

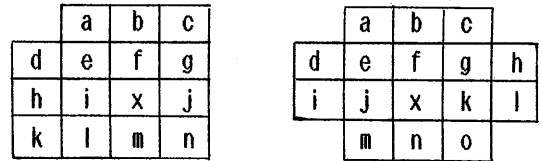
手書き文字認識前処理アルゴリズムと  
連想プロセッサによるその評価

7E-1

大金 顕二 後藤 厚宏 田中 英彦 元岡 達  
( 東京大学 工学部 )

1. はじめに

手書き文字認識の前処理には、平滑化、正規化、細線化などがある。細線化のアルゴリズムは従来数多く発表されているが、そのほとんどは逐次型のアルゴリズムである。しかし、今後、連想プロセッサや画像処理専用プロセッサを用いた高速処理を行なう場合は並列型のアルゴリズムが必要になってくる。本報告では、並列型の一細線化アルゴリズムと、連想プロセッサ DREAM-1<sup>0</sup>によるその評価を報告する。



( a ) 8 連結 ( b ) 4 連結

図1 参照する近傍  
(点 x の削除を判定する)

2. 細線化アルゴリズム

本アルゴリズムの原理は、連結数<sup>2)</sup>=1の点を削除して行くものであるが、並列型の困難な所は、複数の点が同時に削除され連結性が変化する事がある事である。そこで本方式では、近傍点の削除の影響を受けない条件を設定する事により、並列処理を可能にした。

本アルゴリズムの対象は2値画像であり“1”を図形、“0”を背景とする。図形に属する画素の近傍の状態から、削除条件を評価し、これを満たすとき“1”→“0”とする。この判定は全画面並列に行なう事ができ、変化する点がなくなるまで行なう。

並列細線化アルゴリズムとしては、Stefanelle & Rosenfeld (S&R)<sup>3)</sup>や田村<sup>4)</sup>のアルゴリズムが有名であるが、これらのアルゴリズムでは、1回の反復を4つのサブサイクルに分け、1方向からの削除を行なっているため、処理時間が長くなるという問題がある。また、S&R、田村のアルゴリズムでは、最終点条件を用いているので、画面の大きさのメモリが処理用と別にもう1面必要になり、メモリの効率がよくない。それに対し、本方式では1回の削除点判定で4方向からの削除が可能であり、メモリも1画面分でよい。

以下に、8連結の場合と、4連結の場合の2通りについて点削除の条件を示す。削除条件で参照する近傍を図1に示す。

( a ) 8 連結

$$C_{con} = \frac{\{\bar{f} \cdot \bar{m} \cdot (i \cdot j + \bar{e} \cdot \bar{g} \cdot \bar{l} \cdot \bar{n}) + \bar{l} \cdot \bar{j} \cdot (f \cdot m + \bar{e} \cdot \bar{g} \cdot \bar{l} \cdot \bar{n}) + \bar{f} \cdot (e \cdot \bar{l} + g \cdot \bar{j}) + \bar{m} \cdot (l \cdot \bar{l} + n \cdot \bar{j}) + f \cdot m \cdot i \cdot j\}}$$

$$C_{neb} = \{\bar{f} + \bar{e} \cdot (\bar{l} + a \cdot \bar{b}) + \bar{g} \cdot (\bar{j} + c \cdot \bar{b}) + b \cdot (e \cdot g + \bar{e} \cdot \bar{g})\} \cdot (\bar{l} + \bar{e} \cdot (\bar{f} + d \cdot \bar{h}) + \bar{l} \cdot (\bar{m} + k \cdot \bar{h}) + h \cdot (e \cdot l + \bar{e} \cdot \bar{l}))$$

$$C_{iso} = a + b + d + g + h + j + l + m$$

$$C_{dia} = a \cdot e \cdot f \cdot \bar{h} \cdot \bar{l}$$

$$C_{del} = C_{iso} \cdot C_{con} \cdot (C_{dia} + C_{neb})$$

( b ) 4 連結

$$C_{con} = \frac{\{(f \cdot n + \bar{f} \cdot \bar{n}) \cdot (j \cdot k + \bar{j} \cdot \bar{k}) + f \cdot (\bar{e} \cdot \bar{j} + \bar{g} \cdot k) + n \cdot (\bar{m} \cdot j + \bar{o} \cdot k) + \bar{e} \cdot \bar{g} \cdot \bar{m} \cdot \bar{o}\}}$$

$$C_{neb} = \{\bar{f} + \bar{j} + (\bar{n} + i) \cdot (\bar{k} + b)\} \cdot \{\bar{e} + \bar{f} + \bar{j} + d \cdot (a + \bar{l}) + \bar{b} \cdot (a + \bar{g}) + b \cdot g \cdot \bar{l} \cdot \bar{m}\} \cdot \{\bar{f} + \bar{g} + \bar{k} + h \cdot (c + \bar{l}) + \bar{b} \cdot (c + \bar{e}) + b \cdot e \cdot \bar{l} \cdot \bar{o}\}$$

$$C_{iso} = a + b + d + g + i + k + m + n$$

$$C_{dia} = b + n + (g + j + l) \cdot (e + i + k)$$

$$C_{lin} = c \cdot f \cdot k \cdot \bar{e} \cdot \bar{j}$$

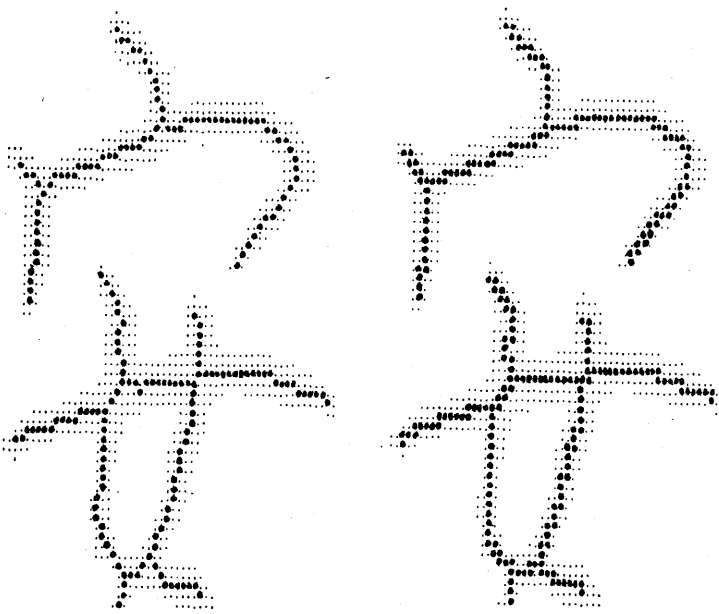
$$C_{del} = C_{iso} \cdot C_{dia} \cdot C_{con} \cdot (C_{lin} + C_{neb})$$

ここで

- ・ : 論理積
- + : 論理和
- $C_{con}$  :  $x$  が周辺点、連結数 = 1、 $x$  の 8 近傍の “1” の数が 2 以上である条件
- $C_{neb}$  : 近傍の変化を受けない条件
- $C_{iso}$  : 孤立点保存の条件
- $C_{dia}$  : 対角方向線保存の条件
- $C_{lin}$  : 縦方向線削除の条件
- $C_{del}$  : 点削除の条件

である。

結果の格納は、 $x \leftarrow x \cdot \overline{C_{del}}$  とし、変化判定は、 $x \cdot C_{del}$  とする。以上のアルゴリズムにより細線化した例を図 2 に示す。



(a) 8 連結 (b) 4 連結  
図 2 細線化の例

### 3. 連想プロセッサによる評価

連想プロセッサ DREAM-11 に、本アルゴリズムを実装し、64×64 の大きさの図形を処理したときの処理時間を表 1 に示す。処理は 1 行 64 ビットを並列に行なった。また、表 1 には、手書き漢字データ（電総研作成）を処理した時の平均反復回数と平均処理時間を示す。田村のアルゴリズムについても、同様の実験を行なった。

		反復一回の 処理時間	平均反復 回数	平均処理 時間
本方式	8 連結	3.04 ms	4.60 回	14.0 ms
〃	4 連結	3.32 ms	4.47 回	14.8 ms
田村	8 連結	5.88 ms	3.60 回	21.2 ms
〃	4 連結	5.40 ms	3.20 回	17.3 ms

表 1 細線化の処理時間

DREAM-11 は 4 モジュールからなる連想プロセッサで、1 マイクロ命令実行時間は 333 ns、ホスト計算機は FACOM U-300 である。DREAM-11 は、ホスト計算機間のデータの転送と、モジュールでの処理をオーバーラップすることが出来る。現在は、処理の約 1/10 が転送とオーバーラップしている。したがって、モジュールを現在の 10 倍の 40 台にすれば処理と転送の時間がマッチングし、1 図形 (64×64) 約 1.5 ms で処理出来ることになる。

### 4. おわりに

本報告では、並列型細線化の一手法を提案し、それが有用であることを示した。また、本アルゴリズムが連想プロセッサ DREAM-11 により高速処理が可能であることを示し、連想プロセッサの有効性を確認することが出来た。

#### <参考文献>

- 1) 後藤 他、”連想プロセッサ DREAM-11 の性能評価” 第 23 回情報学大会。
- 2) 横井 他、“標本化された二値図形のトポジカルな性質について” 信学論 (D) vol. 56-D, No. 11, pp. 662 - 669.
- 3) R. Stefanelli & A. Rosenfeld, “Some parallel thinning algorithms for digital pictures,” J. ACM., vol. 18, No. 2, pp255 - 264, 1971.
- 4) 田村、“細線化法についての諸考察” 信学会 PRL-75-66.