THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2006 ISSN 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

CELLULAR AUTOMATA CODEBOOKS APPLIED TO COMPACT IMAGE COMPRESSION

Radu Dogaru[1], Manfred Glesner[2], *and* Ronald Tetzlaff[3]

[1]Department of Applied Electronics and Information Engineering, Polytechnic University of Bucharest, Romania, e-mail: radu_d@ieee.org,
[2]MES, Technical University of Darmstadt, e-mail: glesner@mes.tu-darmstadt.de
[3]J. W. Goethe-University, Frankfurt/Main, e-mail: R.Tetzlaff@iap.uni-frankfurt.de

Abstract: Emergent computation in semi-totalistic cellular automata (CA) is used to generate a set of basis (or codebook). Such codebooks are convenient for simple and circuit efficient compression schemes based on binary vector quantization, applied to the bitplanes of any monochrome or color image. Encryption is also naturally included using these codebooks. Natural images would require less than 0.5 bits per pixel (bpp) while the quality of the reconstructed images is comparable with traditional compression schemes. The proposed scheme is attractive for low power, sensor integrated applications.

Keywords: cellular automata, emergent computation, image compression, binary vector quantization, nonlinear dynamics.

1. INTRODUCTION

Image compression is a necessary ingredient whenever limited bandwidth channels are available. During the years several techniques and standards have been developed (Sullivan, and T. Wiegand, 1998), (Ostermann *et al*, 2004). Lossy schemes are particularly popular when dealing with natural images because they exploit the human perception system to achieve large compression rates without sensitive degradation in quality. Traditional schemes of lossy image compression usually consider vector quantization (VQ) of image blocks (quite often 8x8 pixel blocks). Discrete cosine transform (DCT) or other transforms (e.g. Discrete Wavelet Transform (DWT) or Integer Wavelet Transform (IWT)) is also often used in traditional compression schemes, while there is a constant focus to reduce the hardware complexity of the implementations. For instance, in (Grangetto, *et al*., 2002) an optimized IWT transform architecture is reported as having a "modest gate complexity" of about 56000 gates (with an area of about $3\ mm^2$ for a 0.35 μm technology).

Both vector quantization and transform coding are computational intensive algorithms, therefore an important research effort is directed to define various circuit architectures requiring a minimum of hardware resources and power consumption. Most of the computational complexity is generated by the need of complex arithmetic operators (i.e. multipliers, adders, cosine evaluation etc.) to process the pixels, regarded as fixed-point numbers.

Recently, cellular automata (Wolfram, 1983) were considered as alternative solutions to traditional VQ and transform coding schemes. For instance, in (Lafe, 1997) a Cellular Automata Transform (CAT) is proposed and applied for image compression. The idea is to replace multiplications with bit Boolean operators and to use a flexible set of basis, which are

This paper was recommended for publication by Dorel Aiordachioaie

binary vectors generated by cellular automata dynamics.
Although CAT is regarded as a compact solution, a recent paper (Chen and Lai, 2004) reports a VLSI implementation of the CAT / ICAT system with a complexity of 6.8 mm^2 in a 0.35 μm technology, i.e comparable and even higher that for the circuit implementing the Integer Wavelet Transform in (Grangetto, *et al*., 2002).

In this paper we propose a different approach for efficient image compression, where the aim is the radical simplification of the circuitry, thus leading to low power and complexity implementations often demanded by autonomous sensor integrated systems. Our estimate for the hardware complexity indicates 5 to 20 times less resources than any of the methods (Grangetto, *et al*., 2002) or (Chen and Lai, 2004) based on transforms.

The method resembles a binary vector quantization (BVQ) scheme applied to the bitplanes of the original image. Unlike in traditional BVQ schemes where computationally intensive optimization algorithms are employed to generate an optimal codebook, here we employ specially designed cellular automata (CA) to generate the codebook. The operating principle of our encoding and decoding scheme is presented in Section 2. The details of the cellular automata and their use for codebook generation are discussed in Section 3. A detailed description of the performances and limits of our proposed compression scheme is given in Section 4 for several representative images. Concluding remarks and considerations regarding the circuit complexity are given in Section 5.

2. THE PRINCIPLE OF THE CA BASED COMPRESSION METHOD

The main idea of our scheme is similar to Binary Vector Quantization i.e. each 8x8 (or generally, $w \times w$) bits block from a binary image is replaced with its closest codeword (in terms of Hamming distance) from a previously defined codebook with *C* code-words. The novelty is that we employ cellular automata to generate the codebook instead of using complex optimization algorithms operating on large training images sets, as shown in Fig.1. The result is a sort of a "universal" codebook, which can be easily tailored to various classes of images by simply changing the CA type (defined by its cell ID) and the number of iteration until reaching a state considered as the codebook.

The following procedure is used to generate the CA-based codebook: An initial binary state (here of 128x128 size containing a „key" (represented by the position of 3 "black" bits within the grid with all "white" state cells) is considered for a cellular automata (CA) with an ID code selected such that after a certain number of iterations (here 128, i.e. comparable with the size of the CA array) will evolve towards a pseudo-random binary pattern. In this example we considered semi-totalistic CA with 5 cells neighborhood (detailed in Section III) with ID=325, however other choices are equally possible, including other types of random number generators.

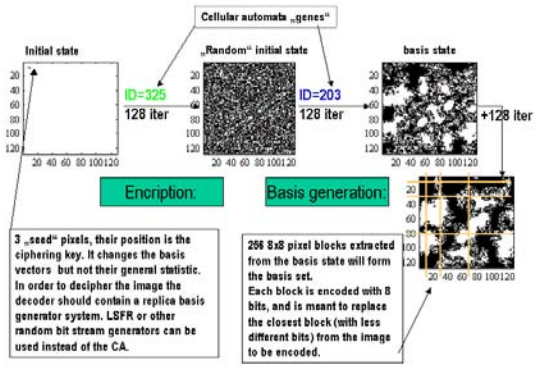

Fig.1. The principle of codebook generation using cellular automata

In order to ensure encryption, the same key should be considered in both encoding and decoding stages, where the above algorithm will be executed to generate identical codebooks. In addition to the advantage of a simple encryption algorithm there is no need to send the entire codebook but only the key.

The random binary state obtained as above is then considered as the initial state for a second CA, also run for a specific number of iterations until spatial patterns emerges, reminiscent of „pink noise" i.e. binary patterns containing a wide range of spatially frequencies. This pattern represents the codebook obtained by consecutively cropping a number of *C* bit-blocks of $w \times w$ size (each representing a codeword). In a simplified practical realization, one can directly store it in a local memory instead of computing it using the CA. Of course it is assumed that the codebook has been previously generated offline using the above algorithm.

In the example in Fig.1, ID=203 was considered for the CA, but as seen in section 3 other choices are possible. Note that running the CA for some additional iterations (128 in our example) allows slightly changes of the spatial frequency content and consequently the tuning of the resulting codebook. The operating principle of the encoder and the decoder employing the CA-generated codebook are presented in Fig2, and Fig.3 respectively.

The encoding process is simple: The input image is split into *m* bitplanes. For instance, if a pixel at

spatial coordinates *(x,y)* in the input image is represented using a *8*-bit binary code, this code is distributed among 8 bitplanes, such that bit "*m*" of the code is located in bitplane *m* on the same *(x,y)* coordinate.

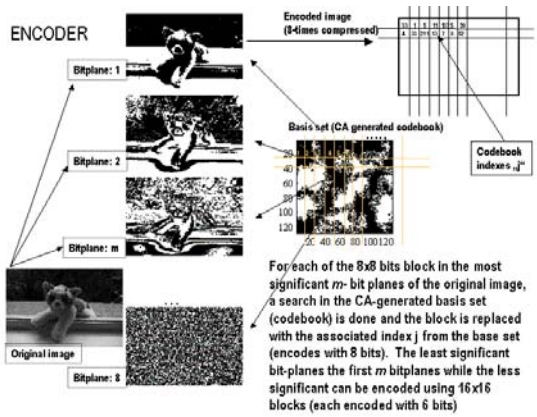

Fig.2. The principle of encoding (compression) using CA-generated codebooks.

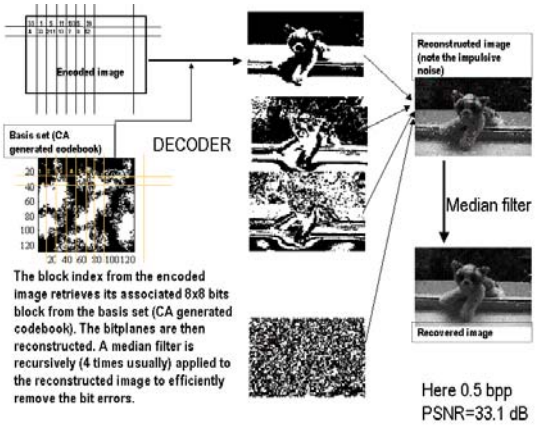

Fig.3. The principle of the decoding (decompression)

Experiments with many natural images indicate that the most significant bitplane (*m=1*) has the lowest content in high frequencies, while the least significant bitplanes look almost random. This observation led to the following simplification in the encoding algorithm:

- The first *m* most significant bitplanes are encoded as indicated next.
- The remaining least significant bitplanes are replaced with pseudo-randomly generated bitplanes. The pseudo-random number generator is defined by a key. The same key has to be used in the decoder stage to reconstruct the least significant planes. It was found that using this technique, the reconstruction error

Each of *m* most significant bitplanes is divided into *MxN* blocks of *wxw* bits size each. Then, for each such block $\mathbf{B}$, a search within the codebook is considered aiming to find the replacement codeword $\mathbf{C}_J$ such that $d_J = \|\mathbf{B} - \mathbf{C}_J\|_1$ is minimum, where $J \in \{1,..C\}$ is the index of the codeword. Instead of sending the entire original block the encoder will consider it replaced by $\mathbf{C}_J$ and therefore, assuming that the codebook can be reconstructed in the decoder, will send only the index $J \in \{1,..C\}$, represented with a reduced quantity of information. The result of this approximation can be measured for each plane as a *biterror Berr(m)* counting the fraction of erroneous bits in the *reconstructed image* among the entire size of a bitplane. As we will see in the decoder stage, a large part of this error can be efficiently removed using a median filter applied to the *reconstructed image*. Therefore the overall quality of the compression-decompression process is evaluated by computing the PSNR[1] expressed in dB of the *recovered image* obtained as a filtered version of the *reconstructed image*. A "bit-per pixel" (bpp) rate is usually computed to estimate the compression capacity of the algorithm.

Assuming a uniform distribution of the index codes, each index will be directly represented with $\log_2(C)$ bits. This is the case of the lowest implementation complexity CA-VQ algorithm.

In this case, the information required to represent a pixel (bits per pixel rate) is:

$$bpp = m\frac{\log_2(C)}{w^2} \quad (1)$$

For instance, if *m=4*, the compressed image is represented with 0.5 bpp if *C=256* and *w=8* as used in the examples herein.

However, entropic encoding (e.g. Huffman coding) of *J*, can substantially improve the coding efficiency for the price of adding a little implementation complexity (that of implementing a Huffman encoder). Indeed, as shown in Table I it turns out that for many typical images, the entropy $H(m)$ is much smaller than $\log_2(C)$, particularly for the most significant bitplanes.

In the next we will denote CA-VQ-E the algorithm with entropic encoding. In this case, the bit per pixel rate is computed as:

[1] Power signal to noise ratio, where the noise image is a difference between the original and the recovered image.

$$bpp = \frac{\sum_{j=1}^{m} H(m)}{w^2} \quad (2)$$

Table 1: Codeword entropies on different bitplanes

Bitplane m:	1	2	3	4	5	6	7	8
Codeword entropy H	2.3	4.3	5.4	6.8	7.5	7.6	7.6	7.6
Bit error Berr(m)	1.8	6.7	10.7	17.9	25.2	30	32.1	32.6

For m=4 and for the image shown in Fig. 1 (and considered in Table 1), the pixel rate results 0.2934 bpp, compared to 0.5 bpp obtained in the CA-VQ case (without entropic encoding).

Using a CA-generated codebook, the bit error on the first bitplane is only around 2%. This is quite impressive while only 256 codebook vectors are used to approximate any of the 2^{64} possible binary 8x8 bit blocks. Such a small bit error indicates the effectiveness of the CA-generated codebook. If a randomly generated codebook with similar size would be used instead, the bit error reaches 50% and the recovery of the original image would be impossible.

The decoder (Fig.3) has an even simpler principle. For each of the most significant *m* bitplanes, a wxw bit-block is reconstructed by simply using the index *J* to address a codebook which is supposed to be identical to the one used in the encoding process. The result is a *reconstructed image*. Usually such an image has a visual quality affected by a sort of impulsive noise, generated by the bit errors during the reconstruction process. The sparse bit errors, particularly for the most significant planes makes possible to recover the original image with a good accuracy by further employing a median filter. It was found convenient to apply a median filter recursively several times (usually no more than 4 times).

3. CELLULAR AUTOMATA AND CODEBOOK GENERATION

In cellular automata, computation is an *emergent phenomenon* (Dogaru, 2003), based on complex nonlinear dynamics (equilibrium, chaos, or "edge of chaos") in a *regular* lattice of any identical *cells*. A cell (Fig.4) communicates only with similar cells in its *neighborhood* and is characterized by a *gene* (i.e. a set of parameters describing the local function of the cell). The cell position within the CA grid is usually specified with a pair of indexes (i, j). However, for simplicity, in the following we will extensively use a relative notation with respect to the central cell (assigned the index k=1). Indexes from 2 to 5 are assigned to the other neighbor cells. Each cell output $y_1^{i,j}$ within the CA grid is binary (i.e. it has only two possible states, e.g. encoded as 0 and 1 numerically or black and white in graphical representations where a pixel is associated to each cell.

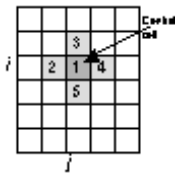

Fig.4. A typical CA lattice with 5 cells in a neighborhood (Von Neumann). Each cell in the neighborhood is assigned a index starting with 1 for the central cell.

Computation in cellular automata assumes a initial state $Y(0) = \{y_1^{i,j}(0)\}$, followed by a number *T* of iterations. During each iteration *t* all identical cells are synchronously updated such that:

$$y_1(t+1) = G(y_1(t), y_2(t), y_3(t), y_4(t), y_5(t)) \quad (3)$$

i.e the output of the cell is updated according to the content (outputs) of all cells in the local neighborhood at the previous iteration. The final state (the collection of all cell states $\mathbf{Y}(T)$ reached at iteration *T*) is a processed version of the initial state. The duration *T* and the specific cell function *G* (with a collection of parameters called a *gene* (Chua and Roska, 2001)), defines the nature of the transformation of the initial state. For binary cells and the 5-cell "von Neumann" neighborhood it is possible to have an "ocean" of $2^{2^5} = 2^{32} \cong 4.295 \cdot 10^9$ different cell genes.

A difficult task was identified (Dogaru, 2003), as the "design for emergence" i.e. a systematic procedure to identify the "emergent" cells in an ocean of cells leading to dull (computationally not meaningful) CA behaviors. In (Dogaru and Glesner, 2004) and (Dogaru, 2005) several methods were developed to automatically locate those meaningful (or emergent) genes within large numbers of possible CA. A first step was to consider *semi-totalistic* (Fig.5) instead arbitrary *Boolean* cells, thus reducing the huge search space from above to only 1024 distinct cells.

Within this search space, a large number of meaningful cells were discovered, some useful for image processing and pattern recognition applications (Dogaru *et al*, 2005), some for cryptographic purposes and others used to generate the CA-codebooks as shown next. Each of the 1024 cells is defined by an ID number, which is the

decimal transcription of the binary string defining the input-output relationship of the semitotalistic cell (Fig.5).

A list of IDs, suitable for good codebooks was determined using tools for detecting emergent phenomena (Dogaru, 2005). Note than for each of these ID, a different iteration number *T* will give different qualities of the recovered image.

In Table 2 we list the percentage of bit errors for the most significant bit plane $Berr(1)$ for all ID listed as potentially useful and for 4 different running times *T*. The initial state is a random one (with equal probability of 1 and 0 states), the same for all runs.

A small run time *T* gives a codebook with a lot of high frequency vectors while a large run time favors code-words with low frequencies. While an image contains blocks from a wide spectrum of spatial frequencies, there should be a optimal value of *T* as seen in the above table. The best (smallest) bit errors for each ID are presented in a shaded entry of the table.

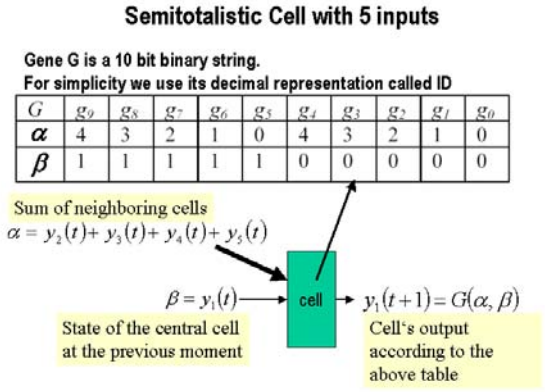

Fig.5. The structure of a semi-totalistic cell with 5 inputs. In such a cell the output (next state) depends only on the previous state β and on the *sum* of the remaining neighbor cells α.

Table 2: Bit errors (%) for different codebooks (a codebook is generated with a cellular automata defined by (ID,T))

	T=32	T=64	T=128	T=256
ID=131	4.15	3.87	3.81	4.12
ID=147	5.63	3.75	3.68	3.87
ID=171	9.37	5.92	3.68	3.84
ID=179	9.11	6.66	4.35	4.14
ID=187	8.74	8.19	4.15	3.99
ID=203	14.56	7.32	4.59	4.38
ID=404	17.2	20.15	20.2	20.1
ID=852	9.37	5.92	3.69	3.84

By further tuning the *T* parameter a slight improvement it is possible. For instance, in the case of ID=147, if *T*=115, a Berr(1)=3.55% is obtained instead of 3.68% which is the better value listed in the above table.

The choice of the random initial state does not affect significantly the bit errors for the same ID and this is the basis for employing the ciphering scheme proposed in Section 2. In our next experiments, we have considered a 128x128 size CA in generating the codebook. The codewords are 8x8 square blocks resulted by dividing the CA array into 16x16 such blocks.

4. RESULTS, COMPARISONS AND CIRCUIT COMPLEXITY

4.1. Comparison with other compression standards

In order to compare our scheme with other compression standards we have considered the "Lena" image. The codebook $(ID = 131, T = 60)$ was used, with a codeword of 8x8 size. Two codebooks were used: 1) one with 256 code-words (bases) generated as above, and a reduced set with only 8 representative bases, for a higher compression rate). The selection of those 8 representative bases from the total of 256 is done with an algorithm, which orders in a decreasing order of their Hamming distances so that the first bases have largest distances among them.

A rate-distorsion curve was determined for both CA-VQ and CA-VQ-E schemes, in both codebook cases. For reference, the same image was encoded in the JPEG standard using Matlab tools and the corresponding curve is plotted in Fig. 6. The pixel rates for CA-VQ correspond to *m* (selected significant bitplanes) ranging from 1 to 8.

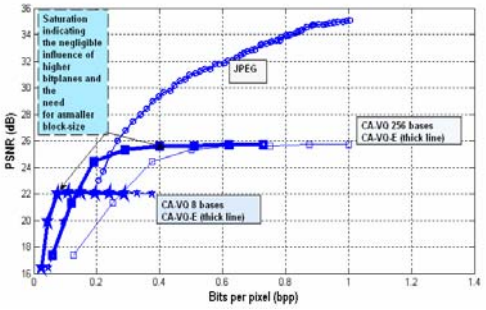

Fig.6. Rate-distorsion curve giving comparison between our method in either simple (CA-VQ) or with entropic encoder (CA-VQ-E) case and well established standards such as JPEG.

Several important conclusions result from the above graph: For either CA-VQ and CA-VQ-E there is a

saturation in quality, reached for $m \geq 3$ bitplanes. This indicates the insignificant ifluence of the less significant bitplanes but also that a smaller block size would be needed to increase the PSNR and avoid saturation. This will be considerd in further coding schemes with variable block-size. In comparison to JPEG both CA-VQ-E situations perform better until the saturation is reached. A better compression rate (with slight decrease in quality) is obtained for the case of 8 bases. Generally speaking, our scheme performs better than JPEG for very low compression rates, here the boundary is at about 0.2 bpp. As seen in Fig.7., by choosing a bigger block size (i.e. w=16) a very large compression rate (of about 150 times) is achieved with a reasonable quality of the reconstructed image. Such rates are not attainable with JPEG.

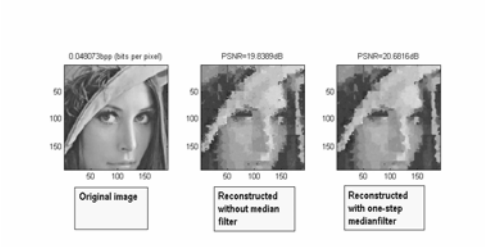

Fig.7. An example of high compression (0.055 bpp, entropic) using 16x16 code-words, 64 basis and m=4 bitplanes. For such high compression rates our scheme outperforms JPEG standards, without significant blocking effects.

The visual quality of Lena image for several of the bit rates considered herein (the case of 8x8 block-size and 256 bases) can be evaluated from Fig.8.

Perceptually, the image compressed with 0.41 bpp still looks close to the original. Unlike in many other compression schemes there are no blocking artifacts and the high frequency edges are nicely preserved. The perceptual quality slightly degrades for 0.29 bpp while the main features (like eyes, hat, hair, etc.) of the image are still recognizable in the recovered image after compressing to 0.048 bpp (almost 170 times).

Other images are considered in Fig. 9 to evaluate the response of the CA-VQ algorithm for a diversity of possible input images. The worse performance (low PSNR and some severe filtering of the high frequency details) is observed only for images with a high content of high frequencies (such as the "cat on the grass" image) although high frequency edges and details are generally well preserved.

4.2. Circuit complexity

The main advantage of our scheme based on CA-generated codebook is a dramatic reduction in the implementation complexity. The number and complexity of different operators involved in computation is estimated in the following:

The encoder: The encoding process is (bit) block oriented therefore blocks in the input image (assumed to be stored in a buffer RAM) can be processed either sequentially or parallel. To implement the encoding operation (see Section II) one requires only:

a) A RAM or ROM memory to store the reference codebook (16 kbits for 256 codewords of 64 bits each);
b) 2 counters to address the codebook RAM;
c) A unit to compute the distance between the input bit-block and the actual codeword from the codebook RAM. This can be simply a counter, which is incremented only when the bits are different;
d) Two registers: one to store the lowest distance found so far and the other to store the index of the associated codeword;
e) One simple arithmetic unit (a comparator) to compare the actual distance with the minimal one stored in a register.

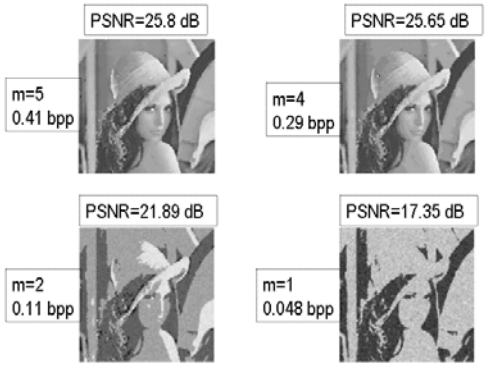

Fig.8. Visual quality of recovered images using CA-VQ with different bitrates (corresponding to *m* significant bitplanes considered in the algorithm)

Relatively simple circuitry (e.g. a properly loaded lookup table) must be considered in addition for the CA-VQ-E scheme. If the encoding is done sequentially (i.e. one block after another) the above resources are the only needed. It can be easily checked that either CAVQ schemes require less resources than any of the processors used for other standardized compression schemes. Moreover, the above scheme can fit quite well in FPGA technologies. We made a rough estimate of the FPGA implementation (using Xilinx SpartanIIE FPGA), which indicates about 17000 equivalent gate counts. Among these, only about 600 gates form the circuits listed above b) to e) while the most gates (i.e. 16384) are assigned to the codebook RAM. This compares quite favorably to the 56000 gates reported in (Grangetto, *et al*., 2002) for an circuit to compute the integer wavelet transform (i.e. about 4 times less

resources needed). Note however that this comparison is a bit against our solution since FPGA technology, although flexible is not the most compact when compared to a fully optimized ASIC solution. In such a technology our compression scheme is expected to be 10 to 20 times less complex than other approaches such as reported in (Grangetto, *et al*., 2002) or (Chen and Lai, 2004).

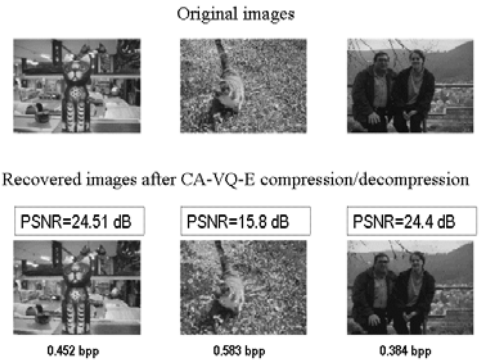

Fig.9. The response of the CA-VQ-E compression – decompression chain for several different images

The decoder: Since decoding is as simple as recovering a block from the codebook, the required resources are reduced to the RAM storing the codebook. However since the Median filter is a crucial component of our compression-decompression scheme we have to consider the circuit complexity of implementing it. Fortunately several solutions were reported so far. For instance the compact solution proposed in (Benkrid and Crookes, 2003) is reported to require only 15 configurable logic blocks (CLBs) on a Xilinx Spartan XCS30XL-4 FPGA chip, while it allows a delivery rate of 26 frames per second for PAL images (*720×576* pixels with 8-bit/pixel)..

5. CONCLUSIONS

A simple to implement, bitplane oriented, lossy compression scheme was introduced. The key ingredient differentiating our scheme for other compression systems is the use of a codebook generated by running cellular automata. Another key ingredient, leading to a dramatic simplification of the circuitry needed to implement the algorithm, is the bitplane oriented processing. The result is a compact and circuit compression system which performs better than JPEG standards for high compression rates (bitrates lower than 0.2 bpp). Given the simplicity of the circuits required to implement CA-VQ systems and its proved capacity to compress images with reasonable reconstruction quality at rates as small as 0.05 bpp (i.e. compression rates of 160 times), a possible target application of CA-VQ is in the area of ubiquitous and low power sensor-networks. Further research will focus on a variable-block size with codebooks optimized per bitplane for eliminating the saturations from the distortion curves. Another focus is on organizing the codebooks in a tree structure so that the compression time will reduce from O(*C*) to O($\log_2(C)$). An in-depth study of the hardware complexity will be further considered for a hardware implementation in FPGA technology.

6. ACKNOWLEDGMENT

The author acknowledges the support of the Alexander von Humboldt Foundation and of the Technical University of Darmstadt (Institute of the Microelectronic Systems) where most of the research work has been done. Special thanks for Dr. Ioana Dogaru for her useful comments and continuous support during the preparation of this manuscript.

7. REFERENCES

Benkrid,K. and Crookes, D. (2003), "New bit-level algorithm for general purpose median filtering", *in Journal of Electronic Imaging* -- April 2003 -- **Volume 12**, Issue 2, pp. 263-269

Chen, R.-J. and Lai, J.-L, (2004). "VLSI Implementation of the universal 2-D CAT/ICAT systems*", in Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004*, 13-15 Dec. 2004 Page(s):187 – 190.

Chua, L. and Roska, T., (2001). *Cellular neural networks and visual computing - Foundations and applications*, Cambridge University Press.

Dogaru, R. (2003). *Universality and Emergent Computation in Cellular Neural Networks*, World Scientific, Singapore.

Dogaru R. and Glesner, M. (2004). "Novel tools and methods for fast identification of emergent behaviors in CNNs with relevance to biological modeling", *in Proceedings CNNA2004 (IEEE Int'l Workshop on Cellular Neural Networks and their Applications*, Budapest 22-24 July), pp. 339-345.

Dogaru, R., Dogaru I, Glesner, M, (2005). "A smart sensor architecture based on emergent computation in an array of outer-totalistic cells", *in Proceedings of SPIE Volume: 5839 Bioengineered and Bioinspired Systems II*, Editor(s): Ricardo A. Carmona, Gustavo Liñán-Cembrano, pp. 254-263.

Dogaru, R. (2005). „A double-sieve method to identify emergent computation in cellular nonlinear networks", *in Proceedings of IEEE Eurocon 2005 – International Conference on „Computer as a Tool"*, Belgrade, 21-24 Nov. CD Proceedings.

Grangetto, M., Magli, E., Martina, M., and Olmo, G. (2002) "Optimization and Implementation of the Integer Wavelet Transform for Image Coding", *in IEEE Trans. on Image Processing*, **Vol. 11**, No. 6, February 2002, pp. 596-604.

Lafe, O. (1997). "Data Compression and Encryption Using Cellular Automata Transforms," *Engng. Applic. Artif. Intell.*, **vol. 10**, no. 6, pp. 581-591, 1997.

Lawson, S. and Szu, J. (2002). "Image compression using wavelets and JPEG 2000: A tutorial", in *Electronics and Communication Engineering Journal*, June 2002, pp.112-121.

Ostermann, J., Bormans, J., List, P., Marpe, D., M. Narroschke, M., Pereira, F., Stockhammer, T., and Wedi, T., (2004). "Video coding with H.264/AVC: Tools, Performance, and Complexity", *in IEEE Circuits and Systems Magazine*, First Quarter 2004, pp. 7-28.

Sullivan, and Wiegand, T., (1998). "Rate-distorsion optimization for video compression", in IEEE Signal Processing Magazine, November 1998, pp. 74-90.

Wolfram, S., (1983). Statistical Mechanics of Cellular Automata, *Rev. Mod. Phys*., 55(3).