# Fast Embedded 3D-Hadamard Color Video Codec

Vanessa Testoni[1] and Max H. M. Costa

***Abstract* - This article introduces a color video codec to be executed by a set-top box on a fiber optics network. Due to its purpose, the codec is focused on reduced execution time, low computational complexity and use of meager computacional resources. The very simple Hadamard transform is used in a three-dimensional fashion, avoiding motion estimation and compensation techniques. A new scan order is proposed for the coefficient's cubes reading. 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.**

***Keywords* - embedded coding, Hadamard transform, color video codec, three-dimensional transform, fast video codec.**

## I. INTRODUCTION

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 verified 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 achieving reduced execution times and using few computational resources, as is the codec proposed in this article.

The color video codec will be executed by a set-top box equipment futurely developed. This set-top box will play the part of interface between a fiber optics network and its users. The equipment will receive digital signals, extract video, audio and data information and send that information to an ouput 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.

Vanessa Testoni and Max H. M. Costa, Department of Communications, Faculty of Electrical Engineering and Computation, State University of Campinas (UNICAMP), Campinas, SP, Brazil. E-mails: {vtestoni, max}@decom.fee.unicamp.br.

[1] The work of this author was supported by a grant from CAPES, Brazil.

The codec will be inserted on SIP (*Session Initiation Protocol*), which through the SDP (*Session Description Protocol*) has extensibility mechanisms that allows the insertion of new audio and video codecs.

Once that the set-box equipment will be inserted on a broadband network, the video codec execution must be fast. In order to reduce the codec execution time, the 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 coefficient's block. It was found that the distribution of the dominant AC coefficients spread along the major axes of the 3D-Hadamard cube, just as found for 3D-DCT cubes [1]. Based on this fact, a new scan order, which is generally better than a three-dimensional zig-zag scan, is adopted for the 3D-Hadamard coefficients.

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 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*), described in [2].

Due to cost restrictions, the set-top box equipment will be designed with a low capacity processor and few computacional memory. Because of that, the entire video codec implementation is designed to perform only fast mathematical operations and to require small computational memory. The multiplications and divisions operations are approximated by multiplications or divisions 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 all 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.

The proposed video codec was named Fast Hadamard Video Codec (FHVC). An overview of the codec stages is provided in Section 2. Section 3 presents the achieved results and Section 4 concludes the paper.

## II. VIDEO CODEC OVERVIEW

The FHVC structure is presented in Fig. 1. The video codec stages are described in this section in the order they appear in the figure.

### 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 separately encoded 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). This simple weighted bit-rate division helps 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 and the conversions among them are described on [3].

### B. *Three-dimensional Hadamard transform*

The Hadamard transform was chosen for the FHVC because of its simple base functions, composed only of +1 and –1 elements. Thus, the transform computations do not require multiplications [4]. The chosen transform implementation is based on the fact that the $H_N$ matrix can be written as a product of $N$ sparce matrices $\tilde{H}$. 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 $N * log_2 N$. Therefore, besides being simple, the transform is also fast.

The video sequence being encoded is partitioned into cubes and the Hadamard transform is separately applied in each cube. To evaluate the cube's size effect in the coding, the FHVC is executed with cubes of sizes 4x4x4 or 8x8x8.

The Hadamard transform computation in each cube dimension requires division of the coefficients by $\sqrt{N}$. Similarly, the divisions must be performed in the decoder, as the same transform is used in the inverse operation. This is the case because the Hadamard transform is real, symmetric and orthogonal. To avoid fractional coefficients, the divisions by $\sqrt{N}$ are grouped and implemented through binary shifts (since the N supported values are powers of 2).

As the Hadamard matrix is composed only of +1 and –1 values, the unidimensional 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 to store the pixels values. This analysis for $N = 8$ is sufficient because the maximum supported cube size is 8x8x8.

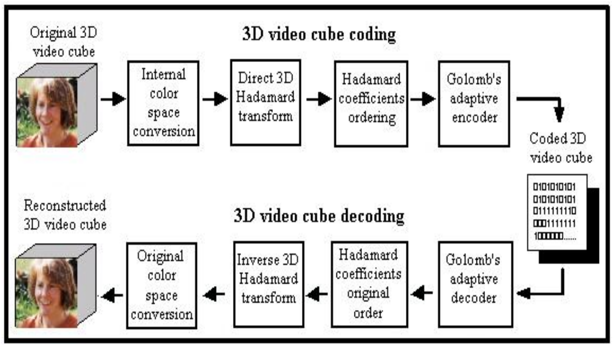

Original 3D video cube
3D video cube coding
Internal color space conversion
Direct 3D Hadamard transform
Hadamard coefficients ordering
Golomb's adaptive encoder
Coded 3D video cube
Reconstructed 3D video cube
3D video cube decoding
Original color space conversion
Inverse 3D Hadamard transform
Hadamard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

Fig. 1 – Block diagram of the FHVC structure.

### C. *Coefficients scan order*

With the energy compaction achieved by the three-dimensional Hadamard transform, the cube's energy is not any more disperse among its values. In fact, the energy becames concentrated in the cube's low frequency coefficientes, which are the AC coefficients close to the DC coefficient in the cube three dimensions. The other AC coefficients are associated with higher frequencies and have smaller values.

The reading of the cube's coefficients in a decreasing, or “decreasing in the average”, order is important because that 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 cube's information content), an analysis was developed with some video sequences and an approximately common spreading energy pattern was identified.

In order to illustrate the cube's coefficients reading order implemented in the FHVC, the "Miss America" QCIF video sequence in the YUV 4:2:0 format will be used. Considering this video sequence and others of the same type (with a fixed background and centrally located content), it was verified that the DC values distribution shows approximately the same behavior, with low absolute values in the cubes correspondent to the central region. Based on this, the DC coefficients's reading order was generated by a spiral curve, beginning with the DC coefficient of the superior left transform cube and finishing with the DC coefficient of the central cube.

The DC coefficient's reading order is illustrated on Fig. 2. In this figure, the frame's block is partitioned into 396 (where 396 = 22 * 18) cubes because there are 22 (where 22 = 176 / 8) cubes in each line and 18 (where 18 = 144 / 8) cubes in each column. Some DC values of all these cubes are represented by small white squares just to improve the reading process understanding, once that these are, in fact, the values read by the spiral that crosses the block and its cubes. This spiral was not drawn in its complete form in order to avoid excessive information in Fig. 2. So, a broken line was used to indicate that the spiral goes on to the figure center, where the white circumference represents its end.

Through the use of this reading order, the DC coefficients of each set of frames could be rearranged in an approximately decreasing order.

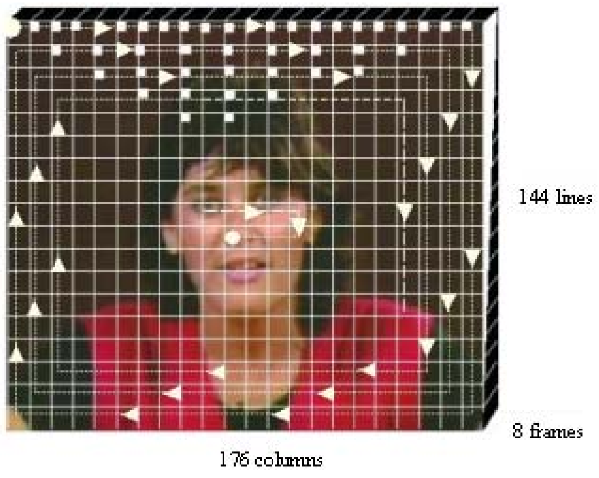


Fig. 2 – Spiral reading process of the DC coefficients in a video frames block.

The estimate of the AC coefficients energy distribution was based on the analysis given in [1]. The frame-line-column reading order of AC coefficients of one typical cube generated high periodic peaks at each 64 coefficients approximately, specially in the sequence beginning, and lower periodic peaks spaced each 8 coefficients. The higher energy peaks correspond to the coefficients located in the low frequencies regions of the cube, specially in the column axis.

Based on this analysis and inspired by the techniques described on [7] and [8], a new scan order was developed for AC coefficients reading.

Initially, the 8x8x8 cubes were partitioned in eight regions, as illustrated by Fig. 3. The cube 1 was divided again into eight regions (generating 2x2x2 sub-cubes) because it concentrates the largest amount of the energy.

To explore the existing correlation between AC coefficients located in the same position of adjacent cubes, the coefficients of the higher energy regions are read according to a spiral curve (as done for previously the DC coefficients). Therefore, the AC coefficients of the sub-cubes 1.1, 1.2 and 1.3 located on the line and column axis are read in alternating order, that is, first the coefficient (frame, line, column) = (0,0,1) of all the cubes in the frame block is read in a spiral form, followed by the coefficient (0,1,0) of all the cubes in the block, and so on.

The sub-cubes 2, 3 and 4 generally have medium magnitude coefficients and are rearranged by a specific reading order that prioritizes the coefficients next to the main cube's axis. These coefficients are illustrated in Fig. 4. The reading of the coefficients in adjacent cubes is also done according to a spiral curve for all the cubes in the frame block.

The graph in Fig. 5(b) illustrates the result obtained with the AC coefficients reordering used in the FHVC. Comparing this sequence with the one shown in Fig. 5(a), which corresponds to a frame-line-column order, it is possible to verify that a better grouping of AC coefficients with similar values is achieved.

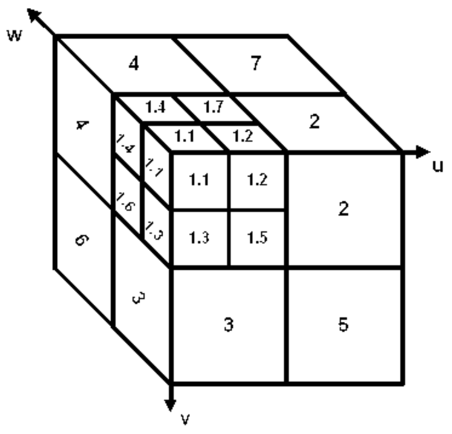


Fig. 3 - Eight region partition for reordering transform coefficients.

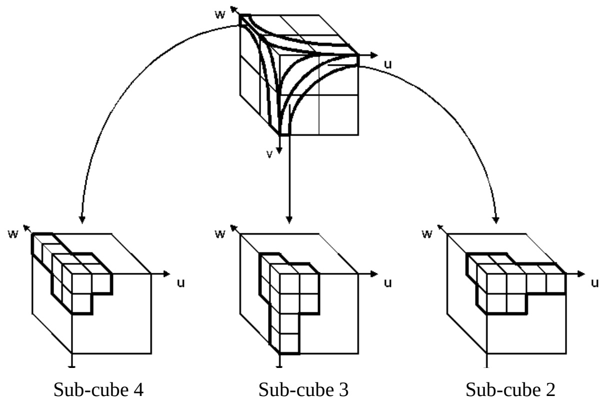


Fig. 4 - Higher coefficients regions of the sub-cubes 2, 3 and 4.

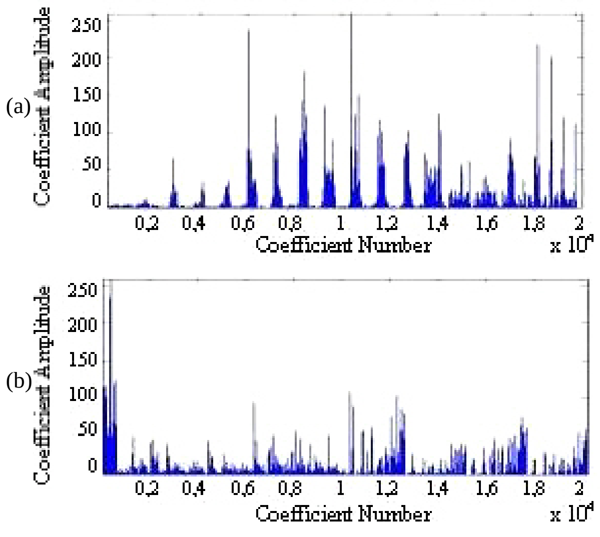


Fig. 5 - (a) AC coefficients reading as frame-line-column of the "Miss America" #64-#71 luminance frames and (b) reading of the same coefficients by the FHVC's scan order.

### 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, 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 at a specific desired rate. Another possibility is to control the bit rate during encoding and, then, generate the coded bitstream at the desired rate.

## III. RESULTS

To evaluate the FHVC's 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.

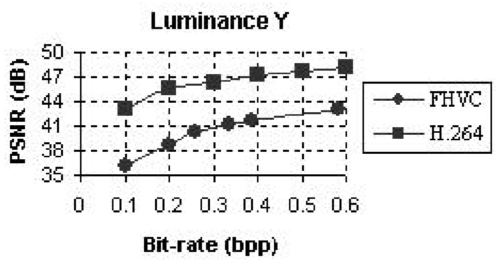
For results comparisons, we used the H.264/AVC official software reference obtained in [9]. The techniques used in this pattern are very different from the used on FHVC. Nevertheless, the comparison with the H.264/AVC is considered interesting because this is the video codec with the best performance results nowadays.

It's important to emphasize that there are H.264/AVC optimized implementations much faster than the official software reference. Even thus, we chose compare FHVC performance with the official software reference 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 to the obtained with the H.264/AVC official software reference and indirectly compared to the obtained with the FHVC.

Besides that, FHVC implementation is also not optimized for the hardware where its being executed, once that C# (which is the FHVC´s programming language) 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 main parameters not set as "default" in the configuration file are as follow: 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. 6 presents the results obtained with the H.264/AVC and the FHVC for the "Miss America" 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.



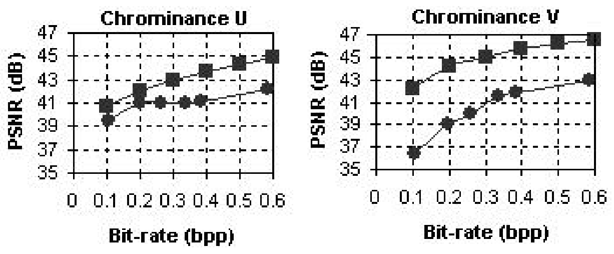


Fig. 6 - PSNR versus bit-rate curves for the "Miss America" sequence.

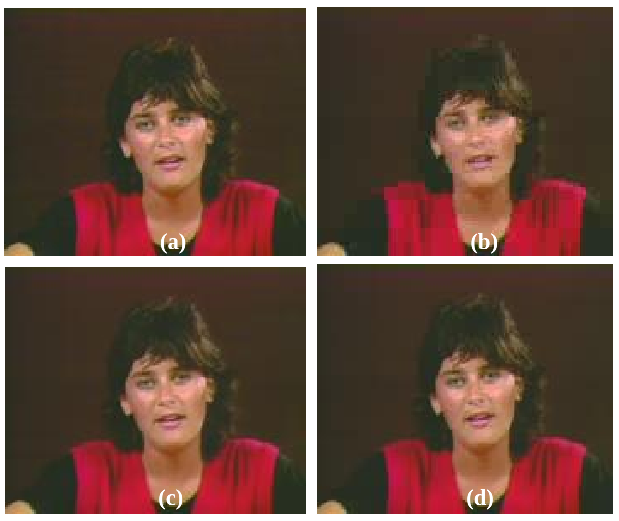


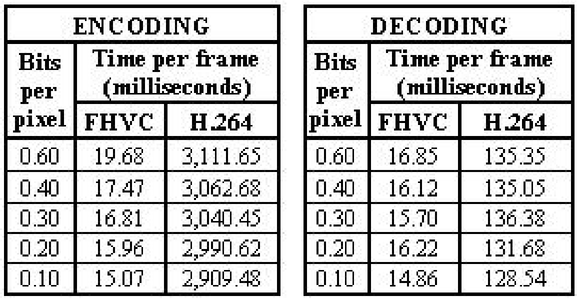
Fig. 7 – (a) "Miss America" original frame #143 and the same frame reconstructed after being encoded at 0.1 bit/pixel by (b) FHVC and (c) H.264 and at 0.38 bit/pixel by (d) FHVC.

The choice of using 8 frames per cube for coding the "Miss America" 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. 6 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 somewhat less significant in the chrominance components.

In terms of visual quality, it can be verified that the FHVC performance is satisfactory even at a low bit-rate of 0.1 bit/pixel, as shown by the frame reproduced in Fig. 7(b). This result was expected, since at the rate of 0.1 bit/pixel, the FHVC achieves a reasonable PSNR value of 36 dB for the luminance component.

TABLE 1
Encoding and decoding times for the "Miss America" sequence.



| ENCODING | | |
|---|---|---|
| Bits per pixel | Time per frame (milliseconds) | |
| | FHVC | H.264 |
| 0.60 | 19.68 | 3,111.65 |
| 0.40 | 17.47 | 3,062.68 |
| 0.30 | 16.81 | 3,040.45 |
| 0.20 | 15.96 | 2,990.62 |
| 0.10 | 15.07 | 2,909.48 |

| DECODING | | |
|---|---|---|
| Bits per pixel | Time per frame (milliseconds) | |
| | FHVC | H.264 |
| 0.60 | 16.85 | 135.35 |
| 0.40 | 16.12 | 135.05 |
| 0.30 | 15.70 | 136.38 |
| 0.20 | 16.22 | 131.68 |
| 0.10 | 14.86 | 128.54 |

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 with H.264/AVC, where the decoding is 22 times, in average, faster than the encoding.

Based on Table 1, we can verify that the FHVC is consistently faster than the H.264/AVC codec, being approximately 160 times faster in encoding and 8 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 verified 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 0.38 bit/pixel, the FHVC produces the frame shown in Fig. 7(d), which has visual quality comparable to the H.264/AVC frame, shown in Fig. 7(c).

According to Table 1, the H.264/AVC codec requires 2,909.48 ms per frame for encoding at 0.1 bit/pixel. The FHVC requires 17.47 ms per frame for encoding at 0.4 bit/pixel, which produces similar visual quality frames. It can be concluded that, at the cost of reducing the H.264/AVC compression rate by a factor of 4, an encoding 166 times faster can be achieved with the FHVC. Another important observation is that the encoding time of 17.47 ms per frame makes it possible to have real time encoding of video sequences at 30 fps.

## IV. CONCLUSIONS

A fast embedded 3D-Hadamard color video codec was presented. The main design criteria was the reduction in computational complexity with respect to standard hybrid predictive/transform based video codecs. This reduction is achieved with the elimination of costly motion compensation operations and with the choice of a very simple, binary transform. 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 compression rate by a factor of 3 to 5, it is possible to get encoding times that are significantly (around 150 times) smaller with the proposed codec. The comparison with H-264 was chosen as this is currently the foremost codec for a wide range of applications.

In order to achieve a multi-plataform code, the codec computational system is implemented in C# language, in 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.

## 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] K. Shen and E. J. Delp, "Wavelet Based Rate Scalable Video Compression". *Proceedings of the IEEE Transaction on Circuits and Systems for Video Technology*, 1999, v. 9, n. 1, p. 109 – 122.

[8] B. Kim, Z. Xiong and W. A. Pearlman, "Low Bit-Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT)". *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, 2000, v. 10, n. 8, p. 1374 – 1387.

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