Learning an Adaptive Dictionary Structure for Efficient Image Sparse Coding

Jérémy Aghaei Mazaheri, Christine Guillemot, Claude Labit

LEARNING AN ADAPTIVE DICTIONARY STRUCTURE FOR EFFICIENT IMAGE SPARSE CODING

Jérémy Aghaei Mazaheri, Christine Guillemot, and Claude Labit
INRIA Rennes, France
jeremy.aghaei_mazaheri@inria.fr, christine.guillemot@inria.fr, claude.labit@inria.fr

***Abstract*—We introduce a new method to learn an adaptive dictionary structure suitable for efficient coding of sparse representations. The method is validated in a context of satellite image compression. The dictionary structure adapts itself during the learning to the training data and can be seen as a tree structure whose branches are progressively pruned depending on their usage rate and merged into a single branch. This adaptive structure allows a fast search for the atoms and an efficient coding of their indices. It is also scalable in sparsity, meaning that once learned, the structure can be used for several sparsity values. We show experimentally that this adaptive structure offers better rate-distortion performances than the "flat" K-SVD dictionary, a dictionary structured in one branch, and the tree-structured K-SVD dictionary (called Tree K-SVD).**

***Index Terms*—Dictionary learning, sparse representations, image coding.**

I. INTRODUCTION

Sparse representations consist in representing a signal y as a linear combination of a few columns, called atoms, from a dictionary matrix $D \in \Re^{n \times K}$. D contains K atoms and is generally overcomplete. The approximation of y can thus be written $y \approx Dx$, where $x \in \Re^K$ is sparse, i.e. has only a few non-zero coefficients. The dictionary D can be either predefined or learned from training signals, to get a dictionary adapted to a specific training set.

Sparse representations, as well as dictionary learning, can be used for denoising [1] or super resolution but we focus here on the problem of dictionary learning for efficient coding. This issue has been treated in [2], [3] or [4] for example, on face images, however with a different dictionary construction method. In this context, each signal y is transformed in a sparse signal x which is transmitted by coding a few pairs of atom indices and coefficient values. The choice of the dictionary is a crucial issue and one wants the dictionary offering the best trade-off between coding cost and approximation error.

With this in mind, the idea of using a structured dictionary for sparse representations has been explored in [5] with the concept of ITD (Iteration-Tuned Dictionary). A pursuit algorithm, such as OPM (Orthogonal Matching pursuit) [6], uses a different dictionary at each iteration. Each dictionary is learned from the residuals of the previous iteration. Then, this concept has been used with Tree K-SVD [4] which learns a tree-structured dictionary, each dictionary in the structure being learned with K-SVD [2] from a subset of residuals of the previous level. Such tree-structured dictionaries allow fast search and efficient coding of the sparse representations since each dictionary can be kept small while having a total number of atoms sufficient for a good trade-off between sparsity and approximation error. However, increasing the number of levels in the tree to select more atoms in it makes the total number of atoms explode, leading to a large storage footprint. The deep levels may suffer from over-fitting and lose efficiency compared to the first levels.

This paper describes a new adaptive dictionary structure for sparse representations, aiming to correct the weaknesses of the tree structure. As for Tree K-SVD, each dictionary in the structure is learned with K-SVD from a subset of residuals of the previous level. But the structure of the dictionary is different and adapts itself to the training data. Indeed, the branches in the structure are progressively pruned, according to their usage rate, and merged into a unique branch. This adaptive structure enables the learning of more levels than the tree structure while keeping the total number of atoms reasonable. It also enables learning more atoms than a "flat" dictionary while keeping the coding cost of the index-coefficient pairs similar. Besides, this structure is capable to represent the data better than a structure composed of only one branch of dictionaries, especially if the dictionaries in the branch are small. The adaptive structure is compared to the "flat" structure, the one-branch structure and the Tree K-SVD structure, each structure using K-SVD to learn the dictionaries in it, and is shown to outperform the other structures in terms of R-D performances.

In Section 2, Tree K-SVD, learning a tree-structured dictionary, is presented. The adaptive structure and its advantages compared to the tree structure are then explained in Section 3. We compare the adaptive structure to the other classical structures in Section 4. Finally, in Section 5, we conclude and present future perspectives.

II. THE TREE STRUCTURE

The tree structure (Fig.1) is composed of one dictionary of K atoms at the first level, K dictionaries of K atoms at the second level, K^2 dictionaries at the third, and so on. We have presented in [4] a tree-structured dictionary learning method, called Tree K-SVD. Each dictionary in the tree is learned using K-SVD [2] with a sparsity of one atom, on a subset of residuals from the previous level.

The dictionary at the first level is learned on all the training data. Then, each vector of the training data is approximated with one atom from this dictionary by the OMP (Orthogonal Matching Pursuit) [6] algorithm. Thus, residuals R_2 are computed, and then split in K sets of residuals, according to the atom selected at the first level. Each set of residuals $R_{2,k}, k \in [1, ..., K]$ is then used to learn a dictionary at the second level. This procedure is applied the same way to learn the next levels in the tree. Note that some dictionaries can be incomplete or even empty in the structure if the corresponding atom at the previous level is not used in any approximation. The learning of the branch is stopped when this case happens.

To use the tree-structured dictionary for sparse representations, one atom is selected per level, beginning by the first level. The input vector is approximated by one atom of the dictionary at the first level, with a coefficient, using OMP. The residual vector is then computed and approximated by an atom from a dictionary of the second level, and so on. The dictionary used at a given level is indicated by the atom chosen at the previous level in the approximation. We have also presented in [4] an alternative way to select the atoms along the tree, called Adaptive Sparse Coding (AdSC). This methods allows selecting more than one atom per level, to increase the quality.

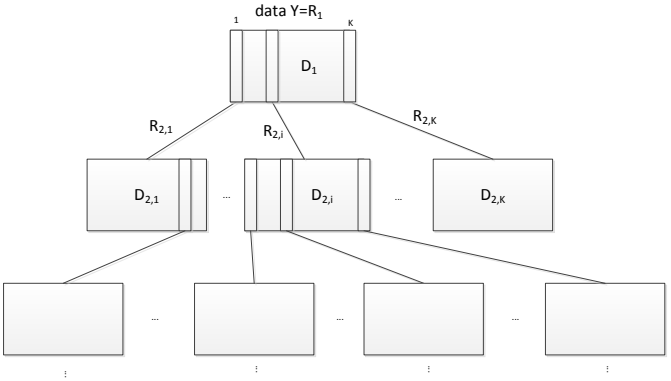

Fig. 1. The tree structure

Composed of small dictionaries, the tree structure offers an efficient coding of the atom indices and a fast search for the atoms along the tree, equivalent to the search in a "flat" dictionary of K atoms. It is also scalable in sparsity in the sense that it is learned for a given number of levels and can be used for any approximation with a number of atoms inferior or equal to the number of levels.

Nevertheless, if a better quality is needed, more atoms have to be selected in the tree. But increasing the number of levels in the tree makes the total number of atoms in the tree quickly increase. An over-fitting problem appears because the deep levels become too adapted to the training data. The too numerous dictionaries are more and more empty and lose efficiency. And the residuals are not structured anymore and can be seen as noise.

This is why it is necessary to learn a structure that adapts itself to the learning data, a structure whose number of levels can increase to select more atoms in it, while keeping a reasonable dictionary size.

III. The proposed adaptive structure

We present in this paper a novel adaptive dictionary structure (Fig.2). This adaptive structure is learned with a top-down approach level after level. Each dictionary in the structure is learned with K-SVD [2] with a sparsity of one atom. As for Tree K-SVD, the only dictionary at the first level is learned on the training data. The data are then approximated by one atom of the dictionary to compute the residuals R_2, which are split in K sets of residuals. For each set $R_{2,k}, k \in [1, ..., K]$, if the number of residuals vectors in it is enough to learn a complete dictionary, i.e. if the number of residual vectors is superior or equal to K, a dictionary is learned. Otherwise, the dictionary is not learned and the set of residuals is saved. At the end of the learning of the second level, all the saved sets of residuals are fused in Rm_2 to learn a "merged dictionary" Dm_2, the first dictionary of the "merged branch" of the structure. The same procedure is applied to the dictionaries of the second level to learn the dictionaries of the third one. The saved sets of residuals from this level, which have not been used to learn any dictionary at the third level, are fused together with the residuals from the "merged dictionary" at the previous level in Rm_3, to learn the "merged dictionary" at the current level Dm_3. That way, the structure is learned until a given level. With this method, the branches with a high usage rate, i.e. the branches where many residuals go, will be well developed whereas the branches with a low usage rate will be quickly pruned and their residual fused to the "merged residuals" to learn the "merged branch". Thus, during this learning process, the structure adapts itself to the training vectors.

The adaptive structure is then used the same way than the tree

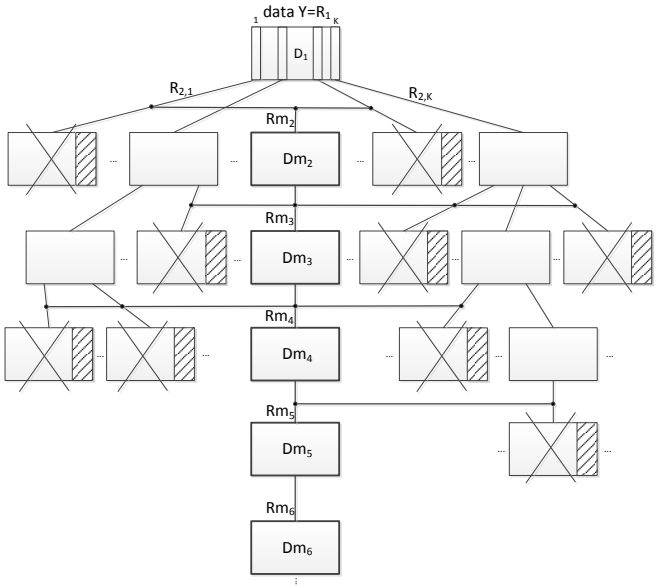

Fig. 2. The adaptive structure

structure for sparse representations. One atom is selected per level, starting with the first level. The difference occurs when we reach the end of a branch: instead of stopping the search, we can, with the adaptive structure, continue to select one atom per level among the "merged branch" of the structure. Thus, more atoms can be selected to reach a better quality of representation.

The adaptive structure presents the same advantages than the tree structure. As it is composed of small dictionaries, coding the atom indices is efficient because which dictionary to use has not to be coded. In the same way, it is fast to search for the atoms. Finally, it is also scalable in sparsity. Besides, the adaptive structure allows the learning of a deeper structure with more levels than in a tree structure, while keeping the dictionary size reasonable, because the branches are progressively pruned. That way, there is no over-fitting. Moreover, the branches are pruned just before reaching an incomplete dictionary, so all the dictionaries in the structure are complete and the deep levels stay efficient, contrarily to the tree structure.

IV. Experiments on satellite images

A. Experimental setup

To show the performances of the adaptive structure, we compare it with the classical structures, with the same dictionary learning algorithm to learn each dictionary in the structures: K-SVD. So the adaptive structure is compared to the Tree K-SVD structure [4], to a structure than can be seen as a tree of dictionaries of only one branch that we will call Branch K-SVD, and to the "flat" dictionary learned with K-SVD [2]. The predefined DCT dictionary is also added to the comparison to evaluate the difference with learned dictionaries.

The dictionary structures are learned on 13 satellite images, representing various scenes (cities, sea, desert, fields...). Patches of 8x8 pixels are extracted to obtain 654 688 learning vectors. The dictionaries are initialized with the DCT dictionary. 50 iterations are used to learn the K-SVD "flat" dictionary. For the structures of several levels, 50 iterations are used at the first level, and 10 iterations at the next levels. The test picture is "New York" (Fig.3), a 2400x2400 picture with both uniform and textured areas. It corresponds to 90 000 test vectors extracted from 8x8 patches.

In the context of compression, we do not intend to compare the methods with the same total number of atoms, but with an equivalent

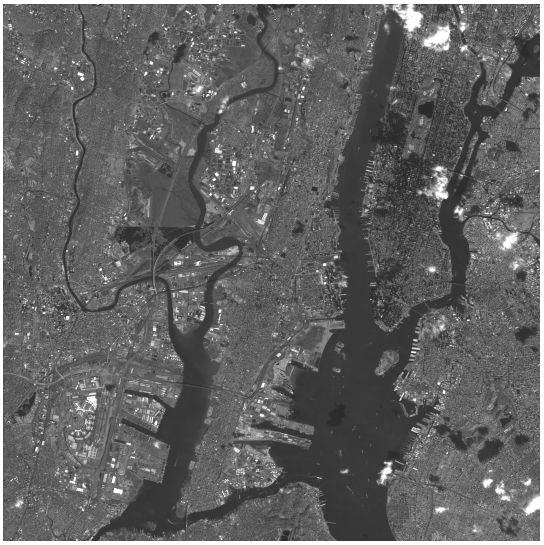

Fig. 3. Test image : New York (2400x2400)

coding cost of the atom indices. So for the "flat" dictionaries, K represents the total number of atoms, whereas for the structured dictionaries, K is the number of atoms per dictionary in the structure.

B. Analysis in function of the sparsity

To compare the structures, the test image is reconstructed for several sparsity values using the learned dictionaries (Fig.4,5).

We first notice that the predefined DCT dictionary is clearly below the learned dictionaries in terms of quality of reconstruction. The tree structure, Tree K-SVD, shows good results when a few atoms are used in the representation, but the PSNR stops increasing when we search for the atoms deeper in the structure. Indeed, as the learning of a branch in the tree is stopped after an incomplete dictionary, many branches cannot reach 10 levels and so 10 atoms cannot be selected. Using the AdSC allows selecting more than one atom per level, and so reaching the 10 atoms, but too many atoms are selected in an incomplete dictionary at the end of a branch, before the 10th level. Hence a loss of efficiency of the tree structure appears when we increase the number of atoms. That is why K-SVD, which is optimal because a dictionary is learned for each sparsity value, becomes better than Tree K-SVD with AdSC. The structure composed of only one branch shows good results, especially for large dictionaries (K=256), however always below the results of the adaptive structure. Indeed, the larger a dictionary is, the better it represents the data, but it implies also a bigger coding cost of each index. The adaptive structure offers a good representation of the data even with small dictionaries in the structure, hence a bigger advantage for K=64. It corrects the weaknesses of the tree and the branch structures by automatically adapting its structure between a branch and a tree of dictionaries and thus outperforms the other structures.

C. Coding analysis

In a context of image coding, the atom indices and their associated coefficients have to be coded for each block of the test picture. The coefficients are quantized by a uniform quantizer with a dead zone and ordered in a sequence, an "End Of Block" (EOB) code delimiting each block. The sequence is then coded thanks to a Huffman entropy coder, similarly to the JPEG coder and with the same table. Finally, the rate of the indices is added with a fixed-length code. For each index, the rate is given by $R = log_2(K)$. By sweeping through various quantization steps, we obtain the R-D comparisons (Fig.6).

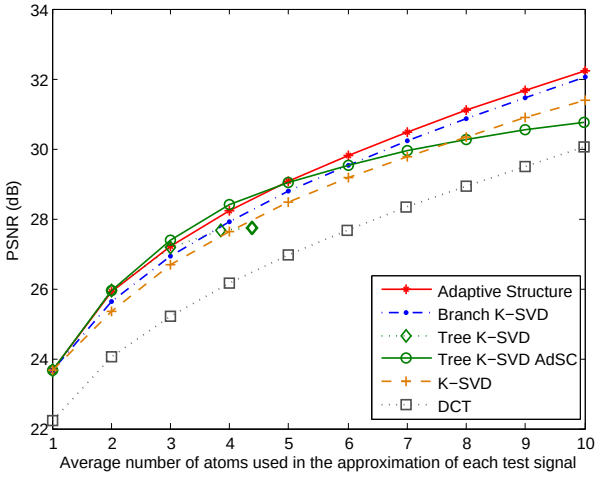

Fig. 4. Sparse representations for K=64

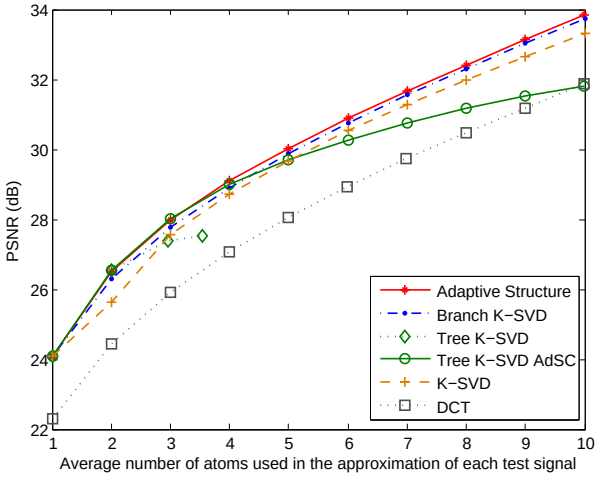

Fig. 5. Sparse representations for K=256

The quantization regulates the number of atoms used for each block. Indeed, the dead zone is function of the quantization step and puts to zero any coefficient strictly inferior to half of the step. Thus, according to the value of the quantization step and the coefficients of the block, more or less coefficients and indices are coded for each block, with a maximum set to 10.

Similarly to the PSNR-sparsity tests, the adaptive structure shows the best R-D performances (Fig.6). The tree structure gives good results at low bitrate but is then outperformed by K-SVD when the rate increases. The one-branch structure presents close results to the adaptive structure when the dictionaries in the structure are big enough to represent the data, thus the advantage of the adaptive structure is clearer for small dictionaries (+0.19dB at 1.64bpp).

We then compare the results of the adaptive structure for K=256 with JPEG 2000 (Fig.7), using the OpenJPEG software [7] . JPEG 2000 is a compression algorithm close to the one currently on board of satellites. We notice that JPEG 2000, an optimized image coder, offers better performances, thanks to an efficient entropy coding with the EBCOT (Embedded Block Coding with Optimal Truncation). Our method is penalized by too simple methods used to code the indices and the coefficients. Indeed, fixed-length codes are used for the indices, which require an important rate (Fig.7). For the coefficients, a coding similar to the one used for the AC coefficients in JPEG is applied, with the same predefined table. That is why we propose to learn Huffman tables for the coefficients and the indices to improve

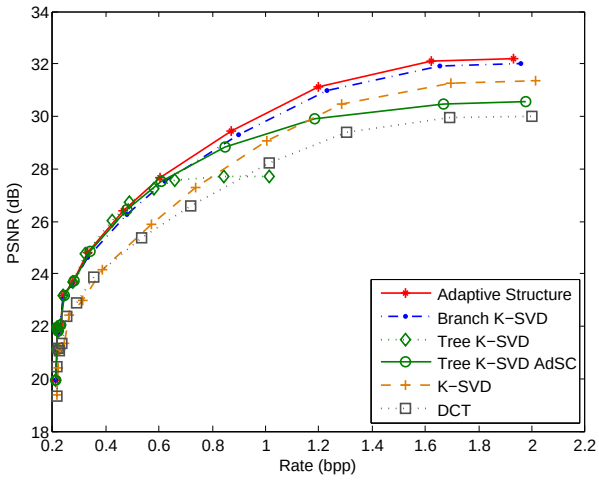

Fig. 6. Rate-distortion curves for K=64

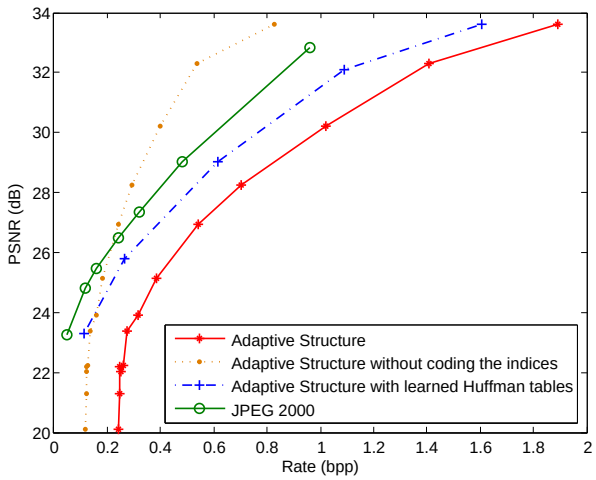

Fig. 7. Rate-distortion curves for K=256 with blocks 8x8

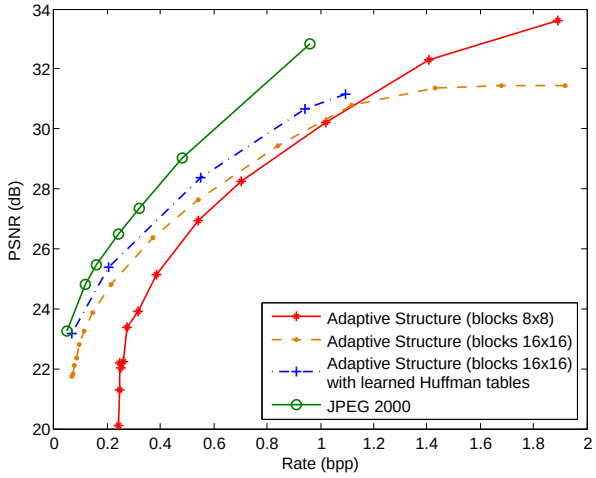

Fig. 8. Rate-distortion curves for K=256 with blocs 8x8 or 16x16

the rate. Two tables, one for the coefficients and one for the indices, are learned for each quantization step and for each level of the structure on the coefficients (and the corresponding indices) which are not put to zero by the quantization. For each Huffman table learned for the coefficients, an "EOB" code is also learned. Using these learned Huffman tables to code the coefficients and the indices significantly improves the rate (Fig.7).

Another option to improve the coding performances of our method is to increase the size of the blocks we work on. Until now, 8x8 blocks have been used, so each signal and each atom of the dictionaries had 64 values. By using 16x16 blocks instead, the dictionaries are able to learn bigger structures and a fewer number of atoms can be necessary to code the picture, particularly for homogeneous parts. But each atom in the dictionaries has 256 values, so the dictionary structures are larger. We learn an adaptive structure of 30 levels on 16x16 blocks of the same basis of satellite images and compare it with the structure of 10 levels learned on the 8x8 blocks. We show that using bigger blocks improves the rate-distortion results (Fig.8). However, when the rate increases, the adaptive structure for the 16x16 blocks is limited in quality and the PSNR stops increasing because 30 atoms maximum are selected for the decomposition per block 16x16, whereas this limit is set to 10 atoms per block 8x8 selected in the adaptive structure learned on 8x8 blocks. This limit has been set to 30 and not 40 atoms for a memory limitation reason. Besides, we notice again that using learned Huffman tables improves the results.

V. Conclusion

We have presented in this paper a method to learn a new adaptive dictionary structure in order to use sparse representations in an image coding context. This structure compensates the weaknesses of the tree and the branch structures, particularly for small dictionaries. We have shown that in a coding context, this adaptive structure outperforms the "flat" K-SVD dictionary, the Tree K-SVD structure and a branch structure. Besides, as the tree structure, the adaptive structure is efficient to code the indices and allows a fast search for the atoms. It is also scalable in sparsity as it can be used for several sparsity values.

We presented at the end of the experiments some ways to improve the coding performances of our method. We believe that the entropy coding of the coefficients and the indices can still be improved. Moreover, we intend to integrate this structure in the standard High Efficiency Video Coding (HEVC) in intra mode, by replacing the DCT-like transform, to take advantage of the efficient predictions and entropy coding of HEVC in order to compare our method to standard coders.

ACKNOWLEDGEMENT

This work has been supported by EADS Astrium.

References

[1] M. Elad, and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries", *IEEE Trans. on Image Process.*, vol. 15, no. 12, pp. 3736-3745, 2006.

[2] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation", *IEEE Trans. on Signal Process.*, vol. 54, no. 11, pp. 4311-4322, 2006. "K-SVD matlab toolbox": http://www.cs.technion.ac.il/∼elad/software.

[3] O. Bryt, and M. Elad, "Compression of facial images using the K-SVD algorithm", *Journal of Visual Commun. and Image Represent.*, vol. 19, no. 4, pp. 270-282, 2008.

[4] J. Aghaei Mazaheri, C. Guillemot, and C. Labit, "Learning a tree-structured dictionary for efficient image representation with adaptive sparse coding", *ICASSP*, 2013.

[5] J. Zepeda, C. Guillemot, and E. Kijak, "The iteration-tuned dictionary for sparse representations", *Proc. of the IEEE Int. Workshop on MMSP*, 2010.

[6] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition", *Conference Record of the 27th Asilomar Conf. on Signals, Syst. and Comput.*, pp. 40-44, 1993.

[7] "OpenJPEG homepage": http://www.openjpeg.org.