ON THE CODING OF DIGITIZED PICTURES

G. FAZEKAS

Mathematical Institute

L. Kossuth University H-4010 Debrecen,

Received August 11, 1988

1. Introduction

The topic of my lecture is rather practical. Some years ago we examined certain coding methods in order to decide whether these methods were applicable to the coding of digitized pictures. The research was sponsored by the Mathematical Laboratory of The Institute for Coordination of Computer Techniques (SZKI Budapest). The problems we considered were posed by József Dénes (former head of Math. Lab.).

Our investigations were based on the following telecommunicational model:

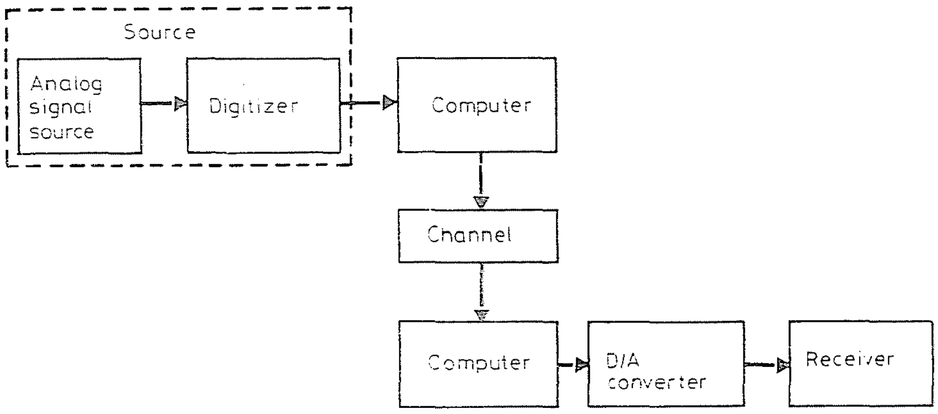

Fig. 1.

In this model, the analog signal source might be, for example, a TV camera, a microphone or any other equipment which produces continuous signals. The digitizer converts the continuous analog signals to digital signals. In our case, a source simply means the analog signal source together with the digitizer. The computer can store the series of digital signals in its own memory.

Then the digital information can be processed by the computer. On many occasions, such a processing may be the encoding of the digital signals before sending them over the channel. Such problems arise, for examples, when the digital signals are redundant and, when parallel processing of data is required. The secrecy (privacy) of the transmission of the signals requires special encoding too.

Mathematically a picture is defined by a function $f(x, y)$ where x and y denote the coordinates in the image plane and the function values are brightnesses or k-tuples of brightness values in several spectra. In the black-and white case the brightness values are called gray levels. The brightness or gray level values are nonnegative and bounded, of course. When a picture is digitized a sampling and quantizing method is used to extract from the picture a discrete set of integer numbers. The image plane is divided into small squares corresponding to a rectangular array system. These squares are called pixels. Samples usually are average gray levels taken over pixels. The samples are then quantized to a set of integers. The result of sampling and quantizing is a digital picture. Consequently, owing to the structure of pixels we can assume that a digital picture is a matrix of nonnegative integer numbers with p rows and q columns.

2. Permutation Coding

The first method we studied is the permutation coding which was originally introduced by D. Slepian in 1965. The basic idea of permutation coding can be formulated in the following way: Let us denote by

$$h = (h_0, h_1, \ldots)$$

the sequence of digital signals received from the source. Each h_i is an m-bit binary integer where m is uniquely determined by the digitizer. Let us denote by S_k the set of all permutations of $K = (0, 1, \ldots, k-1)$ and V_n^k the set of all k-tuples $(j_0, j_1, \ldots j_{k-1})$ with nonnegative integer j_i components where $j_i \leq n = 2^m - 1$. The sequence h can be segmented into consecutive blocks of the same length k, i.e.

$$h = (B_0, B_1, \ldots,) \in (V_n^k)^\infty,$$

where

$$B_i = (h_{ik}, h_{ik+1}, \ldots, h_{(i+1)k-1}) \in V_n^k, \quad i = 0, 1, \ldots$$

Let us give a mapping φ: $V_n^k \to S_k$. A mappin g Φ: $(V_n^k)^\infty \to S_k^\infty$ is called a permutation coding if

$$\Phi(h) = \Phi(B_0, B_1, \ldots) = (\varphi(B_0), \varphi(B_1), \ldots).$$

The number of elements of V_n^k is equal to $(n+1)^k$, and S_k has $k!$ elements. Consequently, a compression of stored data can be achieved by choosing a suitable value of k. The decoding can be defined by a leftinverse of φ. In many practical cases the construction of permutation codes (i.e. the definition of a φ) has been based on the arrangement of the elements of blocks B_i. More precisely: for every block $B = (b_0, b_1, \dots, b_{k-1})$ let

$$\varphi(B) = \begin{pmatrix} 0 & 1 & \dots & k-1 \\ i_0 & i_1 & \dots & i_{k-1} \end{pmatrix}$$

whenever $b_{i_j} \leq b_{i_{j+1}}$ for $j = 0, 1, \dots, k-1$.

It is obvious that this encoding procedure can be performed very quickly by choosing a suitable fast ordering algorithm. A disadvantage of this permutation coding can be that φ is in general not a one to one mapping therefore the decoded information may have some distortion.

If we assume that the samples h have a uniform distribution between their possible minimal and maximal value then the decoding procedure can be given by a look-up table. A look-up table is a vector of k monotonously increasing components $(a_0, a_1, \dots, a_{k-1})$ and the decoded block assigned to $\varphi(B)$ is $B' = (b'_0, b'_1, \dots, b'_{k-1})$ where $b'_{i_j} = a_j$ for $j = 0, 1, \dots, k-1$.

In our experiments, this idea has been used to the real time transmission of digitized voice over a wide band binary channel. We used A/D and D/A converters with 5000×8 bit per second. The permutation codes were permutations of order 128.

The main problem with the above decoding method is that the uniform distribution condition does not hold in the global sense. For instance, since the connected speech can be segmented into natural units (spoken words and intervals) this procedure causes very large noise for the intervals. In order to avoid this noise, i.e. to distinguish between spoken words and intervals, it seems to be necessary to extend the permutation code with some additional information. Very useful additional information can be the minimal and maximal value of the samples taken over the actual block. In this case, for the decoding procedure one should construct a look-up table for every block.

We remark that minimal and maximal sample values of a block can be 'inserted' into the code in the following way:

Let g be an integer valued quasi invertible function defined on the sample which has the property $0 \leq g(h_i) \leq k-1$ $(i = 0, 1, \dots)$. Then for every block B and $\varphi(B)$ there exist α and β such that $i_\alpha = g(b_{i_0})$ and $i_\beta = g(b_{i_{k-1}})$. Instead of $\varphi(B)$ we can construct a new permutation code of the block B as

$$\hat{\varphi}(B) = \begin{Bmatrix} 0 & 1 & 2 & \dots & \alpha & \alpha+1 & \alpha+2 & \dots & \beta & \beta+1 & \dots & k-1 \\ i_\alpha & i_\beta & i_0 & & i_{\alpha-2} & i_{\alpha-1} & i_{\alpha+1} & & i_{\beta-1} & i_{\beta+1} & & i_{k-1} \end{Bmatrix}.$$

This means that we take the indices which correspond to the minimal and maximal value of the sample block to the first positions of the permutation then we write simply the remaining part. The values standing on the i_α-th and i_β-th positions of the decoded block can be approximated by their neighbours. This method requires a little more processing time but it provides better results for digitized voice.

This encoding method can be used for coding pictures. On the other hand, we cannot assume that the sample values have a uniform distribution in a local sense after all. Namely a scene may have different number of patterns which may be situated in various ways on it. However, the following decoding procedure has been proved very useful for many types of scenes.

Let B, B' and $\varphi(B)$ be defined as above. Observe, that if $i_j > i_{j+1}$ in $\varphi(B)$ then $b_{i_j} < b_{i_{j+1}}$ in B. This implies that if the previous relation holds for s different values of j then at least $s+1$ different values occur in B. Suppose this number is exact and suppose we know the minimal and maximal values occurring in B. Let

$$\min_i b_i = a_0 < a_1 < \ldots a_j = \max_i b_i$$

be a look-up table, and we define the decoded block $B' = (b'_0, b'_1, \ldots, b_{k-1})$ assigned to $\varphi(B)$ as follows: Let

$$b_{i_0} = b_{i_1} = \ldots b_{i_{j_1}} = a_0$$

provided

$$i_0 < i_1 < \ldots < i_{j_1} > i_{j_1+1}$$

further let

$$b_{i_{j_1}+1} = b_{i_{j_1}+2} = \ldots = b_{i_{j_i}} = a_1$$

provided

$$i_{j_1+1} < i_{j_1+2} < \ldots < i_{j_2} > i_{j_2+1},$$

etc. We remark that if $\max_i b_i - \min_i b_i = s$ then B can be uniquely reconstructed in this way.

A concrete practical application of the above method has been made to encoding and decoding of digital pictures consisting of 300 by 400 pixels. A pixel has been stored by 6 bits. We used permutations of order 32 and 64. Figures 2, 3 and 4 show the process and effect of permutation coding for a picture consisting of 144 by 192 pixels. In Figure 3, permutation code values are represented by gray levels.

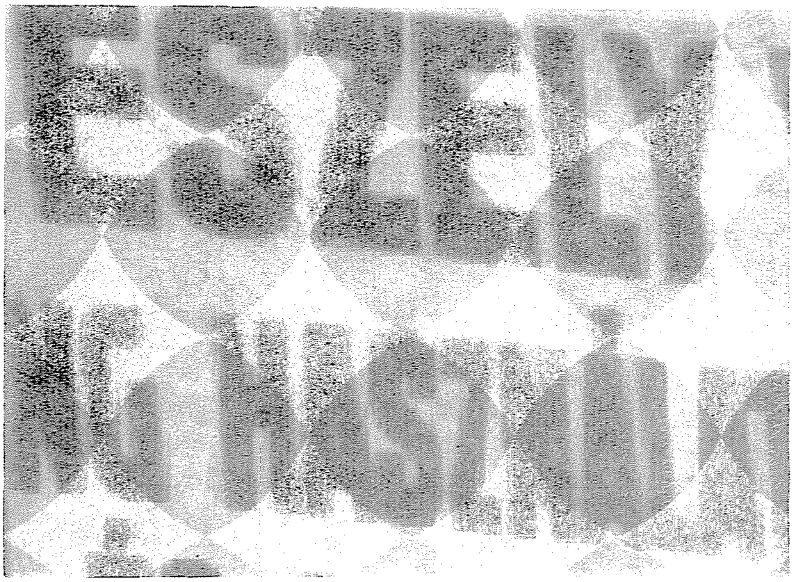

Fig. 2. Original picture

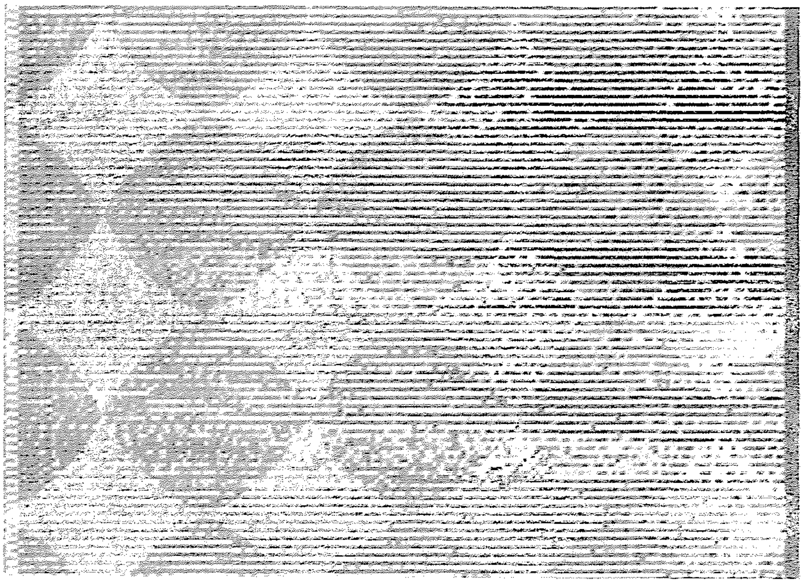

Fig. 3. Permutation code

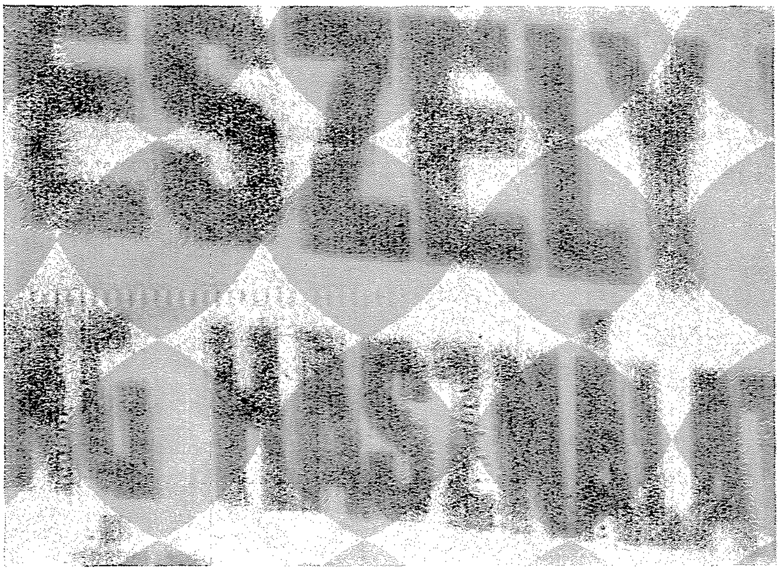

Fig. 4. Decoded picture

3. Coding with latin squares

The other method we applied to picture coding has been proposed by J. Dénes. It is based on latin squares. A latin square of order n is an arrangement of n symbols (e.g. the numbers $0, 1, \dots, n-1$) into a square matrix in such a way that no row and no column contains any symbol twice. Two latin squares are orthogonal, when one is superimposed upon the other, every pair of symbols occurs only once. It is known that not more than $n-1$ mutually orthogonal latin squares of order n can exist. Such a set of latin squares is called a complete set. (For a detailed treatise on latin squares see [3].)

The encoding procedure is very simple. We consider a given complete set of mutually orthogonal latin squares of order n and fix two orthogonal pairs from that set denoting them by A, B and C, D respectively. Suppose that the digital picture is divided into disjoint square windows of the same size $n \times n$. For each window

$$W = |w_{ij}|, \ i, j = 0, 1, \dots, n-1,$$

we construct a code matrix $\mathbf{K} = |k_{ij}|$ in the following way: In the firs step, let

$$w'_{ij} = w_{ij} + na_{ij} + b_{ij} \ (\mathrm{Mod}\ r) \ i, j = 0, 1, \dots, n-1$$

then let

$$k_{c_{ij}d_{ij}} = w'_{ij} \ i, j = 0, 1, \dots, n-1$$

where $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ denote the elements of A, B, C, D respectively and r depends on the storing system.

Obviously, the mapping $W \rightarrow K$ is one-to-one therefore the decoding is not a difficult problem provided we know the latin squares A, B, C, D. It is easy to see that we have $\binom{n-1}{2}^2$ possibilities to choose the pairs A, B and C, D from a fixed complete set, i.e. we have $\binom{n-1}{2}^2$ different keys for encoding.

It is important to choose n as large as possible. The question of the existence of complete set for a given n is an unsolved problem in general. A number of interesting results for several classes of integers can be found in the book of J. DÉNES and A. D. KEEDWELL [3]. To construct complete sets we used a very nice theorem of H. B. MANN [5] from 1942. This theorem states that, for any prime number $n > 2$, the system of squares

$$\mathrm{L} = \begin{bmatrix} 0 & 1 & \dots & n-1 \\ j & 1+j & \dots & n-1+j \\ 2j & 1+2j & \dots & n-1+2j \\ \cdot & \cdot & \dots & \cdot \\ (n-1)j & 1+(n-1)j & & n-1+(n-1)j \end{bmatrix}, \quad j = 1, 2, \dots n-1,$$

where the addition and the multiplication are modulo n, provides a complete set of mutually orthogonal latin squares of order n.

In our practical applications, n was 61 and 71.

References

1. BLAKE, I. F., COHEN, G., DEZA, M.: Coding with Permutations, Information and Control, 43/1979.
2. DÉNES, J.: On Some Connection Between Permutations and Coding, Discrete Mathematics, 56/1985.
3. DÉNES, J., KEEDWELL, A. D.: Latin Squares and Their Applications, Akadémiai Kiadó, Budapest, 1974.
4. FAZEKAS, G., PETHŐ A.: Permutation Source Coding, Alkalmazott Matematikai Lapok, 12/1986. (in Hungarian)
5. MANN, H. B.: On the Construction of Sets of Orthogonal Latin Squares, Ann. Math. Stat. 14/1943.

Gábor FAZEKAS H-4010, Debrecen, Pf. 12.