# IMAGE COMPRESSION ALGORITHM BASED ON HILBERT SCANNING OF EMBEDDED QUADTREES: AN INTRODUCTION OF THE H*i*-SET CODER

*Jaime Moreno*[*]

Superior School of Mechanical and Electrical Engineers, National Polytechnic Institute, IPN Avenue, Lindavista, Mexico City, 07738, Mexico.
jmorenoe@ipn.mx

*Xavier Otazu*

Computer Vision Center, Universitat Autònoma de Barcelona, Building O, Bellaterra Campus, Barcelona, 08193, Spain.
xotazu@cvc.uab.es

## ABSTRACT

*In this work we present an effective and computationally simple algorithm for image compression based on* H*ilbert* S*canning of* E*mbedded quad*T*rees (H*i*-SET). It allows to represent an image as an embedded bitstream along a fractal function. Embedding is an important feature of modern image compression algorithms, in this way Salomon in [1, pg. 614] cite that another feature and perhaps a unique one is the fact of achieving the best quality for the number of bits input by the decoder at any point during the decoding. H*i*-SET possesses also this latter feature. Furthermore, the coder is based on a quadtree partition strategy, that applied to image transformation structures such as discrete cosine or wavelet transform allows to obtain an energy clustering both in frequency and space. The coding algorithm is composed of three general steps, using just a list of significant pixels. The implementation of the proposed coder is developed for gray-scale and color image compression. H*i*-SET compressed images are, on average, 6.20dB better than the ones obtained by other compression techniques based on the Hilbert scanning. Moreover, H*i*-SET improves the image quality in 1.39dB and 1.00dB in gray-scale and color compression, respectively, when compared with JPEG2000 coder.*

***Index Terms***— Fractals, Image Coding, Image Compression, Quadtrees, Wavelet Transforms.

## 1. INTRODUCTION

One of the biggest challenges of image compressors is the massive storage and ordering of coefficients coordinates. Some algorithms, like EZW [2], SPIHT and SPECK [3, 4], are based on the fact that the execution path gives the correct order, as a result of comparison of its branching points[5]. The proposed coder makes use of a Hilbert Scanning, which exploits the self-similarity of pixels. Hence, applying a Hilbert Scanning to Wavelet Transform coefficients takes the advantage of the similarity of neighbor pixels, helping to develop an optimal progressive transmission coder. In this way, at any step of the decoding process the quality of the recovered image is the best that can be achieved for the number of bits processed by the decoder up to that moment. Furthermore, the Hilbert's Space-Filling Curve gives by oneself its coordinate, since each branch belongs to a big one unless this is a root branch. Hence, the decoder just needs the magnitudes in order to recover a coefficient.

The paper is organized as follows: Section 2 outlines the Hilbert Scanning, illustrated by a function for generating recursively Hilbert curves. Section 3 describes the algorithm of the Image Coder based on Hilbert Scanning of Embedded quadTrees, divided in two parts. In the first part, *Startup Considerations*, we show how coordinates of a two-dimensional array are transformed and ordered by a Hilbert Mapping and stored into an one-dimensional array, in addition to test the significance of a quaternary branch. At the second part, *Coding Algorithm*, the stages of the algorithm are described, namely Initialization, Sorting and Refinement Passes. Experimental results applied for sixteen test images are given in section 4. In the last section, conclusions are explained.

## 2. HILBERT SCANNING

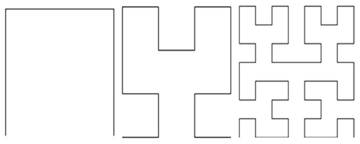
An important image compression task is to maximize the correlation among pixels, because the higher correlation at the preprocessing, the more efficient the data compression. The fractal Hilbert Scanning process remains in an area as long as possible before moving to the neighboring region, thus exploiting the possible correlation between neighbor pixels.

Let $\mathcal{W}$, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be the upper left, lower left, lower right and upper right quadrants, respectively. Let $\mathcal{U}$ (*up*:

[*]This work is mainly supported by the Ph.D grants 207950 and SeAca/COTEPABE/PL-255/10 of the Mexican Science and Technology National Council (CONACyT) and the National Polytechnic Institute of Mexico (IPN), respectively. In collaboration with Spanish Ministry of Education and Science under grant TIN2007-64577 and the Spanish Ministry of Science and Innovation under grant CSD2007-00018.

$\mathcal{W} \rightarrow \mathcal{X} \rightarrow \mathcal{Y} \rightarrow \mathcal{Z}$), $\mathcal{L}$ (*left*: $\mathcal{W} \rightarrow \mathcal{Z} \rightarrow \mathcal{Y} \rightarrow \mathcal{X}$), $\mathcal{R}$ (*right*: $\mathcal{Z} \rightarrow \mathcal{W} \rightarrow \mathcal{X} \rightarrow \mathcal{Y}$), and $\mathcal{D}$ (*down*: $\mathcal{X} \rightarrow \mathcal{W} \rightarrow \mathcal{Z} \rightarrow \mathcal{Y}$) be the alphabet. Each curve of the alphabet represents a $4^m$ coefficient arrangement, where $m$ is its level.

High order curves are recursively generated replacing each former level curve with the four later level curves, namely $\mathcal{U}$ is changed by $\mathcal{LUUR}$, $\mathcal{L}$ by $\mathcal{ULLD}$, $\mathcal{R}$ by $\mathcal{DRRU}$ and $\mathcal{D}$ by $\mathcal{RDDL}$.


(a) Canonical Scanning, Axiom = $\mathcal{D}$ [6].

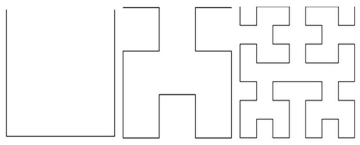

(b) Proposed Scanning, Axiom = $\mathcal{U}$.

**Fig. 1**. Hilbert's geometric constructions.

The original work made by David Hilbert [6], proposed a fractal axiom or initiator with a $\mathcal{D}$ trajectory (Figure 1a), while an $\mathcal{U}$ path is proposed to start with, since in a wavelet transformation the most relevant coefficients are at the upper-left quadrant, namely at the Residual Plane. The first three levels are portrayed in left-to-right order by Figure 1b.

## 3. THE H*i*-SET ALGORITHM

### 3.1. Startup Considerations

#### *3.1.1. Linear Indexing*

A linear indexing is developed in order to store the coefficient matrix into a vector. Let us define the Wavelet Transform coefficient matrix as $\mathcal{H}$ and the interleaved resultant vector as $\overrightarrow{\mathcal{H}}$, being $2^\gamma \times 2^\gamma$ be the size of $\mathcal{H}$ and $4^\gamma$ the size of $\overrightarrow{\mathcal{H}}$, where $\gamma$ is the Hilbert curve level. Algorithm 1 generates a Hilbert mapping matrix $\alpha$ with level $\gamma$, expressing each curve as four consecutive indexes. The level $\gamma$ of $\alpha$ is acquired concatenating four different $\alpha$ transformations in the previous level $\gamma - 1$. Algorithm 1 generates the Hilbert mapping matrix $\alpha$, where $\overrightarrow{\beta}$ refers a 180 degree rotation of $\beta$ and $\beta^T$ is the linear algebraic transpose of $\beta$. Figure 2 shows an example of the mapping matrix $\alpha$ at level $\gamma = 3$. Thus, each wavelet coefficient at $\mathcal{H}$ is stored and ordered at $\overrightarrow{\mathcal{H}}_\alpha$, being $\alpha_i$ the location of the coefficients into $\overrightarrow{\mathcal{H}}$.

**Algorithm 1**: Function to generate Hilbert mapping matrix of size $2^\gamma \times 2^\gamma$.

```
  Input: $\gamma$
  Output: $\alpha$
1 if $\gamma = 1$ then
2     $\alpha = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix}$
3 else
4     $\beta =$ Algorithm 1 $(\gamma - 1)$
5     $\alpha = \begin{bmatrix} \beta^T & (\overrightarrow{\beta})^T + (3 \times 4^{\gamma-1}) \\ \beta + 4^{\gamma-1} & \beta + (2 \times 4^{\gamma-1}) \end{bmatrix}$
```

#### *3.1.2. Significance Test*

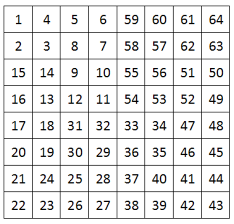
A significance test is defined as the trial of whether a coefficient subset achieves the predetermined significance level or threshold in order to be significant or insignificant. This test also defines how these subsets are formed.

With the aim of recovering the original image at different qualities and compression ratios, it is not needed to sort and store all the coefficients $\overrightarrow{\mathcal{H}}$ but just a subset of them: the subset of significant coefficients. Those coefficients $\overrightarrow{\mathcal{H}}_i$ such that $2^{thr} \leq |\overrightarrow{\mathcal{H}}_i|$ are called *significant* otherwise they are called *insignificant*. The smaller the $thr$, the better the final image quality and the lower the compression ratio.

Let us define a bit-plane as the subset of coefficients $\mathcal{S}_o$ such that $2^{thr} \leq |\mathcal{S}_o| < 2^{thr+1}$. Let $\widehat{\mathcal{H}_i}$ be the significance test of a given subset $\mathcal{S}_o$. It is defined as the $i$-th element of a binary-uncoded output stream $\widehat{\mathcal{H}}$

$$\widehat{\mathcal{H}_i} = \begin{cases} 1, & 2^{thr} \leq |\mathcal{S}_o| < 2^{thr+1} \\ 0, & \text{otherwise} \end{cases} . \quad (1)$$

Algorithm 2 shows how a given subset $\mathcal{S}_o$ is divided into four equal parts (line 6) and how the significance test (lines 7-12) is performed, resulting in four subsets ($\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$) with their respective significance stored at the end of $\widehat{\mathcal{H}}$. The subsets $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$ are four $2 \times 1$ cell arrays. The fist cell of each array contains one of the four subsets extracted from $\mathcal{S}_o$ ($\mathcal{S}_i(1)$) and the second one stores its respective significance test result ($\mathcal{S}_i(2)$).



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 5 | 6 | 59 | 60 | 61 | 64 |
| 2 | 3 | 8 | 7 | 58 | 57 | 62 | 63 |
| 15 | 14 | 9 | 10 | 55 | 56 | 51 | 50 |
| 16 | 13 | 12 | 11 | 54 | 53 | 52 | 49 |
| 17 | 18 | 31 | 32 | 33 | 34 | 47 | 48 |
| 20 | 19 | 30 | 29 | 36 | 35 | 46 | 45 |
| 21 | 24 | 25 | 28 | 37 | 40 | 41 | 44 |
| 22 | 23 | 26 | 27 | 38 | 39 | 42 | 43 |

**Fig. 2**. Example of the mapping matrix $\alpha$ with level $\gamma = 3$.

**Algorithm 2**: Subset Significance.

```
   Data: $\mathcal{S}_o, thr$
   Result: $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ and $\widehat{\mathcal{H}}$
 1 $\gamma = \log_4(length\ of\ \mathcal{S}_o)$
 2 Part 1 of the subsets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ and $\mathcal{S}_4$ is declared with $4^{\gamma-1}$ elements,
   while part 2 with just one element.
 3 $i = 1$
 4 $\widehat{\mathcal{H}}$ is emptied.
 5 for j=1 to $4^\gamma$ do
 6     Store $\mathcal{S}_o(j : i \times 4^{\gamma-1})$ into $\mathcal{S}_i(1)$.
 7     if $2^{thr} \leq \max|\mathcal{S}_i(1)| < 2^{thr+1}$ then
 8         $\mathcal{S}_i(2) = 1$
 9         Add 1 at the end of the $\widehat{\mathcal{H}}$.
10     else
11         $\mathcal{S}_i(2) = 0$
12         Add 0 at the end of the $\widehat{\mathcal{H}}$.
13     $i$ is incremented by 1, whereas $j$ by $4^{\gamma-1}$.
```

## 3.2. Coding Algorithm

Similarly to SPIHT and SPECK [3, 4], Hi-SET considers three coding passes: Initialization, Sorting and Refinement, which are described in the following subsections. SPIHT uses three ordered lists, namely the *list of insignificant pixels* ($LIP$), the *list of significant sets* ($LIS$) and the *list of significant pixels* ($LSP$). The latter list represents just the individual coefficients, which are considered the most important ones. SPECK employs two of these lists, the LIS and the LSP. Whereas H*i*-SET makes use of only one ordered list, the LSP.

### *3.2.1. Initialization Pass*

The first step is to define threshold $thr$ as

$$thr = \left\lfloor \log_2 \left( \max \left\{ \overrightarrow{\mathcal{H}} \right\} \right) \right\rfloor \ , \qquad (2)$$

that is, $thr$ is the maximum integer power of two not exceeding the maximum value found at $\overrightarrow{\mathcal{H}}$.

The second step is to apply Algorithm 2 with $thr$ and $\overrightarrow{\mathcal{H}}$ as input data, which divides $\overrightarrow{\mathcal{H}}$ into four subsets of $4^{\gamma-1}$ coefficients and adds their significance bits at the end of $\widehat{\mathcal{H}}$.

### *3.2.2. Sorting Pass*

Algorithm 3 shows a simplified version of the classification or sorting step of the H*i*-SET Coder. The H*i*-SET sorting pass exploits the recursion of fractals. If a quadtree branch is *significant* it moves forward until finding an individual pixel, otherwise the algorithm stops and codes the entire branch as *insignificant*.

Algorithm 3 is divided into two parts: Sign Coding (lines 2 to 9) and Branch Significance Coding (lines 11 to 16). The algorithm performs *the Sign Coding* by decomposing a given quadtree branch up to level $\gamma = 0$, i.e. the branch is represented by only 4 coefficients with at least one of them being *significant*. Only the sign of the significant coefficients is

**Algorithm 3**: Sorting Pass

```
   Data: $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ and $\gamma$
   Result: $LSP$ and $\widehat{\mathcal{H}}$
 1 $LSP$ and $\widehat{\mathcal{H}}$ are emptied.
 2 if $\gamma = 0$ then
 3     for i = 4 to 1 do
 4         if $\mathcal{S}_i(2)$ is significant then
 5             Add $\mathcal{S}_i(1)$ at the beginning of the $LSP$.
 6             if $\mathcal{S}_i(1)$ is positive then
 7                 Add 0 at the beginning of the $\widehat{\mathcal{H}}$.
 8             else
 9                 Add 1 at the beginning of the $\widehat{\mathcal{H}}$.
10 else
11     for i=1 to 4 do
12         if $\mathcal{S}_i(2)$ is significant then
13             Call Algorithm 2 with $\mathcal{S}_i(1)$ and $thr$ as input data and Store
               the results into $\mathcal{S}'_1, \mathcal{S}'_2, \mathcal{S}'_3, \mathcal{S}'_4$ and $\widehat{\mathcal{H}}'$.
14             Call Algorithm 3 with $\mathcal{S}'_1, \mathcal{S}'_2, \mathcal{S}'_3, \mathcal{S}'_4$ and $\gamma - 1$ as input
               data and Store the results into $\widehat{\mathcal{H}}'$ and $LSP'$.
15             Add $\widehat{\mathcal{H}}'$ at the end of the $\widehat{\mathcal{H}}$.
16             Add $LSP'$ at the end of the $LSP$.
```

coded, 0 for positives and 1 for negatives. Also each significant coefficient is added into a spare $LSP$ or $LSP'$. *The Branch Significance Coding* calls Algorithm 2 in order to quarter a branch in addition to recursively call an entire sorting pass at level $\gamma - 1$ up to reach the elemental level when $\gamma = 0$. The Significance Test results of a current branch (obtained by the Algorithm 2) and the ones of next branches (acquired by Algorithm 3, denoted as $\widehat{\mathcal{H}}'$) are added at the end of $\widehat{\mathcal{H}}$. Also, all the significant coefficients found in previous branches (all the lists $LSP'$) are added at the end of the $LSP$.

### *3.2.3. Refinement Pass*

At the end of $\widehat{\mathcal{H}}$, the $(thr - 1)$-th most significant bit of each ordered entry of the LSP, including those entries added in the last sorting pass, are added. Then, $thr$ is decremented and another Sorting Pass is performed. The Sorting and Refinement steps are repeated up to $thr = 1$.

## 4. EXPERIMENTS AND NUMERICAL RESULTS

The Peak Signal to Noise Ratio (PSNR) between the original image $f(i,j)$ and the reconstructed image $\hat{f}(i,j)$ is employed in order to estimate the degradation introduced during the compression process. The PSNR is defined by

$$PSNR = 10 \log_{10} \left( \frac{\mathcal{G}_{max}{}^2}{MSE} \right) \ , \qquad (3)$$

where $\mathcal{G}_{max}$ is the maximum possible intensity value in $f(i,j)$ ($M \times N$ size) and the MSE has the form:

$$MSE = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left[ f(i,j) - \hat{f}(i,j) \right]^2 \qquad (4)$$

## 4.1. Comparison with Hilbert Curve based algorithms

H*i*-SET has some resemblances with other image compression algorithms, like the ones developed by Kim and Li [7] and Biswas [8]. Similarly to them, H*i*-SET maximizes the correlation between pixels using the Hilbert scanning, namely all three methods use the same fractal structure. Hence it is important to know if there has been a substantial improvement of such methods. The differences between H*i*-SET and these old methods are that the herein presented method is an embedded algorithm and proposes a coding scheme, while the Kim and Biswas methods are not embedded, since the entropy is encoded by means of a Huffman coder.

Table 1 shows the comparison between the algorithm performed by Kim and Li and H*i*-SET only for the case of the image *Lenna* (it is the only image reported result by cited authors). On the average, H*i*-SET reduces the Mean Square Error by 63.07% (Peak Signal-to-Noise Ratio in 4.75dB). In addition, compared to the algorithm proposed by Biswas (Table 2), H*i*-SET diminishes the MSE in 84.66% or 8.15dB. For example, a compressed image with PSNR=28.07dB is stored by H*i*-SET at 4.87kB (0.152 bpp), while the Biswas algorithm needs 21.41kB (0.669 bpp), that is, 4.4 times more than H*i*-SET. Thus, on average our method improves the image quality of these methods in approximately 6.20dB.

**Table 1**. Comparing H*i*-SET against the algorithm of Kim [7].

| bpp (rate) | Kim (PSNR in dB) | H*i*-SET (PSNR in dB) |
|---|---|---|
| 0.25 (32.00:1) | 27.51 | 31.00 |
| 0.50 (16.00:1) | 30.00 | 34.88 |
| 0.75 (10.67:1) | 31.49 | 36.72 |
| 1.00 (8.00:1) | 32.91 | 38.30 |

**Table 2**. Comparing H*i*-SET against the algorithm of Biswas[8].

| bpp (rate) | Biswas (PSNR in dB) | H*i*-SET (PSNR in dB) |
|---|---|---|
| 0.669 (11.96:1) | 28.07 | 36.15 |
| 0.725 (11.03:1) | 28.45 | 36.55 |
| 0.788 (10.15:1) | 28.73 | 36.99 |

## 4.2. Comparing H*i*-SET and JPEG2000 coders

An image compression system is a set of processes with the aim of representing the image with a string of bits, keeping the length as small as possible. These processes are mainly Transformation, Quantization and Entropy Coding. For the sake of comparing the performance between the JPEG2000 standard [9] and H*i*-SET coders, each one develops a near-lossless compression with the same subset of wavelet coefficients. This way, this subset of wavelet coefficients are selected from the original source image data $\mathcal{I}_{org}$ such that $\mathcal{I}_{org} \geq 2^{thr-bpl+1}$, being $bpl$ the desired bit-plane and $thr$ the maximum threshold

$$thr = \left\lfloor \log_2 \left( \max_{(i,j)} \left\{ \left| \mathcal{I}_{org\,(i,j)} \right| \right\} \right) \right\rfloor . \qquad (5)$$

These selected coefficients are inverse wavelet transformed in order to create a new source of image data, i.e. $\mathcal{I}'_{org}$, which are losslessly compressed by each coder, namely until the last bit-plane. Figure 3 depicts this process. The software used to obtain a JPEG2000 compression for the experiments is *JJ2000*, developed by Cannon Research, École Polytechnique Fédérale de Lausanne and Ericsson [10]. The irreversible component transformation (ICT, $YC_bC_r$) is used in addition to the 9/7 irreversible wavelet transform.

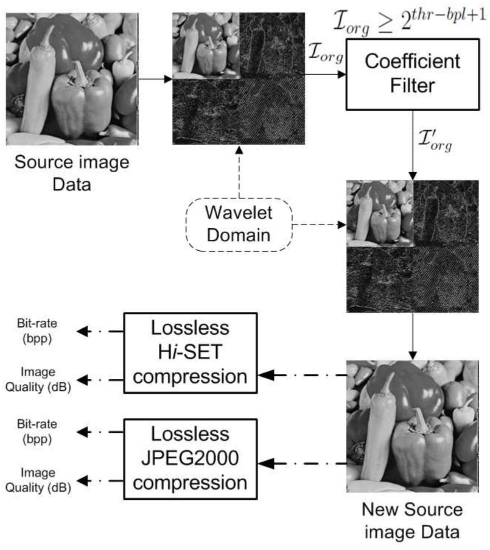


**Fig. 3**. Bit-plane selection. Some coefficients are selected provided that they fulfil the current threshold.

H*i*-SET is tested on the 24-bit-depth color images of the *Miscellaneous volume* of the University of Southern California, Signal and Image Processing Institute image database (USC-SIPI image database) [11]. This image database includes, among others, eight 256 × 256 pixel images (Figure 4) and eight 512 × 512 pixel images (Figure 5).

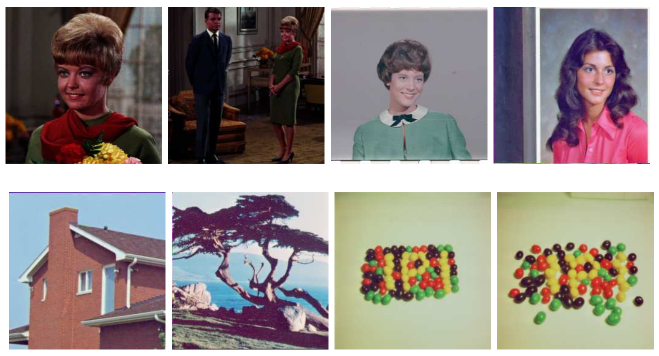


**Fig. 4**. Tested 256 × 256 pixel 24-bit color images, belonging to the USC-SIPI image database.

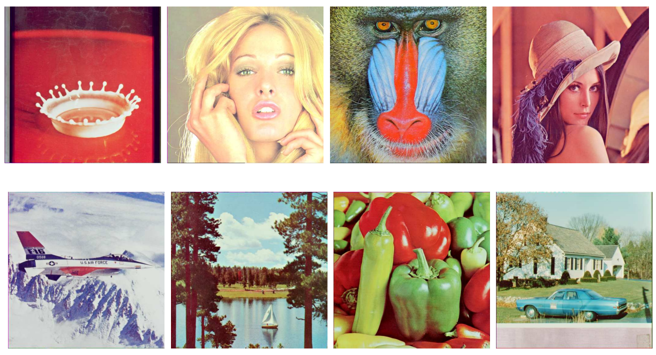
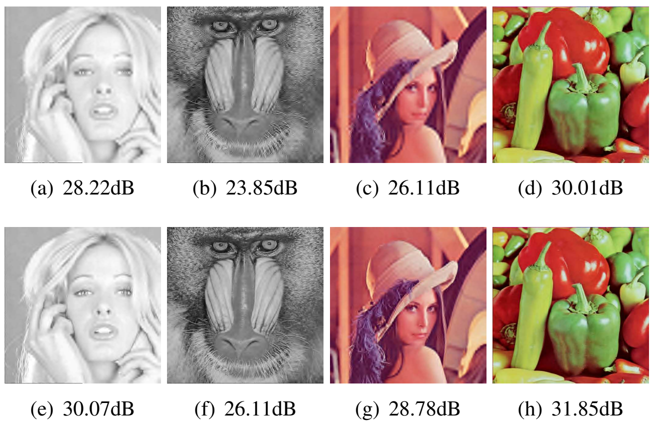


**Fig. 5**. Tested $512 \times 512$ pixel 24-bit Color Images, belonging to the USC-SIPI image database.

The compression algorithms are evaluated in two experiments: gray-scale images (just $Y$ component) and on color images ($YC_bC_r$ components).

**Experiment 1. Gray-scale images.** In this experiment, the source image data both for the JPEG2000 standard coder and H*i*-SET algorithms are the selected images of the USC-SIPI image database (Figures 4 and 5) transformed into gray-scale images ($Y$ component). Figure 7a shows the average quality of the recovered images as a function of compression rate, where the differences between JPEG2000 (heavy dots) and H*i*-SET ( heavy stars) are depicted. H*i*-SET improves either the image quality in approximately 1.39dB with the same compression rate or bit-rate in approximately 0.22bpp with the same image quality. It implies saving around 1.76KBytes or 7.04KBytes for $256 \times 256$ and $512 \times 512$ pixels gray-scale images, respectively. On average, a $512 \times 512$ image compressed by means of JPEG2000 with 30dB needs 15.24KBytes at 0.4763bpp, while H*i*-SET needs 5.7456KBytes less than the standard, namely at 0.2967bpp. The difference in visual quality is depicted in Figures 6a and 6e (image *Tiffany*) and 6b and 6f (image *Baboon*). *Tiffany* is compressed at 0.17bpp, while Baboon at 0.86bpp. The image quality of the recovered image *Tiffany* coded by JPEG2000 is 1.85dB lower than the one obtained by H*i*-SET. Similarly, the quality of the image *Baboon* increases by 2.26dB when H*i*-SET is employed.

**Experiment 2. Color images.** In this second experiment, the tests are made on the selected images of the USC-SIPI image database transformed into the same color space used by JPEG2000 ($Y$, $C_b$ and $C_r$). Figure 7b shows the compression rate and their average quality. On the average, a $512 \times 512$ image compressed by H*i*-SET (heavy stars) with 35dB is stored in 62.82KBytes at 1.963bpp, while JPEG2000 (heavy dots) stores it in 87.97KBytes at 2.749bpp. Figure 6 depicts the differences when the images *Lenna* and



(a) 28.22dB (b) 23.85dB (c) 26.11dB (d) 30.01dB

(e) 30.07dB (f) 26.11dB (g) 28.78dB (h) 31.85dB

**Fig. 6**. Examples of reconstructed images compressed by means of JPEG2000 (a-d) and H*i*-SET (e-h) at 0.17bpp (a & e), 0.86bpp (b & f), 0.39bpp (c & g) and 1.05bpp (d & h).

*Peppers* are compressed at 0.39bpp and 1.05bpp by JPEG2000 (c and d) and H*i*-SET (g and h), respectively. At the same compression ratio, H*i*-SET improves image quality by 2.67dB for *Lenna* and 1.84dB for *Peppers*. On average, H*i*-SET either compresses 0.29bpp more with the same image quality or reduces in 1.00dB the error with the same bit-rate. Hence H*i*-SET saves 2.32KBytes (for $256 \times 256$ images) or 9.28KBytes (for $512 \times 512$ images) in comparison to the JPEG2000 standard coder.

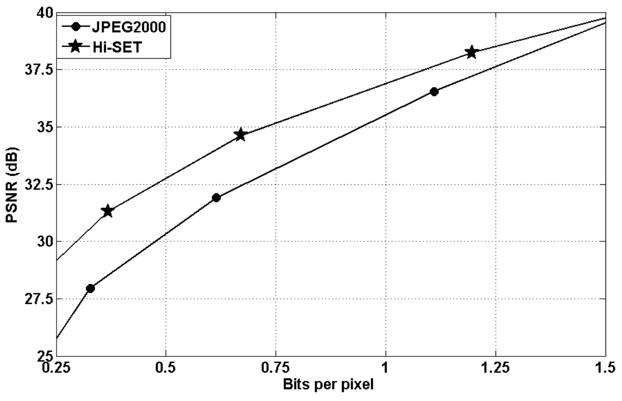
## 5. CONCLUSIONS

The H*i*-SET coder presented in this work is based on Hilbert scanning of embedded quadtrees. It has low computational complexity and important properties of modern image coders such as embedding and progressive transmission. This is achieved using the principles of partial sorting by magnitude when a sequence of thresholds decreases. The image coding results improve 6.20dB the image quality, when compared to other algorithms that use a Hilbert scanning as a method for pixel ordering. H*i*-SET improves the image quality around 1.39dB when compared to the JPEG2000 standard coder for gray-scale images and 1.00dB for color images, in addition to save around 0.22bpp in monochromatic images and 0.29bpp for RGB images.

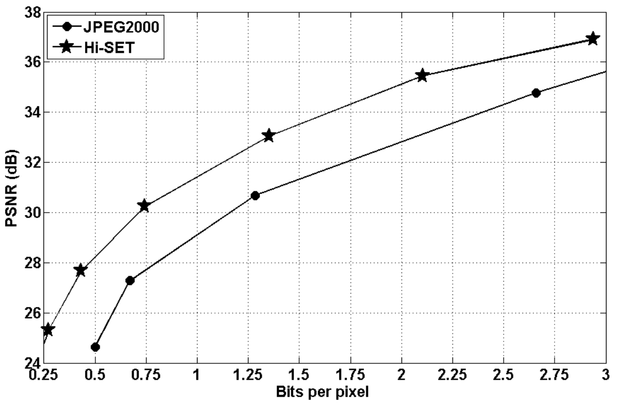
## 6. REFERENCES

[1] David Salomon, *Data Compression: The Complete Reference*, ISBN-13: 978-1-84628-602-5. Springer-Verlag London Limited, fourth edition, 2007.

[2] J.M Shapiro, “Embedded image coding using Zerotrees of wavelet coefficients,” *IEEE Transactions on Acous-*



(a)



(b)

**Fig. 7**. Compression Rate vs Image Quality between JPEG2000 and Hi-SET: (a) Gray-scale and (b) Color compression.

*tics, Speech, and Signal Processing*, vol. 41, no. 12, pp. 3445 – 3462, Dec. 1993.

[3] W. A. Pearlman and A. Said, “Image wavelet coding systems: Part II of set partition coding and image wavelet coding systems,” *Foundations and Trends in Signal Processing*, vol. 2, no. 3, pp. 181–246, 2008.

[4] W. A. Pearlman and A. Said, “Set partition coding: Part I of set partition coding and image wavelet coding systems,” *Foundations and Trends in Signal Processing*, vol. 2, no. 2, pp. 95–180, 2008.

[5] Bryan E. Usevitch, “A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000,” *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 22–35, 2001.

[6] David Hilbert, “Über die stetige Abbildung einer Linie auf ein Flächenstück,” *Mathematische Annalen*, vol. 38, no. 3, pp. 459–460, Sept. 1891.

[7] HyungJun Kim and C.C. Li, “Lossless and lossy image compression using biorthogonal wavelet transforms with multiplierless operations,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 8, pp. 1113–1118, 1998.

[8] Sambhunath Biswas, “One-dimensional B-B polynomial and hilbert scan for graylevel image coding,” *Pattern Recognition*, vol. 37, no. 4, pp. 789 – 800, 2004, Agent Based Computer Vision.

[9] David S. Taubman and Michel W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, ISBN: 0-7923-7519-X. Kluwer Academic Publishers, 2002.

[10] Cannon Research, École Polytechnique Fédérale de Lausanne, and Ericsson, “JJ2000 implementation in Java, available at http://jj2000.epfl.ch/,” 2001.

[11] Signal and Image Processing Institute of the University of Southern California, “The USC-SIPI image database, available at http://sipi.usc.edu/database/,” 1997.