H.264-BASED LOSSLESS VIDEO CODING USING ADAPTIVE TRANSFORMS

Seishi Takamura and Yoshiyuki Yashima

NTT Cyber Space Laboratories, NTT Corporation,
Y517A, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, 239-0847 Japan,
e-mail: {takamura.seishi, yashima.yoshiyuki}@lab.ntt.co.jp

ABSTRACT

In this paper we propose a reversible video coding method that combines adaptive transform with H.264 tools. We extensively compare its lossless coding performance against three different reversible transforms and Motion JPEG 2000. Experimental results show that for I picture coding, the proposed method performed slightly worse (2.0% lower compression ratio on average) than Motion JPEG 2000 while outperforming the new H.264 FREXt standard (9.4 to 14%). For B and P pictures, our method offered the best performance with 0.3 to 6.2% gain over FREXt. Our method requires minimal modification to H.264 software and provides better performance than other existing methods.

1. INTRODUCTION

Lossless video coding is important in many application areas such as source distribution, digital cinema, medical imaging. Although there are lossless still image coding standards such as JPEG-LS and (Motion)JPEG 2000, they do not perform motion compensation and their performance is limited.

Among the *lossy* video coding standards, H.264[1] offers quite good coding efficiency. This is partly due to its powerful spatiotemporal prediction techniques such as flexible block size motion compensation, multiple reference frames and adaptive intra prediction.

H.264's recent Fidelity Range Extension (FREXt) transforms the residual in a verbatim manner to enable lossless coding. In terms of the transform approach, there are several reversible schemes in the literature. This paper first grafts these transforms onto H.264 and evaluated the resulting performance. We propose another reversible transform and implement it for lossless video coding. We also propose an adaptive version. We compare its performance to those of other reversible transforms, as well as with Motion JPEG 2000.

2. REVERSIBLE TRANSFORM METHODS

In this section we deal with conventional reversible 1x4-dimensional transforms. Given a 4x4 image data, the 1x4 transform is applied four times for each horizontal and vertical direction to obtain 4x4 coefficients.

Quite a few reversible transform methods are proposed in the literature. In this paper we implement and test some of these proposals as well as our new method.

2.1. Integer transform of H.264 without quantization

H.264 uses a relatively simple DCT-like integer transform expressed by the following matrix D.

$$D = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \quad (1)$$

With the plain H.264 decoder this is not reversible since it has a dequantization process and an inverse transformation process with right bit shifts. However, the above transofrm is mathematically reversible if the quantization process is not carried out.

Note that its determinant is much larger than one ($\det(D) = 40$). This means that the output coefficient vector is 40 times larger than the input signal vector, i.e., the transformed domain is 40 times coarser than the pixel value domain. This magnification is for only one row or one column of the block, and in 4x4 transform, it is multiplied eight times (four rows and four columns). Therefore the output 4x4 coefficient block is $40^8 \simeq 6.6 \times 10^{12}$ times coarser than the original pixel value block.

2.2. Method 1: Verbatim signal (H.264 FREXt)

H.264 FREXt adopts the technique of skipping the application of both transform and quantization processes to the residual signals and directly passing them to the entropy coder. As H.264's intra-/inter-predictions are all integer-based, the decoded image is identical to the original. In this case, the transform matrix equals the identity matrix, and has the determinant of one. However, decorrelation among the residual signals is never carried out. This may result in inefficiency given the existence of correlated residuals.

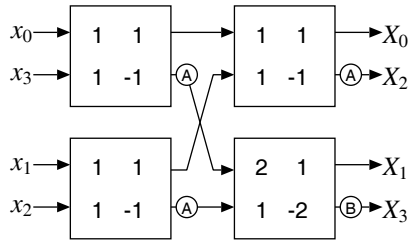

Fig. 1. Block diagram of integer transform. For H.264 transform, nodes Ⓐ and Ⓑ do nothing. For method 2 transform, they down-scale and round to integers.

2.3. Method 2: Piecewise scaled transform

The transform of H.264 is described by the block diagram in Fig. 1. The transform is decomposed into four transforms, expressed by sub blocks, as shown in the figure. For H.264 transform, nodes Ⓐ and Ⓑ make no operation. In each sub-block, output signals are redundant. For example, instead of $X_0 = x_0 + x_1$ and $X_1 = x_0 - x_1$,

$$X_0 = x_0 + x_1, \qquad X_1 = R((x_0 - x_1)/2),$$
$$x'_0 = X_1 + \lfloor X_0/2 \rfloor, \quad x'_1 = X_0 - x'_0,$$

where $R(x) = \lfloor x + 0.5 \rfloor$ retrieves the original signals, i.e., $x_0 = x'_0$ and $x_1 = x'_1$. Likewise, $X_0 = 2x_0 + x_1$ and $X_1 = x_0 - 2x_1$ have redundancy because

$$X_0 = 2x_0 + x_1, \quad X_1 = R((x_0 - 2x_1)/5),$$
$$x'_0 = X_1 + (2X_0 - \mathrm{offset}[X_0 \bmod 5])/5, \quad x'_1 = X_0 - 2x'_0,$$

where "offset" is an array with five elements, $\{0, 2, -1, 1, -2\}$, that retrieves the original values. Mapping of $(x_0, ..., x_3)$ onto $(X_0, ..., X_3)$ is as follows:

$$\begin{aligned} X_0 &= x_0 + x_1 + x_2 + x_3 \\ X_1 &= 2R((x_0 - x_3)/2) + R((x_1 - x_2)/2) \\ X_2 &= R((x_0 - x_1 - x_2 + x_3)/2) \\ X_3 &= R((R((x_0 - x_3)/2) - 2R((x_1 - x_2)/2))/5) \end{aligned}$$

This is equivalent to down-scaling by 1/2 and 1/5 and rounding to integers at nodes Ⓐ and Ⓑ in Fig. 1, respectively. The transform matrix approximately equals the following with determinant of one.

$$D_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1/2 & -1/2 & -1/2 & 1/2 \\ 1/10 & -1/5 & 1/5 & -1/10 \end{pmatrix} \qquad (2)$$

2.4. Method 3: Reversible DCT

Komatsu et al. proposed a type of reversible DCT[2]. Its 1x4 dimensional case is as follows:

$$\begin{aligned} X_0 &= R((x_0 + x_1 + x_2 + x_3)/4) \\ X_1 &= x_0 - x_3 + R((x_1 - x_2)/C) \\ X_2 &= R((x_0 - x_1 - x_2 + x_3)/2) \\ X_3 &= R((x_0 - x_3)/C) - x_1 + x_2, \end{aligned}$$

where C is an arbitrary positive real number. This gives this transform more flexibility than other integer-based transforms. Although it uses real numbers, the transform is reversible. When $C = 2$, the transform parallels H.264's transform. In the literature, disregarding the complexity, it is suggested to use $C = \cos(\pi/8) \,/\, \cos(3\pi/8) = 1+\sqrt{2}$ to make the transform equivalent to mathematical DCT. In the experiment, we tested $C = 1+\sqrt{2}$, 3 and 3.5. The transform matrix approximately equals:

$$D_3 = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1 & 1/C & -1/C & -1 \\ 1/2 & -1/2 & -1/2 & 1/2 \\ 1/C & -1 & 1 & -1/C \end{pmatrix}. \qquad (3)$$

As $\det(D_3) = 1 + 1/C^2 > 1$, the transformed space is always slightly more expanded than the input space.

3. PROPOSED METHOD

3.1. Fixed version

We apply the following autoregressive transform:

$$\begin{aligned} X_0 &= x_0 \\ X_1 &= x_1 - R(\alpha x_0) \\ X_2 &= x_2 - R(\alpha x_1) \\ X_3 &= x_3 - R(\alpha x_2) \end{aligned}$$

where α is a constant. This is identical to Method 1 when $\alpha = 0$. As block boundaries usually have signal discontinuity, we do not use inter-block prediction. The transform matrix approximately equals:

$$D_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 \\ 0 & -\alpha & 1 & 0 \\ 0 & 0 & -\alpha & 1 \end{pmatrix}. \qquad (4)$$

As $\det(D_4) = 1$, the signal is not expanded at all after the transform. The calculation cost for this transform is as small as that of method 1, and much smaller those of than methods 2, 3, and "no quantization".

We apply this transform in both horizontal and vertical directions. The horizontal and vertical values of α do not need to be the same. This is not orthogonal, however it works better than other orthogonal transforms such as methods 1 to 3 as will be shown in the experimental results.

We tuned the parameter α to maximize the average compression ratio. We found $\alpha = 0.7$ for I and $\alpha = 0.4$ for P and B work well, and used them in the experiments.

3.2. Adaptive version

The optimal value of α in Eq. 4 in terms of energy compaction is given as the correlation coefficient value among residual signals x of the current block. This exact value cannot be known since the signal has yet to be decoded. Sending α as side information for each 4x4 block may greatly increase the bit-rate. However, we can estimate its appropriate value from adjacent correlation values.

Table 1. Lossless compression ratios for I pictures (*QCIF, **CIF, MJ2k=Motion JPEG 2000)

sequence	no quantization	method 1 (FRExt)	method 2 (piecewise)	method 3			proposed method		I-only methods	
				C=1+ $\sqrt{2}$	C = 3	C = 3.5	fixed	adaptive	Lee[3]	MJ2k
container*	1.122	1.904	1.902	1.999	2.011	2.015	2.096	2.120	**2.168**	2.111
silent*	1.045	1.691	1.719	1.840	1.846	1.846	1.955	1.948	1.946	**1.981**
news*	1.127	1.846	1.899	1.998	2.008	2.008	2.120	2.124	**2.183**	2.126
stefan*	0.891	1.444	1.440	1.523	1.529	1.531	1.605	1.614	1.613	**1.675**
coast**	1.043	1.738	1.723	1.847	1.852	1.852	1.947	1.963	1.953	**2.061**
foreman**	1.182	2.059	2.032	2.121	2.132	2.134	2.250	2.272	**2.311**	2.265
silent**	1.090	1.841	1.828	1.936	1.946	1.948	2.042	2.050	2.059	**2.088**
paris**	1.007	1.627	1.659	1.734	1.746	1.750	1.825	1.842	**1.894**	1.877
mobile**	0.838	1.327	1.324	1.384	1.387	1.387	1.458	1.463	1.487	**1.571**
Average	1.038	1.720	1.725	1.820	1.829	1.830	1.922	1.933	1.957	**1.973**

Our strategy is, if the horizontal correlation seems stronger than the vertical correlation then the left adjacent correlation value is weighted more than other locations to yield α, and vice versa. If little correlation among adjacent blocks is observed, the default value is used. Our actual estimation method is as follows:

1) hdif = $|\rho_H^{NW} - \rho_H^N| + |\rho_V^{NW} - \rho_V^N|$
2) vdif = $|\rho_H^{NW} - \rho_H^W| + |\rho_V^{NW} - \rho_V^W|$
3) if (hdif < 0.6 vdif) then // *horizontal correlation*
4) $\alpha_H = 0.4\rho_H^W + 0.3\rho_H^N + 0.1\rho_H^{NW} + 0.2\alpha_0$
5) $\alpha_V = 0.4\rho_V^W + 0.3\rho_V^N + 0.1\rho_V^{NW} + 0.2\alpha_0$
6) else if (vdif < 0.6 hdif) then // *vertical correlation*
7) $\alpha_H = 0.3\rho_H^W + 0.4\rho_H^N + 0.1\rho_H^{NW} + 0.2\alpha_0$
8) $\alpha_V = 0.3\rho_V^W + 0.4\rho_V^N + 0.1\rho_V^{NW} + 0.2\alpha_0$
9) else // *little correlation*
10) $\alpha_H = \alpha_V = \alpha_0$

where ρ means the correlation coefficient value of the corresponding block, superscript N denotes north block, W west, NW northwest; subscript H denotes horizontal, and V vertical direction. We let $\alpha_0 = 0.7$ for I and $\alpha_0 = 0.4$ for P and B pictures.

4. EXPERIMENTAL RESULTS

For the experiment we used H.264 reference software JM8.2[4] as the base. The following modifications were made:

- ◇ Skipped quantization/dequantization process.
- ◇ Replaced transform codes with reversible transform.
- ◇ Disabled the deblocking loop filter.

4.1. Video coding conditions

We encoded four QCIF and five CIF sequences of 30 frames/sec. Coding conditions were as follows:

- ◇ Entropy coding = CABAC
- ◇ R-D Optimization enabled
- ◇ Number of reference frames = 5
- ◇ Max search range = 16

For I coding, first 100 frames were intra-encoded, and we tried two other intra-compression approaches. One, proposed by Lee et al.[3], changes the semantics of H.264's intra prediction. It successively uses lossless decoded pixels during intra prediction. This is only applicable for Intra slices. The other approach is Motion JPEG 2000 (kdu_v_compress[5]). For fair comparison, we tuned the coding conditions to maximize the compression ratio as follows:

- ◇ Color transform suppressed (Cycc=no)
- ◇ Decomposition level (Clevels) = 3 for CIF, 2 for QCIF (default=5)
- ◇ Reversible mode enabled (Creversible=yes)

For P coding, we encoded 100 pictures in IPP$\cdots$ structure. The compression ratio was calculated from just 99 P pictures. For B coding, we encoded 298 pictures in IBBP$\cdots$ structure, including one I picture, 99 P pictures and the remaining 198 B pictures. The compression ratio was calculated from these 198 B pictures.

4.2. Discussion

Compression ratios calculated from the total bit amount are shown in Tables 1, 2 and 3. "no quantization" corresponds to the method in subsection 2.1.

It is observed that the "no quantization" method provides poor compression (sometimes even expansion for I pictures).

For I picture coding, the proposed method performed slightly worse (2.0% on average) than Motion JPEG 2000 while outperforming all the conventional methods (by 9.4 to 14% gain to FRExt). The proposed method outperformed Motion JPEG 2000 for container and foreman. It is also observed that Lee's method offers considerably improved intra coding efficiency.

Table 2. Lossless compression ratios for P pictures (*QCIF, **CIF)

sequence	no quantization	method 1 (FRExt)	method 2 (piecewise)	method 3			proposed method	
				C=1+$\sqrt{2}$	C = 3	C = 3.5	fixed	adaptive
container*	1.729	4.472	3.635	3.840	3.854	3.861	4.412	**4.487**
silent*	1.731	4.334	3.588	3.890	3.899	3.920	**4.441**	4.419
news*	2.455	5.909	4.963	5.267	5.287	5.312	5.978	**6.009**
stefan*	1.200	2.312	2.151	2.188	2.203	2.215	2.351	**2.367**
coast**	1.224	2.289	2.145	2.241	2.249	2.286	2.423	**2.442**
foreman**	1.415	2.857	2.617	2.680	2.698	2.751	2.938	**2.964**
silent**	1.580	3.542	3.076	3.204	3.219	3.292	3.622	**3.652**
paris**	1.466	3.171	2.777	2.836	2.855	2.916	3.194	**3.249**
mobile**	1.148	2.124	1.989	2.010	2.023	2.064	2.178	**2.196**
Average	1.550	3.446	2.993	3.128	3.143	3.180	3.504	**3.532**

Table 3. Lossless compression ratios for B pictures (*QCIF, **CIF)

sequence	no quantization	method 1 (FRExt)	method 2 (piecewise)	method 3			proposed method	
				C=1+$\sqrt{2}$	C = 3	C = 3.5	fixed	adaptive
container*	1.757	4.612	3.736	4.068	4.080	4.155	4.651	**4.692**
silent*	1.726	4.288	3.593	3.933	3.941	4.035	**4.511**	4.494
news*	2.520	6.164	5.152	5.536	5.555	5.698	6.343	**6.368**
stefan*	1.191	2.279	2.128	2.174	2.189	2.229	2.351	**2.370**
coast**	1.286	2.492	2.320	2.429	2.438	2.475	2.631	**2.641**
foreman**	1.399	2.841	2.620	2.693	2.711	2.753	2.916	**2.952**
silent**	1.590	3.632	3.162	3.343	3.358	3.408	3.706	**3.735**
paris**	1.521	3.465	2.988	3.106	3.125	3.162	3.436	**3.501**
mobile**	1.188	2.250	2.113	2.139	2.152	2.185	2.296	**2.326**
Average	1.575	3.558	3.090	3.269	3.283	3.344	3.649	**3.675**

Thanks to motion compensation (temporal decorrelation), P, B pictures can be compressed to about half the amount of I pictures. Our method offered the best performance with 0.3 to 6.2% gain over FRExt.

The proposed adaptive method is always better than the fixed method except for silent(QCIF). This is observed in all I, P, and B pictures. This may be because silent(QCIF) has little inter-block correlation which made the adaptive prediction less effective.

5. CONCLUSION

In this paper, we proposed a lossless video coding scheme that uses a reversible transform procedure and compared it against three conventional methods. We investigated the characteristics of each transform.

We also provided practical, good parameter sets and an adaptive parameter adjustment scheme for the proposed method. Even though it is simple, our method outperformed other reversible transforms in all picture types. As for I picture, the proposed adaptive method almost matched the performance of Motion JPEG 2000.

Future work includes developing a method that better decorrelates the residual signal to provide better compression performance.

6. REFERENCES

[1] ISO/IEC 14496-10:2003, *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*, International Standard, Dec. 2003.

[2] Kunitoshi Komatsu and Kaoru Sezaki, "Reversible transform coding of images," *IEICE Trans. Fundamentals*, vol. J79-A, no. 4, pp. 981–990, Apr. 1996.

[3] Yung-Lyul Lee and Ki-Hun Han, "Lossless coding for professional extensions," JVT of ISO/IEC MPEG & ITU-T VCEG, JVT-L017, July 2004.

[4] "JM reference software version 8.2," May 2004, http://bs.hhi.de/˜suehring/tml/.

[5] "Kakadu software version 4.2 linux executable," Feb. 2004, http://www.kakadusoftware.com.