Preprocessing and Lossless Compression of Visual Biomedical Information

Roumiana Kountcheva
T&K Engineering, Sofia, Bulgaria

Abstract

In this work are presented some new approaches for efficient archiving of visual medical information of various kinds. The main attention is aimed at the archiving of scanned paper documents. For this, new algorithms for image preprocessing and object segmentation are presented. The preprocessing is based on adaptive filtration, used to reduce the noises in the image background (corresponding to the image of the paper), retaining the main information (graphics, texts, etc.) untouched. The lossless compression, whose algorithm is given in detail, is based on new method for adaptive run-length coding, which comprises image histogram analysis and data coding. Significant advantage of the method is that it never allows enlargement of the coded files. The work comprises also experimental results, obtained using the software implementation of the algorithm and comparison with the well-known standards JPEG and JPEG 2000. The method is extremely efficient when used for coding of images of biomedical signals of any kind and has relatively low computational complexity, which is a reason to propose its use in telemedicine and in medical support appliances. Same approach is suitable for use in wide variety of applications (for example, telemedicine, etc.), which is proved by the experimental results included.

Keywords: *Image compression, Lossless image coding, Image preprocessing, Object segmentation, Biomedical signals*.

1. Introduction

Physical medical archives often simply consist of paper documents. Of course, there are issues with paper, especially from a long-term storage perspective, but mold, mildew, moisture and improper handling can all ruin the documents. There is no guarantee that paper documents will survive, even in the short term. With the prevalence of scanners, many organizations are turning to digital document archiving solutions. Scanners allow users to capture digital images of their documents, which can then be stored on servers or hard drives. The biggest advantage to digitally archiving documents is that only virtual space is required to house them. Together with the introduction of electronic archives, more and more paper documents are scanned and stored. After scanning, the documents are saved in various image formats, such as TIFF, JPEG, or BMP. Some scanner software supports scanning directly to PDF, which is advantageous since PDF is intended for accurate and clear document representation, and the documents can also be compressed to a very small size, making storage and transmission much simpler. The main problem with digital

document archives is that software and format standards are constantly changing, and because of this, there is no guarantee that a file in a given format will be viewable in the future.

The basic approach for archiving of scanned documents and other imagery, which requires retained visual quality, is to use the famous standards JPEG and JPEG 2000, which answer these requirements to a very high degree. In some cases however, the image quality should be retained unchanged and the images should be losslessly coded. The methods developed to solve this task are based on some kind of lossless data coding, the most important and widely used of which could be classified as follows: Coding based on dictionaries [1]: Lempel-Ziv; Statistical coding: Run-length [7], Shannon-Fano [3], Huffman [4], Arithmetic [5] (Binary arithmetic coding, QM-coder); Coding based on data transforms [2, 6]: Burrows-Wheeler Transform; Predictive coding; Structured universal coding schemes [8] (Elias codes, Exponential-Golomb codes): these schemes generate variable-length code words with a regular structure; Hybrid coding [7, 8] - a mixture of some of the methods, mentioned above: LOCO-I, adopted in the JPEG-LS standard, and others. The coding algorithms, whose parameters are adaptive to some characteristics of the input data, are called "adaptive" [9-13].

In this work is presented one adaptive approach for preprocessing and efficient lossless archiving of scanned documents, used in the medical practice.

The paper is arranged as follows: in Section 2 are presented the algorithms for the image preprocessing; in Section 3 are given the fundamentals of the method for lossless image data compression; in Section 4 are given some Experimental results, and section 5 is the Conclusions.

2. Image preprocessing

The image preprocessing comprises two basic steps: image filtration and image segmentation. The image filtration is needed, because in the scanned images usually exist noises in the part of the image, which corresponds to the paper. In result of the filtration such noises are removed or suppressed, which ensures higher efficiency of the image compression and archiving. The image segmentation is needed, because in result is defined the part of the image, which contains the main information (in this case – the text), which permits the part with the text to be compressed losslessly, while the remaining part of the scanned document can be compressed visually lossless.

2.1. Image filtration

Two kinds of filtration aimed at the scanned documents' efficient preprocessing depending on the image content, are proposed in this paper. The first one is used for the noise removal. These are noises in the scanned documents, which usually represent some small irregularities of the white paper. For this operation, is used a kind of locally adaptive filtration [14], which efficiently removes the additive Gaussian noise and has relatively low computational complexity. The texts/graphics in the image are not affected by the filter performance, because their basic characteristics (dimensions, area, shape, contrast, etc.) are quite different from these of the background noise.

The performance of the locally adaptive digital Wiener filter is presented below:

$$y(i,j)=\begin{cases}\mu_x(i,j)+\dfrac{\sigma_x^2(i,j)-v_x^2}{\sigma_x^2(i,j)}[x(i,j)-\mu_x(i,j)] & \\ & if\ \sigma_x^2(i,j)\geq v_x^2; \\ \mu_x(i,j), & if\ \sigma_x^2(i,j)<v_x^2,\end{cases} \tag{1}$$

where, $x(i,j)$ and $y(i,j)$ are the pixels of the original and the filtered image, respectively.

Here:

$$\mu_x(i,j)=\frac{1}{(2N_1+1)(2N_2+1)}\sum_{m=-N_1}^{N_1}\sum_{n=-N_2}^{N_2}x(i+m,j+n), \tag{2}$$

$$\sigma_x^2(i,j)=[\frac{1}{(2N_1+1)(2N_2+1)}\sum_{n=-N_1}^{N_1}\sum_{m=-N_2}^{N_2}x^2(i+n,j+m)]-\mu_x^2(i,j) \tag{3}$$

$$v_x^2=\frac{1}{M_1\times M_2}\sum_{i=1}^{M_1}\sum_{j=1}^{M_1}\sigma_x^2(i,j)\cdot \tag{4}$$

In these relations, $\mu_x(i,j)$ and $\sigma_x^2(i,j)$ represent the mean value and the variance of the pixel x(i,j) respectively, in a local window of size $(2N_1+1)\times(2N_2+1)$; N_1 and N_2 are positive integer numbers; M_1 and M_2 represent the size (in pixels) of the digital image matrix and ν_x^2 is the noise variance.

The second filtration is aimed at the correction of the uneven background illumination. This operation is necessary, because the next processing step, related to the text detection and segmentation, is simplified. The method for background illumination correction offered here, is based on the use of a *2D* nonlinear digital filter [15], which performance is presented by the equation below:

$$z(i,j)=x(i,j)-MS_B\{x(i,j)\}+\mu_x \tag{5}$$

Here $MS_B\{x(i,j)\}$ is a morphological smoothing filter, defined as follows:

$$MS_B\{x(i,j)\}=\min(\max(\max(\min(x[i-m,j-n])))) \quad \text{for} \quad \{m,n\}\in[B], \tag{6}$$

where the term $[B]$ is a rectangular "structuring element" represented by a matrix of size $(2N_1+1)\times(2N_2+1)$, whose elements are equal to zero. This matrix is used to set a symmetrical

window around the processed pixel $x(i,j)$ and after that is defined by the pixel with minimum or maximum brightness value. This value is used to substitute the value of $x(i,j)$ in the consecutive filtration stages, performed recursively. The parameters N_1, N_2 of the morphological smoothing filter are defined in accordance with the parameters of the processed background (size of the objects, etc.). The described filtration usually causes some specific distortions at the image edges. In order to avoid this, the image matrix was artificially made larger by adding pixels in both directions (horizontal and vertical). As a result, the original matrix of size $M_1 \times M_2$ becomes of size $(M_1+2N_1)\times(M_2+2N_2)$. The easiest way to change the image size is to add zeros in both directions [16], but the result images usually have some distortions, called zero-padding artifacts. In order to avoid this, in the method implementation, presented here, was used image replication of size equal to that of the filter window side.

2.2. Image segmentation

The image segmentation is based on the image histogram analysis. The histograms of the processed images (already filtered in accordance with the algorithms presented above) usually have only one maximum, which corresponds to the image background, and the algorithms based on recursive merging operations [17] are not suitable. For this reason, the proposed text segmentation is based on the so-called "triangle" algorithm [15], modified for this application.

The segmentation threshold is determined by performing the following operations.

1. Calculation of the image histogram:

   $H(x)$ for $x = 0,1,..,Q-1$, where Q is the number of grey levels;

2. In the image histogram $H(x)$, are defined 3 points:

   - First point (H_0,x_0) – the maximum of the histogram, which is usually the mean value of the corrected background;
   - Two points (H_1,x_1) and (H_2,x_2), which are defined by the relations:

   $$H_1(x_1) = H_2(x_2) = \psi H_0(x_0),$$

   $$H1(x_1) < H_0(x_0) > H_2(x_2) \qquad \text{for } x_1 < x_0 < x_2. \tag{7}$$

   The value of the parameter ψ is usually set to be $\psi = 0.1$;

3. The equations of two straight lines are defined, which join the two pairs of points, (H_1,x_1), (H_0,x_0) and (H_2,x_2), (H_0,x_0), respectively. These equations are defined by the well-known straight line equation:

   $$Ax + BH + C = 0, \tag{8}$$

where for the first line:

$$x_1 \le x = x_0,\ A = H_1 - H_0,\ B = x_0 - x_1,\ \text{and}\ C = H_0 x_1 - H_1 x_0; \tag{9}$$

And correspondingly, for the second line we have:

$$x_0 \le x = x_2,\ A = H_2 - H_0,\ B = x_0 - x_2,\ \text{and}\ C = H_0 x_2 - H_2 x_0. \tag{10}$$

4. The distance to the first and second line respectively is calculated for each point of the histogram (H, x), using the relation:

$$D(x) = \frac{Ax + BH + C}{\sqrt{A^2 + B^2}}, \tag{11}$$

where A , B, and C are defined by the equation of the corresponding straight line.

5. The value θ of the variable x is defined, for which the distance $D(\theta) = max$. This value is the needed segmentation threshold, for which from Eq. 11 for the range $x_1 \le x = x_0$ is obtained:

$$H(\theta_1 + 1) - H(\theta_1) \approx \frac{H_0 - H_1}{x_0 - x_1} \tag{12}$$

and correspondingly for the range $x_0 \le x = x_2$ we have:

$$H(\theta_2 + 1) - H(\theta_2) \approx \frac{H_2 - H_0}{x_2 - x_0} \tag{13}$$

Here θ_1 and θ_2 are the image segmentation thresholds, used to separate respectively the dark text and the light parts in the image. Steps 4 and 5 represent the algorithm modification.

6. The image is then binarized in accordance with the relations:

$$p_1(i, j) = \begin{cases} 1, \text{if } x(i, j) \le \theta_1; \\ 0, \text{if } x(i, j) > \theta_1, \end{cases} \tag{14}$$

$$p_2(i, j) = \begin{cases} 1, \text{ if } x(i, j) \ge \theta_2; \\ 0, \text{if } x(i, j) < \theta_2, \end{cases} \tag{15}$$

The binary image $p_1(i,j)$ contains the detected and separated dark signs in the image (in this case, texts and graphics), and $p_2(i,j)$ – the light ones (i.e., parts of the image, whose brightness is higher than that of the equalized background). The image obtained is saved and used for further processing. Usually the images with texts and graphics need the detection of the dark parts only, i.e. Eq. 14 is enough for the processing.

In case, that the image background is not white paper, but some kind of a picture (such images are usually obtained when old handwritten documents are archived), the image processing in accordance with the algorithm, given above, continues as follows:

7. The binary image $p_1(i,j)$, which contains text/graphics only, is used as a mask and each pixel from the original image, which corresponds to dark parts, is substituted by a preset gray level (usually, the most frequent value, detected from the image histogram).

The so obtained image (which corresponds to the image background, i.e. – the image of the paper) is then processed with some kind of lossy compression and the compressed image is saved.

8. The extracted text/graphic image is compressed with lossless compression and the obtained compressed image is saved.

To recover the original, the two restored images (corresponding to the background and the text) are composed together.

3. Adaptive Run-length Data Coding

Basic Principles

The Adaptive Run-Length (ARL) coding method is aimed at the compression of image data, represented as an N-dimensional sequence of n-bits binary words (numbers) x_k for $k=1,2,...,N$, whose values are in the range: $(-2^{n-1}) \le x_k \le (2^{n-1}-1)$. The method comprises two basic steps: the first is assumed as a data preparation, because in result, the data is transformed in such a way, that to enhance the coding, performed in the second step. The image compression is based on new run-length coding of sequences of equal or regularly changing values. In result, each N-dimensional sequence, which represents the image data, is substituted by a shorter one, which contains a header and the compressed data. The method is described below.

3.1. Preliminary processing of the input data

The original image data sequence x_k for $k=1,2,...,N$ is transformed into the sequence v_k of same length N so, that to obtain sequences of maximum length and of same value (in particular, this value is equal to zero). For this, the following operations are performed:

❖ Analysis of the histogram of the processed image (the input data), x_k:

- The histogram $H(x)$ is calculated for $x = -2^{n-1}, -2^{n-1}+1, ..., -1, 0, 1, ..., 2^{n-1}-1$. Here $H(x)$ is the count of the numbers whose value is equal to x.

- The count $L(x)$ is defined, corresponding to the values in the histogram, which are not used (free), and for which $H(x)=0$, for the case, when $x = -2^{n-1}, -2^{n-1}+1, .., -1, 0, 1, .., 2^{n-1}-1$:

$$L(x) = \sum_{x=-2^{n-1}}^{2^{n-1}-1} f(x) \quad \text{for} \quad f(x) = \begin{cases} 1, & \text{if } H(x)=0, \\ 0, & \text{if } H(x)\neq 0. \end{cases} \tag{16}$$

- The positions $p_i=x_i$ are defined, which correspond to start points (values) of the sequences of free values in the histogram $H(x)$. The lengths (Δl_i+1) of these sequences are defined in correspondence with the relation:

$x \in [p_i, p_i + \Delta l_i]$ for $i=1,2,...,T(x)$, when $L(x)>0$ and $H(x)=0$. (17)

- The sequence of free values and of maximum length is detected:

$$p(x)=p_i \text{ and } l(x)=\Delta l_i=max \text{ for } i=1,2,...,T(x). \quad (18)$$

In case that there is more than one sequence of free values and of same maximum length, then is chosen the one, whose start position has smallest value.

❖ The data sequence x_k is transformed into the sequence y_k by using a coding, which was developed for the implementation of this method and was named "size-saving prediction coding" (SSP). The transformation is performed in correspondence with the relation:

$$y_k = \begin{cases} (x_k - x_{k-1}), \text{ if } (-2^{n-1}) \le (x_k - x_{k-1}) \le (2^{n-1}-1), \\ -(x_k+1) \quad - \quad \text{in all other cases,} \end{cases} \quad (19)$$

for $k=1,2,...,N$, and $x_0=0$. In result, the sequences of same numbers in x_k are transformed into sequences of zeros in y_k.

❖ The histogram $H(y)$ of the data y_k is calculated for $y= -2^{n-1},-2^{n-1}+1,...,-1,0,1,..,2^{n-1}-1$ and is analyzed in the way, already done for x_k; the positions $p_i=y_i$ are defined, which correspond to the initial points of the sequences of free values in the histogram. Their lengths are calculated in correspondence with the relation:

$y \in [p_i, p_i + \Delta l_i]$ for $i=1,2,..,T(y)$, when $L(y)>0$ and $H(y) = 0$.

The longest sequence of free values is detected, for which:

$$p(y)=p_i \text{ and } l(y)=\Delta l_i=max \text{ for } i=1,2,...,T(y). \quad (20)$$

❖ The conditions $L(x)=0$ and $L(y)=0$ are checked, which are satisfied if only there are no sequences of free values in the two histograms. A special flag is set, which indicates if the image data is suitable for compression. This flag is used as one bit in the control word of the header of the losslessly coded data, named F_1. In case that conditions $L(x)=0$ and $L(y)=0$ are satisfied, the preliminary transform of the input data is stopped, and the coding ends. In all other cases the flag $F_1=1$. After that is checked which data sequence (x_k or y_k) is more suitable for the lossless coding (i.e. which one will ensure higher compression) The choice is done in accordance to the relation:

$$y_k=\begin{cases} x_k;\ F_2=0, H(y)=H(x), L(y)=L(x), p(y)=p(x), l(y)=l(x) \text{ if } L(y)=0 \text{ or } L(x)>2^n-4; \\ y_k;\ F_2=1 \qquad\qquad - \qquad \text{for SSP coding.} \end{cases} \tag{21}$$

Here F_2 is a flag (another bit in the control word), which indicates the selected sequence ($F_2=1$ if the sequence y_k better suits the coding method).

❖ The value $y=r(y)$, is defined, for which $H(y)=max$, when:

$$y = -2^{n-1}, -2^{n-1}+1, \ldots, -1, 0, 1, \ldots, 2^{n-1}-1.$$

❖ The data are modified transforming every word of the sequence y_k into v_k, by subtracting r(y) without setting carry in correspondence with the relation:

$$v_k=\begin{cases} [y_k-r(y)] & \text{if} \quad [y_k-r(y)]\in[-2^{n-1}, 2^{n-1}-1]; \\ [y_k-r(y)-2^n] & \text{if} \quad [y_k-r(y)]>2^{n-1}-1; \\ [y_k-r(y)+2^n] & \text{if} \quad [y_k-r(y)]<-2^{n-1}. \end{cases} \text{ for } k=1,2,..,N. \tag{22}$$

❖ The histogram $H(v)$ for $v= -2^{n-1}, -2^{n-1}+1, \ldots, -1, 0, 1, \ldots, 2^{n-1}-1$ of the data sequence v_k is calculated and analyzed in the already described way for x_k. The count $L(v)$ of the free histogram values is also calculated and the positions $p_i=v_i$ are defined, which indicate the initial points of the sequences of free values. The lengths (Δl_i+1) of these sequences in the histogram $H(v)$ are defined as follows:

$$v\in[p_i, p_i+\Delta l_i] \text{ for } i=1,2,\ldots, T(v), \text{ for } H(v)=0. \tag{23}$$

The longest sequence of free values is defined below:

$$p(v)=p_i \text{ and } l(v)=\Delta l_i = max \text{ for } i=1,2,\ldots,T(v). \tag{24}$$

❖ The flag F_3 is set, which indicates that the length of the sequence of free values is bigger than 1. It is set in the coded data control word, in accordance with the length [$l(v)+1$] of the detected sequence of free values, as follows:

$$F_3=\begin{cases} 1, \text{if } l(v)>0; \\ 0, \text{ if } l(v)=0. \end{cases} \tag{25}$$

With this, the first step of the data processing is finished. In result is obtained the transformed data sequence v_k for $k=1,2,..,N$ and is defined the additional information about $r(y)$, $p(v)$ and $l(v)$, which are processed in the second step, presented below.

3.2. Coding

In this part of the processing is composed the header of the coded data, needed for the proper coding/decoding of the transformed data sequence, v_k. This part of the processing comprises the following basic operations:

Composition of the coded data header

The header is needed for the proper decoding of the compressed data and consists of control word w_{00}, which comprises the three control flags F_1, F_2 and F_3 and additional information, presented in detail below.

- for $F_1=0$ the header does not contain additional information;

- for $F_1=1$ the header contains additional information, which comprises the numbers $w_{01}=r(y)$ and $w_{02}=p(v)$. Here $r(y)$ is the most frequently met value in the data sequence y_k, and $p(v)$ is the start position of the longest sequence of free values in the data sequence v_k , which was detected first;

- for $F_1=1$ and $F_3=1$ the additional information in the header includes one more number $w_{03}=l(v)$, which defines the length of the sequence with start position $p(v)$, decreased by 1.

Coding of the transformed data

➢ For $F_1=1$ the transformed data are processed in accordance with the new method for adaptive lossless coding. For this, each sequence of numbers of same value in v_k is substituted by w_s, and in result is got a shorter data sequence:

- Each sequence of zeros $v_d=v_{d+1}=....=v_{d+P-1}=0$ of length P, in the range $1<P\leq l(v)+1$, which had been detected in v_k is substituted by one n-bit word $w = p(v)+P-1$, i.e.:

$$(v_d, v_{d+1},.., v_{d+P-1}) \Rightarrow (w = p(v) + P-1). \quad (26)$$

For $P=1$ the sequence contains one zero only ($v_d =0$) and the coding is not performed;

- Each zero sequence $v_d=v_{d+1}=....=v_{d+P-1}=0$ of length P which is in the range $2^{mn}\geq P>l(v)+1$ for $m\geq 1$, detected in v_k, is substituted by $2m$ words, of n bits each. In the first word is stored $p(v)$, in the next $(m-1)$ words is stored zero, and in the remaining m words – the coded number $(P-1)$, i.e.:

$$(v_d, v_{d+1}, \ldots, v_{d+P-1}) \Rightarrow (w_1=p(v), w_2=0, \ldots, w_m=0, w_{m+1}=P_1,..,w_{2m}=P_m), \quad (27)$$

where $P_1,...,P_m$ corresponds to the number $(P-1)$, represented by m words of n bits each (here P_1 is the *MSW*, and P_m - the *LSW*);

- Each sequence of same numbers, not equal to zero $v_d=v_{d+1}=....=v_{d+P-1}=v$, and of length P in the range $2^{mn}\geq P>4$ for $m\geq 1$, detected in v_k, is substituted by $(2m+2)$ words, of n bits each. In the first two words are stored the numbers $p(v)$ and "1", in the next $(m-1)$ words – zeros, in the next

m words – the number (*P-1*) and in the last word – the number, which is different from zero, $v \neq 0$ i.e.:

$$(v_d,\ v_{d+1},..,v_{d+P-1}) \Rightarrow (w_1=p(v),w_2=1,w_3=0,..,w_{m+1}=0,w_{m+2}=P_1,..,w_{2m+1}=P_m,\ w_{2m+2}=v), \quad (28)$$

where $P_1,...,P_m$ presents the number (*P-1*), coded as *m* words of *n* bits each (here P_1 is the *MSW*, and P_m - the *LSW*).

Sequences of non-zero values of length $P \leq 4$ are not losslessly coded, because this does not enhance the method efficiency.

In result of the coding the v_k data is transformed into the compressed sequence w_s for $s=1,2,...,S$ and $(-2^{n-1}) \leq w_s \leq (2^{n-1}-1)$. Here *S* is the count of the words in the sequence w_s, which is smaller than the total count *N* of the words in the original sequence x_k.

➢ For $F_1=0$ the input sequence x_k is not compressed and after the header follows the unchanged original data: $w_s=x_k$ for $s=k=1,2,...,N$ ($S=N$). In this case the volume of the original data is not changed (except the addition of one byte only).

The simplified block diagram of the method is shown on Figure (1).

3.3. Decoding

The decoding of the sequence w_s comprises the following operations:

- The flags in the control word of the coded data header w_{00} are analyzed consecutively. Two main cases are possible:

If $F_1=0$**,** this means that the input data sequence x_k had not been losslessly coded and the relation between the decoded data u_s and w_s is: $u_s = w_s$ for $s=1,2,3,...,S$ ($S=N$);

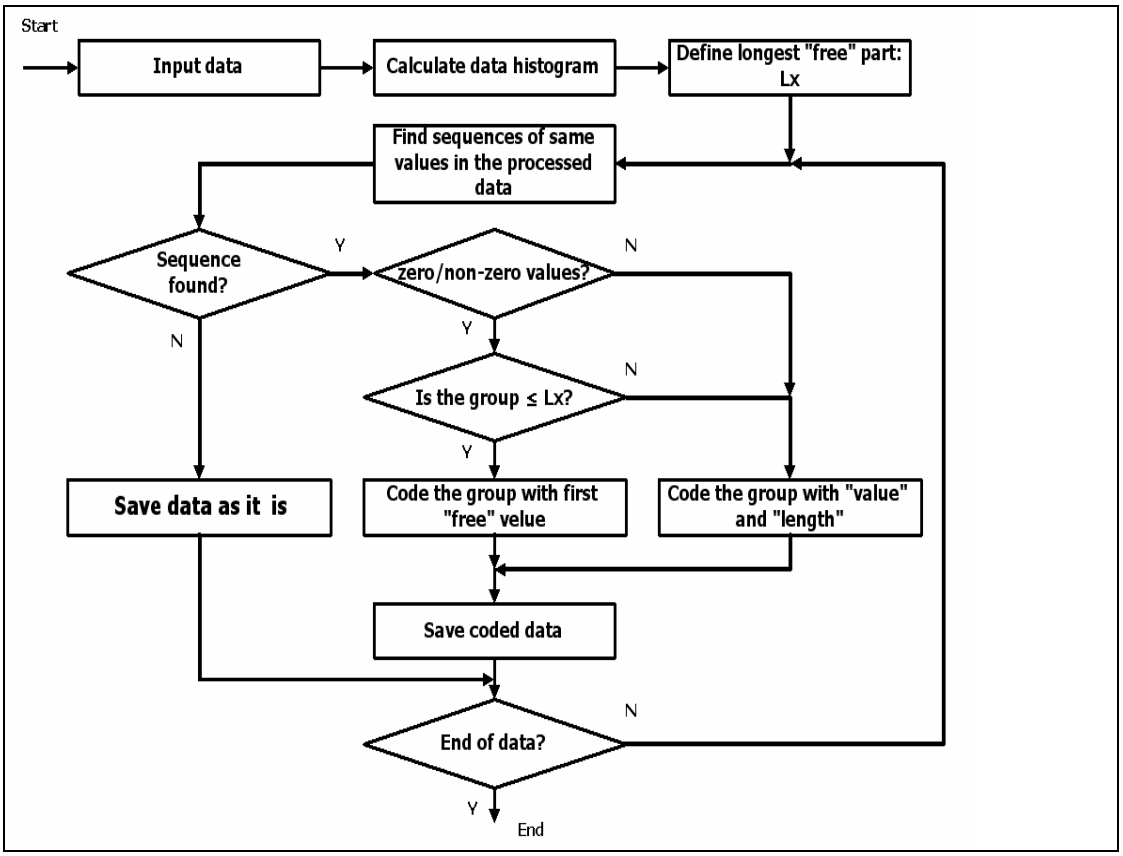


Figure (1): Block diagram of the ARL coding method

If $F_1=1$, this means that the input sequence x_k had been coded and the following steps are performed:

Step 1: The flag F_3 is analyzed (for $F_1=1$) and the decoding is performed. For the case, when $F_3=1$, is analyzed the available additional information from the header of the compressed sequence w_s: the numbers $r(y)=w_{01}$, $p(v)=w_{02}$, $l(v)=w_{03}$. For the case ($F_3=0$) this information comprises only the numbers $r(y)=w_{01}$ and $p(v)=w_{02}$ (for the decoding $l(v)=0$).

Then each value of w_s for $s=1,2,...,S$ and $F_3=0$ is compared with the number $p(v)$. Depending on the difference $\delta_s=w_s-p(v)$ when the value of w_s is decoded, it is retained, or substituted by the sequence of numbers of same value, $v_p=v$ for $p=1,2,...,P$ in correspondence to one of the followings procedures:

- For $\delta_s > l(v)$ or $\delta_s<0$, the value of w_s is not changed. In this case is performed the substitution $w_s \Rightarrow (v=w_s)$;

- For $0<\delta_s \le l(v)$, the value of w_s is substituted by the sequence v_p, of length $P=w_s-p(v)+1$, which consists of zeros only, and $w_s \Rightarrow (v_1=0, v_2=0, .., v_P=0)$;

- For $\delta_s=0$ and $w_{s+1}\neq 0$, the data w_s and w_{s+1} are substituted by the sequence v_p of length $P=w_{s+1}+1$, which consists of zeros only, and in result is obtained the data sequence $(w_s,w_{s+1})\Rightarrow (v_1=0, v_2=0,.., v_P=0)$;

- For $\delta_s=0$, $w_{s+1}=0,..,w_{s+m-1}=0$, $w_{s+m}\neq 0,..,w_{s+2m-1}\neq 0$, the data in w_s up to w_{s+2m-1} are substituted by the sequence v_p, consisting of zeros only, with length $P=(w_{s+m},..,w_{s+2m-1})+1$ (w_{s+m} is the *MSW*, and w_{s+2m-1} is the *LSW*) and then $(w_s,..., w_{s+2m-1})\Rightarrow (v_1=0, v_2=0,.., v_P=0)$;

- For $\delta_s=0$, $w_{s+1}=1$ and $w_{s+2}\neq 0$, the data in w_s up to w_{s+3} are substituted by the sequence v_p, consisting of the non-zero numbers $v=w_{s+3}$ of length $P=w_{s+2}+1$, and is obtained the result: $(w_s,..,w_{s+3}) \Rightarrow (v_1=v, v_2=v,.., v_P=v)$.

- For $\delta_s=0$, $w_{s+1}=1$, $w_{s+2}=0,..,w_{s+m}=0$, $w_{s+m+1}\neq 0,..,w_{s+2m}\neq 0$, the data in w_s up to w_{s+2m+1} are substituted by the sequence v_p, consisting of the non-zero numbers $v=w_{s+2m+1}$ of length $P=(w_{s+m+1},..,w_{s+2m})+1$. Here w_{s+m+1} is the *MSW*, and w_{s+2m} is the *LSW*; and then:
$(w_s,...,w_{s+2m-1})\Rightarrow (v_1=v, v_2=v,.., v_P=v)$.

At the end is obtained the decoded sequence, v_k.

Step 2. Inverse data modification is performed, which transforms every word v_k into y_k by adding $r(y)$ to its value without carry, in accordance to the relations:

$$y_k=\begin{cases}[v_k+r(y)] & if \quad [v_k+r(y)]\in[-2^{n-1}, 2^{n-1}-1];\\ [v_k+r(y)-2^n] & if \quad [v_k+r(y)]>2^{n-1}-1;\\ [v_k+r(y)+2^n] & if \quad [v_k+r(y)]<-2^{n-1}.\end{cases} \text{ for } k=1,2,..,N; \qquad (29)$$

Step 3. The flag F_2 is analyzed when operations, defined in accordance with values of flags F_1 and F_3, are finished. If $F_2=0$, this indicates that the sequence x_k had not been SSP coded and then $u_k=x_k=y_k$ for $k=1,2,...,N$. In case, that $F_2=1$, is necessary to transform y_k into u_k performing the SSP decoding:

$$u_k=\begin{cases}(y_k+u_{k-1}), & if \ 2^{n-1}\le (y_k+u_{k-1})\le (2^{n-1}-1),\\ -(y_k+1) & - \quad in\ all\ other\ cases.\end{cases} \text{ for } k=1,2,..,N \text{ and } u_0=0. \qquad (30)$$

In result is obtained $u_k=x_k$, and the decoding is finished.

3.4. Evaluation of the Lossless Coding Method Efficiency

The compression ratio is calculated as a relation between the original and the compressed data, i.e. $CR=N/(S+W)$, where W is the number of words in the header. The minimum value $CR_{min}=N/(N+1)<1$ is obtained for the case, when the data x_k are not compressed. In this case $S=N$ and $W=1$, because the header consists of one word only (the control word). The maximum value of CR

is obtained for $x_k=x$ ($k=1,2,...,N$). In this case $P=N$, $S=2m$, $W=4$ and $CR_{max} \le N/[(2/n)lg_2N+4]$. From this follows that the compression ratio is in the range:

$$\frac{N}{N+1} \le CR \le \frac{N}{\lg_2 \sqrt[n]{N^2}+4} \text{ ; For } N>>1 \text{ is obtained } 1 \le CR \le N/lg2\sqrt[n]{N^2}. \quad (31)$$

The analysis shows that the compression ratio can achieve significant values when there are long sequences of same numbers in v_k.

The presented method for lossless image compression is suitable for coding of images, whose brightness or color histograms have free values, or they have large areas of same or continuously changing brightness/color. Such histograms are typical for scanned documents, graphics, fingerprints, medical signals, etc. The processing of each of the color components is performed in the way, described above for grayscale images.

3.5. Advantages of the lossless compression method

Specific for the method for lossless coding/decoding, presented above, is that its performance comprises mainly operations "sorting" and "adding", while the operation "multiplication" is not used. In result, the algorithms of the ARL implementation are fast and the decoding is simpler than the coding. The method ensures also very high compression ratios for cases, when the original data contains long sequences of same values. These basic characteristics distinguish the new method from the famous methods for RL data coding.

One more very important advantage of the method is that it never creates a coded file, larger than the original.

4. Experimental Results

The experiments were performed with the still image database of the Laboratory for Image and Sound Processing of the Technical University of Sofia, Bulgaria. The experiments included more than 400 grayscale (8 bpp) and color (24 bpp) still images. An enlarged part of a scanned document is shown on Figure (2.a), and its' histogram - on Figure (2.b). A part of scanned white paper without text is shown on Figure (3.a), and the corresponding histogram - on Figure (3.b).

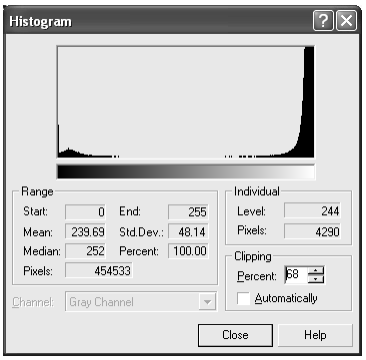
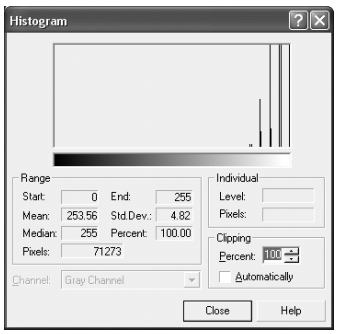


Figure (2.a): A part of scanned document (396×286, 8 bpp) **Figure (2.b): The image histogram**

Figure (2) Test document

The large number of relatively dark points, seen on Figure (3.a), corresponds to the small irregularities of the paper roughness and brightness. Specific for the histograms on Figure (2.b) and Figure (3.b) is that they both have many not used (free) values, which is a sure prerequisite for the efficient lossless coding.

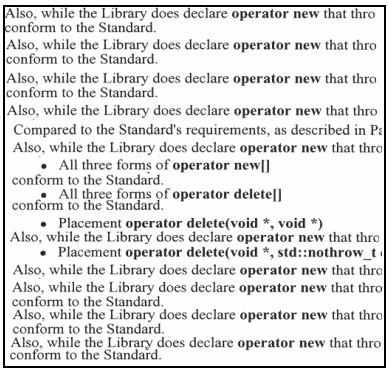


a) Example image of scanned paper **b) the image histogram**

Figure (3): Part of the scanned paper, which does not contain text.

A part of the test images are shown on Figure (4). The results obtained for a part of the test images are shown in Table 1. The results are average for the shown classes of images and prove the efficiency of the presented method. The compression ratios obtained for JPEG 2000LS are much lower. For same compression ratio JPEG 2000 gives much worse quality. In Tables 2 and 3 are shown the results obtained for the widely used formats PNG and GIFF.

a) "ECGs" (210×151)


Also, while the Library does declare **operator new** that thro
conform to the Standard.
Also, while the Library does declare **operator new** that thro
conform to the Standard.
Also, while the Library does declare **operator new** that thro
conform to the Standard.
Also, while the Library does declare **operator new** that thro
Compared to the Standard's requirements, as described in Pa
Also, while the Library does declare **operator new** that thro
- All three forms of **operator new[]**

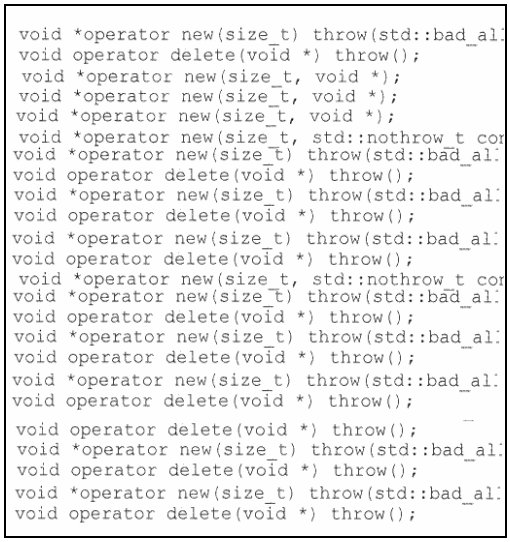
conform to the Standard.
- All three forms of **operator delete[]**

conform to the Standard.
- Placement **operator delete(void *, void *)**

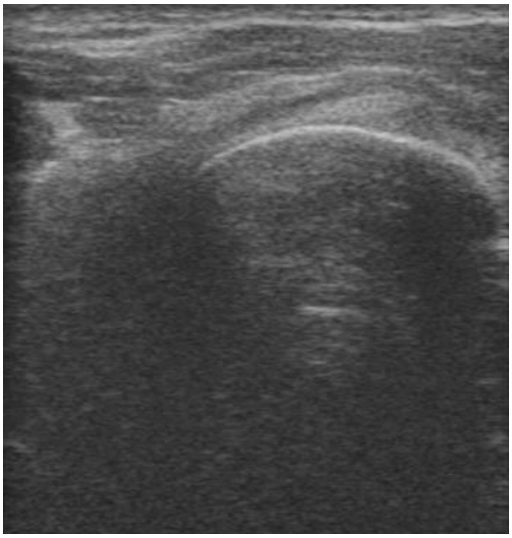
Also, while the Library does declare **operator new** that thro
- Placement **operator delete(void *, std::nothrow_t**

Also, while the Library does declare **operator new** that thro
Also, while the Library does declare **operator new** that thro
conform to the Standard.
Also, while the Library does declare **operator new** that thro
conform to the Standard.
Also, while the Library does declare **operator new** that thro
conform to the Standard.

b) "Text" (512×512)


```
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t, void *);
void *operator new(size_t, void *);
void *operator new(size_t, void *);
void *operator new(size_t, std::nothrow_t cor
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t, std::nothrow_t cor
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
void *operator new(size_t) throw(std::bad_al
void operator delete(void *) throw();
```

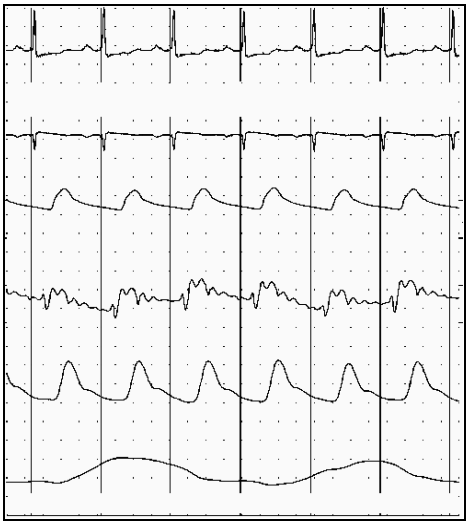
c) Test image "Text 2" (512x512, 8bpp)



d) Test image "Coronal" (350x432, 24bpp)

e) Test image "Axial1" (350x432)

f) Test image "Axial2" (344x430)



g) Test image "Mimic1" (498 x 536, 8bpp)

h) Test image "Slpdb" (498 x 539, 8bpp)

Figure (4): Example test images

Table 1. Results obtained for the size of compressed files

Image	ARL CR	ARL PSNR [dB]	JPEG HQ CR	JPEG HQ PSNR [dB]	JPEG2000 CR	JPEG2000 PSNR [dB]
Coronal	5.11	Infinity	5.59	58.48	6.62	Infinity
Axial1	5.35	Infinity	5.73	58.43	6.67	Infinity
Axial2	5.89	Infinity	5.70	18.52	6.56	Infinity
Text5	12.87	Infinity	2.50	65.96	4.31	Infinity
Text1	13.46	Infinity	0.85	60.83	1.89	Infinity
Text2	7.54	Infinity	1.19	62.79	2.25	Infinity
Cells	2.97	Infinity	1.95	62.43	2.74	Infinity

Table 2. Results obtained for the compressed files size

Image	Size [pixels]	$J2K_{LS}$ [KB]	PNG [KB]	GIF [KB]	ARL [KB]
Slpdb	498 x 539	54,0	10,60	10,6	7,8
Mimic2	711 x 589	40,5	8,36	8,56	6,0
Mimic1	498 x 536	60,9	11,20	12,8	8,59
Mimic3	498 x 536	59,6	12,88	11,23	8,53

Table 3. Results obtained for the compression ratio (CR)

Image	Size [pixels]	$J2K_{LS}$(CR)	PNG (CR)	GIF (CR)	ARL (CR)
Slpdb	498 x 539	4,97	25,32	25,32	34,41
Mimic2	711 x 589	10,34	50,09	48,92	69,43
Mimic1	498 x 536	4,38	23,83	20,85	31,10
Mimic3	498 x 536	4,47	20,72	23,76	31,29

5. Conclusions

The ARL method was initially developed for efficient compression of the data, obtained in result of the Inverse Pyramid Decomposition. The method was further developed and expanded in order to become more adaptive and suitable for efficient compression of still images with various contents. The results obtained confirmed the method efficiency for all cases, when the processed histogram has sequences of not used values.

The experimental results obtained with the software implementation of the method confirmed the theoretical conclusions. In order to obtain higher compression ratio, the ARL method could be further improved, combining it with other famous methods for image pre-processing (histogram modification, image pyramidal decomposition, etc.), and some of the methods for lossless compression (Huffman coding, arithmetic coding, etc.).

On the basis of the method evaluation follows that the main applications of the ARL coding method and its future development are in the areas:

- The creation of new lossless archiving format, which to be used for multimedia databases archiving;
- The creation of new algorithms for image compression, which to be integrated in the existing standards (tiff, etc.), and to be used in the application software for scanners, faxes, photo and video cameras, etc.;
- The development of modifications, which to suit various devices for data transfer: faxes, mobile phones, etc., medical equipment, surveillance systems, remote control systems, smart cards, etc.;

The fast development of new information technologies will ensure further expansion of the possible applications of the ARL compression method.

References

[1] J. Ziv, A. Lempel, "Compression of individual sequences via variable rate coding", IEEE Trans. on Information Theory, IT-24(5), 530-535, 1978.

[2] M. Burrows, D. Wheeler, "A block-sorting Lossless Data Compression Algorithm", Digital Systems Research Center. SRC Report, 124, 1994

[3] R. Fano, "Transmission of Information", MIT Press, Cambridge, MA, 1949

[4] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proc. of the IRE, 40(9), 1098-1101, 1952.

[5] Golomb, W. Solomon, "Run-Length Encodings", IEEE Trans. on Information Theory IT-12(3), 399–401, 1966

[6] I. Witten, R. Neal, J. Cleary, "Arithmetic Coding for Data Compression", Communications of the ACM, 30(6), 1987

[7] T. Raita, J. Teuhola, "Predictive text compression by hashing", ACM Conf. on Information Retrieval, 1987.

[8] L. Karam, "Lossless Image Compression", The Essential Guide to Image Processing, Ch. 16, Al Bovik (Ed.), Academic Press, 385 – 420, 2009

[9] D Y. Shi, H. Sun, "Image and video compression for multimedia engineering: fundamentals, algorithms, and standards, 2nd Ed., CRC Press, 2008

[10] T. Bell, I. Witten, J. Cleary "Text Compression", Prentice-Hall, 1990

[11] A. Storer, H. Helfgott. "Lossless Image Compression by Block Matching". The Computer Journal 40(2/3), 137–145, 1997

[12] T. Acharya and P. Tsai, "JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures", John Wiley & Sons Ltd. UK, 2005

[13] Kh. Sayood (Ed), "Lossless compression Handbook". Academic Press, 2003

[14] J. Lim, "Two-dimensional Signal and Image Processing", Prentice Hall, 1990.

[15] I. Young, J. Gerbrands, L. van Vliet, "Image Processing Fundamentals", CRC Press, 2000.

[16] R. Gonzalez, R. Woods, "Digital Image Processing", Prentice Hall, 2002.

[17] J. Delon, A. Desolneux, J.L. Lisani, A. Petro, "A Nonparametric approach for Histogram Segmentation", *IEEE Trans. on Image Processing*, 16(1), 253-261, 2007.