Real-Time Imaging 1, 339–353 (1995)

A Novel Approach to Lossy Real-Time Image Compression: Hierarchical Data Reorganization on a Low-Cost Massively Parallel System

This paper discusses an innovative real-time oriented image compression system, based on a simple algorithm designed explicitly to be implemented on a low-cost SIMD computer architecture featuring a much lower power consumption than traditional DSPs or dedicated hardware. For this reason the considered approach is suitable to be integrated on portable systems, where power consumption is a critical design issue.

The algorithm, based on a hierarchical decomposition of the input image, has been tested on a special purpose SIMD system, PAPRICA, exploiting its features such as its massive parallelism and its capability to operate on local data as well as to handle hierarchical data structures.

According to the proposed approach, the quality of the decompressed image can be traded for a lower power consumption as well as a higher processing speed. A comparison between the discussed algorithm and the standard JPEG compression technique is also presented.

Alberto Broggi

Dipartimento di Ingegneria dell'Informazione
Università di Parma, Italy
E-Mail: broggi@CE.UniPR.IT

Introduction and Motivation

One of the fastest growing segments of the electronic industry is represented by Portable Communication Systems (PCS), integrating the facilities offered by personal and laptop computers together with the capabilities of modern multimedia systems. The PCS demand of computational power is continuously increasing, and it will become even more cumbersome with the integration of future PCS applications, where not only voice will be transmitted, but also text, still images, full motion videos, and data in general (1).

Thus, a system (hardware and software) for real-time video compression and decompression will be soon required by PCS: it must be carefully optimized taking into account the extra power it requires. A key parameter in the design of the video processing subsystem, as well as portable equipment in general, is in fact the power consumption of the system.

The power consumption of dynamic systems can be considered proportional to CfV^2, where C represents the capacitance of the circuit, f is the clock frequency, and V is the voltage swing. Power can be saved in three different ways (2), minimizing C, f, and V respectively:

(i) using a greater level of VLSI integration, thus reducing the capacitance, C;
(ii) trading computer speed (using a lower clock frequency, f) for a lower power consumption (already implemented on many portable PCs);
(iii) reducing the supply voltage, V_{DD}.

Recently, new technological solutions have been exploited to reduce the IC supply voltage from 5 to 3.3 V. Unfortunately, there is a speed penalty to pay for this reduction: for a CMOS gate (3), the device delay, T_d, (following a first order approximation) is proportional to

$$\frac{V_{DD}}{(V_{DD}-V_T)^2},$$

which shows that the reduction of V_{DD} determines a quasi-linear increment (until the device threshold value V_T) of the circuit delay, T_d. On the other hand, the reduction of V_{DD} determines a quadratic reduction of the power consumption. Thus, for power saving reasons, it is desirable to operate at the lowest possible speed, but, in order to maintain the overall system performance, a compensation for these increased delays is required.

The use of a lower power supply voltage is investigated by Courtois (4) and Chandrakasan *et al.* (5), where different architectural solutions are considered in order to overcome the undesired side-effects caused by the reduction of V_{DD}. The reduction of power consumption while maintaining computational power can be obtained through the use of low cost SIMD computer architectures, formed by a large number of extremely simple and relatively slow-clocked Processing Elements (PEs). These systems, using slower device speeds, provide an effective mechanism to trade power consumption for silicon area, while maintaining the computational power unchanged. The four major drawbacks of this approach are:

(i) a solution based on hardware replication increases the silicon area, and thus it is not suited for extremely area-constrained designs;
(ii) parallelism must be accompanied by extra-routing, requiring extra power; this issue must be carefully considered and optimized;
(iii) the use of parallel computer architectures involves the redesign of the algorithms with a different computational model;
(iv) since the number of processing units must be high, if the system has size constraints the PEs must be extremely simple, performing only simple basic operations.

This paper investigates a novel approach to real-time image compression and recovery based on the use of a massively parallel system and a data-parallel algorithm. In fact, the traditional sequential algorithms running on power-consuming systems based on expensive hardware platforms cannot meet the specific requirements of real-time processing on low-power and low-cost architectures. The algorithm discussed in this work has the following advantages over the traditional transform-based compression algorithms [such as JPEG (6)]:

(i) it can be efficiently expressed by a fine grain data-parallel computational model;
(ii) it makes use of extremely simple operations only (additions, subtractions, and comparisons);
(iii) the output quality can be traded for a higher processing speed and lower power consumption.

Algorithm Structure

This section presents a parallel algorithm for lossy data reorganization which, in conjunction with conventional lossless serial compression algorithms (7), delivers quite promising performances in terms of both processing time and output quality. The underlying basic structure of the discussed approach is common to the one utilized by the JPEG standard; it can be divided into three phases:

(i) lossy filtering of the image data;
(ii) scanning of the 2D image array to produce a 1D data stream;
(iii) final compression of the 1D data stream by means of traditional lossless techniques.

In this work only Phase 1 and Phase 2 have been redesigned according to the specific requirements presented in the previous section, while phase 3 is implemented utilizing the same techniques of the JPEG standard (serial lossless compression) such as Huffman coding, arithmetic coding, or LZW coding (7).

Phase 1, which acts essentially as a low-level preprocessing filter, has been tested on PAPRICA (8, 9), exploiting its specific features such as its massive parallelism (256 PEs), its ability to handle hierarchical data structures (10, 11) efficiently, and its simple morphological instruction set (12, 13). As shown in Figure 1, images are acquired by a camera, and preprocessed by the massively parallel PAPRICA system with simple low-level computations (Phase 1). The output data structure, still a 2D array of values, is scanned serially (Phase 2) and then compressed by means of traditional sequen-

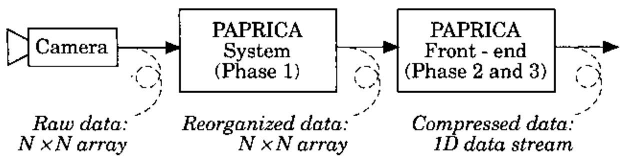

Camera
PAPRICA System (Phase 1)
PAPRICA Front - end (Phase 2 and 3)
Raw data: N ×N array
Reorganized data: N ×N array
Compressed data: 1D data stream

Figure 1. Block diagram of data acquisition and processing with PAPRICA system

tial algorithms (Phase 3) by PAPRICA front-end (The considered hardware system is presented in detail later).

The Data Reorganization Procedure

This section describes the data reorganization process (Phase 1), which is divided into two steps (lossless reorganization and non-linear filtering), and presents their hierarchical extension.

Let $\mathcal{F}^{N\times N} \subset \mathcal{N}^2$ be the set of all $N \times N$ frames, and consider the original image $P \in \mathcal{F}^{N\times N}$ formed by pixels $p_{i,j}$, with $0 \le i,j < N$. Image P is the input image to the lossy data reorganization procedure.

Assuming N even, the first transformation applied to image P, is a decomposition denoted by $\mathcal{D}(\cdot) : \mathcal{F}^{k\times k} \to \mathcal{F}^{k\times k}$:

$$P' = \mathcal{D}(P), \quad \text{where } P, P' \in \mathcal{F}^{N\times N}, \qquad [1]$$

and defined as follows:

$$p'_{i,j} = \begin{cases} p_{i,j} & \text{when both } i \text{ and } j \text{ are even} \\ \dfrac{p_{i-1,j}+p_{i+1,j}}{2} - p_{i,j} & \text{when } i \text{ is odd and } j \text{ is even} \\ & \text{for } 0 \le i,j < N, \\ \dfrac{p_{i,j-1}+p_{i,j+1}}{2} - p_{i,j} & \text{when } i \text{ is even and } j \text{ is odd} \\ \dfrac{p_{i-1,j-1}+p_{i+1,j+1}}{2} - p_{i,j} & \text{when both } i \text{ and } j \text{ are odd} \end{cases} \qquad [2]$$

where $p_{i,j} = 0$, if either $i \ge N$ or $j \ge N$, thus assuming the image surrounded by zeros.

As shown in the following figure, image P' can be considered as being composed of four interleaved images: $S, X, Y, Z \in \mathcal{F}^{\frac{N}{2}\times\frac{N}{2}}$, defined as:

$$S = \left\{ s_{i,j} = p'_{2i,2j} \in P', \text{ for } 0 \le i,j < \frac{N}{2} \right\}$$

$$X = \left\{ x_{i,j} = p'_{2i+1,2j} \in P', \text{ for } 0 \le i,j < \frac{N}{2} \right\} \qquad [3]$$

$$Y = \left\{ y_{i,j} = p'_{2i,2j+1} \in P', \text{ for } 0 \le i,j < \frac{N}{2} \right\}$$

$$Z = \left\{ z_{i,j} = p'_{2i+1,2j+1} \in P', \text{ for } 0 \le i,j < \frac{N}{2} \right\}$$

Image S represents the result of a mere 1:2 subsampling of image P, while each pixel of X, Y, and Z represents the deviation from a local linear signal in P (represented by the average between the pixel's neighbors in the i, j, and ij directions respectively). Obviously, there exists an inverse transform that, starting from P', recovers P: $\mathcal{R}(\cdot) : \mathcal{F}^{k\times k} \to \mathcal{F}^{k\times k}$:

$$R = \mathcal{R}(P'), \quad \text{with } P', R \in \mathcal{F}^{N\times N}, \qquad [4]$$

where $R = P$. It is defined as:

$$r_{i,j} = \begin{cases} p'_{i,j} & \text{when both } i \text{ and } j \text{ are even} \\ \dfrac{p'_{i-1,j}+p'_{i+1,j}}{2} - p'_{i,j} & \text{when } i \text{ is odd and } j \text{ is even} \\ & \text{for } 0 \le i,j < N. \\ \dfrac{p'_{i,j-1}+p'_{i,j+1}}{2} - p'_{i,j} & \text{when } i \text{ is even and } j \text{ is odd} \\ \dfrac{p'_{i-1,j-1}+p'_{i+1,j+1}}{2} - p'_{i,j} & \text{when both } i \text{ and } j \text{ are odd} \end{cases} \qquad [5]$$

The advantage of image P' with respect to the original image P is that its histogram is much more polarized around low values, basically because three-quarters of its pixels now represent only variations from a common (due to spatial correlation) local behaviour (linear signal). Nevertheless, the effectiveness of the compression of image P' depends on the specific mapping of the 2D image array to a 1D data stream. Section 2.2 presents a possible mapping and compares its effectiveness to the conventional left-to-right and top-to-bottom scanning.

Non-linear filtering

The contributions of X, Y, and Z to the image histogram can be of two different kinds:

(i) low values, meaning small variations from the local linear behaviour of the image brightness;
(ii) high values, given by pixels which carry significant information such as edges (brightness discontinuities).

In order to improve the compression ratio further, a non-linear filtering is performed on subimages X, Y, and Z.

The small values in X, Y, or Z can be reset to zero, causing an obvious (but limited) information loss, but allowing a higher compression ratio due to the considerable number of equal values (zeros) in the image histogram. The smaller the values which are reset to zero, the lower the information loss but the lower the compression improvement. Conversely, the preservation of high values tends to keep the information content unchanged.

Images X, Y, and Z are in fact filtered by a non-linear pointwise operator $\mathcal{T}t(\cdot) : \mathcal{F}^{k\times k} \rightarrow \mathcal{F}^{k\times k}$ which is depicted in Figure 2:

$$\tilde{X}\ \mathcal{T}_t(X)\ ,\ \tilde{Y}\ \mathcal{T}_t(Y)\ ,\ \tilde{Z} = \mathcal{T}_t(Z)\ , \qquad [6]$$

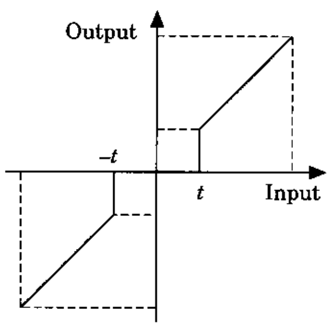


Figure 2. The non-linear operator implemented in the $\mathcal{T}(\cdot)$ filter

defined as:

$$\tilde{x}_{i,j} = \begin{cases} x_{i,j} & \text{if } |x_{i,j}| \geq t \\ 0 & \text{if } |x_{i,j}| < t \end{cases} \quad \tilde{y}_{i,j} = \begin{cases} y_{i,j} & \text{if } |y_{i,j}| \geq t \\ 0 & \text{if } |y_{i,j}| < t \end{cases} \quad \tilde{z}_{i,j} = \begin{cases} z_{i,j} & \text{if } |z_{i,j}| \geq t \\ 0 & \text{if } |z_{i,j}| < t \end{cases}, \qquad [7]$$

for $0 \leq i,j < \frac{N}{2}$, which deletes the small (less than threshold t) local deviations from a linear signal and preserves the information content carried by high brightness discontinuities (higher than threshold t). This transformation is lossy, and the information content that is lost cannot be fully recovered by the application of the inverse transform $\mathcal{R}(\cdot)$. In fact $\mathcal{R}(\cdot)$, applied to image $\tilde{P}$ (composed of subimages $\tilde{S}$, $\tilde{X}$, $\tilde{Y}$, and $\tilde{Z}$), is able to reconstruct only an approximated version $\tilde{R}$ of the original image P.

Image Restoration

It is anyway important to note that a zero in a $\tilde{X}$, $\tilde{Y}$, or $\tilde{Z}$ subimage generally comes from the non-linear cut-off due to the application of $\mathcal{T}_t(\cdot)$. In this case, the reconstruction of the value of a generic pixel $p_{i,j}$ (with i and/or j odd) is affected by an approximation which may range from $-t$ to $+t$. In order to maintain the information about this error, during the recovery process two images are generated: the approximated version of the original image, $\tilde{R}$, and an error image, E, encoding the modulus of the maximal error which could affect the corresponding pixels in the recovered image $\tilde{R}$. The difference, D, between images P and $\tilde{R}$

$$D = P - \tilde{R} = \left\{ d_{i,j} = p_{i,j} - \tilde{r}_{i,j} \text{ for } 0 \leq i,j < N \right\}, \qquad [8]$$

is such that $|d_{i,j}| \leq e_{i,j}$, for $0 \leq i,j < N$. Let us consider a generic image $K \in \mathcal{F}^{N\times N}$ such that $k_{i,j} = \tilde{r}_{i,j} + h_{i,j}$, for $0 \leq i,j < N$ and for any $|h_{i,j}| \leq e_{i,j}$. Due to the specific definition of the generic image K, the application of $\mathcal{D}(\cdot)$ and $\mathcal{T}_t(\cdot)$ to K again produces image $\tilde{R}$:

$$\mathcal{T}_t[\mathcal{D}(K)] = \tilde{R}\ . \qquad [9]$$

Thus, the couple of images $\tilde{R}$ and E represents a family of images $K_1, K_2, \ldots, K_w$ which[†], by the application of $\mathcal{D}(\cdot)$ and $\mathcal{T}_t(\cdot)$, may have generated set $\{S, \tilde{X}, \tilde{Y}, \tilde{Z}\}$. A further operation $\mathcal{C}(\cdot) : \mathcal{F}^{k\times k} \rightarrow \mathcal{F}^{k\times k}$ is thus needed to choose the one maximizing the *a posteriori* probability, or, in other words, the one that best matches some specific criteria determined by *a priori* knowledge on the image. The assumptions used in this restoration process depend on the specific application: the restoration filters can range from image smoothing (continuity constraint), edge enhancement [weak continuity (14, 15) constraint], median filtering (to reduce noise spikes).

Extension to a Hierarchical Process

Provided that $N = 2^n$, with $n \in \mathcal{N}$, and indicating with $S^{(0)}$ the original image P, decomposition $\mathcal{D}(\cdot)$ can be iterated at most n times: first on $S^{(0)}$ to produce $S^{(1)}, X^{(1)}, Y^{(1)}, Z^{(1)} \in \mathcal{F}^{\frac{N}{2}\times\frac{N}{2}}$, then on $S^{(1)}$ to produce $S^{(2)}, X^{(2)}, Y^{(2)}, Z^{(2)} \in \mathcal{F}^{\frac{N}{4}\times\frac{N}{4}}$, until $S^{(n)}, X^{(n)}, Y^{(n)}, Z^{(n)} \in \mathcal{F}^{1\times 1}$. After the hierarchical decomposition, the non-linear filter $\mathcal{T}_t(\cdot)$ is applied n times with a set of different thresholds $t^{(1)}, t^{(2)}, \ldots, t^{(n)}$. As an example, Figure 3 shows a block diagram of two iterations of the hierarchical decomposition and lossy filtering, while Figure 4 shows the corresponding hierarchical image recovery procedure. As in the previous mono-resolution case, at each iteration of the recovery process two images are generated (the reconstructed image, $\tilde{R}$, and the error image, E):

[†] From the specific definition of the generic image K, it follows that $\Omega = \prod_{0\leq i,j<N} e_{i,j}$.

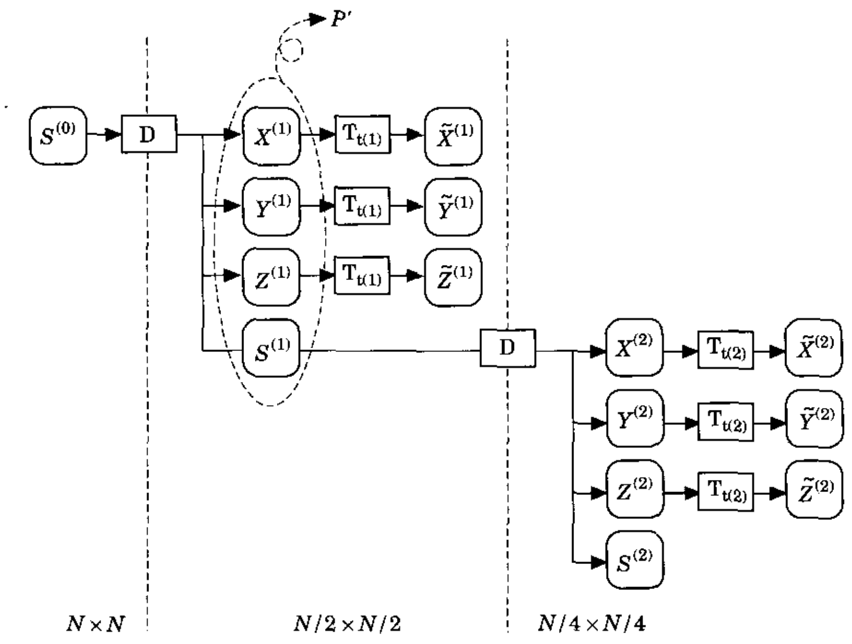


Figure 3. Two iterations of the hierarchical process

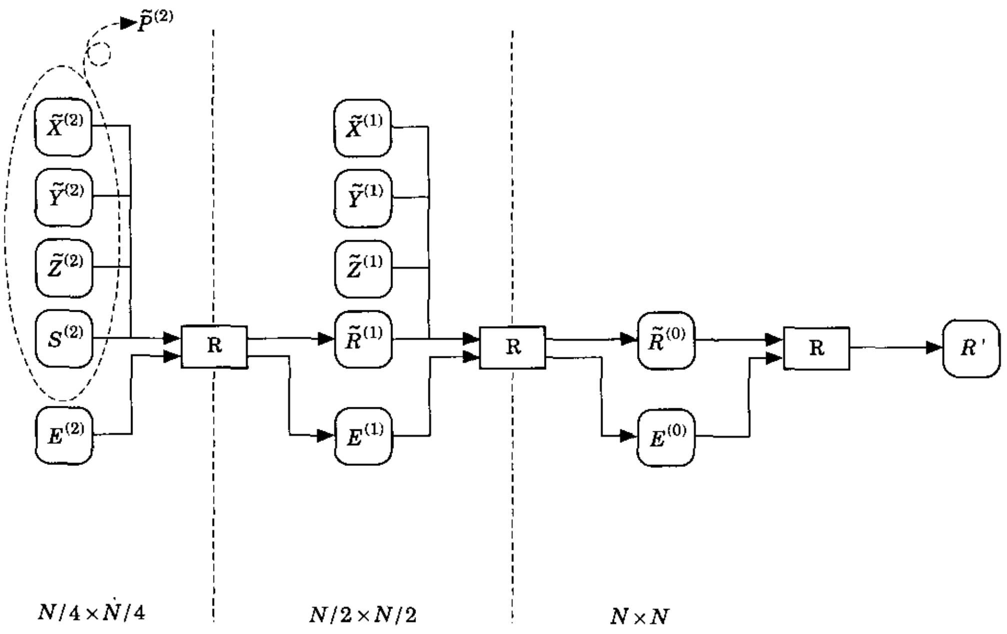


Figure 4. Two iterations of the recovering procedure, followed by the restoration process

$$\tilde{r}_{i,j}^{(k)} = \begin{cases} \tilde{p}_{i,j}^{(k+1)} & \text{when both } i \text{ and } j \text{ are even} \\ \dfrac{\tilde{p}_{i-1,j}^{(k+1)} + \tilde{p}_{i+1,j}^{(k+1)}}{2} - \tilde{p}_{i,j}^{(k+1)} & \text{when } i \text{ is odd and } j \text{ is even} \\ \dfrac{\tilde{p}_{i,j-1}^{(k+1)} + \tilde{p}_{i,j+1}^{(k+1)}}{2} - \tilde{p}_{i,j}^{(k+1)} & \text{when } i \text{ is even and } j \text{ is odd} \\ \dfrac{\tilde{p}_{i-1,j-1}^{(k+1)} + \tilde{p}_{i+1,j+1}^{(k+1)}}{2} - \tilde{p}_{i,j}^{(k+1)} & \text{when both } i \text{ and } j \text{ are odd} \end{cases} \quad \text{for } 0 \le i, j < \frac{N}{2^k} \text{ and}$$

$$\tilde{e}_{i,j}^{(k)} = \begin{cases} e_{\frac{i}{2},\frac{j}{2}}^{(k+1)} & \text{when both } i \text{ and } j \text{ are even} \\ \dfrac{e_{\frac{i-1}{2},\frac{j}{2}}^{(k+1)} + e_{\frac{i+1}{2},\frac{j}{2}}^{(k+1)}}{2} + t^{k+1}\delta\left[\tilde{p}_{i,j}^{(k+1)}\right] & \text{when } i \text{ is odd and } j \text{ is even} \\ \dfrac{e_{\frac{i}{2},\frac{j-1}{2}}^{(k+1)} + e_{\frac{i}{2},\frac{j+1}{2}}^{(k+1)}}{2} + t^{k+1}\delta\left[\tilde{p}_{i,j}^{(k+1)}\right] & \text{when } i \text{ is even and } j \text{ is odd} \\ \dfrac{e_{\frac{i-1}{2},\frac{j-1}{2}}^{(k+1)} + e_{\frac{i+1}{2},\frac{j+1}{2}}^{(k+1)}}{2} + t^{k+1}\delta\left[\tilde{p}_{i,j}^{(k+1)}\right] & \text{when both } i \text{ and } j \text{ are odd} \end{cases} \quad \text{for } 0 \le i, j < \frac{N}{2^k} ,$$

where the Dirac function δ(x) is defined as follows:

$$\delta(x) = \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ otherwise} \end{cases} . \qquad [12]$$

Expression [10] is the hierarchical extension of Equ [5], while Equ [11] has been obtained by considering that the approximation in the determination of a pixel value is composed of two contributions:

(i) a function (average) of the interpolating values, and
(ii) in case the exact information has been removed through the application of the threshold, the threshold value itself.

Figure 5 shows two iterations of the discussed procedure on a linear profile of an image.

As an example, Figure 7 shows the effect of two iterations of the lossless decomposition on the original image of Figure 6. Figure 8 presents the set of 10 images generated by three iterations of the lossless decomposition followed by the non-linear filtering with thresholds $t^{(1)} = 60$, $t^{(2)} = 40$, and $t^{(3)} = 10$. The compression ratio achieved by the LZW algorithm on each image is also presented.

The choice of the set of thresholds, as well as the specific

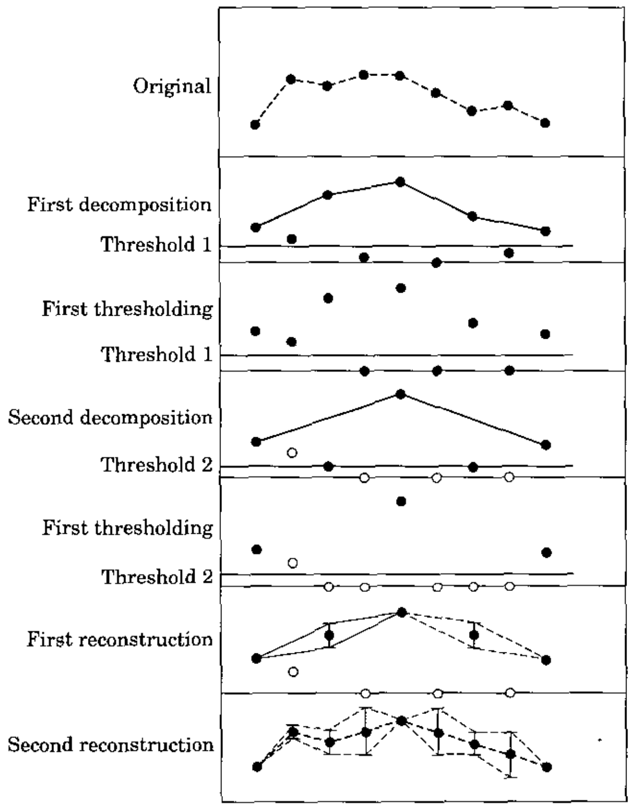


Figure 5. Two iterations of the discussed procedure on a linear profile of an image. The active points are shown in black

Figure 6. Original Lena image: 512 × 512 8 bit/pixel. The lossless compression of this image with the LZW algorithm reaches about 6·67 bit/pixel

non-linear filter, are key points in the lossy process. In Mordonini (16) the subjective quality of the recovered images has been correlated to the number of hierarchical iterations and to the set of thresholds in order to devise an automatic algorithm for their determination as a function of both the desired output quality and the required compression.

Different non-linear filters can be used, depending on the specific characteristics of the images to be compressed. For example a filter which introduces a high quantization (smaller number of bit/pixel) to represent images $X^{(1)}$, $Y^{(1)}$, and $Z^{(1)}$ and increases the number of bit/pixel for the successive hierarchical iterations is currently under evaluation and testing for videoconference applications (facial images).

Mapping a 2D image to a 1D data stream

The mapping of the 2D image array to a stream of values is a critical phase, which determines the effectiveness of the subsequent lossless compression. With a LZW algorithm, the best compression can be reached when the data stream is formed by multiple occurrencies of the same chains of values; the longer the chains, the higher the compression. This feature is achieved by the JPEG standard with a zigzag scanning of each sub-block in which the image is partitioned (6). This specific scanning assures that in the final data stream similar values are grouped in nearby positions.

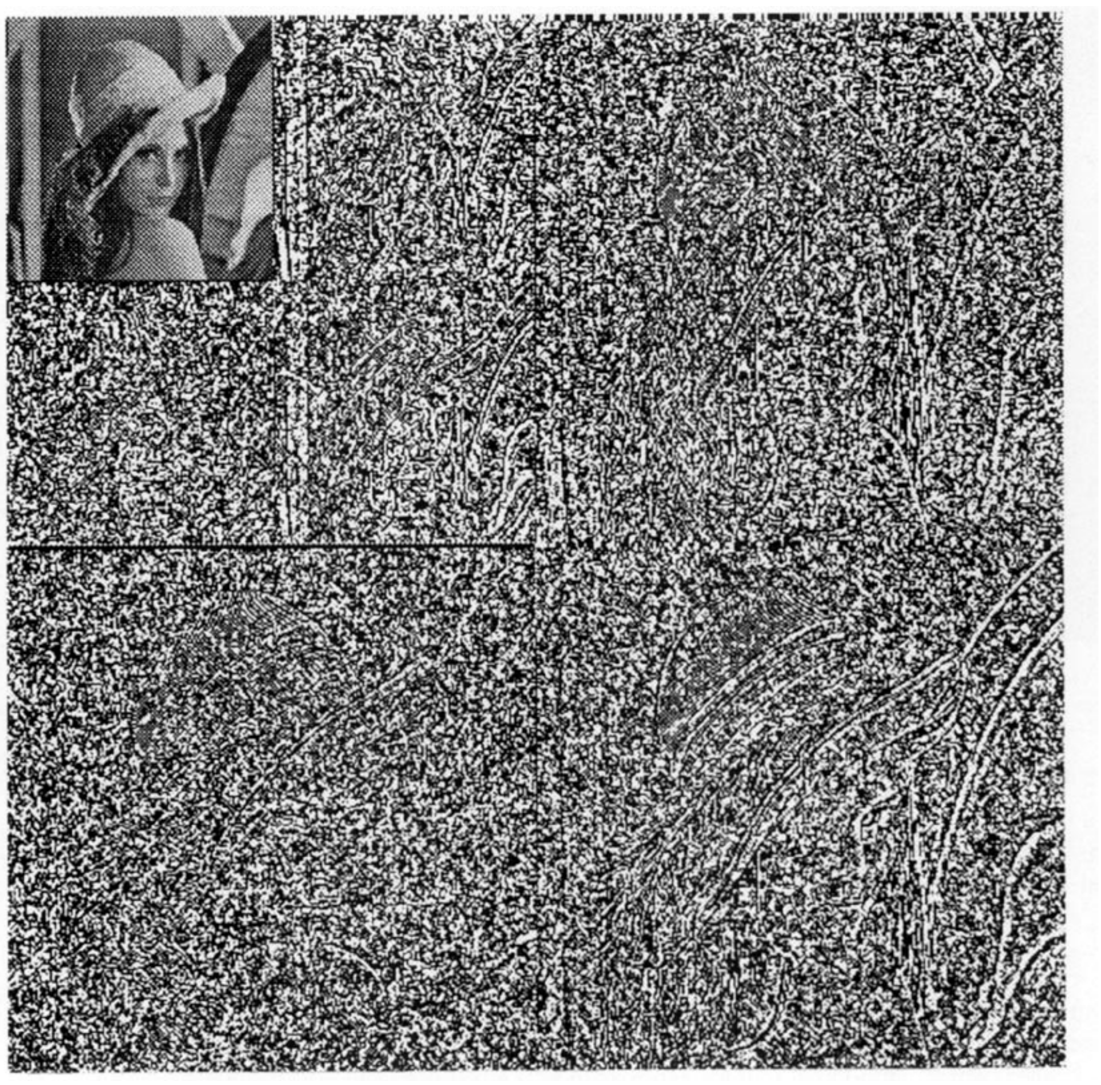
The traditional left-to-right and top-to-bottom scanning applied to the original image and to the image obtained by the discussed reorganization procedure provides two data streams whose compression is not sensibly different. This is due to the fact that the subimages which compose the reorganized image are interleaved. As an example, the first column of Table 1 shows the effect of the compression of the data streams generated by a left-to-right and top-to-bottom scanning on the original image (depicted in Figure 6) and on the images obtained after 1,2,3,4, and 5 *lossless* hierarchical decomposition (Figure 7 shows the set of the seven images $S^{(2)}$, $X^{(2)}$, $Y^{(2)}$, $Z^{(2)}$, $X^{(1)}$, $Y^{(1)}$, $Z^{(1)}$, produced by two iterations of the *lossless* decomposition). The improvement in the compression of the reorganised images (1,2,3,4, or 5 iterations) with respect to the compression of the original one (0 iterations) is due to the higher polarization of the image histogram around low values.

Different mappings $\mathcal{M}(\cdot) : \mathcal{F}^{k \times k} \to \mathcal{F}^{k^2 \times 1}$ have been considered and tested: their common underlying structure is aimed at the grouping of pixels with similar values in subsequent clusters. The one which has been proven to be the most effective is based on the left-to-right and top-to-bottom scanning of the subsampled image, S, first, followed by the contemporary scanning of the other three images $\tilde{X}$, $\tilde{Y}$, and $\tilde{Z}$ one pixel each at a time. The contemporary scanning of images $\tilde{X}$, $\tilde{Y}$, and $\tilde{Z}$ is justified by the fact that near a significant brightness discontinuity all three images may present a non-zero value which has not been thresholded by the non-linear filter $\mathcal{T}_t(\cdot)$. At the same time, all the pixels with similar values (where 0 is a special case) are grouped together in subsequent clusters.

The specific mapping between the 2D image array and the 1D data stream can be extended to the hierarchical set of images generated by n iterations of the process. If n hierarchical decompositions have been performed, the 2D $\to$ 1D mapping is based on the scanning of $S^{(n)}$,

Table 1. Effectiveness of the serial LZW compression of the output image obtained after 0 (original), 1,2,3,4, and 5 iterations of the *lossless* decomposition. The values presented refer to an average computed on a set of standard test images (512 × 512, 8 bit/pixel).

Number of iterations	Compression ratio (standard scanning)	Compression ratio (proposed scanning)
0 (original)	6·67 bit/pixel	–
1	3·34 bit/pixel	3.10 bit/pixel
2	2·49 bit/pixel	2·23 bit/pixel
3	2·23 bit/pixel	2·04 bit/pixel
4	2·15 bit/pixel	2·00 bit/pixel
5	2·14 bit/pixel	2·00 bit/pixel



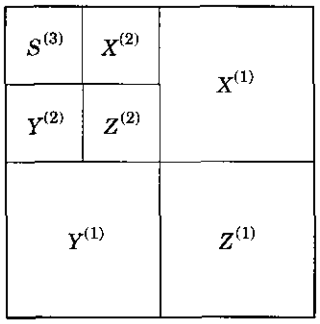


Figure 7. Lena image after two hierarchical iterations of the lossless deomposition (512 × 512). The lossless compression of this image with the LZW algorithm reaches about 2·26 bit/pixel

followed by the contemporary scannings of $\{\tilde{X}^{(i)}, \tilde{Y}^{(i)}, \tilde{Z}^{(i)}\}$ with $i = n, \ldots, 1$. The second column of Table 1 shows the effect of the compression when this new 2D → 1D mapping is used. As an example, Figure 9 shows the order of scanning in the case of two hierarchical iterations.

A further advantage of the hierarchical approach to the 2D → 1D mapping allows its application to the efficient transmission of images over different channels with different bandwidth. In fact, since the 1D data stream is formed by the concatenation of the sequence of the subsampled

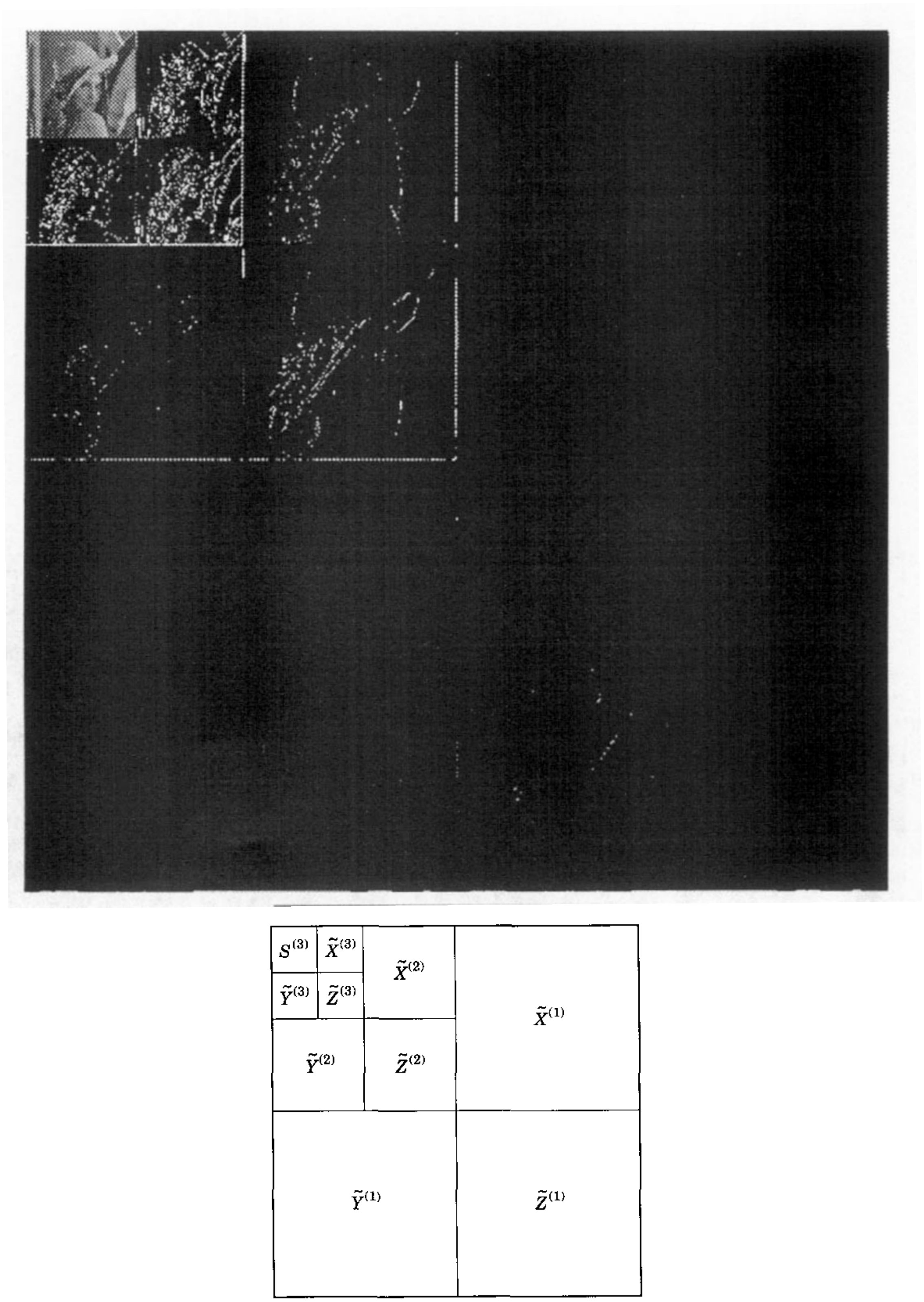


Figure 8. Lena image after three hierarchical iterations of the lossless deomposition followed by the non-linear filther with thresholds 60, 40 and 10 (512 × 512). The lossless compression of this image with the LZW algorithm reaches about 0·43 bit/pixel

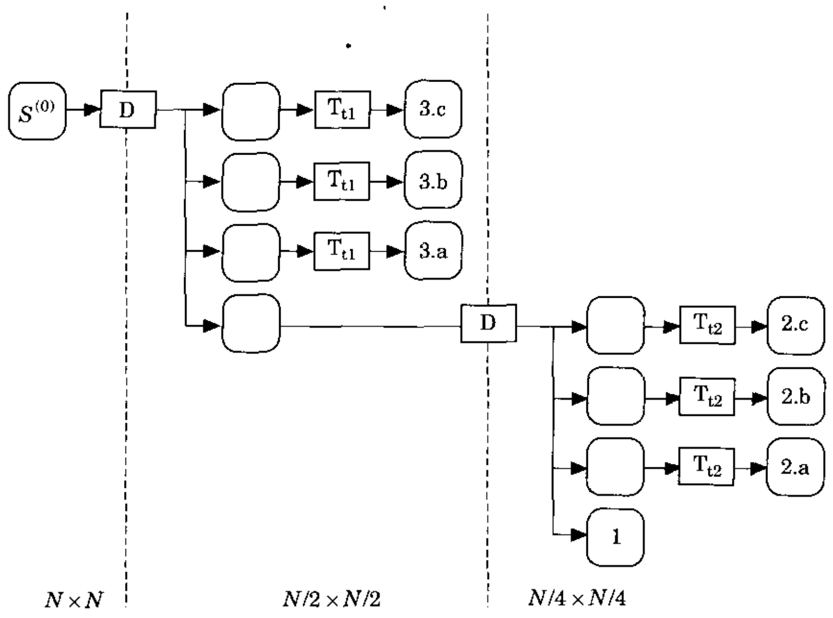


Figure 9. Mapping between the 2D image array and a 1D data stream: first the image labeled with '1' is scanned in a left-to-right and top-to-bottom fashion; then images labeled with '2.a', '2.b' and '2.c' are contemporary scanned, one pixel at a time; the same is then iterated on images '3.a', '3.b' and '3.c'

image $S^{(n)}$ and its local corrections $\tilde{X}^{(i)}$, $\tilde{Y}^{(i)}$, and $\tilde{Z}^{(i)}$, with $i = n, \ldots, 1$, this specific mapping:

(i) achieves a higher compression ratio, since similar values are grouped together, and
(ii) starts the reconstruction process before the transfer of the image has been completed.

The main advantage of (ii) lies in the possibility of recovering a low resolution image first, and increasing its detail with the integration of the information which follows. Thus, it offers the possibility of aborting the transfer if the coarse resolution image is considered not significant. Moreover, as in the case of the MPEG-2 for image sequence compression, it modifies the resolution of the recovered image (or sequence of images) dynamically according to the channel bandwidth and/or computational power available.

Table 2. Expected performances of the complete system

Operation	Time required [ms]
Acquisition time	≃ 20 ms
Data reorganization on PAPRICA	≃ 100 ms
Transfer time from PAPRICA system to its front-end processor	≃ 30 ms
Serial compression on PAPRICA front-end	in parallel with the acquisition and processing of the next frame

The Computing Architecture

The following section presents the target architecture which has been selected to run the algorithm discussed above: PAPRICA (PArallel PRocessor for Image Checking and Analysis) (8, 9), developed in collaboration with the Politecnico di Torino, Italy.

The PAPRICA system, based on a hierarchical mathematical morphology computational model, has been designed as a specialized co-processor to be attached to a general purpose traditional host (PAPRICA front-end): in the current implementation it is integrated on a single VME board (6U) connected to a SPARC-based workstation. It comprises five major functional parts:

(i) the Program Memory, storing up to 256k instructions;
(ii) the Image Memory (up to 8 MBytes of static RAM), whose 16-bit memory fetch time is 150 ns;
(iii) the Processor Array (PA), whose processors' cycle time is 250 ns;
(iv) the Frame Grabber device, able to grab 512 × 512 8 bit/pixel grey-tone images at video rate (25 frames/s);
(v) the Control Unit, which manages the activity of the whole system.

The first prototype of the PA is composed of an array of 4 × 4 full custom ICs (1·5 μm CMOS, 45 mm^2, ≃ 35000 transistors), each of them containing a sub-array of 4 × 4 PEs. In the present implementation, the PA is a 16 × 16

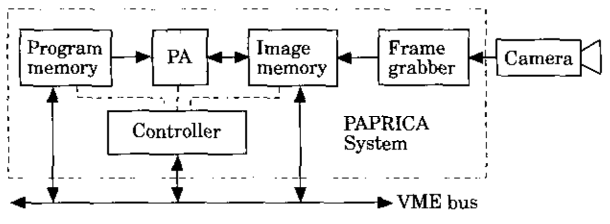


Figure 10. Block diagram of PAPRICA system

square matrix of 1-bit PEs each one with full 8-neighbors connectivity; each PE has an internal memory composed of 64 bits; data flow between the image memory and the PA through a 16-bit data bus. A block diagram of the complete system is presented in Figure 10.

Since the number of PEs is normally less than the number of image pixels used in generic low-level image processing applications, a significative part of the controller has been dedicated to the implementation of a processor virtualisation mechanism. Two virtualization mechanisms are generally implemented on massively parallel systems: the choice depends mainly on the amount of memory associated to each PE. In the first one (17–19) the computation is serialized within each processor, which contains several data. This mechanism cannot be implemented on all SIMD machines, due to its high memory requirements. Low-cost SIMD systems, such as PAPRICA, are characterized by a low amount of memory owned by each PE, and thus they implement a different virtualization mechanism. In fact, in the second case (20–23) the computation is serialized in windows: the PA is loaded with a sub-window of the data set, then the computation [or a segment of the computation (24)] is performed, and finally the result is stored back into the image memory, on a different image plane. These steps are iterated until all the sub-windows have been processed. PAPRICA control unit drives the sequential scanning of the image sub-windows. This virtualization mechanism is less efficient than the first one in terms of speed since it requires a lot of access to an external memory, but it offers a number of advantages, such as the possibility of handling hierarchical data structures on the 2D mesh with a trivial modification to the controller.

In fact, it is possible to take advantage of the fact that for each parallel computation $2Q^2$ sequential accesses to the image memory are required (where Q is the linear size of the PA). The data to be transferred into the PA can be non-logically adjacent to each other in the image memory, undersampling the image or increasing its resolution. This behavior is controlled by a set of registers which can be altered run-time. The possibility of reducing and increasing the image size allows the handling of hierarchical data structures.

Figure 11 shows an undersampling process during the loading of the data from Image Memory to the PA. After the processing, data are stored back again into the Image Memory in adjacent positions.

PAPRICA system has been designed to run real-time image processing tasks. It has already been efficiently utilized in applications for the processing of images acquired from a camera installed on a vehicle moving on rural roads, where the aim of the real-time analysis of images is the detection of the road boundaries and/or eventual obstacles (25–27).

System Performances

The first version of this parallel preprocessing filter has been implemented on a Connection Machine CM-2 [20] for testing and tuning purposes, while currently a non-optimized preliminary version runs on the target architecture PAPRICA.

The following section presents the performances obtained processing a 256×256 8 bit/pixel still image. The extremely simple set of operations that must be performed by each PE is listed in the following:

(i) determination of the average over the i, j, and ij directions in its 3×3 neighborhood;
(ii) comparison with a fixed threshold;
(iii) subsequent conditional reset of the values below the threshold.

All these operations must be iterated for each level of the hierarchical data set (pyramid). Contrary to general purpose parallel architectures (like the CM-2), pyramidal operations cause no computational overhead to PAPRICA system, as shown previously.

According to the theoretical analysis presented by Broggi (24), the PAPRICA processing of a $N \times N$ image with an algorithm based on the analysis of a 3×3 neighborhood [Moore Cellular Automaton (28, 29)] must be split in $\left(\left\lceil \frac{N-2}{Q-2} \right\rceil\right)^2$ subwindows, where Q represents the linear size of the PA.

Indicating with T_M the memory access time, and with T_C the PA clock cycle time, the time required to process a single image subwindow (24) is given by the sum of the

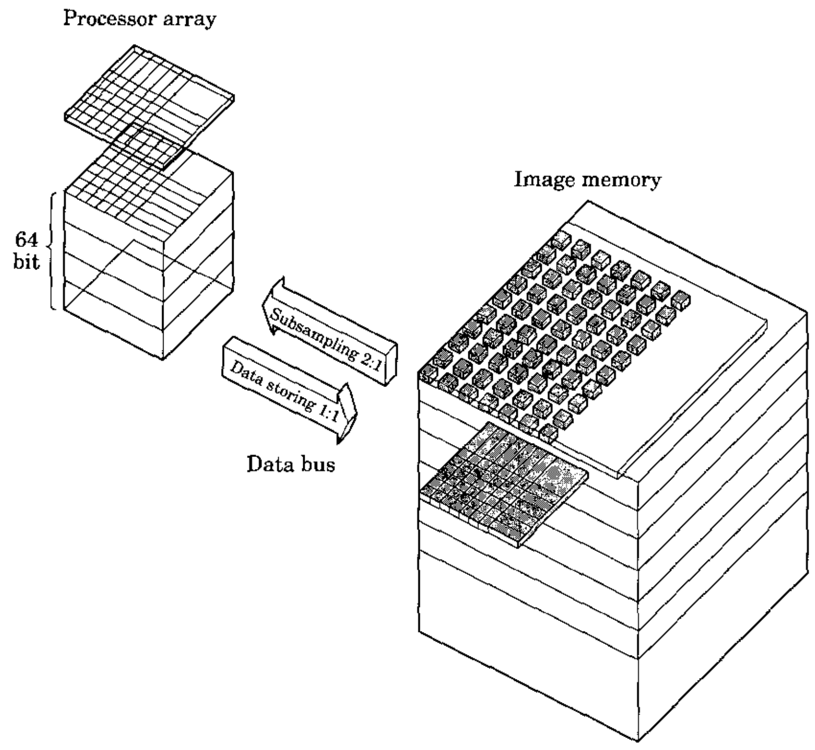


Figure 11. Hierarchical processing on PAPRICA system

loading time, processing time, and saving time. The loading and saving times can be considered equal and given by the number of pixels to be loaded ($Q \times Q$) times the memory access time.

$$T_{\text{loading}} = T_{\text{saving}} = Q^2 T_M = 16^2 \cdot 150 \text{ ns} = 38{,}400 \text{ ns} \quad [13]$$

The processing time is given by the number of assembly instructions executed by the PA (L, roughly 500) times the processors clock cycle time:

$$T_{\text{processing}} = LTc \simeq 500 \cdot 250 \text{ ns} = 125{,}000 \text{ ns} . \quad [14]$$

The total time required to process a single subwindow on the current version of the PAPRICA system is given by:

$$T_{\text{single window}} = T_{\text{loading}} + T_{\text{processing}} + T_{\text{saving}} \simeq 0{\cdot}2 \text{ ms} , \quad [15]$$

which, multiplied by the number of subwindows in which a 256 × 256 image is partitioned becomes:

$$T_{\text{total}} \simeq 0{\cdot}2 \cdot 361 \simeq 73 \text{ ms} . \quad [16]$$

As an example, four hierarchical iterations on a 256 × 256 image require the processing of 361 + 81 + 25 + 9 = 476 subwindows, and the total time becomes about 96 ms. The approximated values presented so far have been derived theoretically, but the current preliminary non-optimized version runs on PAPRICA system in about 110 ms. From the experience gained in the development of similar applications (26, 27), it follows that the optimization of the code can lead to the processing of more than 10 frames per second.

Finally, the time required for the whole acquisition, processing, and compression of a single still image is given by the following contributes:

(i) the acquisition time for a 256 × 256 image is 20 ms;
(ii) the parallel preprocessing takes about 100 ms;
(iii) the 2D → 1D mapping (basically the I/O time required to transfer the preprocessed image from PAPRICA .system to its front-end) takes about 30 ms;
(iv) the serial compression of the 1D data stream is performed by PAPRICA front-end in parallel with the acquisition and preprocessing of the following frame by PAPRICA system, and thus the serial compression time is not considered.

The expected processing speed achievable with the discussed system is evaluated in about 6-7 frames per second; these performances are summarized in Table 2. Moreover, if the computing architecture could be able to load images in the image memory in parallel with the processing (30),

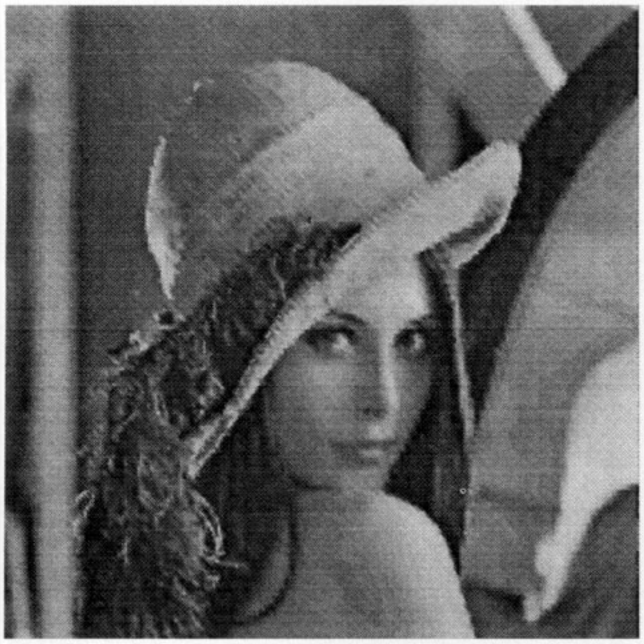


Figure 12. Lena image: 0·23 bit/pixel compression obtained with five hierarchical iterations of the discussed procedure (with thresholds 100, 60, 15, 6 and 0)

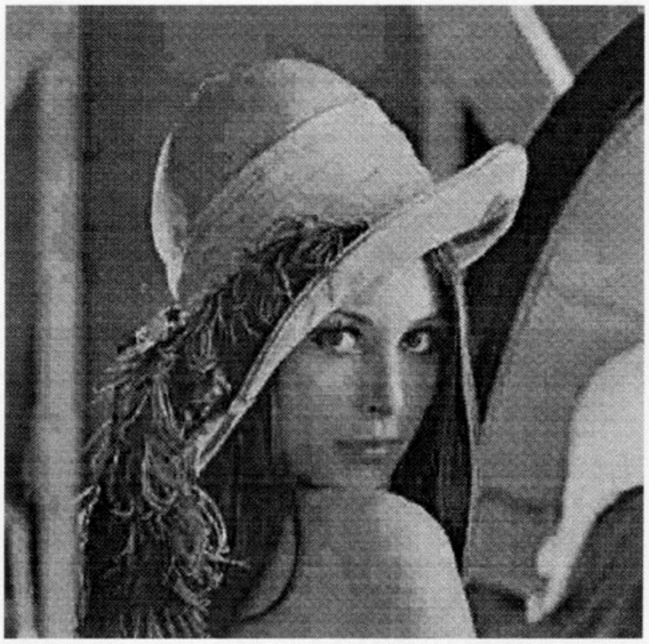


Figure 13. Lena image: 0·23 bit/pixel compression obtained with the JPEG algorithm: edges are preserved better than in the previous image, but an annoying blocky effect appears

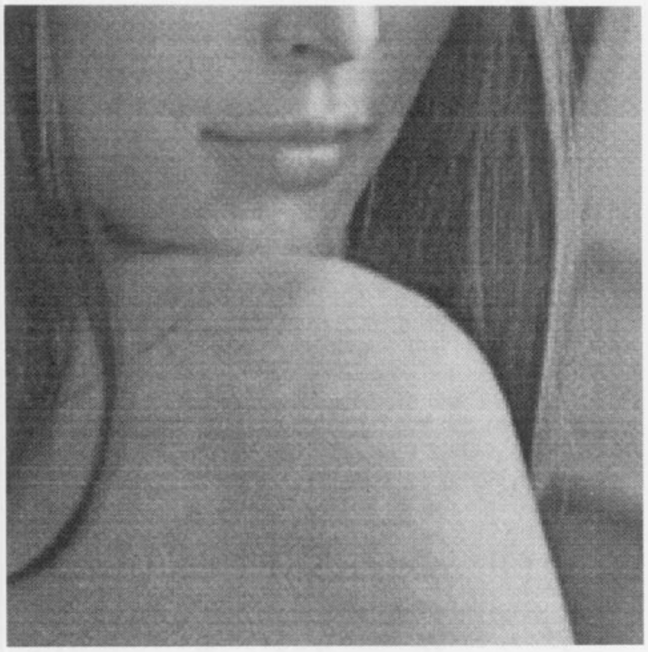


Figure 14. Part of the original Lena image

the loading time could be neglected, and the processing speed could be furthermore increased.

The output quality of the discussed procedure has been compared to the standard JPEG compression technique (at the same compression ratio), and although the former looks better than the latter only in few cases, the extremely fast computation achievable on low cost and low power systems justifies the consideration and the study of this new technique.

Figures 12 and 13 show the quality of a 0·23 bit/pixel compression of Figure 6 with the discussed procedure (five iterations with thresholds 100, 60, 15, 6, 0) and the standard JPEG technique respectively. Figure 14 shows an enlarged portion of the original image of Figure 6, while Figures 15 and 16 show an enlarged portion of Figures 12 and 13 respectively; the blocky effect, which represents the main artifact introduced by the JPEG compression (see Figure 16), is replaced by a me tolerable linear behavior (see Figure 15).

Conclusion

In this paper a novel massively parallel approach to real-time image compression has been presented. It has been designed to run a low-cost SIMD architecture featuring a much lower power consumption than dedicated hardware boards. This characteristic allows its porting to portable systems, where the power consumption is a critical design issue. Nevertheless, the computing architecture must be able to handle hierarchical data structures efficiently.

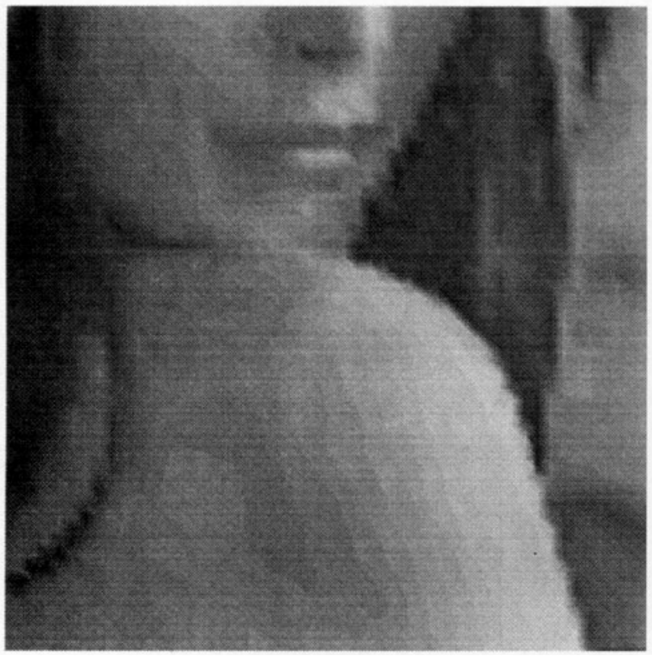


Figure 15. Part of the original Lena image: 0·23 bit/pixel obtained with the discussed procedure (five iterations with thresholds 100, 60, 15, 6 and 0): the linear behavior is visible

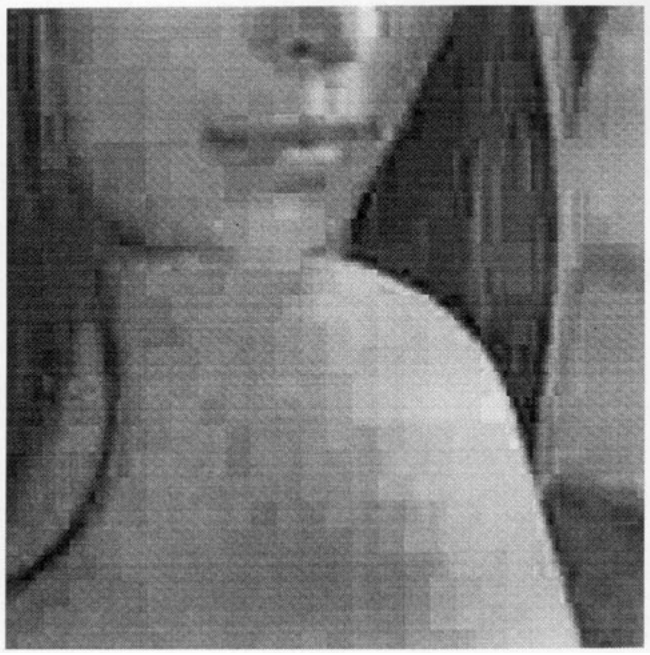


Figure 16. Part of the Lena Image: 0·23 bit/pixel obtained with the JPEG algorithm: the blocky effect is clearly visible

Moreover, due to the specific hierarchical decomposition and 2D $\longrightarrow$ 1D mapping, it is possible to adapt dynamically the image resolution to the channel bandwidth and/or computational power available.

The discussed algorithm has been implemented on PAPRICA system and it is currently under optimization, while an extension of the considered approach (based on similar interpolation methods) to the compression of image sequences is under evaluation.

Acknowledgments

The author would like to thank Monica Mordonini and Luigi Scaramuzza for their help in testing the algorithm on a Connection Machine CM-2 and porting it to PAPRICA system.

This work was partially supported by the Italian Research Council under contract number 9304759.ST74.

References

1. Chandrakasan, A., Sheng, S. & Brodersen, R. (1990) Design considerations for a future multimedia terminal. *Proc. WIN-LAB Workshop* – Rutgers Univ., New Brunswic.
2. Forman, G.H. & Zahorjan, J. (1994) The challenge of mobile computing. *Computer*, 27(4):38–47.
3. Shoji, M. (1988) *CMOS Digital Circuit Technology*. Prentice Hall.
4. Courtois, B. (1993) CAD and Testing of ICs and systems: Where are we going? Technical report, TIMA & CMP.
5. Chandrakasan, A., Sheng, S. & Brodersen, R. (1992) Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484.
6. Wallace, G.M. (1991) The JPEG still picture compression standard. *Communication of ACM*.
7. Welch, T.A. (1984) A technique for high performance data compression. *Computer*, 17(6):8–19.
8. Broggi, A., Conte, G., Gregoretti, F., Sansoè, C. & Reyneri. L.M. (1994) The PAPRICA Massively Parallel Processor. In *Proceedings MPCS – IEEE International Conference on Massively Parallel Computing Systems*, pages 16–30, Ischia, Italy. IEEE Computer Society – EuroMicro.
9. Gregoretti, F., Reyneri, L.M., Sansoè, C., Broggi, A. & Conte, G. (1993) The PAPRICA SIMD array: critical reviews and perspectives. In L. Dadda and B. Wah, editors, *Proceedings ASAP'93 – IEEE Computer Society International Conference on Application Specific Array Processors*, pages 309–320, Venezia, Italy. IEEE Computer Society – EuroMicro.
10. Rosenfeld, A. (1984) *Multiresolution Image Processing and Analysis*. Springer Verlag, Berlin.
11. Tanimoto, S.L. & Kilger, K. (1980) *Structured Computer Vision: Machine Perception through Hierarchical Computation Structures*. Academic Press, NY.
12. Serra, J. (1982) *Image Analysis and Mathematical Morphology*. Academic Press, London.

13. Haralick, R.M., Sternberg, S.R. & Zhuang, X. (1987) Image analysis using mathematical morphology. *IEEE Trans Patt Anal Mach Intell*, 9(4):532–550.
14. Blake, A. & Zisserman, A. (1987) *Visual Reconstruction*. MIT Press, Cambridge MA.
15. Geiger, D. & Girosi, F. (1991) Parallel and deterministic algorithms for Markov random fields: surface reconstruction and integration. *IEEE Trans Patt Anal Mach Intell*, **13**:401–412.
16. Mordonini, M. (1994) Compressione di immagini mediante tecniche massivamete parallele. Master's thesis, Università degli Studi di Parma – Facoltà di Ingegneria.
17. Hillis, W.D. (1985) *The Connection Machine*. MIT Press, Cambridge, Ma.
18. Duff, M. (1979) Parallel Processors for Digital Image Processing. In P. Stucki, editor, *Advances in Digital Image Processing*, pages 265–279. Plenum Press, London.
19. Fountain, T. & Matthews, K. (1988) The CLIP 7A image processor. *IEEE Trans Patt Anal Mach Intell*, **10**(3):310–319.
20. Schmitt, L.A. & Wilson, S.S. (1988) The AIS–5000 Parallel Processor. *IEEE Trans Patt Anal Mach Intell*, **10**(3)320–330.
21. NCR Corporation, Dayton, Ohio. *Geometric Arithmetic Parallel Processor*, 1984.
22. Reddaway, S. (1973) DAP – A Distributed Array Processor. In *1st Annual Symposium on Computer Architectures*, pages 61–65, Florida.
23. Conte, G., Gregoretti, F., Reyneri, L. & Sansoè, C. (1991) PAPRICA: Parallel Architecture for VLSI CAD. In A. P. Ambler, P. Agrawal, and W. R. Moore, editors, *CAD Accelerators*, pages 177–189. North Holland, Amsterdam.
24. Broggi, A. (1994) Performance Optimization on Low-Cost Cellular Array Processors. In *Proceedings MPCS – IEEE International Conference on Massively Parallel Computing Systems*, pages 334–338, Ischia, Italy. IEEE Computer Society – EuroMicro.
25. Adorni, G., Broggi, A.,. Conte, G & D'Andrea, V. (1995) Real-Time Image Processing for Automotive Applications. In P. Laplante, and A. D. Stojuko, editor, *Real-Time Image Processing: Theory, Techniques and Applications*. IEEE Press and SPIE Press. In press.
26. Broggi, A. (1995) Parallel and Local Feature Extraction: a Real-Time Approach to Road Boundary Detection. *IEEE Trans Imag Process*. In press.
27. Broggi, A. & Bertè, S. (1994) A Morphological Model-Driven Approach to Real-Time Road Boundary Detection for Vision-Based Automotive Systems. In *Proceedings Second IEEE Workshop on Applications of Computer Vision*, pages 73–90, Sarasota, FL. IEEE Computer Society.
28. Culik II, K., Hurd, L.P. & Yu, S. (1990) Computation theorethic aspects of Cellular Automata. *Physica D*, 45:431–440.
29. Farmer, D. Toffoli, T. & Wolfram, S. eds (1983) *Cellular Automata*, Amsterdam. North-Holland.
30. Cucchiara, R. (1993) *Parallelismo Spaziale e Temporale nelle Architetture per l'Elaborazione di Immagini*. PhD thesis, DEIS, Università degli Studi di Bologna, Bologna, Italy.