アルゴリズム 5－1
（1989. 1. 25）

文字正規化の並列処理アルゴリズム

大町　真一郎　　阿曽　弘具　　木村　正行

東北大学工学部

近年、情報処理技術の発展により、文書の管理が計算機によって行なわれるようになってきた。それに伴い、既存の文書を自動的に読み取り、計算機に効率よく入力する光学読み取り装置（ＯＣＲ： Optical Character Reader）の需要が増えてきた。漢字を読み取れる低価格の装置も市場に出て来つつあるが、認識速度や認識率などの点でまだ改良の余地は多い。本報告では、この２つの要素のうち認識速度の改善を目的に、漢字認識の前処理である正規化を並列に処理するＶＬＳＩ向きアルゴリズムを提案する。

A PARALLEL PROCESSING ALGORITHM FOR NORMALIZING CHARACTER PATTERNS

Shin'ichiro OHMACHI　　Hirotomo ASO　　Masayuki KIMURA

Faculty of Engineering, Tohoku University
Sendai 980, Japan

Recently, as the progresses of the technology of information processings, computers become to manage documents. For this situation, the demand for an optical character reader(OCR) is occured that reads documents automatically and inputs them into a computer. Although low cost recognition systems that can read Japanese characters are coming into the market, recognition speed, recognition rates and others leave plenty of room for improvement. In order to accelerate the recognition speed, in this paper we propose a VLSI-oriented algorithm for parallel processing of normalizing character patterns which is one of the preprocessings for character recognition.

1．まえがき

近年、情報処理技術の発展により、文書の管理が計算機によって行なわれるようになってきた。文書を計算機で管理することにより、文書の保存・変更・検索が容易になる。しかし、既存の文書を人間のオペレーターが入力するのは大変な手間がかかるため、文書を計算機に自動的に読み取らせ、効率よく入力する光学読み取り装置（ＯＣＲ：Optical Character Reader）についての研究がさかんに行なわれてきている。漢字を読み取れる低価格の製品も、市場に出て来つつある[3]。

ところで、ＯＣＲに求められる条件としては、

①認識率が高いこと。

②認識が高速に行えること。

の２つが挙げられる。このうち、認識率を上げるためには、文字認識アルゴリズムそのものを改良する必要がある。認識速度の向上のためには、高速な文字認識アルゴリズムを用いることが必要であるが、漢字認識では字種数も多く（JIS第一水準だけで約3000種類）、計算量が膨大になる。従って、１個のプロセッサのみで処理を行なったのでは限界があり、処理速度を更に上げるためには並列処理を行なう必要がある。

文字認識の手法としては、これまでに様々なものが開発されてきた。文字パターンを重ね合わせたときの類似度をもとに認識する方法（重ね合わせ法）をはじめ、パターンからベクトルを抽出して特徴量とする方法、パターンから線分を抽出して線分間の関係を調べて文字を決定する方法（構造解析法）などがある。我々の研究室で現在用いているのは、以下の方法である。まず、未知入力パターンを正規化・細線化・線素化し、線素化された図形から方向線素ベクトル[4]を抽出して特徴量とする。未知入力パターンから線素化された図形を得るまでの流れを図１に示す。(b)のノイズ除去・スムージングは、スキャナによる読み取りの際に生じるノイズや図形のかすれなどの影響を取り除くために行なう。(c)の正規化は、入力図形を一定の大きさに揃える処理であり、位置や大きさの違いによる影響を取り除くために必要である。これらは文字の特徴量を抽出する以前の処理であり、文字認識における前処理とも呼ばれる。(d)の細線化は、太さを持った図形を幅１の図形に変換する。(e)の線素化は、細線化された図形の各黒画素について、その周囲の状況からどの方向の線素であるかを判断する。線素化が終了した後はこの方向線素パタンから方向線素ベクトルを抽出し、連想整合法[4]で、あらかじめ用意しておいた標準パターンのベクトルとのマッチングを行なうことによって、候補を選出する。

ベクトル化された後の候補選出をシストリックアルゴリズムを利用して並列に処理する一方法は、既に提案した[5]。

また、図１の各処理のうち、ノイズ除去・スムージング・細線化・線素化は、各画素の近傍のみに注目し

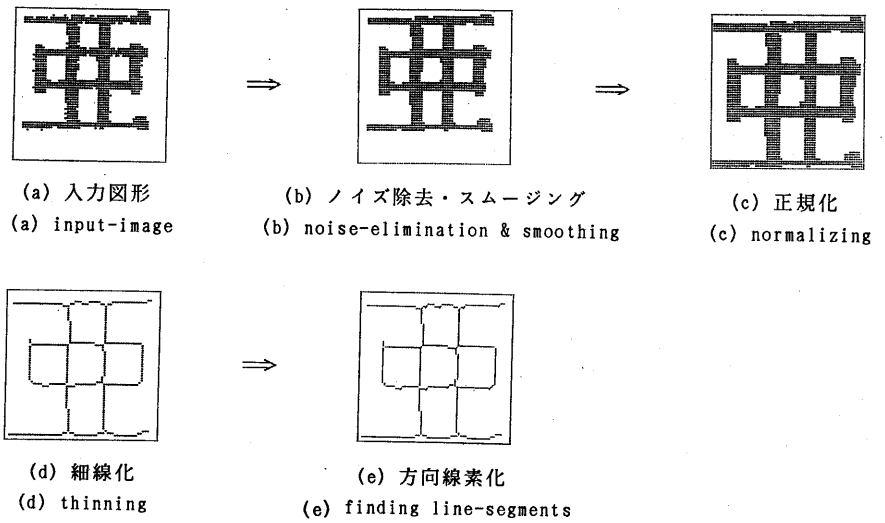
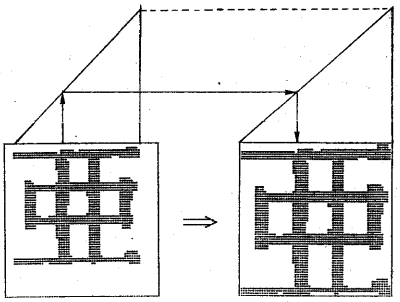
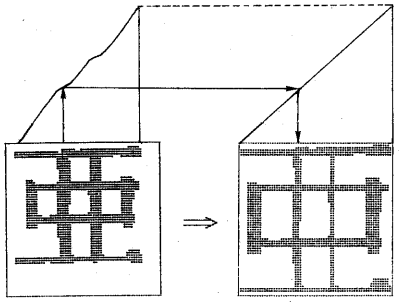


図１　方向線素化までの入力図形の処理

Fig.1　Processing of finding line-segments for input-image



(a) 線形正規化
(a) Linear normalizing



(b) 非線形正規化
(b) Non-linear normalizing

図2　二種類の正規化
Fig. 2　Two types of normalizing

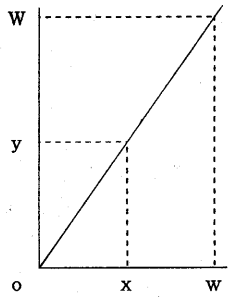


図3　線形変換
Fig. 3　Linear transform

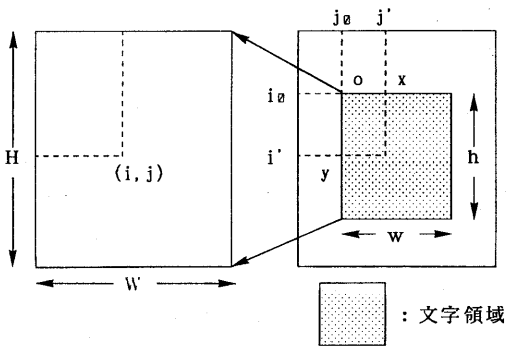


図4　線形正規化
Fig. 4　Linear normalizing

た操作を繰り返すことによって実現することができるが、このような局所的な反復処理をパイプライン形アルゴリズムを用いて効率的に行なう方法は、既に提案されている[2][3]。しかし、正規化では画像全体を見て位置や大きさを判断しなければならないため、従来は逐次的な方法で行なわれてきており、認識速度を制限するものとなっていた。本稿では、正規化を局所的な演算とデータ転送のみで実現できる、ＶＬＳＩ向き並列処理アルゴリズムを提案する。n×nの領域にある文字図形の正規化は、従来は$O(n^2)$の時間を必要としていたが、このアルゴリズムにより、O(n)に高速化できる。

2．二種類の正規化アルゴリズム

正規化は、先に述べたように、もとの図形を一定の大きさに拡大（または縮小）し、位置や大きさの違いによる影響を吸収しようとする処理である。最も単純な正規化は、入力図形を縦・横両方向に一定の大きさに線形伸縮することで実現できる。これを線形正規化と呼ぶ。線形正規化は、具体的には図2(a)のような直線の変換関数を用いた座標変換で行える。また、これとは別に、入力文字の線密度の大きい部分を拡大、線密度の小さい部分を縮小し、全体を一定の大きさにすることで、線密度を全体に分散させようとする正規化も考えられている。これを非線形正規化と呼ぶ。非線形正規化は、まず図2(b)のような、線密度の大きい部分の傾きが大きく、線密度の小さい部分の傾きが小さい変換関数を求め、これをもとに座標変換することで得られる。　なお、変換関数は、線形・非線形の場合とも、横方向（列方向）についてのみ示してある。

活字のように線間隔が文字によってほぼ一定のものは線形正規化でよいが、手書き文字のように線間隔が不規則なものは、非線形正規化を行なうことによって線密度をある程度分散した方が認識率が上がる。

次に、線形・非線形の正規化についてもう少し具体的に述べる。

まず、１次元の幅ｗの図形を幅Ｗに線形拡大するこ

とを考えてみる。これは、図3のx軸上の$1 \leqq x \leqq w$の領域をy軸上の$1 \leqq y \leqq W$の領域に写像する問題と考えられる。点xに対応する点yの決め方は幾通りかあるが、ここでは、

$$y=\lceil x \cdot W/w \rceil \qquad (1)$$

と定める。ただし、$\lceil\ \rceil$は切上げを表わす。

2次元図形も同様に線形拡大することができる。いま、入力図形で、黒画素が存在する最も上の行をi_0、最も左の列をj_0とし、図形の幅をw、高さをhとする（図4）。正規化後の図形の幅、高さをそれぞれW、Hとすれば、正規化後のi行j列に対応するもとの図形の座標は、

$$(i_0+\lceil i \cdot h/H \rceil,\ j_0+\lceil j \cdot w/W \rceil)$$

である。

$$y=\lceil i \cdot h/H \rceil,\ x=\lceil j \cdot w/W \rceil \qquad (2)$$

とおけば、

$$y-1 < i \cdot h/H \leqq y,\ x-1 < j \cdot w/W \leqq x \qquad (3)$$

すなわち、

$$(y-1) \cdot H < i \cdot h \leqq y \cdot H \qquad (4)$$

$$(x-1) \cdot W < j \cdot w \leqq x \cdot W \qquad (5)$$

となる。ここで、行・列方向の変換テーブル f(i'), g(j')を、

$$\left.\begin{array}{l} f(i')=0\ ;\ i'<i_0 \\ g(j')=0\ ;\ j'<j_0 \\ f(j_0)=g(j_0)=1 \\ f(i')=f(i'-1)+1\ ;\ i'>i_0 \\ g(j')=g(j'-1)+1\ ;\ j'>j_0 \end{array}\right\} \qquad (6)$$

と定義すれば、(4)(5)式は、

$$f(i'-1) \cdot H < i \cdot h \leqq f(i') \cdot H \qquad (7)$$

$$g(j'-1) \cdot W < j \cdot w \leqq g(j') \cdot W \qquad (8)$$

と表わされる。正規化後の図形のi行j列の画素を、(7)(8)式を満たす入力図形のi'行j'列の画素とすることにより、線形正規化を行なうことができる。

非線形正規化の場合は、次の手順で非線形の変換テーブルを求め、(7)(8)式を適用すればよい。

①ストローク密度を求める。

②ストローク密度を累積する。

③線形の変換テーブルの値を加える。

ただし、ストローク密度とは、図形を行方向（または列方向）にスキャンしていったときに、黒画素の領域を横切った回数である。また、(7)(8)式の'W' 'H'は、それぞれ、画素の存在する最も右の列と最も下の行の変換テーブルの値とする。列方向の非線形の変換テーブルを求める過程を図5に示す。

3．正規化の並列処理アルゴリズム

本章では、2.の考え方を適用し、正規化を並列に処理するセル構造回路網を構成する。全体の構成を図6に示す。但し、入力画像データはM×Nの二値図形とし、D×Dの大きさに正規化するものとする。

全体は、幅高さ計測モジュールと正規化モジュールから成る。幅高さ計測モジュールでは線形または非線形の変換テーブルを生成し、正規化モジュールではその変換テーブルをもとに正規化を行なう。

以下、各モジュールの機能について詳しく述べる。

3.1 幅高さ計測モジュール

幅高さ計測モジュールの構成を図7に示す。これは、M×Nの入力バッファと、N個の幅高さ計測セル及び1クロックの遅延機能を持つ(M-1)N個のセルから成る。幅高さ計測セルで、線形及び非線形の変換テーブルを生成する。

3.1.1 モジュールの機能とデータの流れ

入力データは最初バッファに格納され、1行ずつセルに送られて処理される。

配置されたセルはすべてクロックに同期して動作する。いま、データをバッファに格納した時刻をt＝0、1クロックごとにtが1ずつ増えていくものとして、時刻tでの各セルの動作を次のように定める。

① $t \leqq M$のとき、セル(N,1)～セル(N,N)は入力データのt行目を読み込む。

$M < t$のとき、セル(N,1)～セル(N,N)は0を読み込む。

② 逆対角線上にある幅高さ計測セル(i,N-i+1)は、セル(i+1,N-i) から送られてきた演算結果と、セル

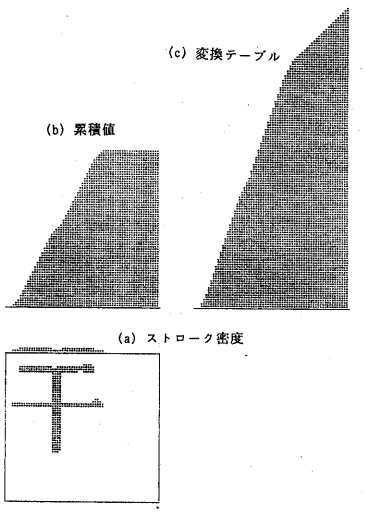


図5 非線形正規化の変換テーブルを求める過程

Fig.5 Process of making transform-table for non-linear normalizing

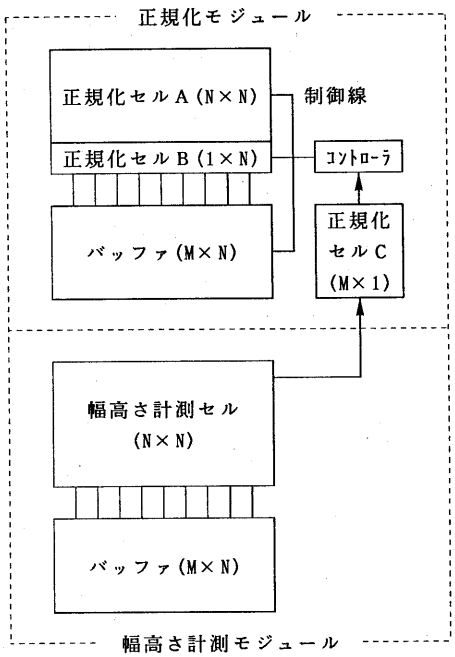


図6　全体の構成

Fig.6　Configuration

(i+1,N-i+1) から送られてきたデータを用い、後述のセル機能にしたがって演算を行なう。

③　幅高さ計測セル(i,N-i+1)は、演算結果をセル(i-1,N-i+2)に送る。また、1～N－1行のすべてのセル(i,j)は、セル(i+1,j)から送られてきたデータをそのままセル(i-1,j)に送る。

但し、N≦tのとき、セル(1,N)の演算結果は正規化セルCに格納していく。この値は、後述するように入力データの行方向変換テーブル ((N-t+1)行目)の値となっている。

以上の動作は、M＋N－1クロックですべて完了し、j列目の幅高さ計測セルに入力データのj列目の列方向変換テーブルの値が格納され、正規化セルBのi行目に入力データのi行目の行方向変換テーブルの値が格納されることになる。

セルを図7のように配置することにより、入力図形のi行目の各列のデータは、時刻iで1列目のデータがセル(N,1)で処理され、時刻(i+1)でその処理結果と2列目のデータがセル(N-1,2)で処理される、という具合に同じ行のデータが順に出会っていくので、行方向と列方向の変換テーブルが同時に作られる。

次に、幅高さ計測セルのセル機能を、線形の場合と非線形の場合に分けて示す。

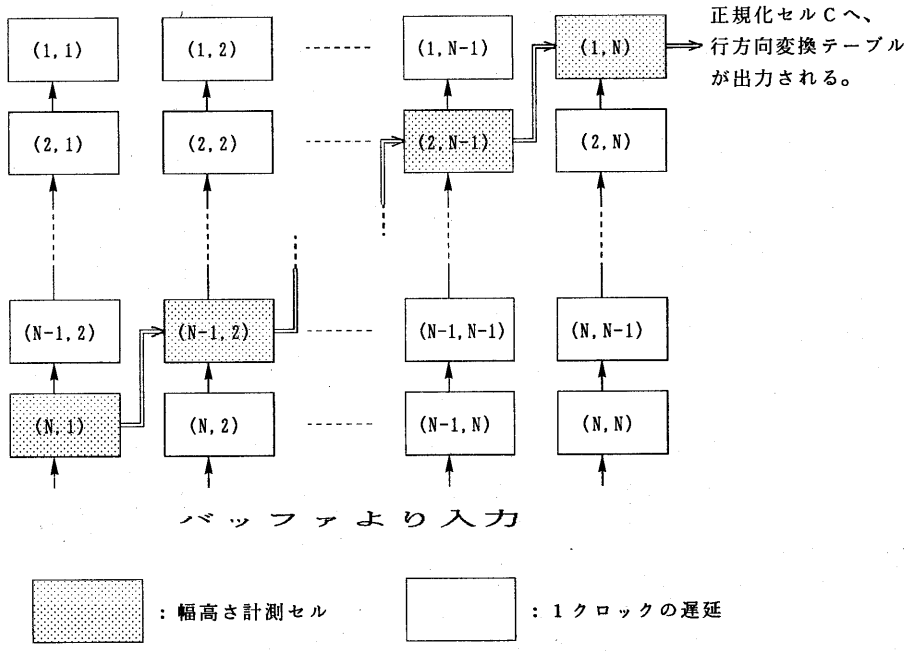


図7　幅高さ計測モジュール

Fig.7　Width･height-measuring module

3.1.2 線形の場合のセル機能

ここでは列方向の変換テーブルを例にとって示す。

列方向の変換テーブルは、前述したように、画像データ上で文字の領域に含まれる最も左の点が属する列の値を1とし、それより右の列は1ずつ値を増やしていくことで得られる。従って、入力画像のある列を走査したとき、もしその列に黒画素が含まれていて、しかもその列より左の列には黒画素が含まれていないとき、その列の変換テーブルの値を1とし、その列より右の列は、変換テーブルの値を1ずつ増やしていけばよい。

また、文字の幅は、入力画像データ上の最も右の黒画素の存在する列の変換テーブルの値に等しく、これを求めればよい。

なお、行方向の変換テーブルも、全く同様の考え方で得ることができる。但し、列方向のときに空間的に配置させていた機能を時間的に、時間的に配置させていた機能を空間的に配置することになる。

セル機能を関数で表わしたものを図8(a)に示す。図で、flagは黒画素があったかどうかを判定するフラグであり、countは変換テーブルの値を格納し、maxは文字幅の値を格納する。'lc'は列方向、'lr'は行方向であることを表わしている('l'はlinear, 'c'はcolumn, 'r'はrow)。また、lr_flag, lc_max, lc_countは、左の列のセルの値を用いて処理して結果を右のセルに送り、lc_flag, lr_max, lr_count は自分のセルの値を用いて処理して結果を自分のセルに格納する。また、各代入文の左辺が時刻tでの値であるとすれば、右辺は時刻(t-1)での値である。

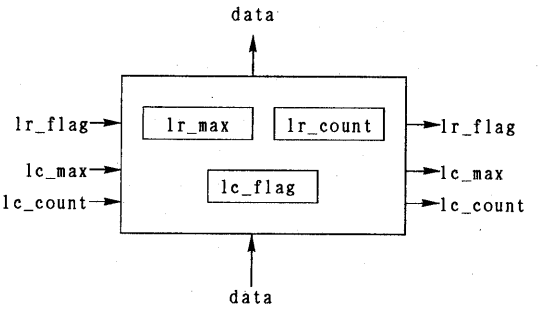
なお、セル機能は列方向の関数のみについて示してあるが、行方向の場合は、式中の'lc_'を'lr_'に読みかえて得られる。

このようなセルを用いて処理を行なうと、最終的に、列方向の変換テーブルは各セル内のlc_countに、文字幅はセル(1,N)のlc_maxに、文字の高さはセル(1,N)のlr_maxに格納され、行方向の変換テーブルは、時刻N以降セル(1,N) のlr_countとして順に出力されることになる。

3.1.3 非線形の場合のセル機能

非線形の場合は、ストローク密度を累積した値(図5(b))を求めれば、あとは3.1.3の線形正規化用変換テーブルの値を加算すればよいので、ここではストローク密度を累積するセル機能を与える。図7の幅高さ計測セルは、前節で述べた線形のセル機能と本説で述べる非線形のセル機能の両方を持ち、外からの制御により、線形正規化の場合は線形の機能のみを、非線形正規化の場合は線形と非線形の両方の機能を動作させればよい。

セル機能を関数で表わしたものを図8(b)に示す。図で、flagは黒画素を横切っている途中であるかどう

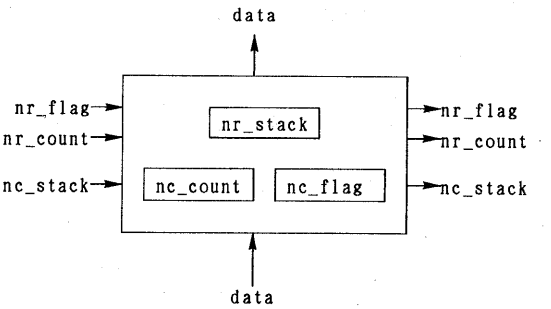


列方向の関数

```
lc_flag = if data==1 then 1
          else lc_flag
lc_count = if lc_count>0 then lc_count+1
           else if lc_flag==1 or data==1 then 1
           else lc_count
lc_max= if lc_flag==1 or data==1 then lc_count+1
        else lc_max
```

(a) 線形の機能

(a) Function for linear



列方向の関数

```
nc_flag = if data==0 then 0
          else 1
nc_count = if data==1 and nc_flag==0
                    then nc_count+nc_stack+1
           else nc_count+nc_stack
nc_stack = if data==1 and nc_flag==0
                     then nc_stack+1
           else nc_stack
```

(b) 非線形の機能

(b) Function for non-linear

図8 幅高さ計測セルの機能

Fig.8 Function of width·height-measuring cell

かを判定するフラグであり、countは累積値を格納する。nc_stackはある行内、nr_stackはある列内で、それぞれ白画素から黒画素に変化する点を横切った回数を格納する。'nc' 'nr' は列方向、行方向の計算に使われることを表わしている（'n' は non_linearを表わす）。また、nr_flag,nr_count,nc_stack は、左の列のセルの値を用いて処理して結果を右のセルに送り、nc_flag,nc_count,nr_stack は自分のセルの値を用いて処理して結果を自分のセルに格納する。なお、行方向のセル機能を表わす関数は、列方向の関数の'nc_' を 'nr_' に変更することで得られる。

このようなセルを用いて処理を行なうと、最終的に、列方向の累積値は各セル内のnc_countに格納され、行方向の累積値は、時刻N以降セル(1,N)の nr_countとして順に出力されることになる。

3.2 正規化モジュール

正規化モジュールの構成を図9に示す。これは、N×N個のセルA、N個のセルB、M個のセルC、読み込みコントローラから成る。セルAでは列方向の正規化を行ない、セルB，Cではそれぞれ幅高さ計測モジュールで作られた列・行方向変換テーブルを読み込んで正規化に必要な値を計算する。また、読み込みコントローラでは、バッファからの読み込みを制御することによって行方向の正規化を行なう。

なお、以下では正規化後の大きさをD×D (D<M, D<N)として議論するが、行・列方向の大きさが違っても同様に正規化することができる。

3.2.1 正規化モジュールの初期化

(7)(8)式の考えで行・列両方向の正規化を行なうため、最初に、正規化に必要な値を計算しておく。

セルB(j) では、まず、幅高さ計測モジュールで計算された値から、j列と(j-1)列の変換テーブルg(j), g(j-1)を求める。線形正規化の場合と非線形正規化の場合のj列の変換テーブルg(j)は、

線形の場合：g(j)=lc_count(j)

非線形の場合：g(j)=lc_count(j)+nc_count(j)

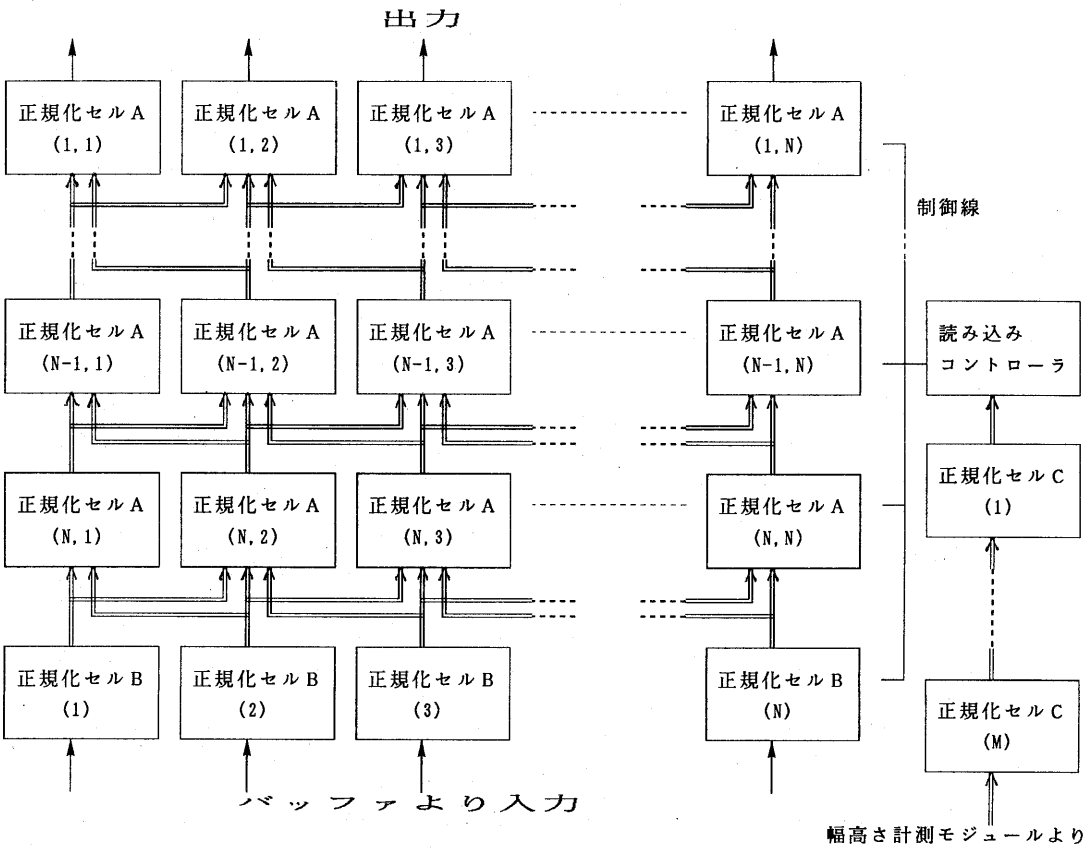
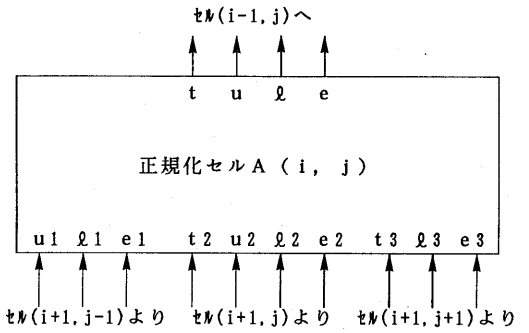


図9　正規化モジュール

Fig.9　Normalizing module



セル機能

```
t = t2
u = if ℓ2≧ t2 then u1
    else if u2< t2 then u3
    else u2
ℓ = if ℓ2≧ t2 then ℓ1
    else if u2< t2 then ℓ3
    else ℓ2
e = if ℓ2≧ t2 then e1
    else if u2< t2 then e3
    else e2
```

図１０　正規化セルA

Fig.10　Normalizing cell A

と設定する。但し、‘(j)’は、幅高さ計測セル(N-j+1, j)の値であることを意味する。

また、(8)式の‘w’に相当する値は、

線形の場合：w=lc_max(N)

非線形の場合：w=lc_max(N)+nc_count(N)

と設定する。

以上の値を用い、セルB(j)では、

$t = i \cdot w$ ， $u = D \cdot f(j)$ ， $\ell = D \cdot f(j-1)$　　(9)

を計算しておく。

また、セルC(i)には行方向変換テーブルf(i)を、読み込みコントローラには(7)式の‘h’に相当する値を、それぞれg(i), wと同様に計算し、格納しておく。

入力画像は、M×Nのバッファに格納する。

3.2.2　列方向の正規化

列方向の正規化は、セルAを用いて行なう。初期値としてj'=jとし、

$D \cdot g(j'-1) \geqq j \cdot w$　ならば　$j' \rightarrow j'-1$

$D \cdot g(j') < j \cdot W$　ならば　$j' \rightarrow j'+1$

と変換していけば最終的に(8)式を満たすj'の値が得られることを利用する。正規化セルAを図９のように配置し、一段ごとに変換を行なうものとすると、セルAには図１０の機能を持たせておけばよい。但し、図中の‘e’は、画素（１ビット）を、t, u, ℓは(9)式の値を表わす。

3.2.3　行方向の正規化

行方向の正規化は、前述したように、(7)式をもとにバッファからセルBへ送る行をコントローラで制御することによって行なう。その手順は次のようになる。

①i←1, t←1とおく。

②セルC(i)から変換テーブルf(i)を読み込む。

③もしt･h>D･f(i)かつi<Mなら、i←i+1として②へ。

④入力バッファのi行目をセルBに送る。

⑤セルAを動作させる（データは一段流れる）。

⑥t<Nならt←t+1として①へ。 t=Nならセルを AをN回動作（データを出力）させて終了。

以上の動作は最大M+D+Nクロックで終了し、 D×Dに正規化された図形が得られる。

4．むすび

文字認識の前処理である正規化を並列に処理するＶＬＳＩ向きアルゴリズムを提案した。本アルゴリズムの特徴は、グローバルな処理である正規化を局所的な演算とデータの流れのみで実現し、しかも除算器の使用を避けた点にある。しかし、本アルゴリズムによって実際にハードウェアを作る場合には、克服しなければならない問題は多い。その中でも、ピン数及び回路規模の縮小が最大の課題であり、実用的には、並列性をある程度犠牲にしてハードウェア量を減らすことも必要になるであろう。

謝辞　本研究を進めるにあたり、数々のご指導ご助言を頂いた富士通株式会社の岩城博氏および勝山裕氏、富士通東北ディジタルテクノロジ株式会社の鈴木健司氏に感謝します。

参考文献

[1] 中山，吉田：パイプライン型画像処理アルゴリズムについて，総合研究(A)（課題番号 62302032）報告書，名古屋大学，昭和63年３月

[2] 中山，木村，吉田，福村：大規模画像に対する細線化アルゴリズムのパイプライン方式による効率化，電子情報通信学会論文誌(D), J67-D, 7, pp.761-767

[3] 浅見：印刷漢字ＯＣＲが市場に本格参入、一般オフィスの文書入力用に，日経エレクトロニクス，No.454，pp.179-184(1988)

[4] 孫，市村，木村：活字漢字の高速認識に関する研究，昭和63年電子情報通信学会春季全国大会講演論文集，D-456

[5] 大町，孫，阿曽，木村：印刷文字認識の並列処理に関する基礎研究，昭和63年電子情報通信学会春季全国大会講演論文集，D-455