DWT-SPIHT IMAGE CODEC IMPLEMENTATION

J. Malý, P. Rajmic

Department of Telecommunications, Brno University of Technology, Brno, Czech Republic

Abstract

Lossy image compression is a subject of great importance today, as it represents an efficient way of reducing storage requirements. This paper proposes an implementation of discrete-time wavelet transform based image codec using Set Partitioning of Hierarchical Trees (SPIHT) coding in the MATLAB environment.

1 Introduction

When we speak of image compression, there are generally two different solutions, the lossless and lossy concept of operation. Lossy compression methods most often rely on transforming spatial image domain into a domain, that reveals image components according to their relevance, making it possible to employ coding methods that take advantage of data redundancy in order to suppress it.

Since first attempts, the discrete cosine transform (DCT) domain has been used. Image is divided into segments (due to the fact DCT was designed to work with periodic states) and each segment is then a subject of the transform, creating a series of frequency components that correspond with detail levels of the image. Several forms of coding are applied in order to store only coefficients that are found as significant. Such a way is used in the popular JPEG file format, and most video compression methods are generally based on it.

The other approach is based on discrete-time wavelet transform (DWT), which produces a multi-scale image decomposition. By employing filtering and subsampling, a result in the form of the decomposition image (for classical dyadic approach) is produced, very effectively revealing data redundancy in several scales. A coding principle is then applied in order to compress the data. We can use scalar-based coding (such as JPEG2000's EBCOT scheme), or process the image with vector methods, taking advantage of so-called zerotrees (structures that result from data similarity across different subbands). Typically this is a case of Embedded zerotree wavelet (EZW) or Set partitioning in hierarchical trees (SPIHT).

This article deals with a complete implementation of SPIHT codec in the MATLAB environment. The implementation is designed to handle grayscale (256 level) images of square resolutions that are a power of two. We also introduce simulation of the algorithm on several different images and explore the results by subjective and objective means of comparison.

2 Discrete wavelet transform implementation

There exist two ways how to implement the computation of the discrete-time wavelet transform. The first approach uses convolution (filtering) with appropriate boundary handling, the second is a fast lifting approach, a refined system of very short filters which are applied in a way that produces the same result as the first approach, introducing significant computational and memory savings [1].

Lifting scheme is derived from a polyphase matrix representation of the wavelet filters, a representation that is distinguishing between even and odd samples. Using the algorithm of filter factoring, we split the original filter into a series of shorter filters (typically Laurent polynomials of first degree). Those filters are designed as lifting steps; each step one group of coefficients are lifted (altered) with the help of the other one (classical dyadic transform always leads to two groups of coefficients, low-pass and high-pass). Data flow in the final algorithm is presented in Fig. 1.

Lifting scheme has one disadvantage – we have to factorize the filters before we can start the computation. In the case of the most widely used image processing wavelet filters – the Cohen-Daubechies-Feauveau 9/7-tap filters (CDF 9/7) – it is an easy task since the most efficient scheme has already been proposed and used (for example in the JPEG 2000 codec). Computational savings of the scheme are gained from the length of the filters (convolution with a 9-tap filter is slower than with a series of two-tap filters) and due to minimum dependency between the coefficients, the whole

computation can be done in one memory block of original signal size. We also resolve the problem of boundary effect handling very easily, as we only have to copy one missing sample on each side, using half-sample symmetry – as seen at Fig. 1.

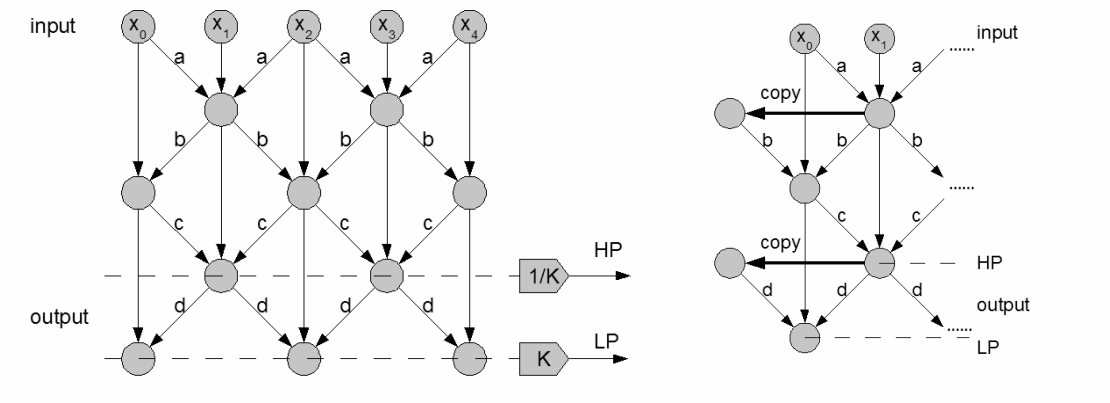

Figure 1: Data dependency diagram showing the data flow according to CDF9/7 lifting scheme implementation. Source signal is defined as x, the result is taken from HP and LP branches for high-pass and low-pass coefficients, respectively; a,b,c,d and K are constants computed by filter factorization process. Picture on the right is demonstrating boundary handling.

Since images are two-dimensional signals, we have to extend the scheme to 2D space by applying the transform row- and column-wise, respectively (taking separability of the transform into account). As a consequence four subbands arise from one level of the transform – one low-pass subband containing the coarse approximation of the source image called LL subband, and three high-pass subbands that exploit image details across different directions – HL for horizontal, LH for vertical and HH for diagonal details. In the next level of the transform, we use the LL band for further decomposition and replace it with respective four subbands. This forms the decomposition image. An example can be observed in Fig. 2.

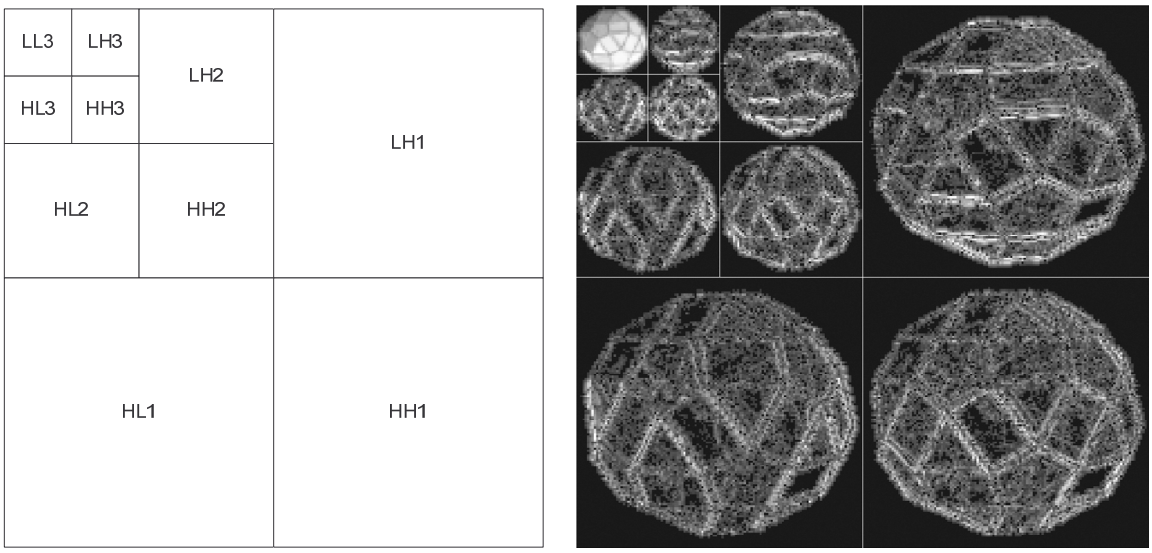

Figure 2: Frequency domain (wavelet coefficients) in the form of a decomposition image. Three-level image decomposition is presented.

3 SPIHT coding scheme

When the decomposition image is obtained, we try to find a way how to code the wavelet coefficients into an efficient result, taking redundancy and storage space into consideration. SPIHT [2] is one of the most advanced schemes available, even outperforming the state-of-the-art JPEG 2000 in some situations.

The basic principle is the same; a progressive coding is applied, processing the image respectively to a lowering threshold. The difference is in the concept of zerotrees (spatial orientation trees in SPIHT). This is an idea that takes bounds between coefficients across subbands in different levels into consideration. The first idea is always the same: if there is an coefficient in the highest level of transform in a particular subband considered insignificant against a particular threshold, it is very probable that its descendants in lower levels will be insignificant too, so we can code quite a large group of coefficients with one symbol.

SPIHT makes use of three lists – the List of Significant Pixels (LSP), List of Insignificant Pixels (LIP) and List of Insignificant Sets (LIS). These are coefficient location lists that contain their coordinates. After the initialization, the algorithm takes two stages for each level of threshold – the sorting pass (in which lists are organized) and the refinement pass (which does the actual progressive coding transmission). The result is in the form of a bitstream. Detailed scheme of the algorithm is presented in Fig. 3.

The algorithm has several advantages. The first one is an intensive progressive capability – we can interrupt the decoding (or coding) at any time and a result of maximum possible detail can be reconstructed with one-bit precision. This is very desirable when transmitting files over the internet, since users with slower connection speeds can download only a small part of the file, obtaining much more usable result when compared to other codec such as progressive JPEG. Second advantage is a very compact output bitstream with large bit variability – no additional entropy coding or scrambling has to be applied. It is also possible to insert a watermarking scheme into the SPIHT coding domain [3] and this watermarking technique is considered to be very strong regarding to watermark invisibility and attack resiliency.

But we also have to consider disadvantages. SPIHT is very vulnerable to bit corruption, as a single bit error can introduce significant image distortion depending of its location. Much worse property is the need of precise bit synchronization, because a leak in bit transmission can lead to complete misinterpretation from the side of the decoder. For SPIHT to be employed in real-time applications, error handling and synchronization methods must be introduced in order to make the codec more resilient.

SPIHT also offers several possibilities for processing color information (CSPIHT [4] is believed to produce best results) and can marginally be found in digital video or 3D image processing (3D-SPIHT).

4 MATLAB implementation of the algorithm

We have proposed an implementation of DWT-SPIHT coder and decoder for demonstration purposes. This coder is limited to images of square resolutions of power of 2 and grayscale (256 levels, that is 8 bit per pixel) information only. It produces a reconstructed output in spatial domain for comparison and also automatically computes PSNR (Peak Signal to Noise Ratio) as a measure of the difference between source and compressed (destination) images. Input parameters are: source image, wavelet name, transform depth and requested algorithm efficiency in bits per pixel (bpp).

From wavelet name we recognize whether it is a part of MATLAB wavelet toolbox and use the appropriate wavelet toolbox functions [5] then, or employ a self-made CDF9/7 lifting scheme implementation (which is widely available). Both transforms produce the same decomposition image of specified depth.

SPIHT coder is applied then, processing the exact number of bits computed from the bpp quantifier. Resulting bitstream is taken as a source for the decoder, that produces a reconstructed image. The result is then compared with the source image using standard MATLAB functions and methods.

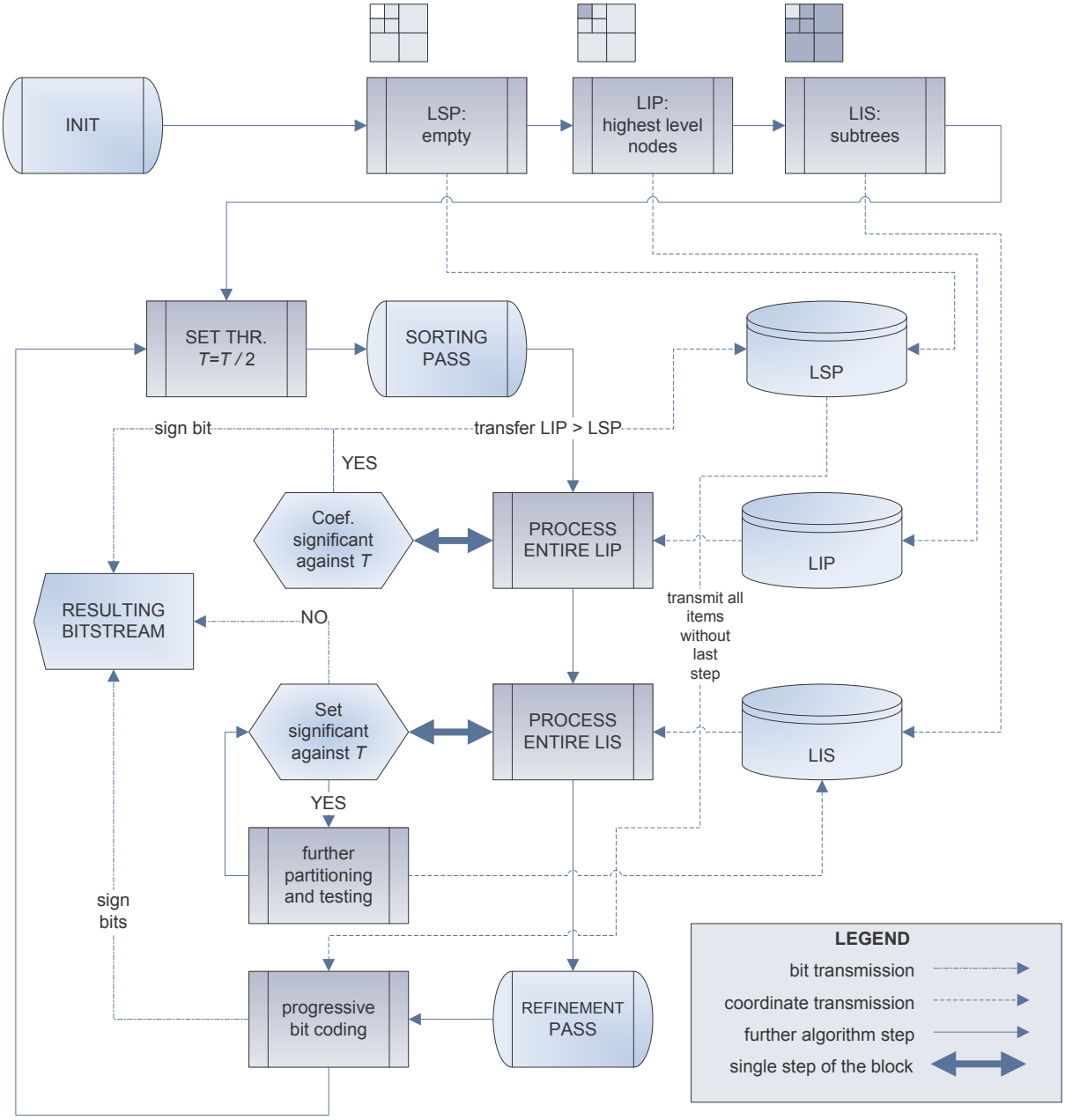

Figure 3: SPIHT algorithm scheme

5 Simulation results

We conducted several simulations on a group of images in order to test the influence of the coder settings. The results are mostly measured via PSNR as described above. PSNR is computed from the mean square error, obtained from the difference of source and processed image, and defined in dB. Unless otherwise noted, all tests are performed using the well-known "Lena" image with dimensions 256×256 pixels.

At first we believe it is a good idea to try various wavelet families and see their influence. The measurements were performed for a decomposition of depth 5, using several bit per pixel final quality settings. The results are presented in Table 1. We also try to describe final image quality by subjective terms. The results confirm what has been expected; mostly the generally introduced fact that CDF9/7 filterbank produces best results in image compression. The only real choice that produces comparable result are the very similar biorthogonal wavelet families (called 'bior' in MATLAB).

Table 1: COMPRESSION TESTING USING VARIOUS WAVELET FILTERS

wavelet	0.3bpp	0.5bpp	0.7bpp	0.9bpp	subjective result
haar	27.09	29.62	31.52	33.26	block artifacts
db2	27.98	30.70	32.56	34.55	blurry
db6	28.36	31.15	33.00	35.00	grooved
db15	28.08	30.70	32.56	34.40	very grooved
bior1.5	26.52	28.74	30.85	32.45	extensive block artifacts
bior3.1	25.24	28.01	30.60	31.74	very blurry
bior4.4	28.93	31.74	33.75	35.68	slightly blurry
bior6.8	28.93	31.80	33.82	35.71	good
rbio6.8	28.43	31.30	33.12	35.16	slightly blurry
cdf9/7	29.43	32.07	34.22	36.03	excellent

Based on the previous observation, we use only CDF9/7 filterbank implementation for further testing. In the next step we try to discover the influence of level settings on various bpp scales. As can be seen from Fig. 4, the wavelet decomposition depth is an important criteria that really matters. Lower level settings cause the spatial orientation trees to be handicapped by low algorithm depth. This is interesting due to the fact that three-level decomposition is usually enough for other image compression methods based on DWT. This simulation revealed a fact that SPIHT is most efficient when using level 5 and higher. The difference between level 3 and 4 at 0.3bpp is huge, especially in terms of subjective means, as level 3 resulting image is almost unusable.

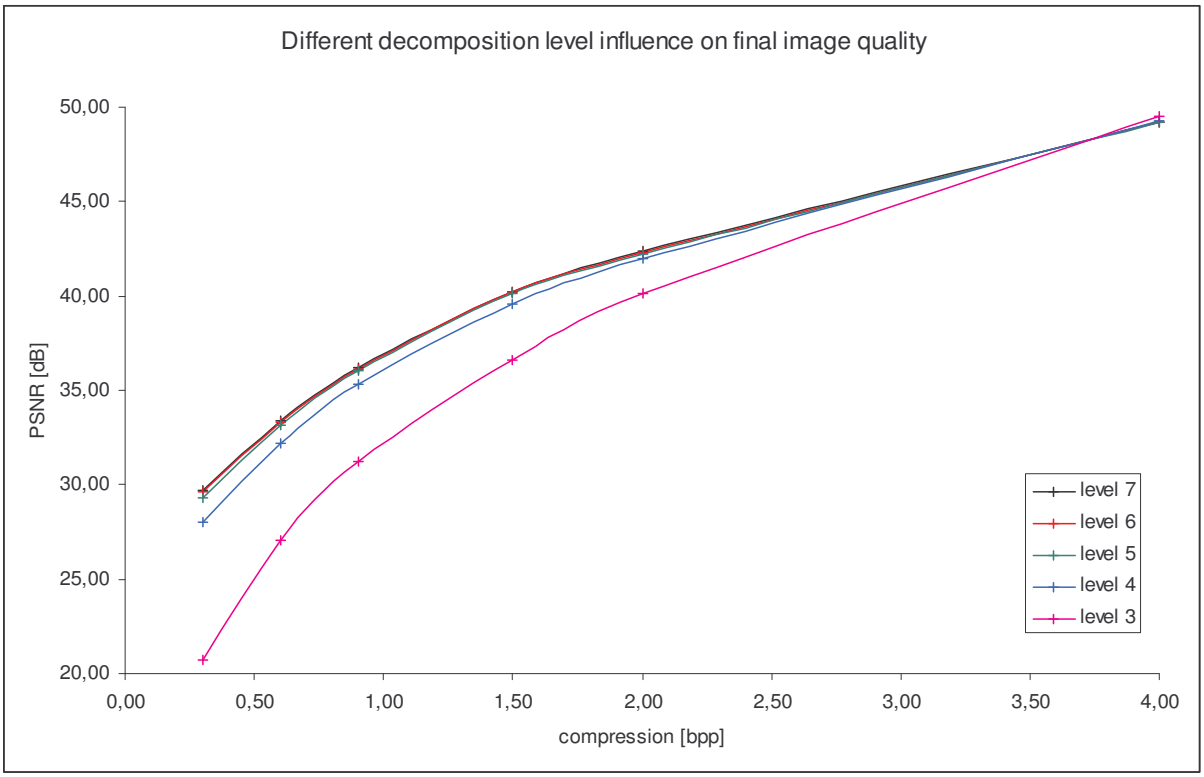

Figure 4: A wavelet decomposition depth influence on final image quality. Testing performed on Lena image 256×256 px, CDF9/7 wavelet filters.

It is also important to understand how image size affects the compression. At the following test we try to compress certain image sizes of Lena with varying bpp at level 5. Results are shown in Figure 5.

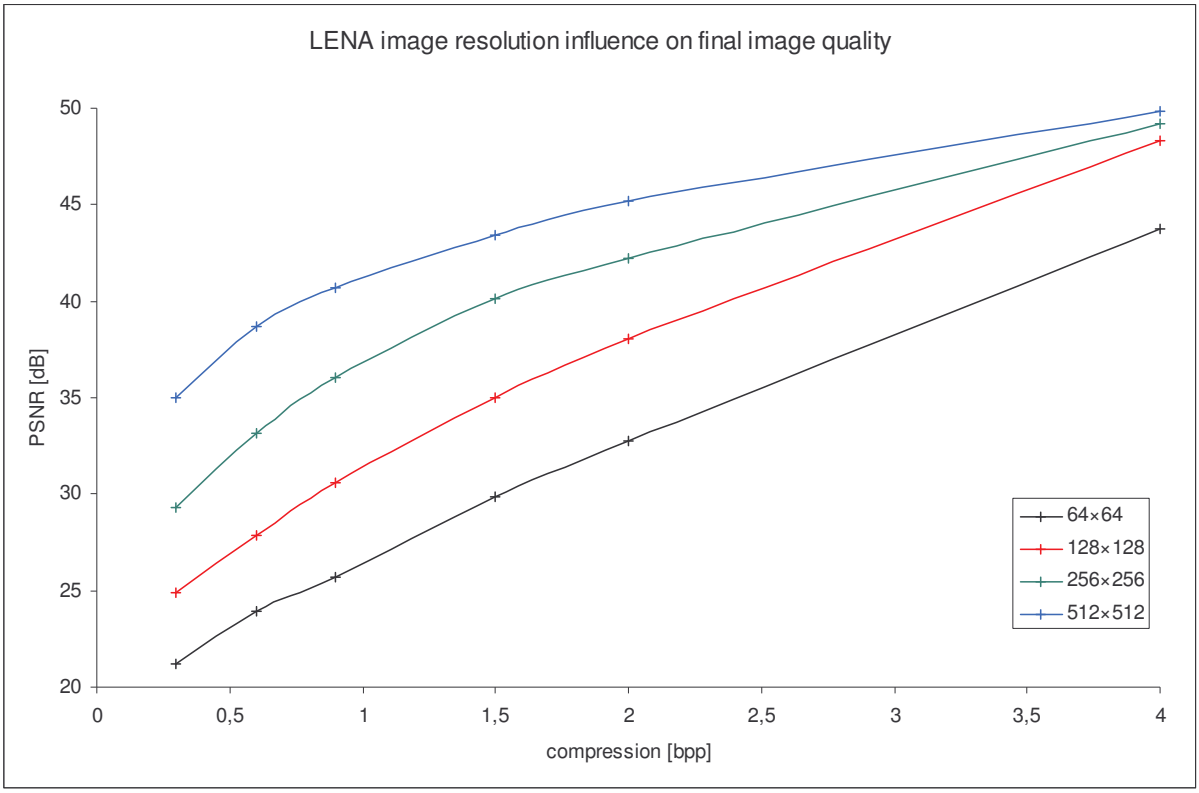

Figure 5: Image resolution influence on final image quality

It is obvious that higher image resolution leads to better quality results as the tree structures hold more detail. Also, for every resolution a different transform depth is desirable. This is very important due to the fact a square of fixed size is usually used as a processing segment for compression of this type, so we have to decide between quality and size of the block carefully.

Last simulation was aimed to estimate algorithm complexity. We simply measured the time (in seconds) necessary to encode and decode the image respective to certain depth and quality settings. Testing was again performed on Lena image 256×256 using CDF 9/7, see Table 2. From the results it is obvious that complexity of SPIHT is very unpredictable, as the results vary a lot. This is probably due to the nature of the algorithm that relies on a concept of predicted tree structures. The complexity is dependent on the source image and decomposition depth and its not easily determinable. Encoding times are noticeably higher than decoding as a descendant checking scheme must be applied.

Table 2: ALGORITHM COMPLEXITY (TIME ELAPSED IN SECONDS) BASED ON DIFFERENT SETTINGS

	0.3bpp		**0.6bpp**		**0.9bpp**	
level	**encode**	**decode**	**encode**	**decode**	**encode**	**decode**
2	3.36	3.08	7.47	4.86	9.02	5.39
3	1.81	0.38	3.83	0.80	6.14	2.47
4	2.00	0.38	5.08	2.19	8.63	5.64
5	2.11	0.67	5.25	2.81	10.06	6.98
6	2.86	0.66	5.53	3.02	10.08	7.14
7	1.13	0.75	5.51	3.14	10.36	7.53

As last part of the article we would like to present three different examples of the output. CDF 9/7 was used with wavelet decomposition of depth 5. Lena image (Fig. 6), low-detail Scenery image (Fig. 7) and high-detail City image (Fig. 8) are shown below. All of these images have dimensions of 512×512 pixels and were tested by 0.3bpp compression (which means generated file size is 27 times smaller that the original). Notice that subjective quality of reconstructed images is very good even though compression ratio is set noticeably high. This fact confirms the exceptional SPIHT efficiency.

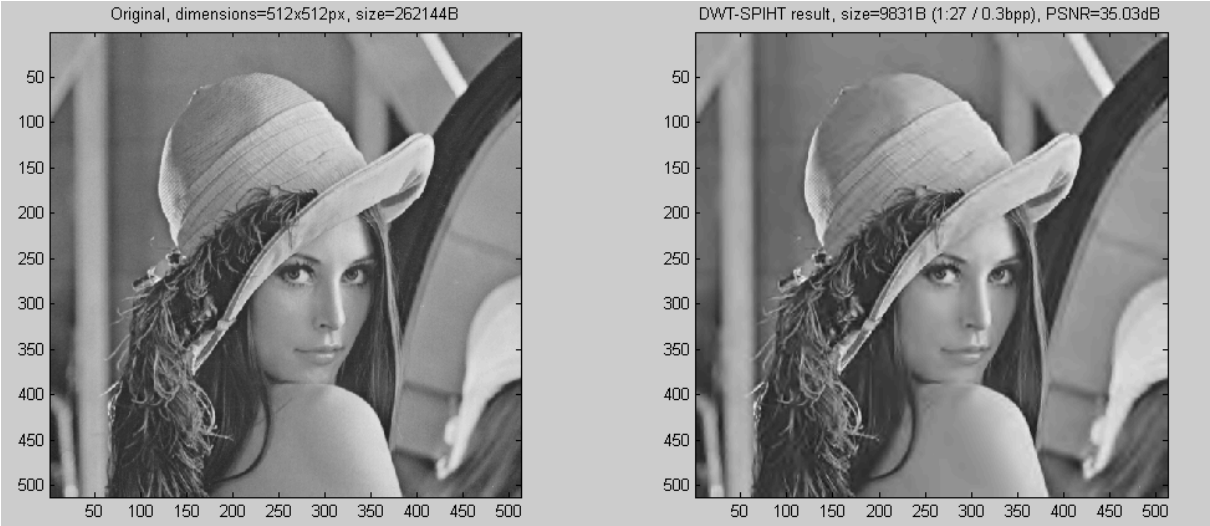

Figure 6: Lena image compression result at 0.3bpp

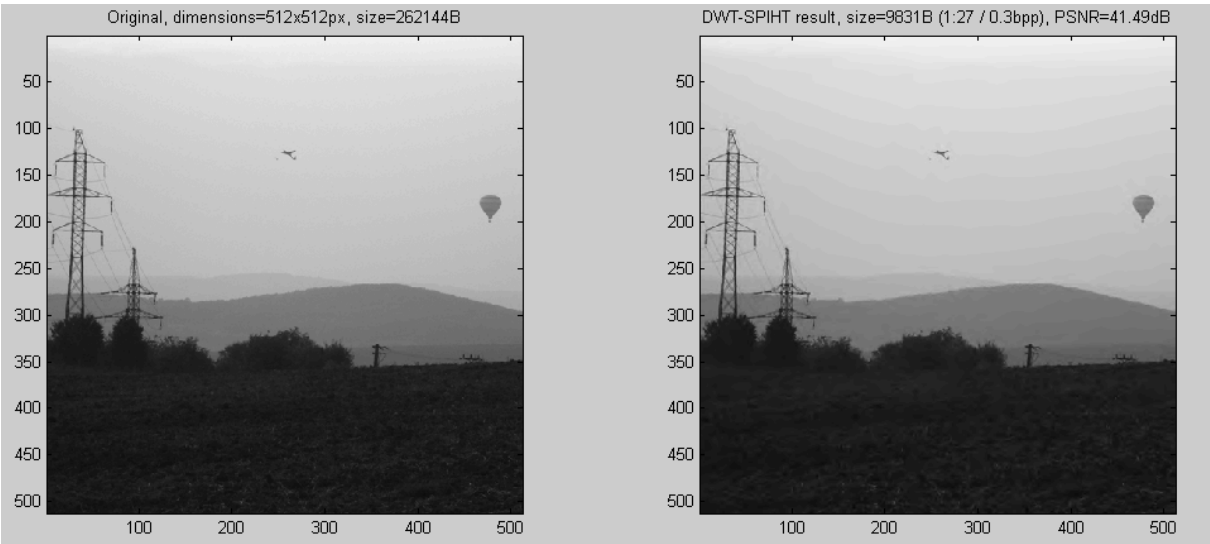

Figure 7: Scenery image compression result at 0.3bpp

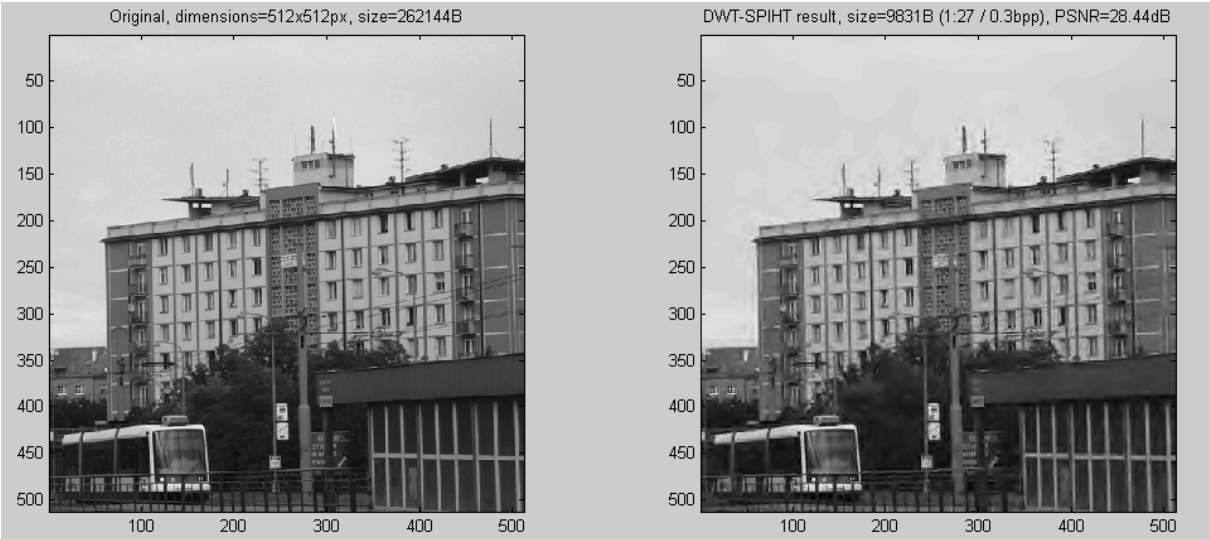

Figure 8: City image compression result at 0.3bpp

6 Conclusion

We have demonstrated the outstanding efficiency of the DWT-SPIHT image compression solution. Our simulations confirm the proposed ideas of proper wavelet filters usage. The fact that CDF9/7 were able to perform best results among all tested wavelet families and types has been expected. It is interesting to observe how the decomposition depth influences the final image quality. We have also measured algorithm complexity in terms of elapsed compression time and we found that SPIHT coding results are very unpredictable in that sense.

In the future much more effort must be emerged in order to make the codec more resilient against bit or synchronization errors, which should be quite a challenging task in order to keep the embedded character of the bitstream together with the progressive behavior.

All MATLAB source codes along with City and Scenery testing images are available at WWW[1].

Acknowledgements

The paper was prepared within the framework of 102/07/1303 project of the Grant Agency of the Czech Republic, project FRVS 1442/2007/F1a, project 1E1850015 of Grant Agency of Academy of Sciences of the Czech Republic and project MS1850022.

References

[1] A. Jensen, A.la Cour-Harbo: *Ripples in Mathematics*, Springer, 2001.
[2] A. Said, W.A. Pearlman: *A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, 1996.
[3] S. H. Yang, Y. L. Chang, H. C. Chen: *A Digital Watermarking Scheme Based on SPIHT coding*, IEEE International Conference on Multimedia and Expo (ICME'01), pp. 441-444, 2001.
[4] A. A. Kassim, W. S. Lee: *Embedded Color Image Coding Using SPIHT With Partially Linked Spatial Orientation Tree*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, pp. 203-206, 2003.
[5] M. Misiti, Y. Misiti, G. Oppenheim, J.M. Poggi: *Wavelet Toolbox for use with MATLAB®*, MathWorks, 2002.

Ing. Jan Malý
Department of Telecommunications, Brno University of Technology
Purkyňova 118, 612 00 Brno, Czech Republic
tel.: +420 541 149 217
email: xmalyj05@stud.feec.vutbr.cz

Mgr. Pavel Rajmic, PhD.
Department of Telecommunications, Brno University of Technology
Purkyňova 118, 612 00 Brno, Czech Republic
tel.: +420 541 149 166
email: rajmic@feec.vutbr.cz

[1] http://wavelets.triablo.net/spiht/