# Generalized PCM coding of images

José Prades-Nebot*, *Member, IEEE,* Marleen Morbee, and Edward J. Delp, *Fellow, IEEE*

***Abstract*—Pulse-code modulation (PCM) with embedded quantization allows the rate of the PCM bitstream to be reduced by simply removing a fixed number of least significant bits from each codeword. Although this source coding technique is extremely simple, it has a poor coding efficiency. In this paper, we present a generalized PCM (GPCM) algorithm for images that simply removes bits from each codeword. In contrast to PCM, however, the number and the specific bits that a GPCM encoder removes in each codeword depends on its position in the bitstream and the statistics of the image. Since GPCM allows the encoding to be performed with different degrees of computational complexity, it can adapt to the computational resources that are available in each application. Experimental results show that GPCM outperforms PCM with a gain that depends on the rate, the computational complexity of the encoding, and the degree of inter-pixel correlation of the image.**

**EDICS**. COM-LOC: Lossy coding of images and video.

## I. Introduction

Today, most signals of interest (e.g., voice, audio, image, video) are digitally acquired (*digitized*) using A/D converters. A/D converters perform pulse-code modulation (PCM) with uniform quantization and fixed-length binary coding. This type of coding has several advantages. First, PCM is the simplest source coding technique [1]. Second, it is trivial to locate (and decode) any codeword in the PCM bitstream (*random access*). Third, when embedded quantization is used, the rate of the PCM bitstream can be easily reduced by discarding a fixed number of bits in each codeword (*scalability*). Despite all these advantages, PCM is not usually used in the storing or transmission of signals due to its poor coding efficiency. For this reason, the PCM bitstream is usually compressed using a sophisticated and efficient coding algorithm [1].

Apart from coding efficiency, encoding simplicity is also an important factor in applications with very limiting constraints on computational power and/or energy [2]. A simple encoder is also necessary when the signal must be acquired at a very high sampling rate. This happens, for instance, in applications that require capturing video at very high frame rates [3]–[6].[1] These applications, require source coding algorithms that are extremely simple and, consequently, many times the PCM signal (*raw video*) is directly transferred and stored. However, the high bit rate of raw video can make its transmission or storage difficult. Thus, the bus may not be fast enough to transfer the video out of the sensor or the writing speed of the storage device may not be high enough to save the raw video [5]. In all these applications, a lossy coding algorithm that has properties that are similar to those of PCM but with a better coding efficiency would be of interest.[2] These constraints discard most of the image coding algorithms (e.g., JPEG) since their computational complexity is too high for very high-speed video applications (even low-complexity DPCM coders may be too complex for applications of this type [7], [8]). Moreover, most image coding algorithms do not preserve the random access property and are sensitive to transmission or storage errors (especially DPCM coders). The simplest solution to reduce the bit rate of raw video is to decrease the temporal resolution and/or the spatial resolution of the signal provided by the camera sensor.

In this paper, we present a generalized-PCM (GPCM) image coding algorithm for applications that require an extremely low computational complexity. The GPCM encoding is divided into two stages: the *analysis* and the *encoding* itself. The encoding is done by simply discarding bits of each pixel value (as PCM with embedded quantization does).[3] The type (least significant bits (LSBs) or most significant bits (MSBs)) and number of bits discarded in each pixel are determined by using the information provided by the analysis stage. Although the analysis stage involves some numerical processing, its complexity can be reduced by using statistical sampling [10], [11]. Moreover, GPCM can operate at different degrees of computational complexity which allows it to adapt to the computational resources that are available in each application. Although GPCM has a worse coding performance than most image coding algorithms, it is extremely simple, it facilitates random access to the data, it is robust to transmission errors, it is scalable in complexity, and it outperforms PCM in coding efficiency. These features make the algorithm useful for the recording of video at extremely high frame rates.

The rest of the paper is organized as follows. In Section II, we describe the GPCM coding of images. In Section III, we experimentally test the coding efficiency of GPCM and compare it to other image coding algorithms. Finally, in Section IV, we summarize our results.

José Prades-Nebot is with the Institute of Telecommunications and Multimedia Applications, at the Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain. Phone: +34 96 330 5821. FAX: +34 96 387 7309. E-mail: jprades@dcom.upv.es. Marleen Morbee is with the Department of Telecommunications and Information Processing, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium. Gent, Belgium. Phone: +32 9 264 4225. Fax: + 32 9 264 4295. E-mail: marleen.morbee@telin.ugent.be. Edward J. Delp is with the School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Avenue, West Lafayette, IN 47907-2035 USA. Phone: +1 765 494 1740. FAX: +1 765 494 3358. E-mail: ace@ecn.purdue.edu.

[1] In some ultra high-speed applications, the frame rate is so high that not even PCM can be used and only a few *analog* frames are captured and stored [6].

[2] We are considering applications that permit a lossy compression of the video signal.

[3] In preliminary versions of this work (i.e., [9], [10]), our algorithm was named Modulo-PCM; however, in this paper, we use the name GPCM in order to prevent confusion with the Modulo-PCM technique. In fact, our algorithm combines three coding techniques (one of which is very similar to Modulo-PCM).

## II. GPCM CODING OF IMAGES

Let us consider a monochromatic digital image encoded with PCM at $R_0$ bits per pixel (bpp). Let $\Delta_0$ be the quantization step-size of the digital image. Let us also assume that uniform embedded quantization is used, i.e., the removal of the $l$ LSBs of each codeword of the image provides the same bitstream as the PCM encoding of the analog image with $R_0-l$ bpp. Our GPCM algorithm divides the input image into blocks of $b\times b$ pixels and encodes each block at the target rate $R$ using appropriate parameter values. In Section II-A, we describe the GPCM encoding and decoding of each image block. In Section II-B, we describe the different coding techniques used in GPCM. In Section II-C, we show how the GPCM encoder assigns values to the coding parameters of each block. Section II-D is devoted to evaluating the computational complexity of GPCM. Finally, in Section II-E, we show how GPCM can achieve different degrees of coding complexity and efficiency.

### A. GPCM coding

Let $x[n_1, n_2]$ be a block of $b \times b$ pixels of the input image. For the sake of brevity, we will drop variables $n_1$ and $n_2$ when this does not jeopardize clarity. The GPCM encoder first divides $x$ into four decimated subblocks (Figure 1):

$$\begin{aligned} x_0[n_1,n_2] &= x[2n_1,2n_2], \\ x_1[n_1,n_2] &= x[2n_1+1,2n_2], \\ x_2[n_1,n_2] &= x[2n_1,2n_2+1], \\ x_3[n_1,n_2] &= x[2n_1+1,2n_2+1]. \end{aligned}$$

We refer to $x_0$ as the *PCM subblock*, and we refer to $x_1$, $x_2$, and $x_3$ as the *GPCM subblocks*. In each subblock $x_k$, the encoder simply removes the $l_k$ LSBs and the $m_k$ MSBs from each pixel value. Parameter $m_0$ is always set to zero. To simplify the assignment of values to the coding parameters, we set $l_1 = l_2 = l_3$ and $m_1 = m_2 = m_3$. In this way, the encoder only has to assign values to three parameters: $l_0$, $l_1$, and $m_1$. The resulting codewords $\tilde{x}_k$ are transmitted to the decoder (Figure 1). Each codeword of $\tilde{x}_0$ represents a quantization interval of length $2^{l_0}\Delta_0$. Similarly, each codeword of $\bar{x}_k$ ($k \in \{1,2,3\}$) represents an interval of size $2^{l_k}\Delta_0$. If $m_k > 0$, the encoder performs *binning* over the codewords of $\bar{x}_k$ [12], i.e., each codeword of $\tilde{x}_k$ represents a *bin* of $2^{m_k}$ codewords $\bar{x}_k$.

At the decoder, $x_0$ is first decoded using midpoint reconstruction (i.e., using the midpoint of each quantization interval). Then, the decoder obtains an approximation of $x_1$, $x_2$, and $x_3$ by performing interpolation over the reconstructed PCM subblock $\hat{x}_0$. *Bilinear interpolation* is used since it represents a good trade-off between complexity and interpolation accuracy [13].[4] The interpolated subblocks $y_1$, $y_2$, and $y_3$ act as side information (SI) for the decoding of $x_1$, $x_2$, and $x_3$, respectively (Figure 1).

[4]To interpolate the pixels of the GPCM subblocks that are placed in the last column and row of a block $x$, the first column of PCM subblock that is placed to the right of $x$ and the first row of the PCM subblock that is placed below $x$ must be previously decoded.

The decoding of each codeword of $\tilde{x}_k$ ($k \in \{1,2,3\}$) is divided into two steps: *decision* and *reconstruction*. In the decision step, the decoder selects one of the $2^{m_1}$ quantization intervals represented by the codeword (no decision is performed when $m_1 = 0$). Among all the potential intervals of the codeword, the decoder selects the one that is closest to its SI. If the decision is correct, the $m_1$ MSBs that were removed by the encoder are correctly recovered. Otherwise, the decoder incurs a *decision error* which may generate a large-amplitude decoding error. After the decision, each codeword represents a single interval.

In the reconstruction step, the aim is to recover the $l_1$ LSBs that were removed from each pixel codeword by the encoder. The decoder first estimates the value of each pixel using its SI and its quantization interval. If we assume that the interpolation error $e$ between a pixel value $x$ and its SI $y$ ($e \triangleq y - x$) follows a unimodal and symmetric probability distribution, then the maximum a posteriori (MAP) estimation of $x$ when $x$ belongs to a quantization interval $[a, b]$ is given by the clipping function

$$\check{x} = \begin{cases} a, & y < a \\ y, & a \le y \le b \\ b, & y > b \end{cases}. \tag{1}$$

A better but more complex reconstruction can be performed if minimum mean squared error (MMSE) estimation is used [9]. When $m_1 = 0$ and $l_0 = l_1$, midpoint reconstruction instead of MAP reconstruction is used since, in this case, the SI does not offer any significant help for the decoding of the GPCM subblocks. Each estimated value $\check{x}$ is then quantized with $R_0$ bits. Finally, the four decoded subblocks are multiplexed (Figure 1). Since all the blocks are encoded at the same rate and bilinear interpolation only operates over a small number of pixel values (2 or 4), random access is facilitated (i.e., locating the codeword of any pixel and decoding it is simple).

### B. Coding techniques

GPCM combines three simple techniques: PCM, interpolative coding, and binning. GPCM behaves like PCM when $m_1 = 0$ and $l_0 = l_1$. PCM is optimum at high rates or for blocks with very poor spatial correlation (i.e., when no benefit is obtained from exploiting the similarity among pixels). When $m_1 = 0$ and $l_0 < l_1$, a GPCM encoder behaves like a PCM encoder that uses two quantization step-sizes. At the decoder, the pixels of $\hat{x}_0$ are interpolated to generate SI, which helps in the reconstruction of the three GPCM blocks. This technique can outperform conventional PCM at low rates.

When $l_1 + m_1 = R_0$, interpolative coding is used [14]: the GPCM encoder only encodes and transmits $x_0$, which is reconstructed and interpolated at the decoder. This technique provides good results at low rates since, at these rates, only transmitting $x_0$ and interpolating $\hat{x}_0$ provides better results than spending the available bits in the encoding of the four subblocks.

When $m_1 > 0$ and $l_1+m_1 < R_0$, binning is performed over the codewords of the GPCM subblocks. The use of binning is useful in blocks with a high spatial correlation since most

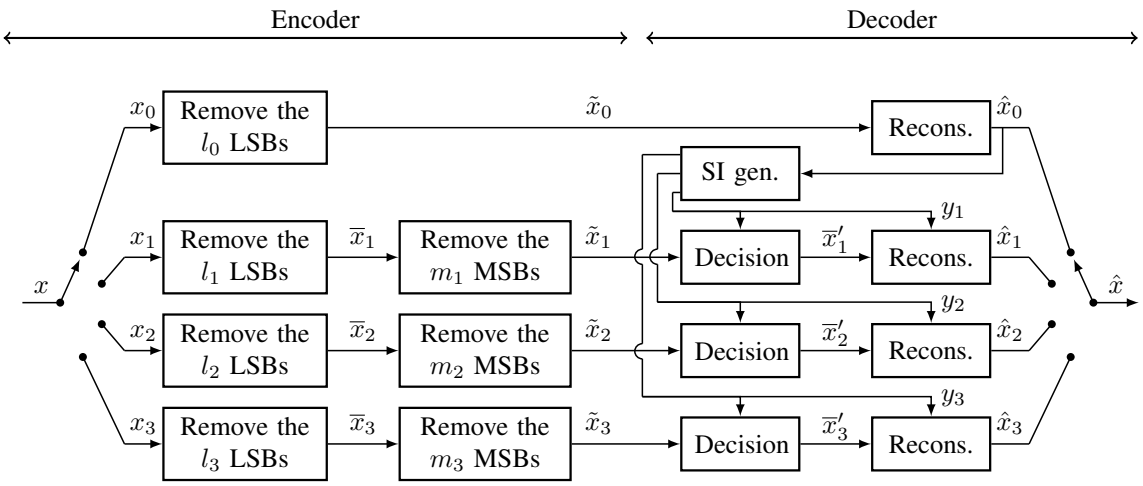



Fig. 1. Block diagram of the GPCM coding of an image block.

of the removed MSBs bits can be correctly recovered by the decoder by exploiting the similarity among the pixels.

### C. Assignment of values to the coding parameters

The encoder has to assign values to the parameters $l_0$, $l_1$, and $m_1$ of each block. For a target rate $R$, the optimum $(l_0, l_1, m_1)$ is the one that minimizes the distortion $D$ and fulfills the rate constraints:

$$\begin{aligned} &\text{minimize} \quad D(l_0, l_1, m_1) \\ &\text{subject to} \quad R = R_0 - \frac{1}{4}\left[l_0 + 3(l_1 + m_1)\right] \\ &\qquad\qquad 0 \le l_0, l_1, m_1 \le R_0 \\ &\qquad\qquad 0 \le l_1 + m_1 \le R_0 \end{aligned} \tag{2}$$

In [15], we developed a model for $D$ when the interpolation error follows a Laplacian distribution. Hence, by solving (2) using the distortion model in [15], a GPCM encoder could obtain the optimum assignment for each block. Nevertheless, since solving (2) is a complex task and encoding complexity is our main concern, the encoder uses a series of assignment rules (ARs) to perform the assignment. Thus, for each possible $R$ and $b$, there is an AR that provides an appropriate triplet $(l_0, l_1, m_1)$ for each block as a function of the peak signal-to-noise ratio of its SI ($\mathrm{PSNR_{SI}}$) defined as

$$\mathrm{PSNR_{SI}} = 10 \log \frac{\left(2^{R_0} - 1\right)^2}{\mathrm{MSE_{SI}}} \quad (\mathrm{dB}) \tag{3}$$

where $\mathrm{MSE_{SI}}$ is the mean squared *interpolation error* of the block when $l_0 = 0$ (i.e., $\mathrm{MSE_{SI}}$ can be computed from pixels of the original image). We experimentally obtained the ARs for a set of values of $b$ ($b \in \{4, 8, 16, 32, 64, 128, 256, 512\}$) and $R$ ($R \in \{1, 2, 3, 4, 5, 6\}$) using 50 gray-scale digital images. To obtain the AR for a pair of $b$ and $R$ values, we classified each consecutive block of $b \times b$ pixels of the 50 images in accordance with its $\mathrm{PSNR_{SI}}$ value when $l_0 = 0$. In this classification, intervals with a width of 1 dB were used. Then, we encoded the blocks with all the possible triplets $(l_0, l_1, m_1)$ that corresponded to each target rate $R$. Finally, we selected the $(l_0, l_1, m_1)$ that provided the minimum mean squared decoding error of the blocks in each $\mathrm{PSNR_{SI}}$ interval.[5]

### D. Computational complexity

The GPCM encoding of a block essentially involves two processes: the computation of its $\mathrm{MSE_{SI}}$ (the $\mathrm{PSNR_{SI}}$ can be obtained from the $\mathrm{MSE_{SI}}$ through a look-up table) and the encoding itself. Computing the $\mathrm{MSE_{SI}}$ of a block of $b \times b$ pixels involves: the bilinear interpolation of $x_0$ ($5b^2/4$ additions and $3b^2/4$ shifts), the computation of the interpolation errors ($3b^2/4$ subtractions), the squaring of the errors ($3b^2/4$ multiplications), and the summation of all the errors ($3b^2/4$ additions).[6] Therefore, on average, $4.25$ integer operations per pixel (iopp) are necessary to compute the $\mathrm{MSE_{SI}}$. The complexity of this computation can be reduced by computing the $\mathrm{MSE_{SI}}$ using only one out of $f$ pixels that must be interpolated. In this way, $\mathrm{MSE_{SI}}$ is computed with $4.25/f$ iopp. The higher the *sampling factor* $f$, the lower the number of operations to compute but the less accurate the assignment is (since $\mathrm{MSE_{SI}}$ is less accurate). Nevertheless, for a given block size, there is a maximum factor $f_{\max}$ such that assignments that are different to the optimum one rarely occur if $f \le f_{\max}$[7]. Hence, if $f = f_{\max}$, the complexity is reduced without incurring any appreciable loss in coding efficiency.

The encoding of each pixel of a PCM subblock requires zero shifts when $l_0 = 0$ and one shift when $l_0 > 0$. The encoding of each pixel in a GPCM subblock requires: zero shifts when $l_1 + m_1 = 8$, one shift when $l_1 > 0$ and $m_1 = 0$, and two shifts in the rest of the cases.

Let us assume that the pixel values are accessed line by line in raster order and that $w$ is the width of the image in pixels. Then, the encoding has a maximum latency of $b(w+1)$

[5]The ARs can be found on http://personales.upv.es/jprades/publications.html.

[6]The multiplications and divisions by integer powers of two and the removal of $m_1$ MSBs and $l_1$ LSBs can be implemented through shift operations.

[7]Since the $\mathrm{MSE_{SI}}$ is the sample variance of a finite population, the error in estimating $\mathrm{MSE_{SI}}$ depends on both the sampling factor and the size of the population (i.e., the size of the image). To achieve a target error variance in estimating the $\mathrm{MSE_{SI}}$, the larger the size of the population, the larger the sampling factor to use [16].

| Process | Maximum average number of integer operations per pixel | | | | | |
|---|---|---|---|---|---|---|
| | Additions | Subtractions | Multiplications | Shifts | Comparisons | Total |
| Encoding | $\frac{2}{f}$ | $\frac{3}{4f}$ | $\frac{3}{4f}$ | $\frac{3}{4f}+\frac{7}{4}$ | 0 | $\frac{17}{4f}+\frac{7}{4}$ |
| Decoding | $\frac{9}{4}$ | 0 | 0 | $\frac{10}{4}$ | $\frac{12}{4}$ | $\frac{31}{4}$ |

TABLE I
MAXIMUM NUMBER OF OPERATIONS PER PIXEL REQUIRED IN THE GPCM ENCODING AND DECODING OF AN IMAGE.

pixels and requires at most $b(w+1)+1$ memory cells. When $b$ is large, the memory usage and the encoding latency can be drastically reduced by encoding each block of a frame using the coding parameters that were computed for its colocated block in the previous frame[8].

The decoding of a block has three stages: the reconstruction of the PCM subblock, its bilinear interpolation, and the decoding of the three GPCM subblocks. The decoding of $\hat{x}_0$ involves multiplying each codeword by $\Delta_0$ and adding $2^{l_0-1}\Delta_0$ to the result (i.e., $b^2/4$ shifts and $b^2/4$ additions). The bilinear interpolation of $\hat{x}_0$ requires $5b^2/4$ additions and $3b^2/4$ shifts. The decoding of each pixel of the GPCM subblocks has two steps: the decision (only if $m_1 > 0$) and the reconstruction. The decision is equivalent to a uniform quantization which involves: two shifts, one addition, and two comparisons (to clip the result of the division) per pixel of the GPCM images. Finally, the MAP reconstruction of each pixel involves two comparisons. Table I shows the *maximum* average number of operations per pixel that is necessary to encode and decode an image. The maximum decoding latency is $2w$ pixels and $2w+1$ memory cells are required at most.

### *E. Complexity scalability*

The smaller the block size $b$, the higher the adaptation of GPCM to the local statistics of the image, and, consequently, the higher its coding efficiency. However, the smaller $b$, the smaller the maximum value of the sampling factor ($f_{\max}$) that can be used in the computation of $\mathrm{MSE_{SI}}$, and, hence, the higher the computational complexity of the encoding. As a consequence, the block size $b$ trades off coding efficiency and computational complexity. This scalability in complexity allows our algorithm to adapt to the computational resources that are available in each application. In the following, we describe how this adaptation can be made (Algorithm 1).

First, the value of the sampling factor $f$ is determined by the computational complexity that is attainable by the application. For each value of $f$, there is a minimum block size $b_{\min}$ such that the assignments that are different to the optimum ones rarely occur if $b \geq b_{\min}$, and, hence, using this $b_{\min}$ provides both the highest spatial adaptation and accurate assignments in most cases. Appropriate values $b$ for each value of $f$ are stored in a *block-size table*, which is used by the GPCM encoder to determine the value of $b$. To build a block-size table for our GPCM encoder, we experimentally encoded 50 images at different rates ($R \in \{1,2,3,4,5\}$) with different values of $b$ and $f$. Then, for each value of $f$, we selected the minimum value of $b$ that provides good encoding results (i.e., the minimum value of $b$ such that the loss in quality with this $f$ value is negligible with respect to $f=1$ in most of the encodings). These values are shown in Table II.

| $f$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 256 |
|---|---|---|---|---|---|---|---|---|
| $b$ | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |

TABLE II
BLOCK-SIZE $b$ FOR DIFFERENT VALUES OF $f$.

Once the values of $f$ and $b$ are known, the algorithm selects an AR according to the value of $R$ and $b$. Finally, for each block in the image, the algorithm first computes its $\mathrm{PSNR_{SI}}$, obtains the optimum $(l_0, l_1, m_1)$ from the AR, and encodes the block using the selected parameter values.

**Algorithm 1** GPCM coding

1: Select $f$ depending on the attainable complexity
2: Select $b$ depending on $f$ using the block-size table
3: Select an assignment rule depending on $R$ and $b$
4: **for** "each block of $b \times b$ pixels" **do**
5: Compute the $\mathrm{PSNR_{SI}}$ using a sampling factor $f$
6: Obtain $l_0$, $l_1$, and $m_1$ from the assignment rule
7: Encode the block
8: **end for**

## III. EXPERIMENTAL RESULTS

In this section, we experimentally analyze the performance of GPCM and compare it to other image coding algorithms. We encoded six gray-scale images of 512×512 pixels and 8 bpp. Each image was encoded with GPCM using six different values of $f$ (1, 2, 4, 8, 16, and 256) and their corresponding block sizes (4, 8, 16, 32, 64, and 512, respectively). Figure 2 shows the PSNR of the six images as a function of the rate $R$ and $b$. In Figure 2, the images are ordered according to their degree of spatial correlation (*Zelda* is the most correlated

[8]If $b$ is large, most of the texture content of two colocated blocks will not vary significantly from frame to frame. A similar approach is used in real-time rate control of video coders where some parameters computed in a frame of a certain type (I-, P- or B- frame) are used to control the rate in the following frame of the same type [17], [18].

image and *Baboon* is the least correlated one). The results of encoding each image using PCM, JPEG, and JPEG-LS (near-lossless mode) [19] are also shown for comparison.

Note in Figure 2 that the higher the computational complexity (or equivalently, the smaller $f$ or $b$), the higher the coding efficiency of GPCM. Thus, the best coding efficiency is obtained when $b = 4$ and $f = 1$, while the worst coding efficiency is obtained when $b = 512$ and $f = 256$ (i.e., each image is encoded using a single block). At 1 bpp and 2 bpp, the use of small values of $b$ does not generally provide significant improvements. The reason is that most blocks are encoded using the same assignment at these rates. Thus, at 1 bpp, most of the blocks are encoded using the assignment (4,8,0) (only those blocks that have a very small $\mathrm{PSNR_{SI}}$ are encoded using the assignment (7,7,0)). In the rest of the rates, decreasing $b$ improves the coding efficiency significantly (the greater the degree of spatial correlation, the greater the improvement).

Decision errors may occur in those pixels that belong to edges or textured regions (i.e., when the interpolation error may be large). Optimal parameter assignments provide decodings with a very small number of decision errors. Thus, the maximum percentage of decision errors among *all* the encodings of Figure 2 was 0.42% (*Peppers* at 3 bpp with $b = 4$). Sometimes, an inaccurate assignment provides a value of $m_1$ that is large. As a consequence, a large number of decision errors occur in the decoding, which can significantly increase the distortion (e.g., see *Zelda* encoded at 6 bpp with $b = 4$ in Figure 3). Nevertheless, the visibility of the distortion in those pixels that are affected by decision errors is reduced thanks to the limited sensitivity of the human visual system to amplitude errors in edges and high-activity regions.

The gain in coding efficiency of GPCM with respect to PCM is large at 1 bpp and 2 bpp and generally decreases with $R$. When $b = 512$, this gain is zero above a certain rate (GPCM behaves like PCM) except for very correlated images such as *Zelda*. Figure 3 shows a portion of the *Barbara* image encoded at 2 and 3 bpp using PCM, and at 2.003 and 3.003 bpp using GPCM with $b = 32$. In the images encoded with PCM, luminance is poorly represented and there is *false contouring* [20], which is even more visible at 1 bpp. In the images encoded using GPCM, the degradation comes mostly from the under-sampling and interpolation performed which reduces the edge sharpness and introduces stair-case effect in slanted edges (see for instance the back of the chair and the right part of the scarf in image (b)). False contouring is also visible in the GPCM decoded images in those blocks that are encoded using PCM (see for instance the left part of the chin in images (b) and (d)).

Predictive and transform coding algorithms perform better than GPCM at the expense of a higher computational complexity. Transform-based algorithms such as JPEG and JPEG2000 perform much better than GPCM at all rates (see Figure 2). The high coding efficiency of these algorithms comes at the expense of using two-dimensional transforms, quantization, and lossless coding techniques (e.g., Huffman or arithmetic coding). In particular, only the DCT stage of JPEG requires more operations than the whole GPCM encoding for any value of $b$.[9] Hence, although today's hardware allows the use of transform coders in many applications, they are not appropriate when an extremely low complexity coding is required [7], [8].

JPEG-LS in *near-lossless mode* is less complex than other image coding algorithms (e.g., JPEG, JPEG-2000, CALIC) and allows a controlled maximum coding error ($\delta$) [19]. As shown in Figure 2, JPEG-LS performs better than JPEG at high rates and much better than GPCM at all rates (the gap between JPEG-LS and GPCM increases as the rate increases). However, JPEG-LS is also much more complex than GPCM. In particular, in order to decide the encoding mode (*regular* or *run*) for each pixel, the JPEG-LS encoder must compute three gradients $g_i$ ($i = 1, 2, 3$) and check that $|g_i| \leq \delta$ (i.e., in each pixel, three subtractions, three absolute values, and three comparisons must be computed) [19]. Therefore, just the decision of the coding mode in JPEG-LS requires more operations than the entire GPCM encoding even when $b = 4$.

Recently, very simple DPCM-based image coders have been proposed for applications with high constraints in energy and/or computational resources [7], [8]. These algorithms perform better than PCM and GPCM thanks to the use of prediction and entropy coding. These features, however, make the compressed bitstream highly sensitive to transmission/storage errors and complicate random access. Moreover, even the simplest DPCM encoders (i.e., those that use low-order prediction with fixed coefficients, scalar uniform quantization, and simple entropy coding) require computing more operations than the GPCM encoder even when $b = 4$. Additionally, these DPCM coders are not scalable in complexity and, in some cases, part of their simplicity is due to the specific properties of the video of the targeted applications [7], [8]. One of the advantages of DPCM image coders (and JPEG-LS) is their small memory requirements and coding latency. When $b$ is large, GPCM encoders can obtain these features by using in each block the coding parameters of the colocated block in the previous frame (see Section II-D).

## IV. Conclusion

We have presented a GPCM coding algorithm for images which, after properly assigning values to the coding parameters, simply discards bits from each codeword of the PCM bitstream. Our algorithm is very simple computationally, is robust to transmission or storage errors, facilitates random access, and is scalable in complexity. These features make GPCM coding useful in those applications that need to reduce the bitrate of raw video in an extremely simple way. Experimental results show that GPCM provides improvements with respect to PCM. The magnitude of these improvements depends on the rate (the smaller the rate, the larger the gain), the encoding complexity (the greater the complexity, the greater the gain), and the degree of inter-pixel correlation (the greater the correlation, the greater the gain). Standard coding algorithms such as JPEG and JPEG-LS perform much better than GPCM at the expense of a significantly larger computational complexity.

[9] The Feig-Winograd fast-DCT algorithm requires computing 462 additions, 54 products, and 6 divisions in each $8\times8$ block, which involves 8.15 iopp on average [21]. However, the GPCM encoder requires 6.0 iopp at most when $b = 4$.

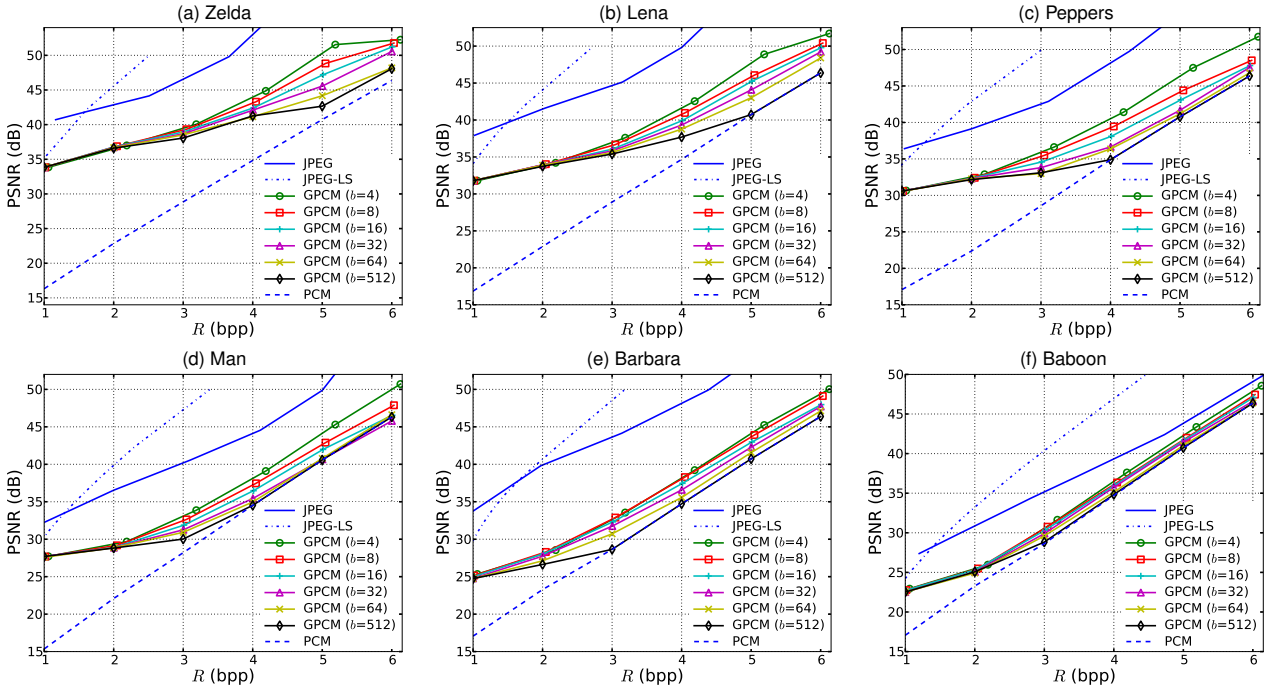
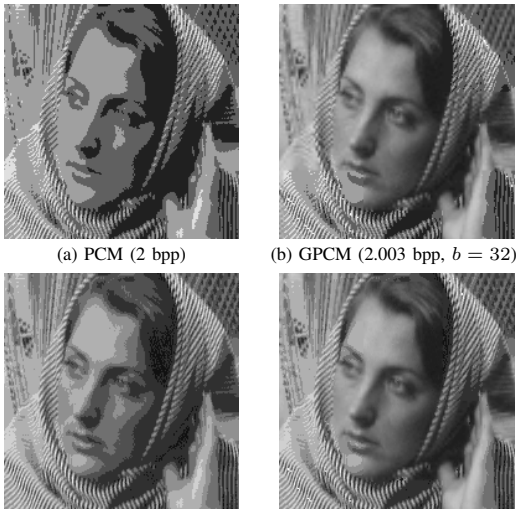



Fig. 2. PSNR as a function of the rate of six digital images coded using GPCM, PCM, JPEG, and JPEG-LS.



(a) PCM (2 bpp) (b) GPCM (2.003 bpp, $b = 32$)

(c) PCM (3 bpp) (d) GPCM (3.003 bpp, $b = 32$)

Fig. 3. Image *Barbara* encoded using PCM and GPCM (with $b = 32$).

## REFERENCES


[1] N. S. Jayant and P. Noll, *Digital coding of waveforms*. Prentice-Hall, Inc, 1984.

[2] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 80–94, Sep. 2004.

[3] H. Shum and T. Komura, "Tracking the translational and rotational movement of the ball using high-speed camera movies," in *Proc. IEEE Int. Conf. Image Processing*, Genoa, Italy, Sep. 2005, pp. 1084–1087.

[4] J. Sekikawa and T. Kubono, "Spectroscopic imaging observation of break arcs using a high-speed camera," in *Proc. IEEE Conf. Electrical Contacts*, Sep. 2007, pp. 275–279.

[5] P. Gemeiner, W. Ponweiser, P. Einramhof, and M. Vincze, "Real-time SLAM with high-speed CMOS camera," in *Proceedings IEEE Int. Conf. Image Analysis and Processing*, Sep. 2007, pp. 297–302.

[6] M. El-Desouki, M. J. Deen, Q. Fang, L. Liu, F. Tse, and D. Armstrong, "CMOS image sensors for high speed applications," *Sensors*, vol. 9, no. 1, pp. 430–444, 2009.

[7] A. N. Kim, T. A. Ramstad, and I. Balasingham, "Very low complexity low rate image coding for the wireless endoscope," in *International Symposium on Applied Sciences in Biomedical and Communication Technologies*, Barcelona, Spain, Oct. 2011, pp. 90:1–90:5.

[8] T. H. Khan and K. A. Wahid, "Low power and low complexity compressor for video capsule endoscopy," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 10, pp. 1534–1546, Oct. 2011.

[9] J. Prades-Nebot, A. Roca, and E. Delp, "Modulo-PCM based encoding for high speed video cameras," in *Proc. IEEE Int. Conf. Image Processing*, San Diego, USA, Oct. 2008, pp. 153–156.

[10] J. Prades-Nebot, "Very low-complexity coding of images using adaptive Modulo-PCM," in *Proc. IEEE Int. Conf. Image Processing*, Brussels, Belgium, Sep. 2011, pp. 313–316.

[11] N.-M. Cheung, H. Wang, and A. Ortega, "Sampling-based correlation estimation for distributed source coding under rate and complexity constraints," *IEEE Trans. Image Process.*, vol. 17, no. 11, pp. 2122–2137, Nov. 2008.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.

[13] G. Wolberg, *Digital Image Warping*. Wiley-IEEE Computer Society, 1990.

[14] B. Zeng and A. Venetsanopoulos, "A JPEG-based interpolative image coding scheme," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Minneapolis, MN, Apr. 1993, pp. 393–396.

[15] M. Morbee, "Optimized information processing in resource-constrained vision systems: From low-complexity coding to smart sensor networks," Ph.D. dissertation, Faculty of Engineering and Architecture, Ghent University, Mar. 2011.

[16] E. Cho, M. J. Cho, and J. Eltinge, "The variance of sample variance from a finite population," in *International Journal of Pure and Applied Mathematics*, vol. 21, May 2005, pp. 387–394.

[17] MPEG-2, *Test Model 5 (TM5), Doc. ISO/IEC JTC1/SC29/WG11/93-225b*, Test Model Editing Committee, Apr. 1993.

[18] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using a new rate-distortion model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 246–250, Feb. 1997.

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

[20] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc, 1989.

[21] E. Feig and S. Winograd, "Fast algorithms for the Discrete Cosine Transform," *IEEE Trans. Signal Process.*, vol. 40, no. 9, pp. 2174–2193, Sep. 1992.