# 細線化アルゴリズムの高速化に関する考察†

鈴木 智††

本論文では，2値画像の基本処理手法の一つである細線化について，アルゴリズムの収束性に関する性質を明らかにし，その性質を用いた高速化法を提案する．まず，電子技術総合研究所で作成された画像処理サブルーチンパッケージ SPIDER に収録されている反復型細線化アルゴリズムについて，細線化画像において図形部分として残る画素を細線化処理の途中で検出できることを示す．これらの画素を収束点と呼ぶ．収束点を検出して処理対象からはずすことにより，細線化アルゴリズムを高速化できる．次に，各反復において収束点と背景画素のみからなる長方形小領域（部分画像と呼ぶ）を処理対象から外すことにより，さらに高速化が可能であることを示す．また，部分画像単位の収束判定が少数個の処理装置による並列処理に適することを示す．

## 1. まえがき

画像処理の分野では，新しい処理手法の開発とともに，既存の手法を体系的に整理して，頻繁に用いられる手法に対して有効なアルゴリズムを蓄積することも必要である．ハードウェアの発達によって，扱うべき画像は大規模になる傾向にあることから，各処理手法に対して高速アルゴリズムを開発することは，一つの重要な課題である．

画像処理の基本手法の一つに，文字認識，図面認識などに広く用いられている細線化がある．しかし，期待される処理結果が応用により異なること，処理時間が長いことなどから多くの細線化アルゴリズムが提案されている[1)~14)]．細線化の処理時間が長い原因は，良い品質の線図形を得るために背景から図形を1画素分ずつ細めているからである．この細め処理は図形の最大幅の半分だけの回数の画面走査を必要とする．そこで，背景から図形を細めずに中心線を求める高速細線化アルゴリズム[8)~10)]と一定の画面走査を用いる高速細線化アルゴリズム[11),12)]がいくつか提案されている．しかし，これらの方法の多くは得られる中心線の形状に問題があることと，期待される細線化の処理結果が応用により異なることから，多くの細線化アルゴリズムに適用できる高速化法が期待されている．このような高速化法として，背景画素への走査を省く方法[13)]，線順次処理により，注目行に含まれるすべての画素について値が変化しないならば，その行を演算対象から外す方法が提案されている[14)]．

本論文では，まず，電子技術総合研究所で作成された画像処理サブルーチンパッケージ SPIDER に収録されているまたは収録が予定されている反復型細線化アルゴリズム[15)]について，細線化画像において図形部分として残る画素を細線化処理の途中で検出できることを示す．これらの画素を収束点と呼ぶ．収束点を検出して処理対象から外すことにより，細線化アルゴリズムを高速化できる．

次に，処理する画像全体を長方形をした小領域（部分画像と呼ぶ）から成ると捉える．そして，各反復において収束点と背景画素のみからなる部分画像を処理対象から外すことにより，さらに高速化が可能であることを述べる．さらに，部分画像単位の収束判定が少数個の処理装置による並列処理に適することを述べる．

最後に，地図データを用いた実験結果を示し，収束点を用いることにより，反復型細線化アルゴリズムを逐次型の汎用計算機において最高2倍高速に実行できることを示す．また，部分画像の大きさを適当に設定することにより，$N$ 個の処理装置を用いて $N$ 倍に近い高速化の可能性があることを示す．

## 2. 収束点の導出

2値画像の細線化処理は，背景にある画素を 0-画素，図形上にある画素を 1-画素とした入力画像から，図形の中心線上にある 1-画素を抽出する処理手法である．そのアルゴリズムとして，形状が良い線図形が得られるものは背景と図形の境界にある 1-画素（境界 1-画素と呼ぶ）を1画素分ずつ細めるというものである[1)~7)]．

これらのアルゴリズムは画素の削り方から大きく2つに分けることができる．1つは，消去できる画素の

† A Consideration of Speedup for Thinning Algorithms by SATOSHI SUZUKI (NTT Electrical Communications Laboratories).

†† NTT 電気通信研究所

条件を設定して，それらの画素を除去するものである[1)~4)]．これを消去条件設定型と呼ぶ．もう1つは，中心線上にあるための条件を設定して，その条件を満足した画素を処理対象から外し，その条件を満足しない境界画素を除去するものがある[5)~7),12)]．これを中心線条件設定型と呼ぶ．

中心線条件設定型アルゴリズムでは，いったん中心線上にあるための条件を満足した画素は以降の処理対象から外されるため，処理時間を短縮できる．したがって，消去条件設定型アルゴリズムについても，消去できない境界 1-画素を中心線上にある画素として処理対象から外すことにより，後者のアルゴリズムのように高速化ができる．しかし，ほとんどの細線化アルゴリズムにおいて，消去できない境界 1-画素は必ずしも処理結果において 1-画素として残る画素ではない．すなわち，ある時点において消去できない画素がそれより後の時点において消去可能になることがある．強制的に消去できない境界 1-画素を中心線上にある画素として処理対象から外すと，結果が変化してしまうという問題が生じる．

本論文では，細線化アルゴリズムについて処理結果において 1-画素として残る画素の条件を考察する．これらの画素を収束点と呼ぶ．収束点を抽出して処理対象から外すことにより，処理結果を変えることなく高速化ができる．なお，収束点は，注目点を中心とした8近傍の画素値から判定できるものだけとする．

以下では，画像処理サブルーチンパッケージ SPIDER[15)] に収録されているまたは収録が予定されている Hilditch の方法[1)]，鶴岡の方法[2)]（4連結のみ），木本の方法[3)]，Deutsch の方法[4)]，Pavlidis の方法[6)]，田村の方法[7)]，鈴木の方法[12)] について考察する．

Pavlidis の方法，田村の方法，鈴木の方法は中心線条件設定型であるため，中心線上にある条件を満足した画素を収束点と考えればよい．したがって，これら以外の方法についてのみ考える．

まず，Hilditch の方法について収束点の導出の考え方を説明する．

[アルゴリズム1]（Hilditch アルゴリズム）

入力2値画像を TV ラスタによって繰り返し走査しながら次の条件をすべて満足する 1-画素を除去する．そして，1回の画面走査において除去される画素がなくなったとき処理を終了する．

[各回の画面走査を開始したときの画像の状態（並列状態と呼ぶ）における条件]

（1）4近傍に 0-画素が存在する．

（2）8近傍に2つ以上の 1-画素が存在する．

（3）8連結数[5)] が1である．

[注目画素を走査したときの画像の状態（逐次状態と呼ぶ）に対する条件]

（4）8近傍に 1-画素が存在する．

（5）既走査の4近傍画素の除去により，8連結数が変化しない．

まず，端点と孤立点を保存するための消去条件に着目する．境界 1-画素の中で，この条件によって消去されない画素が収束点になることは明らかである．Hilditch の方法の場合，条件(2)と(4)が端点と孤立点を保存するための条件に対応する．したがって，並列状態において次の条件(6)を満たす画素と，逐次状態において次の条件(7)を満たす画素は収束点になる．

（6）8近傍に存在する 1-画素の個数は1以下である．

（7）8近傍に存在する 1-画素の個数は0である．

次に，残りの消去されない境界 1-画素から，さらに収束点を求める．具体的には，まず，8近傍にある1-画素を，注目画素が消去されないかぎり，消去条件のいずれか1つを満たさない画素（Aタイプと呼ぶ）とそれ以外の画素（Bタイプと呼ぶ）に分ける．次に，Bタイプの 1-画素について，画素単位の消去・保存をすべての場合について仮定し，注目点が消去条件を満足するか調べる．もし，消去される場合がないならば収束点である．

Hilditch の方法では，並列状態の条件と逐次状態の条件により消去できる画素を決定しているため，並列状態の条件を満足しない境界 1-画素のなかから収束点を導くことと，並列状態の条件を満足するが逐次状態の条件を満足しない境界 1-画素のなかから収束点を導くことができる．

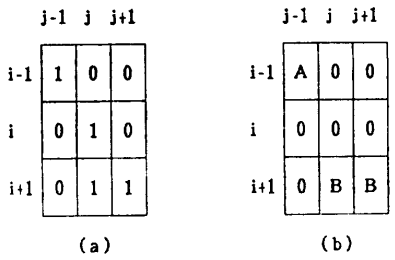


図1 8近傍画素の分類
Fig. 1 Classification of 1-pixels in the 8-neighborhood.

並列状態の条件を満足しない境界 1-画素のなかから収束点を導く例を**図1**に示す．図1(a)に示す 3×3 パターンを持つ画素は，上記の条件(3)を満足しないため消去されない．8近傍にある 1-画素をAタイプとBタイプに分けると，画素 $(i-1, j-1)$ がAタイプ，画素 $(i+1, j)$ と $(i+1, j+1)$ がBタイプとなる．すなわち，画素 $(i-1, j-1)$ は画素 $(i, j)$ が消去されないかぎり消去条件(1)～(3)のすべてを満たすことはない．理由は，消去条件(3)を満足するためには，画素 $(i-1, j-1)$ の8近傍には画素 $(i, j)$ 以外の 1-画素は存在できなく，そのとき，画素 $(i-1, j-1)$ は条件(2)を満足しないことになるからである．次に，Bタイプの画素の消去・保存を仮定し，消去条件(1)～(3)のすべてを満たす場合があるか調べる．この例では，3つの場合が考えられる．すなわち，①画素 $(i+1, j)$ を消去して画素 $(i+1, j+1)$ を保存した場合，②画素 $(i+1, j)$ を保存して画素 $(i+1, j+1)$ を消去した場合，③画素 $(i+1, j)$ と画素 $(i+1, j+1)$ をともに消去した場合の3つである．①および②の場合は連結数が 2 になるため，注目画素 $(i, j)$ は消去条件(3)を満足しない．③の場合，注目画素 $(i, j)$ は消去条件(2)を満足しない．したがって，画素 $(i, j)$ は消去されることがないことがわかる．これより，図1(a)に示す3×3パターンを持つ画素は収束点になる．同様にして議論を進めると，並列状態において，次の条件(8)を満足する画素は収束点になる．

(8) **図2**に示すパターンのいずれか1つを満たす．

次に，逐次状態について考える．逐次状態において既走査である画素と 0-画素は，次の画面走査を開始するときの並列状態に一致する．したがって，注目画素の8近傍を考えると，既走査の近傍画素が消去条件(1)～(3)のいずれか1つを満足しないならば，その近傍画素は消去されない．一方，未走査の8近傍画素


**図2** 条件(8)のパターン（下線は注目画素を表す．$X$ は少なくとも1つは 1-画素を含む画素集合を表す．条件(8)は 90° ずつ回転したパターンも含む．）

Fig. 2 Patterns of condition (8). (The pixels under consideration are underlined. $X$'s mean that at least one pixel must be 1-pixel. Condition (8) also includes the patterns obtained by rotations of multiples of 90 degrees.)

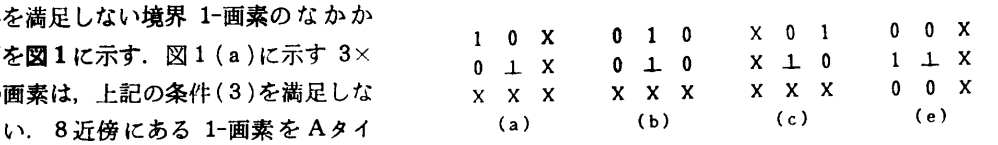


**図3** 条件(9)のパターン（下線は注目画素を表す．$X$ は少なくとも1つは 1-画素を含む画素集合を表す．）

Fig. 3 Patterns of condition (9). (The pixels under consideration are underlined. $X$'s mean that at least one pixel must be 1-pixel.)

について，消去されないものは8近傍の画素値からは判定できない．この点に留意して，並列状態のときと同様にすると，並列状態の消去条件を満足しかつ逐次状態の消去条件を満足しない画素のなかで，次の条件を満足するものは収束点であることがわかる．

(9) **図3**に示すパターンのいずれか1つを満たす．

木本の方法は，Hilditch の方法の交差部分のひずみを改良したものである．上記の方法と同様にして考察した結果，木本の方法により付加された条件により，注目画素の8近傍の状態より判定できる収束点はオリジナルな方法より増えないことがわかった．

Deutsch の方法では，次の条件(10)または(11)のいずれかを満たす画素が収束点となることを Hilditch の方法と同様にして導ける．

(10) 8近傍に 1-画素が1つだけ存在する．

(11) **図4**に示すパターンのいずれか1つを満たす．

4連結の鶴岡の方法では，次の条件(12)，(13)のいずれかを満足する画素が収束点となることを導ける．

(12) 8近傍に 1-画素が2つ以下だけ存在し，かつ鶴岡の方法の除去判定マスクを満足しない．

(13) 2回目以降の反復において，**図5**に示すパターンのいずれか1つを満たす．

鶴岡の方法では，得られる中心線の形状を良くする

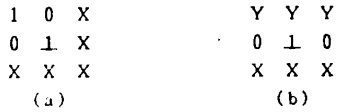


**図4** 条件(11)のパターン（下線は注目画素を表す．$X$ は少なくとも1つは 1-画素を含む画素集合を表す．$Y$ も同様である．条件(11)は 90° ずつ回転したパターンも含む．）

Fig. 4 Patterns of condition (11). (The pixels under consideration are underlined. $X$'s and $Y$'s mean that at least one pixel must be 1-pixel. Condition (11) also includes the patterns obtained by rotations of multiples of 90 degrees.)

```
d 1 d   0 d 0   0 0 d   d d d   0 1 0
0 ⊥ 0   d ⊥ d   1 ⊥ d   0 ⊥ d   0 ⊥ d
d 1 d   0 d 0   0 d d   0 1 0   d d d
  (a)     (b)     (c)     (d)     (e)

d 0 0   0 1 0   0 1 0   0 1 d   0 1 d
d ⊥ 1   1 ⊥ 0   1 ⊥ 1   1 ⊥ 0   1 ⊥ 0
d d 0   0 0 d   d 0 d   0 1 d   d 0 0
  (f)     (g)     (h)     (i)     (j)
```

図 5 条件(13)のパターン(下線は注目画素を表す．$d$は1または0のいずれの値もとれる画素を表す．条件(13)はパターン(a)および(b)を90°ずつ回転したパターンも含む．)

Fig. 5 Patterns of condition (13). (The pixels under consideration are underlined. $d$'s represent don't care condition. Condition (13) also includes the patterns obtained from patterns (a) and (b) by the rotations of multiples of 90 degrees.)

ため端点の保存条件に異方性があり複雑になっている．そのため，収束点も複雑になる．また，条件(13)において，「2回目以降の反復において」という条件がついているのは，鶴岡の方法では1回目の反復では端点を縮退するように端点保存条件が設定されているため，8近傍の状態から収束点と判定できる点が少ないことから，これを避けて収束点の条件を簡単にするためである．

## 3. 収束点を用いた高速化法

### 3.1 部分画像単位の収束判定

収束点を用いた高速化の簡単なものは，収束点と判定された画素にマークを付けて，以後の処理対象から外すというものである．これでは，各画素の値を一度は参照しなければならない．そこで，以下の方法により画素の値の参照の回数も減らす．

処理する画像全体を長方形をした小領域（部分画像と呼ぶ）から成ると捉える．そして，各反復において，各部分画像単位に含まれる画素がすべて収束点または0-画素になったかを調べる．そして，この条件を満足した部分画像を以降の反復の演算の対象から外す．各部分画像に対してその部分画像を処理するか否かを示すフラッグを設け，初期値を“真”にしておく．そして，フラッグが“真”である部分画像のみに1-画素の消去処理をほどこし収束点でない1-画素が存在しなかった部分画像のフラッグを“偽”にする．

本方法が細線化を高速にするか否かは，本方法により付加される演算量と，省略される演算量によって決まる．したがって，付加される演算をできるだけ簡単にするため，部分画像の形状を画面走査方法に適した形，すなわち，画面走査によって部分画像が順に現れるようにする．具体的には，TV ラスタ走査において連続的に現れる $n$ 個（$n$=[行の長さ]/[正整数]）の画素の集合を1つの部分画像と設定する．これにより，各画素がどの部分画像に属するかを簡単に決定できる．

部分画像の大きさは，画像の大きさと内容に依存させて変化させなければならない．部分画像の大きさは以下のようにしておおまかに決めることができる．画像の大きさを $M\times N$ $(M>N)$，部分画像の大きさを $n$，$k$ 回目の画面走査において0-画素または収束点からなる部分画像の個数を $N_k$ と表すことにする．部分画像を設定することにより1回の画面走査の間に減少できる演算は収束した部分画像に含まれる画素への参照である．一方，増加する演算は，全部分画像のフラッグの参照，収束点になっていない1-画素を含む部分画像のフラッグへの書き込み，フラッグを決定するための演算等である．フラッグを決定する演算等を無視すると，残りは2次元配列の要素の参照と書き込みになる．参照と書き込みを同等の演算量として扱うと，本方法により1回の画面走査の間に減少できる演算は次式により表せる．

$$N_k n - MN/n - (MN/n - N_k) \qquad (1)$$

ここで，第1項は減少する画素への参照回数，第2項は全部分画像のフラッグの参照回数，第3項は収束点になっていない1-画素を含む部分画像へのフラッグへの書き込み回数である．これを整理すると次式になる．

$$N_k(n+1) - 2\,MN/n \qquad (2)$$

$N_k$ は画面走査1回ごとに単調に増加し，実際に細線化を実行しないと正確には求められない．そこで，$N_k$ の初期値である0-画素からなる部分画像の個数 $N_0$ $(N_0 \leqq N_k)$ を用いた次式を最大にする部分画像の大きさ $n_0$ と表すと，$n_0$ より大きな値に部分画像の大きさ $n$ を設定すればよい．

$$N_0(n+1) - 2\,MN/n \qquad (3)$$

後述する実験から，$n_0$ より大きい $n$ のなかで式(3)の値を正にするものを選択すれば，最適値に近い高速化ができることがわかる．

### 3.2 部分画像単位の収束判定を利用した並列処理

画像処理では高速化のためにマルチプロセッサシステムが利用される場合が多い．例えば，全画素数分だけのプロセッサを各画素に割り当て並列に動作させるもの，処理を幾つかの段階に分け，各段階に1つのプ

ロセッサを割り当て並列に動作させるものなどが提案されている．近年では，扱うべき画像は大規模になっていることから，少数個のプロセッサによる並列処理を行うことが多い．部分画像単位の収束判定を行う細線化方式は，このようなマルチプロセッサシステムに適することを述べる．本論文では，1台の制御用プロセッサと $m$ 台の並列に動作するプロセッサからなり，各並列プロセッサは共通メモリから必要なデータをローカルメモリに転送して処理を行うようなシステムを想定する．

まず，1回の画面走査において各画素の値を同時に計算することが可能であるアルゴリズムの場合を考える．例えば，Pavlidis の方法，Deutsch の方法，田村の方法がこの型である．収束してない部分画像を1つずつ各並列プロセッサに割り当て処理を実行する．各並列プロセッサは割り当てられた部分画像とそれに隣接する部分画像を自分のローカルメモリに転送して処理を行う．部分画像単位の収束判定を行うことにより，原アルゴリズムおよび画素単位の収束判定を導入したアルゴリズムに比べて共通メモリとローカルメモリ間の転送を減少させることができる．理由は，収束した部分画像は処理対象からはずされることと，処理する領域がある程度の大きさの長方形になることである．

次に，1回の画面走査において各画素の値を一定の走査順序により1つずつ計算しなければならないアルゴリズムの場合を考える．例えば，Hilditch の方法，鶴岡の方法，木本の方法がこの型である．この型のアルゴリズムは，並列処理に不向きであるが，部分画像単位の収束判定を利用することにより以下のようにして1回の画面走査の処理に並列処理を導入できる．

制御プロセッサは，収束していない部分画像のなかから，処理可能なものを1つずつ各並列プロセッサに割り当て処理を実行する．処理可能な部分画像は次のようにして決めることができる．すなわち，注目部分画像に隣接している部分画像のうち，画面走査において注目部分画像より先に走査されるものはすべて次の条件(1)，(2)のいずれかを満足しているならば，注目部分画像は処理可能であるとする．

(1) 収束している．

(2) 今回の画面走査における計算が終了している．

例えば，**図6**は，1行ずつ上から下へ，同一行では左から右に部分画像を1つずつ順次走査するラスタ走査を用いて，$I$ 行の左から $J$ 番目の部分画像に到達した時の状態を表すとする．この場合，$I$ 行の $J-1$ 番目，$I-1$ 行の $J-1$ 番目，$J$ 番目，$J+1$ 番目に位置する4つの部分画像について今回の画面走査における計算が終了しているため，$I$ 行の $J$ 番目の部分画像は処理可能である．次に，$I$ 行の $J+1$ 番目の部分画像に着目すると，$I-1$ 行の $J$ 番目，$J+1$ 番目，$J+2$ 番目に位置する部分画像について今回の画面走査における計算が終了しているが，$I$ 行の $J$ 番目の部分画像について計算が終了していないため，$I$ 行の $J+1$ 番目の部分画像は処理可能でない．次に，$I$ 行の $J+3$ 番目の部分画像に着目すると，$I-1$ 行の $J+2$ 番目，$J+3$ 番目，$J+4$ 番目に位置する部分画像について今回の画面走査における計算が終了し，$I$ 行の $J+2$ 番目の部分画像は収束しているので，$I$ 行の $J+3$ 番目の部分画像は処理可能となる．したがって，$I$ 行の $J$ 番目と $I$ 行の $J+3$ 番目に位置する2つの画像は並列に処理可能となる．

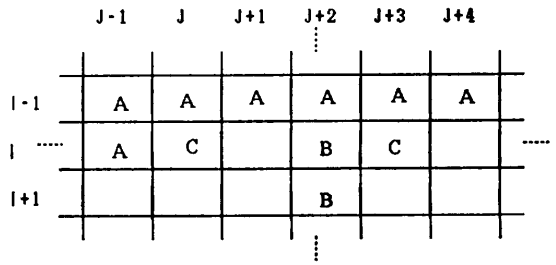


**図 6** 並列に処理される部分画像（Aは今回の画面走査において処理済みの部分画像を表す．Bは収束している部分画像を表す．Cは並列に処理可能な部分画像を表す．）

Fig. 6 Subpictures processed in parallel. (A's denote subpictures which have been processed. B's denote subpictures which have reached an invariant state. C's denote subpictures which can be processed in parallel.)

各並列プロセッサは割り当てられた部分画像とそれに隣接する部分画像をローカルメモリに転送して処理を行う．細線化処理では，収束している部分画像は単調に増加するため，この方式により待ち状態になるプロセッサを少なくして並列処理を行える．ただし，制御プロセッサは，処理可能でない部分画像を検出したとき，その部分画像の位置を記憶して，次に並列プロセッサに部分画像を割り当てるとき，再度，実行可能でなかった部分画像が実行可能か検査して，可能ならば並列プロセッサに割り当てるという管理をしなければならない．そのため，その分のオーバヘッドが増える．

## 4. 実　　験

汎用の逐次型電子計算機によるシミュレーション実験の結果を示す．入力2値画像として，**図7**に示す，スキャナより入力した国土基本図（縮尺 1/2500，大きさ 800 画素×800 画素）を用いた．また，画像全体を主記憶に置き処理時間を測定した．

［実験1］（収束点の効果）

**表1**は，Hilditch の方法，Deutsch の方法，鶴岡の方法に画素単位の収束判定を導入したアルゴリズムについて，処理時間を測定した結果である．表1より，収束点の導入により Hilditch の方法，Deutsch の方法，鶴岡の方法はそれぞれ 34%，35%，21%，処理時間が短縮されたことがわかる．実験に用いた画像は1画素から 10 画素の幅が一定した図形から構成されている．したがって，複数の線幅の線から構成されている画像に対して，収束点は効果的であることがわかる．

［実験2］（部分画像単位処理の効果）

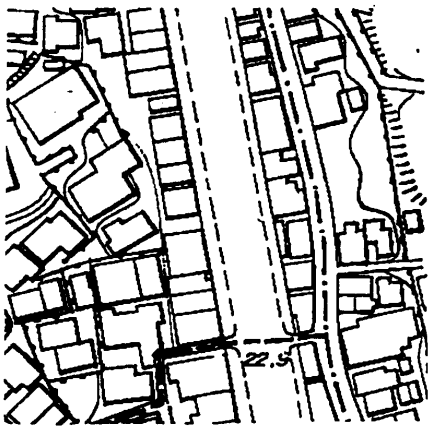

**図7** 入力2値画像の一部
Fig. 7 Part of input digital binary picture.

**表1** 収束点を用いたアルゴリズムの原アルゴリズムに対する処理時間† の比（† VAX11-780 により測定した.）
Table 1 Computation time† ratio of the algorithms skipping final points to the original algorithms. († measured by a sequential computer VAX11-780.)

| 方　法 | 収束点を用いたアルゴリズムの処理時間の比 |
|---|---|
| Hilditch の方法 | 0.660 |
| Deutsch の方法 | 0.649 |
| 鶴岡の方法（4連結） | 0.794 |

**表2** 部分画像単位の収束判定を用いたアルゴリズムの収束点を用いたアルゴリズムに対する処理時間† の比（† VAX11-780 により測定した.）
Table 2 Computation time† ratio of algorithms skipping invariant subpictures to algorithms skipping final points. († measured by a sequential computer VAX 11-780.)

| 部分画像の大きさ（画素） | Hilditch の方法 | Deutsch の方法 | 鶴岡の方法 | 田村の方法 |
|---|---|---|---|---|
| 800 | 0.574 | 0.635 | 0.801 | 0.765 |
| 100 | 0.542 | 0.507 | 0.779 | 0.682 |
| 50 | 0.542 | 0.486 | 0.735 | 0.665 |
| 20 | 0.553 | 0.473 | 0.699 | 0.659 |
| 16 | 0.553 | 0.480 | 0.699 | 0.652 |
| 10 | 0.596 | 0.493 | 0.684 | 0.682 |

**表2**は，Hilditch の方法，Deutsch の方法，鶴岡の方法，田村の方法* に部分画像単位の収束判定を導入したアルゴリズムについて，処理時間を測定した結果である．表2より，部分画像の導入により Hilditch の方法，Deutsch の方法，鶴岡の方法，田村の方法は原アルゴリズムよりさらにそれぞれ最大 11%，18%，11%，35% の処理時間が短縮されたことがわかる．すなわち，Hilditch の方法，Deutsch の方法，鶴岡の方法，田村の方法は原アルゴリズムのそれぞれ最短 55%，47%，68%，65% の処理時間で実行できる．3.1 節の(3)式を実験に用いた画像について計算すると，部分画像の大きさの下限は5であり，32以下の大きさにおいて(3)式は正の値を示した．表2より，この範囲にある 10，16，20 の部分画像の大きさでは，ほぼ最短処理時間に近い結果を得られることがわかる．なお，田村の方法** を除いた上記の方法について，収束点を検出せず，消去された画素も含めて 0-画素からのみなる部分画像を処理対象からはずす処理のみを導入した実験も行った．その結果は，いずれの方法についても原アルゴリズムの処理時間にほぼ等しく，部分画像単位の収束判定の導入のみでは処理を高速化できなかった．したがって，収束点検出が高速化に寄与していることがわかる．

［実験3］（並列処理の可能性）

3.2 節において述べた並列処理の効果を確かめるため，並列プロセッサの個数を 4，6，8，10 として，Hilditch のアルゴリズムについて，並列プロセッサの個数ずつ並列処理可能な部分画像を処理して並列処

* 中心線上にあるための条件を満足した画素が収束点に対応するため，それを用いて部分画像単位の収束判定を導入した．
** 田村の方法では中心点を検出することが収束点を検出していることになるため，収束点検出は必ず行われる．

表 3 少数個の処理装置による並列処理において同時に処理できる部分画像の割合†（† 処理された部分画像の延べ数に対する処理装置をすべて使って同時に処理できた部分画像の延べ数の百分率である.）

Table 3 Percentage† of subpictures processed in parallel by using a few processors. († ratio of the total number of subpictures processed in parallel by using the all processors to the total number of subpictures processed.)

| 部分画像の大きさ（画素） | 処理装置の個数 | | | |
|---|---|---|---|---|
| | 4 | 6 | 8 | 10 |
| 800 | 27.5% | 24.1% | 18.9% | 17.6% |
| 100 | 91.6% | 20.1% | 18.7% | 17.7% |
| 50 | 99.3% | 95.6% | 85.6% | 25.4% |
| 20 | 99.9% | 99.7% | 99.6% | 98.5% |
| 16 | 99.9% | 99.7% | 99.7% | 99.5% |
| 10 | 99.9% | 99.7% | 99.8% | 99.7% |

理をシミュレートした．その結果として，すべての並列プロセッサが動作した並列処理により処理された部分画像の延べ数の，処理された部分画像の延べ数に対する割合を**表 3** に示す．これより，並列プロセッサの個数が 4, 6, 8, 10 の場合，それぞれ部分画像の大きさ 100 以下，50 以下，20 以下，20 以下において効率的な並列処理を行える可能性があることがわかる．3.2 節で示した並列処理によって，どの程度高速化が図れるかは，共通メモリからローカルメモリへの転送時間等を考慮する必要があるため，実際のハードウェアによる測定が必要である．

## 5. むすび

本論文では，2値画像の細線化手法について，細線化画像において図形部分として残る画素（収束点）を細線化処理の途中で検出できることを示した．また，収束点を処理対象から外すことと，収束点と 0-画素からなる部分画像を処理対象から外すことにより，最高 2 倍程度の高速化が可能であることを実験により示した．さらに，部分画像単位の収束性の判定が，少数個のプロセッサからなる並列処理に有効であることをシミュレーション実験により示した．

今後の課題として，濃淡画像の細線化アルゴリズムの収束点の導出がある．

**謝辞** 日頃，御指導を頂く小杉信主幹研究員，山田豊通主幹研究員，金子透主任研究員に感謝する．実験には電子技術総合研究所で作成された画像処理サブルーチンパッケージ SPIDER を利用した．

## 参考文献

1) Hilditch, C. J.: Linear Skeletons from Square Cupboards, in Meltzer, B. and Michie, D. eds., *Machine Intelligence*, pp. 403-420, University Press, Edinburgh (1969).
2) 鶴岡，木村，吉村，横井，三宅：ディジタル図形の一細線化法と手書き文字認識システムへの応用，電子通信学会論文誌，Vol. J66-D, No. 5, pp. 525-532 (1983).
3) 木本，安田：分岐歪みの少ない細線化，第12回画像工学コンファレンス報告集，pp. 67-70 (1981).
4) Deutsch, E. S.: Thinning Algorithms on Rectangular, Hexagonal, and Triangular Arrays, *Comm. ACM*, Vol. 15, No. 9, pp. 827-837 (1972).
5) 横井，鳥脇，福村：標本化された2値図形のトポロジカルな性質について，電子通信学会論文誌，Vol. 56-D, No. 11, pp. 662-669 (1973).
6) Pavlidis, T.: A Thinning Algorithm for Discrete Binary Images, *Comput. Graphics and Image Process.*, Vol. 13, No. 2, pp. 142-157 (1980).
7) Tamura, H.: A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint, *Proc. 4th Int. Joint Conf. on Pattern Recognition*, pp. 715-719 (1978).
8) Fischler, M. A. and Barrett, P.: An Iconic Transformation for Sketch Completion and Shape Abstraction, *Comput. Graphics and Image Process.*, Vol. 13, No. 4, pp. 334-360 (1980).
9) 岡田，小倉，村上：水平・垂直線要素の分類を用いた線順次型細線化法，電子通信学会論文誌，Vol. J64-D, No. 5, pp. 403-410 (1981).
10) Takahashi, K., Amanuma, H. and Takefushi, H.: A Fast Thinning Method by the Vector Peripheral Crossing Number, *IEE Conf. Publ.*, Vol. 265, pp. 117-121 (1986).
11) Arcelli, C. and Baja, G. S.: A Width-independent Fast Thinning Algorithm, *IEEE Trans.*, Vol. PAMI-7, No. 4, pp. 463-474 (1985).
12) 鈴木，阿部：距離情報を利用した逐次型細線化アルゴリズム，電子通信学会論文誌，Vol. J68-D, No. 4, pp. 473-480 (1985).
13) 塩野，蔭山，真田，手塚：平滑度可変の細線化アルゴリズム，電子通信学会研究会資料，IE 78-42 (1978).
14) 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，電子通信学会論文誌，Vol. J67-D, No. 7, pp. 761-767 (1984).
15) 阿部，小川：画像処理アルゴリズムの最新動向 6—幾何学的特徴の処理（2）—，*O plus E*, Vol.

72, No. 11, pp. 79-93 (1985).

(昭和62年7月29日受付)
(昭和63年9月 5 日受付)

**鈴木 智 (正会員)**

昭和31年生. 昭和54年静岡大学工学部情報工学科卒業. 昭和56年同大学院修士課程修了. 昭和59年同大学院博士課程修了. 工学博士. 同年, 日本電信電話公社に入社. 以来, 線図形認識, 地図データベース構築システムの研究開発に従事. 現在, NTT ヒューマンインタフェース研究所視覚情報研究部主任研究員. 画像認識・理解, 画像処理アルゴリズム, 並列処理に興味をもつ. 電子情報通信学会, IEEE 各会員.