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 scale. In this paper, we discuss the utilization of reversible cellular automata for scalable compression of binary and grayscale images. 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 scale.

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].

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 $c : \mathbb{Z}^d \rightarrow Q$. Let $C^\Lambda_Q$ denote the set of all $d$-dimensional configurations over $Q$.

The neighborhood vector $\mathbf{N} = [v_1, v_2, \ldots, v_n]^T$ of the CA is an ordered and finite list of elements of $\Lambda$ 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 $i = 1, 2, \ldots, n$. The local rule $f : Q^n \rightarrow Q$ 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 $F : C^m_Q \rightarrow C^0_Q$ defined as

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

(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 $q = [q_1, q_2, \ldots, q_m]^T \in Q^m$ 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 $c \in C^m_Q$, we denote the $m$ subbands of $c$ by $c_1, c_2, \ldots, c_m$, where $c_i \in C^m_Q$ would be the configuration of the CA if the state of each cell was given only by its $i$-th component. The local rule $f : (Q^m)^n \rightarrow Q^m$ of an $m$-band CA may be seen as composed of $m$ different local rules $f_i : (Q^m)^n \rightarrow Q$ 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. $N = \{v\}$, and the local rule is a permutation $\pi_m : Q^m \rightarrow Q^m$ of the state set. The inverse CA is obviously the trivial CA with neighborhood $N = \{-v\}$ and local rule $\pi_m^{-1}$.

An elementary $m$-band CA is a CA in which the neighborhood $N = [0, v_2, v_3, \ldots, v_m]^T$ 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 $\pi : Q \rightarrow Q$ depending on the states of the other $m-1$ subbands for the neighboring cells.

In particular, if $\Pi_Q$ denotes the set of all permutations of $Q$, $h : (Q^{m-1})^n \rightarrow \Pi_Q$ 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 $f_k$ is the identity function for each $i \neq k$, while

$$f_k(q_1, q_2, \ldots, q_n) = h(q_{1-k}, q_{2-k}, \ldots, q_{n-k})(q_{k}),$$

(2)

where $q_{-k}$ denotes the subvector of $q$ 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 $2 \times 2$ pixels are mapped to cells of a multi-band RCA (arranged over $\Lambda = 2^2 \mathbb{Z}^2$). 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.

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 $2 \times 2$ 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 touples $00, 01, 10, 11$. 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 $N = \{0\}$ and $\pi_4$ is defined as

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

(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 $\pi_4$ 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 0.368 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 $R_{00}^{(i-1)} (R_{00}^{(0)} = P$
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 $R_{10}^{(L+1-i)}$, $R_{11}^{(L+1-i)}$, 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 3, while a progressive decoder can also decode all enhancement layers. 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 3, while a progressive decoder can also decode all enhancement layers. 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 $R_{b,j}^{(i)}$ of the $i$-th enhancement layer ($b = 01, 10, 11$ and $j = L + 1 - i$), the coding algorithm identifies a positional mask $M_b$ 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 $R_{b,j}^{(i)}$ subband. Then, masks $M_b$ are created as follows

$$M_{b1} = \{(x, y) | \nabla_y (x, y) > 0 \text{ or } \nabla_x (x, y) < 0\} \quad (4)$$

where $x$ and $y$ are the horizontal and vertical coordinates in the subband $R_{b,j}^{(i)}$. 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 $R_{b,j}^{(i)}$ (see Fig. 1). As a consequence, it is convenient to enlarge the positional masks $M_b$ for each subband $b$ into the sets $M_b^+ = M_b \cup M_b^-$ according to the following operations.

For each pixel in $M_b$ the algorithm computes the vertical and the horizontal Sobel operators ($S_h(x, y)$ and $S_v(x, y)$) and defines the sets

$$M_{b0}^+ = \{(x, y) | (x, y) \in M_{b0} \text{ and } S_h(x, y) > 0\}$$
$$M_{b0}^- = \{(x, y) | (x, y) \in M_{b0} \text{ and } S_h(x, y) < 0\}$$
$$M_{b1}^+ = \{(x, y) | (x, y) \in M_{b1} \text{ and } S_v(x, y) > 0\}$$
$$M_{b1}^- = \{(x, y) | (x, y) \in M_{b1} \text{ and } S_v(x, y) < 0\}$$

which permit extending the pixel maps as

$$M_{b0}^+ = \{M_{b0}^+ \oplus \{(1, 0)\} \cup \{M_{b0}^- \oplus \{(-1, 0)\}\}$$
$$M_{b1}^+ = \{M_{b1}^+ \oplus \{(0, 1)\} \cup \{M_{b1}^- \oplus \{(0, -1)\}\}$$

Fig. 2. Binary image representation in various domains.

Fig. 3. Coding contexts for the LFC and the HFC algorithm.
where $\oplus$ denote the Minkowski sum (morphological dilation). Fig. 4 shows the subbands $R_{01}^{(1)}, R_{10}^{(1)}, R_{11}^{(1)}$ relative to a detail of the image child underlining the pixels to be coded and the pixel mask $M_x$. 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 $M_x$ 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 $R_{01}^{(1)}$ and $R_{11}^{(1)}$ subbands and in a column-wise order for the $R_{10}^{(1)}$ subband. For each row (column), the encoder checks whether it contains positions $(x,y)$ such that $R_{0}^{(1)}(x,y) \neq 0$ and $(x,y) \in M_x(x,y)$. 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 $M_x$ are coded by the same arithmetic coder using the pixels $R_{00}^{(1)}(x,y)$, $R_{00}^{(1)}(x,y-1)$, $R_{00}^{(1)}(x-1,y)$, $R_{00}^{(1)}(x-1,y-1)$, $R_{00}^{(1)}(x-1,y)$, and the number $d$ of pixels interlying between the current pixels and the last pixel outside $M_x$ in the current row (column). Fig. 3(b) graphically displays the coding contexts for the subband $R_{00}^{(1)}$ 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 $M_x$. More precisely, the encoder partitions the input subband $R_{0}^{(1)}$ into blocks of $32 \times 32$ pixels and computes how much (in terms of pixels) the corresponding mask $M_x$ 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.

![Fig. 4. Subbands of the last enhancement layer (black pixels) and the support points ($M_x$) identified by the segmentation routine (blue points): points of the signals that do not lie within $M_x$ are marked with red $\times$s.](image-a)

![Fig. 5. Inter context windows used for context formation.](image-b)

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 $B_i'$) 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, $E_i = B_i \oplus B_{i+1}$, where $\oplus$ denotes difference in the binary domain. Then, each $E_i$ is independently transformed (into $E_i'$) with the local rule, and entropy coded.

In Algorithm 1, a substantial amount of correlation remains between $B_i'$ and $B_{i+1}'$, in particular for smooth images such as depth maps. In practice, most of the features in $B_i'$ occur at the same position of the features of $B_{i+1}'$, 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 $H(B_t'|B_{t+1}')$ and $H(E_t'|E_{t+1}')$ (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 $R_{00}^{(L-j)}$ subband of $B_t'$ (denoted by $R_{00}^{(L-j)}$) 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 $R_{00}^{(L-j)}$ subband of $E_t'$ (again, denoted by $R_{00}^{(L-j)}$).

In the RCAT domain, context-adaptive arithmetic coding is employed to code the samples in the various subbands of $B_t'$ (or $E_t'$). The subbands are always coded in a resolution-progressive order: data belonging to subband $R_{00}^{(L)}$ are encoded first for $i = N-1$, $N-2$, $\ldots$, 0; then, data needed for reconstruction of subband $R_{00}^{(L-j)}$ are encoded (with the same bit-plane order), for $j = 1, 2, \ldots, L$. Inside each resolution, subbands are scanned in this order: $R_{01}^{(L)}$, $R_{10}^{(L)}$ and $R_{11}^{(L)}$, similarly to the processing order used in the standard JPEG2000 [5]. Inside subbands $R_{01}^{(L)}$ (and $R_{10}^{(L)}$), samples are coded in raster column order (for following vertical features); inside subbands $R_{11}^{(L)}$, samples are coded in raster row order (for following horizontal features); inside subbands $R_{11}^{(L)}$ 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 $3 \times 3$ context window in the corresponding subband of $B_{i+1}^{(1)}$ (or $B_{i+1}^{(1)}$), to capture the inter-plane (IP) correlation ($i < N-1$), (ii) a $3 \times 3$ context window in the $B_0^{(1)}$ subband, to capture the inter-resolution (IR) correlation of subbands $R_{01}^{(L)}$, $R_{10}^{(L)}$, and $R_{11}^{(L)}$, and (iii) a $3 \times 3$ context window in the $B_0^{(1)}$ subband and another $3 \times 3$ context window in the $R_{00}^{(1)}$ subband (if it has already been coded), to capture the residual inter-band (IB01 or IB10) correlation that the subbands $R_{01}^{(1)}$ and $R_{11}^{(1)}$ may have with subbands relative to the same bit-plane and resolution. In addition,
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 (\(2^{24}\cdot 9+8 = 2^{14}\)) 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 \(R_{00}^{(4)}, R_{01,N-1}^{(4)}, R_{01,N-1}^{(8)}, R_{10,N-1}^{(8)}, R_{11,N-1}^{(8)}, R_{00}^{(12)}, R_{01,N-1}^{(12)}, \) and \(R_{10,N-1}^{(12)}\) subbands, respectively; 8, 8, 10, and 10 contexts are used for coding the \(R_{00}^{(10)}, R_{01}^{(10)}, R_{10}^{(10)}, R_{11}^{(10)}, R_{11}^{(12)}\), and subbands \(i \leq N-1\), 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.

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 \((0,1), (1,0), (1,1)\) 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

\(^1\)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.

\(^3\)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. 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.

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)\(^5\).

\(^5\)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.
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-bit 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 used for the depth, would drop only to 0.25. For example, if the natural images we re

\[
\text{loss}(\%) = \frac{\text{Loss}}{\text{Length}} \times 100
\]

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`.

6These 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`.

C. Compression Performance on Detailed Binary Images

In the end, we compared the performance of the ARCA integrity coder with the lifting scheme proposed in Section VI, which is referred as **lifting CA-based binary image coder (LCA-CA)**. To this purpose, we compared 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 ARCA, LCA-CA, 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 ARCA 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 ARCA 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 3.5 times the ones of JBIG. As for the ARCA solution, computational complexity is lower since it does not include a lifting scheme. Experimental results have proved that the coding time increment for

<table>
<thead>
<tr>
<th>Bitplanes</th>
<th>Layer(s)</th>
<th>Algorithm 2 (reference)</th>
<th>Algorithm 1</th>
<th>JBIG</th>
<th>JBIG2000</th>
<th>JBIG-LS</th>
<th>Loss (%) (Algorithm 1)</th>
<th>Loss (%) (JBIG)</th>
<th>Loss (%) (JPG2000)</th>
<th>Loss (%) (JPEG-LS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0.476 0.542 0.726 0.782</td>
<td>n/a</td>
<td>13.0</td>
<td>50.9</td>
<td>64.6</td>
<td>n/a</td>
<td>0.076 0.078 0.118</td>
<td>0.173</td>
<td>3.3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.401 0.441 0.502 0.604</td>
<td>n/a</td>
<td>9.9</td>
<td>23.3</td>
<td>52.7</td>
<td>n/a</td>
<td>0.040 0.041 0.056</td>
<td>0.098</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.337 0.360 0.312 0.454</td>
<td>n/a</td>
<td>7.4</td>
<td>-8.4</td>
<td>38.0</td>
<td>n/a</td>
<td>0.024 0.025 0.025</td>
<td>0.056</td>
<td>4.2</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>0.479 0.519 0.501 0.671</td>
<td>0.620 8.7</td>
<td>3.7</td>
<td>42.9</td>
<td>35.2</td>
<td>n/a</td>
<td>0.042 0.044 0.051</td>
<td>0.096 0.092</td>
<td>3.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitplanes</th>
<th>Layer(s)</th>
<th>Algorithm 2 (reference)</th>
<th>Algorithm 1</th>
<th>JBIG</th>
<th>JBIG2000</th>
<th>JBIG-LS</th>
<th>Loss (%) (Algorithm 1)</th>
<th>Loss (%) (JBIG)</th>
<th>Loss (%) (JPG2000)</th>
<th>Loss (%) (JPEG-LS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>0.828 0.959 1.246 1.146</td>
<td>n/a</td>
<td>15.2</td>
<td>49.8</td>
<td>41.5</td>
<td>n/a</td>
<td>0.131 0.140 0.202</td>
<td>0.306</td>
<td>11.3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.701 0.791 0.877 0.900</td>
<td>n/a</td>
<td>12.9</td>
<td>23.7</td>
<td>31.6</td>
<td>n/a</td>
<td>0.072 0.080 0.097</td>
<td>0.174</td>
<td>10.4</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>0.844 0.945 0.888 1.003</td>
<td>0.908 12.3</td>
<td>4.0</td>
<td>22.6</td>
<td>11.4</td>
<td>n/a</td>
<td>0.076 0.084 0.089</td>
<td>0.172</td>
<td>10.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitplanes</th>
<th>Layer(s)</th>
<th>Algorithm 2 (reference)</th>
<th>Algorithm 1</th>
<th>JBIG</th>
<th>JBIG2000</th>
<th>JBIG-LS</th>
<th>Loss (%) (Algorithm 1)</th>
<th>Loss (%) (JBIG)</th>
<th>Loss (%) (JPG2000)</th>
<th>Loss (%) (JPEG-LS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1.355 1.543 1.998 1.603</td>
<td>n/a</td>
<td>15.9</td>
<td>49.7</td>
<td>23.3</td>
<td>n/a</td>
<td>0.222 0.263 0.335</td>
<td>0.489</td>
<td>18.7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.157 1.311 1.469 1.310</td>
<td>n/a</td>
<td>13.7</td>
<td>25.9</td>
<td>15.4</td>
<td>n/a</td>
<td>0.127 0.149 0.166</td>
<td>0.287</td>
<td>17.8</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>1.417 1.594 1.524 1.465</td>
<td>1.322 13.2</td>
<td>6.4</td>
<td>5.9</td>
<td>-4.7</td>
<td>n/a</td>
<td>0.134 0.157 0.152</td>
<td>0.290 0.194</td>
<td>17.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitplanes</th>
<th>Layer(s)</th>
<th>Algorithm 2 (reference)</th>
<th>Algorithm 1</th>
<th>JBIG</th>
<th>JBIG2000</th>
<th>JBIG-LS</th>
<th>Loss (%) (Algorithm 1)</th>
<th>Loss (%) (JBIG)</th>
<th>Loss (%) (JPG2000)</th>
<th>Loss (%) (JPEG-LS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>4.241 4.542 5.831 4.142</td>
<td>n/a</td>
<td>7.2</td>
<td>37.9</td>
<td>-2.1</td>
<td>n/a</td>
<td>1.916 2.170 2.632</td>
<td>1.998</td>
<td>13.3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.004 4.265 5.179 3.791</td>
<td>n/a</td>
<td>6.6</td>
<td>29.4</td>
<td>-5.3</td>
<td>n/a</td>
<td>1.352 1.491 1.612</td>
<td>1.543</td>
<td>10.4</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>5.179 5.478 6.126 4.667</td>
<td>4.575 6.0</td>
<td>18.3</td>
<td>-9.7</td>
<td>-11.7</td>
<td>n/a</td>
<td>1.466 1.612 1.438</td>
<td>1.698</td>
<td>12.8</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Resolutions</th>
<th>Layer(s)</th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ARCABIC</td>
<td>LCABIC</td>
<td>JBIG (reference)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.162</td>
<td>0.116</td>
<td>0.185</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>0.322</td>
<td>0.247</td>
<td>0.031</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.173</td>
<td>0.238</td>
<td>0.067</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.076</td>
<td>0.118</td>
<td>0.045</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.322</td>
<td>0.247</td>
<td>0.037</td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td>0.337</td>
<td>0.248</td>
<td>0.049</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 automaton 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