| | |
|---|---|
| タイトル | 技術報告２値画像の細線化における連結数の役割の再検討 |
| 著者 | 高井，信勝; TAKAI，Nobukatsu |
| 引用 | 北海学園大学工学部研究報告(38)：155-172 |
| 発行日 | 2011-02-14 |

# 技術報告

# 2値画像の細線化における連結数の役割の再検討

髙　井　信　勝*

# Reexamination of Thinning Algorithm Based on the Number of Connection

**Nobukatsu TAKAI***

**Abstract**

Although there are many techniques on a thinning algorithm, most is based on the Hilditch algorithm known for many years. With this algorithm, the number of connection derived from 8 neighborhood pixels of the 3x3 mask which scans a binary image plays a role of importance, and the deletion condition of a current pixel (central pixel of a mask) is decided based on that value. However, the literatures described vaguely are scattered and this deletion condition is variously used by users. It may be stemmed from the fact that the relation between the number of connection and a current pixel is not used in a correct form.

In this report, by investigating the number of 8 neighborhood connection about all the cases of 8 neighborhood pixels, mask patterns are conversely arranged on the basis of the result of the number of connection. By reconfirming the role of a current pixel for various values of the number of connection, the delete condition of a current pixel in thinning algorithm has been refined. At that time, the MATLAB program of thinning algorithm under the condition which gives the current pixel deletion is presented.

## 1．はじめに

現実に画像として得られた文字，指紋，設計図面，地図などは白と黒からなる2値画像として扱うことで十分であり，同時に，描かれている線図の線幅情報は意味をなさない場合が多い．細線化とは，不要な線幅を削除して，芯線と呼ばれる文字や図形の骨格部分だけを取り出す処理である．つまり，2値画像の細線化は，計算機による図形解析を単純化するために，線

* 北海学園大学工学部電子情報工学科
* Department of Electronics and Information Engineering, Faculty of Engineering, Hokkai−Gakuen University

の連結性を保ちながら線幅を１，つまり線幅が１画素の線図形を作成するディジタル画像処理である．

細線化アルゴリズムには，幾つもの手法がある[1)−4)]が，多くは古くから知られているHilditchのアルゴリズム[5)]に基づいている．大づかみに言うと，このアルゴリズムでは，２値画像上を走査する３×３のマスクの８近傍画素から連結数を求め，その値に基づいて注目画素（マスクの中央位置の画素）の削除の可否を決めていく．ところがこの削除条件が曖昧に記述されている文献が散在し，煩雑で不完全な条件下で使われていることも見受けられる．なぜこのような混乱が生じているかは，判然としないが，その原因のひとつは連結数と注目画素の関係が明瞭に把握されていないことにあるように思われる．

本稿では，走査マスクの８近傍画素に関して考えられるすべての場合について連結数（８近傍連結数）を求め，その結果に基づいて逆に連結数ごとに８近傍のマスクパターンを整理した．そして，これにより連結数の違いによる注目画素の役割を明らかにし，それに基づいて細線化の注目画素削除条件を再確認した．同時に，この削除条件を用いた細線化のMATLABプログラムを提供する．

## ２． ２値画像における近傍処理

画素の並びからなる画像は，２次元配列の数値データ（行列）として扱われる．画像の細線化の対象になるのは，１と０からなる２値画像である．いま，処理する画像上のどこかに画像からはみ出さないように３×３のサイズのマスク（行列）をおき，その９個の要素値として，その位置での画像の画素値を入れる．マスクの中央の画素，同時に対応する画像の画素を注目画素とよぶ．注目画素の値を目的に応じてマスク全体の９要素あるいはその一部を用いて決める処理が近傍処理である．ただし，ここでは２値画像を扱うので，すべての要素値は１か０の値であり，論理演算を用いて処理がなされる場合が多い．

むろん一つの注目画素だけを新たな値に置き換えただけでは画像処理にならない．マスクを画像の行ごとに順に走査し，近傍処理を繰り返すことで全面の処理を行う．この全面走査はテレビ映像のフレーム走査に倣ってラスタースキャンとよばれる．ラスタースキャンで注意しなければならないことは，走査マスクが画像からはみ出さないように走査することである．したがって，３×３の走査マスクでは，画像の外周部分の画素には処理は適用できない．この部分は未処理のままにしておくか，０とか１とかの都合のよい値を埋め込むことが通例なされる．

３×３の走査マスクを用いた場合，中央の注目画素を除いた周りの８個の画素を８近傍画素という．近傍処理では，このマスクを走査して位置$x_0$の注目画素の値$f(x_0)$を周囲の８画素の値から決める処理である（全９要素を用いる処理もある）．これを説明するとき，習慣的に，８近傍画素の位置は，**図１**に示すように反時計方向周りに$[x_1, x_2, \cdots, x_8]$と表し，おのお

| | | |
|---|---|---|
| $x_4$ | $x_3$ | $x_2$ |
| $x_5$ | $x_0$ | $x_1$ |
| $x_6$ | $x_7$ | $x_8$ |

**図1**　走査マスクの要素の位置番号．注目画素 $x_0$ と8近傍画素 $x_1 \sim x_8$．

のの位置における値を $f_k = f(x_k)$ として $[f_1, f_2, \cdots, f_8]$ と表す．

8近傍画素のうち注目画素に隣接する上下2個と左右2個の4つの画素を4近傍画素という．白あるいは黒の画素のつながりを「連結」というが，**図2**（左）は4近傍画素の連結で画素Aから画素Bまで，白い画素が連結している．しかし，8近傍画素の連結でみると，同じ連結は**図2**（右）のようになる．このように画素の連結は，4近傍画素で考えるか，8近傍画素で考えるかで異なっていているが，境界画素の検出や細線化では，後者の8近傍連結が通常用いられる．なお，連結している画素のひとかたまりは連結成分と呼ばれる．

以下では，細線化の対象とする連結成分の値を1，他方を0とし，図の表示においては，図2のように，1を白，0を黒で表示する．

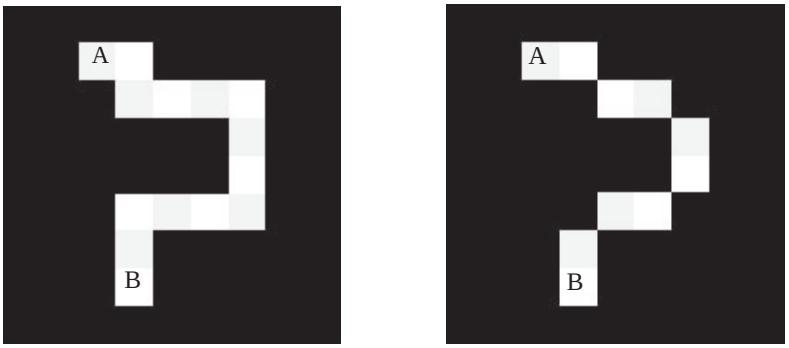


**図2**　画素Aから画素Bまでの4近傍連結（左）と8近傍連結（右）．黒い部分が0，白い部分が1．

## 3．連結数の定義

以下で用いられる「連結数」は，引き続く境界線検出や細線化処理で重要な役割を果たす．それは，連結数によって注目画素と近傍画素の連結関係が異なるからである．連結数は4近傍画素に関する連結と8近傍画素に関する連結のそれぞれに対して定義される．ここでは，連結数の具体的な意味は後述するとして，おのおのは次のように定義される．

［4近傍連結数］ $$N_C^4 = \sum_{k \in S} \{f_k - f_k f_{k+1} f_{k+2}\} \tag{1}$$

［8近傍連結数］ $$N_C^8 = \sum_{k \in S} \{\bar{f}_k - \bar{f}_k \bar{f}_{k+1} \bar{f}_{k+2}\} \tag{2}$$

ここで，$f_k$は$x_k$の位置（図1）の画素値（1あるいは0）であり，$\bar{f}_k = 1 - f_k$，$x_9 = x_1$である．また，総和は$S = \{1, 3, 5, 7\}$に関してとられる．

式(1)の4近傍連結数$N_C^4$は，和を開いて表すと

$$N_C^4 = f_1(1 - f_2 f_3) + f_3(1 - f_4 f_5) + f_5(1 - f_6 f_7) + f_7(1 - f_8 f_1) \tag{3}$$

とかける．ここで，$f_k$は1か0であるので，右辺の4項の各項の値はどれも1か0である．したがって，$N_C^4$の最小値は0，最大値は4であり，一般的には，$N_C^4 = 0 \sim 4$の整数値であることがわかる．たとえば，

- 8近傍のすべてが0，あるいはすべてが1の場合 ⇒ $N_C^4 = 0$
- 辺に位置する4つの画素がすべてゼロ（つまり$f_1 = f_3 = f_5 = f_7 = 0$） ⇒ $N_C^4 = 0$
- 辺の画素がすべて1（つまり$f_1 = f_3 = f_5 = f_7 = 1$），かつ4隅に位置する画素がすべてゼロ（つまり$f_2 = f_4 = f_6 = f_8 = 0$） ⇒ $N_C^4 = 4$

である．

同様に，式(2)の8近傍連結数を展開して表すと，

$$\begin{aligned} N_C^8 = {} & (1 - f_1)(f_2 + f_3 - f_2 f_3) + (1 - f_3)(f_4 + f_5 - f_4 f_5) \\ & + (1 - f_5)(f_6 + f_7 - f_6 f_7) + (1 - f_7)(f_8 + f_1 - f_8 f_1) \end{aligned} \tag{4}$$

と4項の和で表される．そして，この結果も各項は1か0であり，$N_C^8$もまた$N_C^8 = 0 \sim 4$の整数値であることがわかる．たとえば，つぎの場合は$N_C^8$の値を直ちに知ることができる．

- 8近傍のすべてが0，あるいはすべてが1の場合 ⇒ $N_C^8 = 0$
- 辺に位置する4つの画素がすべて1（つまり$f_1 = f_3 = f_5 = f_7 = 1$）⇒ $N_C^8 = 0$
- 辺の画素がすべてゼロ（つまり$f_1 = f_3 = f_5 = f_7 = 0$），かつ4隅に位置する画素がすべて1（つまり$f_2 = f_4 = f_6 = f_8 = 1$）の場合 ⇒ $N_C^8 = 4$

［プログラムリスト1］は，3×3の走査マスクWを引数として，式(1)および(2)に基づいて連結数$N_C^8$と$N_C^4$を求めるMATLABのプログラム（ファンクションMファイル）である．このプログラムは，

```
[N8C, N4C] = fnc_N8N4connect(W);
```

と記述することで実行され，8近傍連結数と4近傍連結数の値は，それぞれ出力変数N8C，N4Cとして得られる．

［プログラムリスト１］４近傍および８近傍連結数を求めるファンクションMファイル

```
% 連結数の算出 Filename : fnc_N8N4connect.m
 function [N8C,N4C]=fnc_N8N4connect(W) ;
 % 8近傍ベクトル
  N8=[W(2,3),W(1,3),W(1,2),W(1,1),W(2,1),W(3,1),W(3,2),···
  W(3,3),W(2,3)] ;
 % 4近傍ベクトル
  N4=[N8(1),N8(3),N8(5),N8(7)] ;
%連結数
       N4C=0 ;
       N8C=0 ;
       for k=[1,3,5,7]
       N4C=N4C+N8(k)-N8(k)*N8(k+1)*N8(k+2) ;
       N8C=N8C+(1-N8(k))-(1-N8(k))*(1-N8(k+1))*(1-N8(k+2)) ;
       end
% End of program
```

## ４．８近傍画素と連結数

ここでは，細線化処理で重要な８近傍連結数$N_C^8$に関してもっと詳細に調べる．ここでも理解を容易にするために，１と０からなる走査マスクの内容を白と黒からなるパターンで表示する．

３×３の走査マスク内の注目画素の周りの８近傍画素をベクトル[$f_1, f_2, \cdots, f_8$]で表すと，これは，０と１の並びからなる８ビット列である．したがって，これを白黒のパターンで表すと256通りのパターンが考えられる．しかし，よく吟味して，回転や対称・反転操作で同一パターンになるものを除くと，上述のパターン数は，全部で51パターンであることが知られる．これだけのパターンに対して８近傍連結数$N_C^8$を［プログラムリスト１］のファンクションMファイルを用いて計算し，得られた連結数ごとに３×３のマスクパターンを分類した．その結果が，**図３**～**図６**のパターンである．

なお，これらのパターンにおいて，中央の注目画素はグレーで表示している．これは注目画素も白（１）か黒（０）であるが，式(1)および(2)で定義される連結数が，注目画素に無関係にその周りの８近傍画素で計算され，連結数の算出では注目画素の値はどちらでもよいからである．しかし，注目画素を白（あるいは黒）に置くことで画素間の連結性（画素間のつながり）に果たす注目画素の役割の特性を読み取ることができる．以下，連結数ごとにこれらを考

察する．

（1）$N_C^8 = 0$ の場合：

**図 3** がこの場合の 8 近傍パターンである．これらのパターンにおいて，もし注目画素が白であれば，図の最初のパターンでは白が黒の 8 近傍画素で囲まれる．この状況の注目画素を孤立点という．また，末尾のパターンでは，白の 8 近傍画素の中に同じ白の注目画素が埋もれる．これは内部点といわれる．残りの 6 パターンは 4 近傍がすべて白の中に埋もれている．この意味では注目画素が白であれば，それは内部点である．また，注目画素が白でも黒でも 8 近傍連結の意味では白画素はどれも繋がっている．

注目画素が 8 近傍あるいは 4 近傍の意味で内部点になるのは，$N_C^8 = 0$ の場合に限られる．引き続いて議論される $N_C^8 = 1 \sim 4$ の場合では，白の注目画素は必ず 4 近傍連結の意味で，少なくとも 1 ケ所は周囲の黒に接する．したがって，黒の連結成分との境界に位置する白画素（つまり境界画素）は，$N_C^8 \neq 0$ 画素である．この事実は，2 値画像の境界線を検出する条件として利用できる．

（2）$N_C^8 = 1$ の場合：

**図 4** が $N_c^8 = 1$ の場合の 8 近傍パターンである．この場合の注目画素の特徴として次の二点を挙げることができる．

- 不可分な連結成分：どのパターンとも注目画素の値に無関係に（白であっても黒であっても），8 近傍連結の意味では白の領域はつながっていて，一つの連結成分（塊）を形成している．言い換えると，もし注目画素が白の場合，それを黒に変えても連結成分の数は一つであり，変わることはない．
- 端点：図 4 の先頭の二つの 8 近傍パターンにおいて，注目画素が白であると，白画素の 8 近傍連結のつながりがこの点で終えている．これは細線化処理における「端点」である．このことから，端点は，走査マスク行列を W とすると

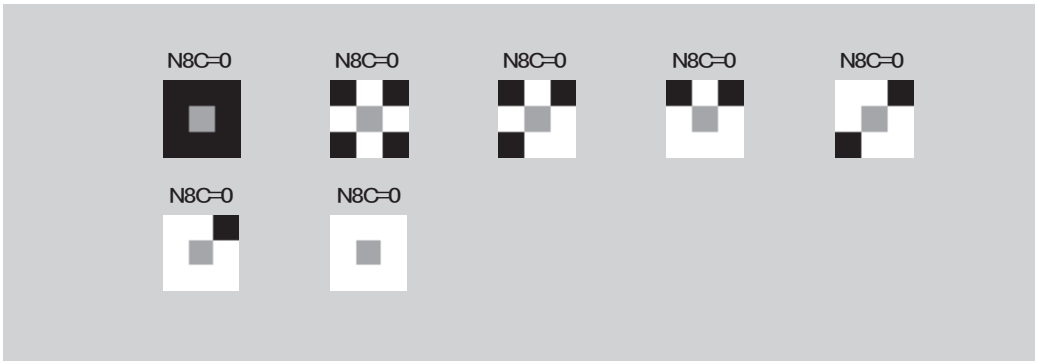


**図 3**　連結数 $N_C^8 = 0$ の 8 近傍画素パターン．0 か 1 である中央の着目画素はグレーで表示している．

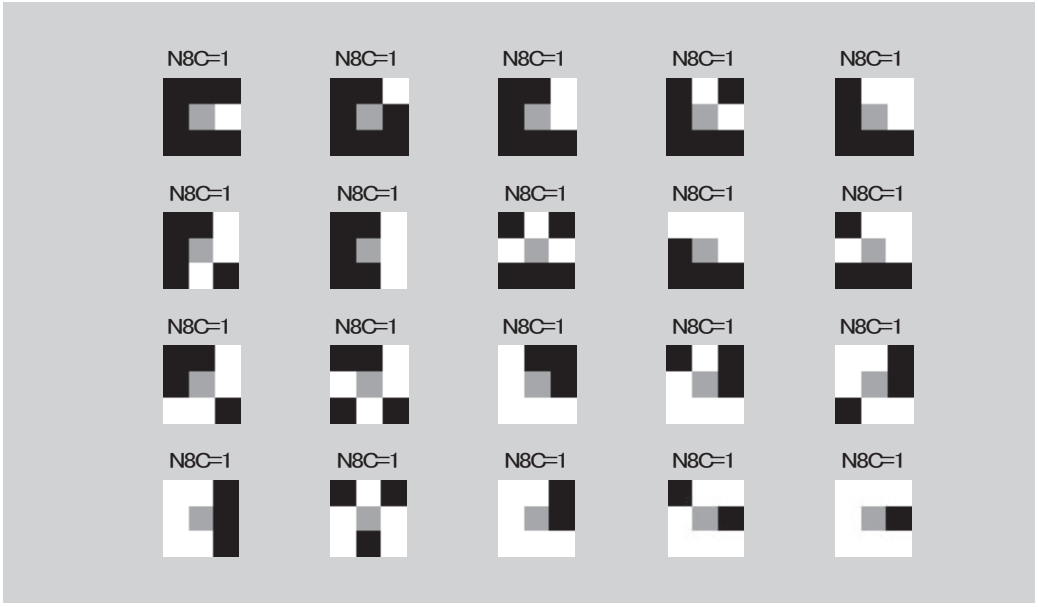


**図 4**　連結数$N_C^8 = 1$の 8 近傍画素パターン．注目画素が白でも黒でも 8 近傍連結の意味で白の連結成分は一つである．

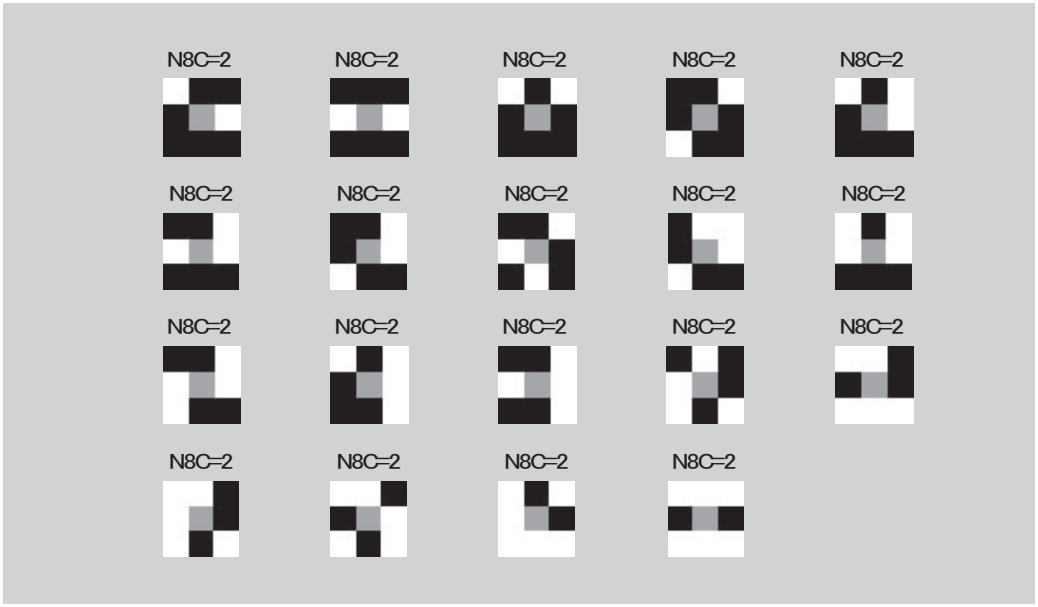


**図 5**　連結数$N_C^8 = 2$の 8 近傍画素パターン．中央の注目画素が白であれば，白の連結成分は一つであるが，黒であれば白の連結成分は二つに分かれる．

W(2,2)=1　かつ　$N_C^8 = 1$　かつ　sum(W( : ))=2

を満足する注目画素の位置と定義できる．ここで，sum(W( : ))はWの全要素の総和とい

うMATLAB表現である．

（3）$N_C^8 = 2$の場合：

**図5**の8近傍パターンは，注目画素が白であれば，白の連結成分がマスク内に一つ存在することは$N_C^8 = 1$の場合と同じである．しかし，それが黒であると，白の連結成分は2つに分かれる．言い換えると，注目画素は二つの連結成分の連結点（通過点ともいう）になっている．したがって，注目画素を反転すると画像の連結性が変化する．

（4）$N_C^8 = 3$および$N_C^8 = 4$の場合：

**図6**に示す場合である．ここでも注目画素が白であれば8近傍連結の意味で存在する白の画素がすべて繋がって一つの連結成分になる．しかし，それが黒に変わると，連結数が示すと通りに連結成分は3および4個（末尾のパターン）に分かれる．したがって，注目画素は連結点であってそれが反転すると連結性が変化する．なお，$N_C^8 = 3$と$N_C^8 = 4$の連結点は，それぞれ，分岐点および交差点と呼ばれる．

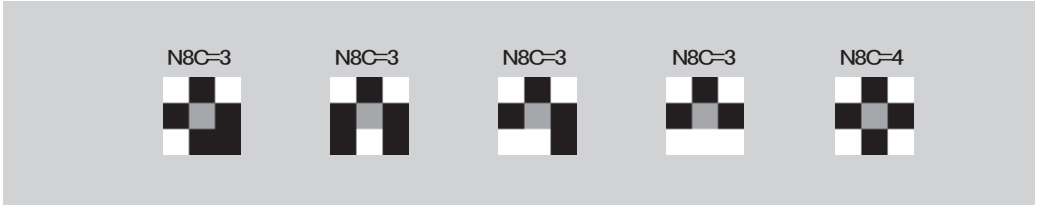


**図6** 連結数$N_C^8 = 3$および$N_C^8 = 4$の8近傍画素パターン．中央の注目画素は，いずれも連結点で$N_C^8 = 3$の場合は分岐点，$N_C^8 = 4$の場合は交差点．

上でみたように，注目画素が境界画素であるか，あるいは連結点（通過点，分岐点，交差点）かは，8近傍連結数を求めることにより，直ちに知ることができる．**図7**は，代表的な走査マスクに対して注目画素が果たす役割を連結数で分類して示したものである．

**図7**（A）は$N_C^8 = 0$の場合で，注目画素が白であれば，左が孤立点，右は内部点である．当然，それが黒であれば立場は逆になる．(B）は$N_C^8 = 1$の場合で，ここに示す二つのパターンでは，注目画素は文字通り「端点」である．細線化処理は，連結成分を切り削いでいって骨格部分の線幅を最小の画素幅にする処理であるが，端点を切り削ぐことは許されない．端点を切り削ぐと線長が縮小するからである．(C）は$N_C^8 = 2$の場合で，二つのパターンとも注目画素が白であれば，二つの連結成分を繋ぐ連結点（通過点）である．(D）は連結点（分岐），そして（E）は連結点（交差点）である．つまり，（C）～（E）の注目画素は，それが白であれば，それぞれ$N_C^8$個の連結成分を繋ぐ連結点である．

繰り返しになるが，上のように画素の連結を特徴付ける重要な地点である分岐点や交差点を含む連結点は，8近傍連結数$N_C^8$求めることにより，直ちに知ることができる．

**表1**に，8近傍連結数と注目画素が果たし得る役割，あるいはその点の特徴の関係を示して

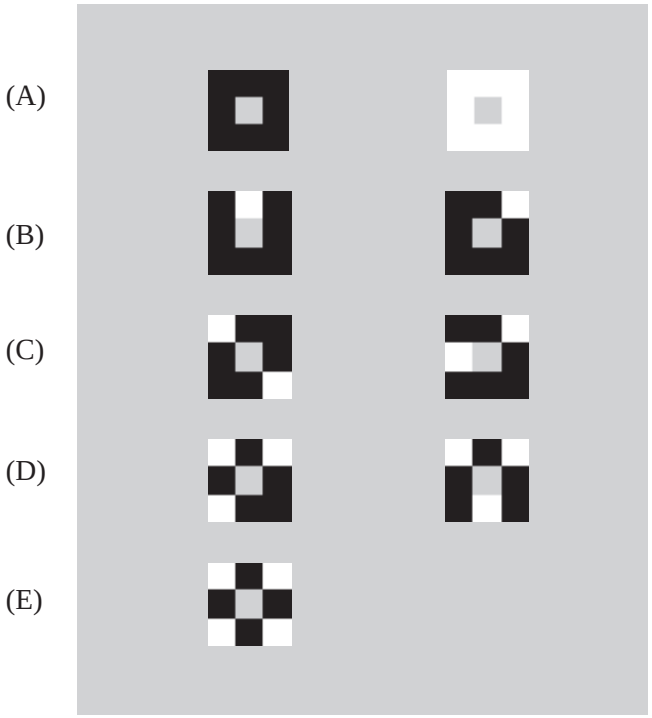


**図 7**　連結数と注目画素の役割（注目画素が白の場合）．
（A）$N_C^8 = 0$：（左）孤立点，（右）内部点，（B）$N_C^8 = 1$：端点，（C）$N_C^8 = 2$：連結点（通過点）
（D）$N_C^8 = 3$：連結点（分岐点），（E）$N_C^8 = 4$：連結点（交差点）．

**表 1**　8 近傍連結数 $N_C^8$ と注目画素の役割・特徴

| 連結数 | 注目画素の役割・特徴 |
|---|---|
| $N_C^8 = 0$ | 孤立点あるいは内部点 |
| $N_C^8 = 1$ | 端点あるいは境界点 |
| $N_C^8 = 2$ | 2 つの連結成分の連結点（通過点） |
| $N_C^8 = 3$ | 3 つの連結成分の連結点（分岐点） |
| $N_C^8 = 4$ | 4 つの連結成分の連結点（交差点） |

おく．これらの対応関係は，引き続く境界線検出や細線化に用いられる．

## 5．境界画素検出

次節で述べる細線化処理は，連結成分の境界画素の検出を基礎にしている．そのためまず境界画素検出について述べる．

**表 1** にあるように，8 近傍連結数 $N_C^8$ が 1 のときには，注目画素は連結成分の境界線に位置している．ただし，連結点も境界画素と見なせるのでこの条件は $N_C^8 \geq 1$ としてもよい．この

ことを用いると，一度のラスタースキャンによって，連結成分の境界にある画素は直ちに見いだすことができる．そして，見いだされた画素だけからなる画像を作ると境界線画像が得られる．

［プログラムリスト２］は，２値画像をラスタースキャンして，８近傍連結数が$N_C^8 \geq 1$である注目画素を見いだし，その画素だけの画像を作成するプログラム（ファンクションMファイル）である．このプログラムを使うと，境界画像Yは，あらかじめ用意されている２値の画像データXを入力引数として，

Y=fnc_EdgeDetection(X)；

記述して得られる．このプログラムでは，８近傍連結数を求めるために［プログラムリスト１］が用いられている．**図８**は実行結果の一例である．

［プログラムリスト２］

２値画像データXの境界線検出プログラム（ファンクションMファイル）

```
%境界線検出  Filename：fnc_EdgeDetection.m
    function Y=fnc_EdgeDetection(X)
%入力画像の規格化
X=double(X)； X=(X-min(X(：)))/(max(X(：))-min(X(：)))；
%エッジの画素を取り出すラスタースキャン
    [m,n]=size(X)；
    Y=zeros(m,n)；
    for kx=2：m-1；
    for ky=2：n-1；
        if X(kx,ky)==1；
            W=X(kx-1：kx+1,ky-1：ky+1)；
            [N8C, N4C]=fnc_N8N4connect(W)；
            if N8C>=1；Y(kx,ky)=1；end
    end %ラスタースキャンkxの終了
    end %ラスタースキャンkyの終了
% End of program
```

**図 8**　2値の原画像（左）と境界線検出画像（右）

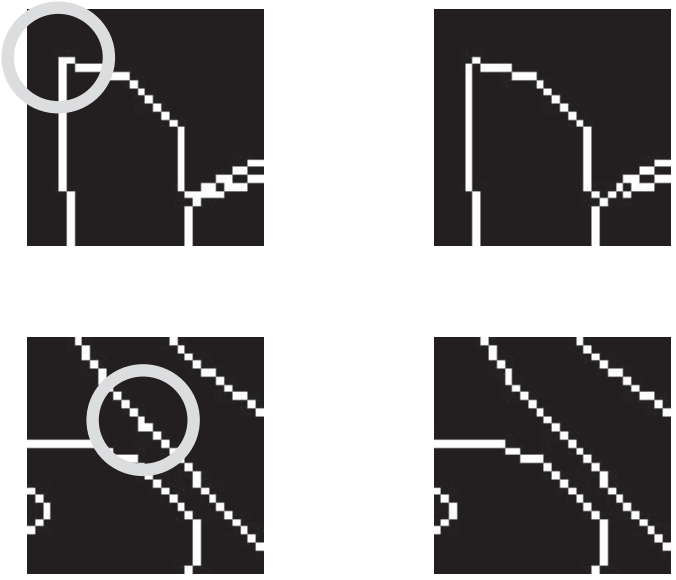

**図 9**　輪の中心付近にみえる残存 4 連結画素（左）とそれを削除した結果（右）.

ところが，**図 8** の結果は完全ではない．**図 9**（左）の輪の中にみられるような 4 近傍連結が一部に残る．このような画素を取り除くには，次節で述べる細線化の方法が役立つ．つまり，8 近傍連結で繋がる境界画素のパターンは，削除しても連結性が保たれる画素を残すことで得られる．**図 8** の結果に細線化処理を加えると 4 近傍連結は消えて，**図 9**（右）に示すように，境界画素がすべて 8 近傍連結のパターンになる．ただし，ここで示す結果の場合は，残存する 4 近傍連結の個数が少ないので，それを取り除いた境界点画像はほとんど**図 8**（右図）と

同じようにみえる．

## 6．細線化処理

### 6.1 細線化手順の概要

画像の文字認識などでは，線幅は特に意味を持たない．意味のあるのは，文字の骨格である．細線化処理は線幅のある２値の連結成分を削って，骨格部分にあたる芯線，つまり中心線に近い画素を，線幅１（１画素幅）として取り出す処理である．

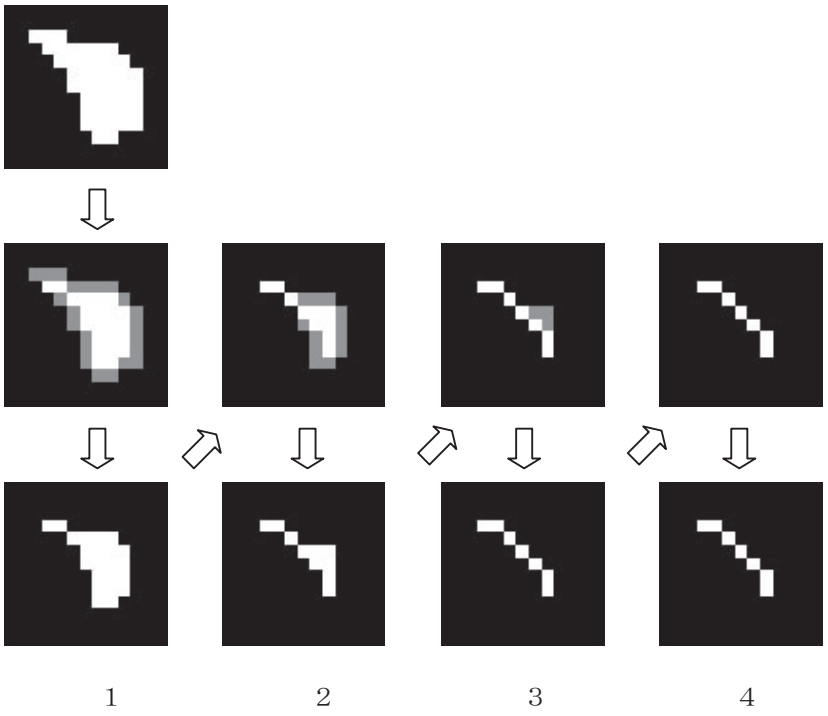


**図10** 細線化の処理過程．左上が入力画像で白の領域が細線化の対象．中段のグレーの画素が削除できる境界画素．下段はそれらを削除した結果．この処理をラススキャンごとに繰り返す．

細線化処理は，前節で述べた境界線検出を基礎にしている．つまり，連結成分の境界線を検出し，それを切り落とす処理を繰り返す．こうすることで，どの連結成分も周囲から細線化されていく．むろん，無条件ではなく，芯線が得られるための幾つかの条件の下で連結成分の画素が切り落とされる．ここで，「画素を切り落とす」と表現したが，実際には，２値画像が黒地（０）に白（１）の連結成分からなるとすると，白と黒の境界にある白（１）の注目画素を黒（０）に変える処理のことである．

まず，細線化アルゴリズムを説明する前に，処理過程を**図10**に示す．この図で，最上段の図が入力画像で黒地に白のテスト用の2値画像である．処理はラスタースキャンを繰り返して行われるが，ラスタースキャンごとの結果を図の列に1，2，3，4と付して示してある．各列の中段の図には，ラスタースキャン中に削除できると判定された画素をグレーで表示している．そして，下段はラスタースキャンが終えるごとにグレーの部分を削除し，次のラスタースキャンに渡される画像である．各ラスタースキャンで，削除が一度も起きないとき，処理は終了する．

### 6.2　画素削除条件

細線化で注目画素が削除できる条件は，

（1）対象とする連結成分の注目画素（ここでは値が1の白の画素）であること

（2）境界画素であること

（3）内部点でも孤立点でもないこと

（4）連結点でないこと

（5）端点でないこと

のすべてを満足する画素である．ここで，条件（1），（2）は削除の前提条件と言える当然の条件である．条件（3）は，内部点を削除すると「穴」ができるし，孤立点を削除すると「点」が消える．そのため，これらの画素の削除は許されない．条件（4）は，そもそも細線化処理が連結成分間の連結の要素を残す処理であるので必須の条件である．最後の条件（5）も必須で，端点を削除すると，線はどんどん短く削られていく．

以上の5条件を画像データXに対して定式化すると次のようになる．つまり，**表1**に整理した8近傍連結数$N_C^8$と位置(kx, ky)の注目画素X(kx, ky)の関係として上の条件を記述すると，削除できる画素はつぎのすべての条件を満足する必要がある．

（1）[注目画素条件] $\mathrm{X}(\mathrm{kx}, \mathrm{ky}) = 1$

（2）[境界画素条件] $N_C^8 \geq 1$

（3）[内部点，孤立点保存条件] $N_c^8 \neq 0$

（4）[連結点保持条件] $N_C^8 \neq 2, 3, 4$

（5）[端点保持条件] $N_C^8 = 1$ かつ $\mathrm{sum}(\mathrm{W}(:)) \neq 2$

なお，（5）の条件におけるWは3×3の走査マスクの行列

$$\mathrm{W} = \mathrm{X}(\mathrm{kx}-1 : \mathrm{kx}+1, \mathrm{ky}-1 : \mathrm{ky}+1)$$

であり，sum(W(:))はWの行列要素の総和を与えるMATLAB関数である．ここで，これらの5条件をよく見ると，条件が重複していることに気がつく．それは，（2）～（4）の条件で，これらは単に$N_C^8 = 1$ですべて満足される．したがって，この$N_C^8 = 1$の条件と（1）の［注目画素条件］および（5）の［端点保持条件］の3条件が注目画素削除条件である．

よって，X(kx, ky)の画素が削除できるのは，

$$X(kx, ky) = 1 \quad \text{かつ} \quad N_C^8 = 1 \quad \text{かつ} \quad \text{sum}(W(:))\tilde{}=2$$

と与えられる３条件が同時に満足されるときである．

### 6.3 線幅２の線消失防止

ところが，前節の画素削除条件下で細線化を実行しても，目的通りには芯線が得られない．芯線が得られる箇所と線が消える箇所が現れる．これは，これから述べる線幅２の処理に原因がある．**図11**はこれを説明するための連結成分のモデル図である．つまり，大きな連結成分の境界画素を上述の画素削除条件に従って繰り返し削除していくと，芯線が得られる一歩手前でこの図に示すような二通りのどちらかの連結成分になる．

**図11**の上部に示す連結成分は線幅３，下部は線幅２であって，いずれも簡単のために，水平（スキャン方向）に延びた細線化の一歩手前の連結成分である．線幅３の連結成分では上側と下側の境界画素（グレーの部分）が削除されると，芯線に相当する線幅１の線（白い部分）が得られ，首尾よく細線化が達成される．一方，線幅２の連結成分を同様に処理すると，連結成分の上下両方の画素とも削除条件を満足するので両方とも削除され，結果として連結成分のすべてが消失する．むろん芯線は得られない．このように，線幅２の削除においては，線消失防止の工夫が別に必要になる．

細線化アルゴリズムは，基本的に連結成分の周囲，つまり境界点を削除していくので，削除が左右，上下に偏ることなく実行され中央線に近い芯線が得られる．このとき，一回のラスタースキャンが終えると，画像は一回りスリムになる．いま，ある時点でのラスタースキャンの対象画像をXとすると，つぎのラスタースキャンでは一回りスリムになった画像が入れ替わって対象画像がXになる．

**図11** 線幅３と線幅２の連結成分（モデル図）．削除の可否を説明するために画素の並びを濃淡で区別している．

線幅２の線消失防止は，入れ替わった画像Xだけではなく，そのラスタースキャン途上の画像YをXと区別して扱うことで達成できる．つまり，

画像X：ラスタースキャン中は不変で，全面のラスタースキャンが終えるごとに更新される画像.

画像Y：ラスタースキャン開始時は画像Xで，ラスタースキャンが進んで連結成分の削除が進行するのに伴って変化する画像.

である．そして，画像Xは，境界画素を決める際に用い，画像Yに対して注目画素の削除条件を用いる．こうすると，一巡のラスタースキャンごとに連結成分の境界画素が削除の対象になり，それらの削除の可否が削除が進行中の画像Yで決まる．**図11**の線幅2の場合には，連結成分の上部が削除されたあとの画像Yに「画素削除条件」を適用することになる．このときには，端点と連結点からなる線幅1の線であるので「画素削除条件」は満足されない．したがって，この線が削除されることはなく細線化が達成される．

### 6.4　細線化プログラムと実行例

［プログラムリスト3］が上述した線幅の線消失防止を含んでいる細線化アルゴリズムのプログラムである．これは，ファンクションMファイルであるので，メインプログラムで2値の入力画像データXを用意し，このプログラムを

Y=fnc_Thinning (X) ;

と記述することによって，細線化画像データが出力変数Yとして得られる．

［プログラムリスト3］細線化プログラム（ファンクションMファイル）

```
%細線化プログラム fnc_Thinning.m
function Y=fnc_Thinning(X) ;

disp('') ; disp('.....Now thinning')
[m,n]=size(X) ;
X=(X-min(X( : )))/(max(X( : ))-min(X( : ))) ;

% 周囲の前処理(初期化)
X(1, : )=0 ; X(end, : )=0 ; X( : ,1)=0 ; X( : ,end)=0 ;
Y=X ;
% 細線化処理
key=0 ;   %while文の条件パラメータ
while key==0
     key=1 ;    %while文の終了を制御するパラメータ
     % ラスタースキャン
     for kx=2 : m-1 ;
```

```
    for ky=2：n-1；
        if X(kx,ky)==1；%■注目画素条件
            WX=X(kx-1：kx+1,ky-1：ky+1)；  %Xの3×3マスクデータ
            [NC8X,NC4X]=fnc_N8N4connect(WX)；  %Xの連結数
                if NC8X==1； %■境界画素の条件(削除条件)
                WY=Y(kx-1：kx+1,ky-1：ky+1)； %Yの3×3マスクデータ
                [NC8Y,NC4Y]=fnc_N8N4connect(WY)；        %Yの連結数
                 if NC8Y==1&&sum(WY(：))~=2；Y(kx,ky)=0；key=0；end
                                    %■削除中データYに対する削除条件
                                    % key=0は処理継続key=1は処理終了へ
            end
        end
    end %ラスタースキャンkxの終了
    end %ラスタースキャンkyの終了
X=Y；
end  % End of while
%End of program
```

**図12**は，［プログラムリスト３］の実行例である．ここでは，中央の図が，左図の文字「四季」の細線化画像である．ここに見られるように，筆の跳ね返しの箇所もほぼ忠実に細線化されている．しかし，筆の跳ね返しでない箇所にも，文字の骨格には関係しない「ヒゲ」が現れている．このヒゲの縮減は細線化処理では必要な課題である．最後にこれについて述べる．

### 6.5「ヒゲ」の縮減処理

細線化の結果には目的とする芯線の他に，**図12**（中央図）の所々にみられるように，不要なヒゲが予期しないところに現れることが常である．ヒゲは原画像に起因するだけでなく，ディジタル画像が空間的なサンプリングの結果であることにも関係している．処理のアルゴリズムに従って，予期しない点の画素が端点と一度判定されると，その画素は「端点」として保存され，結果としてヒゲとして最後まで残る．

このようにヒゲは，アルゴリズム上は端点と同様に見なされる．そのため，細線化された結果の画像の中から端点を検出し，その画素を切り落とすと縮減できる．しかし，この縮減処理は，本来の端点も縮減する．したがって，この処理が許容されるのは，ひげを縮減する程度の端点の縮減が，芯線全体に対して大きな影響を与えないとみなされる範囲に制限される．

［プログラムリスト４］が端点縮減のプログラム（ヒゲ縮減処理）のファンクションMファイルである．これは，

**図12**　（左）入力2値画像，（中）細線化された画像，（右）ヒゲ削減処理後の細線化画像（削減処理回数10）．

Y=fnc_Shaving (X, nShave) ;

と記述して実行される．ここで，引数Xが細線化処理が済んだヒゲのある入力画像（入力変数）であり，第2の引数nSaveは端点を削除する回数である．つまり，この回数だけヒゲも端点は短縮する．**図12**の右図はnShave＝10で端点を縮減した結果で，不要なヒゲの大半が消失している．

［プログラムリスト4］端点縮小のプログラム（ヒゲ縮減処理）

```
%端点縮小のプログラム（ヒゲ縮減処理）　fnc_Shaving.m
function Y=fnc_Shaving(X, nShave) ;
Y=X ;
[m,n]=size(X) ;
for loop=1 : nShave　% nShave : 端点縮減回数
   %ラスタースキャン
   for kx=2 : m-1 ;
   for ky=2 : n-1 ;
      if X(kx,ky)==1 ;
               W=X(kx-1 : kx+1,ky-1 : ky+1) ;　%Xの論理フィルタ
               [NC8,N4C]=fnc_N8N4connect(W) ;　%Xの8近傍連結数
```

```
      if NC8==1&&sum((W( : )))==2 ; Y(kx,ky)=0 ; end %ひげの条件と縮減
      end
    end %ラスタースキャンkxの終了
    end %ラスタースキャンkyの終了
X=Y ;
end
% End of program
```

## 7．おわりに

本稿を纏める最後の時期に，筆者は「Hilditch細線化アルゴリズムにおける連結性保存条件の誤用に関する注意喚起」と題する最近の論文を発見した．これは本稿と同様な観点からHilditchアルゴリズムの正しい使い方を喚起したもので，本稿はこれと重複する部分があるかもしれない．しかし，ここでは筆者自身の視点から独自に細線化のアルゴリズムを再検討した．そのため研究論文ではなく，Hilditchの細線化アルゴリズムの精査として技術報告として著した．いずれにしても，本稿がいくつかの文献やwebにみられるこのアルゴリズムに関する誤った記述の一掃の役立つことを期待している．

本研究は，戦略的研究基盤形成支援事業「電磁・光センシングを主体とする生体関連情報の先進的計測・処理技術の開発と応用」の一環として行った。

### 参考文献

1）谷口慶治（編）：画像処理工学（基礎編）第7章，共立出版（1996）.
2）田村秀行（編著）：コンピュータ画像処理　第5章，オーム社（2002）.
3）岡崎彰夫：はじめての画像処理技術　第3章，工業調査会（2000）.
4）大関和夫：入門画像処理　第3章，コロナ社（2010）.
5）Hilditch, C. J. : Linear Skeletons from Square Cupboards, Machine Intelligence 4, edited by B.Meltzer et al., University Press, Edinburgh, pp.403–420 (1969).
6）吉田大海，田中直樹，井上健：Hilditch細線化アルゴリズムにおける連結性保存条件の誤用に関する注意喚起，画像電子学会誌 Vol. 39，No. 4，Page 490-493（2010）.