it can be achieved through 1D transforms, i.e., column transforms followed by row transforms. Hence, we shall describe only the computation of the 1D transform.

Let  $\mathbf{z}$  be an 8-point column vector, and a vector  $\mathbf{Z}$  be the 1D transform of  $\mathbf{z}$ . The following steps provide a method to determine  $\mathbf{Z}$  efficiently from  $\mathbf{z}$ , which is also shown in Fig. 4 as a flow-graph.

$$\begin{aligned}
 m1 &= a \times z[1] \\
 m2 &= b \times z[2] - c \times z[4] + d \times z[6] - e \times z[8] \\
 m3 &= g \times z[3] - j \times z[7] \\
 m4 &= f \times z[2] + h \times z[4] - i \times z[6] + k \times z[8] \\
 m5 &= a \times z[5] \\
 m6 &= -l \times z[2] + m \times z[4] + n \times z[6] - o \times z[8] \\
 m7 &= j \times z[3] + g \times z[7] \\
 m8 &= p \times z[2] - q \times z[4] + r \times z[6] - s \times z[8] \\
 Z[1] &= m1 + m2 \\
 Z[2] &= m3 + m4 \\
 Z[3] &= m5 + m6 \\
 Z[4] &= m7 + m8 \\
 Z[5] &= m1 - m2 \\
 Z[6] &= m4 - m3 \\
 Z[7] &= m5 - m6 \\
 Z[8] &= m8 - m7
 \end{aligned}$$

The method needs 22 multiplications and 22 additions. It follows that the 2D S-transform needs 352 ( $=16 \times 22$ ) multiplications and 352 additions, for a total of 704 operations.

The pixel domain implementation, as illustrated in Fig. 2, includes one IDCT and four HT operations. Chen's fast IDCT implementation [4], referred to herein as *the reference IDCT*, needs 256 ( $=16 \times 16$ ) multiplications and 416 ( $=16 \times 26$ ) additions. Each HT needs 16 ( $=2 \times 8$ ) shifts and 64 ( $=8 \times 8$ ) additions [5]. The four HT then need 64 shifts and 256 additions. It follows that the overall computational requirement of the pixel domain processing is 256 multiplications, 64 shifts and 672 additions, for a total of 992 operations.

Thus, the fast S-transform saves about 30% of the operations when compared to the pixel domain implementation. In addition, the S-transform can be implemented in just two stages, whereas the conventional pixel domain processing using the reference IDCT requires six stages, where the reference IDCT needs four and the HT needs two.

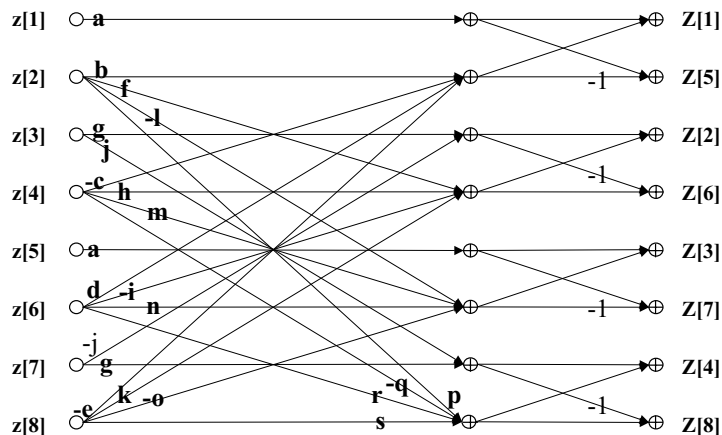


Fig. 4. Fast algorithm for the transform domain DCT-to-HT conversion

## 4 Integer Approximation of DCT-to-HT Conversion

Floating-point operations are generally more expensive to implement than integer operations. Therefore, we also provide an integer approximation of the S-transform.

We multiple  $S$  by an integer that is a power of two, and use the integer transform kernel matrix to perform the transform using an integer-arithmetic. Then, the resulting coefficients are scaled down by proper shifting. In video transcoding applications, the shifting operations can be absorbed in the quantization. Therefore, no additional operations are required to use integer arithmetic.

The larger the integer we select, the better accuracy we may achieve. In many applications, the number is limited by the microprocessor on which the transcoding is performed. We describe how to choose the number such that the computation can be performed using a 32-bit arithmetic, which is within the capability of most microprocessors.

For the case of the DCT-to-HT conversion, the DCT coefficients lie in the range of -2048 to 2047. This is a dynamic range of 4096, and needs 12 bits to represent. The maximum sum of absolute values in any row of  $S$  is 6.44, so the maximum dynamic range gain for the 2D S-transform is  $6.44^2 = 41.47$ , which means  $\log_2(41.47) = 5.4$  extra bits. Therefore, 17.4 bits are needed to represent the final S-transform results. To be able to use the 32-bit arithmetic, the scaling factor must be smaller than the square root of  $2^{32-17.4}$ , i.e. 157.4. The maximum integer satisfying this condition while being a power of two is 128.

Therefore, the integer transform kernel matrix is  $SI = \text{round}\{128 \times S\}$ . Similar to  $S$ ,  $SI$  has the form (6), but with the values  $a$  through  $s$  changed to

the following integers:

$$\begin{aligned} a &= 181, b = 164, c = 58, d = 38, e = 33, \\ f &= 118, g = 285, h = 228, i = 111, j = 20, \\ k &= 62, l = 14, m = 93, n = 139, o = 68, \\ p &= 15, q = 12, r = 133, s = 253. \end{aligned}$$

The fast algorithm derived in Sect. 3 for the S-transform can be applied to the above transform since  $SI$  and  $S$  have the same symmetric property.

## 5 Simulation Results

### 5.1 Simulation Conditions

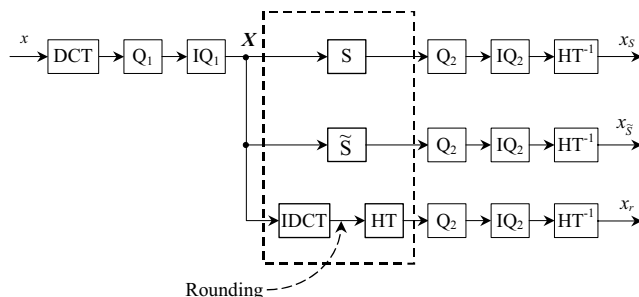
Fig. 5 shows the simulation setting. An  $8 \times 8$  block,  $x$ , is DCT-transformed, quantized ( $Q_1$ ) and inverse-quantized ( $IQ_1$ ). The reconstructed DCT coefficients,  $X$ , are sent to three processing systems. Each of the three systems map  $X$  into the HT-domain, which are then reconstructed through quantization ( $Q_2$ ), inverse-quantization ( $IQ_2$ ), and inverse-transform ( $HT^{-1}$ ). The DCT-to-HT conversion schemes used are: the real-arithmetic S-transform ( $S$ ), the integer-arithmetic S-transform ( $\tilde{S}$ ), and the reference IDCT-HT (IDCT followed by HT). The pixel-blocks reconstructed from the three systems are denoted as  $x_S$ ,  $x_{\tilde{S}}$  and  $x_r$  respectively. We use PSNR to measure the distortion between the source and the reconstructed pixel-blocks.

In the simulations,  $x$  is generated using a stationary Gaussian random process, with zero mean and standard deviation  $\sigma$ . If  $u$  is the distance between two pixels in  $x$ , their correlation coefficient is  $\rho^u$ . We calculate the parameters for 200 frames of the following sequences: Akiyo, Stefan, Container and Mobile&Calendar. We choose  $(\rho, \sigma)$  to be  $(0.99, 10)$  and  $(0.90, 30)$  to cover these typical sequences. We use  $Q_1=2, 4, 6, 8$ , and  $Q_2$  from 10 to 295. We take the average of 10000 runs for each experiment.

### 5.2 Integer S-transform vs. Real S-transform

Fig. 6 shows the PSNR difference between using the integer and the real S-transform. It is observed that the PSNR loss resulted from the integer approximation decreases with increasing  $Q_2$  and/or  $Q_1$ , which is expected since the increasing quantization error makes the approximation error less significant. In addition, when  $Q_2$  is low, the PSNR differences vary with  $Q_1$ , and appear to be signal dependent. It is interesting to note that for some values of  $Q_1$  and  $Q_2$ , the integer arithmetic achieves a positive PSNR gain. Nevertheless, the PSNR difference is small, with the maximum difference around 0.03dB.

In practice, it may be hardly useful to have a finer re-quantization ( $Q_2$ ) than the input quantization ( $Q_1$ ) since it is impossible to improve upon the quality of the input coded video. The equivalent quantization step sizes in the DCT domain and the HT domain are different due to the non-orthonormal HT. The



**Fig. 5.** Simulation settings

equivalent quantization step size in the HT domain is about 5~6 times that in the DCT domain for fine quantizations. Therefore, it can be observed that in the practical use range of  $Q_2$ , the quality loss is even less (around 0.01dB or less).

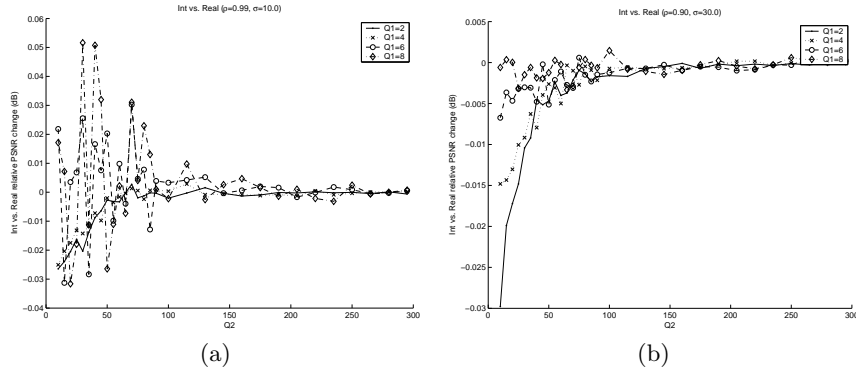
### 5.3 Integer S-transform vs. Reference IDCT-HT

Fig. 7 shows the PSNR difference between using the integer S-transform and the reference IDCT-HT. Interestingly, the integer S-transform actually outperforms the reference implementation. The inferior performance of the reference method is caused by the rounding operation after the IDCT. The rounding is the standard decoding step following IDCT in order to reconstruct pixels. In addition, the HT is an integer-transform and demands integer input. For the integer S-transform, the rounding takes place for the result HT coefficients, but not for the intermediate results. The improvement of the integer S-transform could be as much as almost 0.35dB for  $Q_1=2$  and  $Q_2=10$ . For practical transcoding, where only coarser re-quantization may be useful, the PSNR gain is generally within 0.2dB. When  $Q_2$  and/or  $Q_1$  increases, the gain diminishes as the quantization error dominates the distortion.

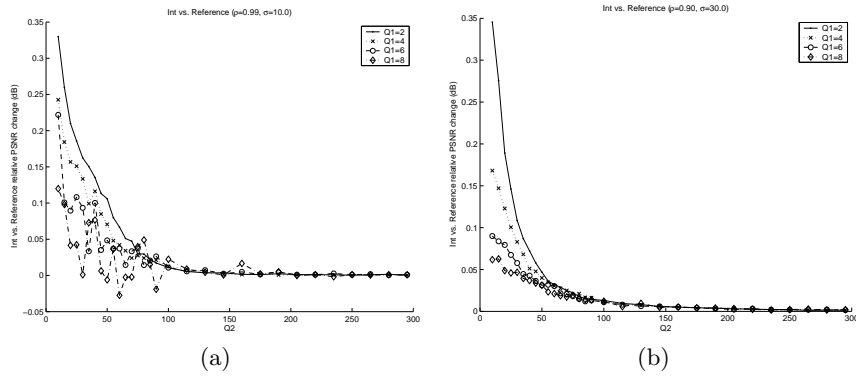
## 6 Concluding Remarks

We introduced a transform domain approach for the conversion of DCT coefficients to HT coefficients. We showed that the DCT-to-HT conversion can be implemented in the transform domain by a single transformation. We derived the transformation kernel matrix and developed efficient algorithms for computing the transform. We also provided an integer approximation of the transform. Our simulation results showed that the proposed transformation achieved improved PSNR performance over the conventional pixel domain implementation while requiring reduced computational complexity.

Our developed transform domain DCT-to-HT conversion can be applied to the transcoding of DCT-based video such as MPEG-2 to HT-based H.264/AVC video. We are currently developing techniques for such a transcoder.



**Fig. 6.** Relative PSNR difference of integer vs. real S-transform



**Fig. 7.** Relative PSNR difference of integer S-transform vs. reference pixel domain conversion

## References

1. ISO/IEC 13818-2: Information technology - Generic coding of moving pictures and associated audio information: Video. Edition 2 (2000)
2. Weigand, T., Sullivan, G.J., Bjøntegaard G., Luthra A.: Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7 (2003) 560-576
3. Keesman, G., Hellinghuizen, R., Hoeksema, F., Heideman, G.: Transcoding of MPEG bitstreams. *Signal Processing: Image Communication*, vol. 8 (1996) 481-500
4. Chen, W.H., Smith, C.H., Fralick, S.C.: A Fast Computation Algorithm for The Discrete Cosine Transform. *IEEE Trans. Commun.*, vol. COM-25 (1977) 1004-1009
5. Malvar, H.S., Hallapuro, A., Karczewicz, M., Kerofsky, L.: Low-Complexity Transform and Quantization in H.264/AVC. *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7 (2003) 598-603