May 2006, Vol.21, No.3, pp.354–359 J. Comput. Sci. & Technol.

Low Complexity Integer Transform and Adaptive Quantization Optimization

Si-Wei Ma[1] (马思伟) and Wen Gao[1,2] (高 文)

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China*

[2] *Department of Computer Science, Harbin Institute of Technology, Harbin 150001, P.R. China*

E-mail: {swma, wgao}@jdl.ac.cn

Received October 31, 2005; revised March 8, 2006.

Abstract In this paper, a new low complexity integer transform is proposed, which has been adopted by AVS1-P7. The proposed transform can enable AVS1-P7 to share the same quantization/dequantization table with AVS1-P2. As the bases of the proposed transform coefficients are very close, the transform normalization can be implemented only on the encoder side and the dequantization table size can be reduced compared with the transform used in H.264/MPEG-4 AVC. Along with the feature of the proposed transform, adaptive dead-zone quantization optimization for the proposed transform is studied. Experimental results show that the proposed integer transform has similar coding performance compared with the transform used in H.264/MPEG-4 AVC, and would gain near 0.1dB with the adaptive dead-zone quantization optimization.

Keywords AVS, discrete cosine transform (DCT), integer transform, quantization

1 Introduction

Transform coding is an important video coding technique and has been widely used in many international video coding standards, such as MPEG-1/2, H.261/2/3, etc. After transform, the spatial redundancy is attenuated and the compression can be achieved by the following quantization and entropy coding. From the energy compaction viewpoint, KLT (Karhunen-Loeve transform) is the best transform. However, it is difficult to use KLT in image and video coding because it is signal-dependent. DCT is a better approximation of KLT and easy for implementation due to its signal independence. But the float point multiplication in DCT is too complex and cannot map integer to integer losslessly.

In the past, many researches have been done on the integer-friendly approximation of the float DCT, such as binDCT[1,2] and IntDCT[3], where the float DCT coefficient is approximated as an integer coefficient by a multiplier and a right shift. So the DCT transform can be implemented only by using shift and add. In the development of H.264, the transform is improved, and the integer transform was first proposed to H.264 in [4] (in fact, it was first used in TML). Then many techniques on integer cosine transform (ICT, or integer transform: IT) have been proposed to H.264, such as 16×16 ICT[5], 4×4 IT[6,7] and adaptive block transform (ABT)[8]. Finally, a low complexity 4×4 integer transform was adopted[9] (8×8 transform was developed for H.264 the FRx profile later, and ABT was adopted by the FRx profile too[10,11]). The integer transform has many advantages, such as low complexity, exact invertibility, etc. Combined with the normalization to the integer transform, a division-free quantization scheme is used in H.264/MPEG-4 AVC. But in the quantization/dequantization scheme of H.264/MPEG-4 AVC, a big quantization/dequantization table should be stored to reach transform normalization and dequantization/quantiztion, as the bases of the transform coefficients are different.

AVS is the latest video/audio coding standard developed in China. In AVS standard, the part 2 and the part 7 are two video coding standards, called as AVS1-P2 and AVS1-P7 respectively. In AVS, the transform was improved and a new kind of integer transform was developed[12,13]. AVS transform is characteristic as the bases of the transform coefficients are very close, so that the transform normalization can be realized only on the encoder side, and that the dequantization table size in the decoder is reduced. In this paper a low complexity integer transform is proposed, which has been adopted by AVS1-P7. In conjunction with the new characteristic of the proposed transform, adaptive dead-zone quantization optimization is also studied. Experimental results show that the proposed transform has similar coding performance as that of H.264/MPEG-4 AVC, and would gain nearly 0.1dB with the optimized adaptive dead-zone quantization.

The rest of the paper is organized as follows. Section 2 makes an in-depth study on the integer transform and quantization scheme in H.264/MPEG-4 AVC. In Section 3, the proposed transform and adaptive dead-zone quantization are discussed. Experimental results are provided in Section 4. Section 5 concludes the paper.

2 Integer Transform

A typical 4×4 orthogonal transform can be expressed as:

Short Paper

Supported by the National Natural Science Foundation of China under Grant No. 60333020, and the National Basic Research 973 Program of China under Grant No. 2001CCA03300.

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}. \quad (1)$$

To keep low distortion from 4×4 DCT, the following condition should be met:

$$b/c \approx 2.4143 \quad (2)$$

In H.264/MPEG-4 AVC, the forward transform is approximated with $a = 1$, $b = 2$, $c = 1$; and the transform process can be expressed as:

$$Y = AXA^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}.$$

Since the integer transform is not a unitary matrix, Y must be normalized with:

$$W = Y \otimes E = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix} \otimes \begin{bmatrix} 1/m^2 & 1/mn & 1/m^2 & 1/mn \\ 1/mn & 1/n^2 & 1/mn & 1/n^2 \\ 1/m^2 & 1/mn & 1/m^2 & 1/mn \\ 1/mn & 1/n^2 & 1/mn & 1/n^2 \end{bmatrix},$$

where $m = 2$, $n = \sqrt{10}$ are the norms of the first row and the second row of A respectively. In H.264/MPEG-4 AVC, the scale matrix E is combined together with quantization/dequantization process, in which the transform normalization and quantization can be reached only through the multiplication and right shift, as follows:

$$\begin{aligned} Y_Q(i,j) &= Y(i,j) \times E(i,j)/Q_{\text{step}}(QP) \\ &= Y(i,j) \times Q(QP,i,j)/2^{r1}, \end{aligned}$$

where QP is the quantization parameter, and $Q_{\text{step}}(QP)$ is the corresponding quantization step size. $Q(QP,i,j)$ is the quantization table on encoder side and $QP(QP,i,j)$ equals $QP(QP\%6,i,j)$ in H.264/MPEG-4 AVC. For the inverse quantization, it is implemented in the same way as:

$$\begin{aligned} Y_{DQ}(i,j) &= Y_Q(i,j) \times E(i,j) \times Q_{\text{step}}(QP) \\ &= (Y_Q(i,j) \times DQ(QP,i,j))/2^{r2}, \end{aligned}$$

and

$$Q(QP,i,j) \times M(i,j) \times DQ(QP,i,j) \times M(i,j) = 2^{r1+r2}.$$

$DQ(QP,i,j)$ is the dequantization table on the decoder side and $DQ(QP,i,j)$ equals $DQ(QP\%6,i,j)$ in H.264/MPEG-4 AVC. The above transform, quantization, dequantization and inverse transform all can be implemented in 16-bit operation.

However, in H.264/MPEG-4 AVC, the quantization parameter QP ranges from 0 to 51. In hardware implementation, if we extend the quantization/dequantization table for all QPs, the table size would be $52 \times 4 \times 4$. In fact, if all the rows of the transform have the same norm, the quantization/dequantization table size would be reduced to be 52×1. In old version of H.264/MPEG-4 AVC, such as TML9[15], an integer transform with $a = 13$, $b = 17$, $c = 7$ in (1) was used. Every row of the transform has the same norm, and both the quantization and dequantization table size is 32×1 (QP ranges from 0 to 31 for TML9). But the transform and quantization process in TML9 require 32-bit arithmetic operation that would increase hardware complexity.

In the development of AVS1-P2, an 8×8 transform is adopted, shown as follows:

$$T_8 = \begin{bmatrix} 8 & 10 & 10 & 9 & 8 & 6 & 4 & 2 \\ 8 & 9 & 4 & -2 & -8 & -10 & -10 & -6 \\ 8 & 6 & -4 & -10 & -8 & 2 & 10 & 9 \\ 8 & 2 & -10 & -6 & 8 & 9 & -4 & -10 \\ 8 & -2 & -10 & 6 & 8 & -9 & -4 & 10 \\ 8 & -6 & -4 & 10 & -8 & -2 & 10 & -9 \\ 8 & -9 & 4 & 2 & -8 & 10 & -10 & 6 \\ 8 & -10 & 10 & -9 & 8 & -6 & 4 & -2 \end{bmatrix}$$

The transform has such a good feature that the bases of the transform coefficients are very close, so that the transform normalization can be realized only on the encoder side, and the dequantization table size on the decoder side is reduced. In the following section, the details for this feature will be discussed and a low complexity 4×4 integer transform is proposed. Then adaptive quantization was studied for the new kind of integer transform. The proposed integer transform and quantization/dequantization can reduce the dequantization table size while using 16-bit arithmetic operation.

3 Low Complexity Integer Transform and Adaptive Quantization Optimization

3.1 Low Complexity Integer Transform

To develop a low complexity 4×4 transform from (1), besides the above constraint (2), an additional constraint should also be met:

$$b^2 + c^2 = 2a^2. \quad (3)$$

That means the basis function is the same, which would reduce transform normalization complexity. But only one solution exists for both (2) and (3) in the range [0..20], just as the transform used in TML9 ($a = 13$, $b = 17$, $c = 7$). In the development of AVS1-P7, we propose a new low complexity 4×4 integer transform, where (2) and (3) are approximated with $a = 2$, $b = 3$, $c = 1$, as the 8×8 transform used in AVS1-P2. The proposed transform is shown as follows:

$$T_A = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 1 & -1 & -3 \\ 2 & -2 & -2 & 2 \\ 1 & -3 & 3 & -1 \end{bmatrix}$$

In [15], the distortion of transform with respect to DCT is used to evaluate the performance of a transform. The distortion of a transform T with respect to DCT (T_{DCT}) is defined as:

$$d_2 = 1 - \frac{1}{N}\|\mathrm{diag}(T_{DCT} \cdot T^T)\|_2^2,$$

where $\mathrm{diag}(X)$ denotes the main diagonal of matrix X. Table 1 shows the distortion of the proposed transform T_A with respect to DCT and that of the transform $T_{\mathrm{H.264}}$ in H.264/MPEG-4 AVC. From Table 1, we can see that the proposed transform has the same distortion with respect to DCT as that of the transform in H.264/MPEG-4 AVC.

Table 1. Distortion of $T_{\mathrm{H.264}}$ and T_A from DCT

d_2	$T_{\mathrm{H.264}}$	T_A
$d_{2,0}$	0.0000	0.0000
$d_{2,1}$	0.0050	0.0050
$d_{2,2}$	0.0000	0.0000
$d_{2,3}$	0.0050	0.0050
Average	0.0025	0.0025

The norms of the first row and second row in the proposed transform T_A is 4 and $\sqrt{20}$ respectively. After the two dimensional transform in (2), the original coefficients in X are scaled by the following matrix compared with DCT:

$$M = \begin{bmatrix} 4\times 4 & 4\times\sqrt{20} & 4\times 4 & 4\times\sqrt{20} \\ 4\times\sqrt{20} & \sqrt{20}\times\sqrt{20} & 4\times\sqrt{20} & \sqrt{20}\times\sqrt{20} \\ 4\times 4 & 4\times\sqrt{20} & 4\times 4 & 4\times\sqrt{20} \\ 4\times\sqrt{20} & \sqrt{20}\times\sqrt{20} & 4\times\sqrt{20} & \sqrt{20}\times\sqrt{20} \end{bmatrix}.$$

According to the quantization scheme in H.264/MPEG-4 AVC, the following expression should be true for the quantization and dequantization table:

$$Q(QP,i,j)\times M(i,j)\times DQ(QP,i,j)\times M(i,j) = 2^{total_bits}, \quad (4)$$

where *total_bits* is the total right shift bits for quantization/dequantization in encoder and decoder. If we separate the dequantization parameter $DQ(QP,i,j)$ into two parts: one is used to do normalization, and the other is used to do dequantization, then (4) can be expressed as:

$$Q(QP,i,j)\times M(i,j)\times DM(i,j)\times DQ(QP) \times M(i,j) = 2^{total_bits}$$

As the bases of the transform coefficients are very close, $DM(i,j)$ can be implemented on encoder side with little effect on the coding efficiency. That is to say:

$$(Q(QP,i,j)\times M(i,j)\times DM(i,j))\times DQ(QP) \times M(i,j) = 2^{total_bits}$$

Table 2 is the quantization/dequantization table used in our experiments.

Table 2. Quantization/Dequantization Table

QP%6	$Q(i,j)=$ $(0,0),(0,2)$ $(2,0),(2,2)$	$Q(i,j)=$ $(1,1),(1,3)$ $(3,1),(3,3)$	$Q(i,j)=$ others	DQ
0	26214	20972	16777	10
1	23831	19065	15252	11
2	20165	16132	12906	13
3	18725	14980	11984	14
4	16384	13107	10486	16
5	14564	11651	9321	18

Here, $Q(QP,i,j)\times M(i,j)\times DQ(QP)\times M(i,j) = 2^{26}$. For the forward transform, total right shift is $18 + QP/6$ bits, and for the inverse transform, total right shift is $8 - QP/6$ bits.

In AVS1-P2, a new quantization scheme is used. For AVS1-P7, the proposed 4×4 transform can share the same quantization/dequantation table with AVS1-P2. In AVS, after transform, the transform coefficient matrix Y is normalized with a scale table *ScaleTbl*:

$$Y'_{i,j} = (Y_{i,j} \times ScaleTbl[i][j] + 2^{r-bit1-1}) \gg r,$$

where r is equal to 19 for AVS1-P2 reference model and 15 for AVS1-P7 reference model respectively. The scale table is defined in Table 3 and Table 4 for AVS1-P2 and AVS1-P7 respectively. i and j are equal to 0..7 for AVS1-P2, and 0..3 for AVS1-P7.

After transform normalization, the quantization is done as:

$$YQ_{i,j} = (Y'_{i,j} \times Q[QP] + 2^{b-1}) \gg b.$$

$Q[QP]$ is the quantization table on the encoder side. The dequantization is shown as follows:

$$X'_{i,j} = (\,YQ_{i,j} \times DQ[QP] + 2^{s(QP)-1}) \gg s(QP),$$

where QP is the quantization parameter, $DQ(QP)$ is the inverse quantization table and $s(QP)$ is the shift value for inverse quantization. AVS1-P7 uses the same quantization and dequantization table as that in AVS1-P2 except for a little difference on the value of b. b is 15 for AVS1-P2 reference model and 19 for AVS1-P7 reference model. $Q(QP)$, $DQ(QP)$ and $s(QP)$ are shown in Table 5.

3.2 Transform Complexity Analysis

Fig.1 shows the fast implementation of the proposed 4×4 transform. The proposed transform can be implemented only by addition and shift. Compared with the 4×4 transform used in H.264, only 16 additions and 16 shifts are added for 2D forward transform. The total complexity is 80 additions and 32 shifts for 2D 4×4 transform.

Table 3. *ScaleTbl* for AVS1-P2, $i,j = 0,\ldots,7$

i%4	j%4			
	0	1	2	3
0	32768	37958	36158	37958
1	37958	43969	41884	43969
2	36158	41884	39898	41884
3	37958	43969	41884	43969

Table 4. *ScaleTbl* for AVS1-P7

i	j			
	0	1	2	3
0	32768	26214	32768	26214
1	26214	20972	26214	20972
2	32768	26214	32768	26214
3	26214	20972	26214	20972

Table 5. $Q(QP)$, $DQ(QP)$ and $s(QP)$

QP	0	1	···	63
$Q[QP]$	32768	29775	···	140
$DQ[QP]$	32768	36061	···	60099
$s[QP]$	14	14	···	7

Table 6. Bit Width for the Transform Coefficients

Operation	$(i,j)=$ $(0,0),(0,2),(2,0),(2,2)$		$(i,j)=$ $(1,1),(1,3),(3,1),(3,3)$		Other (i,j)	
	Bit operation	Bit width (bit)	Bit operation	Bit width (bit)	Bit Operation	Bit width (bit)
X	$9+0$	9	$9+0$	9	$9+0$	9
$A=X\cdot T^T$	$9+3$	12	$9+3$	12	$9+3$	12
$B=T\cdot A$	$12+3$	15	$12+3$	15	$12+3$	15
$C=ScaleTbl\cdot B\gg 15$	$15+15-15$	15	$15+14.36-15$	14.36	$15+14.68-15$	14.68
$D=Q\cdot C\gg 19$	$15+15-19$	11	$14.36+15-19$	10.36	$14.68+15-19$	10.68
$E=D\cdot DQ\gg s[QP]$	$11+15-14$	12	$10.36+15-14$	11.36	$10.68+15-14$	11.68
$F=E\cdot T$	$12+1$	13	$11.36+1.32$	12.68	$11.68+1.16$	12.84
$G=T^T\cdot F$	$13+1$	14	$12.68+1.32$	14.00	$12.84+1.16$	14.00
$H=G\gg 5$	$14-5$	9	$14.00-5$	9.00	$14.00-5$	9.00

The bit accuracy of the proposed transform can be guaranteed in 16 bits. For transform coefficients at different positions, the bit width variation is shown in Table 6. For example, at (0, 0) position, the input residual data is 9 bits. After row transform, the bit width is 12 bits, and after column transform it will be 15 bits. The scale factor is 32768 for (0, 0) position, so it will be 15 bits after multiplication with the scale table and 15 bits right shift followed. Then the result is multiplied with quantization factor 32768 (for $QP=0$) with 19 bits right shift. After quantization, the final bit width is 11 bits. So far, all operations on the encoder side are finished. On the decoder side, the input is processed with inverse quantization first. After the inverse quantization, the bit width is 12 bits. After the inverse transform and 5 bits right shift, the final result is 9 bits.

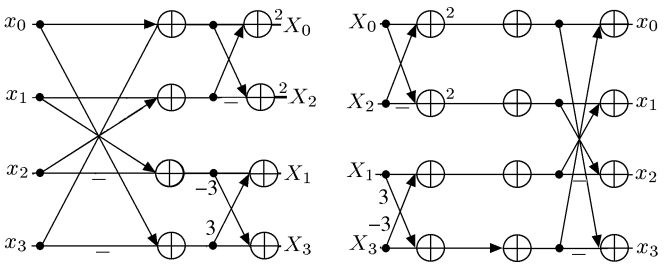

Fig.1. Fast implementation of the proposed 4×4 forward transform (left) and backward transform (right).

3.3 Adaptive Quantization Optimization

In fact, the above normalization scheme has its own shortcoming. As $DM(i,j)$ varies with (i,j), and the normalization scheme on encoder would lead to uneven magnification for different coefficients. That is to say the coefficients that are scaled more by the transform in decoder would be scaled lower at encoder side, and would lead to higher quantization degree relatively. On the contrary, the coefficients that are scaled less at decoder side would be scaled more at encoder side and would lead to lower quantization degree relatively. So sometimes the non-zero coefficients would emerge at high frequency. In this paper, adaptive dead-zone quantization scheme[16] was studied to resolve the problem. A typical dead-zone quantization scheme is shown as Fig.2.

In Fig.2, the value near a neighbor area of zero (from $-\Delta$ to Δ) is quantized to zero. In this paper, the adaptive dead-zone quantization scheme is implemented as follows:

$$Y_Q(i,j)=[Y(i,j)\times Q(QP,i,j)+f(i,j)\times 2^{15+QP/6})/2^{15+QP/6}.$$

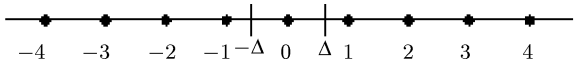

Fig.2. Dead-zone quantization scheme.

In [17], an adaptive quantization encoding technique was proposed by using an equal expected-value rule. Here, a near optimal solution can be found with an iterative method by using the following pseudo-code:

First, initialize a set S of dead-zone value $d_0,d_1,\ldots,d_{15}$ in non-increase order, e.g., $d_0=1/2$, $d_1=1/3,\ldots$. Second, set a default dead-zone weighting matrix—all values initialized with d_0. Third, compute the rate-distortion cost *min_rdcost* with the dead-zone matrix. Finally, set $d=d_0$, where d is the dead-zone for the current coefficient in zig-zag scan order. d is 0 now. Then,

```
for (each coefficient in zig-zag order)
{
  find_new = false;
  for (each value d' in the range from d to d15 in the dead-zone set in decrease order)
  {
    Compute the r-d cost rdcost with the current dead-zone weighting matrix (all coefficients after the current coefficient are quantized with d');
    if (rdcost < min_rdcost)
    {
      Set d'' = d';
      find_new = true;
      min_rdcost = rdcost;
    }
  }
  if (find_new)
    d = d''; //the best dead-zone is found for the current coefficient
}
```

The complexity of the above algorithm is too complex for the real time encoder. Based on the idea, two experiential dead-zone matrices are found and used in

our experiments:

$$f = \begin{bmatrix} 1.0/2.0 & 3.0/7.0 & 2.0/5.0 & 1.0/3.0 \\ 3.0/7.0 & 3.0/7.0 & 1.0/3.0 & 1.0/4.0 \\ 2.0/5.0 & 1.0/3.0 & 1.0/5.0 & 1.0/5.0 \\ 1.0/3.0 & 1.0/4.0 & 1.0/5.0 & 1.0/5.0 \end{bmatrix}$$

for intra macroblocks and

$$f = \begin{bmatrix} 1.0/3.0 & 2.0/7.0 & 2.0/7.0 & 2.0/9.0 \\ 2.0/7.0 & 4.0/15.0 & 2.0/9.0 & 1.0/6.0 \\ 2.0/7.0 & 2.0/9.0 & 1.0/7.0 & 1.0/7.0 \\ 2.0/9.0 & 1.0/6.0 & 1.0/7.0 & 2.0/15.0 \end{bmatrix}$$

for inter macroblocks. In a word, f is selected with a basic rule that the dead-zone length for low frequency coefficients is larger than that for high frequency coefficients, and that the dead-zone for those coefficients which are scaled more on encoder is larger than that for other coefficients.

4 Experimental Results

The proposed transform has ever been proposed to AVS, and it has been adopted by AVS1-P7. Test conditions are shown in Table 7. Fig.3 shows the performance of proposed transform on the AVS platform when it was submitted to AVS[18]. At that time, the proposed transform for AVS1-P7 can share quantization/dequantization table with AVS1-P2 compared with the original transform in AVS1-P7 anchor.

Table 7. Test Conditions for Proposed 4×4 Transform

Tools	Option
Multi reference frame	2 frames
Variable block size MC	$16 \times 16 - 4 \times 4$
RDO	on
Entropy coding	C2DVLC

The proposed transform is also implemented on the JVT reference software JM7.3. The test conditions are the same as that defined in Table 7 except that the entropy coding scheme is CAVLC. Table 8 shows the experimental results on several typical test sequences. From the table we can see that the proposed transform has similar performance compared with the transform in H.264/MPEG-4 AVC. With the adaptive dead-zone quantization scheme, the proposed transform can gain 0.09dB over JM7.3. Fig.4 further shows the results.

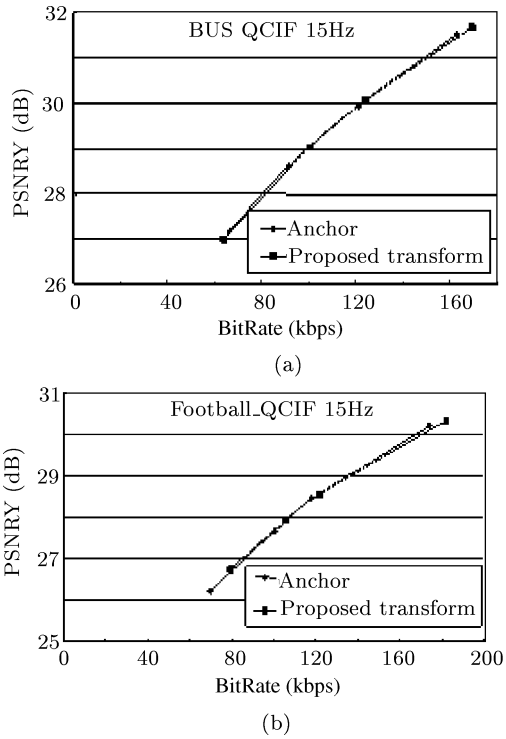

Fig.3. Proposed transform on AVS1-P7. (a) Bus. (b) Football.

Fig.5 shows the visual quality comparison between the original JM7.3 and the JM7.3 with the proposed transform/adaptive quantization scheme. As the adaptive dead-zone quantization scheme can reduce partial high frequency information, the proposed scheme gives better visual quality with less block artifacts.

Table 8. Experimental Results on Test Sequences

Test Sequences	Frame Rate	QP	JM7.3		Proposed Transform		Proposed Transform + Adaptive Deadzone	
			PSNR (dB)	Bitrate (kbps)	PSNR (dB)	Bitrate (kbps)	PSNR (dB)	Bitrate (kbps)
Foreman	30 QCIF	28	35.50	111.24	35.17	102.03	35.613	110.96
		32	32.78	57.76	32.54	55.07	32.95	58.36
		36	30.37	33.76	30.24	32.88	30.65	34.98
		40	28.04	21.63	27.84	21.14	28.31	22.33
Mobile	30 QCIF	28	32.56	398.90	32.55	402.75	33.10	434.52
		32	28.99	197.75	28.97	199.50	29.49	214.92
		36	26.00	95.09	26.00	96.83	26.45	103.06
		40	23.28	47.49	23.27	47.94	23.7	51.45
Bus	30 QCIF	28	33.93	355.09	33.37	324.42	34.00	351.48
		32	30.68	198.20	30.19	180.59	30.78	197.67
		36	27.77	106.00	27.49	100.14	28.09	111.88
		40	25.21	56.91	24.93	53.39	25.43	59.58
Football	30 QCIF	28	35.21	513.37	34.74	479.39	35.30	511.54
		32	32.37	315.12	31.96	293.12	32.46	317.12
		36	29.85	185.66	29.62	176.52	30.07	193.98
		40	27.46	107.58	27.24	100.63	27.67	112.23
Average Gain over JM7.3					−0.030dB		0.092dB	

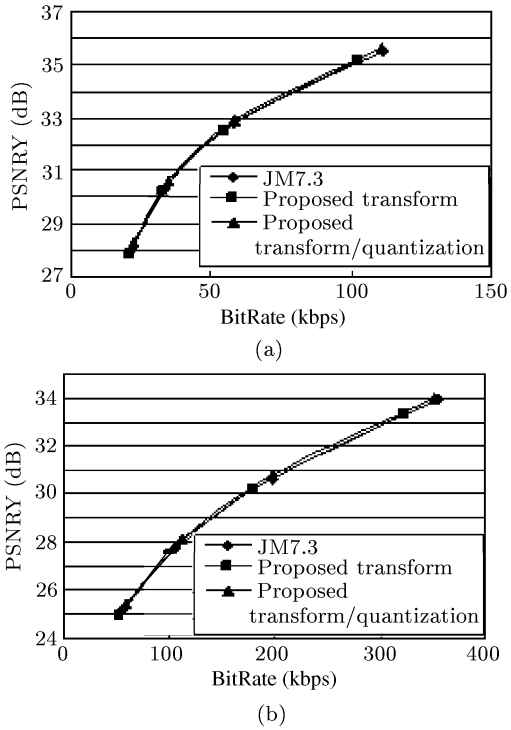

Fig.4. PSNR curve of JM7.3 and JM7.3 with proposed transform/quantization scheme. (a) Foreman. (b) Bus.

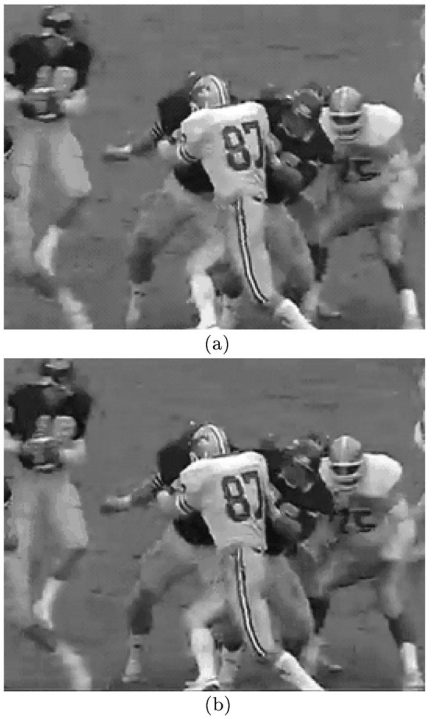

Fig.5. Visual comparison between JM7.3 and JM7.3 with proposed transform/quantization scheme. (a) Original JM7.3. (b) JM7.3 with proposed transform/quantization.

5 Conclusions

In this paper, a low complexity integer transform is proposed and it has been adopted by AVS1-P7. The proposed transform can share the same quantization/dequantization table with AVS1-P2. As the bases of the transform coefficients are very close, the transform normalization can be implemented only on the encoder side and the dequantization table size can be reduced. Along with the feature of the proposed transform, adaptive dead-zone quantization optimization for the proposed transform is studied. Experimental results show that the coding performance of the proposed transform is similar to that of the transform in H.264/MPEG-4 AVC, and would gain nearly 0.1dB with the adaptive dead-zone quantization scheme.

Acknowledgements The authors would like to thank the editors and the anonymous reviewers for their invaluable comments and suggestions.

References

[1] Tran T D. The BinDCT: Fast multiplierless approximation of the DCT. *IEEE Signal Processing Letters,* 2000, 7(6): 141–144.

[2] Chen Y J, Oraintara S, Tran T D *et al.* Multiplierless approximation of transforms with adder constraint. *IEEE Signal Processing Letters,* 2002, 9(11): 344–347.

[3] Chen Y J, Oraintara S, Nguyen T Q. Video compression using integer DCT. In *Proc. Int. Conf. Image Processing,* Vancouver, Canada, September 10–13, 2000, pp.844–845.

[4] Bjontegaard G. Coding improvement by using 4 × 4 blocks for motion vectors and transform. ftp://ftp3.itu.int/video-site/9712_Eib/q15c23.doc.

[5] Wien M, Sun S J. ICT comparison for adaptive block transform. ftp://ftp3.itu.int/video-site/0101_Eib/ VCEG-L12.doc.

[6] Hallapuro A, Karczewicz M, Malvar H. Low complexity transform and quantization—Part I: Basic implementation. ftp://ftp3.itu.int/video-site/0201_Gen/JVT-B038.doc.

[7] Hallapuro A, Karczewicz M, Malvar H. Low complexity transform and quantization—Part II: Extensions. ftp://ftp3.itu.int/video-site/0201_Gen JVT-B039.doc.

[8] Wien M. Clean-up and improved design consistency for ABT. ftp://ftp3.itu.int/jvt-site/2002_10_Geneva/JVT-E025.doc.

[9] Malvar H S. Low-complexity length-4 transform and quantization with 16-bit arithmetic. ftp://ftp.imtc-files.org/jvt-experts/0109_San/VCEG-N44.doc.

[10] Gordon S, Marpe D, Wiegand T. Simplified use of 8 × 8 transforms—proposal. ftp://ftp3.itu.int/jvt-site/2003_12_Waikoloa/JVT-J029.doc.

[11] Gordon S, Marpe D, Wiegand T. Simplified use of 8 × 8 transforms—updated proposal and results. ftp://ftp3.itu.int/jvt-site/2004_03_Munich/JVT-K028.doc.

[12] Zhang C X, Lou J, Yu L *et al.* The techniques of pre-scaled integer transform. In *Proc. Int. Symp. Circuits and Systems,* Kobe, Japan. May 23–26, 2005, pp.316–319.

[13] Lou J, Dong J, Yu L. 8 × 8 integer transform. ftp://159.226.42.57/pulic/AVS_ M1182.doc.

[14] Wiegand T. H.26L Test Model Long-Term Number 9 (TML-9) draft0. ftp://ftp.imtc-files.org/0109_San/VCEG-N83 d1.doc.

[15] Wien M, Sun S J. ICT comparison for adaptive block transforms. ftp://ftp.imtc-files.org/jvt-experts/0109_San/VCEG-L12.doc.

[16] Tourapis A M, Boyce J. Performance evaluation of the 8 × 8 transform vs. coefficient adaptive deadzone. ftp://ftp.imtc-files.org/jvt-experts/0403_Munich/JVT-K035.doc.

[17] Sullivan G. Adaptive quantization encoding technique using an equal expected-value rule. ftp://ftp3.itu.int/jvt-site/2005_01_HongKong/JVT-N011.doc.

[18] AVS FTP Site. ftp://159.226.42.57.