A Representation of Digital Pictorial Pattern

Endo, Tsutomu
Department of Information Science and Systems Engineering, Oita University

Kawaguchi, Eiji
Department of Information Systems Interdisciplinary Graduate School of Engineering Sciences, Kyushu University

https://doi.org/10.15017/17511

出版情報：九州大学大学院総合理工学報告. 2 (2), pp.1-16, 1981-01-20. 九州大学大学院総合理工学研究科
バージョン：
権利関係：

COMPREHENSIVE ARTICLE

A Representation of Digital Pictorial Pattern

Tsutomu ENDO* and Eiji KAWAGUCHI**

(Received Oct. 24, 1980)

A new method of representing a binary-valued picture titled "DF-expression" or "DF-coding" is developed, where the picture is decomposed into a set of square regions of various sizes with uniform gray level (black or white). A simple context-free grammar having three terminal symbols "0", "1", and "(" is introduced to specify the gray levels and sizes of such regions. With such grammar, every picture is represented as a terminal string of the grammar. The DF-expression of a picture is defined as the reduced terminal string. It preserves every information for reproducing the original picture in spite of high data compressionability. The coding algorithm of raw pictorial data into DF-expression is very simple. Some type of picture processings, such as shifting, expansions and reductions of original pictures, can be performable on DF-expression itself. Moreover, logical operations are available. After the experimental studies on many test pictures, it was made clear that DF-expression is really useful not only for storing or transmitting pictorial data but also for digital picture processing.

1. Introduction

In the study of pattern recognition, scene analysis and image understanding, we are often troubled with a large amount of data. Even if we attempt to treat only binary patterns with 1024×1024 pixels, a 1 Mbit memory is occupied by a single picture. In order to reduce the amount of storage, pictorial data need to be represented by more compact form using some coding scheme if any.

Today we have many coding schemes for pictorial data compression. Most of them are developed for facsimile data transmission[1]. The major goal of the coding is to reduce the number of bits required to represent the picture, but it is much better if we can perform typical processings such as shiftings, rotations, expansions and reductions of the original picture, on the coded form itself. An example of such coding scheme is the chain code devised by H. Freeman[2]. Another example is the method which reduces original data into short computer commands for the storage of the input picture[3]. However, they do not preserve all information about the original picture, still worse, they are only applicable to line drawings. Recently, we have another type of coding schemes called pyramidal or quad tree representation[4)5)6]. Their point of concern would be the picture processing algorithms on the coded form rather than the data compression.

In this paper, we show a new representation method called "DF-expression" or "DF-coding", in which, (1) every in-

* Present address: Department of Information Science and Systems Engineering, Oita University

** Department of Information Systems

formation is preserved, (2) it is applicable to any pictorial pattern, and (3) several types of picture processings can be performed on the coded form. The idea comes from considering the pyramidal type representation as the generation process of some context-free grammar. But from the viewpoint of data compression in facsimile, DF-expression is the same as the generalized idea of autoadaptive block coding[7][8]. This method is originally developed for binary-valued pictures, but it is easy to apply it to multivalued picture.

2. Definition of DF-expression

Pictures we consider in this paper are all digitized binary (black and white) square pictures, which consist of $2^R \times 2^R = 4^R$ pixels. We associate 0 and 1 with white and black data respectively.

2.1. Fundamental notions and definitions

[Definition 1] Subframes denoted by S_α^r's are sequentially defined by the iterative quartic partitions of the total frame S^0, where $r(r=1, 2, \cdots, R)$ and α specify their orders and addresses respectively as illustrated in Fig. 1. An S_α^r is called a lower subframe of $S_{\alpha'}^{r'}$, if and only if S_α^r contains $S_{\alpha'}^{r'}$, and upper subframe vice versa.

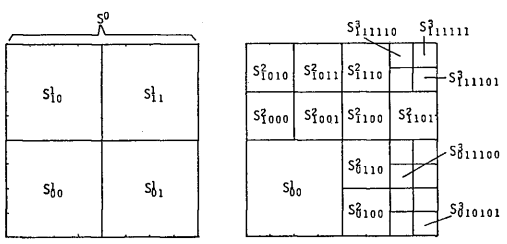


Fig. 1 Picture subframes

[Definition 2] A picture on S^0 is to be denoted by Γ. If a part of Γ is uniformly black or white over a subframe S_α^r, it is said that such S_α^r is an r-th order monotonous subpicture, and is denoted by e_α^r. Consequently, any pixel is an R-th order monotonous subpicture. We call the total of them, i.e., the set of 4^R 0 or 1's, the pictorial raw data. A picture primitive (or primitive) p_α^r of Γ is an e_α^r which has no lower monotonous subpicture. Every e_α^r and p_α^r takes the value 0 or 1, but we put superscript and subscript on each 0 or 1 in the explicit representation of a given monotonous subpicture, e. g., 0^2_{0110}, 1^3_{101011}, and so on.

[Definition 3] The primitive sequence of Γ is the concatenation of all primitives of Γ arranged in their address order. We denote this sequence by $P(\Gamma)$. For Fig. 2(a), it is,

$$P(\Gamma) = 0^2_{0000}\ 1^2_{0001}\ 0^2_{0010}\ 1^2_{0011}\ 0^1_{01}\ 0^2_{1000}\ 1^2_{1001}\ 0^3_{101000}\ 0^3_{101001}\ 0^3_{101010}\ 1^3_{101011}\ 1^2_{1011}\ 1^3_{110000}\ 1^3_{110001}\ 0^3_{110010}\ 0^3_{110011}\ 1^3_{110100}\ 1^3_{110101}\ 0^3_{110110}\ 1^3_{110111}\ 0^3_{111000}\ 0^3_{111001}\ 1^3_{111010}\ 1^3_{111011}\ 0^3_{111100}\ 1^3_{111101}\ 1^3_{111110}\ 1^3_{111111}\ . \quad (1)$$

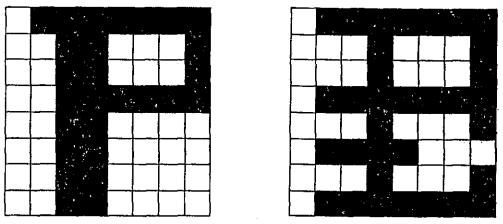
[Definition 4] We define the complexity of Γ as,

$$C_p(\Gamma) = \frac{\text{Total number of primitives of } \Gamma}{4^R}. \quad (2)$$

Apparently, $0 < C_p \leqq 1$. Such Γ as $C_p = 1$ is called a most complicated picture. We denote it by the expression Γ^R. Fig. 2(b) is an example of Γ^R for $R=3$.

[Definition 5] For such a Γ that $e_{\alpha 00}^{r+1} = e_{\alpha 01}^{r+1} = e_{\alpha 10}^{r+1} = e_{\alpha 11}^{r+1}$ does not hold for some r and α, a modification of Γ into Γ', for which $p_\alpha^r = 0$ or $p_\alpha^r = 1$ holds, is called a single simplification of Γ. A picture which is obtained from Γ^R after n successive simplifications is denoted by Γ_n^R. Fig. 2(a) is an illustration of Γ_{11}^3.

It is obvious that any non-most complicated picture is obtained after successive simplifications from a most complicated one.



(a) (b)

Fig. 2 An example of binary picture (a), and a most complicated picture (b)

2.2. Relation between binary pictures and context-free grammar

Let us introduce a context-free grammar (CFG). We focus our attention on the analogy of generation process of a terminal string of the grammar and that of a binary picture on S^0. Let G be a CFG defined as follows:

$G=\{V_N, V_T, P, U\}$

$U=A$: the start symbol

$V_N=\{A\}$: the set of nonterminal symbol

$V_T=\{0, 1, (\}$: the set of terminal symbols

P: production rules

(1) $A\rightarrow 0$ (2) $A\rightarrow 1$ (3) $A\rightarrow(AAAA$

This CFG is interpreted as a binary-picture generative grammar in the following way:

Start symbol A—a total frame S^0

Nonterminal symbol A—subframe S^r_α

Terminal symbols:

0—white subpicture of some order

1—black subpicture of some order

(—order relation among subpictures

Production rules:

(1)—generation of a white subpicture

(2)—generation of a black subpicture

(3)—generation of higher order subframe from S^r_α.

The left most, the next, the third, and the last nonterminals after "(" corresponds to $S^{r+1}_{\alpha 00}$, $S^{r+1}_{\alpha 01}$, $S^{r+1}_{\alpha 10}$, and $S^{r+1}_{\alpha 11}$ respectively. Also, the effective interval of r-th order "(" is its successive four elements of order $r+1$.

A simple example of picture generation by G is illustrated in Fig. 3. Fig. 3(a) is the terminal string, (b) is the derivation tree, and (c) is the binary picture generated. In our discussion, we write derivation tree in quad form. That is, 1) we omit every "(" from the tree, and 2) for such production as $A\rightarrow 0$ and $A\rightarrow 1$, we put 0 and 1 just on the A position. We associate with each node a nonnegative integr called node level which show the order of each subframe.

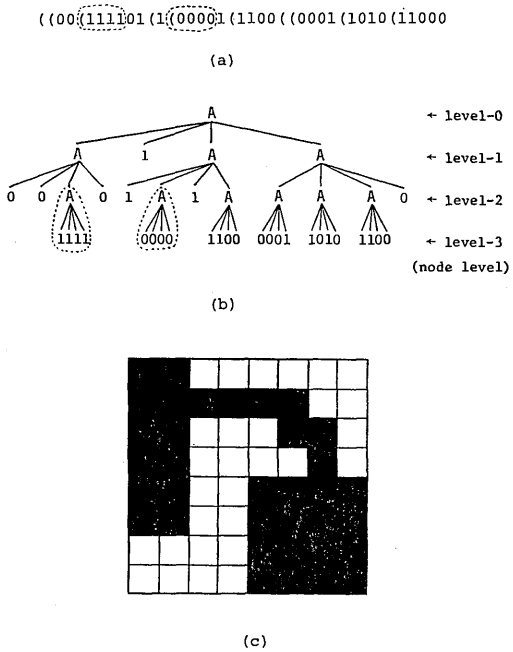


Fig. 3 An example of picture generation by G

If a terminal string generated by G has such substrings as "(0000" and "(1111", we apply to them the reduction rules shown in Fig. 4(a). The corresponding reduction rules for derivation trees are shown in Fig. 4(b). After recursive applications of reduction rules to a given string (or tree), we finally obtain a unique string (or tree). In case of Fig. 3, reduction rules are applicable to two parts which are enclosed by dotted lines.

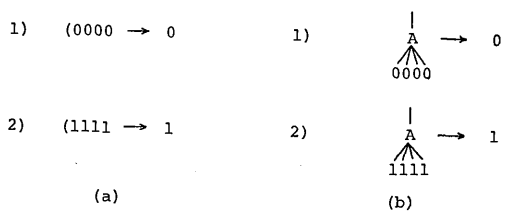


Fig. 4 Reduction rules

The most important property of G is expressed in the next theorem.

[Theorem 1] Let m be the number of "("'s in an arbitrary terminal string of G. Then the total number of "0 or 1" 's in that string equals to $3m+1$. Inversely, any such string w_m that consists of "(" 's, "0" 's, and "1" 's is always generatable by that G if it satisfies the following:

1) w_m has m "(" 's and $3m+1$ "0 or 1" 's,
2) no proper substring w_m', which begins from the top of w_m, has m' "(" 's and $3m'+1$ "0 or 1" 's for some $m'(<m)$.

Proof: It is proved by an induction on m. We use traditional notations in the derivation of terminal string.

1) For $m=0$, the statement is true.
2) Suppose that the statement is true for $m=m'-1$.
3) For $m=m'$;

[the first half of the statement]
The derivation process of a terminal string of G which contains m' "(" 's is as follows:

$$A\overset{*}{\Rightarrow}xAy\Rightarrow x(AAAAy\overset{*}{\Rightarrow}x(a_1a_2a_3a_4y \overset{\text{def}}{=}w_{m'} \tag{3}$$

where $x, y\in\{0,1, (\}^*$, $a_i\in\{0,1\}$. Then a terminal string $w_{m'-1}$ defined by

$$A\overset{*}{\Rightarrow}xAy\Rightarrow x0y\overset{\text{def}}{=}w_{m'-1} \tag{4}$$

consists of $m'-1$ "(" 's and $3(m'-1)+1$ "0 or 1" 's.
Consequently, $w_{m'}$ contains m' "(" 's and $3m'+1$ "0 or 1" 's;

[the last half of the statement]
Let $w_{m'}$ be an arbitrary string consisting of "(" 's, "0" 's, and "1" 's which satisfies two conditions of the statement for $m=m'$. It is easily seen that $w_{m'}$ contains such a substring as "$(a_1a_2a_3a_4$", where a_i="0 or 1". That is,

$$w_{m'}=x(a_1a_2a_3a_4y. \tag{5}$$

If not, according to the second condition on $w_{m'}$, $w_{m'}$ never contains m' "(" 's and $3m'+1$ "0 or 1" 's. Consequently, such $w_{m'}$ is actually generated by G through a derivation process like (3). Q. E. D.

The above theorem suggests that any string that consists of "(" 's, "0" 's and "1"'s is interpretable as a concatenation of binary pictures if it is possible to assume that $R=\infty$. Each picture in the concatenation corresponds to every first substring consisting of m "(" 's and $3m+1$ "0 or 1" 's in the residual string. And if the final residue does not contain m' "(" 's and $3m'+1$ "0 or 1" 's, we can conclude that some illegal data processing or transmission occurred.

In the following discussion, G is supposed to complete its generation process

by R-th order node level. Under this assumption any terminal string of G is interpretable as a binary picture Γ. We denote this string by $\Omega(\Gamma)$.

2.3. Picture expression

If a terminal string $\Omega(\Gamma)$ is given in a nonreduced form, it is easy to transform it into the reduced form through recursive application of reduction rules. It is also possible to expand it into more redundant form, if necessary, by the use of inversed reduction rules. So we can express a Γ in various forms.

[Definition 6] We define the reduced form of $\Omega(\Gamma)$ the DF-picture expression (depth first picture expression) of Γ, or more simply, the DF-expression, and denote it by $\Omega^*(\Gamma)$.

Every "0" and "1" in a DF-expression corresponds to a white and a black primitive in the picture, respectively. Any primitive sequence $P(\Gamma)$ is translatable into DF-expression. For each p_α^z in a $P(\Gamma)$, if $2m$ or $2m+1$ consecutive bits from LSB (or the right most bit) of α are all 0's, we insert m "("s just before the value of p_α^z (i. e., 0 or 1) and delete both superscript and subscript. The inverse translation is also easy. For the picture in Fig. 3, $\Omega^*(\Gamma)$ and $P(\Gamma)$ are;

$$\Omega^*(\Gamma)=((00101(101(1100((0001(1010 \quad (11000 \tag{6}$$

$$P(\Gamma)=0^2_{0000}\,0^2_{0001}\,1^2_{0010}\,0^2_{0011}\,1^1_{01}\,1^2_{1000}\,0^2_{1001}\,1^2_{1010}\,1^3_{101100}\,1^3_{101101}\,0^3_{101110}\,0^3_{101111}\,0^3_{110000}\,0^3_{110001}\,0^3_{110010}\,1^3_{110011}\,1^3_{110100}\,0^3_{110101}\,1^3_{110110}\,0^3_{110111}\,1^3_{111000}\,1^3_{111001}\,0^3_{111010}\,0^3_{111011}\,0^2_{1111} \tag{7}$$

[Definition 7] The length of a DF-expression is defined as the total number of symbols in the $\Omega^*(\Gamma)$ and is expressed as $L(\Gamma)$. Evidently, $L(\Gamma)$ equals to the total number of nodes in the reduced derivation tree.

[Definition 8] The complexity of a DF-expression is defined as

$$C_e(\Gamma)=\frac{L(\Gamma)}{L(\Gamma^R)}=\frac{L(\Gamma)}{1+4+\cdots+4^R}=\frac{L(\Gamma)}{\frac{4^{R+1}-1}{3}}. \tag{8}$$

Consequently, for any Γ, C_e satisfies

$$0<C_e\leqq 1. \tag{9}$$

The relation between C_p and C_e is stated in the next theorem.

[Theorem 2] All pictures with equal complexity C_p have equal complexity of C_e. And for a large 4^R,

$$C_p\fallingdotseq C_e \tag{10}$$

hold for every Γ.

Proof: The total number of primitives in Γ^R_n equals to the total number of "0 or 1"s in $\Omega^*(\Gamma^R_n)$. Every single simplification reduces one "(" and three "0 or 1"s. Then the complexity of Γ^R_n is

$$C_p(\Gamma^R_n)=\frac{4^R-3n}{4^R}. \tag{11}$$

The complexity of DF-expression $\Omega^*(\Gamma^R_n)$ is

$$C_e(\Gamma^R_n)=\frac{\frac{4^{R+1}-1}{3}-4n}{\frac{4^{R+1}-1}{3}}=\frac{4^R-3n-\frac{1}{4}}{4^R-\frac{1}{4}}. \tag{12}$$

Equation (11) means that all pictures with equal complexity come from a Γ^R through equal number of simplifications. Equation (12) shows that the value of $C_e(\Gamma^R_n)$ is also determied only by n. So the first part of the theorem is evident. The latter part is also clear from (11) and (12) for a large 4^R.

Q. E. D.

DF-expression uses three symbols "(", "0", and "1". So the data rate of a DF-expression is $\log_2 3$ bit/symb. Therefore, the total data of a DF-expression Ω^*

(Γ_n^R)は is

$$H=\left(\frac{4^{R+1}-1}{3}-4n\right)\log_2 3$$

$$=\left(\frac{4}{3}\cdot 4^R\cdot C_p-\frac{1}{3}\right)\log_2 3. \qquad (13)$$

As 4^R is usually a large number, (13) is approximated by

$$H\fallingdotseq\frac{4}{3}\log_2 3\cdot 4^R\cdot C_p\fallingdotseq 2.1\cdot 4^R\cdot C_p(\text{bits}). \qquad (14)$$

Let C_{p0} be a special value of C_p for which the right side of (13) equals to 4^R. That is,

$$C_{p0}=\frac{3}{4}\cdot\frac{1}{4^R}\left(4^R\log_2 3+\frac{1}{3}\right). \qquad (15)$$

For a large 4^R, $C_{p0}\fallingdotseq 0.47$. Fig. 5 shows these relations.

These discussions lesd to the next self-evident theorem.

[Theorem 3] Any picture Γ, for which $C_p(\Gamma)<C_{p0}$ holds, is representable by the data less than 4^R bits in terms of $\Omega^*(\Gamma)$.

According to this theorem, effectiveness of DF-expression depends definitely on the value of C_p. We show some actual examples in Section 5.

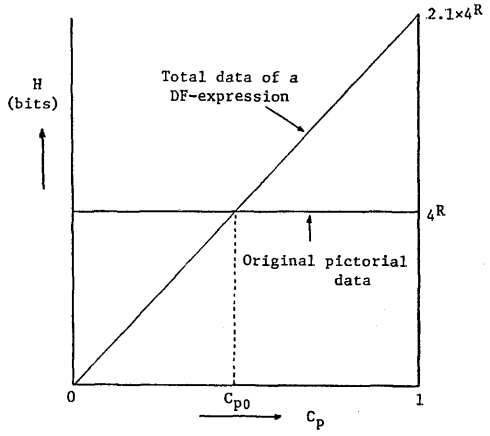


Fig. 5 Relation between C_p and the total data of the corresponding DF expression

3. Coding algorithm

In order to utilize DF-expression in an actual system, an effective coding algorithm is needed. In this Section, we show such an algorithm, named DF-coding algorithm.

A standard I/O devices for pictorial data processing have X and Y coordinates, and the position of each pixel is specified by two binary numbers such as

$$\left.\begin{aligned} x&=a_1a_2\cdots a_r\cdots a^R \quad (a_r=0,1),\\ y&=b_1b_2\cdots b_r\cdots b^R \quad (b_r=0,1).\end{aligned}\right\} \qquad (16)$$

So the pixel address α defined previously and (x,y) is simply related as

$$\alpha=b_1a_1b_2a_2\cdots b_Ra_R. \qquad (17)$$

3.1. DF-encoder

In the DF-coding algorithm we need a device titled DF-encoder which is schematically shown in Fig. 6. This encoder fetches pictorial data from the memory device (ME) and sequentially produces DF-expression from left to right order. We assume that 4^R data are located from address 0 to (4^R-1) on ME. In an actual system we realize this encoder by software. But it is useful to understand its essential operation by an imaginary hardware device. The DF-encoder consists of the following five fundamental units.

1) Reader (RD): This is a data reading unit consisting of address counter (AC) and data reader (DR). DR reads pictorial data from ME, and AC indicates their addresses.

2) Symbol generator (SG): This generates "(", "0", and "1".

3) Q-Register (QR): This is a temporal memory which keeps a queue (or a

string) of "("'s, "0"'s, and "1"'s. We symbolize this queue by Q. Output from QR becomes a prat of DF-expression.

4) Input Controller (IC): According to information from AC and DR, IC controls input symbols into QR.

5) Q-Controller (QC): QC shifts out Q from QR if one of the following conditions is satisfied.

Condition 1: Q contains several "("'s, and the last (or the right most) "(" is followed by the sequence including both "0" and "1".

Condition 2: Q contains no "("'s.

In another circumstance where Q contains "(0000" or "(1111", QC reduces Q by two reduction rules, i. e.,

1) (0000→0••••, 2) (1111→1••••

where • means a space.

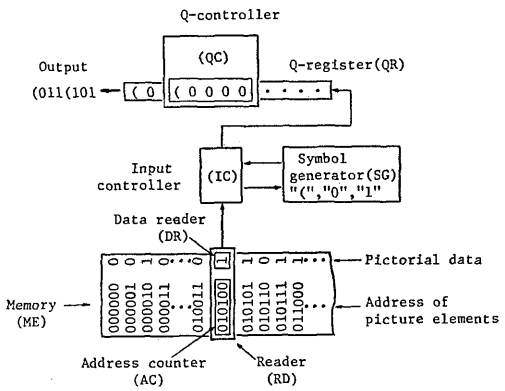


Fig. 6 DF-encoder

3.2. DF-coding algorithm

In this algorithm we use such a convention that the notation $[X]$ means the content of X.

1) Clear QR.

2) Clear AC. That is, set [AC]=000•••0, where 000•••0 is a 2R bit binary number of value 0.

3) Add m "("'s to Q under the control of IC if and only if consecutive $2m$ or $2m+1$ bits from the LSB (or the right most bit) to upper (or left) bits in AC are all 0's.

4) Repeat next a) and b) four times.

a) Add "0" or "1" to Q. If RD reads 0, then "0", and if 1, then "1" must be added.

b) [AC]=[AC]+1 (binary addition).

5) Apply two reduction rules to Q repeatedly.

6) If Q satisfies condition 1 or condition 2, shift left Q out of QR.

7) If [AC]=1000•••0 (the value is 4^R), shift out all Q from QR and halt. If not, go to 3).

We give supplementary explanations about step 3) and 6) in the next.

Step 3): Let α and β be such that

α: a $2R-2m$ bits binary number, whose last two bits are not 11,

β: a binary number of $2m$ 1's.

Step 3) occurs immediately after DR has read the $e^R_{\alpha\beta}$ which is the last datum in S^{R-m}_{α}. At this stage the effective interval of the preceding "(" corresponding to that S^{R-m}_{α} is over. Therefore, another "(" must be inserted. It corresponds to the next subframe.

Step 6): Reduction rules in step 5) are applicable only to the last part of Q. So, if Q satisfies condition 1, there is no possibility that this Q will be reduced in the future. The situation of condition 2 occurs only when Q is reduced into a single "0" or "1". But this symbol is in fact in the effective interval of the preceding "(" which have been already shifted out from QR. Therefore, this symbol will never be reduced again in the later cycles.

Now, let us consider the maximum length of Q in our algorithm. Q becomes the longest at the final stage of step 4) in a certain cycle of DF-algorithm. But it never exceeds $4R+1$[9]. This

means that we do not need a large temporal memory in DF-encoder. For instance, in case of wholly black picture,

$$Q=\underbrace{(111(111(111\cdots(111(1111}_{4R+1}$$

is the longest, which appears just after RD have read the last datum (i. e., $1^{R}_{111\cdots11}$) on ME.

The algorithm of recovering the original pictorial data from DF-expression, that is, DF-decoding algorithm, is also described with similar imaginary hardware device titled DF-encoder[10].

4. Picture processings on DF-expression

In this Section, we describe the several kinds of basic picture processings, or operations, where some are carried out at the level of primitive sequence, and others are on the DF-expression itself.

4.1. Centroid of black pixels

Any "1" in $\Omega^*(\Gamma)$ corresponds to a certain r-th order black primitive which consists of 4^{R-r} black pixels. Accordingly, the coordinates (X_g, Y_g) of the picture centroid in binary numbers are calculated as follows:

Let

$$P(\Gamma)=p^{r_1}_{\alpha_1}p^{r_2}_{\alpha_2}\cdots p^{r_i}_{\alpha_i}\cdots p^{r_n}_{\alpha_n} \qquad (18)$$

be the primitive sequence of Γ. From such α_i (that is, $\alpha_i=b_1a_1b_2a_2\cdots b_{r_i}a_{r_i}$, where b's and a's are 0 or 1's), we can extract two binary numbers; $X_i=a_1a_2\cdots a_{r_i}$ and $Y_i=b_1b_2\cdots b_{r_i}$. Then it is easily seen that

$$X_g=\frac{\sum_{i=1}^{n}\left\{\left(2^{R-r_i}\cdot X_i+\frac{2^{R-r_i}-1}{2}\right)4^{R-r_i}\cdot p^{r_i}_{\alpha_i}\right\}}{\sum_{i=1}^{n}4^{R-r_i}\cdot p^{r_i}_{\alpha_i}}, \qquad (19)$$

$$Y_g=\frac{\sum_{i=1}^{n}\left\{\left(2^{R-r_i}\cdot Y_i+\frac{2^{R-r_i}-1}{2}\right)4^{R-r_i}\cdot p^{r_i}_{\alpha_i}\right\}}{\sum_{i=1}^{n}4^{R-r_i}\cdot p^{r_i}_{\alpha_i}}. \qquad (20)$$

In equation (19) and (20), $2^{R-r_i}\cdot X_i$ and $2^{R-r_i}\cdot Y_i$ are X and Y coordinates of the left most bottom pixel of $p^{r_i}_{\alpha_i}$, and $(2^{R-r_i}-1)$ shows the width of the square. As we see later, the number of primitives is much fewer than the number of total pixels. Therefore, a great deal of computing time will be saved in this method.

4.2. Circular shifting

We shall define circular shifting. The r-th order right circular shifting (r-th right shifting. for short) is the right circular shifting operation of the original picture by the distance of 2^{R-r} pixels. We symbolize this operation by $X^r(\Gamma)$ (cf. Fig. 7(a)). Then k right shifting is recursively defined as;

$$X^{k_n}(\cdots X^{k_2}(X^{k_1}(\Gamma))\cdots),$$

where $k=2^{R-k_1}+2^{R-k_2}+\cdots+2^{R-k_n}$,

and $\quad 0<k_1<k_2<\cdots<k_n\leqq R.$

In the same way, the r-th left, up, and down shiftings are denoted by $X^{-r}(\Gamma)$, $Y^r(\Gamma)$, and $Y^{-r}(\Gamma)$ respectively.

Now we define the r-th (order) subsequence expression of a picture. The r-th sub-sequence expression of a Γ, denoted by $\Omega^r(\Gamma)$, is the sequence of symbols that is obtained from $\Omega^*(\Gamma)$ after applications of inverse reduction rules. It includes all DF-expressions of r-th order subpictures in Γ. That is, let $\omega^r_{\alpha_{100}}$, $\omega^r_{\alpha_{201}}$, $\omega^r_{\alpha_{310}}$, $\cdots$ be DF-expressions of r-th order subpictures, then $\Omega^r(\Gamma)$ is of the form;

$$\Omega^r(\Gamma)=((\cdots(\omega^r_{\alpha_{100}}\omega^r_{\alpha_{101}}\omega^r_{\alpha_{110}}\omega^r_{\alpha_{111}}(\omega^r_{\alpha_{200}}\omega^r_{\alpha_{201}}\omega^r_{\alpha_{210}}\omega^r_{\alpha_{211}}\cdots(\omega^r_{\alpha_{i00}}\omega^r_{\alpha_{i01}}\omega^r_{\alpha_{i10}}\omega^r_{\alpha_{i11}}\cdots(\omega^r_{\alpha_{m00}}\omega^r_{\alpha_{m01}}\omega^r_{\alpha_{m10}}\omega^r_{\alpha_{m11}}, \qquad (21)$$

where $m=4^{r-1}$, and α_i is the binary

number with $2(r-1)$ digits. For example, the 2nd sub-sequence expression of Γ in Fig. 2(a) is:

$$\Omega^2(\Gamma)=((\omega^2_{0000}\,\omega^2_{0001}\,\omega^2_{0010}\,\omega^2_{0011}\,(\omega^2_{0100}\,\omega^2_{0101}\,\omega^2_{0110}\,\omega^2_{0111}\,(\omega^2_{1000}\,\omega^2_{1031}\,\omega^2_{1010}\,\omega^2_{1011}\,(\omega^2_{1100}\,\omega^2_{1101}\,\omega^2_{1110}\,\omega^2_{1111}, \quad (22)$$

where $\omega^2_{0000}=0$ $\omega^2_{0\ 01}=1$ $\omega^2_{0010}=0$ $\omega^2_{0011}=1$
$\omega^2_{0100}=0$ $\omega^2_{0101}=0$ $\omega^2_{0110}=0$ $\omega^2_{0111}=0$
$\omega^2_{1000}=0$ $\omega^2_{1001}=1$ $\omega^2_{1010}=(0001$
$\omega^2_{1011}=1$ $\omega^2_{1100}=(1100$ $\omega^2_{1101}=(1101$
$\omega^2_{1110}=(0011$ $\omega^2_{1111}=(0111.$

Naturally, we have $\Omega^0(\Gamma)=\Omega^*(\Gamma)$ for any Γ. Suppose

$$\Omega^r(\Gamma)=((\cdots(\omega^r_{\alpha_1}\,\omega^r_{\alpha_2}\,\omega^r_{\alpha_3}\,\omega^r_{\alpha_4}\,(\cdots \quad (23)$$

and

$$\Omega^r(X^r(\Gamma))=((\cdots(\omega^r_{\beta_1}\,\omega^r_{\beta_2}\,\omega^r_{\beta_3}\,\omega^r_{\beta_4}\,(\cdots \quad (24)$$

are r-th sub-sequence expression of the original, and the shifted pictures respectively. Since r-th shifting causes no effect to the DF-expression of any sub-picture of order greater than or equal to r, $\Omega^r(X^r(\Gamma))$ is just the sequence that is obtained after rearranging ω^r_α's in $\Omega^r(\Gamma)$. That means, every ω^r_β in $\Omega^r(X^r(\Gamma))$is just the same as some ω^r_α in $\Omega^r(\Gamma)$, and vice versa. So the r-th right shifting operation is a rearranging process of each ω^r_α in $\Omega^r(\Gamma)$. In other words, $\Omega^r(X^r(\Gamma))$ is to be obtained by sequential concatenation of all ω^r_α's in their new address (in terms of β) order, with suitable number of "("s. Therefore, we only need to know which α corresponds to that β. The following is the summary of the algorithm of finding unique α from a given β.

Let $\beta=\beta_1\beta_2\cdots\beta_i\cdots\beta_r$,
where $\beta_i(i=1,2,\cdots,r)=00$, 01, 10, or 11. Then the corresponding $\alpha=\alpha_1\alpha_2\cdots\alpha_i\cdots\alpha_r$ is decided by the following steps.

1) Set $i=r$.
2) If $\beta_i=01$ then $\alpha_i=00$, else if $\beta_i=00$ then $\alpha_i=01$, else if $\beta_i=11$ then $\alpha_i=10$, else if $\beta_i=10$ then $\alpha_i=11$.
3) If $\alpha_i=01$ or 11 then go to 4), else $\alpha_j=\beta_j(j=i-1,i-2,\cdots 2,1)$ and halt.
4) $i=i-1$, and go to 2).

After rearranging $\Omega^r(\Gamma)$ into $\Omega^r(X^r(\Gamma))$ we must apply reduction rules to $\Omega^r(X^r(\Gamma))$, then we get the DF-expression of r-th right shifted picture, i. e., $\Omega^*(X^r(\Gamma))$.

For example, 2nd right shifted form of $\Omega^*(\Gamma)$ for such Γ in Fig. 2(a) is as follows:

$$\Omega^2(X^2(\Gamma))=((\omega^2_{0101}\,\omega^2_{0000}\,\omega^2_{0111}\,\omega^2_{0010}\,(\omega^2_{0001}\,\omega^2_{0100}\,\omega^2_{0011}\,\omega^2_{0110}\,(\omega^2_{1101}\,\omega^2_{1001}\,\omega^2_{1111}\,\omega^2_{1010}\,(\omega^2_{1001}\,\omega^2_{1100}\,\omega^2_{1011}\,\omega^2_{1110}$$
$$=((0000(1010((11010(0111(0001(11001(0011, \quad (25)$$

$$\Omega^*(X^2(\Gamma))=(0(1010((11010(0111(0001(1(11001(0011. \quad (26)$$

This is illustrated in Fig. 7(b).

Other shifting operations such as $X^{-r}(\Gamma)$, $Y^r(\Gamma)$ and $Y^{-r}(\Gamma)$ are also carried out through similar rearranging processes.

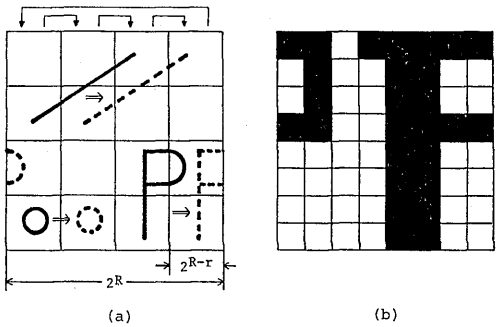


Fig. 7 Circular shifting of picture

4.3. Rotation by multiples of 90° and mirror operation

The picture which is obtained by rotating an original Γ with respect to the center of S^0, by the degree of 90, 180 and 270 counterclockwise, are denoted

by $K^1(\Gamma)$, $K^2(\Gamma)$ and $K^3(\Gamma)$ respectively. Also we give the notation $M(\Gamma)$ to the picture transformed through the mirror operation about Y axis. As primitives are subpictures in square frame, so the shape of primitives after the rotation or mirror operation remains unchanged. These operations are another types of rearranging process. For example, $K^1(\Gamma)$ is obtained as follows.

1) We transform the address expression α of p_α^r in $P(\Gamma)$ to the address expression β of p_β^r in $P(K^1(\Gamma))$. Let $\alpha=\alpha_1\alpha_2\cdots\alpha_i\cdots\alpha_r$ and $\beta=\beta_1\beta_2\cdots\beta_i\cdots\beta_r$, where $\alpha_i, \beta_i=00$, 01, 10, or 11. The relationship between α_i and β_i is shown in Table 1.

2) We rearrange every p_β^r obtained in step 1), in order of address β. The result is $P(K^1(\Gamma))$.

3) We translate $P(K^1(\Gamma)$ to $\Omega^*(K^1(\Gamma))$.

$K^2(\Gamma)$, $K^3(\Gamma)$ and $M(\Gamma)$ are accomplished in the same manner. The address relations are also listed in Table 1.

Table 1 Address transformation for rotation and mirror operation

$K^1(\Gamma)$		$K^2(\Gamma)$		$K^3(\Gamma)$		$M(\Gamma)$	
α_i	β_i	α_i	β_i	α_i	β_i	α_i	β_i
00	01	00	11	00	10	00	01
01	11	01	10	01	00	01	00
10	00	10	01	10	11	10	11
11	10	11	00	11	01	11	10

4.4. Expansion and reduction

The expansion by the order r is the expanding operation of original picture by the scale factor 2^r with central point of Γ being fixed. We represent this operation by $Z^r(\Gamma)$, where $r=1, 2, 3\cdots$. In the meantime, the reduction of order r is the inverse operation of Γ by the factor $1/2^r$, and is denoted by $Z^{-r}(\Gamma)$. These operations are performed by the combination of circular shiftings and simplifications. The procedure for $Z^r(\Gamma)$ is as follows.

1) We repeat left-down shiftings of Γ until the original center of Γ moves to the central position of $S^r_{00\cdots00}$. Let Γ' be such shifted picture.

2) Then we regard $S^r_{00\cdots00}$ as the total frame. Let $\Omega^r(\Gamma')=((\cdots(\omega^r_{\alpha100}\ \omega^r_{\alpha101}\ \omega^r_{\alpha110}\ \omega^r_{\alpha111}\cdots$ be the r-th sub-sequence expression of Γ'. The sub-sequence $\omega^r_{\alpha100}$ in $\Omega^r(\Gamma')$ is the DF-expression of $Z^r(\Gamma)$.

The procedure of obtaining $Z^{-r}(\Gamma)$ is as follows.

1) Let $\Omega^{R-r}(\Gamma)$ be the $(R-r)$-th subsequence expression of Γ. We replace every $\omega^{R-r}_{\alpha_i r}$ in $\Omega^{R-r}(\Gamma)$ with 0 or 1. This is a simplification of the original picture. In this case, we should execute such simplification in such a manner as to preserve the shape of original picture as much as possible. Also we apply reduction rules to this expression. The resultant is denoted by $\Omega^*(\Gamma')$.

2) We add r "("'s to the left side of $\Omega^*(\Gamma')$ and $3r$ "0"'s to the right. The picture corresponding to this DF-expression is denoted by Γ''. Γ'' is the reduced picture by the factor $1/2^r$, but its location is still in $S^r_{00\cdots00}$.

3) We repeat right-up shiftings of Γ'' until the center of the reduced picture moves to the center of S^0. The resultant picture is the $Z^{-r}(\Gamma)$.

4.5. Logical operations on DF-expression

We can produce new pictures by applying logical operations to each pixel value of original pictures. The logical operations on DF-expession are defined as follows:

(1) AND $\Omega^*(\Gamma_1)\cdot\Omega^*(\Gamma_2)=\Omega^*(\Gamma_1\cdot\Gamma_2)$,

(2) OR $\Omega^*(\Gamma_1)+\Omega^*(\Gamma_2)=\Omega^*(\Gamma_1+\Gamma_2)$,

(3) NOT $\overline{\Omega^*(\Gamma)}=\Omega^*(\overline{\Gamma})$.

These three operations are interpreted as (1) intersection, (2) union, and (3) complementation of pictures.

To execute logical operations on DF-expression directly, we now define the following operations on a symbol, or pair of symbols in DF-expressions. Operations must be carried out from leftto right order on the strings.

"0" • "0"="0", "1" • "0"="0", "1" • "1"="1", "0"+"0"="0", "1"+"0"="1",
"1"+"1"="1", "(" • "("="(", "("+"("="(",
"(" • "1"="(" [in this case, four "1"'s must be inserted after "1"],
"(" • "0"="0"
"(" + "1"="1" } [in this case, all symbols in the effective interval of "(" are neglected],
"(" + "0"="(" [in this case, four "0"'s must be inserted after "0"],
"$\bar{0}$"="1", "$\bar{1}$"="0", "$\bar{(}$"="(".

The operations • and + are commutative. After these operations, we finally apply the reduction rules to get the DF-expression of the new picture.

The logical operation is very useful when we apply DF-expression to picture series or gray images. Outline of such applications will be shown later.

5. Experimental studies

We have performed several kinds of experiments to examine the effectiveness of DF-expression in view of data compression.

5.1. Experimental apparatus and test pictures

The apparatus used in this experiment is PDS (Photo Digitizing System)-200 pictorial data processing system controlled by TOSBAC-40C mini-computer as in Fig. 8. Test pictures are (a) weather chart, (b) circuit diagram, (c) prints, (d) crest patterns, (e) bold-faced characters, and (f) English text as shown in Fig. 9. These pictures were all 20 cm by 20 cm size, and were placed in front of PDS camera 85 cm apart from its lens. The illumination on the picture surface was 3800 lx and the diaphragm of the camera was 5.6. Analog images were digitized into binary pictures with 1024 1024 pixels (that is, $R=10$) by thresholding analog data from image dissector. These values were determined in considerations of each set of data being not influenced by the image quality (e. g., each background of picture may not be uniform, or the inking of the black areas may not be uniform, etc.). In this system every pixel is randomly accessible.

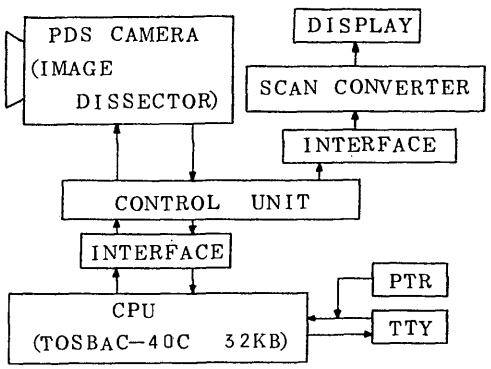
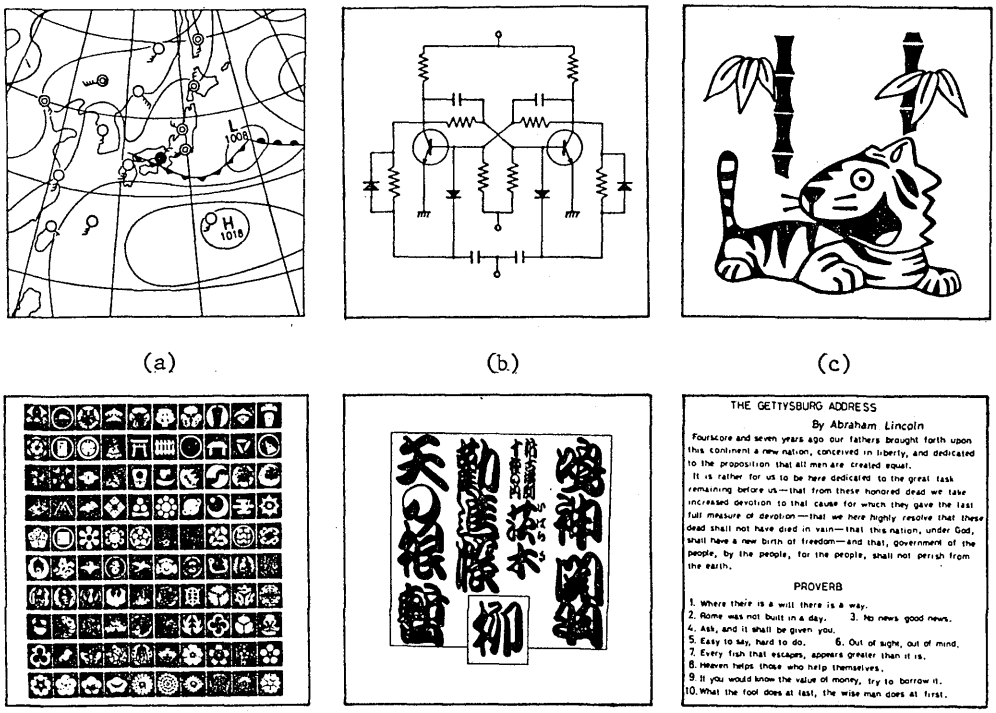


Fig. 8 Block diagram of the PDS

5.2. Picture complexity and spectrum of primitives

The number of r-th primitives (for $r=3, 4, \cdots, 10$) and the picture complexity C_p for test pictures were calculated in order to know the statistical properties of



(a)　(b)　(c)

(d)　(e)　(f)

Fig. 9 Test pictures

Table 2 Complexity and number of primitives

r \ Γ	(a)		(b)		(c)		(d)		(e)		(f)	
	W	B	W	B	W	B	W	B	W	B	W	B
3	0	0	15	0	9	0	0	0	17	0	2	0
4	20	0	57	0	48	1	15	0	22	0	31	0
5	228	0	229	0	144	17	91	2	104	6	117	0
6	1004	1	497	1	554	184	107	316	497	288	1162	0
7	3037	23	1267	34	1530	1030	1146	3252	1262	1447	2898	0
8	7252	875	3036	1120	3546	3520	5591	10058	3397	3413	7623	548
9	15966	14562	5604	5802	7230	7305	23109	26259	8633	8522	17600	14193
10	13960	13960	5988	5988	5794	5794	27144	27144	7690	7690	14086	14086
Total	41467	29421	16693	12945	18855	17851	57203	67031	21622	21366	43519	28827
Pixel	0.916	0.084	0.953	0.047	0.785	0.215	0.443	0.557	0.744	0.256	0.924	0.076
C_p	0.068		0.028		0.035		0.118		0.041		0.069	

binary pictorial data. The results are shown in Table 2. The white primitive is predominant except only one case (test picture (d)). Generally such predominance at primitive level is not so remarkable as that of pixel level.

Complexities of all test pictures were less than $C_{p0}(\fallingdotseq 0.47)$. So, we became convinced that even an appparently very complicated picture will still have less complexity than C_{p0}. For example, even a weather chart on a very noisy

background and a pattern of char-acters that was printed very closely by line-printer, were found to be less complica ted than C_{p0}. With our preliminary experimental studies, we have concluded that DF-expression is a good method for data compression.

Complexity of a picture gives us good global information, but it does not tell us anything about sizes of predominant primitives. If a picture contains many low order primitives, the picture will consist of many areas. While, if not, it will consist of many curves and points. Taking account of these features, we introduced the notion of spectrum of primitives. Let m_w^r and m_b^r be the number of r-th order white and black primitives respectively. The spectrum of primitives is the set of such r-th order components of primitives as follows:

$$Q_w^r=4^{r-}\cdot m_w^r,\quad Q_b^r=4^{-r}\cdot m_b^r,\quad Q^r=Q_w^r+Q_b^r.$$

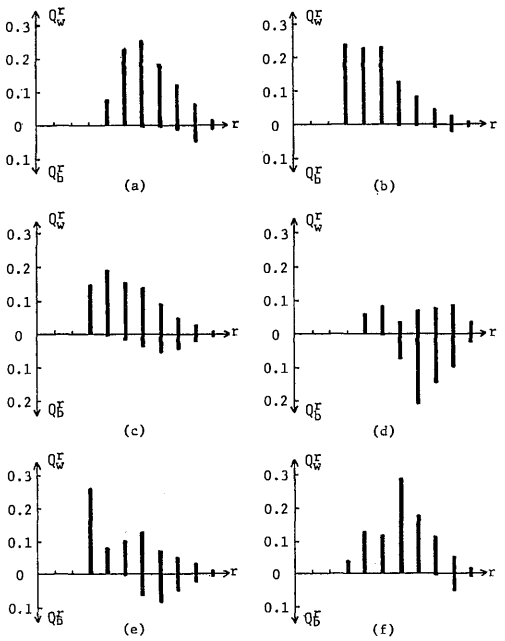


Fig. 10 Spectrum of primitives

The spectrums of test pictures are demonstrated in Fig. 10. It is clear that spectrum patterns vary according to the class of pictures. For example, the significant differences are observed between (a) and (f), but both complexities nearly equal to each other. Also we have made certain that various pictures of a same class (e. g. weather chart) have some similar spectrum patterns. Therefore, the spectrum will serve as the feature for picture classification and as the index for pictorial database.

5.3. Coding method and compression ratio

DF-expression uses three symbols "(", "0" and "1". We must code them with 0 and 1. In our study, the following three coding systems were tested.

(2) "(" → 11, "0" → 00, "1" → 01.
(2) "(" → 11, "0" → 0, "1" → 10.
(3) "(" → 11, "0" → 0,

$$\text{"1"} \rightarrow \begin{cases} 1 & \text{(if "1" specifies the 10-th order primitive.)} \\ 10 & \text{(otherwise.)} \end{cases}$$

(1) is the most simple and consntat length but apparently redundant system. (2) comes from the Theorem 1 and the estimation that "0" is the most frequent symbol in an ordinary DF-expression. (3) is based upon the fact that only "0" and "1" appear in the effective interval of an R-th (in our case, $R=10$) order "(".

The program of DF-coding algorithm was written in the assembly language of TOSBAC-40C. The compression ratio is defined by

$$\eta=\frac{\text{Total number of pixels } (1024\times1024)}{\text{Number of code bits}}. \qquad (27)$$

The compression ratios of test pictures

are listed in Table 3. These results are very satisfactory. Especially, for (b) and (c), markedly high ratios were obtained because of the predominancy of lower order primitives.

In order to compare the efficiency of DF-expression with typical facsimile bandwidth compression methods, we also performed three facsimile simulations using the same data as our DF-coding methods. The results are all listed in Table 3, where A, B, and C are the followings.

(A) Block coding[11].

(B) Two-dimensional prediction[12].

(C) Mode run-length coding[13].

We summarize these experiments as in the following statements.

(1) The compression ratio depends on the statistical properties of pictorial data. For example, since facsimile method (A) is based on the estimation that the white pixel is predominant in the picture, so its ratio becomes worse when a picture contains many black pixels. In case of (C), the ratios for such pictures as (a) and (f) are rather worse because the mode transitions are very frequent. While, DF-expression (1) is not influenced by which (black or white) pixel is predominant in the picture.

(2) The compression ratio, both in facsimile methods and obviously in our method, tends to decrease with increasing picture complexity in our sense. It will be concluded that the complexity in our sense is a good measure of the complicatedness of pictures from the standpoint of data compression.

(3) The ratio of (B) is highest in facsimile methods because the strong two-dimensional correlations are observed among pixels. Incidentally, DF-expression is a generalized two-dimensional run-length coding, and its ratio (3) is about 60 % better than (B) on the average.

Table 3 Compression ratio

Γ	DF-expression			Facsimile method		
	(1)	(2)	(3)	A	B	C
(a)	5.55	7.11	7.85	3.51	4.86	2.87
(b)	13.27	16.82	18.61	5.46	11.93	6.69
(c)	10.71	13.27	14.32	2.61	8.44	4.87
(d)	3.17	3.83	4.25	1.11	2.59	1.44
(e)	9.15	11.27	12.29	2.26	7.38	3.96
(f)	5.44	7.02	7.75	3.38	4.97	2.94

5.4. Application to picture series

A set of pictures, or a picture series, is coded into a set of respective DF-expressions in the usual manner. Furthermore, we may be able to get more compact form according to the next operation.

Let $\Gamma_3=\Gamma_1\oplus\Gamma_2$ be the picture that is produced by EOR (exclusive OR) operation of Γ_1 and Γ_2. Then $\Gamma_1=\Gamma_2\oplus\Gamma_3$ and $\Gamma_2=\Gamma_1\oplus\Gamma_3$ holds. EOR operation on DF-expression is defined as;

$$\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2)=\overline{\Omega^*(\Gamma_1)}\cdot\Omega^*(\Gamma_2)+\Omega^*(\Gamma_1)\cdot\overline{\Omega^*(\Gamma_2)}=\Omega^*(\Gamma_1\oplus\Gamma_2).$$

Then we also have $\Omega^*(\Gamma_1)=\Omega^*(\Gamma_2)\oplus(\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2))$ and $\Omega^*(\Gamma_2)=\Omega^*(\Gamma_1)\oplus(\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2))$. This can be said as follows. When we store the data of a picture series $\{\Gamma_1,\Gamma_2\}$ in some database, we are allowed to store $\{\Omega^*(\Gamma_1),\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2)\}$ or $\{\Omega^*(\Gamma_2),\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2)\}$ instead of $\{\Omega^*(\Gamma_1),\Omega^*(\Gamma_2)\}$. Especially, if Γ_1 and Γ_2 are similar to each other, $\Omega^*(\Gamma_1)\oplus\Omega^*(\Gamma_2)$ will be more compact than $\Omega^*(\Gamma_1)$ or $\Omega^*(\Gamma_2)$. We illustrate a simple example of picture series in Fig. 11. The effect of EOR operation in this case proved to reduce 50 % data of the ordinary coded DF-ex-

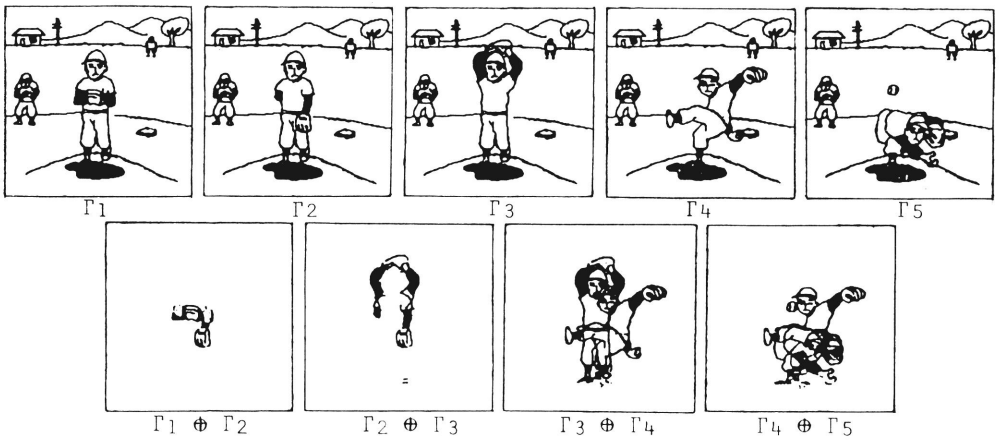


Fig. 11 An example of EOR operation on a picture series

pression.

5.5. Application to multi-valued picture

So far, we have restricted our discussion to binary-valued pictures. But we can easily remove such restriction. The outline of the coding method of multi-valued pictures by DF-expression is as follows. Let there be 2^k bits possible values in each pixel. Then we are to have k bit-sliced binary patterns. By applying DF-coding method to such k binary patterns, we can represent every multi-valued picture as a set of k DF-expressions. Still more, EOR operation may be worth trying.

This method was tested for 4 and 8 gray level pictures with 256 by 256 pixels. Table 4 shows the resulted compression ratio by three coding systems. We decoded these expressions into original pictures, and displayed them on a storage type graphic display terminal, where the cell size (the number of dots per pixel) was 4 by 4. They are shown in Fig. 12.

Table 4 Compression ratio of multi-valued picture

Level	(1)	(2)	(3)
2	3.79	4.52	5.39
4	1.59	1.90	2.30
8	1.03	1.25	1,56

(a) 4 level

(b) 8 level

Fig. 12 Decoded multi-valued pictures

6. Conclusions

The authors have presented DF-expression and demonstrated its effectiveness in data compression. This method has the following desirable advantages:

(1) It is applicable to multi-valued pictures as well as binary-valued pictures.

(2) It preserves every information for the reconstruction of the original picture.

(3) It yields much higher compression than traditional methods which are often found in facsimile transmission.

(4) The coding algorithm of raw data into DF-expression is very symple, and it is executed recursively by an ordinary processor if the pictorial data are randomly accessible.

(5) Some typical picture processings, such as circular shiftings, rotations by multiples of 90°, expansions and reductions of a original picture, can be performed on DF-expression itself. This is useful for further picture processings and flexible display of pictures.

(6) The logical operations of original pictures can be performed on it. This is applicable to the representation of a picture series.

(7) Such concepts as a picture complexity and a spectrum of primitives, are defined based on DF-expression. They will serve as the features for picture processing and the indexes for pictorial information retrieval system[14].

We are convinced that DF-expression is effective and efficient representation of pictorial data not only for the construction of a pictorial database but also for several types of picture processings.

References

1) T. S. Huang, "Coding of Two-Tone Images", IEEE Trans. on Comm., vol. COM-25, No. 11, pp. 1406-1424, Nov. 1977.

2) H. Freeman, "On the Encoding of Arbitrary Geometric Configurations", IRE Trans. on Elec. Comput., vol. EC-10, No. 2, pp. 260-268, June 1961.

3) S. Murakami, M. Okada and T. Tsuchiya, "A Study on a Method of Inputting Line Drawings", Trans. IECE Japan, vol. J59-D, No. 2, pp. 117-124, Feb. 1976.

4) A. Klinger and C. R. Dyer, "Experiments on Picture Representation Using Regular Decomposition", Computer Graphics and Image processing, vol. 5, pp. 68-105, March 1976.

5) T. Ichikawa, "A Pyramidal Representation of Images", Proc. 4th IJCPR, pp. 603-606.

6) G. M. Hunter and K. Steiglitz, "Operations on Images Using Quad Trees", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, No. 2, pp. 145-153, April 1979.

7) F. de Coulon and O. Jhonsen, "Adaptive Block Scheme for Source Coding of Black-and-white Facsimile", Electronics Letters, vol. 12, pp. 61-62, Feb 1976.

8) M. Kunt and O. Jhonsen, "Codes de blocs a prefixe pour la reduction de redondance d'images", Ann. Telecommunic., vol. 33, No. 7-8, p. 61, July 1976.

9) E. Kawaguchi and T. Endo, "On a Method of Binary Picture Representation and Its Application to Data compression", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, No. 1, pp. 27-35, Janu. 1980.

10) T. Endo and E. Kawaguchi, "Some Properties of DF-expression of Binary Pictures", Tech. Report of Kyushu Univ., vol. 51, No. 2, pp. 147-154, March 1978.

11) K. Kubota, "Facsimile", J. IECE Japan, vol. 57, No. 11, pp. 1370-1376, Nov. 1974.

12) M. Takagi and T. Tsuda, "Band-width Compression for Facsimile using Twodimensional Prediction", Trans. IECE Japan, vol. 56-D, No. 3, pp. 170-177, March 1973.

13) T. Ishiguro, "Facsimile", J. IECE Japan, vol. 59, No. 11, pp. 1224-1230, Nov. 1976.

14) T. Endo, E. Kawaguchi and T. Tamati, "DF-coding and Weather Chart Database", National Convention Record of IECE Japan, S19-10, March 1980.