3D-Hadamard Coefficients Sequency Scan Order 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), Campinas, SP, Brazil
{vtestoni, max}@decom.fee.unicamp.br

Abstract - **This work presents a new 3D-transform coefficients scan order based on the multiplication of the three dimensions sequency numbers of each coefficient. This scan process is developed to a fast embedded color video codec (also described in this article) to be executed in a set-top box on a fiber optics network. Due to its purpose, the codec is focused on reduced execution time, not on high compression rates. Low computational complexity and use of meager computational resources are also required. The Hadamard transform is used in a three-dimensional fashion, in order to avoid costly motion estimation and compensation techniques. The proposed scan procedure allows the coefficients reading in an idealistic "decreasing in the average" order. After the scan procedure, the encoding of the bits of the 3D-Hadamard coefficients is done, bit-plane-by-bit-plane, with an efficient adaptive version of Golomb's run-length encoder, which produces a fully embedded output bitstream. Even with the use of these constrained techniques, good distortion versus rate results were achieved.**

Keywords - **Embedded coding, Hadamard transform, color video codec, three-dimensional transform, fast video codec, C# language.**

I. INTRODUCTION

The new coefficients sequency scan order proposed in this article was created for a color video codec named Fast Hadamard Video Codec (FHVC). This video codec was developed in C# language to be executed in a set-top box device futurely developed. The set-top box will be the interface between a fiber optics network and its users. This device will receive digital signals, extract video, audio and data information and send that information to an output device. Among other functions, as Internet accessibility and voice over IP, the set-top box will be able to receive and send video signals proceeding from, for instance, video on demand and video conference applications. The codec will be inserted on SIP (Session Initiation Protocol), which through the SDP (Session Description Protocol) has mechanisms that allow the insertion of new video codecs.

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 continuous increase in computational power. However, in some video coding and transmission applications, the use of high capacity processors is not possible or desirable. These situations require video codecs focused on reduced execution times and in few computational resources, just as the FHVC.

In order to reduce the codec execution time, the very simple Hadamard transform is used instead of the traditional DCT (Discrete Cosine Transform). The Hadamard transform was chosen because, even though it doesn't provide the same energy concentration advantages of the DCT, it is able to reduce the correlation of the coefficients and its implementation requires only additions and subtractions. To reduce even more the execution time, the highly efficient, but time-consuming, motion estimation and compensation techniques are avoided and the Hadamard transform is implemented in a three-dimensional fashion.

After Hadamard-transforming 3D blocks of pixels, the codec reads and reorders each coefficients block. It was found that the distribution of the dominant AC coefficients spreads along the major axes of the 3D-Hadamard cube, just as found for 3D-DCT cubes [1]. It was also found that the cube energy is concentrated according to the coefficient sequency number in the three dimensions. Based on this, a new scan order based on the multiplication of the three sequency numbers of each coefficient is developed to be adopted for the 3D-Hadamard coefficients. Generally, a specific scan order is better than a traditional three-dimensional zig-zag scan.

The codec encodes the resulting sequency reordered coefficients in a bit-plane-by-bit-plane fashion, refining their precision at each turn. This process renders a completely embedded encoded video file bitstream. The encoding of each bit plane of the 3D-Hadamard coefficients is accomplished using an adaptive version of Golomb's RLE (Run-Length Encoder) [2].

Due to cost restrictions, the set-top box device will be designed with a low capacity processor and few computational memory. Because of this, the entire video codec implementation is designed to perform only fast mathematical operations and to require small computational memory. All the multiplications and divisions operations are done by powers of two, so that they can be performed by variable binary shifts. Moreover, the system is implemented exclusively with 16-bit integer arithmetic, which also requires approximations. The errors introduced by these approximations can be compensated by the reduction of the compression rate. This reduction is acceptable, once the video codec is focused on speed, not on high compression performance.

An overview of the codec stages is provided in Section 2. The optimized 3D-Hadamard transform implementation is presented in Section 2.2 and the new coefficients sequency scan order is explained in Section 2.3. The results are presented in Section 3.

II. VIDEO CODEC OVERVIEW

The FHVC block diagram is depicted in Fig. 1. The video codec stages are described in this section in the order they appear in the figure below.

A. Video Codec Color Spaces

The FHVC is able to read color video sequences stored in tri-stimulus color space, such as RGB and YUV 4:2:0. Each such color plane is encoded separately and the allowed pixel bit-rate is divided among the color planes according to its significance. So, for the RGB format, the pixel bit-rate is equally divided, but in the YUV 4:2:0 format, the luminance plane receives more bits than the chrominance planes (because the U and V chrominance planes are one-fourth the size of the luminance Y plane). In the FHVC, only approximately 10% of the luminance rate is spent on the chrominance signal. This simple weighted bit-rate division procedure allows achieving higher compression rates.

In order to get the well-known advantages of the L-C (Luminance - Chrominance) formats, it is possible to convert an original RGB video sequence to a different internal color space (such as YUV 4:2:0) before beginning the coding process. Other color spaces are also supported by the FHVC and the conversions among them [3] are also implemented.

B. Three-dimensional Hadamard Transform

The Hadamard transform was chosen for the FHVC because of its base functions, composed by +1 and −1 elements. Thus, the transform computations do not require multiplications [4].

Although the use of the 3D Hadamard presents no innovation, this transform was chosen because it is simple (just as the normalizations stages), it is identical to its inverse and it is easy to extend results to 8x8x8 (or greater) transforms (generally not true for other transforms).

The Hadamard transform matrices H_n are NxN matrices, where $N = 2^n$. These can be generated by the core matrix

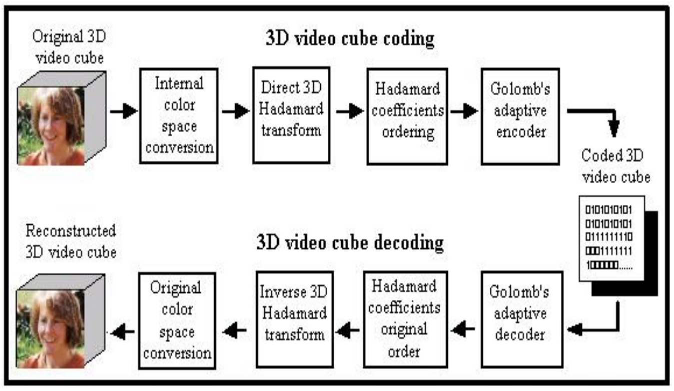
$$H_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1}$$

and the Kronecker product recursion

$$\begin{aligned} H_n &= H_{n-1} \otimes H_1 = H_1 \otimes H_{n-1} \\ H_n &= \frac{1}{\sqrt{2^n}}\begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix} . \end{aligned} \tag{2}$$

As an example, for n = 3, the Hadamard matrix becomes

$$H_3 = \frac{1}{\sqrt{8}} \begin{array}{c} \\ \left[\begin{array}{cc:cc:cc:cc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hdashline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ \hdashline 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right] \end{array} \begin{array}{c} \textbf{\textit{Sequency}} \\ \mathbf{0} \\ \mathbf{7} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{1} \\ \mathbf{6} \\ \mathbf{2} \\ \mathbf{5} \end{array} \tag{3}$$



Original 3D video cube
3D video cube coding
Internal color space conversion
Direct 3D Hadamard transform
Hadamard coefficients ordering
Golomb's adaptive encoder
Coded 3D video cube
0101010101
0101010101
0111111110
0001111111
1000000......
Reconstructed 3D video cube
3D video cube decoding
Original color space conversion
Inverse 3D Hadamard transform
Hadamard coefficients original order
Golomb's adaptive decoder

Fig. 1. Block diagram of the FHVC structure.

The basis vector of the Hadamard transform can also be generated by sampling the class of functions called Walsh functions shown in Fig. 2. These functions also take only the binary values ±1 and form a complete orthonormal basis for square integrable functions.

The number of zero crossings of a Walsh function or the number of transitions in a basis vector of the Hadamard transform is called its *sequency* (as well as the sinusoidal signals frequency can be defined in terms of the zero crossings).

In the Hadamard matrix generated through (2), the row vectors are not sequency ordered, as shown in (3) by the column named "*Sequency*". The existing sequency order of these vectors is called the *Hadamard order* because this is the order used by the transform.

Some Hadamard transform fast calculation methods were developed and the chosen one to be used by the FHVC is based on the fact that the H_n matrix can be written as a product of N sparse matrices $\tilde{H}$ [4]. Each multiplication by $\tilde{H}$ implies the execution of $log_2\ N$ additions or subtractions. As this multiplication is repeated N times, the total number of operations is $N * log_2 N$. Without this method, N^2 operations would be realized. Therefore, besides being simple, the Hadamard transform can also be fast.

The video sequence being encoded is partitioned into cubes and the Hadamard transform is applied separately in each cube and in each cube dimension (first in columns, then in lines and finally in frames). That can be done because the three-dimensional Hadamard transform is a separable transform. To evaluate the cube size effect in the coding performance, the FHVC can be executed with cubes of sizes 4x4x4 and 8x8x8.

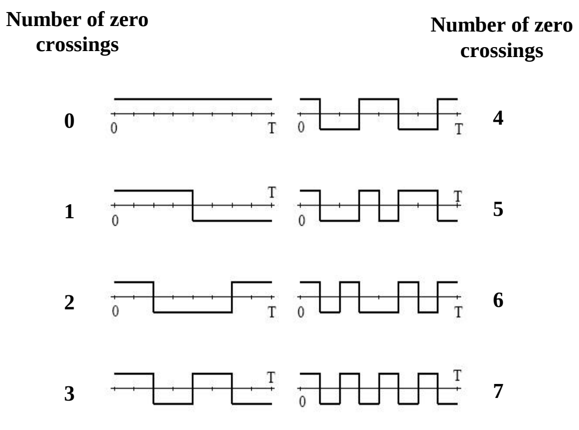


Fig. 2. Walsh functions sampled to Hadamard matrix generation.

The Hadamard transform computation in each cube dimension requires the division of the coefficients by $\sqrt{N}$ (as shown in (3)). This division is done to preserve the signal energy in the transform domain. Similarly, the divisions must be performed in the decoder because the same transform is used in the inverse operation, once the Hadamard transform is real, symmetric and orthogonal.

To avoid the fractional coefficients (generated by the $\sqrt{N}$ divisions) and to reduce the coefficients magnitude after the three Hadamard transform calculations (one in each cube dimension), a new implementation for the FHVC was developed. In this new approach, first, the Hadamard transform is applied in the cube's columns and after in the cube's lines. These two Hadamard transform calculations require two divisions of the coefficients by $\sqrt{N}$. In order to remove the square root operation, these two divisions are grouped, generating only one division by *N*. It is possible to move and group the division terms because the Hadamard transform is a linear process. Since the supported values for *N* are powers of 2, the division by *N* can be implemented through binary shifts. The consequence is that after the two Hadamard transform calculations, the coefficients suffer $log_2 N$ right binary shifts, instead of being divided twice by $\sqrt{N}$.

After the application of the Hadamard transform in the cube's columns and in the cube's lines, the transform must be applied in the cube's frames. This application requires a new division of the coefficients by $\sqrt{N}$. However, this is the last one Hadamard transform calculation done during the coding process. So, there is no other division term to be grouped with this division term in order to repeat the anterior procedure and remove the square root operation. The solution found, which is a new approach developed, was to bring the first transform division term of the decoding process to the end of the coding process. That can be done because, as presented before, the same Hadamard transform is performed in the decoder and the first decoding calculation was the coefficients division by $\sqrt{N}$.

In such case, the Hadamard transform is applied in the cube's frames, its division term is grouped with the first decoder Hadamard transform division term and the coefficients suffer new $log_2 N$ right binary shifts.

Two other simple ways to remove the square root operation of the Hadamard transform calculated in the cube's frames are:

- To group the $log_2 N$ right binary shifts with the previous one and perform all the binary shifts together *before* the transform calculation in the cube's frames.
- To group all the right binary shifts and perform them together *after* the transform calculation in the cube's frames.

The two options described above are simpler than the developed to the FHVC, but can't be used because in the first one, the coefficients values became too small and lost significant precision. In the second one, the coefficients values became too large, which required more computational memory.

So, the binary shifts inserted between the transform calculations, as described before, in addition to avoid fractional coefficients, provided final 3D-Hadamard coefficients with smaller values. This procedure improved the coding process through the reach of higher compression rates and allowed the complete implementation with a 16-bit integer arithmetic.

In the decoding process, the Hadamard transform is calculated in the cube's frames, then in the cube's lines and finally in the cube's columns. After all, the recovered pixels values suffer $log_2 N$ right binary shifts and the decoding is finished.

In order to achieve a better understanding of the proposed method, the traditional coding and decoding 3D-Hadamard transforms are shown, respectively, by the left and right sides of (4), where *PC* means *Pixels Cube*, *CC* means *Coefficients Cube* and $H' = \sqrt{N} * H_n$. The same transforms implemented according to the method proposed specially for the FHVC are shown by (5).

$$\underbrace{\frac{1}{\sqrt{N}}\left(H'\left(\underbrace{\frac{1}{\sqrt{N}}\left(H'\left(\underbrace{\frac{1}{\sqrt{N}}\left(H'(PC)\right)}_{columns}\right)\right)}_{lines}\right)\right)}_{frames} \Leftrightarrow$$
$$\underbrace{\frac{1}{\sqrt{N}}\left(H'\left(\underbrace{\frac{1}{\sqrt{N}}\left(H'\left(\underbrace{\frac{1}{\sqrt{N}}\left(H'(CC)\right)}_{frames}\right)\right)}_{lines}\right)\right)}_{columns} \qquad (4)$$

$$\underbrace{\frac{1}{\sqrt{N}}*\frac{1}{\sqrt{N}}\left(\underbrace{H'\left(\underbrace{\frac{1}{\sqrt{N}}*\frac{1}{\sqrt{N}}\left(\underbrace{H'\left(\underbrace{H'(PC)}_{columns}\right)}_{lines}\right)}_{shifts}\right)}_{frames}\right)}_{shifts} \Leftrightarrow$$
$$\underbrace{\frac{1}{\sqrt{N}}*\frac{1}{\sqrt{N}}\left(\underbrace{H'\left(\underbrace{H'\left(\underbrace{H'(CC)}_{frames}\right)}_{lines}\right)}_{columns}\right)}_{shifts} \qquad (5)$$

As the Hadamard matrix is composed only by +1 and −1 values, 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 $\sqrt{8}$. Considering that the transform is three-dimensional and that the first decoding division by $\sqrt{N}$ is carried out at the encoder, the total dynamic range gain is $8^3 / (\sqrt{8})^4 = 8$. Since $log_2 8 = 3$, only 3 additional bits are necessary to store the transform coefficients than the necessary to store the pixels values, which ensures the requirement of using few computational memory. This analysis for $N = 8$ is enough and sufficient because the maximum supported cube size is 8x8x8.

C. *Coefficients Sequency Scan Order*

With the energy compaction achieved by the 3D-Hadamard transform, the cube's energy is not any more disperse among its values and becomes concentrated in some coefficients of the cube. The reading of these coefficients in a decreasing, or “decreasing in the average”, order is important because increases the entropy coding efficiency (which is the next stage in the coding process).

To determine a fast and fixed reading order (independent of the information content in each cube), an analysis was developed with some video sequences and an approximately common spreading energy pattern was identified. In fact, the energy became concentrated in DC coefficient (correspondent to the sequency 0 row in (3)) and in AC low frequency coefficients (correspondent to the sequency 1, 2, 3 and 4 rows in (3)). The other AC coefficients are associated with higher frequencies and have smaller energy values. So, to assure that the coefficients with the higher energy values will be initially scanned, it is necessary to take in account the three sequency numbers of each coefficient (one sequency number in each dimension of the cube).

The new method developed for the appropriate consideration of the three sequency numbers of each coefficient performs the *multiplication* of them. Once each sequency number can vary from 0 to 7, the multiplication result value can vary from 0 to 343 (which corresponds to 7*7*7). This value is the one that will be used to order all coefficients of each cube. In order to do this, each multiplication value will be associated with all the three sequency numbers possible combinations that produce the value. For instance, the multiplication value 8 will be associated with the triples: (1, 2, 4), (1, 4, 2), (2, 1, 2), (2, 2, 2), (2, 4, 1), (4, 1, 2) and (4, 2, 1), where the triples are composed by (frame sequency number, line sequency number, column sequency number).

Once the multiplication value 0 could have almost two thousand associated triples (because would embrace all the triples with the sequency number 0 in any position), all the sequency numbers were added with the value 1 before the multiplication operations. Because of this, the final multiplication result can vary from 1 to 512 (which corresponds to 8*8*8).

Not only all the three sequency numbers possible triples must be associated with the correspondent multiplication value, but also these triples must be ordered for each multiplication value according, initially, by the *frame* sequency number, then by the *line* sequency number and finally by the *column* sequency number. That must be done because the cube’s energy is concentrated first in the *low frame* sequency numbers, then in the *low line* sequency numbers followed by the *low column* sequency numbers. In the example shown before for the multiplication value 8, the triples are already organized through this order.

The last coefficients scan order procedure stage is to translate the sequency numbers to the real cube coordinates. This translation is shown in Fig. 3 and it is necessary because of the transform *Hadamard order* explained in Section 2.2.

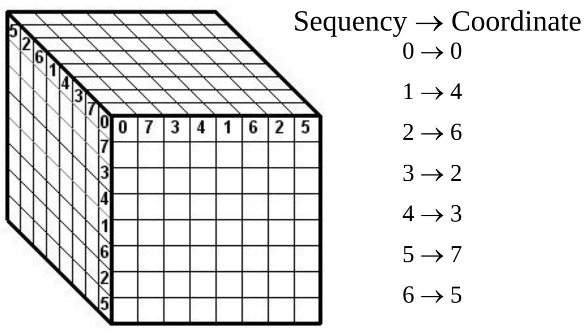


Fig. 3. Translation of the sequency numbers to the real cube coordinates.

In order to illustrate the coefficients cube sequency reading order implemented in the FHVC, the "Hall Monitor" QCIF video sequence in YUV 4:2:0 format is used with a cube size of 8x8x8.

Fig. 4(a) shows the sequence correspondent to the AC coefficients of the cube 7x8 belonging to the #264-#271 luminance frames block read by a column-line-frame scan order. This reading order generates high energy periodic peaks at each 64 coefficients approximately and low energy periodic peaks spaced each 8 coefficients.

Each higher energy peak in Fig. 4(a) corresponds to the coefficient read in sequency number 0 line and sequency number 0 column of each frame in the coefficients cube. The eight coefficients in this position are in the transverse axis in Fig. 3. The periodicity reading for these coefficients is 64 because all 63 coefficients from the previous frames are read first.

For the lower energy peaks in Fig. 4(a), the periodicity is 8 because the reading is line by line. Besides, the energy of the peaks in the sequency number 0 column is higher than the energy of the coefficients located in the other columns.

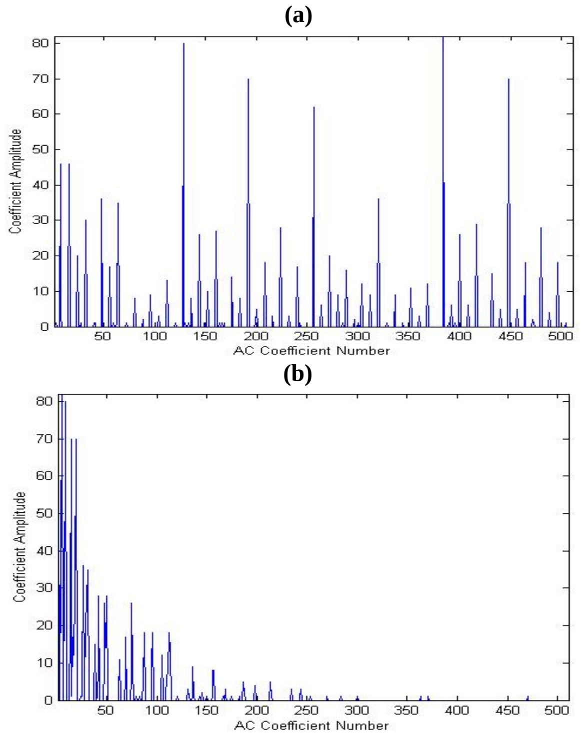


Fig. 4. AC coefficients of the "Hall Monitor" cube 7x8 belonging to the #264-#271 luminance frames block read by (a) column-line-frame scan order; (b) FHVC sequency scan order.

With the analysis of the graph in Fig. 4(a) it was possible to identify the spreading energy pattern of the Hadamard coefficients cube and to develop the new sequency scan order approach presented. The Fig. 4(b) shows the same coefficients of the Fig. 4(a), but read according to this new scan order. One can see that the coefficients are in a “decreasing in the average” order, which increases the efficiency of the next stage in the coding process (the entropy coding stage).

The first frame of the #264-#271 frames block is shown in Fig. 5 with the cube 7x8 used in Fig. 4 detached. It can be visually realized that the detached cube has a uniform content. So, the AC coefficients values are low, which agree with the low energy values shown by the lower AC coefficients numbers of the Fig. 4(a). In fact, these initial coefficients are proceeding from the frame with sequency number 7, which is the highest frequency of the cube.

Once the scan order for one cube of the block was developed, it was necessary to develop a scan order to read all the cubes of the block. If the cube size is 8x8x8 and the video sequence is in QCIF format, there are 22 * 18 = 396 cubes in each 8 frames block.

To explore the correlation between coefficients located in the same position of adjacent cubes, the coefficients of all the cubes are read according to a *spiral curve*, beginning with the coefficient of the superior left cube and finishing with the coefficient of the central cube. Initially, the DC coefficient of the superior left cube is read, then the DC coefficient of the right neighbor cube is read and so on until the DC coefficient of the central cube. Following, each AC coefficient (in the scan order described before) is read similarly for all the cubes.

The graph in Fig. 6(a) corresponds to the AC coefficients of the #264-#271 luminance frames block read by a column-line-frame scan order. Comparing this sequence with the one shown in Fig. 6(b), which corresponds to the FHVC sequency scan order, it is possible to verify that a better grouping of AC coefficients with similar values is achieved in fact.

D. Adaptive Entropy Coding with Golomb's RLE

Entropy coding is performed in the FHVC by an adaptive version of Golomb's RLE described in [2]. This entropy coding technique uses concepts extracted from well-known wavelets transforms methods, such as EZW (Embedded Zerotree Wavelet) [5] and SPIHT (Set Partitioning in Hierarchical Trees) [6].

Most video codecs perform quantization of the coefficient values before the entropy coding stage. The FHVC doesn't perform this explicit quantization and so, it can be used in a lossless manner. In fact, the FHVC performs an implicit coefficient quantization because the encoding is applied to bit planes (beginning with the most significant bit plane), which generates an embedded encoded bitstream. Thus, the decoding can be done aiming a specific desired rate. Another possibility is to control the bit-rate during encoding generating the coded bitstream at the desired rate.

III. RESULTS

In order to achieve a multi-platform code, the codec computational system is implemented in C# language, in the Microsoft Visual C# . NET environment. The encoding and the decoding processes, as well as all other supported operations Microsoft Visual C# .NET environment. The encoding and decoding processes, as well as all other supported operations (e. g. parameters settings and optimized coded video file bit-rate reduction), are controlled by the user through graphical interfaces.

With the FHVC, it is possible to perform the video sequence encoding and decoding separately or in sequence. Fig. 7 presents the screen for the last one option, where the video sequence decoding begins immediately after the end of the coding process. Through this interface, the user can inform the original video file path, format, resolution and quantity of frames. The desired coded file bit-rate can also be chosen. Bit-rates lower than the available in the interface are obtained through other graphical interface of the FHVC. This interface enables a fast reduction of the coded video file bit-rate to the bit-rate desired by the user. All the encoding and decoding operations can be followed by the user through the messages shown in the “Coding and Decoding Status” area. The messages shown in Fig. 7 correspond to the end of a successful video sequence coding and decoding process.

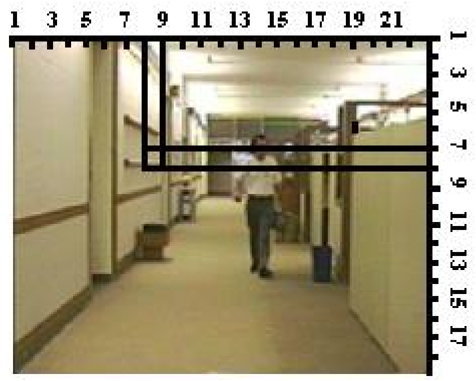


Fig. 5. Detached cube 7x8 belonging to the #264 frame of the "Hall Monitor" sequence whose AC coefficients of the luminance plane are shown in Fig. 4.

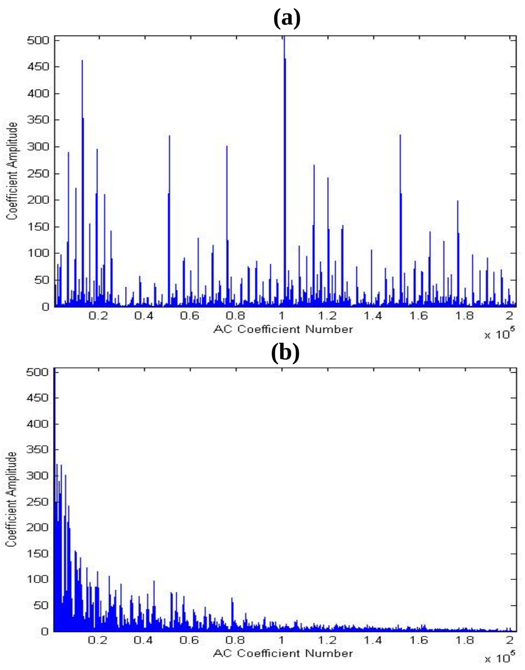
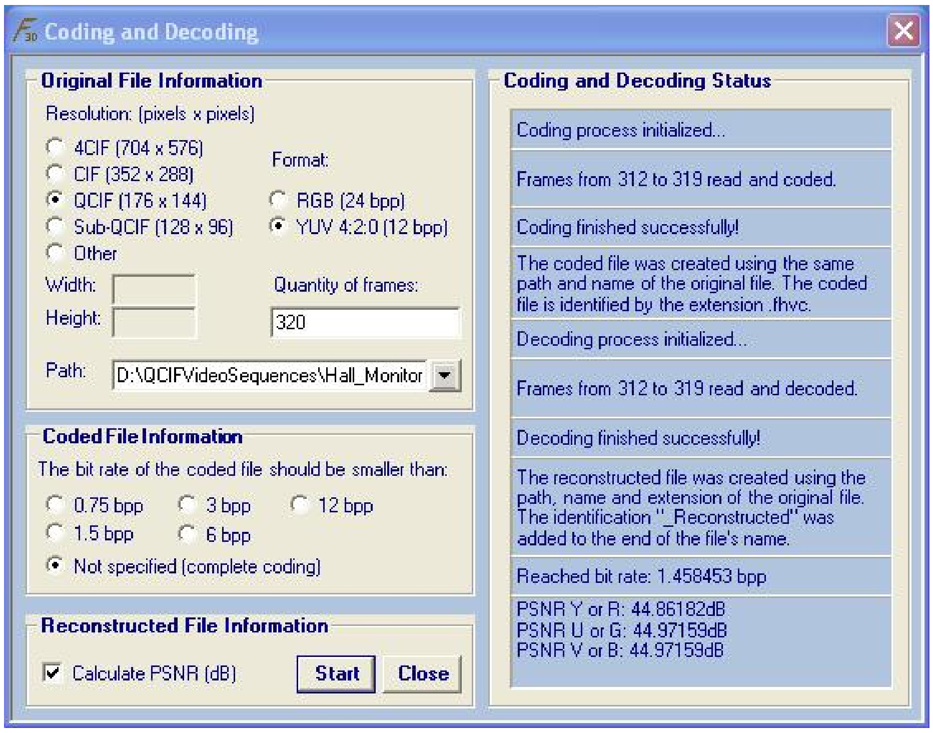


Fig. 6. AC coefficients of the "Hall Monitor" #264-#271 luminance frames block read by (a) column-line-frame scan order;
(b) FHVC sequency scan order.


Coding and Decoding

Original File Information

Resolution: (pixels x pixels)

- 4CIF (704 x 576)
- CIF (352 x 288)
- [✓ QCIF (176 x 144)]
- Sub-QCIF (128 x 96)
- Other

Width:

Height:

Format:

- RGB (24 bpp)
- [✓ YUV 4:2:0 (12 bpp)]

Quantity of frames: 320

Path: D:\QCIFVideoSequences\Hall_Monitor

Coded File Information

The bit rate of the coded file should be smaller than:

- 0.75 bpp
- 3 bpp
- 12 bpp
- 1.5 bpp
- 6 bpp
- [✓ Not specified (complete coding)]

Reconstructed File Information

[✓ Calculate PSNR (dB)]

Start Close

Coding and Decoding Status

Coding process initialized...

Frames from 312 to 319 read and coded.

Coding finished successfully!

The coded file was created using the same path and name of the original file. The coded file is identified by the extension .fhvc.

Decoding process initialized...

Frames from 312 to 319 read and decoded.

Decoding finished successfully!

The reconstructed file was created using the path, name and extension of the original file. The identification "_Reconstructed" was added to the end of the file's name.

Reached bit rate: 1.458453 bpp

PSNR Y or R: 44.86182dB
PSNR U or G: 44.97159dB
PSNR V or B: 44.97159dB

Fig. 7. FHVC graphical interface showing the successful result of a video sequence coding and decoding process.

In order to evaluate the FHVC computational efficiency, the encoding and decoding times of given 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 comparisons, we used the H.264/AVC official reference software obtained in [7]. The techniques used in this pattern are very different from the used on FHVC. Nevertheless, the comparison with the H.264/AVC is considered interesting because it is the video codec with the best performance nowadays.

It's very important to emphasize that there are H.264/AVC optimized implementations much faster than the official reference software. Even thus, we chose compare FHVC performance with the official reference software performance because this is a non-proprietary implementation and is always enabled complete, without restrictions. So, coding and decoding times of any other codec can also be compared with the H.264/AVC official reference software and then, be indirectly compared to the performance obtained with the FHVC.

Besides that, FHVC implementation is also not optimized for the hardware where it is being executed, once that C# (which is the programming language used in FHVC) is interpreted and a compiled code version was not generated.

Most H.264/AVC configuration parameters were set as "default", according to the software official manual developed by the Joint Video Team (JVT). The parameters not set as "default" in the configuration file are: Main profile, level 2.0, GOP of size 15 given by I-B-B-P-B-B-P-B-B-P-B-B-P-B-B, 5 reference frames and CABAC (Context-based Adaptive Binary Arithmetic Coding) entropy coding.

Fig. 8 presents the results obtained with the H.264/AVC and the FHVC for the "Hall Monitor" QCIF sequence in the YUV 4:2:0 format considering 8 frames per cube. Other sequences in QCIF and CIF formats were also tested with similar results.

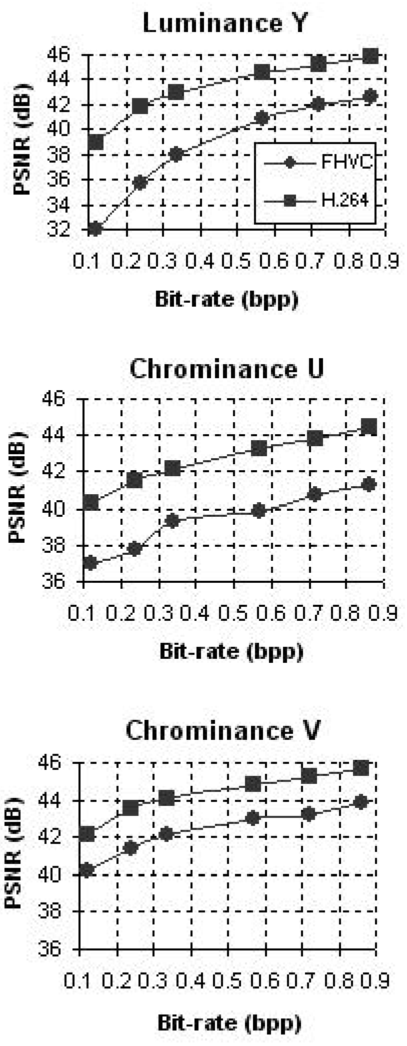


Fig. 8. PSNR versus bit-rate curves for the luminance and chrominance components of the "Hall Monitor" sequence.

The choice of using 8 frames per cube for coding the "Hall Monitor" sequence was done because this sequence has reduced motion and fixed background. Other sequences with high motion contents and rich detailed frames would be better coded with 4 frames per cube.

It is possible to verify in Fig. 8 that the H.264/AVC codec always achieves superior results in terms of peak signal to noise ratio (PSNR) versus bit-rate and that the difference in the codecs performance is less significant in the chrominance components.

According to Fig. 8, the FHVC uses approximately 3 times the bit-rate of H.264. This rate-distortion result is justified in applications where high capacity is available (e. g. optical links) but computational resources (complexity) at the end nodes are limited. In fact, this is the case for FHVC, once it was developed to be executed by a set-top box on a fiber optics network.

The visual quality comparison is presented in Fig. 9, where Fig. 9(a) is the original frame. Fig. 9(b) and Fig. 9(c) present this frame reconstructed after being encoded at 0.12 bit/pixel by H.264 and FHVC, respectively. It can be verified in Fig. 9(c), that the FHVC visual performance is satisfactory at a bit-rate of 0.12 bit/pixel (which implies a good compression by a factor of 100).

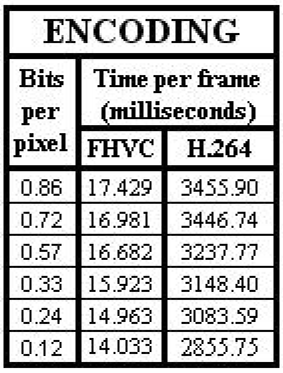
The good FHVC result at Fig. 9(d) was already expected, since at the rate of 0.33 bit/pixel, the FHVC achieves a reasonable PSNR value of 38 dB for the luminance component, as shown in Fig. 8.

The encoding and decoding times obtained with the FHVC and the H.264/AVC codec (measured at the same bit-rates) are shown in Table 1. As the FHVC is a symmetric codec, the encoding and decoding times are almost the same, unlike H.264/AVC, where the decoding is 23 times faster, in average, than the encoding.

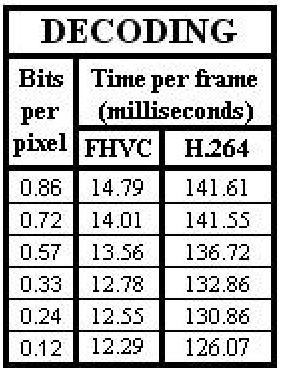
Based on Table 1, we can verify that the FHVC is consistently faster than the H.264/AVC official reference software, being approximately 200 times faster in encoding and 11 times faster in decoding.

As the FHVC is always superior to H.264/AVC in terms of execution time, but always inferior in terms of PSNR versus bit-rate, we looked for at which bit-rate the FHVC could achieve a sequence with similar visual quality to that achieved by H.264/AVC. This analysis showed that at the rate of 0.33 bit/pixel, the FHVC produces the frame shown in Fig. 9(d), which has visual quality comparable to the H.264/AVC frame, shown in Fig. 9(b).

TABLE 1. ENCODING AND DECODING TIMES FOR THE "HALL MONITOR" SEQUENCE.



ENCODING		
Bits per pixel	Time per frame (milliseconds)	
	FHVC	H.264
0.86	17.429	3455.90
0.72	16.981	3446.74
0.57	16.682	3237.77
0.33	15.923	3148.40
0.24	14.963	3083.59
0.12	14.033	2855.75



DECODING		
Bits per pixel	Time per frame (milliseconds)	
	FHVC	H.264
0.86	14.79	141.61
0.72	14.01	141.55
0.57	13.56	136.72
0.33	12.78	132.86
0.24	12.55	130.86
0.12	12.29	126.07

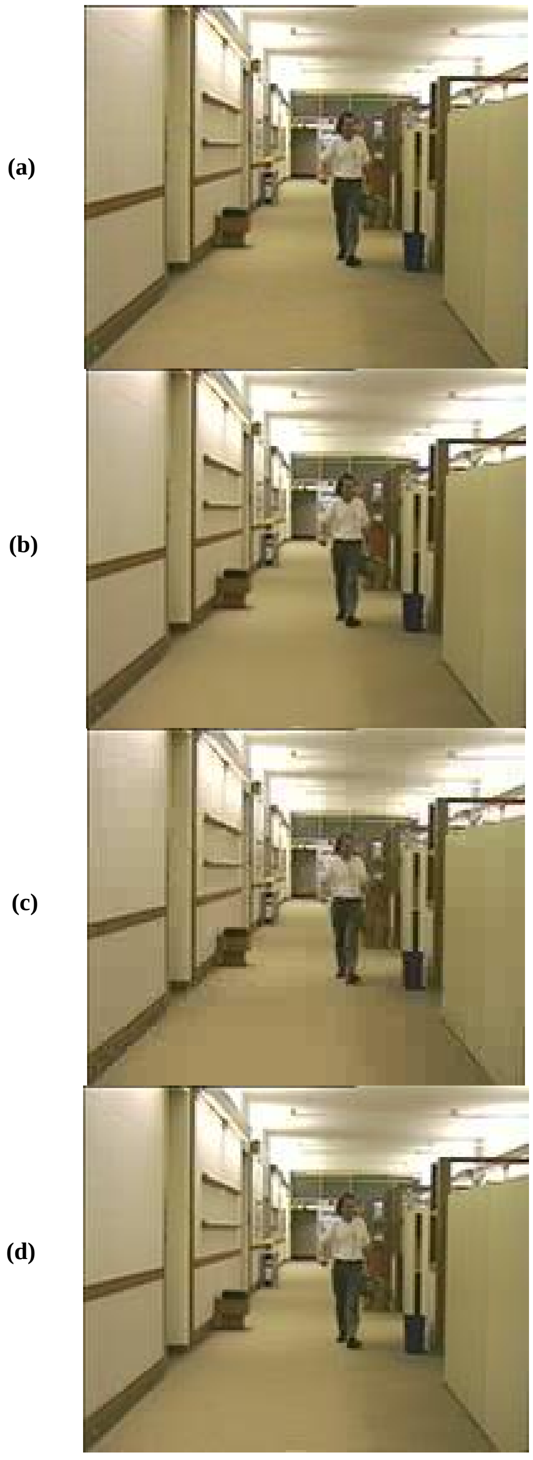


Fig. 9. "Hall Monitor" frame #264 (a) original; (b), (c) and (d) reconstructed after being encoded respectively by H.264 (0.12 bit/pixel), FHVC (0.12 bit/pixel), and FHVC (0.3 bit/pixel).

According to Table 1, the H.264/AVC codec requires 2,855.75 ms per frame for encoding at 0.12 bit/pixel. The FHVC requires 15.923 ms per frame for encoding at 0.33 bit/pixel, which produces similar visual quality frames for this sequence. It can be concluded that, at the cost of reducing the H.264/AVC compression rate by a factor of 2.75, an encoding 180 times faster can be achieved with the FHVC. Another important observation is that the encoding time of 15.923 ms per frame makes it possible to have real time video sequences encoding at 30 fps.

IV. CONCLUSIONS

A new 3D-Hadamard transform coefficients scan order based on the multiplication of the coefficients sequency numbers was presented. This scan order enabled the reading of the cubes coefficients in a decreasing, or “decreasing in the average”, order (which increases the entropy coding efficiency).

The scan order presented was created for a fast embedded 3D-Hadamard color video codec named FHVC, which was developed to be executed by a set-top box device on a broadband network.

With the modifications implemented in the normalizing factors (especially the groupings and the displacement of the normalizing factors from the decoder to the encoder) it was possible to adjust the dynamic range of the data to the available fixed point 16-bit representation. Besides, doing the normalization in pairs allows for implementation by simple bit shifts.

The codec behaves well for general video applications. For high bit rates applications (around 0.9 bpp), the PSNR degradation with respect to H.264 is less pronounced (around 3 dB) than what happens in the low bit rate scenario (around 0.1 bpp), where this degradation may be in excess of 6 dB.

The applicability of this codec is best directed to systems with complexity and storage limitations, possibly using fixed point processes, but enjoying high bit rates network connections (low cost codec but making use of high performance links). An added advantage is the exception of intellectual fees.

Performance results for one particular video sequence were shown. Results with other video sequences led to similar conclusions and indicated that, at the cost of a reduction in H.264/AVC compression rate by a factor of 2 up to 4, it is possible to get encoding times that are significantly (around 200 times) smaller with the FHVC.

V. REFERENCES

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

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

[3] G. Sullivan and S. Estrop, "Video Rendering with 8-bit YUV Formats". Redmond, WA, USA: Microsoft Digital Media Division, 2003.

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

[5] J. M. Shapiro, “Embedded Image Coding Using Zerotrees of Wavelet Coefficients”. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1993, v. 41, n. 12, p. 3445 - 3462.

[6] A. Said and W. A. Pearlman, “A New Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees”. *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, 1996, v. 6, p. 243 – 250.

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

[8] 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, p. 405-422, 1997.

[9] V. Testoni and M. H. M. Costa, "Fast Embedded 3D-Hadamard Color Video Codec", *XXV Simpósio Brasileiro de Telecomunicações - SBrT'2007*, Recife, PE, Brazil, Sept. 2007.