



|              |                                                                                   |
|--------------|-----------------------------------------------------------------------------------|
| Title        | 選択画素による画像の符号化および位相的な特徴の抽出                                                         |
| Author(s)    | 森田, 啓義                                                                            |
| Citation     | 大阪大学, 1983, 博士論文                                                                  |
| Version Type | VoR                                                                               |
| URL          | <a href="https://hdl.handle.net/11094/2213">https://hdl.handle.net/11094/2213</a> |
| rights       |                                                                                   |
| Note         |                                                                                   |

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

選択画素による画像の  
符号化および位相的な特徴の抽出

1983年2月

森 田 啓 義

## 要 旨

本論文では、デジタル白黒2値画像およびデジタル濃淡画像のそれぞれに適した符号化方式を提案する。デジタル2値画像の符号化では、従来多くの符号化方式がラスタ走査方向に色の变化する変化点を選び出し、変化点の位置の統計的性質に基づいて構成された符号を用いて符号化を行っている。この方法では、統計的性質を調べるための前処理や、性質の異なる画像ごとにそれに適した符号を用意する必要がある。一方2値画像でも特に文字や漢字あるいは機械部品の設計図などの幾何学的なパターンから構成されるものに対しては、画面内の物体の配置や、どの点とどの点が直線で結ばれているあるいは結ばれていないといった幾何学的な特徴が画像の情報として重要な意味をもっている。そこで画面から幾何学的特徴を有する点を選び出せば、単に変化点を選び出す場合に比べ、選出される点の数も少なく、後に続く選出された点を符号化する場合にも、変化点の場合に比べ、より構成の簡単な符号の使用が期待できる。この見地に立ち、幾何学的な特徴点として画像のコーナー点に対応する選択画素を定義し、さらにできるだけ少ない選択画素によって画像を一意に表現する理論を展開し選択画素符

号化方式を導出した。さらに選択画素から知りうる元の画像の位相的な性質について調べ、一旦画面から取り出しておいた選択画素の情報のみから、元の画像に含まれる連結成分とホールの検出や周囲長・面積を計測する方法について論じる。

一方、デジタル濃淡画像の符号化における問題の一つに、濃淡レベル数がある程度以上ある場合には、画像を直接符号化することがハードウェア上非常に困難になることが挙げられる。この問題を解決するために、元の画像を正確に復元するという制約のもとで、濃淡画像を複数個の2値情報源に効率よく還元する Bit-Plane 変換を提案し、この変換を用いて濃淡画像を符号化する方法について論じる。

## 謝 辞

本研究を遂行するにあたり終始御指導いただいた本学機械工学科教授有本卓博士に深く感謝します。さらに機械工学科教授林卓夫博士，制御工学科教授辻三郎博士，生物工学科教授鈴木良次博士，そして機械工学科助教授門田良実博士に感謝します。また選択画素の定義に関して有意義な御助言をいただいた金沢工業大学教授林枕博士に感謝します。そして有本研究室の橋本猛助手，生物工学科の小林欣吾助手の二人には，常日頃から適切な助言と励しを受けていました。ここにそれを記すとともに厚く感謝します。さらに宮崎文夫助手を始めとする有本研究室の皆様感謝します。最後に研究に協力していただいた，北島浩司氏，樋野文夫氏，向井信彦氏，古市浩朗氏，そして北田茂氏に感謝します。特に最後の二人には編集段階において並々な御助力を受けました。改めてこの二人に感謝します。

# 目 次

## 第1章 序論 1

## 第2章 準備 5

- 2.1 画像の入力と記述 5
- 2.2 デジタル2値画像における用語の定義 5
- 2.3 符号化について 7

## 第3章 デジタル2値画像の符号化 11

- 3.1 問題の記述と歴史 11
- 3.2 2値画像の $(I, R)$ -表現 19
- 3.3 2値画像の $(i, r, C)$ -表現 29
- 3.4 選択画素の右変形と左変形 43

## 第4章 漢字パターンのデータ圧縮 55

## 第5章 選択画素の画像処理への応用 65

- 5.1 連結性の定義とC点の取扱い方 66
- 5.2 連結成分とホールの検出 73
  - 5.2.1 問題の記述と歴史 73
  - 5.2.2  $(i, r, C)$ -表現からの連結成分とホールの同時検出 76
- 5.3 連結成分およびホールの周囲長と面積の計算 93

## 第6章 デジタル濃淡画像の符号化 99

### 6.1 Bit-Plane変換 99

#### 6.1.1 Bit-Plane変換とは 99

#### 6.1.2 問題の定式化 103

#### 6.1.3 指数分布の場合 106

#### 6.1.4 デジタル濃淡画像に対する Bit-Plane 変換 111

### 6.2 算術符号化方式の符号化効率について 117

#### 6.2.1 問題と歴史 117

#### 6.2.2 算術符号の構成 119

#### 6.2.3 算術符号の符号化効率 128

#### 6.2.4 算術符号の応用 132

#### 6.2.5 他の算術符号との比較 137

## 第7章 まとめ 141

### 付録1 SECT アセンブラ・プログラム 143

### 付録2 論文で提案した算術符号のFORTRAN によるプログラム・リスト 153

### 参考文献 163





# 第1章

## 序 論

本論文では画像の符号化を白黒2値画像の場合と濃淡画像の場合とに分けて、それぞれに適した符号化方式について論じる。

まず最初に、2値画像の符号化について考察する。2値画像の中でも符号化の対象として特に重要なものに、言語処理システムで欠かせない漢字や文字あるいは計算機援助システムによる製図・設計で扱われる機械部品の設計図やLSI回路の配線図などが挙げられる。これらの2値画像の符号化は画像の特徴抽出と密接にかかわっている。というのはこれらの画像においては、画面内の物体の配置やどの点とどの点が直線で結ぶ結ばれないの関係にあるのかといった幾何学的特徴が画像の情報として重大な意味をもっているからである。

この観点から従来の2値画像の符号化方式を見直してみると、第3章でも指摘するように実に多くの方式が一種の特徴抽出を行っていることが分る。しかしながら従来の方式では、どういう特徴点をどう選出するかという問題には深く立ち入らないで、選出後の特徴点を効率よく符号化することが主眼点であった。と

ころがこれらの方法にはいくつかの問題点が含まれている。すなわち、特徴点を符号化するための処理が複雑になればなるほど処理の必然性があいまいになってゆく点や、画像の統計的性質を調べる前処理を行う必要がある点、そして性質の異なる画像ごとにそれに適した符号を用意しなければならないという点である。そこでこれらの問題点を解決するために従来の方式とは異なり、選り出す特徴点自体をできるだけ少なくする方向から2値画像の符号化を考察する。

第3章ではまず新しい特徴点として画像のコーナ点に対応した選択画素を定める。次に選択画素によって画像を一意に表現する方法について議論し、新しい符号化方式すなわち選択画素符号化方式 (Selective Element Coding Technique, 略してSECT) を導出する。

SECTによる符号化例として漢字パターンのデータ圧縮を第4章において行い、さらにSECTの有効性を他の方式との比較において論じる。第5章では画像の特徴点としての選択画素から知ることのできる元の画像の位相的な性質について調べ、画面全体ではなく選択画素の情報のみから、元の画像に含まれる連結成分とホール検出ならびに連結成分の周囲長・面積の計測を行う方法について論じる。この方法は処理形式からいうと画面を一定順序で1回走査するだけですべての連結成分あるいはホールの計測を行うことのできる、いわゆる1パス走査方式になっている。

一方、濃淡画像の符号化における問題の一つに、濃淡レベル数がある程度以上ある場合には、画像を直接符号化することがハードウェア上非常に困難になるということが挙げられる。そこでハードウェア規模の小型化、処理の高速化を図るために、復元画像と元の画像の間にある程度の誤差を許して符号器への入力となるシンボル数を削減することが従来行われてきたが、第6章ではこの方法とは異なり、元の画像を正確に復元するという制約のもとで、濃淡画像データを複数個の2値情報源に効率よく還元する Bit-Plane 変換を提案し、さらにこの変換を用いて濃淡画像を符号化する方法について論じる。

さて次の第2章ではデジタル画像の表現方法と使用する語句、そして符号化に関する一般的な知見について説明する。

なお本文の第3章、第5章におけるアルゴリズムは文献[73]に基づく ALGOL 流の記述がなされている。



## 第 2 章

### 準 備

#### 2.1 画像の入力と記述

本論文で取扱う画像とは、実世界の物体がTVカメラや光センサなどによって電気信号に変換され、さらにA/Dコンバータにより標本化ならびに量子化を行ったものである。このような処理を施した画像を一般に、デジタル画像と呼ぶ。このデジタル画像は、数学的には大きさ $M \times N$ の行列 $P$ によって表現される。これより以後では、デジタル画像とその行列表現 $P$ を同一視する。直積集合 $\{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$ を画像 $P$ の画面といい、画面内に含まれる要素 $(i, j)$ 、 $1 \leq i \leq M$ 、 $1 \leq j \leq N$ を画素と呼ぶ。さらに $P$ の要素 $p(i, j)$ を画素 $(i, j)$ の画素値と呼ぶ。 $p(i, j)$ の値が0と1に限る行列 $P$ を特にデジタル2値画像という。デジタル2値画像は、A/D変換の際にあるしきい値によって作られた白黒2値画像に相当し、0が白、1が黒に対応しているとみなす。 $p(i, j)$ のとり値の種類が2より大きな $P$ をデジタル濃淡画像と呼ぶ。

#### 2.2 デジタル2値画像における用語の定義

以下の章では各種の処理を簡単に記述するために、

Pの上下左右に1行1列ずつ、値が0の要素を付け加えた大きさ  $(M+2) \times (N+2)$  の行列  $P_{ex}$  をしばしば用いる。このとき  $P_{ex}$  の画面は  $\{0, 1, \dots, M+1\} \times \{0, 1, \dots, N+1\}$  である。また明らかに

$$P_{ex}(i, 0) = P_{ex}(i, M+1) = 0, \quad 0 \leq i \leq M+1$$

$$P_{ex}(0, j) = P_{ex}(M+1, j) = 0, \quad 0 \leq j \leq N+1$$

$$P_{ex}(i, j) = p(i, j), \quad 1 \leq i \leq M, \quad 1 \leq j \leq N$$

である。 $P_{ex}$  において画素  $(i, j)$ ,  $0 \leq i \leq M+1$ ,  $1 \leq j \leq N+1$  が左隣りの画素  $(i, j-1)$  と異なる値を持つとき,  $(i, j)$  を 変化点 と呼ぶ。さらに  $(i, j)$  が

$$P_{ex}(i, j) = P_{ex}(i-1, j-1)$$

$$P_{ex}(i, j) = 1 - P_{ex}(i-1, j)$$

$$P_{ex}(i, j) = 1 - P_{ex}(i, j-1)$$

を満たすとき,  $(i, j)$  を 交差点 と呼ぶ。また同じ値をとる変化点が縦に一列に並んだもの, そして真上の画素と異なる値をとる画素で同じ値のものが横に一列に並んだものを画像Pの辺あるいは境界と呼ぶ。また  $P_{ex}$  の各行において同じ値の画素が一列に並んだものを, その値が0ならば 0-ラン, 1ならば 1-ラン と呼ぶ。

以後では特に混乱が生じない限り  $P_{ex}$  の要素は  $p(i, j)$  と略して記すことにする。また画素  $(i_1, j_1)$  と  $(i_2, j_2)$ ,  $0 \leq i_1, i_2 \leq M+1, 0 \leq j_1, j_2 \leq N+1$  の間には

$$(i_1, j_1) \leq (i_2, j_2) \Leftrightarrow \begin{cases} i_1 < i_2 & \text{あるいは} \\ i_1 = i_2 & \text{かつ } j_1 \leq j_2 \end{cases}$$

の意味での辞書式順序を与え、画素の位置関係に大小比較を許すものとする。

### 2.3 符号化について

標準TV信号やTVカメラから得られた画像データをそのままデジタル化すると膨大な量のデータが発生する。このため画像をデジタル画像として計算機内の記憶装置に記憶する場合、あるいは電波や回線ケーブルなどを用いて伝送する場合には、一意に復元が可能な変換でしかも変換後のデータを減少しうるものをデジタル画像に施すことによって、占有記憶容量の削減あるいは回線使用時間の短縮が行われる。これらの処理を称してデジタル画像の符号化（またはデータ圧縮）と呼ぶ。

符号化を行うシステムは、図2.1に示すように符号器と復号器とから構成される。ここで符号器は情報源から出力されるシンボル列を符号語（符号化データ）へ変換する。復号器は符号語から元のシンボル列を復元する。復元に際しては誤りを全く許さずに元のシンボル列を正確に復元する場合と、許容範囲内の誤りを許して復元を行う場合が考えられる。本論文では前者の場合についてのみ議論する。

符号化の効率を議論する場合、情報源は一般にある確率過程としてモデル化される。このとき情報源からの長さ $N$ の出力シンボル列はある確率変数列となり、

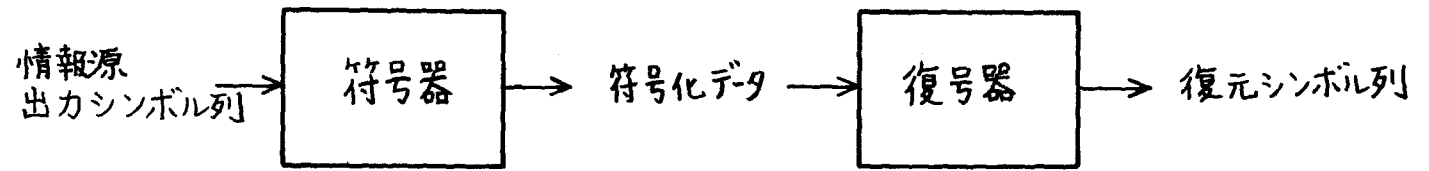


図 2.1 符号化の概略図



これに対応する符号語の長さ  $L$  も一般に確率変数となる。このとき符号化の効率  $R$  は圧縮率

$$R = \frac{EL}{N} \quad (2.1)$$

によって測られる。ここで  $EL$  は  $L$  の期待値、すなわち平均符号長を意味する。

適当な符号器と復号器を用いることによって  $R$  をどこまで下げることができるといふ問題は、確率モデルに対応して定まるエントロピー

$$H^N = \sum_{x^N: \text{長さ } N \text{ のシンボル列}} p(x^N) \log_2 p(x^N)^{-1} \text{ (ビット)} \quad (2.2)$$

ここで  $p(x^N)$  はシンボル列  $x^N$  が生起する確率と密接に関係する。Shannon[1]とFano[2]は  $EL$  を  $H^N$  より少なくすることができるといふことは存在しないことと、次の不等式を満足する符号が存在することを証明した。

$$H^N \leq EL \leq H^N + 1 \quad (2.3)$$

そして Huffman[3]は最適な符号を構成するアルゴリズムを求めた。以来、彼のアルゴリズムに基づいて構成されるいわゆるハフマン符号は広く応用されるに至っている。しかしながらハフマン符号を構成するためには、シンボル列の長さ  $N$  の指数乗オーダの計算手数が必要となるので、画像データなどで確率パラメータが変動する場合や実時間処理には不向きである。これらの状況に適した符号化方法については第6章において

詳しく議論する。

## 第3章

### ディジタル2値画像の符号化

#### 3.1 問題の記述と歴史

ディジタル2値画像の符号化は、主に、ファクシミリ通信分野において盛んに研究され、すでに数多くの符号化方式が提案されている[4][5]。これらの方式には一つの共通した特徴がある。それは図3.1のように画面を左から右、上から下へラスタ走査を行いながら変化点を検出するという点である。検出された変化点の情報から元の画像は正確に復元できるので、これらの符号化方式で問題となるのは変化点の情報をできるだけ効率よく符号化することである。実際、変化点の符号化をどのように行うかを主題として2値画像の符号化技術は発展してきたといっても過言ではない。この節では、その中でも特に代表的な符号化方式のいくつかを紹介し、問題点を明らかにする。

ランレングス符号化方式：歴史的に一番古いものとしては、隣り合う変化点間の距離(ランレングス)によって各変化点の位置を表わすランレングス符号化方式[6]がある。ランレングス符号化方式は画像に限らず、一般のデータ伝送においても幅広く用いられており[7][8]、後述の各種符号化方式の基礎となる方式である。

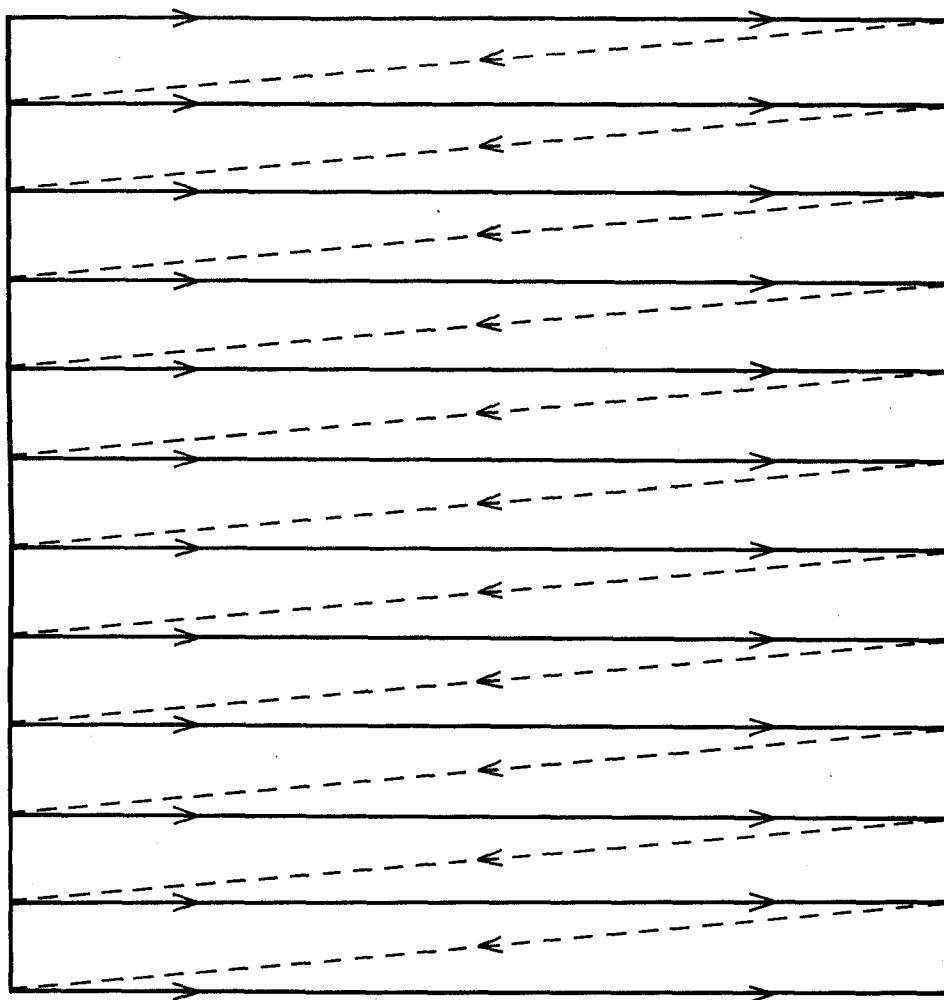


図3.1 ラスタ走査形式

このランレングス符号化方式では、あらかじめランレングスの頻度分布を調べ、分布にもっとも適した符号（ハフマン符号）を構成しておく。そして、ランレングスを対応する符号語に変換する。また、場合によっては、白のランと黒のランの分布が異なることを考慮して、より圧縮効果を上げるために白ランと黒ランとに別々の符号を使い分けることが行なわれる[9][10]。

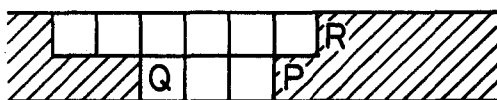
予測符号化方式：予測符号化方式の符号器は予測器とランレングス符号を直列に結合した構成からなる。予測器では、すでに走査した過去の画素値から現在走査中の画素値の予測を行なう[1][2][3]。予測の結果が正しいければ予測器は0を、誤まっていれば1を出力する。このとき、うまく予測器を構成しておけば0の頻度に較べ1の頻度は遙かに少なくなる。その結果、白ランでは長いランが、黒ランでは短かいランが多く表われる。すなわち、予測器出力列のランレングスは元の画像のランレングスに比べて偏った分布に従う。一般に、確率分布が偏ればよるほど適当な符号を用いて符号化データの平均量を少なくすることができる。したがって、ランレングス符号化を行う前に予測処理を施すことにより、より効率の良い符号化が可能となる。復号器では、最初にランレングスの復元を行ない予測器出力列を求め、次に符号器で用いた予測器を駆動して、いま得られた出力列から元の画像データを復元する(図3.2参照)。

再整列化方式：予測符号化方式にさらに手を加えたものに、再整列化（Reordering）操作を行う方式がある[4][5]。この再整列化方式では1走査ラインごとに画素値をまとめて次の処理を行う。まず予測器の入力を予測結果がある確率以上で的中する入力とそうでない入力とに分ける。次に、前者に対する予測器の出力はラインの左端から、後者に対する予測器の出力はラインの右端から整列し直してゆく。このとき双方からの出力が会する位置（境界点）を付加情報として記憶しておく。再整列化後のデータには、境界点より左側は非常に白っぽく、境界点より右側は白黒が入り混じるという空間的な偏りが生じる。したがって境界点より左側と右側でそれぞれに適したランレングス符号を用いて符号化すれば、単なる予測符号化方式より以上の圧縮効果を得ることが出来る。この再整列化方式の一般化として予測器の各入力ごとに予測器出力を整列し直し、それぞれに最適な符号を用いてランレングス符号化を行うという方式も提案されている[16][17]。

RACおよびEDIC方式：今まで紹介してきた符号化方式では、変化点位置を符号化するためにランレングスが用いられてきた。ランレングスは白あるいは黒の画素の連なりの長さであるが、詳しく言うと、走査中の変化点に対し直前に検出された変化点を基準点として、基準点から測った現在走査中の変化点までの相対距離である。このランレングスの概念を一般化し



**图 3.2**



$$\overline{PQ} = 3 \quad \overline{PR} < \overline{PQ} \text{ より } \overline{PR} \text{ を符号化する。}$$

The diagram shows a two-layer system. The top layer is divided into three segments labeled  $S_1$ ,  $S_2$ , and  $S_3$ . Below these, the top layer is further divided into six segments labeled  $T_1$  through  $T_6$ . The bottom layer is also divided into six segments. The first segment of the bottom layer is labeled  $d$ . The second segment is labeled  $l_1$ . The third segment is labeled  $l_2$ . The segments  $T_1$  through  $T_6$  are represented by different patterns:  $T_1$  and  $T_2$  are white,  $T_3$  and  $T_4$  are hatched,  $T_5$  is white, and  $T_6$  is hatched. The bottom layer segments are also represented by different patterns: the first is white, the second is hatched, the third is white, and the fourth is hatched.

$$S_1(+2) \cdot S_2 \cdot S_3(1, 2)$$

### (b) EDIC方式の構成

**图 3.3**

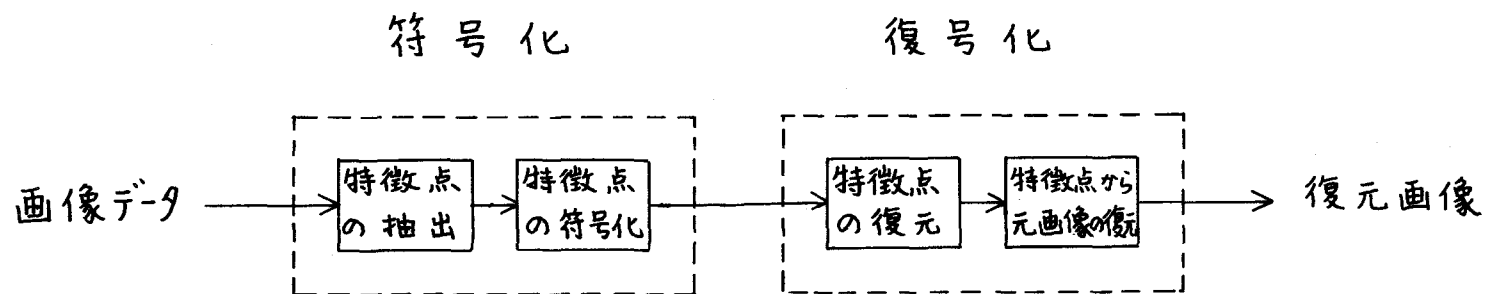
て、現在走査中の変化点からの相対距離ができるだけ短くなるように、前走査ライン上の変化点をも含めて基準点となる変化点を選び分ける符号化方式が開発されている。その代表として変化点相対アドレス (RAC) 方式[18]、境界差分符号化 (EDIC) 方式[19]などがある。RAC方式では、現在走査中の変化点Pを次のように符号化する。まずPとその直前に走査された変化点Qとの距離 $\overline{PQ}$ を求めると同時に、前走査ライン上においてPと同種の変化点の中でQより右側にありQに最も近い変化点Rを捜し出してPからの距離 $\overline{PR}$ を求める。次にこれらの距離を比較して短かい方を、いずれを採用したかを示す符号とともに符号化する (図3.3a 参照)。EDIC方式では連続した2走査ライン上で相続いて現われる二つの変化点を対とする。そしてこの対を変化点が各走査ラインに1個ずつ存在する場合 $S_1$ 、両方とも前走査ラインに存在する場合 $S_2$ 、両方とも現走査ラインに存在する場合 $S_3$ に分類する。変化点对が $S_1$ であるときは、 $S_1$ を示す符号とともに変化点間の距離 $d$ を符号化する。 $S_2$ であるときは $S_2$ を示す符号のみを符号化する。最後に $S_3$ であるときは直前の変化点对と現在の変化点对との距離 $l_1$ と現在の変化点对に含まれる変化点間の距離 $l_2$ とを符号化する (図3.3b 参照)。さらに、RAC方式とEDIC方式を組合せた符号化方式としてModified READ方式[20]が考案され実用化されている[21]。



上で紹介したこれらの符号化方式には幾つかの問題点が含まれている。すなわち、処理が複雑になるにつれて処理の必然性がいまいちになってゆく点、ランレングス分布などの画像の統計的性質を調べるための前処理を行なう必要がある点、そして、性質の異なる画像ごとにそれに適した符号を用意しなければならないという点である。そこで、これらの問題点を改善するために新しい符号化方式の検討を行おう。

まず、2値画像の符号化を図3.4のように図式化して解釈する。すなわち、符号器は2値画像からある特徴点を抽出する部分と抽出された特徴点を符号化する部分から構成され、復号器は符号化データから特徴点を抽出する部分と復元した特徴点から元の2値画像を正確に再生する部分から構成されると考える。上で紹介した各種方式においては、特徴点として変化点を抽出し、変化点を如何に符号化するかに独自の工夫を凝らしているという具合に解釈できる。この構成に沿ってさらに考えると、特徴点の符号化を工夫する前にできるだけ抽出する特徴点の数を少なくすることがデータ圧縮効果をより高めるために、また、後続く特徴点の符号化を簡単化するために必要であると思われる。

もっとも直観的で理解しやすい特徴点の候補としては、画面に描かれた図形の“角”(コーナ点)を挙げることができる。実際、任意の2値画像は描かれた図形のコーナ点によって一意に特徴づけられる。さら



画像符号化の構成

図 3.4

に少し考えると，すべてのコーナ－点を抽出する必要はなく，図 3.5 のように一つおきにコーナ－点をコーナ－の方向とともに抽出すれば，残りのコーナ－点の位置と方向は抽出したコーナ－点の情報から同定できるように思われる。尚，コーナ－点の方向としては四種類を考えておけば十分なので，コーナ－の方向の情報は高々 2 ビットで符号化できる。もしこの方法でうまくゆくなれば，特徴点の数をコーナ－点（あるいは辺）の数の半分にまで抑えることができるので非常に効率的である。しかしながら，この方法は一般性に欠けている。例えば，図 3.6 に示すように一つの画面内に複数の図形が描かれている画像に対しては，右側と左側の図形に対し抽出されるコーナ－点の方向と位置は一致するので，もとの画像の正確な復元は不可能となる。そこで，幾つのコーナ－点を抽出すればコーナ－点の情報からもとの画像を正確に復元できるかという問題が生じる。この章の以下では，この問題に一つの解答を与えることを目的として考察を行なってゆく。そして，さらに得られる知見をもとにして新しい 2 値画像の符号化方式を提案する。

### 3.2 2 値画像の (I, R) - 表現

この節では，コーナ－点に相当する選択画素 I 点と R 点を導入して，I 点と R 点によって任意の 2 値画像が表現できることを復元アルゴリズムを与えることに

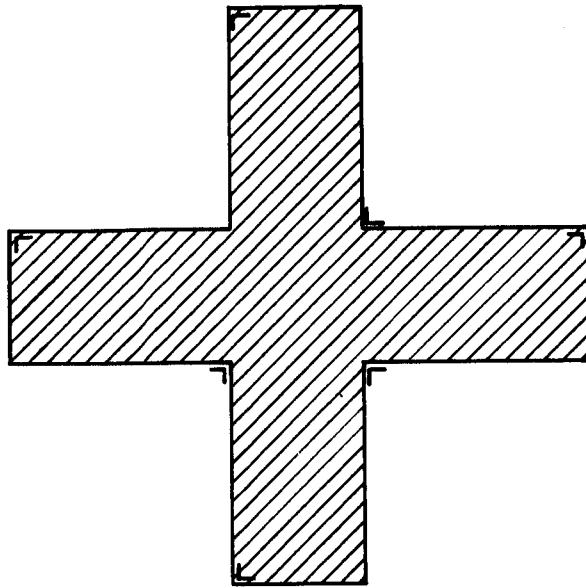


図 3.5 一つおきにコーナー点を抽出する方法

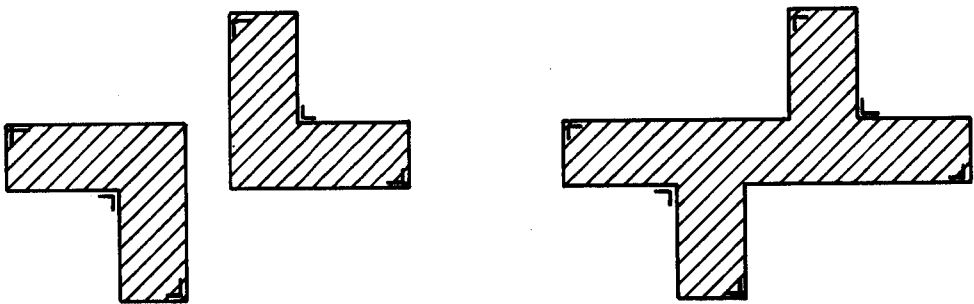


図 3.6 図 3.5 に示した抽出法では  
うまく抽出できない例

よって示す。I点とR点の定義は次のとおりである。

[定義 3.1]

2値画像 $P_x$ の区間 $\{(i,j) | 1 \leq i \leq M, 1 \leq j \leq N+1\}$ 内の画素に対し、I点とR点を次のように定義する。

1) 画素 $(i,j)$ がI点であるための必要十分条件は、

$$p(i-1, j-1) = p(i, j-1) \quad \text{と}$$

$$p(i-1, j) \neq p(i, j)$$

がともに成立することである。

2) 画素 $(i,j)$ がR点であるための必要十分条件は、

$$p(i-1, j-1) \neq p(i, j-1) \quad \text{と}$$

$$p(i-1, j) = p(i, j)$$

がともに成立することである。

I点とR点の例を図3.7に示す。I点とR点を称して選択画素と名付ける。さらに、変化点であるI点(R点)をT I点(T R点)、変化点でないI点(R点)をC I点(C R点)と呼ぶことにする。例からも明らかのように、I点とR点はコーナー点に対応している。I点とR点に関する主な性質を次にまとめておく。

[定理 3.1]

- 1)  $(i,j)$  が T I 点ならば、 $(i-1,j)$  は変化点でない。
- 2)  $(i,j)$  が C I 点ならば、 $(i-1,j)$  は変化点である。
- 3)  $(i,j)$  が T R 点ならば、 $(i-1,j)$  は変化点でない。
- 4)  $(i,j)$  が C R 点ならば、 $(i-1,j)$  は変化点である。

(証明)

各場合について次の等式が成立する。

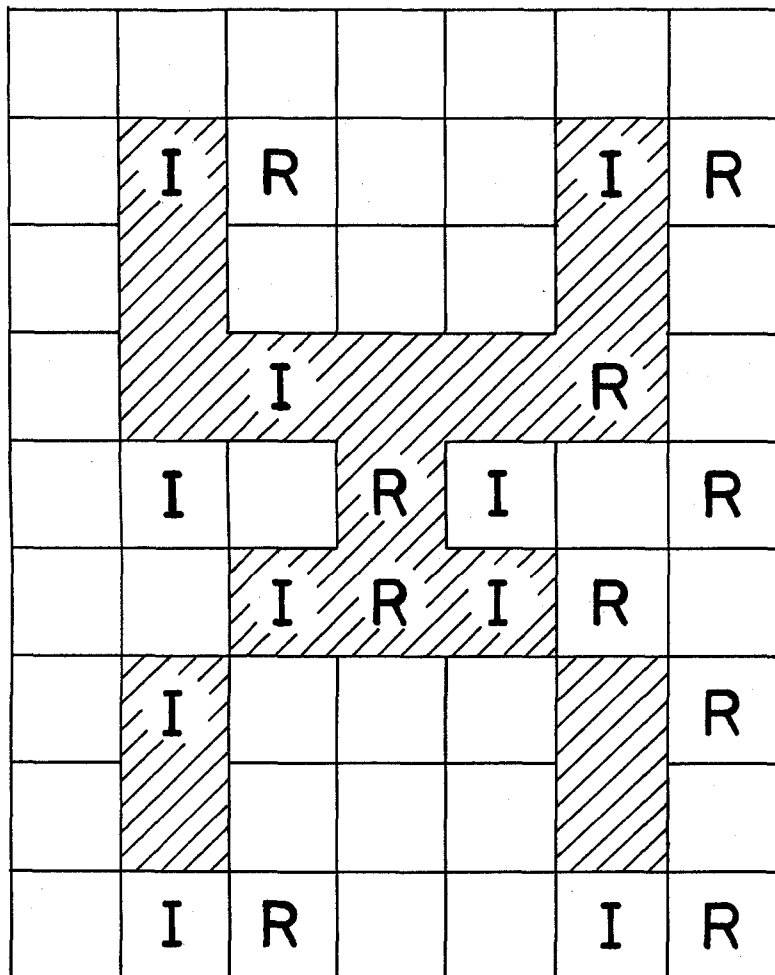


図 3.7 I 点と R 点の例

$$\begin{aligned}
 1) \text{ の場合: } & p(i-1, j-1) = p(i, j-1) \\
 & (a) = 1 - p(i, j) \\
 & (b) = p(i-1, j) \tag{1}
 \end{aligned}$$

$$\begin{aligned}
 2) \text{ の場合: } & p(i-1, j-1) = p(i, j-1) \\
 & (c) = p(i, j) \\
 & (d) = 1 - p(i-1, j) \tag{2}
 \end{aligned}$$

$$\begin{aligned}
 3) \text{ の場合: } & p(i-1, j-1) = 1 - p(i, j-1) \\
 & (e) = p(i, j) \\
 & (f) = p(i-1, j) \tag{3}
 \end{aligned}$$

$$\begin{aligned}
 4) \text{ の場合: } & p(i-1, j-1) = 1 - p(i, j-1) \\
 & (g) = 1 - p(i, j) \\
 & (h) = 1 - p(i-1, j) \tag{4}
 \end{aligned}$$

ここで、(1)と(2)の最初の等式および(b)と(d)はI点の定義から、また、(3)と(4)の最初の等式および(f)と(h)はR点の定義から導かれる。さらに、(a)と(e)は(i, j)が変化点であることから、また、(c)と(g)は(i, j)が変化点でないことから導かれる。(1)(2)(3)(4)は定理の主張が正しいことを示している。(証明終り)

[定理 3.2]

2値画像 $P_x$ の*i*行に選択画素が存在すると仮定する。このとき、*i*行の選択画素について次の主張が成立する。

- 1) 一番左端の選択画素はI点である。
- 2) 一番右端の選択画素はR点である。
- 3) I点とR点は同数あり、しかも、必ず交互に位置す

る。

(証明)

1)  $i$  行で一番左端の選択画素を  $(i, j_L)$  とおく。  $(i, j_L)$  が第 1 列に存在したとする (すなわち,  $j_L = 1$ )。このとき,  $P_{ex}$  の定義から  $p(i, 0) = p(i-1, 0) = 0$  なので,  $(i, 1)$  が選択画素であるならば定義から I 点に他ならない。次に,  $2 \leq j_L \leq N+1$  の場合を考える。もし,  $(i, j_L)$  が R 点であると仮定すると, R 点の定義から,  $p(i, j_L-1) \neq p(i-1, j_L-1)$  が成立しなければならない。ところで,  $p(i, 0) = p(i-1, 0)$  なので, 第 1 列から第  $j_L-1$  列の間には少なくとも一つの I 点が存在しなければならない。これは,  $(i, j_L)$  が  $i$  行上で一番左端の選択画素であるという前提に矛盾する。したがって,  $(i, j_L)$  は常に I 点である。

2)  $i$  行で一番右端の選択画素を  $(i, j_R)$  とおく。  $(i, j_R)$  が第  $N+1$  列に存在したとする ( $j_R = N+1$ )。このとき,  $P_{ex}$  の定義から  $p(i, N+1) = p(i-1, N+1) = 0$  なので,  $(i, N+1)$  が選択画素であるならば定義から R 点に他ならない。次に,  $1 \leq j_R \leq N$  とおく。さて,  $(i, j_R)$  が I 点であると仮定すると, I 点の定義から  $p(i, j_R) \neq p(i-1, j_R)$  が成立する。ところで,  $p(i, N+1) = p(i-1, N+1) = 0$  なので, 第  $j_R+1$  列から  $N+1$  列の間には少なくとも一つの R 点が存在しなければならない。これは,  $(i, j_R)$  が  $i$  行上において一番右端の選択画素であるという前提に矛盾する。したがって,  $(i, j_R)$  は常に R 点である。



3)  $i$  行上には少なくとも二つの選択画素が存在することを 1) と 2) の結果は示している。そこで、相続く選択画素  $(i, j_1)$  と  $(i, j_2)$ ,  $j_1 < j_2$  が隣接している ( $j_2 = j_1 + 1$ ) 場合をまず考える。もし,  $p(i, j_1) = p(i-1, j_1)$  ならば定義 1 から  $(i, j_1)$  は R 点,  $(i, j_2)$  は I 点である。また,  $p(i, j_1) \neq p(i-1, j_1)$  ならば  $(i, j_1)$  は I 点,  $(i, j_2)$  は R 点である。そこで,  $j_2 > j_1 + 1$  とおく。  $(i, j_1)$ ,  $(i, j_2)$  とともに R 点であれば,  $p(i, j_1) = p(i-1, j_1)$  かつ  $p(i, j_2-1) \neq p(i-1, j_2-1)$  が成立するので,  $j_1 + 1$  列から  $j_2 - 1$  列までの間に少なくとも一つの I 点が存在する。これは  $(i, j_1)$  と  $(i, j_2)$  の間に選択画素が存在しないという前提に矛盾する。また,  $(i, j_1)$  と  $(i, j_2)$  が I 点であるとすれば,  $j_1 + 1$  列から  $j_2 - 1$  列までの間に少なくとも一つの R 点が存在することが導びかれ, やはり矛盾が生じる。したがって, 相続く選択画素は異なる種類のものである。I 点と R 点の個数が同じであることは, 左端の選択画素が I 点, 右端の選択画素が R 点であり, しかも, I 点と R 点が交互に位置することから自明である。

(証明終り)

画像  $P_{ex}$  に  $K$  ( $K$  は偶数) 個の I 点と R 点が存在するとき, これらを大きさの順に並べた系列  $D_p = \{(i_n, j_n), n = 1, \dots, K\}$  を画像  $P$  の (I, R)-表現 と呼ぶ。(I, R)-表現からもとの 2 値画像  $P$  は次のアルゴリズム 1 をもちいて復元できる。

[アルゴリズム 1]

```

begin
1   Q ← 0 - matrix of the size ( M + 1 ) x ( N + 2 ) ;
2   for n ← 0 until K do
3       if n is even then DECODE((In,Jn),(In+1,Jn+1- 1),0)
4           else DECODE((In,Jn),(In+1,Jn+1- 1),1) ;
5   return(Q)
end

procedure DECODE((IS,JS),(IE,JE),FLAG)
1   if FLAG = 0 then for (u,v) ← (IS,JS) until (IE,JE) do q(u,v) ← q(u-1,v)
2       else for (u,v) ← (IS,JS) until (IE,JE) do q(u,v) ← 1-q(u-1,v)

```

### アルゴリズム 1

(I, R) - 表現からの元の画像の復元

図 3.8

入力：ある2値画像Pの(I,R)-表現 $D_P$ を入力とする。

出力：復元画像 $P_x$

方法：大きさ $(M+1) \times (N+2)$ の2次元配列 $Q$ と $(I_0, J_0) = (1, 0)$ ,  $(I_{K+1}, J_{K+1}) = (M, N+1)$ なる定数を使用し、画像の復元を部分区間 $[(I_n, J_n), (I_{n+1}, J_{n+1}-1)]$ ,  $0 \leq n \leq K$ ごとに手続きDECODEをもちいて順次行なっていく。アルゴリズムを図3.8に示す。

[補助定理 3.1]

2値画像Pの $i-1$ 行( $1 \leq i \leq M$ )の画素値と $i$ 行のI点とR点の位置が与えられれば、 $i$ 行の画素値はアルゴリズム1から正確に復元される。

(証明)

アルゴリズム1によって復元された $i$ 行の画素値を $g(i, j)$ ,  $0 \leq j \leq N+1$ とおく。 $i$ 行上の選択画素を $(I_n, J_n)$ ,  $i = I_n$ とすると、定理3.1から $n$ が奇数ならば $(I_n, J_n)$ はI点であり、偶数ならばR点である。一方、手続きDECODEの実行によって、 $n$ が奇数ならば、 $g(i, J_n) \neq g(i-1, J_n)$ 、また $g(i, J_n-1)$ は直前の区間 $[(I_{n-1}, J_{n-1}), (I_n, J_n-1)]$ において復元され、 $n-1$ が偶数なので $g(i, J_n-1) = g(i-1, J_n-1)$ である。同様の議論から、 $n$ が偶数ならば、 $g(i, J_n) = g(i-1, J_n)$ ,  $g(i, J_n-1) \neq g(i-1, J_n-1)$ が成立する。したがって、 $(I_n, J_n)$ は復元画像においても同種の選択画素である。さて、復元された画素値がPの $i$ 行の値と異なっていると仮定しよう。 $j^* = \min \{j \mid p(i, j) \neq g(i, j)\}$ とおく。したがって、 $p(i, j) = g(i, j)$ ,

$0 \leq j \leq j^* - 1$ 。もし、 $p(i-1, j^*-1) = p(i, j^*-1)$  ならば、画素  $(i, j^*)$  はもとの画像と復元された画像のいずれか一方において I 点であり他方においては選択画素ではない。ところで、もとの画像において  $(i, j^*)$  が I 点であることはありえない。なぜなら、もし I 点であるとすれば、ある  $n$  が存在して、 $(i, j^*) = (I_n, J_n)$  が成立する。これは、 $(i, j^*)$  が復元画像においても I 点であることを意味し、矛盾が生じるからである。したがって、 $(i, j^*)$  はもとの画像では選択画素でなく、復元画像において I 点でなければならない。しかし、これもありえない。なぜなら、 $(i, j^*)$  はもとの画像では選択画素でないのだから、ある  $m$  が存在して、 $(I_m, J_m) < (i, j^*) \leq (I_{m+1}, J_{m+1} - 1)$  が成立する。DECODE の実行により、 $m$  が偶数なら、

$$g(i, j^*-1) = g(i-1, j^*-1) \text{ かつ } g(i, j^*) = g(i-1, j^*)$$

奇数なら、

$$g(i, j^*-1) \neq g(i-1, j^*-1) \text{ かつ } g(i, j^*) \neq g(i-1, j^*)$$

が成立し、いずれの場合においても、 $(i, j^*)$  は復元画像において I 点ではありえないからである。また、もし、 $p(i-1, j^*-1) \neq p(i, j^*-1)$  ならば、画素  $(i, j^*)$  はもとの画像と復元された画像のいずれか一方において R 点であり他方においては選択画素ではない。この場合にも上と同じ議論で矛盾が示されるので、結局、 $p(i, j) = g(i, j)$ ,  $0 \leq j \leq N+1$  が成立する。 (証明終り)

### [定理 3.3]

ある  $(I, R)$ -表現  $D_P$  に対してアルゴリズム 1 をもちいることによって、もとの 2 値画像  $P$  は正確に復元される。

(証明)

$P_x$  の第 0 行とアルゴリズム 1 の  $Q$  の第 0 行はともに 0-ベクトルで等しいので、 $i = 1$  において補助定理 3.1 から  $P_x$  と  $Q$  の第 1 行は等しいことが導かれる。以下、補助定理をくり返し用いることにより、 $Q$  が  $P$  に完全に一致することが示される。(証明終り)

### 3.3 2 値画像の $(i, r, C)$ -表現

$(I, R)$ -表現は 2 値画像の表現としては十分すぎる。すなわち、選択画素の定義を少し変更することによって、画像を表現する選択画素の数を画像に含まれるコーナ点の個数、あるいは同じことだが辺の数より確実に少なくすることができる。このことを示すために、新しい選択画素  $i$  点、 $r$  点および  $C$  点を次のように定める。

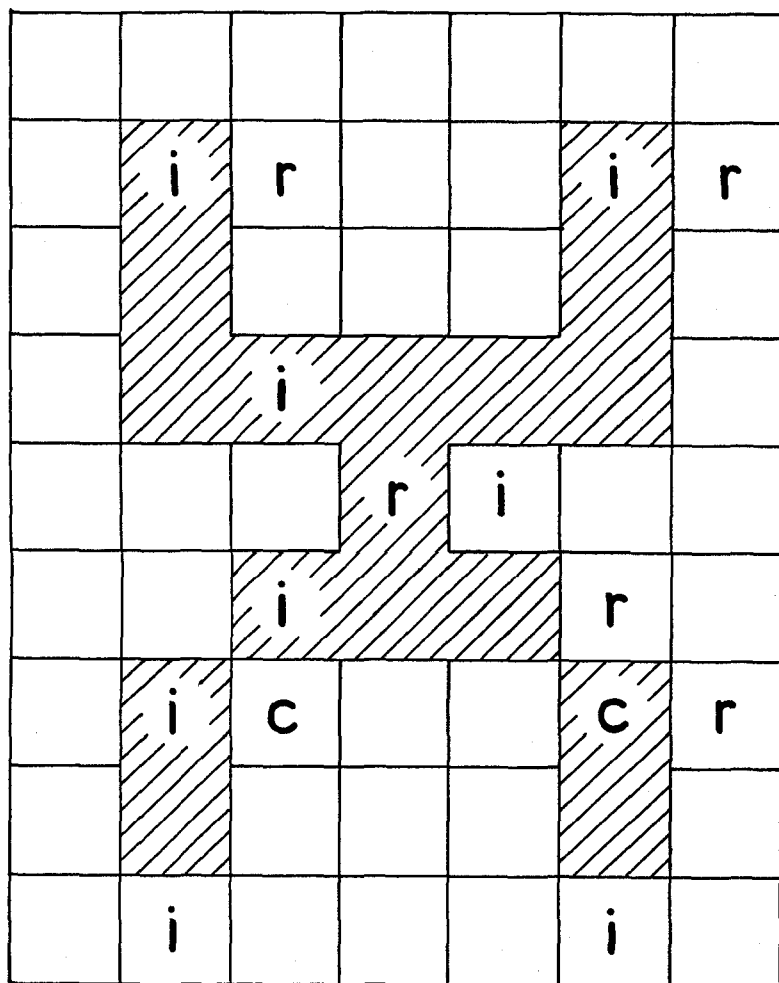
#### [定義 3.2]

i) 画素  $(i, j)$  が  $i$  点であるための必要十分は、次の i) または ii) の条件のいずれか一方が成立することである。

i)  $(i, j)$  は  $T$   $I$  点である。

ii) 次の 3 条件が同時に成立する。

a)  $(i, j)$  は  $C$   $I$  点である。



i 点, r 点, C 点の例

図 3.9

b)  $j^+ = \min\{k \mid (i, k) \text{ は } R \text{ 点, } t > j\}$  に対して,  
 $(i, j^+)$  は CR 点である。

c)  $(i, j)$  と  $(i, j^+)$  の間には交差点が存在しない。

2)  $(i, j)$  が r 点であるための必要十分条件は,  $(i, j)$  が TR 点であることである。

3)  $(i, j)$  が C 点であるための必要十分条件は,  $(i, j)$  が交差点であることである。

注意: 1) の ii) において,  $(i, j)$  が CI 点である場合, 定理 3.2 から必ず  $j^+$  は存在する。

i 点, r 点, C 点の例を図 3.9 に示す。

[定理 3.4]

2 値画像  $P$  に含まれる i 点, r 点, そして C 点の総数を  $N_P$  とし,  $P$  に描かれた図形の辺数を  $L_P$  とすれば, 次の不等式が常に成立する,

$$L_P / 2 \leq N_P \leq 3 L_P / 4$$

(証明)

縦辺の上端の画素  $(i, j)$  は変化点であり, しかも,  
 $(p(i-1, j-1), p(i-1, j)) \neq (p(i, j-1), p(i, j))$   
を満足するので, 選択画素 i 点, r 点, C 点のいずれかである。このことと縦辺の数が  $L_P / 2$  であることより左側の不等号が導かれる。一方, 縦辺の直下の画素  $(m, n)$  は, C 点, CI 点, CR 点のいずれかに限られる。 $(m, n)$  が C 点ならば,  $(m, n)$  はある縦辺の上端でもある。また,  $(m, n)$  が CR 点であれば定義から r 点ではない。そこで,  $(m, n)$  が CI 点のみならず i 点で

あると仮定する。この場合、定義から  $(m, n)$  の右側に、 $C R$  点  $(m, n^+)$ ,  $n^+ = \min\{l \mid (m, l) \text{ は } R \text{ 点}, t > j\}$  が存在する。 $(m, n^+)$  はある縦辺の下に位置しており、 $(m, n)$  にはあわせて二つの縦辺が対応する。すなわち、縦辺の上端に対応していない選択画素の数は高々  $L_P/4$  である。また、選択画素  $i$  点,  $r$  点,  $C$  点は縦辺の上端または下端にしかあらわれないので、結局、 $i$  点,  $r$  点,  $C$  点の総数は高々  $3L_P/4$  である。(証明終り)

例えば、図 3.9 では  $L_P = 24$ ,  $N_P = 15$  で、定理の主張は成立する。

画像  $P$  に  $i$  点,  $r$  点,  $C$  点が  $K$  個含まれるとき、その位置  $(I, J)$  と種類  $X (= i, r, C)$  を組にして、位置の大きさ順に並べた系列  $d_P = \{(I_1, J_1, X_1), \dots, (I_K, J_K, X_K)\}$  を  $P$  の  $(i, r, C)$ -表現と呼ぶ。 $i$  点,  $r$  点,  $C$  点と  $(i, r, C)$ -表現に含まれない  $I$  点,  $R$  点とに対して次の関係が成立する。

[定理 3.5]

1)  $X_n = i$  または  $C$  のとき、もし、

$J_n^f = \min\{j \mid (I_{n-1}, j) \text{ は変化点}, (I_{n-1}, J_n) < (I_{n-1}, j) < (I_{n-1}, J_{n+1})\}$  が存在するならば、 $(I_n, J_n^f)$  は  $C R$  点である。

2)  $X_n = r$  または  $C$  のとき、もし、

$J_n^b = \max\{j \mid (I_{n-1}, j) \text{ は変化点}, (I_{n-1}, J_{n-1}) < (I_{n-1}, j) < (I_{n-1}, J_n)\}$  が存在するならば、 $(I_n, J_n^b)$  は  $i$  点とは異なる  $C I$  点である。

(1) の証明)



$X_n = i$  または  $C$  で,  $J_n^f$  が存在すると仮定する。このとき,  $J_n^f$  の定義から,

$$p(I_{n-1}, j) = p(I_{n-1}, J_n), \quad J_n \leq j \leq J_n^f - 1 \quad (1)a$$

$$p(I_{n-1}, J_n^f) = 1 - p(I_{n-1}, J_n) \quad (1)b$$

が成立する。また, 仮定より,

$$p(I_n, J_n) = 1 - p(I_{n-1}, J_n) \quad (2)$$

である。さらに,

$$p(I_n, j) = p(I_n, J_n), \quad J_n \leq j \leq J_n^f \quad (3)$$

が次のようにして導かれる。

ある  $J_n < j' \leq J_n^f$  をみたす  $j'$  に対して,

$$p(I_n, j) = p(I_n, J_n), \quad J_n \leq j \leq j' - 1 \quad (4)a$$

$$p(I_n, j') = 1 - p(I_n, J_n) \quad (4)b$$

が成立すると仮定する。このとき,

$$p(I_n, j' - 1) = p(I_n, J_n)$$

$$(a) = 1 - p(I_{n-1}, J_n)$$

$$(b) = 1 - p(I_{n-1}, j' - 1), \quad (5)$$

である。ここで, 最初の等式は (4)a より, (a) は (2), (b) は (1)a より導かれる。

もし,  $j' = J_n^f$  であるならば,

$$p(I_n, j') = 1 - p(I_n, J_n)$$

$$(c) = p(I_{n-1}, J_n)$$

$$(d) = 1 - p(I_{n-1}, j') \quad (6)$$

ここで, 最初の等式は (4)b, (c) は (2), (d) は (1)b より導かれる。また (4)a, (4)b から

$$p(I_n, j') = 1 - p(I_n, j' - 1) \quad (7)$$

である。(5), (6), (7)は  $(I_n, j')$  が C 点であることを意味するが,  $(I_n, J_n) < (I_n, j') \leq (I_n, J_n^f) < (I_{n+1}, J_{n+1})$  より,  $(I_n, J_n)$  と  $(I_{n+1}, J_{n+1})$  の間には  $i$  点,  $r$  点, C 点が存在しないという前提に矛盾する。一方,  $J_n < j' < J_n^f$  ならば,

$$\begin{aligned} p(I_{n-1}, j') &= p(I_{n-1}, J_n) \\ (e) \quad &= 1 - p(I_n, J_n) \\ (f) \quad &= p(I_n, j') \end{aligned} \tag{8}$$

ここで最初の等式は(1)a から, (e)は(2)から, (f)は(4)a から導かれる。

(5), (7), (8)より  $(I_n, j')$  は  $r$  点でなければならないが, これは矛盾である。したがって, (3)が成立する。さて,

$$\begin{aligned} p(I_{n-1}, J_n^f) &= 1 - p(I_{n-1}, J_n) \\ (g) \quad &= p(I_n, J_n) \\ (h) \quad &= p(I_n, J_n^f), \end{aligned} \tag{9}$$

$$\begin{aligned} p(I_{n-1}, J_n^{f-1}) &= p(I_{n-1}, J_n) \\ (i) \quad &= 1 - p(I_n, J_n) \\ (j) \quad &= 1 - p(I_n, J_n^{f-1}), \end{aligned} \tag{10}$$

さらに,

$$p(I_n, J_n^{f-1}) = p(I_n, J_n^f) \tag{11}$$

が成立する。ここで, (9)と(10)の最初の等式は(1)b, (1)aそのもので, (g)と(i)は(2)から, (h)と(j)さらに(11)は(3)から導かれる。(9), (10), (11)は  $(I_n, J_n^f)$  が C R 点であることを示している。

(2)の証明)

$X_n = r$  または C で,  $J_n^b$  が存在すると仮定する。この

とき,  $J_n^b$  の定義から,

$$p(I_{n-1}, j) = p(I_{n-1}, J_n - 1), \quad J_n^b \leq j \leq J_n - 1 \quad (12)_a$$

$$p(I_{n-1}, J_n^b - 1) = 1 - p(I_{n-1}, J_n - 1) \quad (12)_b$$

が成立する。また, 仮定より,

$$p(I_n, J_n - 1) = 1 - p(I_{n-1}, J_n - 1) \quad (13)$$

である。さらに,

$$p(I_n, j) = p(I_n, J_n - 1), \quad J_n^b - 1 \leq j \leq J_n - 1 \quad (14)$$

が次のようにして導かれる。

ある  $J_n^b \leq j'' < J_n - 1$  をみたす  $j''$  に対して,

$$p(I_n, j) = p(I_n, J_n - 1), \quad j'' \leq j \leq J_n - 1, \quad (15)_a$$

$$p(I_n, j'' - 1) = 1 - p(I_n, J_n - 1) \quad (15)_b$$

が成立したとする。 (15)<sub>a</sub>, (15)<sub>b</sub> より

$$p(I_n, j'') = 1 - p(I_n, j'' - 1) \quad (16)$$

である。さらに,

$$p(I_n, j'') = p(I_n, J_n - 1)$$

$$(k) = 1 - p(I_{n-1}, J_n - 1)$$

$$(l) = 1 - p(I_{n-1}, j'') \quad (17)$$

が成立しなければならない。ここで, (17) の最初の等式は (15)<sub>a</sub> より, (k) は (13) より, (l) は (12)<sub>a</sub> より導かれる。

もし,  $j'' = J_n^b$  ならば,

$$p(I_n, j'' - 1) = 1 - p(I_n, j'')$$

$$(m) = 1 - p(I_n, J_n - 1)$$

$$(n) = p(I_{n-1}, J_n - 1)$$

$$(o) = 1 - p(I_{n-1}, j'' - 1) \quad (18)$$

となる。ここで (18) の最初の等式は (16) から, (m) は (15)<sub>a</sub>

から, (n)は(13)から, (o)は(12)b から導かれる。

(16), (17), (18) より  $(I_n, j'')$  は C 点でなければならないが,  $(I_{n-1}, J_{n-1}) < (I_n, J_n^b) \leq (I_n, j'') < (I_n, J_n)$  なので,  $(I_{n-1}, J_{n-1})$  と  $(I_n, J_n)$  の間には選択画素 i 点, r 点, C 点は存在しないという前提に矛盾する。一方, もし,  $J_n^b < j'' < J_n - 1$  ならば,

$$\begin{aligned} p(I_{n-1}, j''-1) &= p(I_{n-1}, J_n-1) \\ (p) &= 1 - p(I_n, J_n-1) \\ (q) &= p(I_n, j''-1) \end{aligned} \tag{19}$$

でなければならない。ここで, (19)の最初の等式は(12)aより, (p)は(13)より, (q)は(15)b より導かれる。

(16), (17), (19) より  $(I_n, j'')$  は i 点でなければならないが, これは矛盾である。したがって, (14)が成立する。さて,

$$\begin{aligned} p(I_{n-1}, J_n^b-1) &= 1 - p(I_{n-1}, J_n-1) \\ (r) &= p(I_n, J_n-1) \\ (s) &= p(I_n, J_n^b-1), \end{aligned} \tag{20}$$

$$\begin{aligned} p(I_{n-1}, J_n^b) &= p(I_{n-1}, J_n-1) \\ (t) &= 1 - p(I_n, J_n-1) \\ (u) &= p(I_n, J_n^b) \end{aligned} \tag{21}$$

$$p(I_n, J_n^b-1) = p(I_n, J_n^b) \tag{22}$$

が成立する。ここで, (20)と(21)の最初の等号はそれぞれ(12)b と(14) を書きなおしたものであり, (r)と(t)は(13)より, (s)と(u)そして(22)は(14)より導かれる。

(20), (21), (22) より,  $(I_n, J_n^b)$  は C I 点である。(証明終り)

[定理 3.6]

- 1) 任意の CR 点  $(i, j)$  に対して適当な  $n$  が存在し,  $I_n = i$ ,  $J_n^f = j$ ,  $X_n = i$  または  $C$  が成立する。
- 2) 任意の  $i$  点とは異なる CI 点  $(i, j)$  に対して適当な  $n$  が存在し,  $I_n = i$ ,  $J_n^b = j$ ,  $X_n = r$  または  $C$  が成立する。

(1) の証明)

任意の CR 点  $(i, j)$  を一つ固定する。  $p(i, j-1) = 1 - p(i-1, j-1)$  かつ  $p(i, 0) = p(i-1, 0)$  なので,  

$$j_s = \max \{ k \mid (p(i, k), p(i-1, k)) \neq (p(i, j-1), p(i-1, j-1)), 0 \leq k < j-1 \}$$

が必ず存在する。このとき,

$$p(i, j_s + 1) = 1 - p(i-1, j_s + 1) \quad (1)$$

である。

$p(i, j_s)$  と  $p(i-1, j_s)$  の値に関して, 次の三つの場合が考えられる。

$$\textcircled{1} \quad p(i, j_s) = p(i, j_s + 1) \text{ かつ } p(i-1, j_s) = 1 - p(i-1, j_s + 1)$$

$$\text{このとき, } p(i, j_s) = p(i, j_s + 1) \quad (2)$$

$$(a) = 1 - p(i-1, j_s + 1)$$

$$(b) = p(i-1, j_s) \quad (2')$$

ここで, (a) は (1) から, (b) は  $\textcircled{1}$  の第 2 式から導かれる。

(1), (2), (2') より  $(i, j_s + 1)$  は CI 点である。さらに,  $j_s$  の定義から,  $j \leq k \leq j_s - 1$  に対し,

$$(p(i, k), p(i-1, k)) = (p(i, j), p(i-1, j))$$

であることと  $(i, j)$  が CR 点であることから,  $(i, j_s + 1)$

は  $i$  点である。

$$\textcircled{2} \quad p(i, j_s) = 1 - p(i, j_s + 1) \text{ かつ } p(i-1, j_s) = p(i-1, j_s + 1)$$

$$\text{このとき, } p(i, j_s) = 1 - p(i, j_s + 1) \quad (3)$$

$$(c) = p(i-1, j_s + 1)$$

$$(d) = p(i-1, j_s) \quad (3)'$$

ここで, (c)は(1)から, (d)は②の第2式から導かれる。

(1), (3), (3)'は  $(i, j_s + 1)$  が  $i$  点であることを示している。

$$\textcircled{3} \quad p(i, j_s) = 1 - p(i, j_s + 1) \text{ かつ } p(i-1, j_s) = 1 - p(i-1, j_s + 1)$$

$$\text{このとき, } p(i, j_s) = 1 - p(i, j_s + 1) \quad (4)$$

$$(e) = p(i-1, j_s + 1)$$

$$(f) = 1 - p(i-1, j_s) \quad (4)'$$

ここで, (e)は(1)から, (f)は③の第2式から導かれる。

(1), (4), (4)'は  $(i, j_s + 1)$  が  $C$  点であることを示している。

①②③のいずれにおいても  $(i, j_s + 1)$  は  $i$  点あるいは  $C$  点である。このことは, ある  $n$  が存在して,  $(I_n, J_n) = (i, j_s)$  を意味する。  $(i, j)$  は  $CR$  点なので, 定理3.1より  $(i-1, j)$  は変化点である。したがって,  $J_n^f$  は必ず存在する。また,  $j_s$  の定義から区間  $[(i-1, j_s + 1), (i-1, j-1)]$  には変化点は存在しない。これは, 区間  $[(I_{n-1}, J_{n-1} + 1), (I_{n+1} - 1, J_{n+1} - 1)]$  における最小の変化点が  $(i, j)$  であることを意味する。すなわち,  $J_n^f = j$  である。

(2)の証明)

任意の  $i$  点でない  $CI$  点  $(i, j)$  を一つ固定する。このとき,  $p(i, j) = 1 - p(i-1, j)$  かつ  $p(i, N+1) = p(i-1, N+1) = 0$  より必ず,

$$j_E = \min \{ k \mid (p(i, k), p(i-1, k)) \neq (p(i, j), p(i-1, j)), \\ j < k \leq N+1 \}$$

が存在する。このとき、

$$p(i, j_E - 1) = 1 - p(i-1, j_E - 1) \quad (5)$$

が成立する。 $p(i, j_E)$  と  $p(i-1, j_E)$  の値に関しては、  
1) の場合と同様に、次の三つの場合が考えられる。

$$\textcircled{1} \quad p(i, j_E) = p(i, j_E - 1) \text{ かつ } p(i-1, j_E) = 1 - p(i-1, j_E - 1)$$

$$\text{このとき, } p(i, j_E) = p(i, j_E - 1) \quad (6)$$

$$(g) = 1 - p(i-1, j_E - 1)$$

$$(h) = p(i-1, j_E) \quad (6')$$

ここで、(g) は (5) から (h) は  $\textcircled{1}$  の第 2 式から導かれる。(5),  
(6), (6)' は  $(i, j_E)$  が C R 点であることを意味する。ところ  
ろが、 $j_E$  の定義から、 $j \leq k \leq j_E - 1$  に対して

$(p(i, k), p(i-1, k)) = (p(i, j), p(i-1, j))$  が成立す  
る。これは、 $(i, j)$  が  $i$  点であることを示しているが  
 $(i, j)$  が  $i$  点とは異なるという前提に矛盾する。すな  
わち、 $(i, j)$  が  $i$  点と異なる C I 点のときには  $\textcircled{1}$  は起り  
えない。

$$\textcircled{2} \quad p(i, j_E) = 1 - p(i, j_E + 1) \text{ かつ } p(i-1, j_E) = p(i-1, j_E + 1)$$

$$\text{このとき, } p(i, j_E) = 1 - p(i, j_E + 1) \quad (7)$$

$$(i) = p(i-1, j_E + 1)$$

$$(j) = p(i-1, j_E) \quad (7')$$

ここで、(i) は (5) から (j) は  $\textcircled{2}$  の第 2 式から導かれる。(5),  
(7), (7)' は  $(i, j_E)$  が r 点であることを示している。

$$\textcircled{3} \quad p(i, j_E) = 1 - p(i, j_E + 1) \text{ かつ } p(i-1, j_E) = 1 - p(i-1, j_E + 1)$$

$$\text{このとき, } p(i, j_E) = 1 - p(i, j_E + 1) \quad (8)$$

$$(k) = p(i-1, j_E + 1)$$

$$(l) = 1 - p(i-1, j_E + 1) \quad (8')$$

ここで、(k)は(5)から(l)は③の第2式から導かれる。(5), (8), (8')は $(i, j_E)$ がC点であることを示している。

②, ③のいずれにおいても、 $(i, j_E)$ はr点あるいはC点である。このことは、ある $n$ が存在して、 $(I_n, J_n) = (i, j_E)$ を意味する。 $(i, j)$ はCI点なので定理3.1より $(i-1, j)$ は変化点である。したがって、 $J_n^b$ は必ず存在する。また、 $j_E$ の定義より、区間 $[(i-1, j), (i-1, j_E-1)]$ には変化点は存在しない。これは、区間 $[(I_{n-1}-1, J_{n-1}+1), (I_{n-1}, J_{n-1})]$ における最大の変化点は $(i, j)$ であることを意味している。すなわち、 $J_n^b = j$ である。

(証明終り)

定理3.5と定理3.6を一つにまとめると、2値画像Pの $(i, r, C)$ -表現に対して次の定理が成立する。

[定理3.7]

2値画像Pの $(i, r, C)$ -表現 $dp = \{(I_1, J_1, X_1), \dots, (I_k, J_k, X_k)\}$ が与えられているとする。このとき、Pの部分区間 $[(I_n, J_{n+1}), (I_{n+1}, J_{n+1}-1)]$ にはCR点 $(I_n, J_n^f)$ とi点でないCI点 $(I_{n+1}, J_{n+1}^b)$ が高々一つずつ存在する。

(証明)

定理3.5と定理3.6より明らかである。(証明終り)

$(i, r, C)$ -表現からもとの画像を復元するには、この定理3.7として得られた知見をもとに次のようにして



行なう。

### [アルゴリズム 2]

ある  $(i, r, C)$ -表現  $d_p$  を入力とし  $P_x$  を出力する。

方法: 区間  $[(I_n, J_n), (I_{n+1}, J_{n+1}-1)]$ ,  $0 \leq n \leq k$  ごとに,  $J_n^a$  および  $J_{n+1}^b$  を捜しながら  $(i, r, C)$ -表現を  $(I, R)$ -表現に変換して, アルゴリズム 1 を用いて元の画像を復元する。

$$(I_0, J_0, X_0) \leftarrow (1, 0, r)$$

$$(I_{k+1}, J_{k+1}, X_{k+1}) \leftarrow (M, N+2, i)$$

とおく。アルゴリズムを図 3.10 に示す。

### [定理 3.8]

アルゴリズム 2 は, ある  $(i, r, C)$ -表現  $d_p$  から元の画像  $P$  を正確に復元する。

(証明)

$m-1$  行 ( $m \geq 1$ ) までの  $P_x$  の画素値が正確に復元されているとする。なお第 0 行は  $P_x$  の条件からどの列も値 0 をとるので常に正確に復元される。このとき,  $m$  行の画素値が正確に復元されることを示せばよい。もし,  $m$  行に  $i$  点,  $r$  点,  $C$  点のいずれも存在しなければ, ある  $n$  ( $\geq 0$ ) に対し,  $I_n \leq m < I_{n+1}$  が成立する。但し, 等号成立は  $m=1$ , すなわち  $n=0$  の場合に限る。このとき,  $m$  行はアルゴリズム 2 のステップ 8 の実行中において値が定まる。このステップでの操作はアルゴリズム 1 の場合と同じなので,  $m$  行は正確に復元される。次に,  $m$  行に  $i$  点,  $r$  点,  $C$  点のいずれか

```

begin
1   Q ← 0 - matrix of the size ( M + 1 ) × ( N + 2 )
2   for n ← 0 until K do
      begin
3       CURPEL ← (In, Jn);
4       if ( Xn is " i " or " c " ) then
5           if ( Jnf exists ) then
              begin
6                 DECODE(CURPEL, (In, Jnf - 1), 1);
7                 CURPEL ← (In, Jnf )
              end;
8           if ( In+1 > In ) then
              begin
9                 DECODE(CURPEL, (In+1 - 1, M), 0);
10                CURPEL ← (In, Jnf )
              end;
11          if ( Xn+1 is " r " or " c " ) then
              begin
12              FLAG ← 1;
13              if ( Jn+1b exists ) then
                  begin
14                      DECODE(CURPEL, (In+1, Jn+1b - 1), 0);
15                      CURPEL ← (In+1, Jn+1b )
                  end
              end
16          else FLAG ← 0;
17          DECODE(CURPEL, (In+1, Jn+1 - 1), FLAG)
18      end;
  return(Q)
end

```

図 3.10 アルゴリズム 2: (i, r, C)-表現から元の画像を復元するアルゴリズム

が存在するとする。その一つを  $(I_p, J_p)$ ,  $I_p = m$  とする。定理 3.7 より,  $(I_p, J_p)$  の前後には,  $CR$  点  $(I_p, J_p^f)$  あるいは  $i$  点と異なる  $CI$  点  $(I_p, J_p^b)$  が存在する可能性がある。これらの画素が存在するかしないか, さらに存在するとすればどの位置かということは,  $m-1$  行の値と,  $X_p, J_{p-1}, J_{p+1}$  から正確に判定できる。アルゴリズム 2 ではこの判定をステップ 5 とステップ 13 において行なっている。判定後の操作は, あらかじめ,  $(I, R)$ -表現が得られている場合のアルゴリズム 1 で行なう操作と等しいことが見てとれるので,  $m$  行は正確に復元される。(証明終り)

定理 3.4 と定理 3.8 は, 3.1 節の最後で提起した問題に対して一つの解答を与えている。すなわち, 2 値画像を一意に特徴づけるために抽出しなければならない特徴点の数は辺数の半分では一般に不十分であったが, 高々辺数の  $3/4$  の選択画素を選択画素の種類 ( $i$  点,  $r$  点,  $C$  点) を示す付加情報とともに取り出せば, 元の画像を復元することができることをこれらの定理は示している。

### 3.4 選択画素の右変形と左変形

2 値画像の  $(i, r, C)$ -表現は, 特に, 回路図や機械部品の設計図などのように, 縦・横の直線で構成された画像の表現としては効率的であるが, 斜め  $45^\circ$  の直線部分において他の場所以上に選択画素が多く現われ

る傾向をもつ(図3.11参照)。この節では、斜め45°の直線上にあるi点, r点, C点を省いた、より少ない選択画素による画像表現を試みる。

まず, i点, r点, C点の定義を次のように細分化する。

### [定義3.3]

画素 $(i, j)$ が選択画素i点, r点, C点のいずれかであるとする。このとき

1) もし,  $p(i+1, j-1) = p(i, j)$  かつ  $(i+1, j-1)$  は変化点であるならば, 画素 $(i, j)$  を選択画素の左変形と呼ぶ。そして, i点, r点, C点の左変形をそれぞれ,  $i_L$ 点,  $r_L$ 点,  $C_L$ 点と記す。また,

2) もし,  $(i, j)$  が左変形の条件を満たさず, さらに,  $p(i+1, j+1) = p(i, j)$  かつ  $(i+1, j+1)$  は変化点であるならば, 画素 $(i, j)$  を選択画素の右変形と呼ぶ。そして, i点, r点, C点の右変形をそれぞれ,  $i_R$ 点,  $r_R$ 点,  $C_R$ 点と記す。

図3.11の画像に対し, 定義3.3にしたがって選択画素を列挙すると図3.12のようになる。これだけでは選択画素の種類だけが増えて, 選択画素数はもとのままである。選択画素を減らすためには, 次に述べるランシフト動作を行なう必要がある。

図3.12では, 単に定義3.2と定義3.3を満たす選択画素が同時にその全部が表示されているが, 実際に選択画素の抽出を行なうには, 画面をラスタ走査しながら

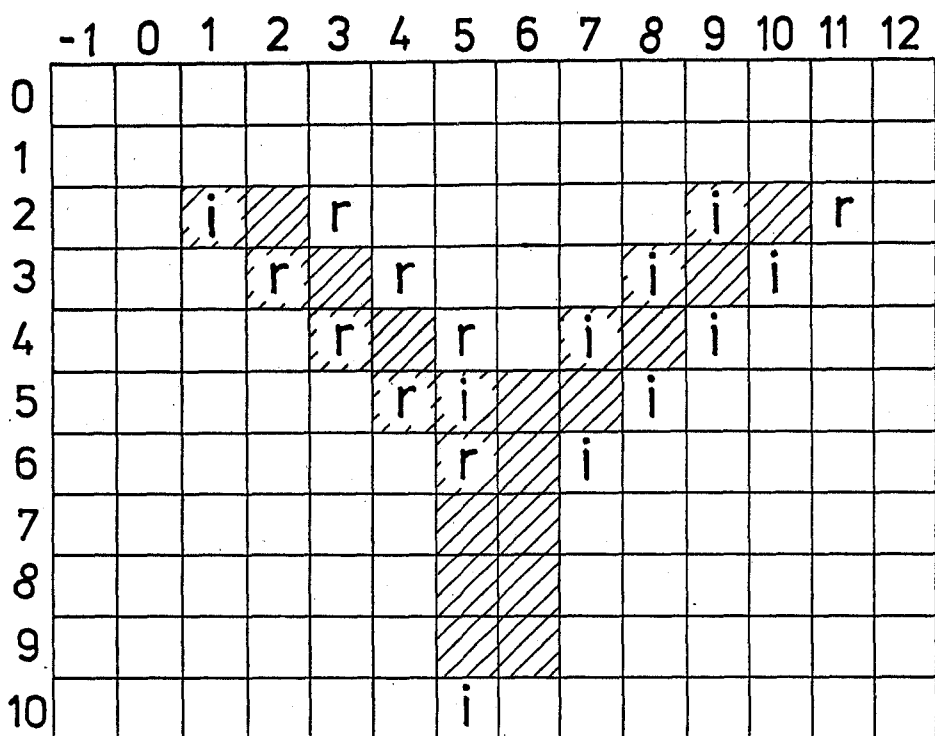


図 3.11 i 点と r 点の例

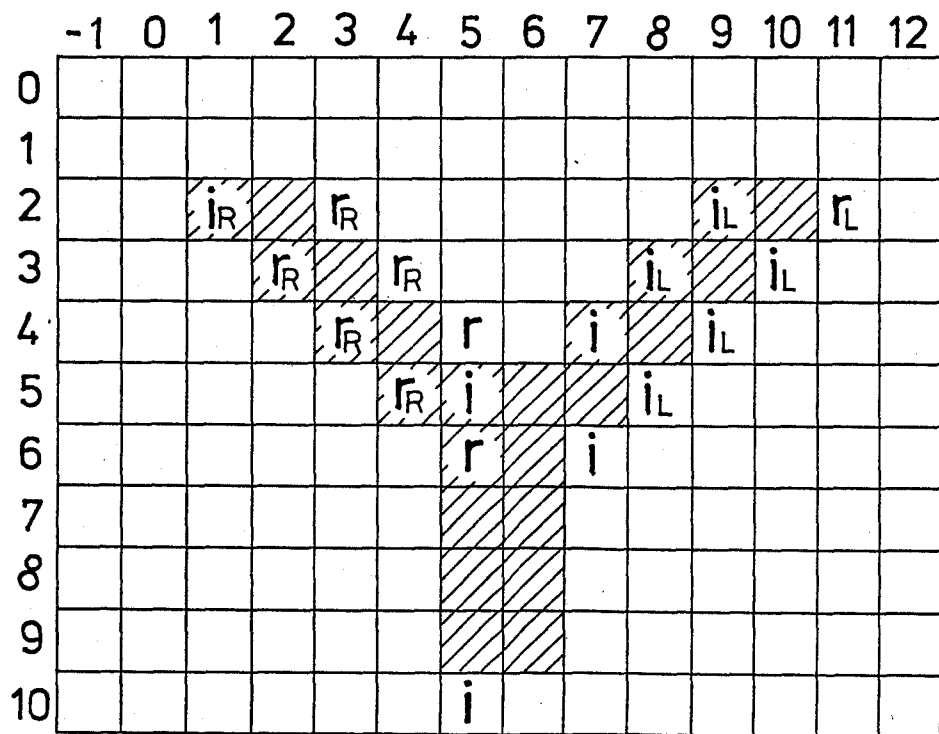


図 3.12 変形選択画素の例

ら、現在選択画素を抽出中の行とその直前の行、そして、次に抽出を行なう行の画素値からこれらの定義を満足する画素を逐次に見つけてゆくという動作を行なう。これを選択画素の抽出過程（あるいは、符号化過程）と呼ぶ。抽出過程を行なう際、ある行の抽出が終了して次の行の選択画素の抽出を始める前に、次の行で抽出される選択画素をできるだけ少なくするために、今抽出し終えた行の画素値を適当な方法にしたがって修正することにする。この修正方法をランシフト動作と呼ぶ。ここでは特に斜め45°の直線部分に対するランシフト動作について議論する。

抽出過程では、各行の選択画素の抽出が終了するごとにその行の画素値をラン・シフト動作によって次々に変更してゆく。抽出過程の処理手順を厳密に与えるまえにまず、図3.12を例にとりランシフト動作の原理を簡単に説明する。

図3.12において第1行、第2行と走査が進み、第2行の最後の列に走査が進んだ時点では、画素(2,1), (2,3), (2,9), (2,11)がそれぞれ  $i_R$  点,  $r_R$  点,  $i_L$  点,  $r_L$  点として抽出される。ここで、図3.13aに示すように左変形の位置にはL, 右変形の位置にはRという印をつけておく。さて、第3行の走査を開始する前に、第2行の画素値を次のように変更する。すなわち、図13bに示すように第2行上でLと印された画素位置を画素値を保ったまま左に一つ、Rと印された画素位置

をやはり画素値を保ったまま右に一つずらす。さらに、第3行において、第2行での“ずらし”によって移動した画素の真下の画素、すなわち、画素(3,2),(3,4),(3,8),(3,10)を調べて、もし、変化点であるならばそれらの真上の画素に印された記号RおよびLをそのまま継承する。もし、変化点でなければ継承は行なわない。この場合、列挙した四つの画素はすべて変化点なので、(3,2)と(3,4)にはR、そして(3,8)と(3,10)にはLという印がつけられている。この一連の画素値の修正により、第2行の1のランは、それ自身に印された記号の示す方向に一つ移動している。これが、この修正方法をラン・シフト動作と呼ぶ理由である。ラン・シフト動作によって、第3行の走査において本来なら、(3,2),(3,4),(3,8),(3,10)はそれぞれ、 $i_R$ 点、 $r_R$ 点、 $i_L$ 点、 $r_L$ 点であったのが選択画素ではなくなっていることが見てとれる。結局、第3行から抽出される選択画素は一つもなくなってしまう。第3行の走査が終了した時点で第2行のときと同様にラン・シフト動作を行ない、図3.13cを得る。このため、第4行においても選択画素は抽出されない。次に、ラン・シフト動作によって第4行は図3.13dのように修正される。このとき、(5,5)と(5,6)は変化点ではないため印の継承は、(5,4)と(5,8)においてのみ行なわれる。第5行にも選択画素は存在せず、図3.13eのように修正が施される。第6行にも選択画素は存在しない。ラン・シフト動作によって、第6行の画素値は

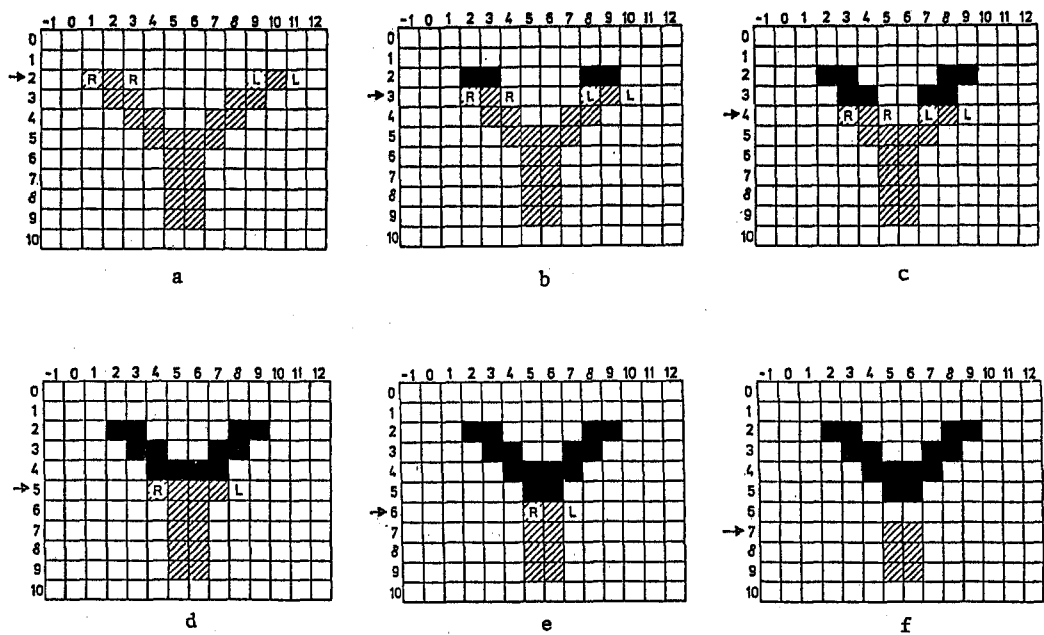


図 3.13 ランシフト動作の説明

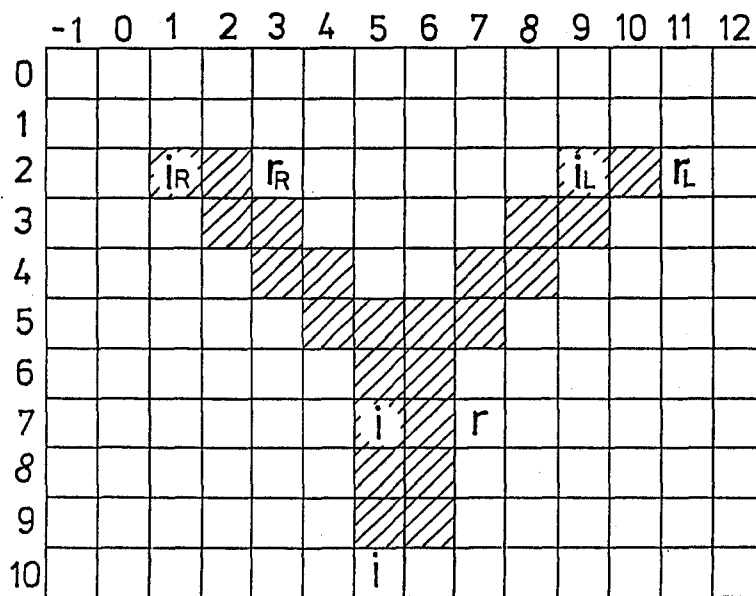


図 3.14 ランシフトの結果選ばれる  
選択画素の例



図 3.13 f に示すようになる。このとき、印 R のついた画素と L のついた画素は、同じ  $(6, 6)$  へと移動するが、 $(7, 6)$  は変化点ではないので印の継承は行なわれない。第 7 行から以降ではラン・シフト動作は行なわれず、 $(7, 5)$  と  $(10, 5)$  が  $i$  点、 $(7, 7)$  が  $r$  点として抽出される。結局、 $(i, r, c)$  - 表現では図 3.12 に 18 個もあった選択画素数を、変形選択画素とラン・シフト動作を導入することによって、図 3.14 に示すように 7 個にまで減らすことができた。上で述べた抽出過程を厳密に述べると次のようになる。

### 〔アルゴリズム 3〕

入力:  $P_{ex}$  の両端に値 0 の列を加えた  $(M+2) \times (N+4)$  行列  $P_{ex}$

出力: 区間  $\{(i, j) \mid 1 \leq i \leq M, 0 \leq j \leq N+2\}$  内に含まれる選択画素とその右および左変形の位置と種類のリスト ENCODE.

方法: ニつの長さ  $N+4$  の 1 次元配列 LEFT と RIGHT をそれぞれ、印 L と R を記憶するために使用する。

LEFT と RIGHT の各要素は値 0 または 1 をとり、その初期値は 0 とする。

各  $i$  行 ( $i \geq 1$ ) ごとに  $i-1$  行と  $i+1$  行の画素値から定義 3.2 と定義 3.3 に基づいて選択画素を捜し、その位置と種類をリスト ENCODE に登録する。このとき変形選択画素が見つかれば、右変形なら RIGHT、左変形なら LEFT の対応する場所の値を 1 とする。 $i$  行の処理が終了したら次行の処理に移る前に、手続

```

1  for (i,j) ← (1,1) until (M,N) do
    begin
2      n ← 0;
3      if (i,j)は選択画素S (=i,r,c)である then
        begin
4          LEFT[j] ← 0;
5          RIGHT[j] ← 0;
6          X ← S;
7          if ( p(i,j) = p(i+1,j-1) かつ p(i+1,j-1)は変化点 ) then
            begin
8              X を左変形  $S_L$  に変える ;
9              LEFT[j] ← 1
            end
10         else if ( P(i,j) = p(i+1,j+1) かつ p(i+1,j+1)は変化点 ) then
            begin
11             X を右変形  $S_R$  に変える ;
12             RIGHT[j] ← 1
            end;
13         n を一つ増し, ENCODE[n] ← (i,j,X) とする。
        end;
14     if j = N then RUNSHIFT(i)
    end

procedure RUNSHIFT(i)
    begin
1      for j ← 1 until N do
2          if (i,j)は変化点でない then RIGHT[j] と LEFT[j] を0とする;
3          for j ← 1 until N-1 do LEFT[j] ← LEFT[j+1];
4          LEFT[N] ← 0;
5          for j ← 1 until N do
6              if( LEFT[j] = 1 or RIGHT[j] = 1 ) then
7                  p(i,j) ← 1 - p(i,j);
8          for j ← N step -1 until 2 do RIGHT[j] ← RIGHT[j-1];
9          RIGHT[1] ← 0
    end

```

図 3.15 アルゴリズム 3 : 変形( $i, r, c$ )-表現から元の画像を復元するアルゴリズム

き  $RUNSHIFT(i)$  を行なう。この手続きは、 $RIGHT$ ,  $LEFT$  のいずれかの値が画像  $P_x$  の  $i$  行に存在する 1-ランまたは 0-ランの先頭の位置において 1 である場合に限り、該当するランの先頭を  $RIGHT$  が 1 なら右へ、 $LEFT$  が 1 なら左へ一つ移す。それと同時に、配列  $RIGHT$ ,  $LEFT$  内で 1 をとる場所もランの先頭の移動に伴ない同じ方向へ一つずらす。アルゴリズムは図 3.15 に示す。

$P_x$  から求まる  $ENCODE$  に登録された選択画素の位置  $(I_n, J_n)$  と種類  $X_n (= i, r, c, i_R, r_R, c_R, i_L, r_L, c_L)$  を組にして位置の大きさ順に並べた系列  $\tilde{d}_P = \{(I_1, J_1, X_1), \dots, (I_K, J_K, X_K)\}$  を 2 値画像  $P$  の 変形  $(i, r, c)$  - 表現 と呼ぶ。変形  $(i, r, c)$  - 表現からの元の画像の復元は次のようにして行なう。

#### [アルゴリズム 4]

リスト  $ENCODE$  を入力とし、元の画像  $P$  を出力する。アルゴリズム 3 と同様に初期値 0 の長さ  $N$  の配列  $LEFT$  と  $RIGHT$  を用いる。アルゴリズムはアルゴリズム 2 に次の動作を付け加える。

- ① アルゴリズム 2 のステップ 3 で  $X_n$  を調べる際、 $X_n$  がある選択画素の右変形なら  $RIGHT[J_n] \leftarrow 1$ 、あるいは左変形なら  $LEFT[J_n] \leftarrow 1$  とおく。
- ② 同様にステップ 10 で  $X_{n+1}$  を調べる際に、 $X_{n+1}$  がある選択画素の右変形なら  $RIGHT[J_{n+1}] \leftarrow 1$ 、あるいは左変形なら  $LEFT[J_{n+1}] \leftarrow 1$  とおく。

③ステップ8において各行 ( $I_n \leq i \leq I_{n+1}$ ) の復元が終った時点で  $RUNSHIFT(i)$  を実行する。

[定理 3.9]

アルゴリズム4は、 $ENCODE$  から原画像を正確に復元する。

(証明)

$ENCODE$  に登録されている最初の右変形あるいは左変形を  $(I_f, J_f)$  とする。  $(1, 0)$  から  $(I_f, N+2)$  までのアルゴリズム4の動作は、 $I_f$  行において右変形と左変形がリスト  $ENCODE$  から取り出されるたびごとに、 $RIGHT$  および  $LEFT$  の対応する位置を1にすることを除いては、本質的にアルゴリズム2の動作と同じである。したがって、 $(1, 1)$  から  $(I_f, N)$  までは正確に復元できる。 $I_f + 1$  行の復元を行なう前に、 $I_f$  行の値は  $RUNSHIFT(I_f)$  によって修正される。この修正された値は抽出過程で  $I_f$  行の処理が終了した後で行なった修正によるものと等しい。したがって、 $I_f + 1$  行の復元された画素値は、アルゴリズム2の復号の一意性から、元のものに等しい。 $I_f + 2$  行以降についても、上の議論をくり返し用いることにより、元の画像が正確に復元されることが示される。

(証明終り)

アルゴリズム3と4を合せて、2値画像の選択画素符号化方式 (Selective Element Coding Techniques; 略して  $SECT$ ) と呼ぶ。

2値画像をSECTによって符号化する場合に、選  
び出された選択画素の種類と位置をどのようにして符  
号語に変換するかが問題となる。選択画素の位置の符  
号化方法については実際の画像データに則して次章で  
考察することにする。ここでは、選択画素の種類の符  
号化に関して少し議論を行ないこの章を終えることに  
する。

### [定理 3.10]

画素  $(i, j)$  が  $r$  点か  $c$  点のいずれかであるとき、第  
 $i-1$  行の画素値を参照すれば、 $(i, j)$  が  $r$  点である  
か  $c$  点であるかは正確に区別できる。同じ主張は、画  
素  $(i, j)$  が  $r_L$  点か  $c_L$  点のいずれかである場合、また、  
 $r_R$  点か  $c_R$  点のいずれかである場合にも成立する。

(証明)

$(i, j)$  が  $r$  点または  $r_L$  点、 $r_R$  点であるならば定義 3.2  
より同時に  $T$   $R$  点でもある。したがって、定理 3.1 か  
ら  $(i-1, j)$  は変化点ではない。一方、 $(i, j)$  が  $c$  点ま  
たは  $c_R$  点、 $c_L$  点であるならば、 $(i-1, j)$  は変化点であ  
る。これから、前行の画素値  $p(i-1, j-1)$  と  $p(i-1, j)$  と  
によって、 $(i, j)$  が  $r$  点であるか  $c$  点であるか、また、  
 $r_R$  点であるか  $c_R$  点であるか、さらに、 $r_L$  点であるか  $c_L$   
点であるかを正確に判定することができる。

(証明終り)

選択画素の種類としては、 $i$ ,  $r$ ,  $c$ ,  $i_R$ ,  $r_R$ ,  $c_R$ ,  
 $i_L$ ,  $r_L$ ,  $c_L$  の 9 種類が考えられるが、定理 3.10 から、

用意する符号語の数は6個でよい。実際、 $r$ 点と $C$ 点、 $r_R$ 点と $C_R$ 点、そして、 $r_L$ 点と $C_L$ 点に、それぞれ、同一の符号語を割り当てたとしても、SECTの復号化過程においてすでに復元されている前行の画素値を用いれば、選択画素の種類を正確に知ることができる。

## 第4章

### 漢字パターンのデータ圧縮

近年，ワード・プロセッサや日本語による計算機言語の開発あるいは自動翻訳など，計算機をもちいた日本語処理技術が盛んに研究されている。日本語処理における問題の一つに漢字パターンの処理がある。周知のように，日常生活において必要とされる漢字の種類は2000～3000程度，漢和辞典に掲載されているものでは約50000もの種類がある。これらの膨大な量の漢字パターンを計算機内に記憶したり，ディスプレイ端末に表示するためには，処理装置の小型化また経済性の面から漢字データをできるだけ少ないビット数で表現すること，すなわち，漢字パターンの符号化が必要となる。従来，多くの漢字パターンの符号化方式が提案されている[22]～[25]が，この章では，SECTを漢字パターンの符号化に適用し，他の方式との性能の比較を行なう。

漢字パターンを得るために，図4.1に示すマイコンをもちいた画像入力装置を製作した。この装置では，ライトペンを使用してブラウン管面上に大きさ32×32の2値画像を描くことができる。マイクロ・コンピュータ・システム部はCPUに8080Aを使用し，RAM 12Kバイト，ROM 4Kバイトの記憶容量を有している。

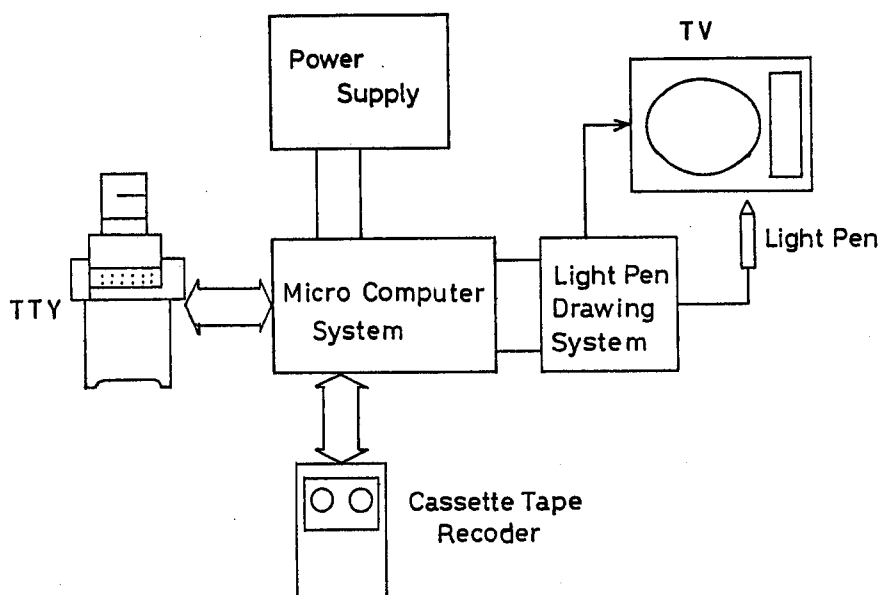
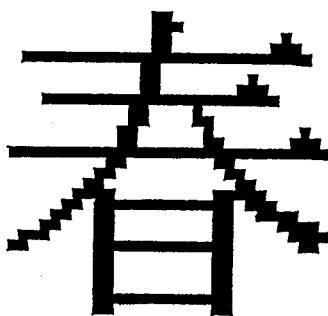


図 4.1 画像入力装置の構成



(a)

原画



(b)

32×32 デジタル画像

図 4.2 漢字パターンの例



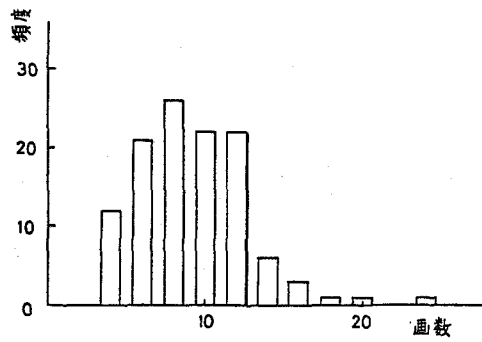


図 4.3 漢字パターンの字画数の分布

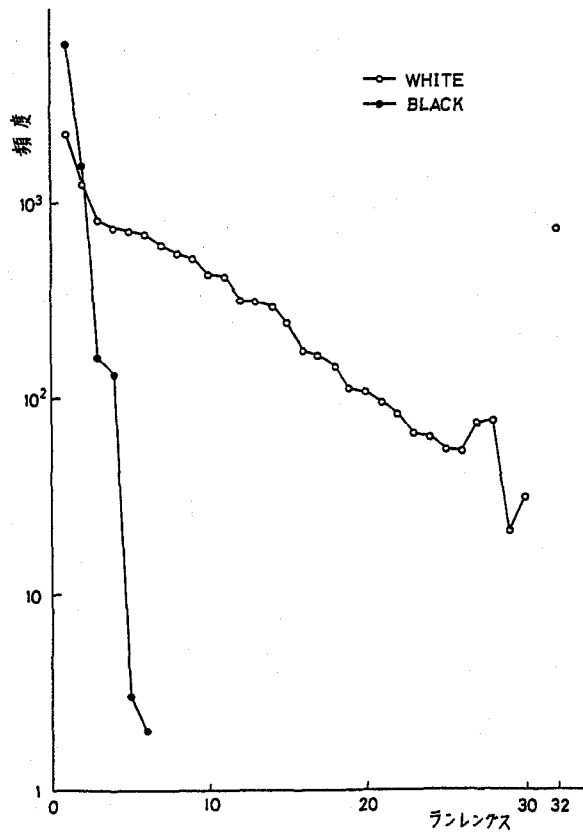


図 4.4 ランレングス分布

このシステムをもちいてブラウン管面上に描かれた画像をカセットテープに記憶することやカセットテープから画像データを画面上に伝送することができる。

漢字パターンの作成は次のようにして行なった。まず、明朝体の当用漢字を任意に選択して写真撮影を行ない、得られた印画を $32 \times 32$ に分割したものを参照しながら画像入力装置によりブラウン管面上においてパターンを作成した。その1例を図4.2に示す。作成した漢字パターンは115個で、字画数で分類したグラフを図4.3に示す。

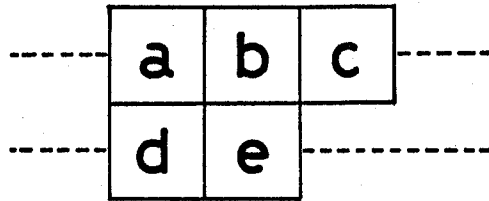
データ圧縮効果をSECTと比較するために、漢字パターン圧縮においてよく知られた次の3種類の符号化方式[26]による符号化を行なった。

#### 1) 1次元ランレングス符号化方式

漢字パターンの各行の1ランと0ランの長さを、それぞれの頻度分布にもとづいて構成された2種類のハフマン符号によって符号化する。2種類のランの分布を図4.4に示す。

#### 2) 2次元予測符号化方式

各画素の値を図4.5に示す近傍画素値の関数によって予測する。この関数は $(a, b, c, d)$ という近傍値が与えられたもとでの出現頻度の高い方の値を示している。実際の漢字パターンから求められた出現頻度と予測関数の出力値を表4.1に示す。次に、もし、画素 $(i, j)$ の予測値が $p(i, j)$ と等しければ0、異なれば1とする。



$$f(a, b, c, d) = \bar{a}b + d(\bar{a} + b)$$

図 4.5 予測関数

| (abcd) |   | (0000) | (0001) | (0010) | (0011) | (0100) | (0101) | (0110) | (0111) |
|--------|---|--------|--------|--------|--------|--------|--------|--------|--------|
| e      | 0 | 0.984  | 0.248  | 0.820  | 0.040  | 0.488  | 0.290  | 0.223  | 0.070  |
|        | 1 | 0.016  | 0.752  | 0.180  | 0.960  | 0.512  | 0.710  | 0.777  | 0.930  |
| *X     |   | 0      | 1      | 0      | 1      | 1      | 1      | 1      | 1      |
| (abcd) |   | (1000) | (1001) | (1010) | (1011) | (1100) | (1101) | (1110) | (1111) |
| e      | 0 | 0.966  | 0.695  | 0.905  | 0.612  | 0.766  | 0.212  | 0.865  | 0.250  |
|        | 1 | 0.034  | 0.305  | 0.095  | 0.388  | 0.234  | 0.788  | 0.135  | 0.750  |
| X      |   | 0      | 0      | 0      | 0      | 0      | 1      | 0      | 1      |

表 4.1 出現頻度と予測関数の出力値との関係

結果的に、もとの画像Pはある0-1行列Eに変換される。最後に、この行列Eを1次元ランレングス符号によって符号化する。

### 3) ブロック符号化方式[27]

画面を縦・横 $m \times n$ の小ブロックに分割して、すべての画素値が0である小ブロックには0、それ以外のブロックには長さ $mn$ の系列の前に1をつけ加えた長さ $nm+1$ の系列を対応させる。さらに、圧縮効果を上げるために、各小ブロックを一つのシンボルとみなし、実験データから得られた頻度分布をもとにしてハフマン符号で符号化を行なう。

SECTにおいて抽出した選択画素の位置を符号化する方法としては上述のランレングス符号化方式なども考えうるが、ここでは、より簡単な方法として以下に示す3種類の方法を用いた。

#### (a) 直接アドレス法

選択画素の位置を縦・横あわせて、 $\lceil \log_2 M \rceil + \lceil \log_2 N \rceil$ ビットで符号化する。選択画素の種類に関しては、前章の最後で指摘したように、 $r$ 点と $c$ 点、 $r_R$ 点と $c_R$ 点、そして、 $r_L$ 点と $c_L$ 点のそれぞれに同じ符号語を割当てる。具体的には、表4.2のように3ビットの符号語を対応づける。このとき、符号化データ量は、選択画素の個数を $K$ とおくと、 $K(\lceil \log_2 M \rceil + \lceil \log_2 N \rceil + 3)$ となる。

| selective element | code word |
|-------------------|-----------|
| $i$               | 0 0 0     |
| $i_R$             | 0 0 1     |
| $i_L$             | 0 1 0     |
| $r$ , $c$         | 0 1 1     |
| $r_R$ , $c_R$     | 1 0 0     |
| $r_L$ , $c_L$     | 1 0 1     |

選択画素と符号語の対応表

表 4.2

### (b) 行番号消去法

(a)において、行番号の表示を省略する。その代り、各行の最後の選択画素の列番号の後には行の終りを示すエンド・マーク 11 を挿入する。このとき、符号化データ量は  $K(\lceil \log_2 N \rceil + 3) + 2N$  となる。

### (c) エンド・マーク消去法

(b)において、ある行の最後(右端)の選択画素の列番号を  $J_E$ 、次の行の最初(左端)の選択画素の列番号を  $J_S$  とするとき、もし、 $J_S \leq J_E$  なら次行に移行したことがわかるので、その間のエンドマークを一つ省略する。

なお S E C T の(c)による符号化・復号化プログラム (8080A アセンブラ) を付録1に掲載する。

実験結果として、符号化データの平均ビット数を表4.3に示す。ここで、ブロック符号の欄の値は上が本来のブロック符号化によるもので、下はハフマン符号を用いた場合のものを示している。S E C T の欄の a, b, c は順に、直接アドレス法、行番号消去法、エンドマーク消去法による場合に相当している。

表4.3 から、エンド・マーク消去法をもちいた S E C T 方式がもっとも高い圧縮効果を示している。ここで注目すべき点は、1) から 3) までの符号化方式ではランレングスなどの統計的分布を調べ、その分布に最適なハフマン符号をもちいているが、一方、S E C T では画像データの統計的性質に依存しない符号化方法を

| Coding Scheme                       | 1-Dimensional<br>Run-Length | 2-Dimensional<br>Prediction | Block Coding |            | SECT |     |     |
|-------------------------------------|-----------------------------|-----------------------------|--------------|------------|------|-----|-----|
|                                     |                             |                             | 2 x 2        | 2 x 4      | a    | b   | c   |
| Average Bits of<br>The Encoded Data | 568                         | 547                         | 650<br>606   | 666<br>503 | 752  | 527 | 487 |

表 4.3 各種符号化方式の  
符号化結果

もちいていることである。これは、符号化すべき画素数をできるだけ少なくして、それらの画素を構成の簡単な符号で符号化するというSECTの基本方針が、十分実用面からみても有意義であることを示している。



## 第5章

### 選択画素の画像処理への応用

従来の画像認識処理の手法には、画像全体の処理を直接行なう形式のものが多い[28]が、画像からあらかじめ抽出した特徴点から各種の画像処理が行なえると、画像全体の処理を行なうことに比べ、処理時間・記憶容量の点で都合がよい。図形の平行移動並びに拡大・縮小などの処理に関しては、線図形に対するチェーン符号化方式[29]やDF-表現方式[30]などが抽出した特徴点を用いた処理方式として知られている。もちろん、SECTにおいても変形 $(i, r, c)$ -表現内の選択画素位置を操作することによって、上述の処理を容易に行なうことができる。

この章では、選択画素から求めることのできる画像の形状に関する位相的な性質について詳しく調べ、2値画像の位相的な性質に関する処理、例えば、画像に含まれる連結成分とホールの検出ならびに連結成分の周囲長、面積の計測などを $(i, r, c)$ -表現から行なう方法について論じる。これらの2値画像の位相的な性質についての処理は、それが画像の基本的構造を理解しようとするものだけに、従来数多く試みられている画像処理のなかでも非常に重要であり、その処理形式を

改良することは有意義であると思われる。

5.1 節では、画像の連結性に関する諸定義を正確に与え、その際問題となる C 点の取り扱い方を議論する。5.2 節では、連結成分とホールの同時検出を行なう方法について論じ、新しい検出アルゴリズムを提案する。最後に 5.3 節では連結成分の面積と周囲長の計算方法を与える。

### 5.1 連結性の定義と C 点の取扱い方法

デジタル 2 値画像における連結性の定義の仕方には、4 - 連結性あるいは 8 - 連結性による方法の 2 種類がある[3]。図 5.1 に示すように、4 - 連結性の場合には一つの画素とそれを囲む上下左右の画素とが同じ値をもつかどうかで連結性が定められる。一方、8 - 連結性の場合にはこれに対角方向の 4 画素が更に加えられる。厳密な定義を次に与える。

[定義 5.1]

画素  $(i_1, j_1)$  と  $(i_2, j_2)$  に対し、両方が同じ画素値をもつとする。このとき、

1)  $|i_1 - i_2| + |j_1 - j_2| \leq 1$  を満たすならば、 $(i_1, j_1)$  と  $(i_2, j_2)$  は 4 - 連結の意味で隣接している という。また、

2)  $|i_1 - i_2| \leq 1$  かつ  $|j_1 - j_2| \leq 1$  を満たすならば、 $(i_1, j_1)$  と  $(i_2, j_2)$  は 8 - 連結の意味で隣接している という。

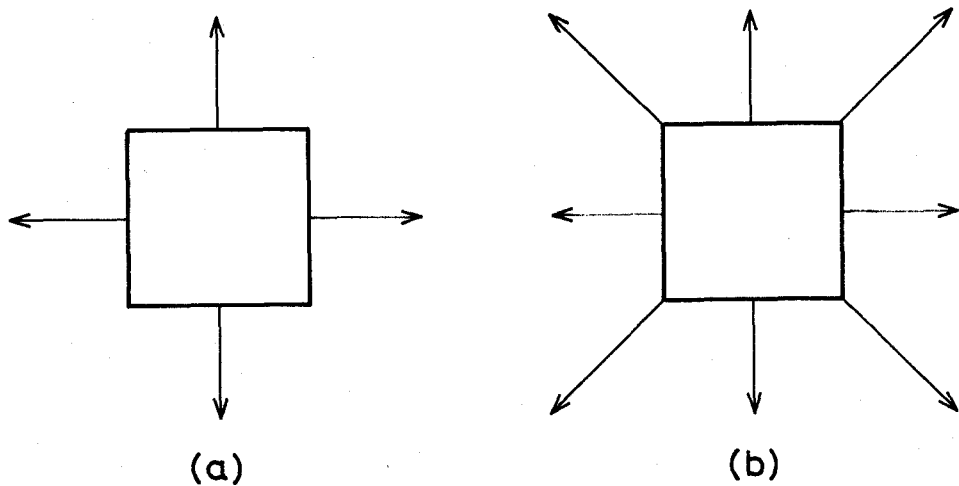


図 5.1 4 - 連結性と 8 - 連結性

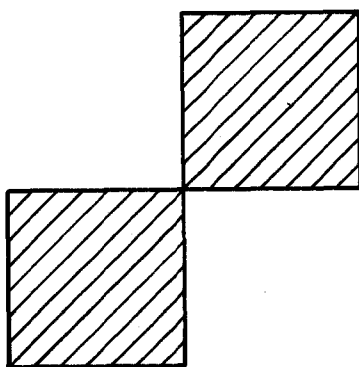


図 5.2 連結成分の例

定義 5.1 の画素間の隣接関係をもちいて、二つの画素の連結性を次のように定める。

[定義 5.2]

画素  $(i_s, j_s)$  と  $(i_E, j_E)$  が 4 - 連結 ( 8 - 連結 ) の意味で 連結している とは、画面  $\{ (i, j) \mid 1 \leq i \leq M, 1 \leq j \leq N \}$  内の画素の適当な長さ  $l$  の組  $((i_1, j_1), (i_2, j_2), \dots, (i_l, j_l))$  で次の条件を満たすものが存在するときで、そのときに限る。

- i)  $(i_1, j_1) = (i_s, j_s)$
- ii)  $(i_l, j_l) = (i_E, j_E)$
- iii)  $(i_t, j_t)$  と  $(i_{t+1}, j_{t+1})$ ,  $1 \leq t \leq l-1$  は 4 - 連結 ( 8 - 連結 ) の意味で隣接している。

定義 5.2 から連結成分およびホールを次のように定める。

[定義 5.3]

画素値が 1 (0) で、互いに 4 - 連結の意味で連結する画素の極大集合を 4 - 連結の意味での 1 - 連結成分 ( 0 - 連結成分 ) という。同様にして、8 - 連結の意味での 1 - 連結成分・0 - 連結成分を定義することができ。

慣例にしたがい、以後は 1 - 連結成分を単に 連結成分、0 - 連結成分を ホール と呼ぶことにする。

[例 5.1]

図 5.2 において、4 - 連結の意味では、二つの連結成分が存在するが、8 - 連結の意味では連結成分の個

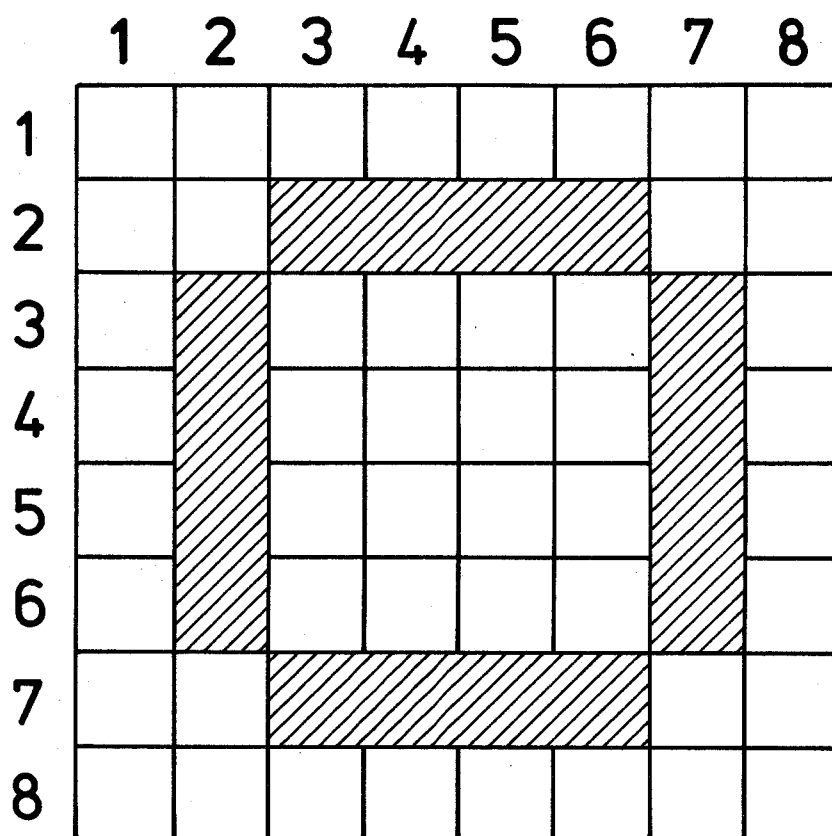
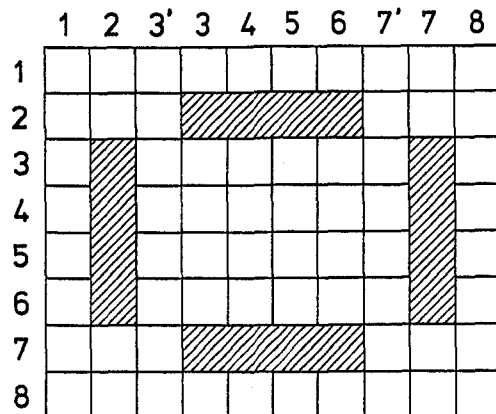


図 5.3 パラドックスを有する“輪”

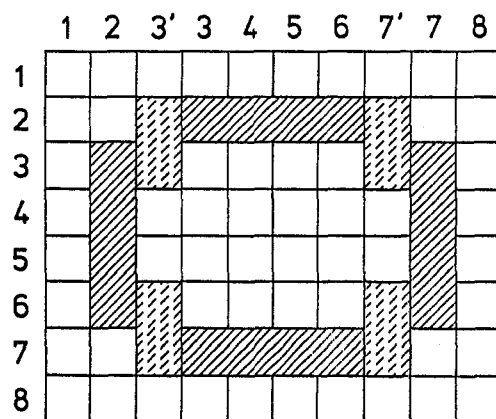
数はただ一つである。

従来，2値画像の位相的性質に関する研究においては，連結成分の4-連結性と8-連結性を明確に区別することなく議論が行なわれている場合がしばしばあった[32][33]。しかし，連結成分とホールを同時に扱うためには，連結性の定義の仕方に注意する必要がある。というのは，連結成分とホールを画一的に4連結性あるいは8-連結性で定義しようとするとき，我々がなじんでいる幾何学的直観に合わないことが生じうるからである。例として，図5.3の画像を考えてみる。まず，4-連結性の場合には，図の斜線で示した“輪”の部分はつながっていないので，直観的には輪の内側と外側は連結しているべきであるが，4-連結による定義によって，輪の内側と外側は離れている。一方，8-連結性の場合にも同様のパラドックスが生じる。すなわち，8-連結による定義によれば，輪は連結しているので直観的には輪の内側と外側は離れているべきであるが，実際は定義から連結してしまうことになる。そこで，以後は常に，連結成分とホールを互いに異なる連結性によって定義することにする。すなわち，連結成分が4-連結（8-連結）の意味で定義される場合には，ホールは自動的に8-連結（4-連結）の意味で定義されるものとする。

連結成分を4-連結の意味で定義するか8-連結の意味で定義するかによって，交差点（C点）の意味に



(a) 4 - 連結性による解釈



(b) 8 - 連結性による解釈

図 5.4 C点の取扱い方

|       | 4-連結    | 8-連結    |
|-------|---------|---------|
| 1-交差点 | CR / TI | TR / CI |
| 0-交差点 | TR / CI | CR / TI |

各連結性における交差点の解釈

表 5.1



違いが生じる。この違いを明確にするために、図5.3の画像の変形として第2列と第3列の間と、第6列と第7列の間にそれぞれ3'列と7'列を挿入した図5.4に示す二つの画像を考える。このとき、4-連結(8-連結)の意味での図5.4 a(同図b)に含まれる連結成分の連結性は図5.3の解釈と一致している。すなわち、この変形を行なっても画像の連結性に関する性質は保存される。また、この変形から選択画素C点を変形後の画像におけるR点とI点が元の画像において重なったものとして解釈することができる。この解釈は連結成分を8-連結の意味で定義した場合、画素値が0のC点はCR点とTI点とが重なったものとして解釈される。一方、画素値が1のC点はTR点とCI点が重なったものとして解釈される。連結成分を4-連結の意味で定義する場合は、この解釈は逆転する。連結性と画素値に対するC点の解釈の仕方を表5.1にまとめておく。

## 5.2 連結成分とホールの検出

### 5.2.1 問題の記述と歴史

本節で扱う連結成分とホールの検出とは、2値画像を対象として画像の連結成分ならびにホールの個数を計数するあるいはそれらの位置情報を求めることを意味する。連結成分の検出は2値画像の位相的性質に関

する処理として、もっとも基本的かつ重要なものである。また、応用面においても細胞や染色体の自動計測などに古くから用いられている[34]。

連結成分の検出方法としては、連結成分の境界部分を連結成分の形状に沿いながら境界線を追跡してゆく境界追跡方式[35]~[38]と、連結性を保ちながら連結成分を収縮してゆく収縮方式のもの[39]~[43]、また、波の伝播をもちいて連結成分を検出する手法[44]などがある。

ここでは、選択画素から求めることのできる元の画像の位相的な性質を調べ、一つの境界追跡方式による連結成分とホールの検出アルゴリズムを導出する。

境界追跡方式はその処理形態から次の2種類に分けられる。一つは、連結成分の境界そのものを連結成分の形状に沿いながら追跡し開始点に戻った時点で一つの連結成分の存在を知る境界依存型のもの[35]である。もう一つは、画面に含まれる連結成分とホールの個数や形状にかかわらず、一定の順序で画面を走査しながら、すべての連結成分を同時に追跡してゆく1パス走査方式である。

計算法処理の面から両方式を比較してみよう。境界依存方式をもちいるためには画面全体を一度に主記憶装置へ入力する必要がある。このため、一般に、非常に大きな記憶容量が必要となり、使用は大型計算機に限られる。一方、1パス走査方式では走査方向にしたがって行単位の処理を行なうので、画面全体を一度に

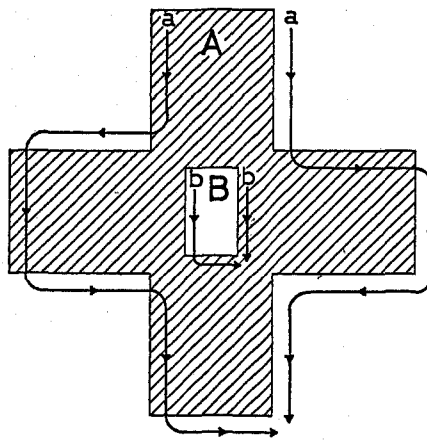
入力する必要はなく外部記憶装置から部分画面を一定間隔ごとに入力すればよい。したがって、1パス走査方式は小型の計算機でも処理が可能である。このように1パス走査方式は処理システムの融通性に富んでいる。

1パス走査による連結成分の検出は最初 Selkow[36]によって論じられた。彼のアイデアは、一つの連結成分に含まれる画素には同じラベルを貼付けてゆこうとするものであった。しかしながら、画面の大きさを縦横  $N \times N$  とした場合、彼の方法ではすべてのラベルを表現するのに高々  $2 \log N$  ビット必要で、しかも最悪の場合、各画素に異なるラベルが貼られたとすると  $2N^2 \log N$  に比例した記憶容量が必要となる。この点は Rosenfeld & Milgram[37]によって改善された。彼らは  $N$  に比例した記憶容量で2値画像の連結性の判定(画像中の連結成分が一つか否かの判定)が行えることを示した。しかしながら、これらの1パス走査形境界追跡方式では連結成分とホールの同時検出を行うことができない。

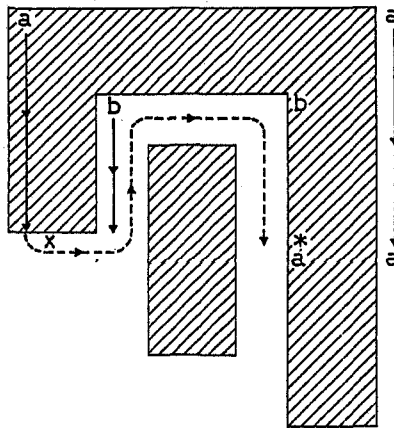
一方、収縮方式では連結成分とホールを同時に検出するアルゴリズムが幾つか知られている[41][42][43]。そこで、次節以下では連結成分とホールの同時検出を1パス走査形境界追跡方式で、しかも、 $N$  に比例した作業領域のみを用いて行えることを示す。

### 5.2.2 (i, r, c)-表現からの連結成分とホールの同時検出

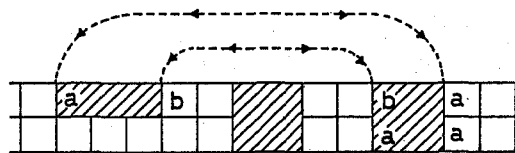
1パス走査形式で連結成分とホールを同時検出するための基本原理を図5.5aに示す。検出のための手順はまず連結成分あるいはホールの最上部の左端と右端にあるマークをおく。図5.5aでは連結成分Aの最上部の両端にはaというマークが、ホールBの最上部の両端にはbというマークがおかれている。このマークを境界部分に沿って下の方へ進めてゆく。同じ種類のマーク同士が同一の底辺にたどり着いた時点で連結成分あるいはホールが検出される。一般の連結成分やホールの場合にはこの操作は少し複雑になる。例えば図5.5bにおいてはマークaとbが底辺 $\alpha$ に到達している。この場合、連結成分の左端から境界を追跡していたマークaは底辺 $\alpha$ を経由して\*印で示した地点まで進むことができる。\*印の点から追跡を続行するために $\alpha$ におけるマークを両方とも消去し、もう一方のマークbをaに変える必要がある。このマークの変更は画面全体をながめることができれば容易に行える。しかし図5.5cのように局所的な画像データが与えられているだけの場合でもマークの種類を参考にして行うことができる。実際、図5.5cではたとえ同種のマーク間の画素がどんな値をとろうとも、マークの位置関係から点線で示すように追跡が行われてきたことが分る。したがって $i+1$ 行において列 $j_3$ に位置するマー



(a)



(b)



(c)

図 5.5 境界の同時追跡

クは $b$ から $a$ に変更すればよい。他にも色々な連結成分とホールが考えうるが、基本的には図5.5cと同じ方法で局所的な画像データを一定順序に用いながら境界の追跡を正しく続けてゆくことができる。このことは1パス走査形式の処理が可能であることを意味している。この節では以上の基本動作にもとづき、 $(i, r, C)$ -表現から連結成分とホールを同時検出するアルゴリズムを与える。さらにアルゴリズムの正当性を証明する。

以下では $C$ 点の取扱い方を明確にするために次の前提を設ける。

[前提5.1]

連結成分は8-連結の意味において定義する。  
なお連結成分が4-連結の意味で定義される場合はこの節の最後で議論する。

[アルゴリズム5](連結成分とホールの同時検出アルゴリズム)

入力: 2値画像 $P$ に含まれる $i$ 点,  $r$ 点,  $C$ 点の位置 $(I_n, J_n)$ とその種類 $X_n (= i, r, C)$ , さらに $p(I_n, J_n)$ の値 $V_n$ ,  $n = 1, \dots, k$ . ここで $X_n$ の値は $i, r, C$ に対応した整数値であるが便宜上,  $i, r, C$ をそのまま用いている。

出力: 連結成分の個数 $CCOUNT$ とホールの個数 $HCOUNT$ である。

方法:  $d_p$ とその画素値 $\{V_n, n=1, \dots, k\}$ から画像の局所的

な形状を逐次に求め、それをもとにマークの位置を変更することにより連結成分とホール境界を追跡してゆく。マークの位置を記憶するために長さ  $N+1$  の 1 次元配列  $B$  を用いる。  $B$  の配列要素としては  $-1, 0, 1$  のいずれかの値をとる。値  $1$  または  $-1$  がマークに対応している。値  $1$  は連結成分あるいはホールの最上部の左端から、値  $-1$  は右端から追跡が始まっていることを示す。どのマークとどのマークが同じ連結成分あるいはホールの最上部から追跡を開始しているかは、後述するように、手続き  $GOAL$  を用いて  $B$  の値だけから同定できる。アルゴリズムを図 5.6 に示す。主プログラム中では  $GOAL$  以外にも  $MOVE 1$ ,  $MOVE 2$ ,  $PAIR$  の手続きを使用する。各手続きの動作を以下に説明しておく。

$GOAL(J) \cdots$

$B[J] = 1$  ならば  $\min \{k \mid \sum_{j=J}^k B[j] = 0, k > J\}$  を返す。

$B[J] = -1$  ならば  $\max \{k \mid \sum_{j=k}^J B[j] = 0, k < J\}$  を返す。

$MOVE 1(i, j) \cdots B[i]$  を  $B[j]$  に変え、 $B[j]$  を  $0$  とおく。

$MOVE 2(i, j, COLOR) \cdots$

$B[i]$  と  $B[j]$  の値に応じて境界の追跡を継続するかそれとも連結成分あるいはホールが検出されたかを判断し、それぞれの処理を行う。判断方法ならびに処理の詳細は後述する。 $COLOR$  は連結成分かホールかを示す指標で、 $COLOR = 0$  なら連結成分、 $1$  ならホールが発見されたことを表わす。

```

1  for n = 1 until K do
    begin
2      if  $X_n = i$  then
3          if  $B[J_n] \neq 0$  then MOVE 2( $J_n, \text{PAIR}(n), V_n$ )
              else begin
4                  if  $((I_n, \text{PAIR}(n)) = (I_{n+1}, J_{n+1})) \wedge (X_{n+1} = r)$ 
5                      then  $B[J_n]$  と  $B[J_{n+1}]$  をそれぞれ 1, -1 とおく ;
6                  if  $((I_n, \text{PAIR}(n)) = (I_{n+1}, J_{n+1})) \wedge (X_{n+1} = c) \wedge (V_{n+1} = 0) \vee$ 
 $((I_n, \text{PAIR}(n)) \neq (I_{n+1}, J_{n+1}))$  then MOVE 1( $J_n, \text{PAIR}(n)$ )
                        end;
7          if  $X_n = r$  then
8              if  $(I_n, \text{PAIR}(-n)) \neq (I_{n-1}, J_{n-1})$  then MOVE 1( $J_n, \text{PAIR}(-n)$ );
9          if  $X_n = c$  then
10             if  $V_n = 0$  then
11                 begin
12                     if  $(I_n, \text{PAIR}(-n)) \neq (I_{n-1}, J_{n-1})$  then MOVE 2( $\text{PAIR}(-n), J_n, 1-V_n$ );
13                      $X_n \leftarrow i$ ;
14                      $n \leftarrow n - 1$ 
15                 end
16             else
17                 begin
18                     OLD  $\leftarrow B[J_n]$ ;
19                     if  $(I_n, \text{PAIR}(-n)) \neq (I_{n-1}, J_{n-1})$  then MOVE 1( $J_n, \text{PAIR}(-n)$ )
20                     else  $B[\text{PAIR}(-n)]$  と  $B[J_n]$  をそれぞれ 1, -1 とおく ;
21                     NEW  $\leftarrow B[J_n]$ ;
22                      $B[J_n] \leftarrow \text{OLD}$ ;
23                     if  $((I_n, \text{PAIR}(n)) = (I_{n+1}, J_{n+1})) \wedge (X_{n+1} = r)$  then
24                         begin
25                             MOVE 1( $J_{n+1}, J_n$ );
26                              $B[J_n] \leftarrow \text{NEW}$ ;
27                              $n \leftarrow n + 1$ 
28                         end
29                     else if  $(B[J_n] = -1) \wedge (\text{NEW} = 1)$  then MOVE 1( $J_n, \text{PAIR}(n)$ )
30                     else begin
31                         MOVE 2( $J_n, \text{PAIR}(n), V_n$ );
32                          $B[J_n] \leftarrow \text{NEW}$ 
33                     end
34                 end
35             end
36         end
37     end
38 end

```

図 5.6 アルゴリズム 5; 連結成分とホールの同時検出(次ページへ続く)



```

procedure GOAL(J)
begin
1   SUM ← B[J];
2   j ← J;
3   if SUM = 1 then repeat j ← j+1 until (SUM ← SUM + B[j]) = 0
4   else repeat j ← j-1 until (SUM ← SUM + B[j]) = 0;
5   return(j)
end

procedure MOVE 1(i,j)
begin
1   B[i] ← B[j];
2   B[j] ← 0
end

procedure MOVE 2(i,j,COLOR)
begin
1   if (B[i] = 1) ∧ (B[j] = -1) then
2       if COLOR = 0 then CCOUNT ← CCOUNT + 1
3       else HCOUNT ← HCOUNT + 1;
4       if (B[i] = 1) ∧ (B[j] = 1) then B[GOAL(j)] ← 1;
5       if (B[i] = -1) ∧ (B[j] = -1) then B[GOAL(i)] ← -1;
6       B[i] ← 0;
7       B[j] ← 0
end

procedure PAIR(n)
begin
1   k ← In;
2   if n < 0 then repeat k ← k-1 until ( (B[k] ≠ 0) ∨ ((In,k) = (In-1,Jn-1)) )
3   else repeat k ← k+1 until ( (B[k] ≠ 0) ∨ ((In,k) = (In+1,Jn+1)) );
4   return(k)
end

```

## 図 5.6 手続き

PAIR( $n$ ) ....

$n$  の値が正ならば,  $J_n^f$  の探索を行い,  $J_n < j \leq J_{n+1}$  なるある  $j$  を返す。また  $n$  の値が負ならば,  $J_n^b$  の探索を行い,  $J_{n-1} \leq j < J_n$  なるある  $j$  を返す。

[定理 5.1]

アルゴリズム 5 にはある  $(i, r, C)$ -表現  $d_p$  とその画素値  $\{V_n, n=1, \dots, K\}$  から元の画像  $P$  に含まれる連結成分とホール数を正確に求めることができる。

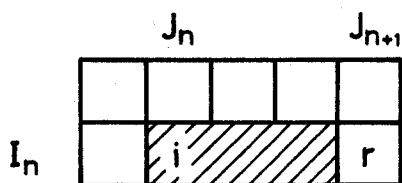
(証明)

しばらくのあいだ画像  $P$  には  $C$  点は含まれていないものと仮定しておく。さて  $(I_n, J_n)$  を  $(i, r, C)$ -表現  $d_p$  に含まれるある選択画素とする。仮定より  $X_n = i$  または  $r$  である。このとき  $(I_n, J_n)$  附近の  $P$  の画素値は, 0 と 1 の反転画像を除けば図 5.7 に示す 4 種類の場合に分類される。ここで  $P$  の第  $I_1$  行上では図 5.7a の場合のみが生じうることに注意する。したがって  $I_1$  行上には偶数個の選択画素  $(I_1, J_1), \dots, (I_{2\ell}, J_{2\ell})$  が存在し, しかも

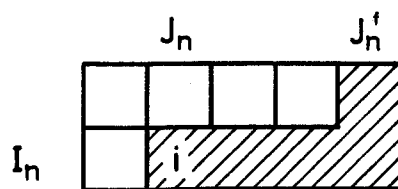
$$X_{2t-1} = i$$

$$X_{2t} = r, \quad 1 \leq t \leq \ell \quad \text{である。}$$

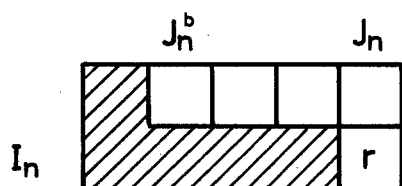
そこでまず  $I_1$  行の選択画素  $(I_n, J_n)$ ,  $1 \leq n \leq 2\ell$  に対してアルゴリズム 5 の動作を調べよう。アルゴリズム 5 のステップ 1 において  $n=1$  の場合,  $X_1 = i$  なのでステップ 3 の if 文が実行される。B の値はすべて 0



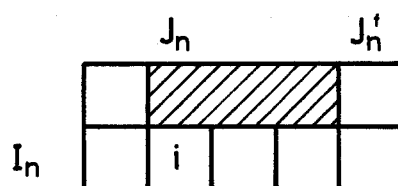
(a)



(b)



(c)



(d)

図 5.7  $i$  点,  $r$  点のまわりの  
画素値

に初期値化されているので、次にステップ4のif文が実行される。手続き PAIR の動作から PAIR(1)として  $J_2$  が返されるので、結局

$$B[J_1] = 1, \quad B[J_2] = -1$$

となる。次に  $n = 2$  の場合は、 $X_2 = r$  よりステップ6のif文の実行後、引き続いてステップ7のif文が実行される。手続き PAIR は PAIR(-2)として  $J_1$  を返すのでif文の条件は成立せずthen以下の文は実行されない。

$\ell > 1$  の場合は、上の動作が  $n = \ell$  までくり返される。その結果、配列  $B$  は

$$B[J_{2t-1}] = 1, \quad B[J_{2t}] = -1, \quad 1 \leq t \leq \ell$$

$$B[j] = 0, \quad j \neq J_n, \quad 1 \leq n \leq 2\ell$$

となる。これは連結成分の最上部の左端に1、右端に-1というマークが置かれたことを意味している。

ところで次に選択画素が存在する行  $I_{2\ell+1}$  に対し、もし  $I_{2\ell+1} > I_{2\ell} + 1$  ならば区間  $[(I_{2\ell}+1, 1), (I_{2\ell+1}-1, N+1)]$  には選択画素は存在しないことから

$p(i, j) = p(I_m, j), \quad I_m < i < I_{m+1}, \quad 1 \leq j \leq N$  である。したがって  $I_{2\ell+1} > I_{2\ell} + 1$  または  $I_{2\ell+1} = I_{2\ell} + 1$  のいずれの場合にせよ、 $I_{2\ell+1} - 1$  行上の変化点位置は  $I_{2\ell} (= I_1)$  行上の変化点位置と完全に一致している。このことは  $I_1$  行上で連結成分の両端にマークを置くだけで  $I_{2\ell+1} - 1$  行まで自動的に追跡が行われることを意味する。

$I_{2\ell+1}$  行上のある  $i$  点  $(I_n, J_n)$  附近の  $P$  の画素値が図 5.7a

の場合は、上と同様に連結成分あるいはホールの最上部にマークを置く動作がアルゴリズムらによって行われる。 $I_{2\ell+1}$ 行上のある $i$ 点 $(I_n, J_n)$ 附近の $P$ の画素値が図5.7bの場合、前行までの処理によって $(I_{n-1}, J_n^f)$ まで追跡が進んでいるので、 $B[J_n^f] \neq 0$ である。したがって $J_n^f$ の値は $B$ と手続きPAIRを用いてPAIR( $n$ )で求めることができる。また今の場合、 $(I_n, \text{PAIR}(n)) < (I_{n+1}, J_{n+1})$ なのでステップ6のMOVE1が実行されて、マークの位置が境界に沿って移動する。なおここで $J_n^f$ は第3章の定理3.5において定義された値で、

$$J_n^f = \min \left\{ j \mid (I_{n-1}, j) \text{は変化点, } \begin{matrix} (I_{n-1}, J_n) < (I_{n-1}, j) \\ < (I_{n+1}-1, J_{n+1}) \end{matrix} \right\}$$

である。また $I_{2\ell+1}$ 行上のある $r$ 点 $(I_n, J_n)$ 附近の $P$ の画素値が図5.7cの場合、前行までの処理によって $(I_{n-1}, J_n^b)$ まで追跡が進んでいるので、 $B[J_n^b] \neq 0$ である。したがって $J_n^b$ の値は $B$ と手続きPAIRを用いてPAIR( $-n$ )で求めることができる。また今の場合、 $(I_n, \text{PAIR}(-n)) > (I_{n-1}, J_{n-1})$ なのでステップ8においてMOVE1が実行されて、マークの位置が境界に沿って移動する。なおここで $J_n^b$ は $J_n^f$ と同じく定理3.5において定義された値で、

$$J_n^b = \max \left\{ j \mid (I_{n-1}, j) \text{は変化点, } \begin{matrix} (I_{n-1}-1, J_{n-1}) < \\ (I_{n-1}, j) < (I_{n-1}, J_n) \end{matrix} \right\}$$

である。

$I_{2\ell+1}$ 行以後の選択画素に対しても、図5.7a, b, cに

対応するものが現われた場合，アルゴリズムらは上述の動作と同じく，連結成分およびホールの最上部を見つけ出して左側と右側の境界を同時に追跡する。問題は，ある  $i$  点  $(I_n, J_n)$  附近の  $P$  の画素値が図 5.7d で示す場合，すなわちアルゴリズムらのステップ 3 で  $MOVE2$  が実行される場合の動作の正当性を示すことである。この  $i$  点  $(I_n, J_n)$  においては前行までの処理によって， $B[J_n]$  と  $B[J_n^f]$  はいずれも 0 でない値をとる。さらに  $PAIR(n) = J_n^f$  である。

ところで  $I_n$  行において， $B[j] = 1$  ならば，画素  $(I_n, j)$  は第 1 行から第  $I_n$  行までの範囲内で境界をたどることによりある連結成分あるいはホールの上端の左側に到達できる。また  $B[k] = -1$  ならば，画素  $(I_n, k)$  は同じ範囲内である連結成分あるいはホールの上端の右側に到達できる。画素  $(I_n, j)$  と  $(I_n, k)$  が第 1 行から第  $i$  行までの範囲である連結成分あるいはホールの上端を経由して境界部分をたどることによって互いに到達できることを単に到達可能という。連結成分とホールは入れ子状にしか配置し得ないことより，どの  $B[j] = 1$  なる  $(I_n, j)$  と，どの  $B[k] = -1$  なる  $k$  とが到達可能であるかの判定は次のようにして行うことができる。すなわち  $B[j] = 1$  なる  $(I_n, j)$  に対しては，ある区間  $[j, t]$  内で  $B$  の 1 と -1 の数が等しくなる最小の  $t$  を求める。この  $(I_n, t)$  が  $(I_n, j)$  から到達可能な画素である。また  $B[k] = -1$  なる  $(I_n, k)$  に対しては，ある区間  $[s, k]$  内で  $B$  の

1 と -1 の数が等しくなる最大の  $S$  を求める。このとき  $(I_n, S)$  が  $(I_n, k)$  から到達可能な画素である。アルゴリズム 5 ではこれらの操作は手続き GOAL によって行われる。

さて図 5.7d に対応する  $i$  点  $(I_n, J_n)$  の処理に戻って議論を続けよう。もし  $B[J_n] = 1$  かつ  $B[J_n^f] = -1$  ならば、上の議論から  $(I_n, J_n)$  と  $(I_n, J_n^f)$  は互いに到達可能なので、一つの連結成分あるいはホールが存在する。いずれであるかは  $V_n$  の値によって判定される。これ以外の場合は、 $(I_n, J_n)$  から到達可能な画素  $(I_n, \text{GOAL}(J_n))$  と  $(I_n, J_n^f)$  から到達可能な画素  $(I_n, \text{GOAL}(J_n^f))$  とが互いに到達可能であるので、次に MOVE2 のステップ 1 の if 文の条件が成立するまで  $(I_n, \text{GOAL}(J_n))$  と  $(I_n, \text{GOAL}(J_n^f))$  から境界の追跡を続けてゆく必要がある。このときあとの処理で連結成分を正確に検出するために  $(I_n, \text{GOAL}(J_n))$  と  $(I_n, \text{GOAL}(J_n^f))$  のうち左側に位置するものに対する  $B$  の値を 1 に、右側に位置するものに対する  $B$  の値を -1 にしておく必要がある。このためには、もし  $B[J_n] = 1$  かつ  $B[J_n^f] = 1$  ならば  $B[\text{GOAL}(J_n^f)]$  を 1 に変え、またもし  $B[J_n] = -1$  かつ  $B[J_n^f] = -1$  ならば  $B[\text{GOAL}(J_n)]$  を -1 に変えればよい。最後にもし  $B[J_n] = -1$  かつ  $B[J_n^f] = 1$  ならば、 $B[\text{GOAL}(J_n)]$  と  $B[\text{GOAL}(J_n^f)]$  の値はそのままよい。手続き MOVE2 のステップ 4 と 5 ではこれらの要求を満たす処理を行っている。さらに MOVE2 のステップ

6と7では、 $(I_n, J_n)$ と $(I_n, J_n^+)$ においてはすでに追跡すべき境界は存在しないことから、 $B[J_n]$ と $B[J_n^+]$ とを0にする動作を行う。

以上の処理を  $I_{2k+1}$  行以後の選択画素に対してくり返し行うことによって、最終的にはすべての連結成分とホールの数をアルゴリズム5から求めることができる。証明を完全にするためにはC点に対するアルゴリズムの処理の正当性を示すことが最後に残っている。

あるC点  $(I_n, J_n)$  附近の画像Pの画素値は図5.8に示す(a)から(d)の四つの場合に分類される。前提5.1から連結成分は8-連結の意味で定義されているので、5.1節におけるC点の解釈に従えば、図5.8の(a)~(d)は画像の連結性を保ったまま同図の(a')~(d')に変形できる。以下ではこのC点の解釈を参照にしながら、(a)~(d)のそれぞれの場合においてアルゴリズム5が行う処理の正当性を示す。

1) C点  $(I_n, J_n)$  が図5.8(a)に対応する場合 ( $V_n = 0$ )

前行までの処理によって  $B[J_n^b]$  と  $B[J_n]$  まで境界の追跡が行われているはずなので、

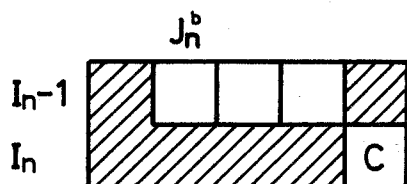
$B[J_n^b] \neq 0$  かつ  $B[i] = 0$ ,  $J_n^b < i < J_n$ ,  $B[J_n] \neq 0$  でなければならない。さらにこのとき

$$\text{PAIR}(-n) = J_n^b$$

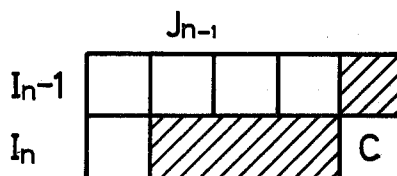
である。したがってアルゴリズム5ではステップ11のif文の条件が成立し、 $\text{MOVE 2}(\text{PAIR}(-n), J_n, 1-V_n)$  が実行され、さらにステップ12~13でi点に対する処



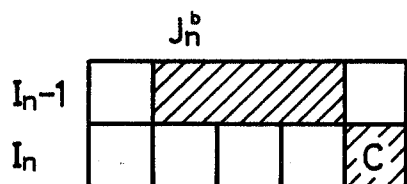
$$V_n = 0$$



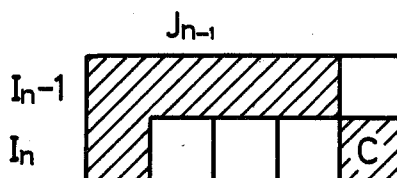
(a)



(b)

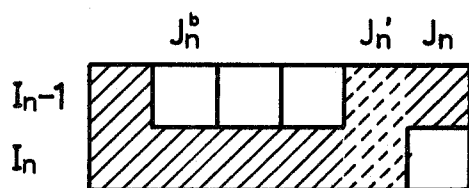


(c)

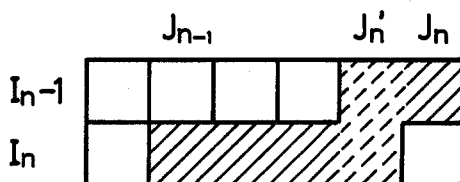


(d)

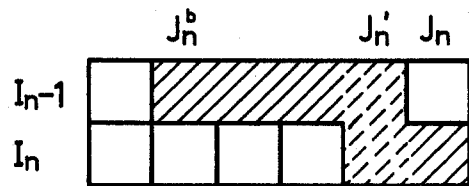
$$V_n = 1$$



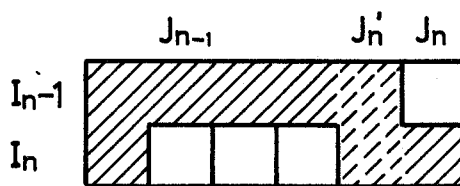
(a)'



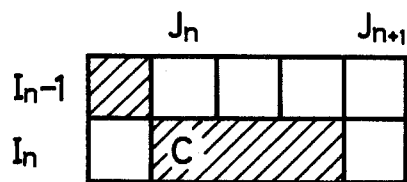
(b)'



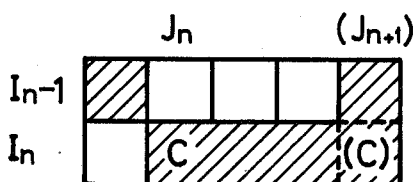
(c)'



(d)'



(e)



(f)

図 5.8 C 点の解釈

理が行われる。これは図 5.8(a)' の解釈から要求される動作を満足している。

2) C点  $(I_n, J_n)$  が図 5.8(b) に対応する場合 ( $V_n = 0$ )

この場合は  $\text{PAIR}(-n) = J_{n-1}$  なのでステップ11のif文の条件は成立しない。よってthen以下の文は実行されず、ステップ12~13のみが実行される。一方  $(I_{n-1}, J_{n-1})$  の処理ではステップ6のif文の条件が成立するので、MOVE 1 が実行されていなければならない。これらの動作は図 5.8(b)' の解釈から要求される動作を満足する。

3) C点  $(I_n, J_n)$  が図 5.8(c) に対応する場合 ( $V_n = 1$ )

この場合は図 5.8(a) において 0 と 1 を反転したものに等しいので、1) と同じ議論から

$B[J_n^b] \neq 0$ ,  $B[j] = 0$ ,  $J_n^b \leq j < J_n$ ,  $B[J_n] \neq 0$ ,  
かつ  $\text{PAIR}(-n) = J_n^b$  である。したがってアルゴリズム  
5ではステップ15のif文の条件が成立し、MOVE 1  
が実行される。図 5.8(c)' からは  $(I_{n-1}, J_n^b)$  に置かれた  
マークを  $(I_n, J_n)$  に移動する必要があることが分る  
が、上の動作はこの要求を満たしている。ところで元  
の  $B[J_n]$  の値は以下の処理で必要なので、あらかじめ  
ステップ14において変数 OLD に代入されている。そ  
してステップ17と18ではMOVE 1 の実行後の  $B[J_n]$  の  
値を変数 NEW に入れ、元の値を  $B[J_n]$  に戻している  
ことに注意する。したがって今の時点では NEW には  
 $B[J_n^b]$  の値が、 $B[J_n]$  には元の値 OLD が入っている。

さて、 $(I_n, J_n)$  以後の画像  $P$  の画素値は次の二つの場合に分類される。すなわち図 5.8(e) のように  $r$  点が存在するか、あるいは図 5.8(f) のように  $CR$  点または  $C$  点が存在するかである。ここで図 5.8(f) における点線は  $C$  点でありうることを示している。図 5.8(e) の場合は、前行までの処理によって、

$$B[J_n] \neq 0, \quad B[j] = 0, \quad J_n < j \leq J_{n+1} \quad \text{かつ} \\ \text{PAIR}(n) = J_{n+1}$$

が成立していなければならない。また  $X_{n+1} = r$  なのでステップ 18 の if 文の条件が成立して  $\text{MOVE } 1$  が実行される。これは  $(I_{n-1}, J_n)$  に置かれたマーク  $B[J_n]$  を  $B[J_{n+1}]$  に移動するために必要な動作である。一方、(f) の場合、ステップ 18 の条件は成立しないので else 以下の文が実行される。ところで図 5.8(c) においてもし、 $B[J_n^b] (= \text{NEW}) = 1$  かつ  $B[J_n] = -1$  ならば、図 5.8(c) による解釈から  $(I_{n-1}, J_n)$  は  $(I_n, J_n)$  に到達可能である。このことは  $(I_{n-1}, \text{PAIR}(n))$  が  $(I_n, J_n)$  に到達できることを意味しており、 $(I_{n-1}, \text{PAIR}(n))$  のマークを  $(I_n, J_n)$  に移動する必要がある。これはステップ 22 によってなされる。また  $B[J_n^b], B[J_n]$  が上記以外の場合は、ホールの最下部に達した際の動作を直接行えばよい。ステップ 23 ~ 24 ではこの要求を満たす動作を行っている。

4)  $C$  点  $(I_n, J_n)$  が図 5.8(d) に対応する場合 ( $V_n = 1$ )

このとき図 5.8(d) から明らかのように  $(I_{n-1}, J_{n-1})$  は

$i$  点である。ところでアルゴリズムが  $(I_{n-1}, J_{n-1})$  を処理した際には、前行までの処理によって

$$B[J_n] \neq 0, \quad B[j] = 0, \quad J_{n-1} \leq j < J_n \quad \text{かつ}$$

$$\text{PAIR}(n-1) = J_n$$

である。したがって、ステップ4のif文さらにステップ6のif文の条件のいずれも成立しないので $B$ の値は変化しておらず、今の時点でもやはり

$$B[J_n] \neq 0, \quad B[j] = 0, \quad J_{n-1} \leq j < J_n$$

が成立し、さらに  $(I_n, \text{PAIR}(n)) = (I_{n-1}, J_{n-1})$  である。したがってステップ15のif文の条件は成立しないので、ステップ16のelse以下の文が実行されて、

$$B[J_{n-1}] = 1, \quad B[J_n] = -1$$

となる。一方図5.8(d)'の解釈によればこの時ホールの最上部の両端にマークが置かれなければならないが上記の動作はこの要求を満たしている。ステップ18以後のアルゴリズムの処理は3)の場合と同じである。図5.8(c)'と(d)'を見比べればこの処理が必要にして十分なものであることが分る。(証明終り)

最後に、連結成分が4-連結性の意味において定義されている場合に連結成分とホールを検出する方法について述べておこう。

4-連結性の意味で定義された連結成分を検出するためには、表5.1における連結性とC点との対応関係から、0と1を反転した画像の  $(i, r, C)$ -表現とそれに含まれる選択画素の画素値に対してアルゴリズムら

を実行すればよい。すなわち画像  $P$  の  $d_p$  に含まれる選択画素の値  $\{V_n, n=1, \dots, K\}$  をすべて反転してからアルゴリズム 5 を実行するのである。あるいは  $V_n$  の反転は行わずに、アルゴリズム 5 のステップ 6 における if 文の条件内の  $V_{n+1}=0$  を  $V_{n+1}=1$  に、そしてステップ 10 における if 文の条件を  $V_n=0$  から  $V_n=1$  に変更するという方法もある。いずれの方法を用いるにせよ、4-連結性の意味での連結成分の検出は、8-連結性の場合の検出アルゴリズムをほんの少し修整するだけで行える。

### 5.3 連結成分およびホール周囲長と面積の計算

前節で導出したアルゴリズム 5 を少し変更することにより、 $(i, r, C)$ -表現から連結成分およびホール周囲長と面積を計測することができる。以下にそのアルゴリズムを示しておく。

[アルゴリズム 6]: 連結成分およびホール周囲長の計測

入力: アルゴリズム 5 の場合と同じく 2 値画像  $P_x$  に含まれる  $i$  点,  $r$  点,  $C$  点の位置  $(I_n, J_n)$  とその種  $X_n$  ( $= i, r, C$ ), さらに  $p(I_n, J_n)$  の値  $V_n, n=1, \dots, K$ .

出力: 各連結成分の周囲長のリスト  $CL$  とホール周囲長のリスト  $HL$ .

方法: アルゴリズム 5 の動作を忠実に実行しながら、マークの位置の画面上での移動距離を計算してゆく。アルゴリズム 5 で用いた配列  $B$  の他に、マークが進んだ

移動距離を記憶するために長さ  $N+1$  の 1 次元配列  $L$  を使用する。さらにリスト  $CL$  と  $HL$  の中で用いられるアドレスとして変数  $NC, NH$  を使用する。これらの変数は 0 に初期値化しておく。これらの配列, 変数を用いて具体的には次の処理を行う。

まず連結成分あるいはホールの最上部の両端に境界の追跡を開始するためのマークがおかれた時点で最上部の辺の長さを計算する。このとき左端のマークの位置を  $j$  とすると  $L[j]$  にいま計算した辺の長さを代入する。以下画面上でマークが縦方向および横方向に移動するごとに, その移動距離を  $L$  に記録してゆく。

処理の途中で, アルゴリズム 5.7.d に対して図 5.7.d の場合に対応する条件が成立したとする。このとき左側のマークを  $m_l$ , 右側のマークを  $m_r$  とおく。もしこれらのマークが一つの連結成分あるいはホールの最下部に達しているならば, 両方のマークが移動した距離と底辺の長さを加えた値  $L$  が求める周囲長である。一方, 最下部でなければ, 左側のマーク  $m_l$  から到達できるもう一つのマーク  $a_l$  を見つけて,  $a_l$  がいままで進んできた距離に  $L$  を加えて次の処理に進めばよい。

上の操作を行うアルゴリズムはアルゴリズム 5.7.c に以下の動作を付け加える。

- 1) ステップ 5.7.c において " $L[j_n] \leftarrow j_{n+1} - j_n$ " を追加する。
- 2) ステップ 16 において " $L[PAIR(-n)] \leftarrow j_n - PAIR(-n)$ " を追加する。

```

procedure MOVE 1'(i,j)
begin
1   B[i] ← B[j];
2   L[i] ← L[j] + |i - j|;
3   B[j] と L[j] を 0 とおく。
end

procedure MOVE 2'(i,j,COLOR)
begin
1   L ← L[i] + L[j] + |i - j|;
2   if (B[i] = 1) ∧ (B[j] = -1) then
3       if COLOR = 0 then CL[(NC ← NC + 1)] ← L
4       else HL[(NH ← NH + 1)] ← L;
5   else L[GOAL(i)] ← L[GOAL(i)] + L;
6   if (B[i] = 1) ∧ (B[j] = 1) then B[GOAL(j)] ← 1;
7   if (B[i] = -1) ∧ (B[j] = -1) then B[GOAL(i)] ← -1;
8   B[i],B[j],L[i],L[j] を 0 とする。
end

procedure INCREMENT(I)
for n = 1 until N do
    if B[n] ≠ 0 then L[n] ← L[n] + I

```

図 5.9 アルゴリズム 6 : 連結成分およびホールの  
周囲長の計測

3) ステップ 9 のif 文 (ステップ 9 から 24 まで) の後に  
“ 25 if ( $I_{n+1} > I_n$ ) then INCREMENT( $I_{n+1} - I_n$ ) ,,  
を加える。

ここで手続き INCREMENT( $i$ ) は図 5.9 に示すように  
各マークの縦方向の移動距離  $i$  を  $L$  に記録する動作を  
行う。さらに MOVE1 と MOVE2 をそれぞれ図 5.9 に  
示す MOVE1', MOVE2' に変更する。

### [アルゴリズム 7]: 連結成分およびホールの面積の 計測

入力: アルゴリズム 6 と同じ。

出力: 連結成分の面積のリスト  $CA$  とホールの面積の  
リスト  $HA$  である。

方法: アルゴリズム 5 の動作を忠実に実行しながら、  
各行におけるマーク間の距離を累積してゆくことによ  
り面積を計算する。アルゴリズム 5 で用いた配列  $B$  の  
他にマーク間の距離を記憶するために長さ  $N+1$  の 1  
次元配列  $A$  を使用する。さらに連結成分およびホール  
の周囲長のリスト  $CL$  と  $HL$  の中で用いられるアドレ  
スとしてそれぞれ 0 に初期値化された変数  $MC, MH$   
を使用する。アルゴリズムはアルゴリズム 5 のステッ  
プ 9 における if 文 (ステップ 9 から 24 まで) の後に  
“ 25 if ( $I_{n+1} > I_n$ ) then AREA ( $I_{n+1} - I_n$ ) ,,  
を加える。

ここで手続き AREA は図 5.10 に示す。この手続きは、



```

procedure MOVE 1"(i,j)
begin
1   A[i] と B[i] をそれぞれ A[j] と B[j] とおく ;
2   A[j] と B[j] を 0 とおく
end

procedure MOVE 2"(i,j,COLOR)
begin
1   if (B[i] = 1) ∧ (B[j] = -1) then
2       if COLOR = 0 then CA[MC ← MC + 1] ← A[i]
3       else HA[MH ← MH + 1] ← A[i]
4   else A[GOAL(j)] ← A[GOAL(j)] + A[i];
5       if (B[i] = 1) ∧ (B[j] = 1) then B[GOAL(j)] ← 1;
6       if (B[i] = -1) ∧ (B[j] = -1) then B[GOAL(i)] ← -1;
7       A[PRE(i)] ← A[PRE(i)] + A[j];
8   A[i],A[j],B[i],B[j] をすべて 0 とおく
end

procedure PRE(i)
begin
1   k ← i;
2   repeat k ← k - 1 until (B[k] ≠ 0);
3   return(k)
end

procedure AREA(I)
begin
1   m ← 1;
2   for k = 1 to N + 1
3       if B[k] ≠ 0 then
4           begin
5               A[m] ← A[m] + (k - m) × I;
6               m ← k
7           end;
8   A[m] ← A[m] + (N + 1 - m) × I
end

```

図 5.10 アルゴリズム 7: 連結成分およびホールの面積の計測

ある  $I$  行から  $I + i$  行までの間の部分画像における連結成分またはホールの面積を計算し、その結果を  $I$  行までの連結成分またはホールの面積値に加える動作を行う。さらにアルゴリズムらの手続き  $MOVE1$  と  $MOVE2$  をそれぞれ図 5.10 の  $MOVE1''$  と  $MOVE2''$  に変更する。ここで  $MOVE1''$  と  $MOVE2''$  は境界に沿ってマークが移動する場合、それに応じていままでそのマークに対応していた面積値も同じところに移動させる動作を行う。

## 第6章

### ディジタル濃淡画像の符号化

実際のTV信号などからディジタル濃淡画像を得る場合、標本化・量子化による誤差が元の画像に比べて目立たないようにしようと思えば、標本化周波数によっても異なるが、一般に、量子化の際に少なくとも8ビット( $2^8 = 256$ )の量子化レベルが必要とされる。これは、符号を構成するハードウェア規模の面で、ディジタル濃淡画像を直接符号化することが非常に困難であることを意味している。なぜなら、ディジタル濃淡画像の各画素値は近傍の画素値と非常に強い相関を持つことが知られており、画像データをその最も簡単なモデルである定常1次マルコフ過程として符号化を行なう場合ですら、符号器・復号器を構成するためには、

$256^2 = 64K$  個もの確率パラメータを扱う必要があり、現実的ではないからである。そこでハードウェア規模の小型化、処理の高速化を目指して、さまざまな符号化方式が提案されてきている[45][46][47]。これらの方式はすべて画像データに前処理を施すことによって符号器への入力となるシンボル数の削減を行い、符号器ならびに復号器のハードウェア負担を軽減しようとするものである。

従来はこの前処理として、復元画像と元の画像との間にある程度の誤差を許して一つ一つの画素値をできるだけ少ないシンボル数で表現するという手法が専ら用いられてきた。その代表的なものとして、 $\Delta$ -変調[48]やDPCM[49]、あるいは画像データをアダマール(Hadamard)変換[50]やカルーネン・レーブ(Karhunen-Loève)変換[51]などによって直交展開する変換方式などが知られている。

この章ではこれらの方式とは異なり、元の画像を正確に復元するという制約のもとでシンボル数の削減を行う前処理として、Bit-Plane変換を提案し、さらにこの変換を用いてデジタル濃淡画像を符号化する方法について論じる。以下6.1節ではBit-Plane変換について考察し、実際の画像データに対する有効性を示す。また6.2節ではBit-Plane変換後の符号化方式として算術符号を用いた場合の符号の性能を解析的にならびに計算機シミュレーションから調べる。

## 6.1 Bit-Plane 変換

### 6.1.1 Bit-Plane 変換とは

Bit-Plane変換は、Shwartz×Barker [52]によって、宇宙線の衝突回数測定を行う探査衛星が観測したデータを地上に伝送するための符号化方式の中で初めて用いられた。その概要を図6.1に示す。すなわち、 $M (= 2^L)$

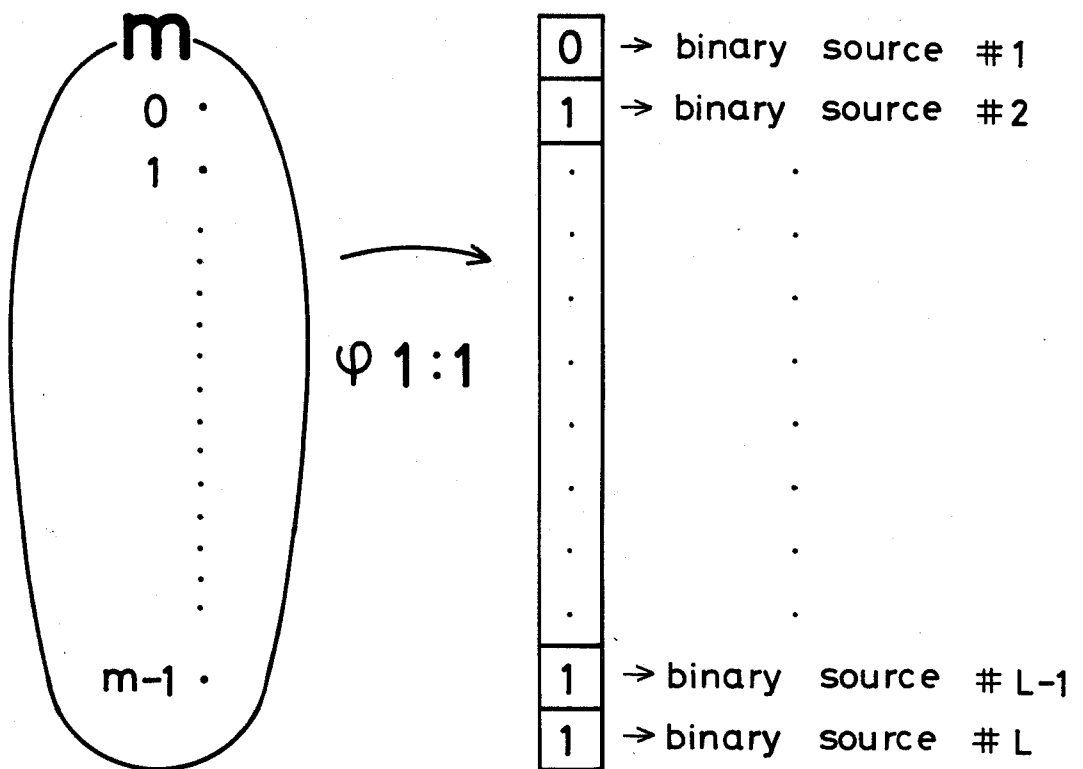


図 6.1 Bit - Plane 変換 の 概 説

個のシンボル0 から  $M-1$  を持つ情報源が与えられたとき、一つのシンボルが生起するたびごとに  $L$  桁の  $0-1$  ベクトルへ1対1変換を行い、各桁の値を0, 1の生起する  $L$  個の互いに独立な2値情報源とみなし、各情報源を個別に符号化する。

このように Bit-Plane 変換は一つの情報源の符号化を  $L$  個の2値情報源の符号化に帰着させるので、個々の2値情報源に対する符号器・復号器の構成は元の情報源のそれに比べて桁違いに容易になる。実際、符号を構成するのに必要な確率パラメータの数は元の情報源では  $2^L-1$  個だったのが、Bit-Plane 変換後の2値情報源では全体として確率パラメータの数は  $L$  個でよい。さらに各2値情報源の符号化を並列処理化することにより高速に符号化することができる。しかしながらその半面、Bit-Plane 変換を用いる符号化方式では、 $L$  個の2値情報源の互いの相関を一切用いないので、Bit-Plane 変換を行った時点で無視できない圧縮効果の劣化を招く。これまでに、画像の符号化に Bit-Plane 変換を用いた例は幾つか報告されているが[53]、Bit-Plane 変換による圧縮効果の劣化に関する考察はほとんど行われていない。そこでこの節の以下では、ある形を持った情報源に対しては変換をうまく選ぶことによって圧縮効果の劣化が問題にならないほど低く抑えられることを示し、画像データに対しどのような変換を選べばよいかを考察する。

### 6.1.2 問題の定式化

有限集合  $\mathcal{M} = \{0, 1, \dots, M-1\}$  (ここで  $M = 2^L$ ,  $L$  は正整数) の中に値をとる確率分布  $\{p(m), m \in \mathcal{M}\}$  をもつ独立な情報源  $(\mathcal{M}, p)$  を考える。この情報源  $(\mathcal{M}, p)$  のエントロピーは,

$$H = \sum_{m \in \mathcal{M}} p(m) \log_2 p(m)^{-1} \quad (6.1.1)$$

で定義される。 $\mathcal{M}$  から  $L$  次元の 0-1 ベクトルへの任意の全単射  $\varphi$  を Bit-Plane 変換といい,  $m \in \mathcal{M}$  に対する  $\varphi$  の値を

$$\varphi(m) = (\varphi_1(m), \varphi_2(m), \dots, \varphi_L(m)) \quad (6.1.2)$$

ここで  $\varphi_l(m) = 0$  または  $1$ ,  $l = 1, 2, \dots, L$

添字  $l$  は 0-1 ベクトルの第  $l$  桁を意味する。

と表わす。また可能なすべての  $\varphi$  の集合を  $\Phi$  とおく ( $|\Phi| = M!$ )。Bit-Plane 変換後の  $L$  個の 2 値情報源に対し,  $P_l^\varphi$  と  $Q_l^\varphi$  の各々を,  $l$  番目の 2 値情報源で 0 と 1 の生起する確率とする。すなわち,

$$P_l^\varphi = \sum_{m: \varphi_l(m)=0} p(m) \quad (6.1.3)$$

$$Q_l^\varphi = \sum_{m: \varphi_l(m)=1} p(m), \quad l = 1, 2, \dots, L.$$

Bit-Plane 変換  $\varphi$  によってシンボル  $m$  のとるベクトル  $\varphi(m)$  に関するエントロピーを  $H^\varphi$ , 各 2 値情報源のエントロピーを

$$H_l^\varphi = h(P_l^\varphi) \quad (6.1.4)_a$$

$$\text{ここで } h(p) \triangleq p \log_2 p^{-1} + (1-p) \log_2 (1-p)^{-1} \quad (6.1.4)_b$$

と置く。さらに

$$\tilde{H}^{\psi} = \sum_{\ell=1}^L H_{\ell}^{\psi} \quad (6.1.5)$$

とする。 $H$ ,  $H^{\psi}$ ,  $\tilde{H}^{\psi}$ については明らかに

$$H = H^{\psi} \leq \tilde{H}^{\psi} = H_1^{\psi} + H_2^{\psi} + \cdots + H_L^{\psi} \quad (6.1.6)$$

が成立する。(6.1.6)の不等号はBit-Plane変換することによって圧縮効果の劣化が生じることを意味している。この劣化が大きいといくら変換後の符号化の回路構成が簡単になってもBit-Plane変換を行う意義は半減する。ここで例を用いて考えてみよう。

表6.1に示す情報源に対しシンボルをそのまま2進展開して(2進展開に対応する変換を仮りに $\beta$ とする)三つの2値情報源が得られたとすると、1番目の情報源で0が生起する確率 $P_1^{\beta}$ は

$$P_1^{\beta} = p(0) + p(1) + p(2) + p(3) = 0.78$$

同様に

$$P_2^{\beta} = P_3^{\beta} = 0.755$$

と求まり、元の情報源のエントロピー $H = 1.534$ ビットに対し、三つの情報源のエントロピーの和 $\tilde{H}^{\beta}$ は

$\tilde{H}^{\beta} = 2.367$  (ビット) となる。ところが変換として表6.1の $\psi$ を用いると、同様の計算から

$\tilde{H}^{\psi} = 1.572$  ビット となる。この場合、変換による劣化はさほど問題にはならない。

そこで、一般の情報源に対しても良い変換を選ぶと



| m | p(m) | $\beta$ | $\psi$ |
|---|------|---------|--------|
| 0 | .7   | 000     | 000    |
| 1 | .01  | 001     | 111    |
| 2 | .015 | 010     | 110    |
| 3 | .055 | 011     | 010    |
| 4 | .015 | 100     | 101    |
| 5 | .03  | 101     | 100    |
| 6 | .025 | 110     | 011    |
| 7 | .15  | 111     | 001    |

$$P_1^{\beta} = .78$$

$$P_1^{\psi} = .93$$

$$P_2^{\beta} = .755$$

$$P_2^{\psi} = .895$$

$$P_3^{\beta} = .755$$

$$P_3^{\psi} = .8$$

表 6.1    2進展開變換  $\beta$  と  
最適變換  $\psi$

エントロピー和の増加を低く抑えることができるのか、またそのような変換はどのようにして求めるのかという疑問が生じる。この問題は定式的には

『ある与えられた情報源  $(m, p)$  に対し、

$$\Delta = \min_{\psi \in \Phi} \frac{\tilde{H}_\psi - H}{H} \quad (6.1.7)$$

および  $\Delta$  を達成する変換  $\psi$  を求めよ。』  
 と言い表わすことができる。

### 6.1.3 指数型分布の場合

一般の分布について最適変換  $\psi$  を探し出す系統だった方法は残念ながら現在のところまだ知られていない。一方、 $\Phi$  に含まれる変換の数はシンボルの個数  $M$  に対し  $M!$  個あり、 $M$  が 10 程度の場合でも非常に大きな数になるので、最適変換  $\psi$  を求めるためには一般に膨大な量の計算を行わなければならない。ところが、次に示す特殊な確率分布に対しては、 $\psi$  は簡単な型で記述される。

[定義 6.1.1]

$\mathcal{M}$  上の確率分布  $\{p(m), m \in \mathcal{M}\}$  が

$$p(m) = a^m / Z \quad (6.1.8)$$

$$-\infty < \lambda < \infty, Z = \sum_{m=0}^{M-1} a^m, a > 0$$

で与えられているとき、 $\{p(m), m \in \mathcal{M}\}$  を指数型分布と呼ぶ。

[定義 6.1.2]

任意の  $m \in \mathcal{M}$  に対し,

$$m = \beta_1^L(m) 2^{L-1} + \beta_2^L(m) 2^{L-2} + \dots + \beta_L^L(m) \quad (6.1.9)$$

を満たす変換  $\beta^L$  を  $L$  桁の 2 進展開変換という。

$P_\ell^{\beta^L}, Q_\ell^{\beta^L}$  に関して次の補助定理が成立する。

[補助定理 6.1.1]

$$1) \quad P_\ell^{\beta^L} = \sum_{k=0}^{2^{L-1}-1} \sum_{m=2^{L-\ell}+k}^{2^{L-\ell}(2k+1)-1} p(m) \quad (6.1.10)$$

$$2) \quad \sum_{m=0}^{M-1} m p(m) = \sum_{\ell=1}^L 2^{L-\ell} Q_\ell^{\beta^L}, \quad \ell=1, 2, \dots, L \quad (6.1.11)$$

(証明)

1) は  $P_\ell^{\beta^L}$  の定義より直接導かれ, 2) は両辺とも  $m$  の中に値を持ち確率分布  $\{p(m), m \in \mathcal{M}\}$  に従う確率変数  $X$  の期待値の異なる表現同士であることから成立する。 (証明終り)

確率分布が指数型の場合は  $P_\ell^{\beta^L}, Q_\ell^{\beta^L}$  は次のように計算される。

[補助定理 6.1.2]

指数型確率分布  $\{p(m) = a^m / Z, m = 0, 1, \dots, 2^L - 1\}$  に対し,

$$1) \quad P_\ell^{\beta^L} = \frac{1}{1 + a^{2^{L-\ell}}} \quad (6.1.12a)$$

$$Q_\ell^{\beta^L} = 1 - P_\ell^{\beta^L} = a^{2^{L-\ell}} P_\ell^{\beta^L} \quad (6.1.12b)$$

$$2) \quad \prod_{\ell=1}^L P_{\ell}^{\beta^L} = \frac{1}{Z} \quad (6.1.12)_c$$

$$\text{ここで } Z = \sum_{m=0}^{2^L-1} a^m = (1 - a^{2^L}) / (1 - a)$$

(証明)

$$P_{\ell}^{\beta^L} = \sum_{k=0}^{2^{\ell-1}-1} \sum_{m=2^{L-\ell+1}k}^{2^{L-\ell}(2k+1)-1} p(m)$$

$$(a) = \sum_{k=0}^{2^{\ell-1}-1} \sum_{m=2^{L-\ell+1}k}^{2^{L-\ell}(2k+1)-1} a^m / Z$$

$$(b) = \sum_{k=0}^{2^{\ell-1}-1} \frac{1}{Z} a^{2^{L-\ell+1}k} \frac{1 - a^{2^{L-\ell}}}{1 - a}$$

$$(c) = \frac{1 - a^{2^{L-\ell}}}{Z(1 - a)} \sum_{k=0}^{2^{\ell-1}-1} a^{2^{L-\ell+1}k}$$

$$(d) = \frac{1 - a^{2^{L-\ell}}}{Z(1 - a)} \frac{1 - a^{2^L}}{1 - a^{2^{L-\ell+1}}}$$

$$(e) = \frac{1 - a^{2^{L-\ell}}}{1 - a^{2^{L-\ell+1}}}$$

$$(f) = \frac{1}{1 + a^{2^{L-\ell}}} \quad (6.1.13)$$

ここで (6.1.13) の最初の等式は (6.1.10) のものである。 (a) は確率分布が指数型であることによる。 (b) ~ (d) は等比数列の和の公式から、 (e) は  $Z$  の定義から、最後の (f) は  $(1 - x^2) = (1 + x)(1 - x)$  より導かれる。  $Q_{\ell}^{\beta^L}$  に関しては (6.1.12)<sub>a</sub> から容易に導かれる。

さらに (6.1.13) の (e) 式を用いると

$$\begin{aligned}
\prod_{\ell=1}^L P_{\ell}^{\beta^L} &= \prod_{\ell=1}^L \frac{1 - a^{2^{L-\ell}}}{1 - a^{2^{L-\ell+1}}} \\
&= \frac{1 - a}{1 - a^{2^L}} \\
&= \frac{1}{Z} \tag{6.1.14}
\end{aligned}$$

よって題意はすべて証明された。 (証明終り)

[定理 6.1.1]

指数型分布  $\{ p(m) = a^{m\lambda}/Z, m \in \mathcal{M} \}$  に対し,

$$\tilde{H}^{\beta^L} = H \tag{6.1.15}$$

(証明)

$$\tilde{H}^{\beta^L} = \sum_{\ell=1}^L H_{\ell}^{\beta^L}$$

$$(g) = \sum_{\ell=1}^L \{ -P_{\ell}^{\beta^L} \log_2 P_{\ell}^{\beta^L} - Q_{\ell}^{\beta^L} \log_2 Q_{\ell}^{\beta^L} \}$$

$$(h) = \sum_{\ell=1}^L \{ -(1 - Q_{\ell}^{\beta^L}) \log_2 P_{\ell}^{\beta^L} - Q_{\ell}^{\beta^L} \log_2 Q_{\ell}^{\beta^L} \}$$

$$(i) = - \sum_{\ell=1}^L \log_2 P_{\ell}^{\beta^L} + \sum_{\ell=1}^L Q_{\ell}^{\beta^L} \log_2 \frac{P_{\ell}^{\beta^L}}{Q_{\ell}^{\beta^L}}$$

$$(j) = - \sum_{\ell=1}^L \log_2 P_{\ell}^{\beta^L} + \sum_{\ell=1}^L Q_{\ell}^{\beta^L} \log_2 a^{-2^{L-\ell}}$$

$$(k) = \log_2 Z - \log_2 a \sum_{\ell=1}^L 2^{L-\ell} Q_{\ell}^{\beta^L}$$

$$(l) = \log_2 Z - \log_2 a \sum_{m=0}^{2^L-1} m p(m)$$

$$(m) = \sum_{m=0}^{2^L-1} -p(m) \{ m \log_2 a - \log_2 z \}$$

$$(n) = \sum_{m=0}^{2^L-1} p(m) \log_2 \left( \frac{a^m}{z} \right)^{-1}$$

$$(o) = \sum_{m=0}^{2^L-1} p(m) \log p(m)^{-1}$$

$$(p) = H \quad (6.1.16)$$

ここで (6.1.16) の最初の等式は (6.1.5) そのままで、  
(g) は (6.1.4) による。(h), (i) の式変形後 (6.1.12) を用い  
て (j) を得る。さらに (6.1.12)c と  $\log$  関数の性質から  
(k) が導かれる。次に (6.1.11) から (l) が求まり、(l) の  
第1項を和の中に入れ、 $p(m)$  が指数型であることを  
用いて (m) から (o) までが導かれる。最後の (p) は (6.1.1)  
による。

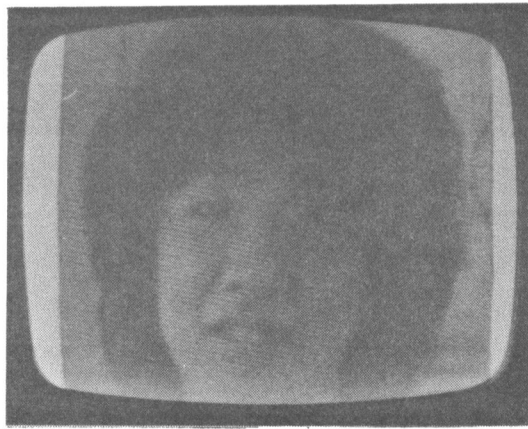
(6.1.16) より定理の主張が成立することが示された。  
(証明終り)

定理 6.1.1 は、理論的には指数型分布の情報源から  
の出力に Bit-Plane 変換として 2 進展開変換  $\beta^L$  を施せ  
ば、変換後の  $L$  個の 2 値情報源は互いに独立になるこ  
とを示している。一方、工学的、特に、画像の符号化  
に関してさらに興味深い示唆を定理 6.1.1 は含んでい  
る。このことについては次節において詳しく論じる。

#### 6.1.4 デジタル濃淡画像に対する Bit-Plane 変換

濃淡画像データに対し効率の良い Bit-Plane 変換を求めるために、幾つかの画像データについてその統計的性質を調べてみた。図 6.2 に三種類のデジタル濃淡画像の例 S, N, O を示す。これらの画像は図 6.3 に示す構成からなる自作の画像入力装置によってデジタル化されたもので、大きさが縦・横  $128 \times 128$ 、256 階調の濃淡レベルを持っている。ところで図 6.2 の画像 S について隣接する画素の差分値の分布を調べると図 6.4 のグラフに示すとおりになる。このグラフは差分値が負の指数型分布にほぼ従っていることを示している。他の二種類の画像に関しても同様の分布が得られた。この結果は文献[4]で調べられたものともよく一致しており、一般の濃淡画像の差分値も負の両側指数分布に従うものと考えられる。

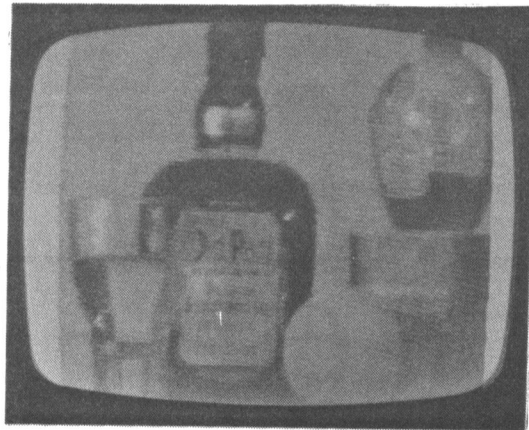
この結果をもとに画像データに対する Bit-Plane 変換を次のように構成する。まず元の画像データの差分をとり、差分値を正または 0 の場合と負の場合とに分けてそれぞれが独立な指数型確率分布に従う情報源からの出力であると考え、この二つの情報源の各々に最適な Bit-Plane 変換を用意し、差分値の正負によって変換を使い分ける。このことは変換後の 2 値情報源の符号化において、差分値が正または 0 の場合と負の場合とによって二種類の符号器・復号器を用意しなければならないことを意味する。このために符号化の処理



S



N



O

図6.2 デジタル濃淡画像の例  
大きさ  $128 \times 128$  256階調



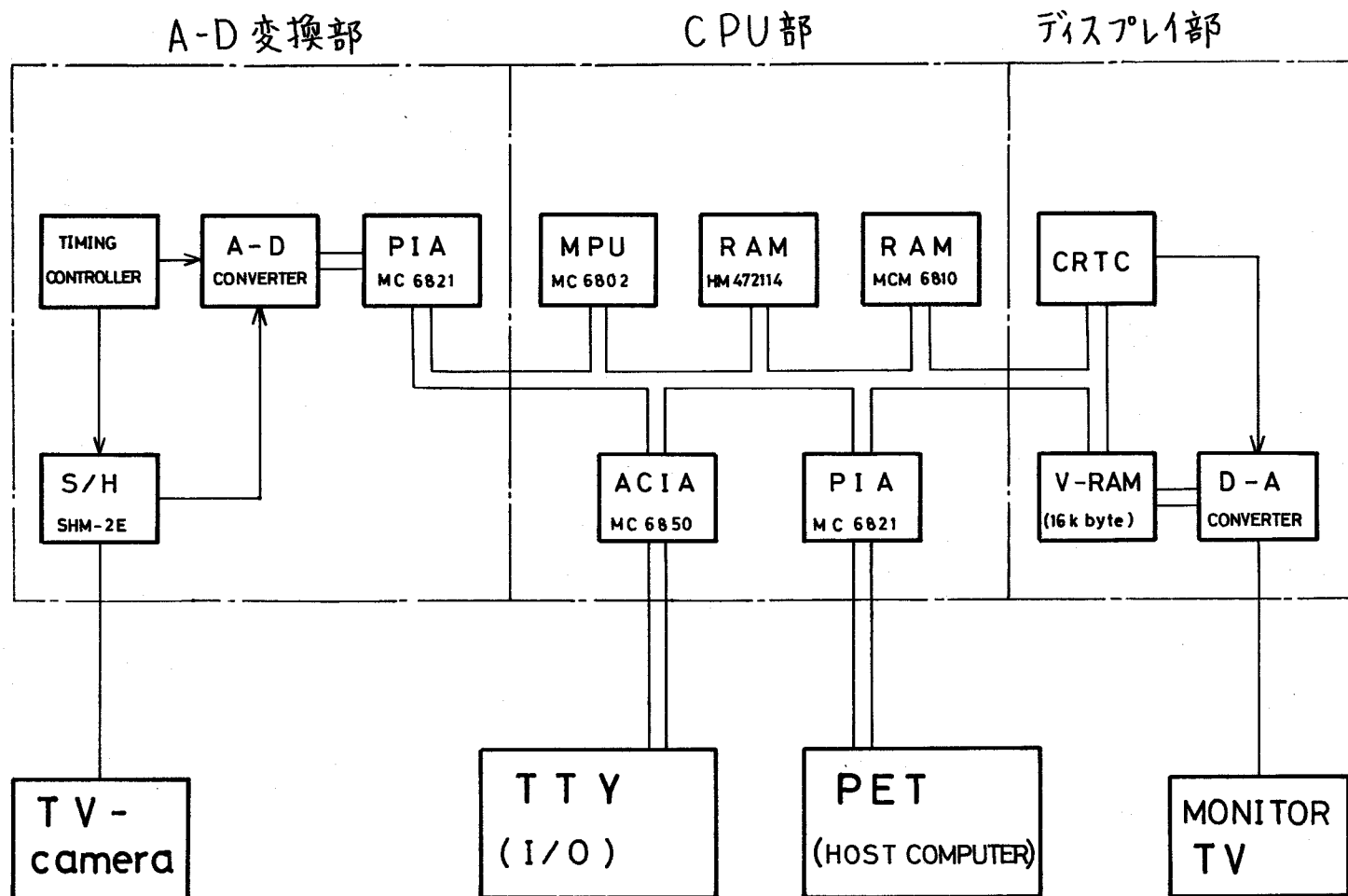


図 6.3 濃淡画像入力装置  
のブロック図

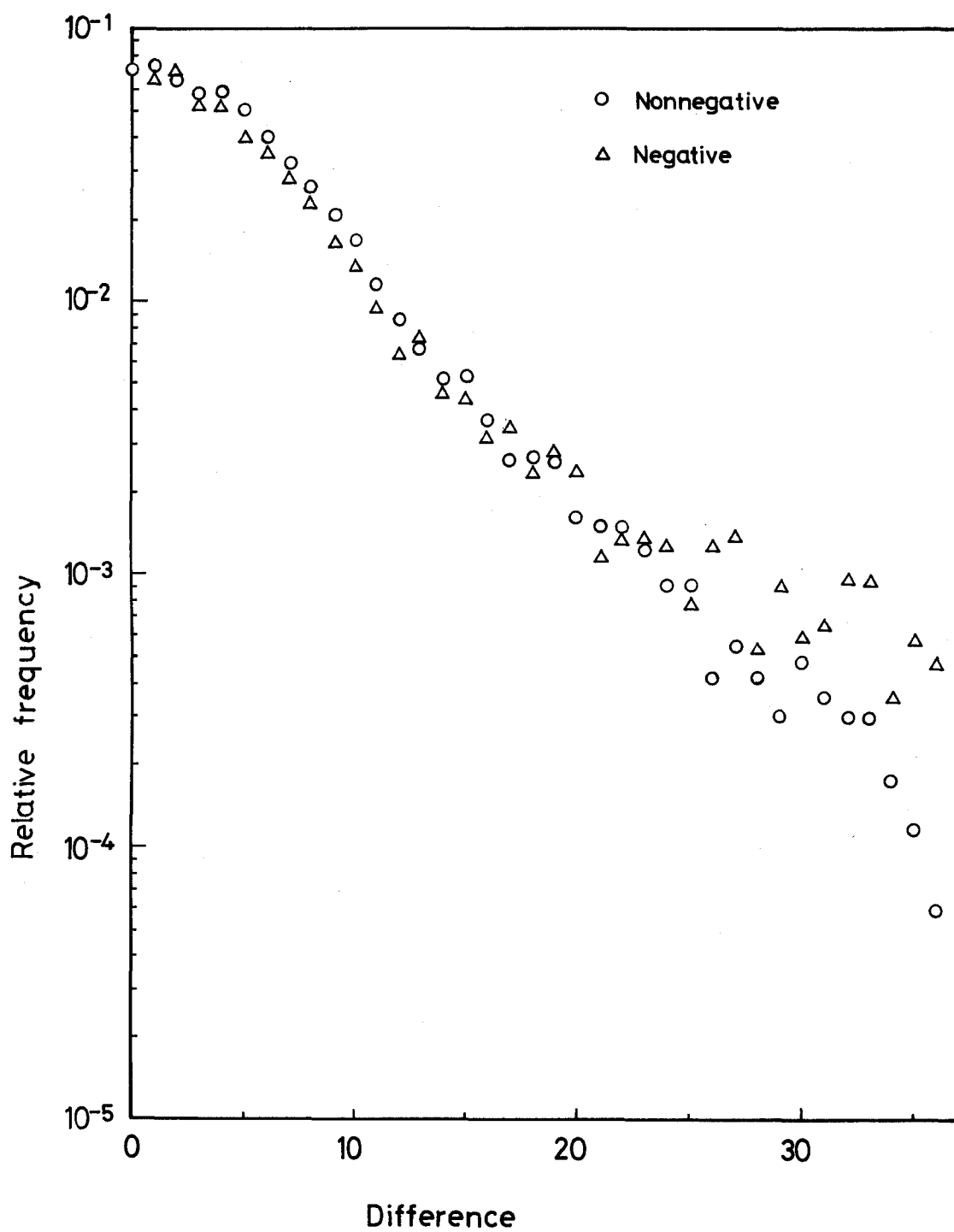


図 6.4 画像 S の差分値の分布

は多少複雑になるが、各情報源に対しては定理 6.1.1 から構成の容易な 2 進展開変換が最適であることが保証される。例として、S, N, O の各画像について差分値を、1) 正または 0 の場合と負の場合に分けて 2 進展開変換した場合と、2) 画素値をそのまま 2 進展開変換したときの変換前のエントロピー  $H$  と変換後の各 2 値情報源のエントロピー和  $\tilde{H}$  の値を表 6.2 に示す。なお表 6.2 における  $\delta$  は  $\delta = (\tilde{H} - H) / H$  で定義される。表 6.2 から次の 2 点が結論づけられる。すなわち、1) 差分値のエントロピー自体に関しては、隣接する画素の差分をとるという形で画素間の相関を利用したために、元の画素値のエントロピーに比べて顕著に低下している。

2) 差分値がほぼ指数型分布をもつために、差分値を 2 進展開変換した場合の圧縮効果の劣化は低く抑えられる。実際、元の画素値をそのまま 2 進展開した場合と比べてみても、圧縮効果の劣化を示す  $\delta$  の比は平均して約  $1/3$  になる。

以上のことから画像データの差分値をとることと 2 進展開変換することによって圧縮効果とハードウェア規模の両面において効率のよいデジタル濃淡画像の符号化が期待できる。残された問題は変換後の 2 値情報源の符号化をどのようにして行うかということである。この点に関しては次節において考察を行う。

|                                              | S     | N     | O     |
|----------------------------------------------|-------|-------|-------|
| Entropy of<br>Memoryless Model<br>$H$ (bits) | 6.683 | 5.665 | 7.362 |
| $H_1$                                        | 0.236 | 0.363 | 0.991 |
| $H_2$                                        | 0.832 | 0.535 | 0.963 |
| $H_3$                                        | 0.893 | 0.828 | 0.959 |
| $H_4$                                        | 0.999 | 0.884 | 0.998 |
| $H_5$                                        | 1.000 | 0.888 | 1.000 |
| $H_6$                                        | 1.000 | 0.909 | 1.000 |
| $H_7$                                        | 1.000 | 0.920 | 1.000 |
| $H_8$                                        | 0.999 | 0.929 | 1.000 |
| $\widetilde{H}$                              | 6.959 | 6.256 | 7.911 |
| $\delta$ (%)                                 | 4.13  | 10.4  | 7.46  |

|                                                             |       |       |       |
|-------------------------------------------------------------|-------|-------|-------|
| Entropy of<br>Differential Model<br>$H_{\text{dif}}$ (bits) | 5.031 | 4.680 | 5.579 |
| $H_0$                                                       | 1.000 | 0.944 | 1.000 |
| Probability of<br>nonnegative value                         | 0.516 | 0.638 | 0.500 |

|                              | negative nonnegative |       | negative nonnegative |       | negative nonnegative |       |
|------------------------------|----------------------|-------|----------------------|-------|----------------------|-------|
| $H_1$                        | 0.121                | 0.005 | 0.149                | 0.000 | 0.032                | 0.015 |
| $H_2$                        | 0.033                | 0.020 | 0.171                | 0.018 | 0.158                | 0.107 |
| $H_3$                        | 0.154                | 0.131 | 0.271                | 0.140 | 0.295                | 0.235 |
| $H_4$                        | 0.294                | 0.304 | 0.407                | 0.316 | 0.497                | 0.453 |
| $H_5$                        | 0.670                | 0.681 | 0.767                | 0.564 | 0.806                | 0.782 |
| $H_6$                        | 0.940                | 0.943 | 0.993                | 0.765 | 0.959                | 0.963 |
| $H_7$                        | 0.985                | 0.989 | 0.987                | 0.841 | 0.993                | 0.988 |
| $H_8$                        | 0.997                | 0.994 | 0.998                | 0.867 | 0.999                | 0.995 |
| $\widetilde{H}_{\text{dif}}$ | 5.127                |       | 4.880                |       | 5.639                |       |
| $\delta$ (%)                 | 1.91                 |       | 4.27                 |       | 1.08                 |       |

表 6.2 Bit-Plane 変換として 2 進展開  
変換を用いた場合のエン트로ピー  
和の比較

## 6.2 算術符号化方式の符号化効率について

### 6.2.1 問題と歴史

ディジタル濃淡画像の Bit-Plane 変換を行う趣旨からすれば、変換後の各 2 値情報源の符号化はできるだけハードウェア化が容易でしかも理想的な圧縮率（エントロピー）に近い圧縮効果を上げることが望ましい。圧縮効果を高めるだけなら、理論的には長いシンボル列を一つの拡大シンボルと見なしてそれぞれに符号語を用意すればよいが、これでは膨大な数の符号語を記憶する必要があり現実的なハードウェア構成であるとは言い難い。そこで符号語を記憶する代りに情報源からシンボルが出力されるごとに符号語を逐次計算によって構成するという方式が考え出されてきた。その先駆的なものに Davisson \* Lynch 符号[55]や Cover 等による数え上げ (enumerative) 符号[56][57]などがある。これらの方式では符号語の計算過程において比較的大きな自然数についての階乗計算を含んでいたために、計算量が膨大になるかあるいは符号語ではなく階乗の値を記憶する必要があった。

Rissanen [58]と Pasco [59]によって独立に提案された算術符号は、この符号語の記憶から計算による構成への流れを汲んだ符号化方式であるが、1 個のシンボルあたりの計算量が数回の加算および乗算で済むので、他のものに比べて処理手数が少なく高速に符号化できる。

このことから算術符号は Bit-Plane 変換後の 2 値情報源の符号化方式として好適であると思われる。さらに算術符号を推すいま一つの理由として、算術符号が個々の画像の統計的性質にすみやかに対処できる符号化方式であることが挙げられる。実際、画像データの差分値がほぼ指数型分布に従うといっても変換後の 2 値情報源を符号化する場合、確率分布は画像によって異なるし、一つの画像の中でも確率分布は一般に変動しているものと考えられる。そのため分布の変化に追従する適応符号化が必要になる。

従来の適応符号化方式[60]~[63]の考え方はあらかじめ用意した数種類の符号器・復号器を確率分布の変動に伴ない使い分けるというものであったが、算術符号では逐次計算の過程においてその時点での確率分布にそくした処理が可能となるので符号構成が他のものと比べて簡単になる。

ところで Rissanen の当初のねらいは算術符号が最適とされるハフマン符号よりも良い効率を有すること、すなわちシンボル列の長さを固定した場合の 1 シンボル当りの符号長がよりエントロピー・レートに近づくこと示す点にあったが、残念ながら符号パラメータを求めるための構成的なアルゴリズムはまだ得られていない[64][65]。一方、Pasco の符号はそれ以前に Elias が提案していた符号[66]に深く関連している。Elias 符号は  $[0, 1]$  区間を確率分布の比に応じて順次分割してゆ

き，ひとつのシンボル列にある部分区間を対応させるというもので符号パラメータは確率分布で表わされる点において Rissanen のものより構成が簡単である。その後も Elias 符号は Jelinek [67] によって精密に議論されている。しかしながら Elias 符号では無制限精度の乗算が必要なために実用的ではなかったが，Pasco は乗算演算を浮動小数点法を用いて有限桁の範囲で行う方法を示して Elias 符号の実用化を図った。浮動小数点法による算術符号はその後も次々に提案されている [68] ~ [72]。

この節では，算術演算をできるだけ簡素化した浮動小数点法に基づく算術符号を構成し，従来あまり報告がなされていなかった記憶のある情報源（1 次マルコフ）や適応符号化に対する効率を計算機シミュレーションによって定量的に調べ，従来の算術符号の一つである C. B. Jones [70] との性能の比較を行う。なお以下で用いる対数の底は  $d$  としておく。

### 6.2.2 算術符号の構成

$\mathcal{M}$  を有限アルファベット  $\{0, 1, \dots, M-1\}$  とし，その中に含まれる要素の任意の片側無限列，

$$S = k_1 k_2 k_3 \dots, \quad k_i \in \mathcal{M}$$

に対し，有限長さの部分列を帰納的に，

$$S[0] = \lambda, \quad S[n] = S[n-1] k_n$$

と定める。ここで  $\lambda$  は空系列である。

シンボル列  $S$  において, シンボル  $k$  が  $n$  番目に生起する確率はそれ以前のシンボル列  $S[n-1]$  にのみ依存するものとし, その値を

$$p(k|S[n-1]) \quad \text{または} \quad p^n(k)$$

と記す。さらに  $p^n(k)$  の  $d$  進数  $w$  桁表現を

$$q^n(k) = \lfloor p^n(k) \cdot d^w + 1/2 \rfloor / d^w, \quad k=0,1,\dots,M-2$$

$$q^n(M-1) = 1 - \sum_{m=0}^{M-2} q^n(m)$$

とおき, 算術符号の符号パラメータと呼ぶ。

またその累積確率  $F^n(k)$  を

$$F^n(0) = 0$$

$$F^n(k) = q^n(0) + q^n(1) + \dots + q^n(k-1), \quad k \in M \setminus \{0\}$$

で定める。

シンボル列  $S[n]$  に対する符号語  $C_n$  と加算数  $A_n$  (augend) は, 帰納的に次式で与える。

[定義 6.2.1]

$$A_0 = d^w - 1 \quad (6.2.1)a$$

$$C_0 = 0 \quad (6.2.1)b$$

$$L_0 = 0 \quad (6.2.1)c$$

$n=1, 2, \dots$  に対して,

$$A_n = \{ \lfloor A_{n-1} \cdot q^n(k_n) \rfloor \} d^{\ell_n} \quad (6.2.1)d$$

$$C_n = \{ C_{n-1} + \lfloor A_{n-1} F^n(k_n) \rfloor \} d^{\ell_n} \quad (6.2.1)e$$

$$L_n = L_{n-1} + \ell_n \quad (6.2.1)f$$

ここで  $\ell_n$  は

$$d^{w-1} \leq A_n < d^w$$

を満足する整数である。



注意 1: 加算数は常に  $w$  桁の正整数, そして符号語も正整数である。また,  $L_n$  は符号語  $C_n$  を求めるために小数点を何桁左へ移動したかを示している。(6.2.1) から明らかのように, 算術演算は  $w$  桁同士の乗算と加算に限られている。加算の際には,  $w$  桁を越えた桁上りが生じる場合があるが, この節の最後で述べる Bit-stuffing 技法[7]を用いることによって, 桁上りの伝播を有限桁で抑えることができる。そのためある桁より上位桁の値は演算によって影響を受けないので, 符号語を上位桁の方から順次伝送あるいは記憶することが可能である。

さて (6.2.1) から次に示す関係式が直ちに求まる。

[命題 6.2.1]

$L_{ab} = \sum_{t=a}^b l_n$  とおくとき,  $m < n$  に対し

$$C_n = C_{m-1} d^{L_{mn}} + [A_{m-1} F^m(k_m)] d^{L_{mn}} + \sum_{i=m+1}^n [A_{i-1} F^i(k_i)] d^{L_{in}} \quad (6.2.2)$$

(証明)

(6.2.1)e を帰納的に展開することにより導かれる。

(証明終り)

(6.2.2) はシンボル列  $S[n]$  とその部分列  $S[m]$  ( $m < n$ ) との関係を示しており, これから算術符号 (6.2.1) の復号化法を次のように定める。

[復号化法]

シンボル列  $S[n]$  に対し, すでに復元された部分列

$S[m-1]$  ( $m < n$ ) に続くシンボル  $k_m$  は,

$$(D1) \quad C_n - C_{m-1} d^{L_{mn}} \geq \lfloor A_{m-1} F^m(k) \rfloor d^{L_{mn}}$$

$$(D2) \quad C_n - C_{m-1} d^{L_{mn}} < \lfloor A_{m-1} F^m(k+1) \rfloor d^{L_{mn}}$$

を満たすシンボル  $k$  として復号化される。ただし  $k=M-1$  に対して (D1) が成立する場合は, (D2) は無視して  $k_m$  を  $k$  として復号する。

[定理 6.2.1]

算術符号(6.2.1)により, 任意の有限長さのシンボル列が復号化法(D1)(D2)を用いて復号可能であるための必要十分条件は, すべての  $n$  に対して

$$q_{\min}^n \triangleq \min \{ q^n(0), q^n(1), \dots, q^n(M-1) \} \quad (6.2.3)$$

が成立することである。

注意 2: (6.2.1)における切り捨て操作は重要で, これを切り上げや四捨五入に変更すると, もはや復号化の一意性は保証できない。

(証明)

i) 必要性の証明

復号化が正確に行われているならば, 任意のシンボル列  $S[n]$  に対し,

$$A_n > 0$$

が成立しなければならない。なぜなら, もしある  $S[n]$  に対して  $A_n = 0$  ならば, それ以降の任意のシンボル列  $S[r]$  ( $r > m$ ) に対して  $A_r = 0$  となり, 元の系列を一意に復号化することはできなくなるから。

$A_n > 0$  と同値な条件は,

$$A_{n-1} q^n(k_n) \geq 1$$

で, これが任意の  $S[n-1]$  と  $k_n$  に対し成立するためには,

$$d^{n-1} q_{\min}^n \geq 1$$

が成立することが必要である。

ii) 十分性の証明

条件より, 加算数の値はすべて正である。 $C_n$  に対し, 部分列 (空系列も含む)  $S[m-1]$  ( $1 \leq m < n$ ) がすでに復号化されているとする。このとき残りの

$$C_n - C_{m-1} d^{L_{mn}}$$

に対して

$$C_n - C_{m-1} d^{L_{mn}} \geq \lfloor A_{m-1} F^m(k_m) \rfloor d^{L_{mn}} \quad (6.2.4)$$

$$C_n - C_{m-1} d^{L_{mn}} < \lfloor A_{m-1} F^m(k_{m+1}) \rfloor d^{L_{mn}} \quad (6.2.5)$$

が成立していることを示せばよい。明らかに任意の  $k_m \in \mathcal{M}$  に対し, (6.2.4) は成立している。したがって  $k_m \neq M-1$  に対して

$$\lfloor A_{m-1} F^m(k_{m+1}) \rfloor d^{L_{mn}} > \lfloor A_{m-1} F^m(k_m) \rfloor d^{L_{mn}} + \sum_{i=m+1}^n \lfloor A_{m-1} F^i(k_i) \rfloor d^{L_{in}} \quad (6.2.6)$$

が成立することが示されれば良い。さらにそのためには

$$A_m d^{L_{(m+1)n}} > \sum_{i=m+1}^n \lfloor A_{i-1} F^i(k_i) \rfloor d^{L_{in}} \quad (6.2.7)$$

が示されれば十分である。 $t > m$  に対して,

$$A_t \leq A_m d^{L_{(m+1)t}} \prod_{j=m+1}^t q^j(k_j) \quad (6.2.8)$$

が成立していることに注意すると, (6.2.7) の右辺は,

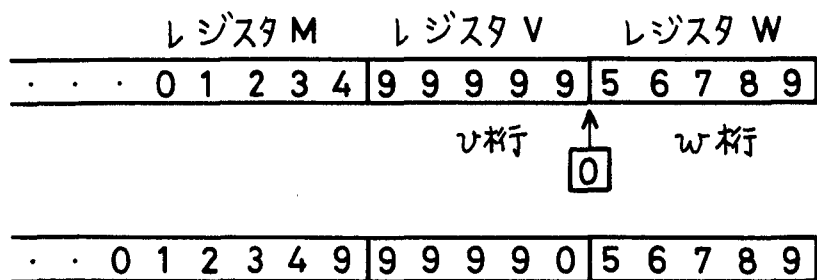
$$\begin{aligned}
& \sum_{i=m+1}^n \lfloor A_{i-1} F^i(k_i) \rfloor d^{Lin} \\
(a) & \leq \sum_{i=m+1}^n A_{i-1} F^i(k_i) d^{Lin} \\
(b) & = A_m F^{m+1}(k_{m+1}) d^{L(m+1)n} + \sum_{i=m+2}^n A_{i-1} F^i(k_i) d^{Lin} \\
(c) & \leq A_m F^{m+1}(k_{m+1}) d^{L(m+1)n} + \sum_{i=m+2}^n A_m d^{L(m+1)(i-1)} \prod_{j=m+1}^{i-1} q^j(k_j) F^i(k_i) d^{Lin} \\
(d) & = A_m F^{m+1}(k_{m+1}) d^{L(m+1)n} + A_m d^{L(m+1)n} \sum_{i=m+2}^n \prod_{j=m+1}^{i-1} q^j(k_j) F^i(k_i) \\
(e) & = A_m d^{L(m+1)n} \left\{ F^{m+1}(k_{m+1}) + \sum_{i=m+2}^n \prod_{j=m+1}^{i-1} q^j(k_j) F^i(k_i) \right\} \\
(f) & < A_m d^{L(m+1)n} \tag{6.2.9}
\end{aligned}$$

ここで (a) は  $|Lx| \leq x$  より求まる。(b) は (a) の級数和を書き直したもので、(c) は (6.2.8) から導かれる。(d) は (c) の第 2 項の共通因子を、(e) は (d) の二つの項の共通因子をくり出している。(f) は  $F^n(k_n) < 1$ ,  $q^j(k_j) + F^j(k_j) \leq 1$  から導かれる。(証明終り)

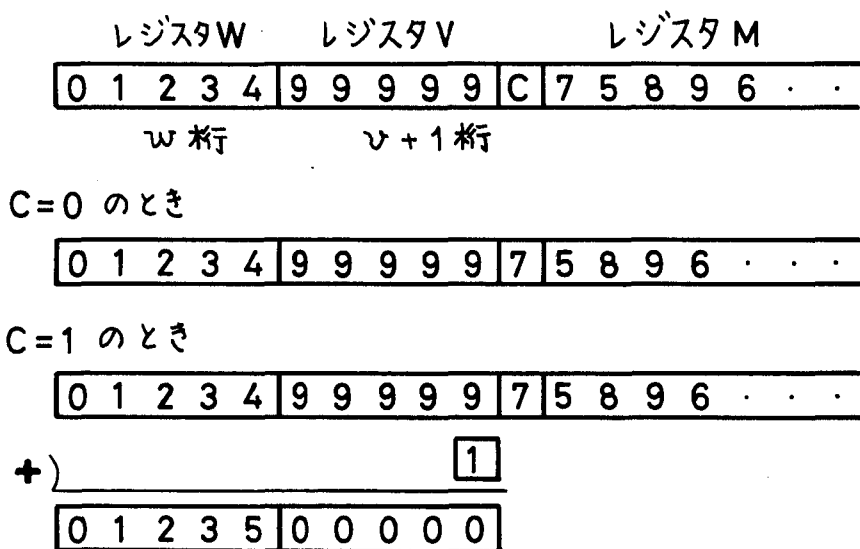
Bit-Stuffing 法は、(6.2.1)e の加算を実行する際に生じる可能性のある演算レジスタの範囲を越える桁上りの伝播を有限桁で抑える、すなわち連続した  $d$  桁の最大シンボル  $(d-1)$  が演算途中に生じた場合、 $(d-1), \dots, (d-1)$  の右端に 0 を挿入することにより、未来において生じる可能性のある桁上りを未然に防ぐものである。この方法は Langdon & Rissanen [7] によって紹介されたが、彼らの記述には不十分な点があるのでそれを指摘し、正確な方法の説明を行う。

彼らの記述によると、まず符号化においては符号語を実際の算術演算を行う  $w$  桁のレジスタ  $W$  と、それに続く桁上りを監視するための  $v$  桁のレジスタ  $V$  と、 $w+v+1$  桁より上位の算術演算には全く無関係のレジスタ  $M$  に分ける。図 6.5a に示すように、(6.2.1)e の加算とシフト演算の実行後、レジスタ  $V$  の値がすべて  $(d-1)$  になれば、その右端に 0 を挿入する。次に復号化においては、符号語を上位桁の方から、算術演算を行う  $W$  桁のレジスタ、桁上りを監視する  $v+1$  桁のレジスタ  $V$ 、それより下位のレジスタ  $M$  に分ける。図 6.5b に示すように、もしレジスタ  $V$  の上位  $v$  桁がすべて  $(d-1)$  ならば、レジスタ  $V$  の  $v+1$  桁目の値  $C$  にしたがって、 $C=0$  なら  $C$  を削除して処理を継続する。もし  $C \neq 0$  なら  $C$  を上位の  $w+v$  桁に加えた後、 $C$  を削除して処理を続ける。

しかしこの方法では不十分である。 $d=10$ ,  $v=5$ ,  $w=5$  の場合の反例を表 6.3a に示す。つまりレジスタ  $V$  の状態だけを監視しているだけでは、符号語の中に 9 が連続して 5 個現われても検出できない。したがって 5 個並んだ 9 の次のシンボルを、桁上りを防ぐために挿入された桁とみなして復号するので、正しいシンボルは復元されない。正しく復号するためには、レジスタ  $V$  の状態だけでなく、レジスタ  $V$  から  $(d-1)$  が連続して何回桁上りしたかをカウントする必要がある。修正した方法で符号化した例を表 6.3b に示す。



(a) 符号化の場合



(b) 復号化の場合

図 6.5 Bit-Stuffing の原理 ( $d = 10$ ,  $v = w = 5$ )

表 6.3 -a Example of Bit Stuffing

| Source |        | Probability |         | Augend |       | Register |       |       | Comments                           |
|--------|--------|-------------|---------|--------|-------|----------|-------|-------|------------------------------------|
| No.    | Symbol |             |         |        |       |          |       |       |                                    |
| i      | k      | $q_k$       | $F_k$   | A      | B     | M        | V     | W     |                                    |
|        |        |             |         | 36710  |       | --451    | 99998 | 79120 | Shift & store code symbol '1' in M |
| 72781  | 1      | 0.09070     | 0.54915 | 3329   | 20159 | --451    | 99998 | 99279 | Encode source symbol '1'           |
|        |        |             |         | 33290  |       | --4519   | 99989 | 92790 | Shift & store code symbol '9' in M |
| 72782  | 7      | 0.02962     | 0.94101 | 986    | 31326 | --4519   | 99990 | 24116 | Encode source symbol '7'           |
|        |        |             |         | 9860   |       | --45199  | 99902 | 41160 | Shift & store code symbol '9' in M |
|        |        |             |         | 98600  |       | --451999 | 99024 | 11600 | Shift & store code symbol '9' in M |
| 72783  | 0      | 0.54915     | 0.00000 | 54146  | 0     | --451999 | 99024 | 11600 | Encode source symbol '0'           |

表 6.3 -b Example of Bit Stuffing

| Source |        | Probability |         | Augend |       | Register  |       |       | Counter | Comments                           |
|--------|--------|-------------|---------|--------|-------|-----------|-------|-------|---------|------------------------------------|
| No.    | Symbol |             |         |        |       |           |       |       |         |                                    |
| i      | k      | $q_k$       | $F_k$   | A      | B     | M         | V     | W     | c       |                                    |
|        |        |             |         | 36710  |       | --451     | 99998 | 79120 | 0       | Shift & store code symbol '1' in M |
| 72781  | 1      | 0.09070     | 0.54915 | 3329   | 20159 | --451     | 99998 | 99279 | 0       | Encode source symbol '1'           |
|        |        |             |         | 33290  |       | --4519    | 99989 | 92790 | 1       | Shift & store code symbol '9' in M |
| 72782  | 7      | 0.02962     | 0.94101 | 986    | 31326 | --4519    | 99990 | 24116 | 1       | Encode source symbol '7'           |
|        |        |             |         | 986    |       | --45199   | 99900 | 24116 | 0       | Bit stuffing & clear c             |
|        |        |             |         | 9860   |       | --451999  | 99002 | 41160 | 1       | Shift & store code symbol '9' in M |
|        |        |             |         | 98600  |       | --4519999 | 90024 | 11600 | 2       | Shift & store code symbol '9' in M |
| 72783  | 0      | 0.54915     | 0.00000 | 54146  | 0     | --4519999 | 90024 | 11600 | 2       | Encode source symbol '0'           |

表 6.3 -c The Probability  $q_k$ ,  $F_k$  of the Example

| k     | 0       | 1       | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |
|-------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| $q_k$ | 0.54915 | 0.09070 | 0.08088 | 0.06912 | 0.06006 | 0.05028 | 0.04073 | 0.02962 | 0.01953 | 0.00984 | -       |
| $F_k$ | 0.00000 | 0.54915 | 0.63985 | 0.72073 | 0.78994 | 0.85000 | 0.90028 | 0.94101 | 0.97063 | 0.99016 | 1.00000 |

### 6.2.3 算術符号の符号化効率の評価

しばらくの間、情報源は定常かつ記憶を持たない、すなわちすべての  $n$  に対して、

$$p^n(k) = p(k), \quad k \in \mathcal{M}$$

が成立しているとする。このとき情報源のエントロピーは、

$$H = \sum_{k \in \mathcal{M}} p(k) \log p(k) \quad (6.2.10)$$

で与えられる。

また符号器、復号器とも情報源の確率分布を知っているものとする。このとき算術符号(6.2.1)のシンボル列  $S[n]$  に対する冗長度と符号化効率を次式で定義する。

冗長度

$$\begin{aligned} r(S[n]) &= \{ C_n \text{ の長さ} \} - nH \\ &= L_n + w + \text{Bit-stuffing の回数} - nH \end{aligned}$$

符号化効率

$$\eta(S[n]) = 1 - \frac{r(S[n])}{H} \quad (6.2.11)$$

まず  $L_n$  の値を評価しよう。(6.2.1)d より

$$\begin{aligned} L_n &= \sum_{i=1}^n l_i \\ &= \sum_{i=1}^n \log \left( \frac{A_i}{[A_{i-1} q(k_i)]} \right) \\ &= \log \left\{ \frac{A_1}{[A_0 q(k_1)]} \times \frac{A_2}{[A_1 q(k_2)]} \times \cdots \times \frac{A_n}{[A_{n-1} q(k_n)]} \right\} \end{aligned}$$



$$L_n = \log \left\{ \frac{A_1}{A_0 q(k_1) - \delta_1} \times \cdots \times \frac{A_n}{A_{n-1} q(k_n) - \delta_n} \right\} \quad (6.2.12)$$

$$\text{ここで, } \delta_i \triangleq A_{i-1} q(k_i) - \lfloor A_{i-1} q(k_i) \rfloor$$

さらに (6.2.12) は

$$\begin{aligned} L_n &= \log \left\{ \frac{A_n}{A_0} \times \frac{1}{q(k_1) - \delta_1/A} \times \cdots \times \frac{1}{q(k_n) - \delta_n/A_{n-1}} \right\} \\ &\leq \log \left\{ \left( q(k_1) - \frac{\delta_1}{A_0} \right) \cdots \left( q(k_n) - \frac{\delta_n}{A_{n-1}} \right) \right\}^{-1} \\ &= - \sum_{i=1}^n \log q(k_i) - \sum_{i=1}^n \log \left( 1 - \frac{\delta_i}{A_{i-1} q(k_i)} \right) \\ &\leq - \sum_{i=1}^n \log q(k_i) - \sum_{i=1}^n \log \left( 1 - \frac{1-d^{-w}}{d^{i+w} q(k_i)} \right) \quad (6.2.13) \end{aligned}$$

と変形される。n が十分大きいと大数の法則が成立し、w を十分大きくすると

$$q(k_i) \cong p(k_i)$$

であることから (6.2.13) の最右辺は

$$nH - n \sum_{k \in \mathcal{M}} p(k) \log \left( 1 - \frac{1-d^{-w}}{d^{w-1} p(k)} \right) \quad (6.2.14)$$

に近似的に等しい。一方 Bit-stuffing の回数は、(d-1) が v 回連続して出現する確率に比例するが、この確率は符号語系列が記憶のない等頻度分布に従うと仮定すると、 $d^{-v}$  である。したがって  $r(S[n])$  と  $\eta(S[n])$  の評価式として、

$$r_n \triangleq w + n d^{-v} - n \sum_{k \in \mathcal{M}} p(k) \log \left( 1 - \frac{1-d^{-w}}{d^{w-1} p(k)} \right) \quad (6.2.15)_a$$

$$\eta_n = 1 - \frac{r_n}{nH}$$

(6.2.15)<sub>b</sub>

が求まる。

今までの議論から、算術符号(6.2.1)は切り捨て操作による誤差のために、不可避免的な冗長度が生じるが、

(6.2.15)<sub>a</sub>より冗長度が大きいシンボルほど生起確率は小さいので、全体としての冗長度は低く抑えられると期待できる。実際、 $r_n$ の値を数組の確率分布 $\{p(k), k \in m\}$ に対して求めてみると、エントロピーの値にはよらず、ほぼ一定で、 $n = 10^5$ ,  $d = 10$ ,  $w = 5$ ,  $m = \{0, 1, \dots, 9\}$ の場合で約50 digitsであり、有限桁の演算に制約したことによって生じる圧縮効果の劣化は非常に少ないことを数値的に示している。符号化シミュレーションから求めた $r(S[n])$ ,  $\eta(S[n])$ と $r_n$ ,  $\eta_n$ の関係を表6.4に示す。

$r_n$ は定常マルコフ情報源に対しても容易に求めることができる。例えば1次マルコフ情報源 $\{p(k|l), k \in m, l \in m\}$ に対する評価式は、

$$r_n = w + nd^{-w} - n \sum_{l \in m} \sum_{k \in m} p(k, l) \log \left( 1 - \frac{1-d^{-w}}{d^{w-1} p(k|l)} \right)$$

ここで $p(k, l)$ は $k$ と $l$ の同時確率である。

1次マルコフ情報源の場合について符号化シミュレーションから求めた $r(S[n])$ ,  $\eta(S[n])$ と $r_n$ ,  $\eta_n$ の関係を表6.5に示す。なおシミュレーションに用いた算術符

表 6.4 Memoryless Source  
Source length  $n = 100000$

| Entropy  | Code length  | Ideal length | Redundancy      |              | Efficiency |            |
|----------|--------------|--------------|-----------------|--------------|------------|------------|
| $H_{10}$ | $L_c$ digits | $L_i$ digits | $\Delta$ digits | $r_n$ digits | $\eta$ %   | $\eta_n$ % |
| 0.105769 | 10590        | 10576.9      | 13.1            | 50.1         | 99.88      | 99.53      |
| 0.301331 | 30146        | 30133.1      | 12.9            | 49.6         | 99.96      | 99.84      |
| 0.498719 | 49886        | 49871.9      | 14.1            | 49.5         | 99.97      | 99.90      |
| 0.699773 | 69991        | 69977.3      | 13.7            | 49.5         | 99.98      | 99.93      |
| 0.899099 | 89923        | 89909.9      | 13.1            | 49.5         | 99.99      | 99.94      |

表 6.5 1st order Markov Source  
Source length  $n = 100000$

| Entropy  | Code length  | Ideal length | Redundancy      |              | Efficiency |            |
|----------|--------------|--------------|-----------------|--------------|------------|------------|
| $H_{10}$ | $L_c$ digits | $L_i$ digits | $\Delta$ digits | $r_n$ digits | $\eta$ %   | $\eta_n$ % |
| 0.166552 | 16669        | 16655.2      | 13.8            | 50.0         | 99.92      | 99.70      |
| 0.285178 | 28523        | 28517.8      | 5.2             | 23.4         | 99.98      | 99.92      |
| 0.531581 | 53173        | 53158.1      | 14.9            | 49.8         | 99.97      | 99.91      |
| 0.796975 | 79710        | 79697.5      | 12.5            | 49.5         | 99.98      | 99.94      |
| 0.999783 | 99991        | 99978.3      | 12.7            | 49.4         | 99.99      | 99.95      |

号のプログラム・リスト (FORTRAN) を付録2に掲載しておく。

#### 6.2.4 算術符号の応用

##### (a) ヒストグラム法

実際の画像データを符号化する場合，前節で議論した方法では，あらかじめ各画素値の確率分布を求める前処理を行い，さらに復号器に確率分布を前情報として送る必要がある。ところで算術符号の構成は柔軟性に富んでおり，情報源が定常な場合には前処理を行わずに逐次に求まる頻度分布（ヒストグラム）を用いて符号化することができる。この方法について次に述べる。

シンボル列  $S[n]$  において，シンボル  $k$  の現われる頻度を  $f^n(k)$  とする。記憶のない定常情報源に対しては， $n$  を大きくするにつれて，確率 1 で，

$$f^n(k)/n \rightarrow P(k)$$

であることから， $n$  番目のシンボル  $k_n$  を符号化する際の確率パラメータを， $f^n(k)$  を用いて，

$$q^n(k_n) = \frac{1 + f^{n-1}(k_n)}{10 + \sum_{k=m} f^{n-1}(k)}$$

と定める。

この方法によって，無記憶定常情報源の符号化シミュレーションを行った結果を表6.6に示す。また情報源が1次マルコフの場合，シンボル列  $S[n]$  において，

文字  $j$  の次に  $k$  が現われる頻度を  $f^n(k|j)$  とすると、確率パラメータ  $q^n(k|k_{n-1})$  は、

$$q^n(k|k_{n-1}) = \frac{1 + f^{n-1}(k|k_{n-1})}{10 + \sum_{k \in \mathcal{M}} f^{n-1}(k|k_{n-1})}$$

で定められる。この確率パラメータ  $q^n(k|k_{n-1})$  を用いて 1 次マルコフ情報源の符号化シミュレーションを行った結果を表 6.7 に示す。

前節で議論した前処理を行う方式で必要な前情報は、無記憶、1 次マルコフのそれぞれの場合で 50, 500 digits (シミュレーションでは 1 シンボルの確率を小数点以下 5 桁で表示している。)であり、これを考慮すると、符号化効率、前処理方式よりヒストグラム法のほうがわずかながら良くなる。したがって定常な情報源の符号化においては前処理操作は、逐次に頻度を求める簡単な算術演算によって置き換えられる。

#### (b) 適応符号化法

(a) の方法を発展させ、情報源の確率に変動する場合に対し、算術符号の適用を考える。従来の適応符号化法の多くは、まず前処理として局所的な確率分布の推定を行い、あらかじめ用意された膨大な符号の集まりの中から推定結果に最も良く合う符号を選び出して符号化を行うというものであったが、算術符号を適用することによって、符号の集まりを省くことができる。そのために確率パラメータの値を次のようにして決める。

表 6.6 Memoryless Source Histogram Method

Source length  $n = 100000$

| Entropy<br>$H_{10}$ | Code length<br>$L_c$ digits | Ideal length<br>$L_1$ digits | Redundancy<br>$\Delta$ digits | Efficiency<br>$\eta \%$ |
|---------------------|-----------------------------|------------------------------|-------------------------------|-------------------------|
| 0.105769            | 10614                       | 10576.9                      | 37.1                          | 99.65                   |
| 0.301331            | 30168                       | 30133.1                      | 34.9                          | 99.88                   |
| 0.498719            | 49906                       | 49871.9                      | 34.1                          | 99.93                   |
| 0.699773            | 70012                       | 69977.3                      | 34.7                          | 99.95                   |
| 0.899099            | 89943                       | 89909.9                      | 33.1                          | 99.96                   |

表 6.7 1st Markov Source Histogram Method

Source length  $n = 100000$

| Entropy<br>$H_{10}$ | Code length<br>$L_c$ digits | Ideal length<br>$L_1$ digits | Redundancy<br>$\Delta$ digits | Efficiency<br>$\eta \%$ |
|---------------------|-----------------------------|------------------------------|-------------------------------|-------------------------|
| 0.166551            | 16857                       | 16655.2                      | 201.8                         | 98.79                   |
| 0.285178            | 28768                       | 28517.8                      | 250.2                         | 99.12                   |
| 0.531581            | 53341                       | 53158.1                      | 182.9                         | 99.66                   |
| 0.796975            | 79859                       | 79697.5                      | 161.5                         | 99.80                   |

表 6.8 Coding Result of Adaptive Method

Source length  $n = 10000$

| Source | Code length<br>$L_c$ digits | Ideal length<br>$L_1$ digits | Redundancy<br>$\Delta$ digits | Efficiency<br>$\eta \%$ |
|--------|-----------------------------|------------------------------|-------------------------------|-------------------------|
| A      | 1457                        | 1392.5                       | 64.5                          | 95.37                   |
| B      | 2206                        | 2083.3                       | 122.7                         | 94.11                   |
| C      | 2369                        | 2147.5                       | 221.5                         | 89.69                   |

$g^n(k)$  をシンボル列  $S[n]$  の  $(n-N+1)$  番目から  $n$  番目までにシンボル  $k$  の現われる頻度とする。シンボル列  $S[n]$  に対し、 $g^n(k)$  は次の漸化式で逐次に求まる。

$$g^n(k_n) = g^{n-1}(k_n) + 1$$

$$g^n(k_{n-N}) = g^{n-1}(k_{n-N}) - 1$$

ここで便宜上

$$S[n] = (k_{-N+1}, k_{-N+2}, \dots, k_{-1}, k_0, k_1, \dots, k_n)$$

と考える。

$n$  番目のシンボル  $k_n$  ( $n \geq 1$ ) を符号化する際の確率パラメータは、

$$g^n(k_n) = \frac{g^{n-1}(k_n)}{N} \quad (6.2.16)$$

で与えられる。つまり、シンボル  $k_n$  のまわりの局所的な確率分布をパラメータとする。

この構成に基づいた算術符号を用いて、次に示す3種類の2値情報源の符号化シミュレーションを行った。

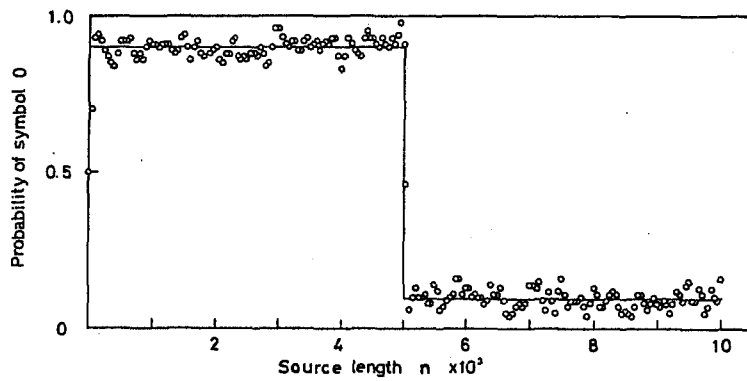
1)  $S[n]$  ( $n = 10000$ ) に対し、確率分布が1度だけ変動する場合。この情報源を Source A と名付ける。

2) 確率分布が不定周期で変動する場合

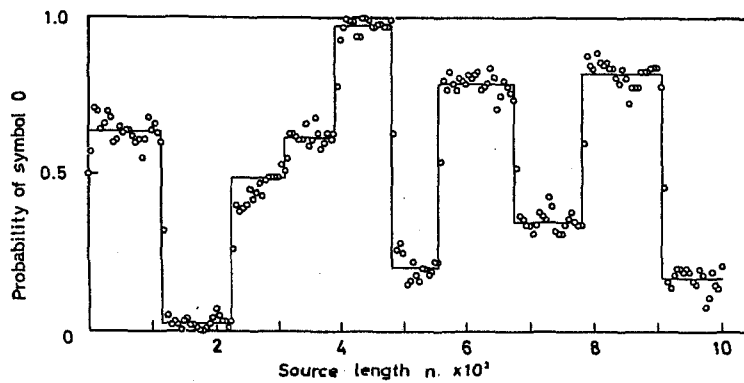
(a) 変動がゆるやかな場合。この情報源を Source B と名付ける。

(b) 変動が激しい場合。この情報源を Source C と名付ける。

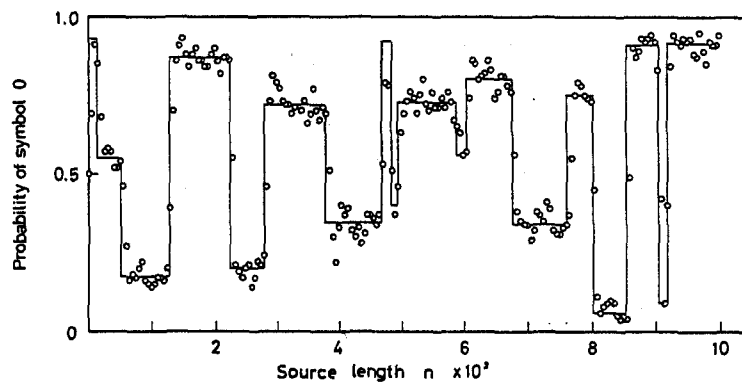
図 6.6 は  $N = 100$  の場合に (6.2.16) の確率パラメー



Source A



Source B



Source C

図 6.6 確率パラメータが不定期に変動する確率モデル



タが確率分布の変動にどの程度追従できるかを示している。変動が増大するにつれて追従におくれが生じ、それに依りて符号化効率も劣化してゆく(表6.8 参照)。しかし最も変化の激しい情報源Cの場合でも効率は90%であり、適応符号化方式として十分実用になりうる。

## 6.2.5 他の算術符号との比較

I (Pasco's Codes [59] 1976)

$$A_n = \lfloor A_{n-1} q^n(k_n) \rfloor \quad (P1)$$

$$C_n = \{ C_{n-1} + A_{n-1} F^n(k_n) \} d^{l_n} \quad (P2)$$

II (Jones' Codes [70] 1981)

$$A_n = \{ \lfloor A_{n-1} F^n(k_{n+1}) + \frac{1}{2} \rfloor - \lfloor A_{n-1} F^n(k_n) + \frac{1}{2} \rfloor \} d^{l_n} \quad (J1)$$

$$C_n = \{ C_{n-1} + \lfloor A_{n-1} F^n(k_n) + \frac{1}{2} \rfloor \} d^{l_n} \quad (J2)$$

III (提案方式 (6.2.1))

$$A_n = \lfloor A_{n-1} q^n(k_n) \rfloor d^{l_n} \quad (M1)$$

$$C_n = \{ C_{n-1} + \lfloor A_{n-1} F^n(k_n) \rfloor \} d^{l_n} \quad (M2)$$

上のIとIIは浮動小数点方式による代表的な算術符号である。ここでは簡単のために加算数と符号語の計

算部分のみを示している。なお I, II における  $l_n$  はそれぞれの  $A_n$  が  $d^{w-1} \leq A_n < d^w$  を満足するように定められる。

この節では上記の I, II の符号と提案符号 III との性能の比較を行う。ここでまず 3 者の計算手数を比較してみると、 $A_n$  の計算に関しては乗算回数は I では 1 回、II では 2 回、III では 1 回、それに II では 1 回の減算が加わる。また  $C_n$  については加算に必要な桁数は II と III では  $w$  桁であるが、I では  $2w$  桁必要になる。したがって手数の上だけでは III が最も簡単な構成になっている。I と III の大きな違いは、 $A_{n-1} \cdot q^n(k_n)$  の演算結果をまずシフトしてから小数点以下の切り捨てを行うか、演算結果の小数点以下を切り捨ててからシフトを行うかという点である。III の方が切り捨て誤差が大きいので、符号化効率はもちろん I の方が良いと思われる。しかしながら I では III の倍の長さの加算用のレジスタを用いているので、同じ条件下で効率の比較は行えない。それに反し、II と III については同じ長さのレジスタを用いているので公平な比較が行える。

定常かつ無記憶な情報源に対し、シンボル列の長さを長くしていった場合の II と III の冗長度を表 6.9 に示す。これを見ると III ではシンボル長の増加に伴ない冗長度も増えているのに対し、II では冗長度は低く抑えられたままであることが見てとれる。この結果は II では区間  $A_{n-1}$  を過不足なくシンボル数で分割して

表6.9 Memoryless prescan method  
Entropy  $H_{10} \approx 0.3$

| Source length | Ideal code length | JONES Code   | Proposed method |
|---------------|-------------------|--------------|-----------------|
| 100000        | 30133.1           | 30138 (4.9)  | 30146 (12.9)    |
| 200000        | 60627.8           | 60631 (3.2)  | 60646 (18.2)    |
| 300000        | 90923.0           | 90926 (3.0)  | 90951 (28.0)    |
| 400000        | 121011.9          | 121013 (1.1) | 121046 (34.1)   |
| 500000        | 151052.1          | 151059 (6.9) | 151102 (49.9)   |
| 600000        | 181057.4          | 181058 (0.6) | 181106 (48.6)   |

表6.10 Memoryless prescan method  
Binary source  
Source length  $n = 100000$

| Entropy  | Ideal code length | JONES Code  | Proposed method |
|----------|-------------------|-------------|-----------------|
| 0.029769 | 2977.0            | 2981 (4.0)  | 2983 (6.0)      |
| 0.150126 | 15012.6           | 15017 (4.4) | 15019 (6.4)     |
| 0.216276 | 21627.6           | 21632 (4.4) | 21636 (8.4)     |

表6.11 Coding Result of Adaptive Method  
Source length  $n = 10000$

| Source | Ideal code length | JONES Code   | Proposed method |
|--------|-------------------|--------------|-----------------|
| A      | 1392.5            | 1460 ( 67.5) | 1457 ( 64.5)    |
| B      | 2083.3            | 2206 (122.7) | 2206 (122.7)    |
| C      | 2147.5            | 2334 (186.5) | 2369 (221.5)    |

表6.9 ~ 11 提案符号とJones符号の  
各種情報に対する性能の比較

いるのに対し，Ⅲでは切り捨て演算を行なっているために区間  $A_{n-1}$  を最大限に符号化へ役立てていないからと思われる。この現象はシンボル数の数に大きく依存し，例えばシンボル数は10から2に変えると，表6.10に示すように両者の冗長度の違いはわずかである。さらに適応符号化を行った場合では，両者の優劣はもはやつけ難い(図6.11 参照)

結論として，定常な情報源に対しては提案方式は確かに切り捨て誤差による効率の劣化が目立つが，実際の画像データは定常というよりは適応符号化の場合に考えた情報源に近いので，提案方式はその計算手数の少なさとも合せて十分実用に耐えうると思われる。

## 第7章

### ま と め

画像の符号化に関して、白黒2値画像の場合と濃淡画像の場合に分けて、それぞれに特有の問題について議論を行った。まず2値画像の場合、選択画素という一種の特徴点を画面内に定義し、できるだけ少ない選択画素による元の画像の一意的な表現方法について論じた。その結果をもとに選択画素符号化方式(SECT)を導き、漢字パターンのデータ圧縮のためにSECTを用いた。結果は、SECTが従来の符号化方式と同等あるいはそれ以上の圧縮効果をもつことを示している。この結果はSECTで用いる符号構成の簡単さと合せて、SECTの有効性を表わしている。

また選択画素から知ることのできる元の画像の位相的性質について調べ、画面全体ではなく (i, r, c)-表現から連結成分やホールの計数や面積の測定ができることを明らかにした。

一方、濃淡画像の場合、濃淡レベルの多さ(例えば $2^8$ )に起因する膨大なハードウェア規模を大巾に削減する方法として、濃淡画像データを複数個の2値情報に効率よく還元する Bit-Plane 変換を提案し、画像データに対する変換の妥当性を示した。



## 付 録 1

### S E C T の 8 0 8 0 ア セ ン ブ ル ・ リ ス ト

```

00010 ; *****
00020 ; * SECT VER 1.3 *
00030 ; * 1980.12.4 *
00040 ; * BY KOJI KITAJIMA *
00050 ; *****
00060 ;
00070 ;
00080 ;
00090 0020 SIZE EQU 32 Nの値
00100 ;
00110 ; MONITOR SUB ROUTINE
00120 ;
00130 001B OUTCPL EQU S1B 改行
00140 00DA OUTHLR EQU SDA HLの内容を出力
00150 0248 CONTRL EQU S248 モニタ入ロ
00160 ;
00170 ; WORK MEMORY
00180 ;
00190 ; ORG S506B
00200 506D CODENC DS 1 点の種類
00210 506C CDTPTR DS 2 符号化データポイント
00220 506E CLMVAL DS 1 行の値
00230 506F ROWVAL DS 1 列の値
00240 ;
00250 ; WORK AREA
00260 ;
00270 7800 CDTADR EQU S7800 符号化データの先頭アドレス
00280 7800 PDTADR EQU S7800 画像データ
00290 6FE0 BLINAD EQU PDTADR-SIZE 前ラインの値
00300 6FC0 LFLCAD EQU BLINAD-SIZE 左方向の値
00310 6FA0 RFLCAD EQU LFLCAD-SIZE 右
00320 ;
00330 ; ORG S5100
00340 ;
00350 ; *****
00360 ; * ENCODER *
00370 ; *****
00380 ;
00390 5100 CDB653 ENCODE CALL INITIAL *符号化プログラムメインルーチン
00400 5103 011151 ENCBGM LXI B,ENCNEY 47-7FはE59-7へ
00410 5106 C5 PUSH B 内部レジスタ
00420 5107 CD1A53 CALL CVALI D:行の値
00430 510A A7 ANA A E:列の値
00440 510D E25951 JPO SEENCP 各点処理へ
00450 510E 5A0B51 JPE TCENCP HL:画像データポイント
00460 5111 23 ENCNEY INY H
00470 5112 1C INR E
00480 5113 7B MOV A,E
00490 5114 FE20 CPI SIZE
00500 511C DA0351 JC ENCBGM 一行終了か?
00510 5119 CD9C52 CALL LINEYP
00520 511C 7A MOV A,D
00530 511D FE20 CPI SIZE
00540 511F DA0351 JC ENCBGM 最終画像か?
00550 5120 2A6C50 LHLR CDTPTR
00560 5125 CDEA07 CALL OUTHLR 符号化データの最終番地表示
00570 5128 C34B02 JMP CONTRL モニタへ
00580 ;

```



|       |      |        |        |      |           |               |
|-------|------|--------|--------|------|-----------|---------------|
| 00590 | 512B | E601   | TCENCP | ANI  | I         | * I 点処理       |
| 00600 | 512D | 3C     |        | INP  | A         |               |
| 00610 | 512E | 47     |        | MOV  | B, A      |               |
| 00620 | 512F | E603   |        | YFI  | 3         |               |
| 00630 | 5131 | 4F     |        | MOV  | C, A      |               |
| 00640 | 5132 | CD2553 |        | CALL | CVAL2     |               |
| 00650 | 5135 | B8     |        | CMP  | B         |               |
| 00660 | 5136 | C0     |        | PNZ  |           | i 点なければリターン   |
| 00670 | 5137 | D5     |        | PUSH | D         |               |
| 00680 | 5138 | E5     |        | PUSH | H         |               |
| 00690 | 5139 | 23     | TCENCI | INX  | H         |               |
| 00700 | 513A | 1C     |        | INP  | E         |               |
| 00710 | 513B | 7B     |        | MOV  | A, E      |               |
| 00720 | 513C | FE20   |        | CPI  | SIZE      |               |
| 00730 | 513E | D24F51 |        | JNC  | TCENCP    |               |
| 00740 | 5141 | CD1A53 |        | CALL | CVAL1     |               |
| 00750 | 5144 | B8     |        | CMP  | B         |               |
| 00760 | 5145 | CA5651 |        | JZ   | SEENCI    |               |
| 00770 | 5148 | CD2553 |        | CALL | CVAL2     |               |
| 00780 | 514D | B9     |        | CMP  | C         |               |
| 00790 | 514C | C23951 |        | JNZ  | TCENCI    |               |
| 00800 | 514F | E1     | TCENCP | POP  | H         |               |
| 00810 | 5150 | D1     |        | POP  | D         |               |
| 00820 | 5151 | 3E06   |        | MVI  | A, 6      | i 点のコード 110   |
| 00830 | 5153 | C36653 |        | JMP  | STACDT    | 画像登録          |
| 00840 |      |        |        |      |           |               |
| 00850 | 5156 | F1     | SEENCI | POP  | PSV       |               |
| 00860 | 5157 | F1     |        | POP  | PSV       |               |
| 00870 | 5158 | 78     |        | MOV  | A, B      |               |
| 00880 | 5159 | 47     | SEENCP | MOV  | B, A      | * i 点以外の点処理入口 |
| 00890 | 515A | CD2553 |        | CALL | CVAL2     |               |
| 00900 | 515D | D8     |        | CMP  | B         |               |
| 00910 | 515E | C0     |        | RZ   |           | 各点なければリターン    |
| 00920 | 515F | 0F01   |        | MVI  | C, 1      |               |
| 00930 | 5161 | E5     |        | PUSH | H         |               |
| 00940 | 5162 | 23     |        | INX  | H         |               |
| 00950 | 5163 | CD3153 |        | CALL | CVAL3     |               |
| 00960 | 5166 | D8     |        | CMP  | B         | 右45°フェッチ      |
| 00970 | 5167 | CA7C51 |        | J7   | FMENCP    | 右変形か?         |
| 00980 | 516A | 0C     |        | INP  | C         |               |
| 00990 | 516D | 2B     |        | DCX  | H         |               |
| 01000 | 516C | 2B     |        | DCX  | H         |               |
| 01010 | 516D | CD3153 |        | CALL | CVAL3     | 左45°フェッチ      |
| 01020 | 5170 | D8     |        | CMP  | B         | 左変形か?         |
| 01030 | 5171 | CA7C51 |        | J7   | FMENCP    |               |
| 01040 | 5174 | E1     |        | POP  | H         |               |
| 01050 | 5175 | 78     |        | MOV  | A, B      |               |
| 01060 | 5176 | 3D     |        | DCR  | A         |               |
| 01070 | 5177 | F604   |        | ORI  | 4         |               |
| 01080 | 5179 | C36653 |        | JMP  | STACDT    | 画像登録          |
| 01090 |      |        |        |      |           |               |
| 01100 | 517C | 21A06F | FMENCP | LXI  | H, PFLGAD | * 変形領域画像処理    |
| 01110 | 517F | 7D     |        | MOV  | A, L      |               |
| 01120 | 5180 | 83     |        | ADD  | E         |               |
| 01130 | 5181 | 6F     |        | MOV  | L, A      |               |
| 01140 | 5182 | 0D     |        | DCR  | C         |               |
| 01150 | 5183 | CA8951 |        | J7   | FMENCI    |               |
| 01160 | 5186 | CD4753 |        | CALL | ADPHL     |               |
| 01170 | 5189 | 3C01   | FMENCI | MVI  | M, 1      | 777°直止        |
| 01180 | 518D | 78     |        | MOV  | A, B      |               |
| 01190 | 518C | 3D     |        | DCR  | A         |               |
| 01200 | 518D | 67     |        | PLC  |           |               |
| 01210 | 518E | D1     |        | OFA  | C         |               |
| 01220 | 518F | CD6653 |        | CALL | STACDT    | 画像登録          |
| 01230 | 5192 | E1     |        | POP  | H         |               |
| 01240 | 5193 | C0     |        | PET  |           |               |

```

01250      ;
01260      ; *****
01270      ; * DECODED *
01280      ; *****
01290      ;
01300  5194  CDB853  DECODE  CALL  INITIAL      *復号化プログラム
01310      ;
01320  5197  0600  UMODE  MVI  B,0
01330  5199  CDB853  MODE   CALL  INPCDT
01340  519C  DAB851  JC      MODF3      データ終了か?
01350  519F  CD3A53  MODE1  CALL  CVAL4
01360  51A2  08      CMP  B
01370  51A3  CA0352  JZ      CODEFP      整合しているか?
01380  51A6  70      MOV  M,B
01390  51A7  23      INX  H
01400  51A8  1C      INP  E
01410  51A9  7B      MOV  A,E
01420  51AA  FE20  CPI  SIZE
01430  51AC  D20052  JNC  CODEPR2      一行終了か?
01440  51AF  CD5C53  CALL  CMPCDT
01450  51B2  C29F51  JNZ  MODE1
01460  51B5  C30A52  JMP  CODEPR2      各点処理へ
01470  51B8  70      MOV  M,B
01480  51B9  23      INX  H
01490  51BA  1C      INP  E
01500  51BD  CD3A53  MODE3  CALL  CVAL4      最終画像まで復元
01510  51BE  08      CMP  B
01520  51BF  C20851  JNZ  MODE2
01530  51C2  CD5553  MODE4  CALL  GETBLD
01540  51C5  7B      MOV  A,E
01550  51C6  FE20  CPI  SIZE
01560  51C8  DAC251  JC      MODE4
01570  51CB  CD9C52  CALL  LINEFP
01580  51CE  7A      MOV  A,D
01590  51CF  FE20  CPI  SIZEF
01600  51D1  DAC251  JC      MODF4
01610      ;
01620  51D4  CD1B00  DECE2D  CALL  OUTCPL      *復元された画像をプリント
01630  51D7  210072  LYI  H,PTADR
01640  51DA  1020  MVI  D,SIZE
01650  51DC  1E20  DECE1  MVI  E,SIZE
01660  51DE  3E20  DECE2  MVI  A," "
01670  51E2  46      MOV  E,M
01680  51E1  05      DCP  B
01690  51E2  05E2  JNZ  DECE3
01700  51E5  3E2A  MVI  A,"*"
01710  51E7  E7      DECE3  PST  A
01720  51E8  23      INX  H
01730  51E9  1D      DCP  E
01740  51EA  C0DE51  JNZ  DECE2
01750  51ED  CD1B00  CALL  OUTCPL
01760  51F0  15      DCP  D
01770  51F1  C2DC51  JNZ  DECE1
01780  51F4  C348A2  JMP  CONTRL      モータへ
01790      ;
01800  51F7  CD5F53  CODEPR1  CALL  GETBLD      次の行の先頭まで前行をコピー
01810  51FA  7B      MOV  A,E
01820  51FB  FE20  CPI  SIZE
01830  51FD  DAF751  JC      CODEPR1
01840  5200  CD9C52  CODEPR2  CALL  LINEFP
01850  5203  3A7F52  CODEPR3  LDA  CLMVAL
01860  5206  DA      CMP  D
01870  5207  C2F751  JNZ  CODEPR1
01880  520A  3A6B52  CODEPR2  LDA  CODENO
01890  520D  FE04  CPI  4
01900  520F  DAB352  JC      PMDECP      変形置換画像
01910  5212  D004  SUI  4
01920  5214  FE02  CPI  2
01930  5216  DA4552  JC      SEDECP      i.r.点

```

```

01950 521C 010000 CODPP3 CALL GETBLD
01960 521F 3A6F50 CODPP4 LDA POINTAL
01970 5220 BE CMP E
01980 5223 C21C52 JNZ CODPP3
01990 5226 CD3A53 CALL CVAL4
02000 5229 EF01 XFI 1
02010 522L 47 MOV B, A
02020
02030 522C 70 TCDECP MOV M, B
02040 522F 23 INX H
02050 522E 1C INR E
02060 522F 7B MOV A, E
02070 5230 FE20 CPI SIZE
02080 5232 D23F52 JNC TCDCPI
02090 5235 CD3A53 CALL CVAL4
02100 5238 BE CMP B
02110 5239 C22C52 JNZ TCDECP
02120 523C C39951 JMP MODE
02130 523F CD9C52 TCDCPI CALL LINEXP
02140 5242 C39751 JMP IMODE
02150
02160 5245 4F SEDECP MOV C, A
02170 524C 0C INP C
02180 5247 EE01 XFI 1
02190 5249 47 MOV B, A
02200 524A C35052 JMP SEDCP1
02210 524D CD5553 CALL GETBLD
02220 5250 3A6F50 SEDCP1 LDA POINTAL
02230 5253 BE CMP E
02240 5254 CA9951 JZ MODE
02250 5257 CD2553 CALL CVAL2
02260 525A D0 CMP C
02270 525B C24D52 JNZ SEDCP1-3
02280 525E E5 PUSH H
02290 525F 70 SEDCP2 MOV M, B
02300 5260 23 INX H
02310 5261 1C INR E
02320 5262 CD5C53 CALL CMPCDT
02330 5265 CA7352 JZ SEDCP3
02340 5268 CD3A53 CALL CVAL4
02350 526B BE CMP B
02360 526C CA5F52 JZ SEDCP2
02370 526F F1 POP PSW
02380 5270 C35052 JMP SEDCP1
02390 5273 7D SEDCP3 MOV A, L
02400 5274 E1 POP H
02410 5275 CD5553 CALL GETBLD
02420 5278 0D DCR C
02430 5279 71 SEDCP4 MOV M, C
02440 527A 23 INX H
02450 527B DD CMP L
02460 527C C27952 JNZ SEDCP4
02470 527F 6F MOV L, A
02480 5280 C39951 JMP MODE
02490
02500 5283 47 RMDECP MOV B, A
02510 5284 E5 PUSH H
02520 5285 21A06F LXI H, PFLCAD
02530 5288 3A6F50 LDA POINTAL
02540 528B 85 ADD L
02550 528C 6F MOV L, A
02560 528D 78 MOV A, B
02570 528E A7 ANA A
02580 528F 1F PAR
02590 5290 D29C52 JNC RMDCPI
02600 5293 CD4753 CALL AFDHL
02610 5296 3601 RMDCPI MVI M, 1
02620 5298 E1 POP H
02630 5299 C2A552 JMP SEFECP

```

\* I 点の前処理

14

\* I 点処理

| 1579 | i 点 | I 点 |
|------|-----|-----|
| B    | 0   | 1   |

復元中に一行終了すれば、777処理後  
自モードへ

\* r, C 点処理

| 1579 | r 点 | C 点 |
|------|-----|-----|
| B    | 01  | 00  |
| C    | 01  | 10  |

\* 変形選択結果処理

| 1579 | i 点 | o 点 | ca 点 | cl 点 |
|------|-----|-----|------|------|
| B    | 00  | 01  | 10   | 11   |

777更新後、r, C 点処理へ

```

02640      ;
02650      ; *****
02660      ; * SUB ROUTINE *
02670      ; *****
02680      ;
02690  529C D5 LINEXP PUSH D
02700  529D 110070 LXI D, PDTADR
02710  52A0 0620 MVI B, SIZE
02720  52A2 1B LEXPI DCM D
02730  52A3 2B DCM H
02740  52A4 7E MOV A, M
02750  52A5 12 STAX D
02760  52A6 05 DCP B
02770  52A7 C2A252 JNZ LEXPI
02780  52AA 1E00 MVI E, 0
02790  52AC CD1A53 LEMP2 CALL CVAL1
02800  52AF A7 ANA A
02810  52B0 EAD352 CPO FLCCM
02820  52B3 23 INX H
02830  52B4 1C INP E
02840  52B5 7B MOV A, E
02850  52B6 FE20 CPI SIZE
02860  52B8 DAAC52 JC LEMP2
02870  52BB 11A26F LXI D, PFLCAD
02880  52BE 0640 MVI B, BLINAD-PFLCAD
02890  52C0 1A LEMP3 LDAX D
02900  52C1 A7 ANA A
02910  52C2 FAC652 JM LEMP4
02920  52C5 AF YRA A
02930  52C6 E001 LEMP4 ANI 1
02940  52C8 12 STAX D
02950  52C9 13 INX D
02960  52CA 05 DCP B
02970  52CB C2C052 JNZ LEMP3
02980  52CE D1 POP D
02990  52CF 1E00 MVI E, 0
03000  52D1 1A INP D
03010  52D2 C9 RET
03020      ;
03030  52D3 E5 FLCCM PUSH H
03040  52D4 3D DCR A
03050  52D5 47 MOV B, A
03060  52D6 21C06F LXI H, LFLCAD
03070  52D9 CDEC52 CALL FLCCM
03080  52DC 3D DCP A
03090  52DD C0D53 C7 LFLCCM
03100  52E0 21A26F LXI H, PFLCAD
03110  52E3 CDEC52 CALL FLCCM
03120  52E6 3D DCP A
03130  52E7 CCF852 C7 PFLCCM
03140  52EA E1 POP H
03150  52ED C9 RET
03160      ;
03170  52EC 7D FLCCM5 MOV A, L
03180  52ED 03 ADD E
03190  52EE 0F MOV L, A
03200  52EF 7E MOV A, M
03210  52F0 A7 ANA A
03220  52F1 FC PP
03230  52F2 C0E1 MVI C, SR1
03240  52F4 71 MOV M, C
03250  52F5 A9 YRA C
03260  52F6 1F FAP
03270  52F7 C9 RET

```

現在復元の終了した行を BLINAD  
にコピー  
7777を調べる BLINADを変更  
7777を消す。

7777が立っているかチェック  
立っていれば処理へ

FLCCMのサブルーチン  
7777のアドレス計算

|       |      |        |        |      |           |                                        |
|-------|------|--------|--------|------|-----------|----------------------------------------|
| 03280 |      |        |        |      |           |                                        |
| 03290 | 52F8 | 7B     | RFLGCK | MOV  | A, E      | 右方向777処理                               |
| 03300 | 52F9 | 1E1F   |        | CPI  | SIZE-1    |                                        |
| 03310 | 52FD | D80453 |        | JNC  | RFCK1     |                                        |
| 03320 | 52FE | 23     |        | INX  | H         |                                        |
| 03330 | 52FF | 7E     |        | MOV  | A, M      |                                        |
| 03340 | 5300 | 07     |        | PLC  |           |                                        |
| 03350 | 5301 | 1681   |        | OPI  | S81       |                                        |
| 03360 | 5303 | 77     |        | MOV  | M, A      |                                        |
| 03370 | 5304 | 48     | RFCK1  | MOV  | C, B      |                                        |
| 03380 | 5305 | 21E06F | RFCK2  | LXI  | H, BLINAD |                                        |
| 03390 | 5306 | 7D     |        | MOV  | A, L      |                                        |
| 03400 | 5309 | 03     |        | ADD  | E         |                                        |
| 03410 | 530A | 6F     |        | MOV  | L, A      |                                        |
| 03420 | 530B | 71     |        | MOV  | M, C      |                                        |
| 03430 | 530C | C9     |        | RET  |           |                                        |
| 03440 |      |        |        |      |           |                                        |
| 03450 | 530D | 2B     | LFLGCK | DCX  | H         | 左方向777処理                               |
| 03460 | 530E | 3681   |        | MOV  | M, S81    |                                        |
| 03470 | 5310 | 78     |        | MOV  | A, B      |                                        |
| 03480 | 5311 | EE01   |        | XRI  | I         |                                        |
| 03490 | 5313 | 4F     |        | MOV  | C, A      |                                        |
| 03500 | 5314 | CD0553 |        | CALL | RFCK2     |                                        |
| 03510 | 5317 | 2B     |        | DCX  | H         |                                        |
| 03520 | 5318 | 71     |        | MOV  | M, C      |                                        |
| 03530 | 5319 | C9     |        | RET  |           |                                        |
| 03540 |      |        |        |      |           |                                        |
| 03550 | 531A | 7B     | CVAL1  | MOV  | A, E      | CVAL1~5は、画素の値を7キラム7行に<br>求めるサブルーチン     |
| 03560 | 531B | A7     |        | ANA  | A         |                                        |
| 03570 | 531C | CA2353 |        | JZ   | CV11      | 現在行<br>$e(i, j) e(i, j)$               |
| 03580 | 531F | 2B     |        | DCX  | H         |                                        |
| 03590 | 5320 | 7E     |        | MOV  | A, M      |                                        |
| 03600 | 5321 | 07     |        | PLC  |           |                                        |
| 03610 | 5322 | 23     |        | INX  | H         |                                        |
| 03620 | 5323 | B6     | CV11   | ORA  | M         |                                        |
| 03630 | 5324 | C9     |        | RET  |           |                                        |
| 03640 |      |        |        |      |           |                                        |
| 03650 | 5325 | E5     | CVAL2  | PUSH | H         |                                        |
| 03660 | 5326 | 21E06F |        | LXI  | H, BLINAD | 前行 (777処理はなし)<br>$e(i-1, i) e(i-1, j)$ |
| 03670 | 5329 | 7D     |        | MOV  | A, L      |                                        |
| 03680 | 532A | 83     |        | ADD  | E         |                                        |
| 03690 | 532B | 6F     |        | MOV  | L, A      |                                        |
| 03700 | 532C | CD1A53 |        | CALL | CVAL1     |                                        |
| 03710 | 532F | E1     |        | POP  | H         |                                        |
| 03720 | 5330 | C9     |        | RET  |           |                                        |
| 03730 |      |        |        |      |           |                                        |
| 03740 | 5331 | E5     | CVAL3  | PUSH | H         | 先行<br>$e(i, j-1) e(i, j)$              |
| 03750 | 5332 | CD4753 |        | CALL | ADDHL     |                                        |
| 03760 | 5335 | CD1A53 |        | CALL | CVAL1     |                                        |
| 03770 | 5338 | E1     |        | POP  | H         |                                        |
| 03780 | 5339 | C9     |        | RET  |           |                                        |
| 03790 |      |        |        |      |           |                                        |
| 03800 | 533A | CD2553 | CVAL4  | CALL | CVAL2     | 現行<br>$e(i-1, j)$                      |
| 03810 | 533D | EE01   |        | ANI  | I         |                                        |
| 03820 | 533F | C9     |        | RET  |           |                                        |
| 03830 |      |        |        |      |           |                                        |
| 03840 | 5340 | E5     | CVAL5  | PUSH | H         | 前行 (777処理はなし)<br>$e'(i-1, j)$          |
| 03850 | 5341 | CD4E53 |        | CALL | SUBHL     |                                        |
| 03860 | 5344 | 7E     |        | MOV  | A, M      |                                        |
| 03870 | 5345 | E1     |        | POP  | H         |                                        |
| 03880 | 5346 | C9     |        | RET  |           |                                        |
| 03890 |      |        |        |      |           |                                        |
| 03900 | 5347 | C5     | ADDHL  | PUSH | B         | HL ← HL + SIZE                         |
| 03910 | 5348 | 010000 |        | LXI  | B, SIZE   |                                        |
| 03920 | 534B | C9     |        | DAD  | B         |                                        |
| 03930 | 534C | C1     |        | POP  | B         |                                        |
| 03940 | 534D | C9     |        | RET  |           |                                        |

|       |      |        |        |      |          |                 |
|-------|------|--------|--------|------|----------|-----------------|
| 03950 |      |        |        |      |          |                 |
| 03960 | 534E | C5     | SUBHL  | PUSH | B        | HL ← HL-SIZE    |
| 03970 | 534F | 01E0FF |        | LXI  | B, -SIZE |                 |
| 03980 | 5350 | 09     |        | DAD  | B        |                 |
| 03990 | 5353 | C1     |        | POP  | B        |                 |
| 04000 | 5354 | C9     |        | RET  |          |                 |
| 04010 |      |        |        |      |          |                 |
| 04020 | 5355 | 0D3A53 | GETBLD | CALL | CMVAL4   | 前行の値をコピー        |
| 04030 | 5358 | 77     |        | MOV  | M, A     |                 |
| 04040 | 5359 | 23     |        | INX  | H        |                 |
| 04050 | 535A | 1C     |        | INF  | E        |                 |
| 04060 | 535D | C9     |        | RET  |          |                 |
| 04070 |      |        |        |      |          |                 |
| 04080 | 535C | 3A6F50 | CMPCDT | LDA  | POWVAL   | 現在の行列の値と符号化デ-7の |
| 04090 | 535F | BB     |        | CMP  | E        | 行列の値を比較         |
| 04100 | 5360 | 0C     |        | RNZ  |          |                 |
| 04110 | 5361 | 3A6E50 |        | LDA  | CLMVAL   |                 |
| 04120 | 5364 | DA     |        | CMP  | D        |                 |
| 04130 | 5365 | C9     |        | RET  |          |                 |
| 04140 |      |        |        |      |          |                 |
| 04150 | 5366 | E5     | STACDT | PUSH | H        | 選択要素の登録         |
| 04160 | 5367 | 0F     |        | PRC  |          |                 |
| 04170 | 5368 | 0F     |        | PRC  |          |                 |
| 04180 | 5369 | 0F     |        | PRC  |          |                 |
| 04190 | 536A | 4F     |        | MOV  | C, A     |                 |
| 04200 | 536B | 2A6C50 |        | LHLD | CDTPTR   |                 |
| 04210 | 536E | 3A6E50 |        | LDA  | CLMVAL   |                 |
| 04220 | 5371 | BA     |        | CMP  | D        |                 |
| 04230 | 5372 | CA7D53 |        | JZ   | STAC1    |                 |
| 04240 | 5375 | 7A     |        | MOV  | A, D     |                 |
| 04250 | 5376 | 326E50 |        | STA  | CLMVAL   |                 |
| 04260 | 5379 | F6E0   |        | ORI  | SE0      |                 |
| 04270 | 537B | 77     |        | MOV  | M, A     |                 |
| 04280 | 537C | 23     |        | INX  | H        |                 |
| 04290 | 537D | 79     | STAC1  | MOV  | A, C     |                 |
| 04300 | 537E | 83     |        | ADD  | E        |                 |
| 04310 | 537F | 77     |        | MOV  | M, A     |                 |
| 04320 | 5380 | 23     |        | INX  | H        |                 |
| 04330 | 5381 | 36A0   |        | MVI  | M, S00   |                 |
| 04340 | 5383 | 226C50 |        | SHLD | CDTPTR   |                 |
| 04350 | 5386 | E1     |        | POP  | H        |                 |
| 04360 | 5387 | C9     |        | RET  |          |                 |
| 04370 |      |        |        |      |          |                 |
| 04380 | 5388 | E5     | INPCDT | PUSH | H        | 符号化デ-7の読み込み     |
| 04390 | 5389 | 2A6C50 |        | LHLD | CDTPTR   |                 |
| 04400 | 538C | 7E     |        | MOV  | A, M     |                 |
| 04410 | 538D | FEA0   |        | CPI  | S00      |                 |
| 04420 | 538F | 37     |        | STC  |          |                 |
| 04430 | 5390 | CAB453 |        | J7   | INPC2    |                 |
| 04440 | 5393 | 7E     |        | MOV  | A, M     |                 |
| 04450 | 5394 | 2F     |        | CMA  |          |                 |
| 04460 | 5395 | E6E0   |        | ANI  | SEP      |                 |
| 04470 | 5397 | C2A153 |        | JNZ  | INPC1    |                 |
| 04480 | 539A | 7E     |        | MOV  | A, M     |                 |
| 04490 | 539D | E61F   |        | ANI  | S1F      |                 |
| 04500 | 539D | 326E50 |        | STA  | CLMVAL   |                 |
| 04510 | 53A0 | 23     |        | INX  | H        |                 |
| 04520 | 53A1 | 7E     | INPC1  | MOV  | A, M     |                 |
| 04530 | 53A2 | E6E0   |        | ANI  | SEP      |                 |
| 04540 | 53A4 | 07     |        | RLC  |          |                 |
| 04550 | 53A5 | 07     |        | RLC  |          |                 |
| 04560 | 53A6 | 07     |        | RLC  |          |                 |
| 04570 | 53A7 | 326B50 |        | STA  | CODENO   |                 |
| 04580 | 53AA | 7E     |        | MOV  | A, M     |                 |
| 04590 | 53AD | E61F   |        | ANI  | S1F      |                 |
| 04600 | 53AD | 326F50 |        | STA  | POWVAL   |                 |
| 04610 | 53B0 | 23     |        | INX  | H        |                 |
| 04620 | 53B1 | 226C50 |        | SHLD | CDTPTR   |                 |
| 04630 | 53B4 | E1     | INPC2  | POP  | H        |                 |

```

04650      ;
04660 53B6 210078 INITIAL LXI H, CDTADR      初期値 設定
04670 53B9 226C50 SHLD CDTPTR
04680 53BC 3EFF MVI A, 5FF
04690 53BE 326E50 STA CLMVAL
04700 53C1 3E60 MVI A, PDTADR-RFLCAD
04710 53C3 21A06F LXI H, RFLCAD
04720 53C6 3C00 INTH1 MVI H, 0
04730 53C8 23 INX H
04740 53C9 3D DCR A
04750 53CA C2C653 JNZ INTH1
04760 53CD 110000 LXI D, 0
04770 53D0 C9 RET
04780      END
PASS 2 END

```

\*\*\* LABELS \*\*\*

```

SIZE 0020 OUTCPL 001B OUTHLR 00DA
CONTRL 0048 CODENO 506B CDTPTP 506C
CLMVAL 506E ROWVAL 506F CDTADR 7800
PDTADR 7800 BLINAD 6FE0 LFLGAD 6FC0
RFLCAD 6FA0 ENCODE 5100 ENCBGM 5103
ENCNEY 5111 TCENCP 512B TCENC1 5139
TCENC2 514F SEENC1 5156 SEENCP 5159
RMENCP 517C RMENC1 5189 DECODE 5194
IMODE 5197 MODE 5199 MODE1 519F
MODE2 51B8 MODE3 51BB MODE4 51C2
DECEND 51D4 DECE1 51DC DECE2 51DE
DECE3 51E7 CODPR1 51F7 CODPR0 5200
CODEPR 5203 CODPR2 520A CODPR3 521C
CODPR4 521F TCDECP 522C TCDCP1 523F
SEDECP 5245 SEDCP1 5250 SEDCP2 525F
SEDCP3 5273 SEDCP4 5279 PMDECP 5283
PMDCP1 5296 LINEXP 529C LEXP1 52A2
LEXP2 52AC LEXP3 52C0 LEXP4 52C6
FLGCHK 52D3 FLGCKS 52EC RFLGCK 52FB
RFCK1 5304 RFCK2 5305 LFLGCK 530D
CVAL1 531A CV11 5323 CVAL2 5325
CVAL3 5331 CVAL4 533A CVAL5 5340
ADDHL 5347 SUBHL 534E GETBLD 5355
QMPCDT 535C STACDT 5366 STAC1 537D
INPCDT 5388 INPC1 53A1 INPC2 53DA
INITIAL 53B6 INTH1 53C6

```





## 付 録 2

本論文で提案した算術符号の  
FORTRANによる  
プログラムリスト

```

00010      PRINT," FILE NAME : A-CODE-7"
00020C*****
00030C
00040C      PROGRAM FOR ARITHMETIC CODE
00050C
00060C      SOUCE SYMBOL : 0,1 (ONLY)
00070C      CODE SYMBOL : 0,1,---,9
00080C      SOUCE MODEL : MEMORYLESS NONSTATIONARY MODEL
00090C
00100C      CHENGE PROBABILITY USING WINDOW SIZE 100
00110C      DOSU ; INITIAL VALUE IS 1 & WEIGHT IS 1
00120C
00130C*****
00140      REAL DEFF
00150      INTEGER M(61000),M1(61000),M2(61000)
00160      INTEGER POWER,POWER1,POWER2,POWER3
00170      INTEGER ANEW,AOLD,CNEW,COLD,Y
00180      INTEGER SHIFT,SHIFTL,OVER,COUNT,CARRY
00190      INTEGER P(10),PS(11),DOSU(10),POINT(11)
00200      INTEGER QQ,Q1
00210      INTEGER NSUM,WBLOCK,NWRITE
00220      INTEGER INDEXW,COUNTER,W(10)
00230      COMMON /BLOCK1/INDEX1,POWER1,M1
00240      COMMON /BLOCK2/ P,PS
00250      COMMON /BLOCK8/ INDEXW,COUNTER,W,Y
00260      COMMON /BLOCK9/ DOSU
00270      DATA QQ/4H#YES/
00280      DATA M(1),M1(1),M2(1) /0,0,0/
00290      SRLLENGTH=0.0
00300      NSUM=0
00310      PRINT,"INPUT HOW MANY TIME CHENGE THE THRESHOLD VALUE "
00320      READ(5,1000) NCHENGE
00330 1000 FORMAT(V)
00340C***** MAKING SOUCE *****
00350      DATA POWER1,INDEX1 /9,1/
00360      DO 10 II=1,NCHENGE+1
00370C
00380          CALL HINDO(N,RLENGTH)
00390C
00400          SRLLENGTH=SRLLENGTH+RLENGTH
00410          NSUM=NSUM+N
00420 10 CONTINUE
00430      WRITE(6,1100) SRLLENGTH
00440 1100 FORMAT(1H0," RIRON LENGTH      =",F15.8)
00450C-----
00460      PRINT," "
00470      PRINT,"INPUT NUMBER OF BLOCK PRINTING P' '(I)"
00480      READ(5,1000) WBLOCK
00490      NWRITE=WBLOCK
00500      PRINT,"INPUT #YES --- IF YOU WANT SIGNAL & CODE"
00510      READ(5,1200) Q1
00520 1200 FORMAT(A4)
00530C
00540C*****
00550C      ENCODER
00560C*****
00570C
00580C***** INITIAL VALUE *****
00590      DATA POWER,INDEX,POWER3,INDEX3 /9,1,-1,0/
00600      AOLD=99999

```

```

00610      COLD=0
00620      SHIFT=0
00630      COUNT=0
00640      CARRY=0
00650      CHEAK=99999
00660C----- SET DOSU(I)=1 ----- 50
00670      DO 20 I=1,2
00680          DOSU(I)=50
00690      20 CONTINUE
00700C-----
00710C----- SET INITIAL WINDOW -----
00720C
00730      CALL INTWIN
00740C
00750C-----
00760C
00770C ***** ENCODER *****
00780C
00790      DO 80 I=1,NSUM
00800C----- CALCURATION P"(I) -----
00810          PS(1)=0
00820          P(1)=INT(1.0*DOSU(1)/100*10**5+0.5)
00830          IF(P(1).LT.10) P(1)=10
00840          IF(P(1).GE.10**5) P(1)=99990
00850          PS(2)=PS(1)+P(1)
00860          PS(3)=10**5
00870          P(2)=PS(3)-PS(2)
00880C-----
00890C----- PRINTING P"(I) IF I=WBLOCK*J -----
00900          IF(I.LT.NWRITE) GO TO 30
00910          NWRITE=NWRITE+WBLOCK
00920          WRITE(6,1300) I,(P(J1),J1=1,2)
00930 1300      FORMAT(1H," I=",I8,2I6)
00940C-----
00950C
00960      30 CALL INOUT(Y,INDEX3,POWER3,M1,0)
00970C
00980      CALL WINDOW
00990C
01000          ANEW=INT(P(Y+1)*AOLD/10**5)
01010          CNEW=COLD+INT(PS(Y+1)*AOLD/10**5)
01020          SHIFTL=0
01030      40 IF(10*ANEW.GE.10**5) GO TO 70
01040          ANEW=10*ANEW
01050          SHIFTL=SHIFTL+1
01060          SHIFT=SHIFT+1
01070          IF(SHIFT.LE.5) GO TO 50
01080          OVER=INT(CNEW/10**9)
01090C
01100          CALL INOUT(OVER,INDEX,POWER,M,1)
01110C
01120          CNEW=CNEW-OVER*10**9
01130      50 CNEW=10*CNEW
01140C----- BIT STUFFING -----
01150          IF(OVER.NE.9) COUNT=-1
01160          COUNT=COUNT+1
01170          IF(INT(CNEW/10**(5+COUNT)).LT.INT(CHEAK/10**COUNT)) GO TO 60
01180          CALL INOUT(9,INDEX,POWER,M,1)
01190          CNEW=CNEW-9*10**(5+COUNT)
01200          CARRY=CARRY+1

```

```

01210          PRINT," "
01220          PRINT," CARRY OVER #02 "
01230          WRITE(6,1400) I,COUNT,CNEW
01240 1400      FORMAT(1H," I=",I8," COUNT=",I2," C(SY)=",I11)
01250C-----
01260          60      GO TO 40
01270C *
01280          70      AOLD=ANEW
01290          COLD=CNEW
01300C *
01310          80      CONTINUE
01320          DO 90 I=1,10
01330              OVER=INT(COLD/10**9)
01340C
01350          CALL INOUT(OVER,INDEX,POWER,M,1)
01360C
01370          COLD=COLD-OVER*10**9
01380          COLD=10*COLD
01390          90      CONTINUE
01400C
01410C***** OUTPUT RESULT *****
01420C
01430          IF(Q1.NE.QQ) GO TO 100
01440          PRINT," "
01450          PRINT,"SOURCE SIGNAL"
01460          WRITE(6,1500) (M1(I),I=1,INDEX1)
01470          PRINT," "
01480          PRINT,"CODE WORD"
01490          WRITE(6,1500) (M(I),I=1,INDEX)
01500 1500      FORMAT(1H,"6I11)
01510          100     LENGTH=SHIFT-SHIFTL+5+CARRY
01520          WRITE(6,1600) LENGTH
01530 1600     FORMAT(1H0," CODE LENGTH=",I12)
01540          DEFF=FLOAT(LENGTH)-SRLLENGTH
01550          WRITE(6,1700) DEFF
01560 1700     FORMAT(1H0," (CODE LENGTH)-(ENTROPY*(SOURCE LENGTH))=",F12.3)
01570          E=100.0-100.0*DEFF/SRLLENGTH
01580          WRITE(6,1800) E
01590 1800     FORMAT(1H0," CODE EFFCIENCY=",F15.8)
01600C
01610C*****
01620C          DECODER
01630C*****
01640C
01650C***** INITIAL VALUE FOR DECODER *****
01660C
01670          DATA POWER2,INDEX2 /9,1/
01680          POWER=-1
01690          INDEX=0
01700          AOLD=99999
01710          COLD=0
01720          COUNT=0
01730C----- SET INITIAL C(NULL) -----
01740          DO 110 K=1,5
01750              CALL INOUT(OVER,INDEX,POWER,M,0)
01760              COLD=COLD+OVER*10**(5-K)
01770              IF(OVER.NE.9) COUNT=-1
01780              COUNT=COUNT+1
01790          110     CONTINUE
01800C----- SET DOSU(I)=1 -----

```

```

01810      DO 120 I=1,2
01820          DOSU(I)=50
01830      120 CONTINUE
01840C----- SET INITIAL WINDOW -----
01850C
01860      CALL INTWIN
01870C
01880C-----
01890C
01900C ***** DECODER *****
01910C
01920      DO 180 K=1,NSUM
01930C----- CALCURATION P"(I) -----
01940          PS(1)=0
01950          P(1)=INT(1.0*DOSU(1)/100*10**5+0.5)
01960          IF(P(1).LT.10) P(1)=10
01970          IF(P(1).GE.10**5) P(1)=99990
01980          PS(2)=PS(1)+P(1)
01990          PS(3)=10**5
02000          P(2)=PS(3)-PS(2)
02010C-----
02020C----- CALCURATION SUBINTERVAL -----
02030          DO 130 J=1,3
02040              POINT(J)=INT(AOLD*PS(J)/10**5)
02050      130 CONTINUE
02060C-----
02070C
02080      DO 140 I=1,2
02090          IF(COLD.LT.POINT(I)) GO TO 140
02100          IF(COLD.GE.POINT(I+1)) GO TO 140
02110          Y=I-1
02120          CALL WINDOW
02130C
02140          CALL INOUT(Y,INDEX2,POWER2,M2,1)
02150C
02160          CNEW=COLD-POINT(I)
02170          ANEW=INT(AOLD*P(I)/10**5)
02180      140 CONTINUE
02190      150 IF(10*ANEW.GE.10**5) GO TO 170
02200          ANEW=ANEW*10
02210C
02220          CALL INOUT(OVER,INDEX,POWER,M,0)
02230C
02240C----- BIT STUFFING -----
02250          IF(COUNT.NE.5) GO TO 160
02260          IF(OVER.NE.0) CNEW=CNEW+1
02270          CALL INOUT(OVER,INDEX,POWER,M,0)
02280          COUNT=0
02290C-----
02300      160 IF(OVER.NE.9) COUNT=-1
02310          COUNT=COUNT+1
02320          CNEW=10*CNEW+OVER
02330          GO TO 150
02340C *
02350      170 COLD=CNEW
02360          AOLD=ANEW
02370      180 CONTINUE
02380C
02390C***** CHECK DECODED SIGNAL *****
02400      DO 190 I=1,INDEX1

```

```

02410      IF(M1(I)-M2(I)) 200,190,200
02420 190 CONTINUE
02430      PRINT," DECODED WITHOUT ERROR"
02440      IF(Q1.NE.QQ) GO TO 210
02450      PRINT," "
02460      PRINT,"DECODED SIGNAL"
02470      WRITE(6,1500) (M2(I),I=1,INDEX2)
02480      GO TO 210
02490C *
02500 200 PRINT,"DECODE ERROR"
02510      WRITE(6,1900) I,M1(I),M2(I)
02520 1900 FORMAT(1H," I=",I10," SOUCE SIGNAL=",I11," DECODED SIGNAL=",
02530      & I11)
02540 210 STOP
02550      END
02560C
02570C *****
02580C ***** SUBROUTINE *****
02590C *****
02600C
02610C***** SUBROUTINE FOR GENERATING RANDAM NUMBER *****
02620C
02630      SUBROUTINE RANDOM(DA,DB,DC,DM,DX,RANSU)
02640      DOUBLE PRECISION DA,DB,DC,DM,DX
02650      DX=DA*DB+DC
02660      DX=DMOD(DX,DM)
02670      DB=DX
02680      RANSU=DX/DM
02690      RETURN
02700      END
02710C
02720C***** SUBROUTINE FOR I/O SYMBOL(SOUC, CODE, DECODED) *****
02730C
02740      SUBROUTINE INOUT(OVER,INDEX,POWER,M,FLAG)
02750      INTEGER OVER,INDEX,POWER,FLAG,MEMORY
02760      INTEGER M(61000)
02770      IF(FLAG.LE.0) GO TO 20
02780      IF(POWER.GE.0) GO TO 10
02790      INDEX=INDEX+1
02800      POWER=9
02810      M(INDEX)=0
02820 10 M(INDEX)=M(INDEX)+OVER*10**POWER
02830      GO TO 40
02840C *
02850 20 IF(POWER.GE.0) GO TO 30
02860      INDEX=INDEX+1
02870      MEMORY=M(INDEX)
02880      POWER=9
02890 30 OVER=INT(MEMORY/10**POWER)
02900      MEMORY=MEMORY-OVER*10**POWER
02910 40 POWER=POWER-1
02920      RETURN
02930      END
02940C
02950C***** SUBROUTINE FOR CALCURATING FREQCNCY *****
02960C
02970      SUBROUTINE HINDO(N,RLENGTH)
02980C
02990      DOUBLE PRECISION DA,DB,DC,DM
03000      COMMON /BLOCK1/ INDEX1,POWER1,M1

```

```

03010      COMMON /BLOCK2/ P,PS
03020      COMMON /BLOCK9/ DOSU
03030      INTEGER SYMBOL,Y
03040      INTEGER INDEX1,POWER1,M1(61000)
03050      INTEGER P(10),PS(11),DOSU(10)
03060      REAL RANSU,F(10),ENTROPY
03070C
03080      PRINT," "
03090      PRINT,"INPUT THRESHOLD FOR RANDOM NUMBER : P(1),P(2) ; ##### "
03100      READ(5,1100) (P(I),I=1,2)
03110      WRITE(6,1000) (P(I),I=1,2)
03120 1000  FORMAT(1H , "P(I)",2I6)
03130      PS(1)=0
03140      DO 10 I=1,2
03150          PS(I+1)=PS(I)+P(I)
03160 10  CONTINUE
03170      PRINT," "
03180      PRINT,"INPUT STRING LENGTH"
03190      READ(5,1100) N
03200      DATA DA,DB /0.22617070509,0.4248322308/
03210      DATA DC,DM /0.15703,0.53687091209/
03220 1100  FORMAT(V)
03230C
03240      DO 20 I1=1,2
03250          DOSU(I1)=0
03260 20  CONTINUE
03270C
03280C ***** HINDO KEISAN *****
03290C
03300      DO 40 I=1,N
03310          CALL RANDOM(DA,DB,DC,DM,DX,RANSU)
03320          SYMBOL=INT(RANSU*10**5)
03330          DO 30 K=1,2
03340              IF(SYMBOL.LT.PS(K)) GO TO 30
03350              IF(SYMBOL.GE.PS(K+1)) GO TO 30
03360              DOSU(K)=DOSU(K)+1
03370              Y=K-1
03380              CALL INOUT(Y,INDEX1,POWER1,M1,1)
03390C
03400 30  CONTINUE
03410 40  CONTINUE
03420C----- CALCURATION PROBABILITY -----
03430      PS(1)=0
03440      P(1)=INT(FLOAT(DOSU(1))/N*10**5+0.5)
03450      PS(2)=P(1)
03460      PS(3)=10**5
03470      P(2)=PS(3)-PS(2)
03480      WRITE(6,1200) (P(I1),I1=1,2)
03490 1200  FORMAT(1H , " P' (I) ",2I6)
03500C----- CALCURATION ENTROPY -----
03510      ENTROPY=0.0
03520      DO 50 I=1,2
03530          F(I)=0.00001*P(I)
03540          IF(F(I).LE.0.0) GO TO 50
03550          SS=-1.0*F(I)*ALOG10(F(I))
03560          ENTROPY=ENTROPY+SS
03570 50  CONTINUE
03580      WRITE(6,1300) ENTROPY
03590 1300  FORMAT(1H0," ENTROPY (DIGIT) =",F15.8)
03600      RLENGTH=ENTROPY*N

```

```

03610      WRITE(6,1400) RLENGTH
03620 1400 FORMAT(1H," (SOURCE LENGTH)*H=",F15.8)
03630      RETURN
03640      END
03650C
03660C***** SUBROUTINE FOR WINDOW *****
03670C
03680      SUBROUTINE WINDOW
03690      INTEGER INDEXW,COUNTER,OUTSYM,Y
03700      INTEGER W(10),DOSU(10)
03710      COMMON /BLOCK8/ INDEXW,COUNTER,W,Y
03720      COMMON /BLOCK9/ DOSU
03730C
03740      IF(COUNTER.LE.10) GO TO 10
03750      INDEXW=MOD(INDEXW,10)+1
03760      COUNTER=1
03770 10 OUTSYM=INT(W(INDEXW)/10**9)
03780      W(INDEXW)=W(INDEXW)-OUTSYM*10**9
03790      W(INDEXW)=10*W(INDEXW)+Y
03800      COUNTER=COUNTER+1
03810C
03820      DOSU(OUTSYM+1)=DOSU(OUTSYM+1)-1
03830      DOSU(Y+1)=DOSU(Y+1)+1
03840      RETURN
03850      END
03860C
03870C***** SUBROUTINE FOR INITIAL W(I) *****
03880C
03890      SUBROUTINE INTWIN
03900      COMMON /BLOCK8/ INDEXW,COUNTER,W,Y
03910      INTEGER INDEXW,COUNTER,W(10)
03920C
03930      INDEXW=1
03940      COUNTER=1
03950      DO 10 I=1,10
03960          W(I)=101100011
03970 10 CONTINUE
03980      RETURN
03990      END

```



07/07/82 19:34:45

FILE NAME : A-CODE-7  
 INPUT HOW MANY TIME CHENGE THE THRESHOLD VALUE  
 =1

← 確率を何回変化させるか?  
 1回変化させる

INPUT THRESHOLD FOR RANDOM NUMBER : P(1),P(2) ; #####  
 =90000,10000

← 確率の指定値(しきい値)  
 $\hat{p}_0^{(1)} = 0.9, \hat{p}_1^{(1)} = 0.1$

P(1) 90000 10000

INPUT STRING LENGTH  
 =300000

← 上のしきい値での source の  
 長さ?  $n^{(1)} = 30万$

P'(1) 89951 10049

← 頻度から求める確率

ENTROPY (DIGIT) = 0.14164873  
 (SOURCE LENGTH)\*H= 42494.61962891

$\hat{p}_0^{(1)} = 0.89951, \hat{p}_1^{(1)} = 0.10049$

$H^{(1)} = 0.14164873$

$L_i^{(1)} = 42494.6$

INPUT THRESHOLD FOR RANDOM NUMBER : P(1),P(2) ; #####  
 =10000,90000

← 次のしきい値?  
 $\hat{p}_0^{(2)} = 0.1, \hat{p}_1^{(2)} = 0.9$

P(1) 10000 90000

INPUT STRING LENGTH  
 =300000

P'(1) 10011 89989

ENTROPY (DIGIT) = 0.14128667  
 (SOURCE LENGTH)\*H= 42386.00097656

RIRON LENGTH = 84880.61718750

←  $L_i = L_i^{(1)} + L_i^{(2)}$

INPUT NUMBER OF BLOCK PRINTING P''(1)  
 =10000

← source の長さ  $n$  より  $n$  とに  
 局所的な確率を指定する?

INPUT #YES --- IF YOU WANT SIGNAL & CODE  
 =##NO

15="と

← source signal to code string  
 を print する? No

I= 10000 94000 6000  
 I= 20000 90000 10000  
 I= 30000 93000 7000

← 15="との局所的  
 な確率

| CARRY OVER #02 |        |                            |
|----------------|--------|----------------------------|
| I=             | 39245  | COUNT= 0 C(SY)= 9999081370 |
| I=             | 40000  | 91000 9000                 |
| I=             | 50000  | 87000 13000                |
| I=             | 60000  | 92000 8000                 |
| I=             | 70000  | 85000 15000                |
| I=             | 80000  | 87000 13000                |
| I=             | 90000  | 87000 13000                |
| I=             | 100000 | 86000 14000                |

← "bit stuffing" を行う時の  
 source の長さ I= 39245  
 符号の長さ C(SY)= 9999081370  
 ↑  
 付加した "0"

| CARRY OVER #02 |        |                            |
|----------------|--------|----------------------------|
| I=             | 102392 | COUNT= 1 C(SY)= 9990573580 |

← count はレジスタ V の外に出て、メモリに記憶  
 されている "9" の連続した列の "9" の個数  
 この数では count=1 からの次のようになる。

| CARRY OVER #02 |        |                            |
|----------------|--------|----------------------------|
| I=             | 106198 | COUNT= 0 C(SY)= 9999081400 |
| I=             | 110000 | 92000 8000                 |
| I=             | 120000 | 89000 11000                |
| I=             | 130000 | 89000 11000                |
| I=             | 140000 | 88000 12000                |
| I=             | 150000 | 89000 11000                |
| I=             | 160000 | 87000 13000                |

メモリ  
 V W  
 --- 9 99995 73580  
 --- 99 99905 73580  
 ↑  
 付加した "0"

↑ "bit stuffing"

CARRY OVER #02  
I= 102392 COUNT= 1 C(SY)= 9990573580

CARRY OVER #02  
I= 106198 COUNT= 0 C(SY)= 9999081400  
I= 110000 92000 8000  
I= 120000 89000 11000  
I= 130000 89000 11000  
I= 140000 88000 12000  
I= 150000 89000 11000  
I= 160000 83000 17000  
I= 170000 93000 7000  
I= 180000 91000 9000  
I= 190000 85000 15000  
I= 200000 90000 10000  
I= 210000 84000 16000  
I= 220000 90000 10000  
I= 230000 85000 15000  
I= 240000 87000 13000  
I= 250000 92000 8000  
I= 260000 93000 7000  
I= 270000 92000 8000  
I= 280000 93000 7000  
I= 290000 91000 9000  
I= 300000 89000 11000  
I= 310000 7000 93000  
I= 320000 9000 91000  
I= 330000 6000 94000  
I= 340000 10000 90000  
I= 350000 8000 92000  
I= 360000 10000 90000  
I= 370000 6000 94000  
I= 380000 11000 89000  
I= 390000 10000 90000  
I= 400000 14000 86000  
I= 410000 12000 88000  
I= 420000 16000 84000  
I= 430000 10000 90000  
I= 440000 8000 92000  
I= 450000 12000 88000  
I= 460000 11000 89000  
I= 470000 6000 94000  
I= 480000 5000 95000  
I= 490000 3000 97000  
I= 500000 11000 89000  
I= 510000 6000 94000  
I= 520000 10000 90000  
I= 530000 7000 93000  
I= 540000 8000 92000  
I= 550000 11000 89000  
I= 560000 12000 88000  
I= 570000 8000 92000  
I= 580000 7000 93000  
I= 590000 9000 91000

CARRY OVER #02  
I= 592294 COUNT= 0 C(SY)= 9999041100  
I= 600000 10000 90000

CODE LENGTH= 86323

(CODE LENGTH)-(ENTROPY\*(SOURCE LENGTH))= 1442.383

CODE EFFICIENCY= 98.30069160  
DECODED WITHOUT ERROR

\*RUNH

← 符号の長さ  $L_c$   
← 差  $d = L_c - L_i$   
← 効率  $e = 1 - \frac{d}{L_i}$   
← 正しく decode した！

## 参考文献

- [1] Shannon, C.E., "A Mathematical Theory of Communication," Bell System Tech. J., vol. 27, pp. 379-423 and pp. 624-656, No. 3, July. 1948
- [2] Fano, R.M., Technical Report No. 65, The Research Laboratory of Electronics, M.I.T., 1948
- [3] Huffman, D.A., "A Method for the Construction of Minimum-Redundancy Codes," Proc. IRE, vol. 40, No. 9, pp. 1098-1101, Sept. 1952
- [4] Hunter, R. and Robinson, R.H., "International Digital Facsimile Coding Standards," Proc. IEEE, vol. 68, pp. 854-867, July., 1980
- [5] Musmann, H.G. and Preuss, D., "Comparison of Redundancy Reducing Codes for Facsimile Transmission of Documents," IEEE Trans. on Comm., vol. COM-25, Nov. 1977
- [6] Wyle, H., Erb, T. and Banow, R., "Reduced-Time Facsimile Transmission by Digital Coding," IRE Trans. on Commun. System, pp. 215-222, Sep. 1961
- [7] Dorreca, G., Desai, U.D. and Van Allen, R.L., "Explorer XII Satellite Instrumentation for Study of Energy Spectrum of Cosmic rays," Proc. Nat'l Telemetering Conf., May 1962
- [8] Golomb, S.W., "Run-Length Encodings," IEEE Trans. Inform. Theory, IT-12, 4 pp. 399-401, July. 1966
- [9] Graphic Sciences, Inc. et al, ( no title, one-dimensional run length code proposed for standardization ), CCITT Study Group XIV, temp. docum. No. 7, Sept. 1976
- [10] Meyer, H., H.G. rodolsky, F. Schaefer, and T.S. Huang, "Optimum run-length codes," IEEE Trans. on Comm., vol. COM-22, pp. 826-835, June 1974
- [11] Elias, P., "Predictive encoding, parts I and II," IRE Trans. on Inform. Theory, vol. IT-1, pp. 16-33, March 1955
- [12] 高木, 津田 : ニ次元予測を用いたファクシミリの変域圧縮, 電信学論 D, vol. 56-D, pp. 170-177, March 1973

- [13] H.kobayashi and Bahi,L.R.,"Image data compression by predictive coding I : prediction algorithms," IBM J.Res. Develop.,vol.18,No.2,pp.164-171,Mar.1974
- [14] Netravali,A.N.,Mounts,F.W.,and Bowen,E.G.,"Ordering Techniques for Coding of Tow-Tone Facsimile Pictures," Bell Sys.Tech.J.,vol.55,No.10,pp.1539-1552,Dec.1979
- [15] Netravli,A.N.,and Mounts,F.W.,"Ordering Techniques for Facsimile Coding : A Review," Proc.of IEEE,vol.68,No.7 pp.796-807,July.1980
- [16] Preuss,D.,"Twodimensional Facsimile Source Encoding Based on a Markov Model," Nachrichtentechnik.z.28,H.10,S.358-363, 1975
- [17] 大西ら, Optimization of Facsimile Data Compression, NTC '77
- [18] 若原ら, ファクシミリ信号の変化点相対アドレス符号化方式の圧縮率, 画像電子学会誌, vol.5, no.3, pp.92-100, 1976
- [19] 山田ら, 高速ファクシミリの変長圧縮符号化方式, 研実報, vol.28, no.5, pp.833-849, 1979
- [20] 山本ら, ファクシミリ2次元符号化方式の国際標準化 -モディファイド-リッド(MR)方式-, テレビジョン学会誌, vol.34, no.5, pp.410-413, 1980
- [21] British Post Office,"Data compression statistics for the R2 coding scheme," CCITT Study Group XIV.Contrib.,Oct.1979
- [22] 富田ら, 漢字パターン圧縮の -方式, 電気通信学会全国大会 1975 972
- [23] 新井ら, 画素形漢字データ圧縮の二,三の方法, 画電学誌, vol.6, No.1, pp.16-24, Jan. 1977
- [24] 堀口ら, 画素形漢字パターンの伝送のためのデータ圧縮に関する -考察, 電信学論 D, vol. J63-D, no.5, pp.371-377, May 1980
- [25] 森ら, 画素形漢字パターンのデータ圧縮に関する二,三の考察, 電信学論 D, vol. J64-D, no.7, pp.617-624, July 1981
- [26] Nagao,M.,"Data Compression of Chinese Character Patterns," Proc.of IEEE,vol.68,No.7,pp.8180829,July.1980
- [27] Kunt,M.,and Johnsen,O.,"Block Coding of Graphics : A Tutorial Review," Proc.IEEE,vol.68,pp.770-785,July.1980

- [28] 画像処理サブルーチン・パッケージ SPIDER USERMANUAL,  
電子技術総合研究所, Sep. 1980
- [29] Freeman, H., "Computer processing of line drawing images,"  
Computing Surveys 6 (1), March 1974, 57-97
- [30] 遠藤ら, DF-画像表現の性質と情報圧縮への応用,  
電信学論 D, vol. J62-D, no. 2, pp. 141-148, Feb. 1979
- [31] Rosenfeld, A., "Connectivity in Digital Pictures," Journal  
of the ACM, vol. 17, No. 1, pp. 146-160, Jan. 1970
- [32] Stefanelli, R., and Rosenfeld, A., "Some Parallel Thinning  
Algorithms for Digital Pictures," Journal of the ACM,  
vol. 18, No. 2, pp. 255-264, April 1971
- [33] Deutsch, E. S., "Thinning Algorithms on Rectangular, Hexagonal,  
and Triangular Arrays," CACM, vol. 15, pp. 827-837, Sep. 1973
- [34] Izzo, N. F., and Coles, W., "Blood-Cell Scanner Identifies,"  
Electronics, 35, 27, pp. 52-57, April 1962
- [35] Papert, S., "Uses of Technology to Enhance Educator,"  
Technical Report 298, AI Lab, MIT, 1973
- [36] Selkow, S. W., "One-Pass Complexity of Digital Pictures  
Properties," J. ACM, vol. 19, pp. 283-259, April 1972
- [37] Rosenfeld, A., and Milgram, D. L., "Parallel/Sequential Array  
Automata," Information Processing Letters 2 (1973) 43-46
- [38] Shah, A., "Pushdown Automata on Arrays," Information Sciences  
25, pp. 175-193, 1981
- [39] 横井ら, 標本化された2値図形のトポロジカルな性質について,  
電信学論 D, vol. 56-D, no. 11, pp. 662, Nov. 1973
- [40] Preston, K. Jr., "The CELLSCAN System, A Leucocyte Pattern  
Analyzer," Proc. W. J. C. C., 19, p. 173 May 1961
- [41] Levialdi, S., "Parallel Counting of Binary Patterns,"  
Electron. Lett. p. 798, Dec. 1970
- [42] Minsky, M., and Papert, S., PERceptrons, MIT Press, New York,  
1969
- [43] 横井ら, 2値図形収縮のための逐次形アルゴリズムについて,  
電信学論 D, vol. J62-D, no. 8, pp. 537-542, Aug. 1979
- [44] Weston, P., "Photocell Fields Counts Random Objects,"  
Electronics, 34, 22, p. 46 Sep. 1961

- [45] Habibi, A. and P.A. Wintz, "Image Coding by Linear Transformation and Block Quantization," IEEE Trans. Comm. Technol., vol. COM-19, pp. 50-62, 1971
- [46] Rosenfeld, A., and A.C. Kak, Digital Picture Processing, Academic Press New York, 1976
- [47] Netravali, A.N. and Limb, J.O., "Picture Coding : A Review," Proc. IEEE, vol. 68, pp. 366-406, March 1980
- [48] de Jager, F., "Delta modulation, a method of PCM transmission using a 1-unit code," Philips Res. Rep. 7, pp. 442-466, July 1976
- [49] O'Neal, J.B. Jr., "Predictive Quantizing Systems (Differential Pulse Code Modulation) for the Transmission of Television Signal," Bell System Tech. J., vol. 45, pp. 689-721, May-June 1966
- [50] Pratt, W.K., J. Kane, and H.C. Andrews, "Hadamard Transform Image Coding," Proc. IEEE 57, pp. 58-68, 1969
- [51] Wintz, P.A., "Transform Picture Coding," Proc. IEEE 60, pp. 809-820, 1972
- [52] Schwartz, J.W., and Barker, R.C., "Bit-Plane Encoding : A Technique for Source Encoding," IEEE Trans. Aerospace and Electronics System, vol. AES-2, pp. 385-392, July 1966
- [53] Schwartz, J.W. and Huang, T., "Bit-Plane Encoding of continuous Tone Picture," Polytechnic Institute of Brooklyn, New York April 1969
- [54] Kretzmer, E.R., "Statistics of Television Signal," Bell Syst. Tech. J., vol. 31, pp. 751-763, July 1952
- [55] Lynch, T.J., "Sequence Time Coding for Data Compression," Proc. of IEEE, vol. 54, No. 10, pp. 1490-1491, Oct. 1966
- [56] Cover, T.M., "Enumerative Source Encoding," IEEE Trans. Inform. Theory, vol. IT-19, pp. 73-77, Jan. 1973
- [57] Schalkwijk, J.P.M., "An Algorithm for Source Coding," IEEE Trans. Inform. Theory, vol. IT-18, pp. 395-399, May 1972
- [58] Rissanen, J.J., "Generalized Kraft Inequality and Arithmetic Coding," IBM J. RES. Develop., pp. 198-203, May 1976
- [59] Pasco, R.C., "Source Coding Algorithms for Fast Data Compression," Ph.D. thesis, Dept. of Electrical Engineering, Stanford University, CA, 1976

- [60] Tasto,M. and Wintz,P.A,"Image Coding by Adaptive Block Quantization," IEEE Trans.Commun.Technol.,vol.COM-19, pt.1,pp.957-972,Dec.1971
- [61] Netravali,A.N.,Prasada,B.,and Mounts,F.W., "Some Experiments in Adaptive and Predictive Hadamard Transform Coding of Pictures," Bell.System tech.J.,vol.56,pp.1531-1547,Oct.1977
- [62] Habibi,A., "Survey of Adaptive Image Coding Techniques," IEEE Trans.on Comm.,vol.COM-25,pp.1275-1284,Nov.1977
- [63] Usubuchi,T.,Omachi,T.,and Iinuma,K., "Adaptive Predictive Coding for Newspaper Facsimile," Proc.of the IEEE,vol.68, No.7,pp.807-813,July 1980
- [64] Rissanen,J.,and Langdon,G.G.,Jr., "Arithmetic Coding," IBM J. RES.Develop.,vol.23,pp.149-162,March 1979
- [65] Rissanen,J., "Arithmetic Coding sa Number Representations," Acta Polytech.Scandinavia Ma31,Helsinki,pp.44-51,1979
- [66] Elias,P. unpublished results in 60's
- [67] Jelinek,F., "Buffer Overflow in Variable Length Coding of Fixed Rate Source," IEEE Trans.Inform.Theory, vol.IT-14,pp.490-501,May 1968
- [68] Rissanen,J. and Langdon,G.Jr, "Universal Modeling and Coding," IEEE Trans.on Inform.Theory,vol.IT-27,pp.12-22, Jan.1981
- [69] Guazzo,M., "A General Minimum-Redundancy Source-Coding Algorithm," IEEE Trans.on Inform.Theory,vol.IT-26, pp.15-25,Jan,1980
- [70] Jones,C.B., "An Efficient Coding System for Long Source Sequences," IEEE Trans.on Inform.Theory,vol.IT-27,No.3 pp.280-291,May 1981
- [71] Langdon,G.Jr.,and Rissanen,J., "Compression of Black-White Images with Arithmetic Coding, : IEEE Trans.on Commun.,vol.COM-29,No.6,pp.858-867,June 1981

- [72] Langdon, G. Jr., and Rissanen, J., "A Simple General Binary Source Code," IEEE Trans. on Inform. Theory, vol. IT-28, No. 5, pp. 800-803, Sep. 1982
- [73] Aho, A. V., J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Algorithms, Addison-Wesley, 1974