Three-Dimensional Transforms and Entropy Coders for a Fast Embedded Color Video Codec

Vanessa Testoni and Max H. M. Costa
School of Electrical and Computer Engineering - State University of Campinas (UNICAMP)
(vtestoni,max}@decom.fee.unicamp.br

Abstract

This work compares the performances of two fast 3-D transforms and two adaptive Golomb entropy coders applied to a video codec system named FEVC (Fast Embedded Video Codec). The compared transforms are Hadamard (4x4x4 and 8x8x8) and H.264/AVC integer DCT (4x4x4). The compared adaptive Golomb entropy coders have different operation modes and adaptation strategies. New 3-D implementation methods for the transforms are presented. After the scan procedure, the encoding of the 3-D coefficients is done, bit-plane-by-bit-plane, by the entropy coders, producing a fully embedded output bitstream. The FEVC (also described here) was developed to be implemented each of a large number of set-top boxes used in a fiber optics network. For that reason, it is focused on reduced complexity and execution time, not on high compression rates. The use of meager computational resources is also required. Even with these constraints, good distortion versus rate results were achieved.

1. Introduction

The comparisons between the two fast 3-D transforms [1] [2] and between the two Golomb entropy coders [3] [4] presented here are studies to improve the performance of a color video codec named Fast Embedded Video Codec (FEVC). The new transform implementation method presented here is also applied to the codec. The FEVC was developed in C# language to be executed in a set-top box device under development. A large number of these set-top boxes will be the interface between a fiber optics network and its users. This device will receive digital signals, extract audio, video and data information and send the processed information to an output device. Among other functions, such as Internet accessibility and voice over IP, the set-top box will be able to receive and transmit video signals coming from, for example, video on demand and video conference applications.

Research on video coding systems typically looks for techniques that can reach the highest possible compression rate while not exceeding a given level of distortion. This compression rate increase is generally achieved by means of increased coding complexity, which is supported by the availability of increasing computational power. However, in some video coding applications, the use of high capacity processors is not the most convenient choice. These situations require video codecs focused on reduced execution times and reduced computational complexity, and less concerned with high compression performance. This is the profile of the FEVC. Also, in some cases, the codecs are to be implemented by software only, as hardware implementations may not be admitted.

In order to reduce the codec execution times, the very simple Hadamard (8x8x8 and 4x4x4) transform and the H.264/AVC integer DCT-like (4x4x4) transform are used instead of the traditional DCT. These transforms were chosen because they are able to reduce the correlation between coefficients and their implementations require only additions and bit shifts. To further reduce execution times, motion estimation (ME) and compensation (MC) techniques are avoided. This are high performance techniques but time consuming. Instead, 3D transforms are used to reduce correlation in both spatial and temporal dimensions.

After transforming 3-D blocks of pixels, the codec reads and reorders each coefficients block. It was found that the probability distribution of the dominant AC coefficients is spread along the major axes of the 3-D cubes, just as found for 3-D DCT cubes [5] [6]. It was also found that the cube energy is concentrated according to the coefficient *sequency number*, a concept related to the notion of frequency, in the three dimensions. To benefit from this energy distribution pattern, a scan order [7] based on the multiplication of the three *sequency numbers* of each coefficient is

adopted for the coefficient reading. This deterministic scan order gives generally better results than the traditional 3-D zig-zag scan.

The codec encodes the resulting reordered coefficients in a bit-plane-by-bit-plane fashion, refining their precision at each turn. This process renders a completely embedded encoded video bitstream. The encoding of each bit plane of the 3-D Hadamard coefficients is accomplished using an adaptive version of Golomb run length encoder. Two Golomb entropy coders [3] [4] were considered and the results obtained are presented here.

The entire implementation is designed to perform only fast mathematical operations and to require little computational memory. All multiplications are done in powers of two and performed by variable bit shifts. Moreover, the system is completely implemented using 16-bit integer arithmetic.

An overview of the codec stages is provided in Section 2. Specifically, the fast 3-D transform implementations are presented in Section 2.2, the 3-D coefficients scan order is presented in Section 2.3 and the entropy coders are described in Section 2.4. Section 3 presents the obtained results.

2. Video codec overview

The FEVC structure is shown in Figure 1. The video codec stages are described in this section in the order they appear in the figure.

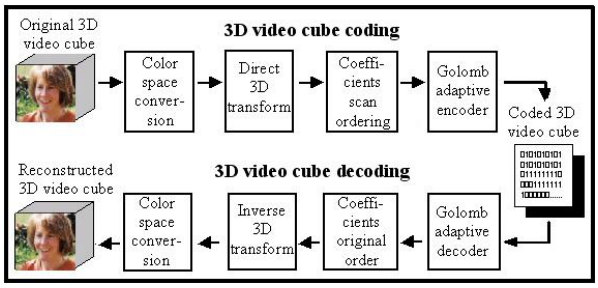


Figure 1. Block diagram of the FEVC structure.

2.1. Video codec color spaces

The FEVC is able to read color video sequences stored in tri-stimulus color spaces, such as RGB and YUV 4:2:0. Each color signal is separately encoded and the allowed pixel bit-rate is divided among the color signals according to its significance. Then, for the YUV 4:2:0 sampling, approximately 10% of the luminance signal rate is spent on the chrominance signals. This simple weighted bit-rate division procedure helps achieving higher compression rates.

In order to get the well-known advantages of the L-C (*Luminance* - *Chrominance*) formats, the FEVC offers the possibility to convert an original RGB video sequence to a different internal color space (such as YUV 4:2:0 and YC_oC_g) before beginning the coding process. Other color spaces are also supported and the conversions among them are described in [8].

2.2. Three-dimensional fast transforms

Two fast transforms are supported in the FEVC: the Hadamard transform and the H.264/AVC integer DCT-like transform. These transforms are applied in a three-dimensional fashion. The video sequence being encoded is partitioned into cubes and the transform is separately applied to each cube dimension per dimension (first to columns, then to lines and finally to frames).

To evaluate the cube's size effect in coding performance, the FEVC is executed with cubes of 4x4x4 or 8x8x8 sizes for the Hadamard transform and only with cubes of 4x4x4 size for the H.264/AVC integer DCT-like transform. The 8x8x8 cube was not implemented for this transform because of its matrix higher complexity [9].

These transforms were chosen because they can be computed exactly in integer arithmetic, thus avoiding inverse transform mismatch problems. Furthermore, only additions and bit shifts are necessary, thus minimizing computational complexity.

Although the use of the 3-D Hadamard presents no innovation, this transform was chosen because it has the simplest basis functions (composed only of +1 and −1 elements), it is identical to its inverse, and it is easy to extend results to larger transforms. As an example, the NxN Hadamard matrix H_n (where $N = 2^n$, for some integer n) for N = 8 is

$$H_3 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} \textbf{0} \\ \textbf{7} \\ \textbf{3} \\ \textbf{4} \\ \textbf{1} \\ \textbf{6} \\ \textbf{2} \\ \textbf{5} \end{matrix} \quad (1)$$

Sequency

The Hadamard transform fast calculation method to be used in the FEVC is based on the fact that the H_n matrix can be written as a product of N sparse matrices $\tilde{H}$ [1]. In this method, the total number of operations is of the order of $N * log_2 N$.

The H.264/AVC integer DCT-like transform [2] is also a very simple transform as shown by its matrix H_2 in Eq. (2). The H_2 is not symmetric, and the inverse transform $\tilde{H}_{INV}$ is also shown in Eq. (2). The multiplications by ½ are implemented by 1-bit right shifts.

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad \tilde{H}_{INV} = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \quad (2)$$

Here $\tilde{H}_{INV}$ is related to the inverse of H, so that

$$\tilde{H}_{INV} * diag\left\{1/4, 1/5, 1/4, 1/5\right\} * H = I \quad (3)$$

2.2.1. Dynamic range gain. As the full implementation with 16-bit integer arithmetic is a FEVC prerequisite, the 3-D transform dynamic range gain was adjusted to avoid loss of data.

The Hadamard matrix is composed only of +1 and −1 values, and the one-dimensional transform has a dynamic range gain of $N / \sqrt{N} = \sqrt{N}$. If $N = 8$, for instance, the dynamic range gain is $\sqrt{8}$. This division by $\sqrt{N}$ (as in Eq. (1)) is done in each cube dimension to preserve the signal energy in the transform domain. For a three-dimensional transform, the total dynamic range gain is $8^3 / (\sqrt{8})^3 = 8 * \sqrt{8}$.

In order to avoid fractional coefficients (generated by the $\sqrt{N}$ divisions) and the square root operation, and to reduce the coefficient magnitudes on the three Hadamard transform calculations, we modified the calculation order in the FEVC [7]. In this modified implementation, the divisions are grouped in a special way, the first decoding division by $\sqrt{N}$ is carried out at the encoder and the total dynamic range gain becomes $8^3 / (\sqrt{8})^4 = 8$, requiring only 3 additional bits to store the transform coefficients than to store the pixel values. This analysis for $N = 8$ is sufficient for the Hadamard transform because the maximum supported cube size in the FEVC is 8x8x8.

For the integer DCT, the maximum sum of absolute values in any row of H_2 in Eq. (2) equals 6, so the maximum dynamic range gain increase for a 3-D transform is $log_2 (6^3) = 7.75$, requiring 8 additional bits to store the transform coefficients.

Because of these additional 8 bits in the coefficient values, the FEVC performance with the integer DCT was not as satisfactory when compared to the performance with the Hadamard transform. The reason was that as much as 16 bit planes could have to be entropy encoded with the integer DCT, while only a maximum of 11 bit planes have to be encoded in the Hadamard transform case.

In order to control the 3-D integer DCT dynamic range gain, a modified calculation was used, in which a scaling factor of ¼ was extracted from the inverse transform, as shown in Eq. (3). Then, since the inverse transform is applied three times, two of the scaling factors were grouped, and applied at the end of the coding process, as 4-bit right shifts. Finally, the last scaling factor was implemented at the end of the decoding process by 2-bit right shifts.

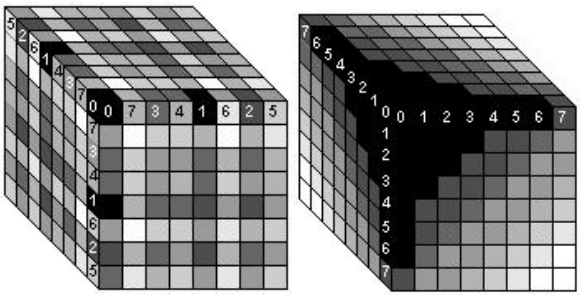
When the scaling factors were applied together at the end or during the coding/decoding process, the coefficient values became too small and lost precision. With the modified computation method, enough coefficient precision could be preserved and the number of additional bits needed to store the coefficient values was reduced by 4, resulting in a maximum of 12 bit planes being entropy encoded.

2.2.2. Transform coding gain. From a compression standpoint, for a stationary Gauss-Markov input with correlation coefficient ρ = 0.9, the one-dimensional transform coding gain of the DCT is 5.387 dB, of H_2 in Eq. (2) is 5.376 dB [2] and of H_3 in Eq. (1) is 5.034 dB. As the transforms are applied in three-dimensional fashion and, in practice, the empirical correlation coefficients tend to be in the neighborhood of 0.9, the Hadamard and integer DCT performances can be quite comparable. This can be seen in Section 3, where the results for both transforms with different video sequences are presented.

2.2.3. Energy concentration. The DCT provides better energy concentration than the Hadamard transform. This DCT advantage can also be perceived when the transforms are applied in cubes [5] [6], as depicted in Figure 2. The reason is that in the Hadamard matrix, the row vectors are not *sequency* ordered, as shown in Eq. (1) by the column named "*Sequency*". The *sequency* number is the number of transitions (zero crossings) in the Hadamard transform basis vector. This is similar to the concept of frequency, defined for sinusoidal signals in terms of zero crossings.

Thus, a special scan order, presented in Section 2.3, was developed for the cube reading. The Hadamard matrix cannot be reordered before the transformation process because, in this way, the fast implementation using the product of sparse matrices could not be applied. Therefore, for the Hadamard cube, the

sequency numbers are assigned to real cube coordinates during the scan process.



(a) (b)

Figure 2. Coefficients energy distribution in (a) Hadamard and in (b) DCT cubes.

2.3. Coefficients scan order

With the energy concentration achieved by the 3-D transform, the cube's energy is not any more spread among its values and becomes concentrated in some cube coefficients, as shown in Figure 2. The cube coefficient reading in a decreasing (or "decreasing in the average") order is important because it increases the entropy coding efficiency and produces a progressive (embedded) encoder.

To determine an efficient and fixed reading order (independent of the cube's information content), such that the coefficients with higher energy values tend to be scanned first, we take into account the three *sequency* numbers of the coefficients (one in each dimension), each incremented by one (to avoid the zeros). We order the product of the three incremented sequency numbers. Also, we explore the correlation between coefficients of adjacent cubes located in the same position through a *spiral curve* reading of coefficients. This scan method is presented in [7]. The multiplication values are obtained just once, at the beginning of the coding process (according to the cube size chosen), and associated with all the three incremented *sequency* numbers of possible combinations that multiply to the same value.

The product of incremented *sequency* numbers is motivated by the observation that the regions of equal coefficient energy tend to be shaped as hyperbolic surfaces [6].

The graph presented in Figure 3(a) corresponds to the "Hall Monitor" AC coefficients of the luminance frames in the range #264 to #271, read by a column-line-frame scan order. Comparing this sequence with the one in Figure 3(b), which corresponds to the FEVC scan order result, one can notice that a better grouping of AC coefficients with similar values is in fact achieved.

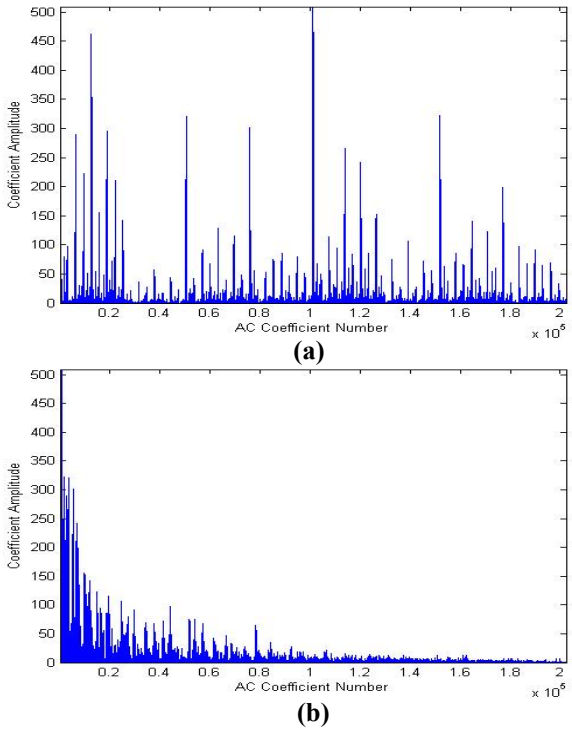


Figure 3. AC coefficients reading by (a) column-line-frame order and in (b) FEVC's scan order.

2.4. Entropy coding

Most video codecs perform quantization of the coefficient values before the entropy coding stage. The FEVC does not perform this explicit quantization and, thus, can be used in a lossless manner. In fact, the FEVC performs an implicit coefficient quantization because encoding is applied to bit planes, which generates an embedded and progressive encoded bitstream. Thus, decoding can be done aiming at a specific desired bit rate. Another possibility is to control the bit rate during the encoding process, generating the bit stream at the desired bit rate.

The entropy coding is performed in FEVC by two choices of Golomb run length encoders (RLE) with adaptive parameter adjustment [3] [4]. The results comparisons between these are given in Section 3. The first entropy coder considered [3] uses concepts extracted from wavelet scalable video compression [10] and the SPIHT encoder [11] and has a single operation mode. Letting M be the Golomb encoder run length parameter, the incomplete zero runs (runs of length M) are mapped to a one-bit codeword ("0") and the complete zero runs (runs of length $0 \le x \le M-1$) are mapped to codewords of length $\lfloor \log_2 M \rfloor$, when

$$x < 2^{\lfloor \log_2 M \rfloor + 1} - M, \qquad (4)$$

and to codewords of length $\lfloor \log_2 M \rfloor + 1$, otherwise.

The adaptation strategy presented in [3] adjusts the coder length in two situations:

- In the middle of a run of zeros, the length parameter M is incremented by $\lfloor (M+1)/2 \rfloor$.
- After the end of the run, a new estimate of the average run of zeros $\bar{X}$ is calculated and the parameter M is set to $\lfloor (\bar{X}+1)/2 \rfloor$. The new $\bar{X}$ is calculated by a first-order auto-regressive model given by

$$\bar{X}_{n+1} = \alpha \bar{X}_n + (1-\alpha) X_n, \quad (5)$$

where $\alpha \in [0,1]$ is the memory factor.

The second Golomb entropy coder considered [4] has four operation modes: a mode where the input symbol is passed uncoded to the output, a mode with a particular Huffman code, a mode with a Rice coder (as described in [3], with $M = 2^k$), and a "half-mode" code with $M = 3 \cdot 2^{k-1}$. This Golomb entropy coder has a complexity nearly identical to that of popular adaptive Rice coders. However, this encoder has an excess rate of less than 2% with respect to the source entropy for binary sources with unknown statistics. The adaptation is based on the last N previously encoded strings and is done by the simple rule

$$N \cdot \bar{X} \leftarrow \frac{N - n_1}{N} (N \cdot \bar{X} + n_0), \quad (6)$$

where n_0 and n_1 are, respectively, the number of "0" symbols and the number of "1" symbols, read from the input to produce the output codeword. This adaptation rule is nearly maximum likelihood (ML) and can be implemented using only additions and bit shifts.

3. Results

In order to achieve a multi-platform code, the codec computational system is implemented in C# language within the Microsoft Visual C# .NET environment. The encoding and decoding processes, as well as all other supported operations, are controlled by the user through graphical interfaces [7].

To evaluate the FEVC computational efficiency, the encoding and decoding times of some video sequences were measured. All execution times were obtained with a Pentium-4 3.20 GHz processor and 3GB of memory, running exclusively the codec.

For comparison, we used the H.264/AVC official reference software obtained in [12]. Although the codecs are different, this performance reference is interesting because it is the video codec with the best performance nowadays. It is important to emphasize that there are H.264/AVC optimized implementations that run much faster than the official reference software. We chose to use the official reference software because this is a publicly available implementation and is always enabled without restrictions. We note that the FEVC implementation is also not optimized for the hardware where it is being executed, since C# is interpreted and a compiled code version was not generated.

Most H.264/AVC parameters were set as "default", according to the software official manual developed by the Joint Video Team (JVT). The parameters not set as "default" are: Main profile, level 2.0, GOP of size 15, 5 reference frames, and CABAC entropy coding.

Fig. 4 presents the PSNR versus bit-rate curves obtained with H.264/AVC and FEVC for three QCIF video sequences in the YUV 4:2:0 format, with different motion and background characteristics. Approximately 300 frames of each sequence were used. One can notice that the Hadamard 8x8x8 transform has superior performance when compared to the integer DCT 4x4x4 for sequences with high spatial and temporal correlations, as the "Hall Monitor" sequence (Figure 4 (a)). For the "Akiyo" sequence (Figure 4(b)), which also presents high temporal redundancy but has more color variations, the transforms performances are more similar. The comparison between the 4x4x4 transforms shows that the integer DCT is slightly better than the Hadamard, which is expected due to the greater transform coding gains, as discussed in Section 2.2.2. The coding and decoding times shown in Table 1 indicate that the integer DCT 4x4x4 is faster than the Hadamard 4x4x4 transform. This difference is due to the DCT simpler scan order, since the transforms complexities are comparable. For the "Foreman" sequence, which has significant motion content, the transform results are very similar, as shown in Figure 4(c).

It is shown in Fig. 4, that the FEVC uses approximately 3 times the bit rate of H.264 for the luminance signals. This rate-distortion result can be justified in applications where high capacity is available (as in optical links). It is also noticed that this difference in codecs performance is inferior in the chrominance signals for sequences with less color variations ("Hall Monitor" and "Foreman").

Fig. 5 shows that the Golomb adaptive entropy coders [3] [4] described in Section 2.4 have similar performances when the parameters are well adjusted. The two entropy coders are only applied to the most significant bit plane of each coefficient because the other bits (after the first most significant bit "1") of each coefficient present an approximately uniform distribution, and are thus left uncoded.

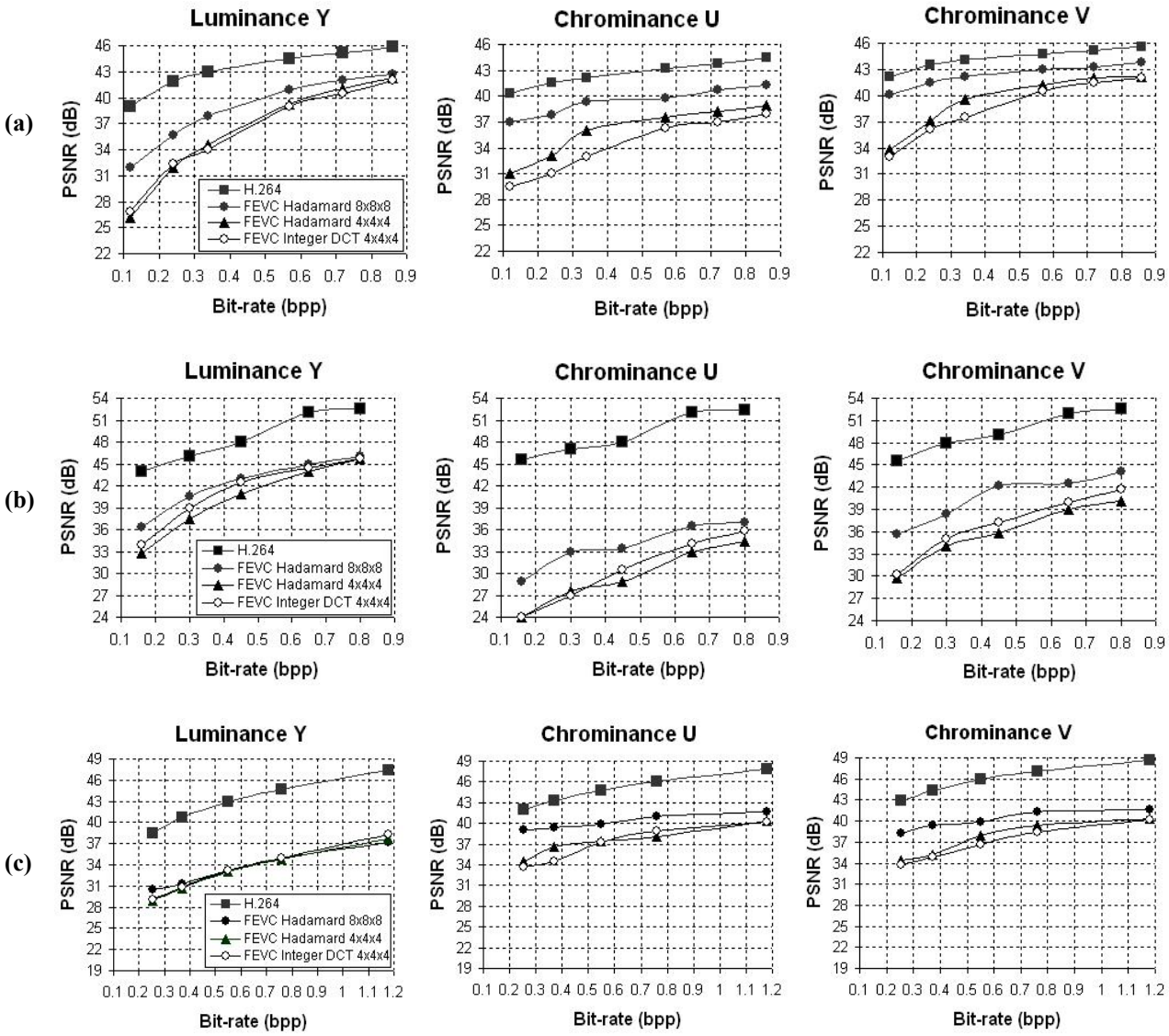


Figure 4. PSNR versus bit-rate curves for luminance and chrominance signals of the (a) "Hall Monitor" sequence, (b) "Akiyo" sequence, and (c) "Foreman" sequence.

Table 1. Encoding and decoding times for "Akiyo" sequence.

ENCODING				DECODING			
Bits per pixel	Time per frame (milliseconds)			Bits per pixel	Time per frame (milliseconds)		
	FEVC Hadamard 4x4x4	FEVC Integer DCT 4x4x4	H.264		FEVC Hadamard 4x4x4	FEVC Integer DCT 4x4x4	H.264
0.81	17.17	13.82	3453.69	0.81	15.67	15.62	130.62
0.65	16.28	12.91	3426.07	0.65	14.78	14.84	130.31
0.45	15.41	12.22	3357.16	0.45	13.91	13.95	128.99
0.31	16.67	11.31	3281.18	0.31	13.17	13.21	128.79
0.16	13.94	10.71	3134.21	0.16	12.44	12.45	126.97

In fact, when the entropy coders were also applied to the least significant bit planes, the differences in terms of encoding times and PSNR versus bit rate curves were negligible, as shown in Fig. 5. The entropy coder in [3] has a fast adaptation strategy that is well adjusted to non-stationary data, just as the bit planes values of the video data. It uses an empirical adaptation strategy. On the other hand, the entropy coder in [4] was originally designed to adapt to i.i.d. Bernoulli data with slowly varying statistics. By making the backward adaptation buffer size as small as N=2, the reasonable performance shown in Fig. 5 was achieved. This indicates that the speed of adaptation is more important than the precision of the data statistical analysis.

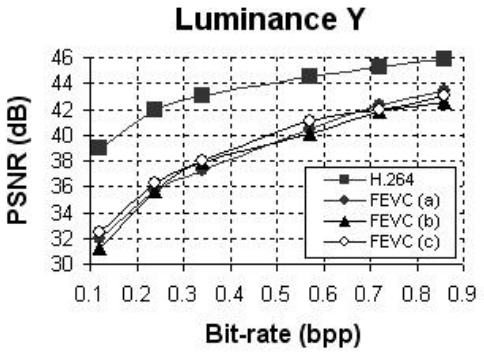


Figure 5. PSNR versus bit-rate curves of the luminance signal of "Hall Monitor" using (a) Golomb's RLE [3], (b) Golomb's RLE [3] with least significant bit planes coded and (c) Golomb's RLE [4] with least significant bit planes coded.

Visual quality comparisons with the H.264 standard are presented in Figs. 6, 7, and 8. As expected, due to the curves shown in Fig. 4, the lowest bit rates are obtained with the sequences "Hall Monitor" and "Akiyo", possibly because they have more spatial and temporal redundancies. Also, as expected from these curves, when compared at the same bit rate, the performance of the Hadamard 8x8x8 transform is very similar to the integer DCT 4x4x4 performance for the "Foreman" sequence (Figs. 8(c) and 8(d)), and slightly superior in the chrominance signals coding for the "Akiyo" sequence (Figs. 7(c) and 7(d), respectively).

We note that the FEVC visual performance can be considered satisfactory for highly compressed pictures. In Fig. 6(b), the bit rate is 0.12 bit/pixel (which implies a compression by a factor of 100). The same good performance is shown in Figs. 7(c) and 8(c).

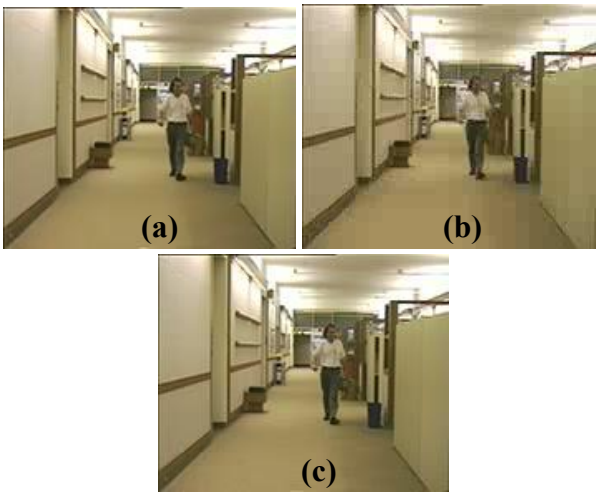


Figure 6. "Hall Monitor" frame #264 encoded by (a) H.264 at 0.12 bit/pixel and by FEVC (Hadamard 8x8x8) at (b) 0.12 bit/pixel, and at (c) 0.33 bit/pixel.

Figure 7. "Akiyo" frame #160 encoded by (a) H.264 at 0.16 bit/pixel, by FEVC (Hadamard 8x8x8) at (b) 0.16 bit/pixel and at (c) 0.30 bit/pixel, and by (d) FEVC (Integer DCT 4x4x4) at 0.30 bit/pixel.

Figure 8. "Foreman" frame #128 encoded by (a) H.264 at 0.25 bit/pixel, by FEVC (Hadamard 8x8x8) at (b) 0.25 bit/pixel and at (c) 0.55 bit/pixel, and by (d) FEVC (Integer DCT 4x4x4) at 0.55 bit/pixel.

Table 2. Encoding and decoding times for the "Hall Monitor" sequence with (b) and (c) as in Fig. 5.

ENCODING				DECODING			
Bits per pixel	Time per frame (milliseconds)			Bits per pixel	Time per frame (milliseconds)		
	FEVC (b)	FEVC (c)	H.264		FEVC (b)	FEVC (c)	H.264
0.86	19.69	78.92	3455.90	0.86	17.13	17.92	141.61
0.72	19.34	78.21	3446.74	0.72	16.39	17.43	141.55
0.57	18.59	77.77	3237.77	0.57	15.81	16.79	136.72
0.33	17.51	76.28	3148.40	0.33	14.89	15.62	132.86
0.24	17.24	75.74	3083.59	0.24	14.41	14.94	130.86
0.12	16.71	75.02	2855.75	0.12	13.81	14.32	126.07

The encoding and decoding times of H.264/AVC and FEVC (measured at the same bit rates) are shown in Table 2. Although the two FEVC entropy coders had similar performances, as shown in Fig. 5, Table 2 shows that the entropy coder in [3] is approximately 4 times faster than the entropy coder in [4]. The main bottleneck of the entropy coder in [4] is the frequent change of operation mode.

Based on Table 2, it is clear that the FEVC (b) is considerably faster than the H.264/AVC official reference software, being approximately 180 times faster in encoding and 8 times faster in decoding. As the FEVC is a symmetric codec, the encoding and decoding times are almost equal, unlike H.264/AVC, where the decoding is 23 times faster, in average.

The H.264/AVC codec requires 2,855.75 ms per frame for encoding at 0.12 bit/pixel. The FEVC requires 17.51 ms per frame for encoding at 0.33 bit/pixel, which produces frames with comparable visual quality to those of the H.264/AVC, as shown in Fig. 6(c). Thus, we can be conclude that, at the cost of reducing the H.264/AVC compression rate by a factor of about 2.75, a significantly faster encoding can be achieved with the FEVC. We also note that, with encoding times of 17.51 ms per frame, it is possible to have real time video sequences encoded by software with the FEVC, at 30 fps.

4. Conclusions

We presented a comparison of two fast three-dimensional transforms and two entropy coders applied to a codec named FEVC. New implementations were proposed in order to have 16-bit integer implementation and to ensure fast and simple operations (using only additions and bit shifts).

For high bit rate applications (around 0.9 bpp), the PSNR degradation with respect to H.264 is less pronounced (around 3 dB for sequences with high spatial and temporal correlations) than for low bit rate applications (around 0.1 bpp), where this degradation may be in excess of 6 dB.

The use of the FEVC is best directed to video streaming and video conferencing, and systems with complexity and storage limitations, possibly using fixed point processors, but enjoying high bit rate network connections (low cost codecs making use of high performance links). An added advantage is the exception of intellectual property restrictions.

The performance results for the video sequences shown indicates that, at the cost of a reduction in H.264/AVC compression rate by a factor of 2 to 3, it is possible to get encoding times that are significantly smaller (around 160 times) with the FEVC.

5. References

[1] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, USA, 1989.

[2] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity Transform and Quantization with 16-bit Arithmetic for H.26L", *Proceedings of the International Conference on Image Processing - ICIP*, pp. 489-492, 2002.

[3] F. C. Oliveira, and M. H. M. Costa, "Embedded DCT Image Encoding", *International Telecommunications Symposium – ITS-2002,* Natal, Brazil, Sept. 2002.

[4] M. H. M. Costa, and H. S. Malvar, "Efficient Run-Length Encoding of Binary Sources with Unknown Statistics", *Proceedings of the Data Compression Conference*, Snowbird, UT, USA, pp. 534-44, 2004.

[5] R. K. W. Chan and M. C. Lee, "3D-DCT Quantization as a Compression Technique for Video Sequences", *Proceedings of the International Conference On Virtual Systems And Multimedia*, Geneva, Switzerland, pp. 188-196, 1997.

[6] R. K. W. Chan and M. C. Lee, "Quantization of 3D-DCT Coefficients and Scan Order for Video Compression", *Journal of Visual Communication and Image Representation*, v. 8, n. 4, pp. 405-22, 1997.

[7] V. Testoni, and M. H. M. Costa, "3D-Hadamard Coefficients Sequency Scan Order for a Fast Embedded Color Video Coded", *Proceedings of the International Conference on Signal Processing and Communication Systems*, Gold Coast, Australia, pp. 75-82, 2007.

[8] G. Sullivan and S. Estrop, *Video Rendering with 8-bit YUV Formats*, Microsoft Digital Media Division, 2003.

[9] G. Sullivan, P. Topiwala, and A. Luthra, "The H.264/ AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", *Conference on Applications of Digital Image Processing*, 2004.

[10] K. Shen, and E. J. Delp. "Wavelet Based Rate Scalable Video Compression", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 9, n. 1, pp. 109 – 22, 1999.

[11] B. Kim, Z. Xiong, W. A. Pearlman, "Low Bit Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT)", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 10, n. 8, pp. 1374 – 87, 2000.

[12] *H.264/AVC reference software version JM 11.0***,** http://iphome.hhi.de/suehring/tml/. Downloaded in Dec. 2006.