Resolution Scalable Image Coding with Reversible Cellular Automata

Lorenzo Cappellari*, *Member, IEEE,* Simone Milani, *Student Member, IEEE,* Carlos Cruz-Reyes, Giancarlo Calvagno, *Member, IEEE*

***Abstract*—In a resolution scalable image coding algorithm, a multi-resolution representation of the data is often obtained using a linear filter bank. Reversible cellular automata have been recently proposed as simpler, non-linear filter banks that produce a similar representation. The original image is decomposed into four subbands, such that one of them retains most of the features of the original image at a reduced scale. In this paper, we discuss the utilization of reversible cellular automata and arithmetic coding for scalable compression of binary and grayscale images. In the binary case, the proposed algorithm that uses simple local rules compares well with the JBIG compression standard, in particular for images where the foreground is made of a simple connected region. For complex images, more efficient local rules based on the lifting principle have been designed. They provide compression performances very close to or even better than JBIG, depending on the image characteristics. In the grayscale case, and in particular for smooth images such as depth maps, the proposed algorithm outperforms both the JBIG and the JPEG2000 standards under most coding conditions.**

***Index Terms*—Scalable image coding, cellular automata, arithmetic coding.**

I. Introduction

During the last two decades, signal processing experts have dedicated significant efforts in designing and implementing new algorithms for the processing and the compression of images and video. One of the most important issues that have been faced is the possibility of delivering the coded information in a flexible and *scalable* way such that the end terminal can receive and reconstruct the coded image or video at different resolutions and quality levels.

Focusing on the compression of bi-level images, a first scalable solution for binary images, i.e. images made of black or white pixels, was proposed within the JBIG coding standard [1]. In this framework, several versions of the same input image, at different spatial resolutions, are formed and encoded. Unfortunately, this paradigm requires coding a whole image for each resolution *layer* introducing a significant information redundancy in the coded bit stream. Therefore, more efficient schemes for scalable coding of binary images have been proposed in literature [2]. However, recent works on this subject have been focusing on obtaining a high compression ratio at the expense of scalability (see [3] and [4]).

As for multi-level images, these coding strategies proved to be ineffective, and therefore most of the successive algorithms adopted a *wavelet*-based decomposition of the original signal, followed by an accurate reordering and modelization of the data to be coded. As a result of this research work, image coding experts finalized the JPEG2000 standard which permits obtaining an embedded scalable bit stream with a high coding gain and at a reasonable computational complexity [5].

Manuscript submitted May 18, 2010. L. Cappellari is the Corresponding Author.

L. Cappellari, S. Milani, and G. Calvagno (Email: {lorenzo.cappellari, simone.milani, calvagno}@dei.unipd.it) are with the Dept. of Information Engineering of the University of Padova, via Gradenigo 6/B, 35131 Padova, Italy. Phone: +39(049)827-7641. Fax: +39(049)827-7699. C. Cruz-Reyes (Email: carlos.cruz@urv.cat) is with the Research Group on Math. Linguistics, Universitat Rovira I Virgili, Tarragona, Spain.

More recently, a novel binary transform, based on *cellular automata* (CA) theory, has permitted the design of effective scalable coders for binary images that inherit many properties of the wavelet-based image coders [6]. The optimization of the transform operation, together with a carefully-designed arithmetic coder, permits obtaining a good coding gain with respect to the JBIG coder [7], [8]. Moreover, it is possible to extend the proposed approach to grayscale images and depth maps [9].

This paper presents a scalable lossless image coding algorithm based on *reversible cellular automata* (RCA). In practice, proper reversible rules are used to transform the input image into 4 subimages with a lower resolution. Each of these is then converted into a bit stream using a context-based adaptive arithmetic coder whose contexts are computed from the values of (already-coded) neighboring pixels, in the same (intra-image) or in the others (inter-image) subimages. The RCA approach is applied to binary images, as well as to grayscale depth map images generated by a structured light camera system. In the end, enhanced RCA rules based on a lifting-based scheme are designed in order to permit further improvement of the compression performance.

The rest of the paper is organized as follows. In Section II we present a short review on CA theory, while in Section III we delineate a strategy to apply RCA in scalable bi-level image coding. In Section IV we explain how arithmetic coding is eventually used for compressing binary images. In Section V and Section VI we review the utilization of the RCA approach for coding grayscale images and the utilization of enhanced local rules for higher compression, respectively. The results of our experiments and comparisons with state of the art standards for image coding are presented and discussed in Section VII. Finally, we draw our conclusions in Section VIII.

II. Cellular Automata

Computer scientists have formalized with the concept of CA a set of *global* transformations resulting from the application of the same *local* rule at infinitely many sites. More precisely, a CA consists of an infinite number of *cells* that are in correspondence with the elements of a lattice $\Lambda \sim \mathbb{Z}^d$ in the d-dimensional Euclidean space $\mathbb{R}^d$. Cells are *finite state machines* that after each *time instant* change their states synchronously according to a local rule that specifies the *new* state of a cell as a function of the *old* states of some neighboring cells.

The *state* of each cell takes values on a *finite* set Q known as the *state set* of the CA. The *configuration* of the CA, at any given time, is the state of all cells of the CA and can be described by a function $c : \mathbb{Z}^d \to Q$. Let C_Q^d denote the set of all d-dimensional configurations over Q.

The *neighborhood vector* $\mathbf{N} = [v_1, v_2, \dots, v_n]^T$ of the CA is an ordered and *finite* list of elements of Λ that specifies the relative locations of the *neighbors* of each cell. Therefore, each cell $x \in \Lambda$ has n neighbors, in positions $x + v_i$, for $i = 1, 2, \dots, n$. The *local rule* $f : Q^n \to Q$ of the CA determines the new cell state of each cell as a function of the old states of its neighbors. As a matter of fact, it is possible to characterize the global state change with the *global function* $F : C_Q^d \to C_Q^d$ defined as

$$F(c)(x) \triangleq f\left[c(x+v_1), c(x+v_2), \ldots, c(x+v_n)\right] \ . \tag{1}$$

In case F is bijective, the CA is a *reversible* CA (RCA) and there exist a CA, known as the *inverse automaton*, whose global function corresponds to F^{-1} (see [10] for more details).

In case every cell stores an m-tuple $\mathbf{q} = [q_1, q_2, \ldots, q_m]^T \in Q^m$ of different state variables (where q_i is called the i-th component of the cell), it is possible to define a *multi-band CA*. For any configuration $c \in C_{Q^m}^d$ we denote the m *subbands* of c by $c_1, c_2, \ldots, c_m$, where $c_i \in C_Q^d$ would be the configuration of the CA if the state of each cell was given only by its i-th component. The local rule $f : (Q^m)^n \to Q^m$ of an m-band CA may be seen as composed of m different local rules $f_i : (Q^m)^n \to Q$ for each subband i. Note that each f_i changes only the i-th component of the cell in position x.

In the CA literature several simple techniques have appeared to construct local rules that make a CA reversible, such as, for example, partitioned CA, Margolous neighborhood, second-order CA, and CA with conserved-landscape permutations [11]. In the following we are going to present two basic techniques to construct reversible multi-band CA.

A *trivial* m-band CA is a CA in which the neighborhood consists of exactly one cell, i.e. $\mathbf{N} = [v]$, and the local rule is a permutation $\pi_m : Q^m \to Q^m$ of the state set. The inverse CA is obviously the trivial CA with neighborhood $\mathbf{N} = [-v]$ and local rule π_m^{-1}.

An *elementary* m-band CA is a CA in which the neighborhood $\mathbf{N} = [0, v_2, v_3, \ldots, v_n]^T$ of a cell includes the cell itself in the first position of the neighborhood vector (without loss of generality), and the local rule changes only one subband according to a certain permutation $\pi : Q \to Q$ of Q depending on the states of the other $m-1$ subbands for the neighboring cells.

In particular, if Π_Q denotes the set of all permutations of Q, $h : \left(Q^{m-1}\right)^n \to \Pi_Q$ denotes the function that determines the actual permutation, and k denotes the index of the subband that is modified by the CA. The local rule of an elementary m-band CA is such that f_i is the identity function for each $i \neq k$, while

$$f_k(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = h\left(\mathbf{q}_{1\sim k}, \mathbf{q}_{2\sim k}, \ldots, \mathbf{q}_{n\sim k}\right)\left(\mathbf{q}_{1_k}\right) \ , \tag{2}$$

where $\mathbf{q}_{\sim k}$ denotes the subvector of $\mathbf{q}$ obtained by removing its k-th component. The inverse CA is obviously the elementary m-band CA with the same neighborhood and with local rule such that f_i is the identity function for each $i \neq k$.

It is clear that by using a composition of elementary m-band CA each one of which changes a different subband it is possible to design more complicated reversible m-band CA [12].

III. A Cellular Automata Rule for Scalable Image Coding

Since a RCA defines a reversible transformation of the state of each cell, it is possible to associate this operation with the transforms employed in the current image coding algorithms: the given original image uniquely defines the initial configuration P of the RCA (*pixel domain*), and then the RCA evolves into a different configuration R, i.e. into the *RCA transform* (RCAT) *domain*.

Recent works have shown that this strategy proves to be extremely effective for the scalable compression of images provided that in the RCAT domain the image is effectively decorrelated and decomposed into a form that permits reconstruction with progressively-increasing spatial resolution. In order to satisfy both requirements, blocks of 2×2 pixels are mapped to cells of a multi-band RCA (arranged over $\Lambda = 2\mathbb{Z}^2$). Then, a local rule is designed such that in the RCAT domain one subband retains most of the visual features of the original image at a reduced scale.

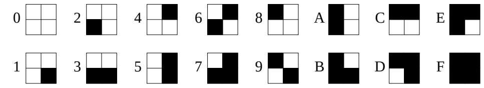


Fig. 1. Association of the elements of the state set Q^m to hexadecimal digits.

Unfortunately, unlike linear systems that are mostly analyzed and designed in the frequency domain, the general RCA transform does not offer any other domain into which the effects of the local rules can be examined. Hence, trivial local rules suitable for scalable image coding have been derived from scratch by mimicking the time domain behavior of traditional linear coding systems.

Binary images are images where each pixel can take at most two values (0 for white pixels and 1 for black ones in our convention). As a matter of fact, it is possible to define the initial state of each cell by reading in raster scan order the binary values of the 2×2 associated pixels. Each pixel value is associated to a subband (i.e. $Q = \{0, 1\}$), and the $m = 4$ resulting subbands are denoted by the tuples 00, 01, 10, and 11. The state of a cell can be equivalently indicated by an hexadecimal digit as shown in Fig. 1.

In the signal processing literature, linear filter-banks for image compression are designed for one-dimensional signals and then extended to the two-dimensional case in a separable fashion (see the case of *discrete wavelet transforms* in JPEG2000 [13], [5]). Due to the frequency responses of the filters, one subband provides a good reduced scale representation of the original signal, and the remaining ones are *sparse*, meaning that most samples are close to zero. More precisely, the non-zero samples in these subbands arise mainly in correspondence of vertical, horizontal, and diagonal edges in the original image, respectively.

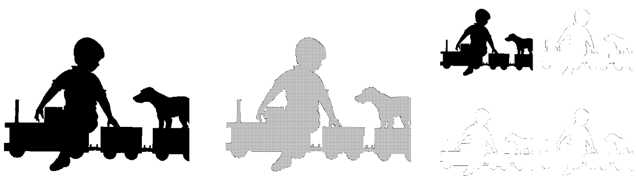
The trivial rule proposed in [6] for the CA described above, in which $\mathbf{N} = [0]$ and π_4 is defined as

$$\begin{array}{llll} \pi_4(0) = 0 & \pi_4(4) = 5 & \pi_4(8) = 6 & \pi_4(C) = A \\ \pi_4(1) = 1 & \pi_4(5) = 4 & \pi_4(9) = F & \pi_4(D) = B \\ \pi_4(2) = 3 & \pi_4(6) = 7 & \pi_4(A) = C & \pi_4(E) = 9 \\ \pi_4(3) = 2 & \pi_4(7) = E & \pi_4(B) = D & \pi_4(F) = 8 \end{array} \ , \tag{3}$$

is such that after one time instant the resulting configuration R in the RCAT domain has the required characteristics. As an example, in uniform areas, $\pi_4(0) = 0$ and $\pi_4(F) = 8$ enforce the similarity of the subband R_{00} to the original image and the sparseness of the subbands R_{01}, R_{10}, and R_{11}. Similar considerations can be made for vertical and horizontal edges (see [6] for more details). Of course, the constraint for π_4 to be a permutation does not really give the possibility to exactly enforce the desired behavior for all possible states. However, it is sufficient that this happens at least for the states which statistically occur more often.

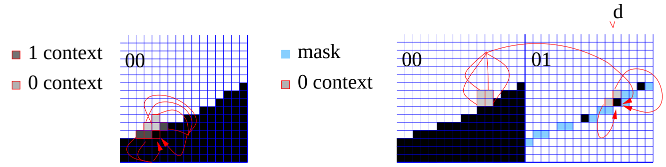
An example of the effect of this trivial CA is shown in Fig. 2. The original binary image shown in Fig. 2(a) has a first order entropy of 0.817 bit per pixel (bpp). In the RCAT domain (see Fig. 2(b)), the *global* first order entropy reduces to 0.358 bpp since three subbands are very sparse. In fact, the first order entropies for subbands R_{00}, R_{01}, R_{10}, and R_{11} (showed in Fig. 2(c)) are respectively 0.817 bpp, 0.057 bpp, 0.061 bpp, and 0.043 bpp. In addition, the relative position of the non-zero entries in subbands R_{01}, R_{10}, and R_{11} gives approximately the relative position of vertical, horizontal, and diagonal edges in the original image, respectively.

In the following, we assume that the RCA decomposition is applied L times: each decomposition is applied to subband $R_{00}^{(i-1)}$ ($R_{00}^{(0)} = P$



(a) pixel domain (b) RCAT domain (c) subbands

Fig. 2. Binary image representation in various domains.



(a) LFC (b) HFC (subband $R_{01}^{(j)}$)

Fig. 3. Coding contexts for the LFC and the HFC algorithm.

is the original pixel domain configuration) and produces subbands $R_{00}^{(i)}$, $R_{01}^{(i)}$, $R_{10}^{(i)}$, and $R_{11}^{(i)}$, for $i = 1, 2, \ldots, L$. The bit stream representing the subband $R_{00}^{(L)}$ is the *base layer* (or the 0-th layer), and gives the lowest available resolution. The bit stream representing subbands $R_{01}^{(L+1-i)}$, $R_{10}^{(L+1-i)}$, and $R_{11}^{(L+1-i)}$, for $i = 1, 2, \ldots, L$, is the i-th *(enhancement) layer*; together with all previous layers, it permits reconstructing the subband $R_{00}^{(L-i)}$, i.e. the i-th resolution.

IV. Compression of Binary Images

Traditional entropy coding strategies for binary images rely on predicting the pixel to be coded according to some *causal* (i.e. already-coded) neighboring pixels. Depending on the value of these neighbors, the coding schemes compute a coding *context* for the current pixel, which is associated to a binary *probability mass function* (pmf).

As an example, in *sequential* coding mode the JBIG standard predicts the current pixel from the previous causal pixels with Manhattan distance less than or equal to 2. In case the image is coded in *progressive* mode (i.e. several resolutions of the image are coded together in order to permit a progressively increasing resolution when displaying the image), the contexts are characterized both by the causal neighboring pixels and by the co-located pixels in the lower resolution image (see [1]). In this way, the lower resolution images are used to characterize the context computation for the higher resolution images, limiting the bit rate increment due to the inclusion of differently-sized versions of the original image. The value of the pixel to be coded, together with the related context, is sent to a binary arithmetic coder that maps it into an interval whose length and extreme points depend on the pmf associated to the context and on the previously coded data, respectively. In the following step of the arithmetic coding process, whenever an interval needs to be *renormalized* it is associated with an emitted bit stream (see [14]). A sequential decoder can decode only the image at the lowest resolution, while a progressive decoder can also decode all enhancement layers.

This kind of approach comes out to be ineffective for the RCA-transformed image. The subband images result highly non-stationary and quite sparse, and therefore no spatial predictor proves to be accurate enough in predicting the pixel value to be coded. As a consequence, we resorted to more powerful coding techniques in order to obtain a coding efficiency comparable to that of the other standards. The entropy coding of the transformed image consists of two separate algorithms: an entropy coding strategy for the base layer (*low-frequency coding*, LFC) and another coding strategy for the enhancement layers (*high-frequency coding*, HFC).

A. Coding Strategy for the Lowest Resolution Image: the LFC Algorithm

Like other coding solutions for binary images [1], the subband $R_{00}^{(L)}$ is scanned in row-wise order checking whether the current row differs from the previous one or not. A binary flag f_{dr} is coded in the bit stream that signals this information. In this way, it is possible to exploit the vertical correlation of binary images and reduce the amount of coded information. In case some differences are found, the whole row is scanned and each pixel is coded with a binary arithmetic coder. The contexts into which the pixels are coded are obtained from the causal pixels with Manhattan distance lower than 3 (see Fig. 3(a)); hence, there are $2^6 = 64$ possible different contexts.

B. Coding Strategy for the Detail Subbands: the HFC Algorithm

For each of the subbands $R_b^{(j)}$ of the i-th enhancement layer ($b = 01, 10, 11$ and $j = L+1-i$), the coding algorithm identifies a positional mask $\mathcal{M}_b$ where non-zero pixel values are likely to be found. At first, a segmentation routine computes the gradients $\nabla_y(x,y)$, $\nabla_x(x,y)$, $\nabla_{pd}(x,y)$, and $\nabla_{sd}(x,y)$, which denote the first order derivative along the vertical, the horizontal, the principal diagonal (oriented at 45 with respect to the horizontal) and the secondary diagonal (oriented at 135 with respect to the horizontal) directions computed on the $R_{00}^{(j)}$ subband. Then, masks $\mathcal{M}_b$ are created as follows

$$\begin{aligned} \mathcal{M}_{01} &= \{(x,y)|\nabla_x(x,y) > 0 \vee \nabla_x(x-1,y) < 0\} \\ \mathcal{M}_{10} &= \{(x,y)|\nabla_y(x,y) > 0 \vee \nabla_x(x,y+1) < 0\} \\ \mathcal{M}_{11} &= \{(x,y)|\nabla_{pd}(x,y) > 0 \vee \nabla_{pd}(x-1,y+1) < 0\} \\ &\cup \{(x,y)|\nabla_{sd}(x,y) > 0 \vee \nabla_{sd}(x-1,y-1) < 0\}, \end{aligned} \quad (4)$$

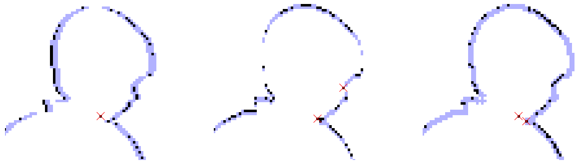
where x and y are the horizontal and vertical coordinates in the subband $R_{00}^{(j)}$. However, the masks in (4) do not permit an accurate localization of non-zero pixels in the high frequency subbands since the adopted RCA transform introduces a one-pixel displacement depending on the position of the vertical or horizontal border. As an example, both blocks 5 and A depict a vertical border, but only the RCA transformed block C corresponding to block A reports a non-zero pixel value in the subband $R_{00}^{(j)}$ (see Fig. 1). As a consequence, it is convenient to enlarge the positional masks $\mathcal{M}_b$ for each subband b into the sets $\mathcal{M}_b^e = \mathcal{M}_b \cup \mathcal{M}_b^d$ according to the following operations.

For each pixel in $\mathcal{M}_b$ the algorithm computes the vertical and the horizontal Sobel operators ($S_h(x,y)$ and $S_v(x,y)$) and defines the sets

$$\begin{aligned} \mathcal{M}_{01}^{s+} &= \{(x,y)|(x,y) \in \mathcal{M}_{01} \wedge S_h(x,y) > 0\} \\ \mathcal{M}_{01}^{s-} &= \{(x,y)|(x,y) \in \mathcal{M}_{01} \wedge S_h(x,y) < 0\} \\ \mathcal{M}_{10}^{s+} &= \{(x,y)|(x,y) \in \mathcal{M}_{10} \wedge S_v(x,y) > 0\} \\ \mathcal{M}_{10}^{s-} &= \{(x,y)|(x,y) \in \mathcal{M}_{10} \wedge S_v(x,y) < 0\} \\ \mathcal{M}_{11}^{s+} &= \{(x,y)|(x,y) \in \mathcal{M}_{11} \wedge S_h(x,y) > 1\} \\ \mathcal{M}_{11}^{s-} &= \{(x,y)|(x,y) \in \mathcal{M}_{11} \wedge S_h(x,y) < -1\} \end{aligned} \quad (5)$$

which permit extending the pixel maps as

$$\begin{aligned} \mathcal{M}_{01}^{d} &= \left\{\mathcal{M}_{01}^{s+} \oplus \{(1,0)\}\right\} \cup \left\{\mathcal{M}_{01}^{s-} \oplus \{(-1,0)\}\right\} \\ \mathcal{M}_{10}^{d} &= \left\{\mathcal{M}_{10}^{s+} \oplus \{(0,1)\}\right\} \cup \left\{\mathcal{M}_{10}^{s-} \oplus \{(0,-1)\}\right\} \\ \mathcal{M}_{11}^{d} &= \left\{\mathcal{M}_{11}^{s+} \oplus \{(0,1),(1,0)\}\right\} \cup \\ &\quad \left\{\mathcal{M}_{10}^{s-} \oplus \{(0,-1),(-1,0)\}\right\} \end{aligned} \quad (6)$$



(a) subband $R^{(1)}_{01}$ (b) subband $R^{(1)}_{10}$ (c) subband $R^{(1)}_{11}$

Fig. 4. Subbands of the last enhancement layer (black pixels) and the support points ($\mathcal{M}^e_b$) identified by the segmentation routine (blue points); points of the signals that do not lie within $\mathcal{M}^e_b$ are marked with red $\times$s.

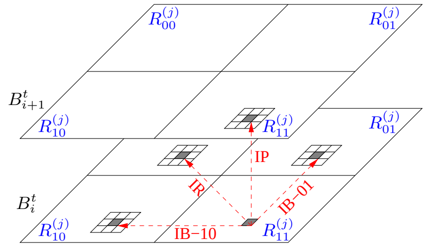


Fig. 5. Inter *context windows* used for context formation.

where $\oplus$ denote the Minkowski sum (morphological dilation). Fig. 4 shows the subbands $R^{(1)}_{01}$, $R^{(1)}_{10}$, $R^{(1)}_{11}$ relative to a detail of the image *child* underlining the pixels to be coded and the pixel mask $\mathcal{M}^e_b$. It is possible to see that the adopted extending rule proves to be effective since the number of black points that do not lie in $\mathcal{M}^e_b$ is quite small. Therefore, their positions are coded separately specifying their Cartesian coordinates.

The arithmetic encoder processes the pixels of each subband in a row-wise order for the $R^{(j)}_{01}$ and $R^{(j)}_{11}$ subbands and in a column-wise order for the $R^{(j)}_{10}$ subband. For each row (column), the encoder checks whether it contains positions (x, y) such that $R^{(j)}_b(x, y) \neq 0$ and $(x, y) \in \mathcal{M}^e_b(x, y)$. In case there are non-zero pixel values, a flag value f_{nz} is coded in the bit stream using a binary arithmetic coder with a separate context for each subband. The structure of the binary adaptive arithmetic coder is the same of the H.264/AVC standard [15], including its renormalization strategy for the coding intervals, the structure of binary contexts, and the context update routine based on a 64-states finite state machine. Then, all the pixels of the non-zero row (columns) within the positional masks $\mathcal{M}^e_b$ are coded by the same arithmetic coder using the pixels $R^{(j)}_{00}(x, y)$, $R^{(j)}_{00}(x-1, y)$, $R^{(j)}_{00}(x, y-1)$, $R^{(j)}_{00}(x-1, y-1)$, $R^{(j)}_b(x-1, y)$, $R^{(j)}_b(x, y-1)$, and the number d of pixels interlying between the current pixels and the last pixel outside $\mathcal{M}^e_b$ in the current row (column). Fig. 3(b) graphically displays the coding contexts for the subband $R^{(j)}_{01}$ as an example. The value of d is truncated whenever it is greater than 3.

Note that the proposed arithmetic coding algorithm requires only $256+1$ binary contexts and avoids using complex prediction schemes for the estimation of the pixels to be coded.

The advantage of this scheme relies on the fact that positional masks identify where the most significant elements of an image are located. A possible extension of this approach has been obtained by adaptively enlarging of the original positional masks $\mathcal{M}_b$. More precisely, the encoder partitions the input subband R^j_b into blocks of 32×32 pixels and computes how much (in terms of pixels) the corresponding mask $\mathcal{M}_b$ needs to be enlarged. The proposed adaptive approach provides better results on a wider range of images, as it will be shown in Section VII.

V. Compression of Grayscale Images

When dealing with grayscale images, we must be able to cope with a substantially more complex correlation structure in pixel domain. These images can be seen as a collection of their bit-planes from B_0 (less significant) to B_{N-1} (most significant), which indeed represent N binary images. The proposed coding algorithms, that first appeared in [9], reuse the trivial CA presented above and employ the following strategies for their compression:

Algorithm 1) each B_i is independently transformed (into B^t_i) using the local rule proposed above; the residual *inter-plane correlation* is then exploited in the RCAT domain;

Algorithm 2) an *inter-plane prediction* operation is used for predicting bit-plane B_i ($i < N-1$) from bit-plane B_{i+1}, such that only the prediction error E_i must be actually coded (the upper bit-plane is encoded first); in particular, $E_i = B_i \ominus B_{i+1}$, where $\ominus$ denotes difference in the binary domain. Then, each E_i is independently transformed (into E^t_i) with the local rule, and entropy coded.

In Algorithm 1, a substantial amount of correlation remains between B^t_i and B^t_{i+1}, in particular for smooth images such as *depth maps*. In practice, most of the features in B^t_{i+1} occur at the same position of the features of B^t_i, indicating that most edges of B_{i+1} are co-positioned with edges in B_i. This justifies the strategy of Algorithm 2, i.e. the choice to deal with the bit-planes of the original image in the Gray code domain. Eventually, by comparing the experimental average values of the conditioned entropies $H(B^t_i|B^t_{i+1})$ and $H(E^t_i|E^t_{i+1})$ (i.e. the expected coding rate) for the various bit-planes, it was found that the expected compression performance of the two algorithms should be about the same [9].

Regarding the reduced resolution, in Algorithm 1 the $R^{(L-j)}_{00}$ subband of B^t_i (denoted by $R^{(L-j)}_{00,i}$) is taken as the i-th bit-plane of the j-th resolution (L denotes the number of iterations of the CA processing). In Algorithm 2, instead, the i-th bit-plane of the j-th resolution is obtained from the $(i+1)$-th bit-plane (of the same resolution) adding the $R^{(L-j)}_{00}$ subband of E^t_i (again, denoted by $R^{(L-j)}_{00,i}$).

In the RCAT domain, context-adaptive arithmetic coding is employed to code the samples in the various subbands of B^t_i (or E^t_i). The subbands are always coded in a *resolution-progressive* order: data belonging to subband $R^{(L)}_{00,i}$ are encoded first for $i = N-1, N-2, \ldots, 0$; then, data needed for reconstruction of subband $R^{(L-j)}_{00,i}$ are encoded (with the same bit-plane order), for $j = 1, 2, \ldots L$. Inside each resolution, subbands are scanned in this order: $R^{(j)}_{01,i}$, $R^{(j)}_{10,i}$ and $R^{(j)}_{11,i}$, similarly to the processing order used in the standard JPEG2000 [5]. Inside subbands $R^{(j)}_{01,i}$ (and $R^{(L)}_{00,i}$), samples are coded in raster column order (for following vertical features); inside subbands $R^{(j)}_{10,i}$, samples are coded in raster row order (for following horizontal features); inside subbands $R^{(j)}_{11,i}$, samples are instead coded in (down-left) diagonal order (for following diagonal features).

Differently from the binary domain, it is not immediately clear which data provide the best prediction for coding. The following *context windows* are hence used for context formation in place of edge predictors (the position of these context windows w.r.t. the sample to be coded is shown in Fig. 5): (i) a 3×3 context window in the corresponding subband of B^t_{i+1} (or E^t_{i+1}), to capture the *inter-plane* (IP) correlation ($i < N-1$), (ii) a 3×3 context window in the $R^{(j)}_{00,i}$ subband, to capture the *inter-resolution* (IR) correlation of subbands $R^{(j)}_{01,i}$, $R^{(j)}_{10,i}$, and $R^{(j)}_{11,i}$, and (iii) a 3×3 context window in the $R^{(j)}_{01,i}$ subband and another 3×3 context window in the $R^{(j)}_{10,i}$ subband (if it has already been coded), to capture the residual *inter-band* (IB-01 or IB-10) correlation that the subbands $R^{(j)}_{10,i}$ and $R^{(j)}_{11,i}$ may have with subbands relative to the same bit-plane and resolution. In addition,

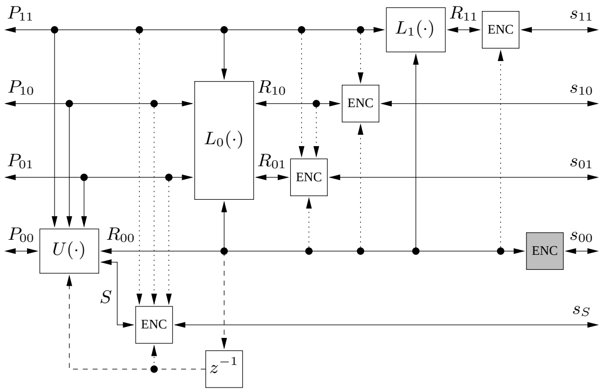


Fig. 6. Scheme of the proposed encoder and decoder based on elementary CA: signals flow to the right side and to the left side, respectively.

an *intra* (I) context window of 8 pixels is used to possibly capture intra correlation[1].

Since up to 4 inter-context windows plus 1 intra-context window are used, the maximum number of contexts is undoubtedly huge ($2^{4\cdot 9+8} = 2^{44}$) and would lead to poor probability estimates for the arithmetic coder. However, using the optimization algorithm described in [9], they are eventually grouped together into a smaller number of contexts. In practice, in each subband, the context window that is expected to contain data with the highest mutual information with the data to be coded (according to an experimental analysis on a suitable set of images) is taken into consideration first and provides a *base context label*. Then, a *refined context label* is formed taking into account both this context label and the context window (out of the unused ones) that is expected to contain data with the highest mutual information with the data to be coded (conditioned on the knowledge of the previous context label), and so forth. The last refined context label is finally used by the arithmetic coder[2].

The proposed algorithms were first applied only to depth maps. In Section VII we will also test their performance over a set of well-known natural images. In this case, 4 images of (*lena*, *mandrill*, *peppers*, and *monarch*) are used as training set for context optimization. 4, 8, 12, and 12 contexts are used for coding the $R^{(L)}_{00,N-1}$, $R^{(j)}_{01,N-1}$, $R^{(j)}_{10,N-1}$ and $R^{(j)}_{11,N-1}$ subbands, respectively; 8, 8, 10, and 10 contexts are used for coding the $R^{(L)}_{00,i}$, $R^{(j)}_{01,i}$, $R^{(j)}_{10,i}$ and $R^{(j)}_{11,i}$ subbands ($i < N-1$), respectively. Details about the number of contexts in the depth map case can be found in [9].

VI. Enhanced Cellular Automata Rules for Detailed Images

The trivial CA discussed above is very suitable for binary images in which the foreground is composed by a few connected regions. When dealing with more complex images, it is likely that (i) the permutation in (3) is no longer good for data compaction, (ii) subband R_{00} is no longer a good representation of the original image at a reduced scale, and (iii) edge prediction is no longer efficient for entropy coding.

Elementary operations in the multi-band CA domain are duals to the *lifting* operations introduced in [16] for designing better linear filter-banks for data compression, and may hence be tuned to achieve similar goals. This fact was observed in [8], where the coding scheme of Fig. 6 was first proposed.

[1]Intra-contexts are designed in order to be *causal*, to contain at least all samples with Manhattan distance less than 3 w.r.t. the sample to be coded, and to limit the number of memory accesses upon context window updating operations.

[2]Run-length coding is actually employed for data in contexts where only a very few *ones* are expected.

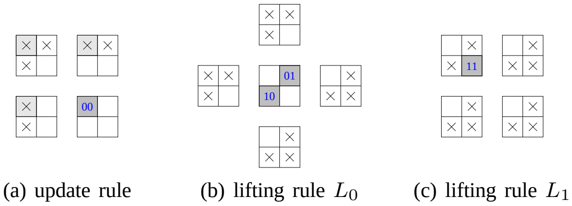


Fig. 7. Contexts used by the lifting rules: the samples corresponding to the white squares determine which permutation is used on the sample corresponding to the shaded square(s). That permutation does not depend from the samples denoted with $\times$.

With the first lifting step (*update*), it is actually possible to enforce a good quality in the reduced image. In particular, we observed that the filtering procedure used by the JBIG standard, when layered coding is employed, can be implemented as a *modified* lifting step. Accordingly, the new value in the R_{00} subband depends from the old value of the sample in the P_{00} subband and from the values taken by its 8-neighbors, which actually belong to different subbands of the employed CA structure (see Fig. 7(a))[3]. However, the JBIG rules are such that in some cases the new sample is actually *independent* from the old one (e.g. a zero could be enforced no matter what the old state value was). In these cases, after the most suitable permutation is applied, the wrongness of the result is indicated in a side information signal S. This conditioned reversible permutation $U(\cdot)$ was designed in order to statistically minimize the occurrences of these *wrong* samples.

Once the R_{00} subband is obtained, a very good prediction can be formed for the samples in the subbands P_{01} and P_{10}, based on their 4-neighbors which again do not belong to these subbands. A joint lifting rule $L_0(\cdot)$ was designed that statistically minimizes the joint entropy $H(R_{01}, R_{10})$. In particular, given the context shown in Fig. 7(b), such rule permutes the most probable tuple taken by states corresponding to subbands 01 and 10 of each cell into the $(0, 0)$ output and the less probable ones into the $(0, 1)$, $(1, 0)$, and $(1, 1)$ outputs respectively.

Finally, the samples in the P_{11} subband are lifted, based on samples in the R_{00} subband only (see Fig. 7(c)). The lifting rule $L_1(\cdot)$ minimizes $H(R_{11})$. Since $I(R_{01}, R_{10}; P_{11})$ is very low, it would make no sense to consider other subbands in this lifting operation. Typically, the subbands obtained with the proposed approach are sparser w.r.t. the subbands of the trivial CA approach, and require less bits even if a fourth subband S must be encoded too.

The update and lifting rules have been optimized based on a training set of 12 images of various sizes taken from Set 3 (see the next section). Data in each subband is arithmetically encoded; the dotted lines in Fig. 6 connect the "ENC" blocks that perform arithmetic coding with the subbands used for context formation, in which context windows similar to the ones of Fig. 5 are used. The outcomes in these windows are efficiently grouped with the algorithm proposed in [9] into 4, 6, 11, and 4 contexts for coding the R_{11}, R_{10}, R_{01}, and S subband, respectively, for compressing the simple images in Set 1 (see the next section), and into 5, 12, 44, and 5 contexts for coding the R_{11}, R_{10}, R_{01}, and S subband, respectively, for compressing the remaining sets of images. The R_{00} subband is

[3]Actually, also the *already updated* samples corresponding to the light shaded squares in Fig. 7(a) are used for JBIG-like subsampling. This is indicated by the z^{-1} box in Fig. 6. Hence, update is not a *true* elementary rule.

coded with the baseline JBIG algorithm [17], or by iterating the proposed algorithm (if more than two resolutions are desired).

VII. Experimental Results

In order to evaluate the effectiveness of the CA-based approaches for image compression, we coded different types of images with different characteristics. At first, we evaluated the performance on simple silhouette-like binary images. Then, the proposed approach was applied to grayscale images. Finally, we tested how the introduced lifting scheme permits improving the coding gain for binary images, in particular for the more detailed ones. In all cases, we present comparisons with other state-of-the-art coders.

A. Compression Performance on Simple Binary Images

In this section, we present compression results obtained coding different silhouette-like binary images using both the approach proposed in Section IV and the JBIG standard [1]. Each picture is made of 512×512 pixels and is coded with multiple resolutions enabling a resolution-scalable decoding of the coded bit stream. The employed JBIG coder [17] is used with all other prediction options offered by the JBIG standard (*differential prediction* – DPON, and *typical prediction* – TPBON and TPDON [1]) switched off for a more fair comparison. In the following, we will refer to the approach that enlarges the positional masks via equation (6) as *RCA-based binary image coder* (RCABIC), while the slightly different approach that adaptively widens these masks will be referred as *adaptive RCABIC* (ARCABIC).

The number of bits per pixel (bpp) used by the RCABIC and the JBIG coder for each compressed picture are reported in Table I. It is possible to notice that the proposed approach proves to be more efficient since it is able to re-use more efficiently the information of a lower resolution image while encoding only the details of a higher resolution image. As a consequence, the proposed approach proves to be much better than JBIG whenever multiple decomposition levels are enabled ($L > 2$).

The RCABIC coder obtains high compression ratios for images where black pixels lies within a connected region (such as shape-like images or binary *alpha channels*) while the performance significantly decreases for detailed images (such as in fax images of scanned text). As an example, the algorithm proves to be ineffective since the concentration capability of the RCA transform is reduced. Experimental results show that for the image *airplane* the coding gain is about 31% with 5 decomposition levels while for the image *grandma* the bit stream increase of 9%. Moreover, it is possible to notice that after a certain number of levels the average compression gain diminishes as the number of decomposition levels augments. This behavior depends on the mask dilation process, which proves to be ineffective for small resolution subimages since the set of pixels to be coded expands with respect to image dimensions.

A possible compromise is offered by the ARCABIC strategy which widens the masks adaptively according to the characteristics of the coded signal. This approach performs better on complex binary images despite additional computational complexity is required to estimate the extension of dilation. Experimental results in Table I show that the ARCABIC strategy has a reduced efficiency on shape-like images, such as *airplane*, but proves to be extremely effective on more complex images (see *grandma*).

B. Compression Performance on Grayscale Images

The algorithms presented in Section V were implemented in C and employ an independently developed variant of the arithmetic coder used in H.264/AVC. The performance was averaged first over a small dataset of 9 512×512 natural images with bit-depth $N = 8$.[4] In order to analyze the performance at various bit-depths, in the various tests one or more of the less-significant bit-planes was discarded prior to coding. The results (see left-hand side of Table II) compare our coders with three well-established standards for lossless image compression, namely JBIG [1] (using the JBIG-KIT software [17]), JPEG2000 [5] (using the JasPer implementation [18] in lossless mode), and JPEG-LS [19] (using the Hewlett-Packard implementation [20]). In order to operate the various coders under the same test conditions, the same number of decomposition levels ($L = 3$) was used for each standard (with the exception of JPEG-LS which is not scalable). Also, in the JBIG coder, only the *typical prediction* of the differential layers (TPDON) was enabled, and a single *stripe* was coded for each image; the other coders were operated with their standard configuration. In all generated bit streams, bytes belonging to headers were not taken into account; Algorithm 2 was found to outperform Algorithm 1 and is hence kept as reference.

TABLE I
Compression performance (bpp) of the RCABIC, the ARCABIC, and the JBIG coder on simple binary images. The average savings (%) are given with respect to the compression performance of the JBIG coder.

Image	 *airplane*			 *grandma*			 *child*				
Levels	**RCABIC**	**ARCABIC**	**JBIG (reference)**	**RCABIC**	**ARCABIC**	**JBIG (reference)**	**RCABIC**	**ARCABIC**	**JBIG (reference)**	**Saving (%) (RCABIC)**	**Saving (%) (ARCABIC)**
2	.0169	.0178	.0150	.0415	.0421	.0356	.0241	.0261	.0211	−14.5	−20.2
3	.0145	.0155	.0175	.0434	.0423	.0399	.0222	.0245	.0247	6.1	2.1
4	.0139	.0151	.0193	.0467	.0436	.0434	.0233	.0255	.0276	12.0	10.0
5	.0142	.0152	.0206	.0496	.0445	.0454	.0248	.0261	.0295	12.6	13.2
6	.0155	.0155	.0214	.0522	.0450	.0462	.0266	.0265	.0305	9.1	14.4
7	.0162	.0157	.0218	.0538	.0452	.0465	.0278	.0267	.0308	6.6	14.7
8	.0167	.0159	.0220	.0544	.0454	.0466	.0287	.0271	.0309	4.8	14.2

When coding only 2, 3, or 4 most significant bit-planes, the proposed coder outperforms both the JBIG and the JPEG2000 standard. In particular, a better performance is obtained not only at the full resolution (ALL layers), but also at the reduced ones, as we measured a compression loss in the separate layers too (in the 0-th one too – not shown), with the exception of the third one, where JBIG outperforms our coder. Of course, this mode of operation somewhat precludes JPEG2000 from achieving high performances since the wavelet filters are not applied to the full 8-bit data. However, this is still the best that JPEG2000 can do in all applications where only low bit-depth data is available. At full bit-depth, in fact, JPEG2000 outperforms our coder, even if the latter is still more performing than JBIG; similarly, the JPEG-LS coder, which exploits the correlation via an adaptive linear prediction over a small *intra* window, outperforms the proposed coder only if at least 4 bit-planes are coded, but of course it does not provide scalability.

As we expect the proposed method to be more suitable for smooth images, performances were averaged also over a database of 40 640×480 *range* images of various objects (with bit-depth $N = 8$)[5],

[4]The database contains the images *barbara*, *boat*, *goldhill*, *lena*, *mandrill*, *peppers*, *washsat*, *zelda*, and *monarch*. These images can be downloaded, for example, from http://links.uwaterloo.ca/Repository.html.

[5]This database is publicly available [21]; the images are the direct output of a *structured light camera* system and so have been denoised with a 5×5 median filter prior to compression.

TABLE II
AVERAGE COMPRESSION PERFORMANCE (BPP) FOR THE TWO PROPOSED, THE JBIG, THE JPEG2000, AND THE JPEG-LS CODERS ON TWO DIFFERENT SETS OF GRAYSCALE IMAGES. THE AVERAGE LOSSES (%) ARE GIVEN WITH RESPECT TO THE COMPRESSION PERFORMANCE OF ALGORITHM 2.

		Natural images									**Depth maps**								
Bit-planes	**Layer(s)**	**Algorithm 2 (reference)**	**Algorithm 1**	**JBIG**	**JPEG2000**	**JPEG-LS**	**Loss (%) (Algorithm 1)**	**Loss (%) (JBIG)**	**Loss (%) (JPEG2000)**	**Loss (%) (JPEG-LS)**	**Algorithm 2 (reference)**	**Algorithm 1**	**JBIG**	**JPEG2000**	**JPEG-LS**	**Loss (%) (Algorithm 1)**	**Loss (%) (JBIG)**	**Loss (%) (JPEG2000)**	**Loss (%) (JPEG-LS)**
2	**1**	0.476	0.542	0.726	0.782	n/a	13.0	50.9	64.6	n/a	0.076	0.078	0.118	0.173	n/a	3.3	56.9	129.7	n/a
	2	0.401	0.441	0.502	0.604	n/a	9.9	23.3	52.7	n/a	0.040	0.041	0.055	0.098	n/a	2.5	38.2	143.6	n/a
	3	0.337	0.360	0.312	0.454	n/a	7.4	−8.4	38.0	n/a	0.024	0.025	0.025	0.056	n/a	4.2	5.6	134.9	n/a
	ALL	0.479	0.519	0.501	0.671	0.626	8.7	3.7	42.9	35.2	0.042	0.044	0.051	0.096	0.092	3.4	20.1	127.5	119.4
3	**1**	0.828	0.959	1.246	1.146	n/a	15.2	49.8	41.5	n/a	0.131	0.146	0.202	0.306	n/a	11.3	54.7	133.6	n/a
	2	0.701	0.791	0.877	0.900	n/a	12.9	23.7	31.6	n/a	0.072	0.080	0.097	0.174	n/a	10.4	35.1	141.1	n/a
	3	0.597	0.661	0.561	0.679	n/a	11.3	−7.4	17.7	n/a	0.044	0.049	0.045	0.100	n/a	11.7	3.4	126.8	n/a
	ALL	0.844	0.945	0.888	1.003	0.908	12.3	4.0	22.6	11.4	0.076	0.084	0.089	0.172	0.129	10.9	17.7	126.8	69.9
4	**1**	1.335	1.543	1.998	1.603	n/a	15.9	49.7	23.3	n/a	0.222	0.263	0.335	0.489	n/a	18.7	52.0	120.8	n/a
	2	1.157	1.311	1.469	1.310	n/a	13.7	25.9	15.4	n/a	0.127	0.149	0.166	0.287	n/a	17.8	31.1	126.0	n/a
	3	1.013	1.130	0.988	0.997	n/a	12.3	−4.0	1.0	n/a	0.080	0.094	0.079	0.169	n/a	18.1	−1.1	112.5	n/a
	ALL	1.417	1.594	1.524	1.465	1.322	13.2	6.4	5.9	**−4.7**	0.134	0.157	0.152	0.290	0.194	17.5	13.6	116.0	44.7
8	**1**	4.241	4.542	5.831	4.142	n/a	7.2	37.9	−2.1	n/a	1.916	2.170	2.632	1.998	n/a	13.3	37.5	4.4	n/a
	2	4.004	4.263	5.179	3.791	n/a	6.6	29.4	−5.3	n/a	1.352	1.491	1.612	1.543	n/a	10.4	19.1	14.2	n/a
	3	3.821	4.025	4.363	3.360	n/a	5.6	14.1	−11.9	n/a	0.956	1.043	0.811	1.109	n/a	9.4	−15.3	16.2	n/a
	ALL	5.179	5.478	6.126	4.667	4.575	6.0	18.3	**−9.7**	**−11.7**	1.466	1.612	1.438	1.698	1.289	10.2	**−2.0**	15.9	**−11.9**

under the same experimental setup. The results (see right-hand side of Table II) confirmed our expectations as in this case we are able to outperform JBIG2000 also at the full bit-depth and to outperform JBIG-LS also at 4 bit-depth; moreover, JBIG gains on average, at full-bith depth, only 2% with respect to our coder.

Despite an ad hoc training set was used for each set of images, we observed that the effect of using a sub-optimal context formation procedure is marginal. For example, if the natural images were compressed with the context formation tables optimized for the depth maps, the performance losses of JBIG and JPEG2000, at full bit-depth, would drop only to 17.3% and -10.5%, respectively.

Finally, despite the complexity of the current, unoptimized implementation is somewhat higher w.r.t. JBIG or JPEG2000 (on average encoding and decoding times are $3 \div 4$ times the ones of JBIG or JPEG2000), we were still able to guarantee a satisfying encoding time of 150 and 220 ms/Mpixel/bit-plane (on a 3 MHz Pentium 4 CPU with 1 GB of RAM), respectively for 512×512 and 640×480 image sizes; the decoding times are similar.

C. Compression Performance on Detailed Binary Images

In the end, we compared the performance of the ARCABIC coder with the lifting scheme proposed in Section VI, which is referred as *lifting CA-based binary image coder* (LCABIC). To this purpose, we compressed three different sets of various-sized images with different characteristics. Set 1 [22] contains 1400 silhouette-like simple binary images, Set 2 is made of 55 comic strips[6], and Set 3 presents a gallery of 1019 clipart images[7]. Average compression results are reported in Table III for 3 and 4 levels of decomposition, for Sets 1-2 and for Set 3 (that is composed of larger images), respectively. For comparison purposes, we also reported the average compression results relative to the enhancement layers (or *differential* layers, in JBIG) only. As a remark, we noted that the best choice for the training set is to include highly detailed images: if only images from Set 1 where included, the performance would be slightly higher for that set, but sensibly lower for the other ones.

Similarly to the results above, the JBIG-KIT software [17] is employed for generating the JBIG-compliant bit streams, from which the headers are discarded before evaluating the compression performance. In particular, only the *typical prediction* of the differential layers (TPDON) has been enabled, and a single *stripe* is coded for each image, in order to simulate the same conditions under which the LCABIC coder operates.

Table III reports the average bpp values obtained by ARCABIC, LCABIC, and JBIG coders on the three data sets. It is possible to notice that the LCABIC solution permits obtaining the best performance with Set 1 and 3 for all the decompositions (made exception for the Set 2 with 2 levels of resolution). The ARCABIC approach requires at least 3 resolution levels (i.e. 2 RCAT applications) in order to be competitive with respect to JBIG and permits obtaining a maximum compression gain of 17.71% for the Set 1 with 4 resolution levels. However, this approach proves to be less effective in presence of complex images, mainly made of thin lines, since the compression efficiency of the adopted transform drastically reduces. As a result, 5 resolution levels are needed for Set 3 in order to make the performance of ARCABIC comparable to that of JBIG. The lifting scheme of LCABIC proves to be more effective since it improves the performance of JBIG in all the cases (or at least it provides approximately the same compression gain).

With respect to the computational complexity, the encoding and decoding times of the current LCABIC implementation resulted $4 \div 5$ times the ones of JBIG. As for the ARCABIC solution, computational complexity is lower since it does not include a lifting scheme. Experimental results have proved that the coding time increment for

[6]These images were obtained from the website http://www.gocomics.com/calvinandhobbes and consist of all daily-strips from April 1st to May 25th; they have been converted into true binary images by thresholding with the threshold value provided by the MATLAB function `graythresh`.

[7]These images consist of true binary images and were obtained from the website http://www.tuttoscout.org/download/clipart; more precisely, we considered all images in the archives of the first three columns of this website.

TABLE III

AVERAGE COMPRESSION PERFORMANCE (BPP) FOR THE ARCABIC, THE LCABIC, AND THE JBIG CODER ON DIFFERENT SETS OF BINARY IMAGES. THE AVERAGE SAVINGS (%) ARE GIVEN WITH RESPECT TO THE COMPRESSION PERFORMANCE OF THE JBIG CODER.

		Set 1					**Set 2**					**Set 3**				
Resolutions	**Layer(s)**	**ARCABIC**	**LCABIC**	**JBIG (reference)**	**Saving (%) (ARCABIC)**	**Saving (%) (LCABIC)**	**ARCABIC**	**LCABIC**	**JBIG (reference)**	**Saving (%) (ARCABIC)**	**Saving (%) (LCABIC)**	**ARCABIC**	**LCABIC**	**JBIG (reference)**	**Saving (%) (ARCABIC)**	**Saving (%) (LCABIC)**
2	**1**	.0162	.0116	.0185	12.34	35.30	.224	.210	.208	−8.28	−0.90	.083	.067	.071	−30.63	5.00
	ALL	.0325	.0244	.0313	−3.90	**20.79**	.331	.325	.323	−2.84	**−0.59**	.135	.115	.119	−22.59	**2.99**
3	**1**	.0173	.0238	.0467	27.57	48.56	.354	.385	.402	11.61	4.06	.151	.138	.156	−0.11	12.14
	2	.0167	.0118	.0185	9.63	34.44	.224	.210	.208	−7.63	−0.91	.083	.067	.071	−17.03	4.93
	ALL	.0322	.0247	.0372	11.14	**31.96**	.346	.349	.351	**1.3**	0.62	.141	.122	.130	−7.84	**6.30**
4	**1**	.0742	.0518	.1123	35.94	54.92	.415	.564	.637	34.74	11.36	.245	.249	.293	16.14	15.53
	2	.0343	.0245	.0467	26.7	47.07	.354	.385	.402	11.61	4.17	.151	.138	.156	0.41	12.00
	3	.0173	.0118	.0185	5.99	34.46	.224	.210	.208	−7.65	−0.90	.083	.067	.071	−17.35	4.94
	ALL	.0337	.0248	.0409	17.71	**38.04**	.348	.355	.362	**3.81**	1.90	.143	.125	.136	−4.99	**8.29**
5	**1**											.357	.391	.472	24.39	17.16
	2											.245	.249	.293	16.18	15.38
	3	–	–	–	–	–	–	–	–	–	–	.155	.138	.156	0.41	12.00
	4											.092	.067	.071	−30.63	4.94
	ALL											.144	.126	.139	−0.14	**9.22**

the ARCABIC solution w.r.t. JBIG is limited 56%, 65%, and 36% for Set 1, 2, and 3, respectively. The average encoding (or decoding) times (on a 3 MHz Pentium 4 CPU with 1 GB of RAM) resulted to be 35, 55, and 40 ms/Mpixel, respectively for Set 1, 2, and 3.

VIII. CONCLUSION

The paper presented several approaches that combine reversible cellular automata and arithmetic coding for scalable compression of binary and grayscale images. In the case of binary images, an effective approach is obtained adopting a simple *reversible cellular automata transform* and designing an effective algorithm that detects where significant data are placed. As for grayscale images, a significant improvement is given by characterizing the contexts of the arithmetic coder according to the inter-layer and inter-band correlation. The original approach is utterly improved by a lifting-based transformation that increases the compression gain of the original approach on binary images. Experimental results show that these solutions prove to be extremely competitive with respect to the state-of-the-art coder JBIG.

REFERENCES

[1] ITU-T and ISO/IEC: JTC1/SC29/WG1, "JBIG bi-level image compression standard," ITU-T Recommendation T.82, ISO/IEC 11544, 1993.

[2] G. R. Martin, R. A. Packwood, and M. K. Steliaros, "Scalable description of shape and motion for object-based coding," in *Proc. of Seventh Intl. Conf. on Image Process. and its Applications, IPA 1999 (Conf. Publ. No. 465)*, vol. 1, Manchester, UK, Jul. 13 – 15 1999, pp. 157 – 161.

[3] P. Nunes, F. Marqués, F. Pereira, and A. Gasull, "A contour-based approach to binary shape coding using a multiple grid chain code," *Signal Processing: Image Commun.*, vol. 15, no. 7-8, pp. 585–599, May 2000.

[4] S. M. Aghito and S. Forchhammer, "Context-based coding of bilevel images enhanced by digital straight line analysis," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2120–2130, Aug. 2006.

[5] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standard and Practice*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002.

[6] C. Cruz-Reyes and J. Kari, "Non-Linear Subband Coding with Cellular Automata," in *Proc. of the XII International Conference on Automata and Formal Languages (AFL 2008)*. Balatonfüred, Hungary: Computer and Automation Research Institute, Hungarian Academy of Sciences, May 27-30 2008, pp. 146–157.

[7] S. Milani, C. Cruz-Reyes, J. Kari, and G. Calvagno, "A binary image scalable coder based on reversible cellular automata transform and arithmetic coding," in *Proc. of IEEE Data Compression Conf.*, Snowbird, Utah, 16-18 Mar. 2009, p. 460.

[8] L. Cappellari and G. Calvagno, "Lifting-based design of reversible cellular automata for scalable coding of binary images," in *Proc. of IEEE Intl. Conf. on Image Process.*, Cairo, Egypt, 7-11 Nov. 2009, pp. 1901–1904.

[9] L. Cappellari, C. Cruz-Reyes, G. Calvagno, and J. Kari, "Lossy to lossless spatially scalable depth map coding with cellular automata," in *Proc. of IEEE Data Compression Conf.*, Snowbird, Utah, 16-18 Mar. 2009, pp. 332–341.

[10] J. Kari, "Theory of cellular automata: a survey," *Theor. Comput. Sci.*, vol. 334, no. 1-3, pp. 3–33, 2005.

[11] T. Toffoli and N. Margolus, "Invertible cellular automata: a review," *Physica D*, vol. 45, pp. 229–253, 1990. [Online]. Available: http://citeseer.ist.psu.edu/426861.html

[12] J. Kari, "Reversible cellular automata," in *Proceedings of DLT 2005, Developments in Language Theory. LNCS 3572*. Springer-Verlag, 2005, pp. 57–68.

[13] ISO/IEC: JTC1/SC29/WG1, "JPEG2000 image coding system: Core coding system," ISO/IEC 15444-1 (JPEG2000), 2000.

[14] I. H. Witten, R. M. Neal, and J. G. Clearly, "Arithmetic coding for data compression," *Comm. of the ACM*, vol. 30, Jun. 1987.

[15] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression*. Chichester, England; Hoboken, NJ, USA: John Wiley & Sons, 2003.

[16] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186–200, 1996.

[17] M. Kuhn, "JBIG-KIT," University of Cambridge. [Online]. Available: http://www.cl.cam.ac.uk/˜mgk25/jbigkit/

[18] M. Adams, "The JasPer project home page," Dept. of Electrical and Computer Eng., University of Victoria. [Online]. Available: http://www.ece.uvic.ca/˜mdadams/jasper/

[19] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[20] "HP labs LOCO-I/JPEG-LS home page," Hewlett-Packard Labs. [Online]. Available: http://www.hpl.hp.com/loco/

[21] "USF range image database – K2T structured light camera images," University of South Florida. [Online]. Available: http://marathon.csee.usf.edu/range/DataBase.html

[22] "Shape data for the MPEG-7 core experiment CE-Shape-1," Temple University. [Online]. Available: http://www.cis.temple.edu/˜latecki/TestData/mpeg7shapeB.tar.gz