画像の等高線表現とその応用

浅野哲夫† 木村宗市†† 嶋津茂昭††

画像は各画素の明るさの情報を表す行列の形で表現されるのが一般的であるが，画素の明るさをその位置における標高と見なすと地形図と同等に扱える．本論文では，地形図と同様に画像を等高線の集合として表現する方法を従来の行列表現と様々な応用の角度から比較検討する．最初に，この表現の効率の良い求め方とデータの蓄え方を提案し，次に，等高線表現を用いた画像処理の応用として，画質改善，自然な画像拡大，キズ領域の修復などの方法を紹介する．

Contour Representation of an Image with Applications

TETSUO ASANO,† SOUICHI KIMURA†† and SHIGEAKI SHIMAZU††

A common representation of an image is due to a matrix with intensity levels as its elements. Regarding intensity levels of pixels as height at corresponding locations, an image can be viewed as a terrain map which is usually represented by a set of contour lines. In this paper the contour line representation of an image is compared with the conventional matrix form from various aspects of applications. We first propose an efficient algorithm for finding such a representation and a data structure for the representation, and then survey several applications of the contour representation, such as image quality improvement, natural image enlargement, and flaw repairing.

1. はじめに

計算幾何学の分野で開発されたアルゴリズム的手法を画像処理やコンピュータ・ビジョンの諸問題に応用する試みは今までにも数多く見受けられる．たとえば，“Graphics Gems”のシリーズ本[15)~19)]にはコンピュータ・グラフィックスにおける多数のアルゴリズムが収録されており，計算幾何学の問題として定式化されている問題の記述も多い．何より，アルゴリズムの記述だけでなく，C言語などで記述されたプログラムのほかに，アルゴリズムとしての計算複雑度の解析についても言及されていることが多いのが特徴である．

さて，計算幾何学の歴史を振り返ってみるとき，必ずしも画像処理やコンピュータ・グラフィックスの分野にポジティブな貢献ばかりをしてきたわけではないが，中には従来にない斬新な考え方に基づいて本質的な計算時間の改良が行われた例もある．たとえば，画像の距離変換（白黒の2値画像が与えられたとき，各白画素について最も近い黒画素までの距離を求める問題）は，画像処理における重要な基本的操作の1つであるが，最近まで最適なアルゴリズムが得られていなかった．これに対して，計算幾何学の分野で，最近ほぼ同時に2つの線形時間アルゴリズムが提案され，この問題に終止符を打つことができたようである．Breu ら[5)]がボロノイ図の概念を用いているのに対して，Hirata ら[9)]は，問題を多数の同型の放物線の下側エンベロープ（下から見たときに見える部分）を求める問題に変換することによって線形アルゴリズムを得ている．

これとは違った形で計算幾何学の結果が応用されている例として，たとえば，2値画像に含まれる曲線成分を検出する問題をあげることができる．この問題はパターン認識における最も基本的な問題の1つであるため，従来から非常に多数の方法が提案されてきている．問題自体の困難さのために，曲線成分とは何かの数学的定義から始めて，定義に合致するすべての曲線成分を検出するという試みはなされず，画像の性質を仮定したうえでの発見的手法が大多数であった．その意味では，筆者らの研究[2)]は，計算幾何学的手法（双対変換と双対平面上での最適セル探索問題）に基づいて，数学的定義にあうすべての可能な曲線成分を多項式時間で列挙できることを保証した最初の方法であ

† 北陸先端科学技術大学院大学情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku
†† 大日本スクリーン製造株式会社
Dainippon Screen MFG

ろう．

さて，量子化された画像の表現方法として最も一般的なものは，各画素の明るさ（カラー画像の場合は赤，青，緑）のレベルを2次元行列の形で表現したものであろう．これとは違った形での表現方法として，等高線表現（あるいは，境界値表現）がある．この方法は，画素の明るさを対応する地点の標高と見なして画像を地形図の形に変換し，等高線の集合によって画像を表現しようというものである．すなわち，それぞれの明るさのレベル i について，レベル i 以上の画素で構成される領域の境界（レベル i の等高線）を求めるという方法である．明らかに，等高線情報から元の画像行列を再構成することはつねに可能である（実際，簡単な方法で線形時間で画像の再構成が可能である）．画像を地形図のように扱う方法は，Sugihara[12]らによっても提案されている．彼らは地形図としての画像をBézier曲面で近似しようとしているが，これは主に高さ方向の近似といえる．これに対して本論文で述べる方法は，高さ方向ではなく，画像内で各対象物が占める領域を幾何学的に表現しようとするものである．

画像の等高線表現法は，1960年代の終わりに画像のデータ圧縮の目的で用いられたことがある[14]．文献14)の方法では，画素の中心を等高線が通るように定めているが，1画素分の幅しかない領域は面積を持たなくなってしまうため，本論文で対象とする応用において問題があると考えられる．これに対して，本論文では等高線は画素と画素の間を通るものとしている．これも違いの1つである．また，文献14)の方法は元来データ圧縮を目的に考案されたものであるが，当時の画像サイズではいざ知らず，256×256 画素程度の小規模な画像でも画像行列による表現の平均20倍程度のデータ量となり，とてもデータ圧縮に用いることはできない．

本論文では最初にすべての等高線を列挙する効率の良いアルゴリズムとデータ構造を提案する．同様のアルゴリズムは，文献10)の中にも記述はあるが，各レベルごとに2値画像に変換してその境界を追跡するという単純な方法であり，レベル数と画素数の積に比例する時間が必要である．本論文で提案する方法は，レベル数と画素数の和と出力の長さに比例する時間で等高線を求めるものであり，効率の面で優っている．また，等高線をすべて正直に蓄えることはメモリー的に非現実的であることが多いので，よりコンパクトな表現が必要になる．本論文では，間接的に等高線情報を蓄える方法を提案する．ここで，レベル i 以上の領域の境界が必要になったとき，すべての等高線が蓄えられていれば，もちろん対応する等高線の長さに比例した時間だけで求めることができる．しかしながら，等高線の全体を知らなくても，等高線を構成する1辺だけが分かっていれば，画像行列上でその辺から順に等高線をたどることができる．各辺では，接続する3辺を調べるだけで次にたどるべき辺を特定することができるから，やはり等高線の長さに比例する時間で等高線を出力することができる．そこで，各等高線について，その構成辺を正確に1辺ずつ含むような辺集合（シード辺集合という）を求めておけばよいことになる．

本論文では，画像の等高線表現を用いると，画像の画質改善，自然な画像の拡大・回転，キズ領域の自動修復などの問題が無理なく扱えることも示す．

2. 準　　備

本文では，画像を $G = (g_{ij})$, $i, = 1, 2, \ldots, v-2$, $j = 1, 2, \ldots, h-2$ という行列の形で指定するものとする．ここで，g_{ij} は，画素 (i, j) の階調値を表すものであり，0から $L-1$ までの整数値をとる．通常，L の値は2のベキ乗の値にとられることが多い．

取扱いを容易にするために，本来は2次元の画像行列を1次元配列によって表現する．すなわち，画像行列の (i, j) 要素を1次元配列の $k = i * h + j$ 番目の要素に対応させる．画像行列の列数と行数をそれぞれ h, v とすると，番号 k の画素の上下左右の隣接要素は，順に $k-h, k+h, k-1, k+1$ ということになり，2次元の場合に比べて簡潔である．本文では，番号 k の画素を p_k という記号で表すことにする．

画素と画素の間の格子辺にも通し番号をつける．本論文では，格子に囲まれた正方形の中心に画素があると考え，格子と格子を結ぶ水平および垂直の格子辺の系列として等高線を表現する．したがって，各画素は上下左右の4個の格子辺で囲まれている．画素が全部で n 個あるとき，格子辺はその約2倍だけある．格子辺に通し番号をつける方法は何通りもあるが，ここでは，画素 p_k の下を通る水平格子辺に $2k$ という番号を，p_k の右を通る垂直格子辺に $2k+1$ という番号をつけることにする．このままでは画像全体の左と上の境界上の格子辺には番号がつかないので，与えられた画像の周囲に1画素分だけダミーの画素（レベルは -1 とする）を置くものとする．このとき，辺には0から $2hv+h+v-1$ まで $E = 2hv+h+v$ 通りの番号がつくことになる．各レベル i について，そのレベル以上の領域 R_i の境界（等高線）は格子辺の系列として表現するが，同時にそれぞれの格子辺はレベル i

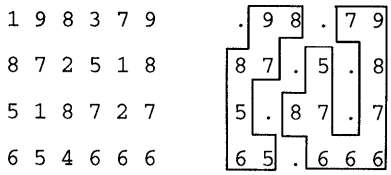

図1 画像行列と対応する等高線
Fig. 1 Image matrix and its corresponding set of contours.

以上の画素を右に見るように方向づけるものとする．すなわち，等高線は山領域では時計回りに，谷領域では反時計回りに方向づける．

理解を助けるために簡単な例を示そう．**図1** (a) に示した行列を画像行列とする．レベルが1から9まであるので，9通りのレベルについて等高線を求めることになる．同図 (b) に示したのがレベル5以上の領域の境界に相当する等高線である．地形図の等高線と違って，同じ場所を複数の等高線が通過するので見やすい形で等高線を表現するのが難しい．

さて，ここまでは単に「レベル i 以上の画素からなる領域 R_i」という表現を用いてきたが，この表現では画素の4連結性を仮定している．すなわち，ある画素の集合 S が領域 R を形成するとは，どの2個の画素についても，一方の画素から S に属する画素だけをたどって他方の画素に至る道で毎回水平または垂直方向にしか移動しないようなものがあることを意味する．

定義より，すべてのレベル i, $0 \leq i < L-1$ について，$R_{i+1} \subseteq R_i$ が成り立つ．領域 R_i の境界は1つとは限らない．穴を含む場合があるからである．本論文を通じて，各等高線は領域の内部（すなわち，レベルが高い画素のある方）を右手に見るように方向づけされていると仮定する．したがって，領域の外部境界は時計回りに，内部境界は反時計回りに方向づけされている．

3. 等高線表現を求めるアルゴリズム

画像の等高線表現は，画像を等高線を用いて表現したものである．L レベルの階調を持つ画像を表現するには，L 通りの領域 $R_0, R_1, \ldots, R_{L-1}$ すべてについて，その境界線を求めなければならない．以下では，領域 R_i の境界線のことをレベル i の等高線と呼ぶことにする．ただし，各 R_i は1つの領域とは限らず，一般に多数の連結成分に分かれている．各レベル i について，画像を1回のラスタースキャンするだけで領域 R_i の境界を求める方法[6]も知られているが，その方法ではレベル数 L と画素数 n の積 nL に比例する時間が必要である．ここでは，まず，すべての境界線を出力の長さ（境界線の長さの総和，すなわち，境界線を構成する格子辺の個数）に比例する時間で求めるアルゴリズムを示す．すなわち，n を画像サイズ（画素数），K をすべての境界線（等高線）の長さ（辺の個数）の総和とするとき，$O(n+K)$ の時間ですべての等高線を出力するアルゴリズムとデータ構造を示す．

以下に述べるアルゴリズムは，次の簡単な観察に基づいている．

観察 格子辺 e_i の両隣の画素 p と q の濃淡レベルが，それぞれ $g[p], g[q]$ であるとする．$g[p] \neq g[q]$ であるとき，この格子辺は，レベル $\min(g[p], g[q])+1$ から $\max(g[p], g[q])$ までの等高線に含まれる．

そこで，まず画像全体を走査し，レベル g の等高線に含まれるべき格子辺を連結リスト $RS[g]$ として管理しながら，各格子辺 e_i に対して次の操作を行う．ただし，レベルは昇順に順次処理していくので，レベル g を処理しようとするときに，リスト $RS[g]$ にレベル g の等高線に含まれるべき格子辺がすべて含まれるようになっていればよい．e_i の両隣の画素のレベルを g_1, g_2 $(g_1 \leq g_2)$ とする．$g_1 = g_2$ なら，e_i が領域の境界になることはないので何もしない．$g_1 \neq g_2$ の場合，この辺はレベル g_1+1 から g_2 までの等高線に含まれる．そこで，レコード (e_i, dir, g_2) をリスト $RS[g_1+1]$ に加える．ただし，dir は，辺 e_i の方向を表す．本論文では，高レベルの画素が（有向）辺の右にくるように辺の方向を定める．ここで，垂直辺は上下の方向，水平辺は左右の方向しかないから，方向 dir は1ビットで表すことができる．つまり，レコード (e_i, dir, g_2) がリスト $RS[g_1+1]$ に含まれているということは，格子辺 e_i が g_1+1 から g_2 までのレベルの等高線に含まれるということを意味している．

与えられた画像に対して -1 という特別なレベルを持つ画素をその周囲に配置したから，上記の操作を各格子辺について実行したとき，リスト $RS[0]$ は，レベル0以上の画素からなる領域 R_0 の境界線（レベル0に対応する等高線）上のすべての格子辺を含んでいる．リスト $RS[0]$ に含まれる任意の格子辺から始めて境界（レベル0の等高線）上をたどっているとき，1以上のレベルの画素に隣接している各格子辺は，後でレベル1以上の等高線にも含まれるから，この格子辺を $RS[1]$ に転送しておく必要がある．

一般に，レベル i 以上の画素からなる領域の境界

(レベル i の等高線) をたどっているとき，その境界上の格子辺の中に $i+1$ 以上のレベルを持つ画素に隣接するものがあれば，その格子辺をレベル $i+1$ の等高線上の格子辺を管理するリスト $RS[i+1]$ に転送する．

このように，スタートの格子辺さえ指定されれば，領域の境界をたどることができる．1つのレベル i に対して領域 R_i は，一般に複数の連結領域から構成されるので，1つの等高線をたどり終わったとき，次のスタート格子辺を知る必要がある．そのために，最初，R_i に属するすべての格子辺集合を求めておき，リスト $RS[i]$ に入れておく．レベル i の等高線を順にたどっていくとき，等高線上の各格子辺 e_j について，次の操作を行う．すなわち，まずこの格子辺 e_j をリスト $RS[i]$ から削除し，さらにこの格子辺が次のレベル $i+1$ の等高線にも含まれるとき，e_j をリスト $RS[i+1]$ に加える．このように格子辺の集合を管理しておけば，1つの等高線をたどり終わったとき，リスト $RS[i]$ が空でなければ，リスト $RS[i]$ にある任意の格子辺をスタート辺として次の等高線をたどることができる．

したがって，ここでは以下の操作を効率良く実行するためのデータ構造が必要になる．

(1) 格子辺 e_j をリスト $RS[i]$ に挿入する．
(2) 格子辺 e_j をリスト $RS[i]$ から削除する．
(3) 格子辺 e_j がリスト $RS[i]$ に存在するかどうかを判定する．
(4) リスト $RS[i]$ が空でないとき，そこに含まれる任意の格子辺を取り出す．

格子辺には 0 から $E-1$ までの整数の番号がつけられていることを考えると，上記の操作をすべて定数時間で実行するデータ構造を考案するのは難しいことではない．実際，格子辺，その方向ビット，および濃淡レベルからなる3項組 (e_i, dir, g) を双方向連結リストで管理し，各格子辺 e_i に対して e_i を含む3項組へのポインタを持っておけばよい．したがって，次の補題を得る．

補題1 n 個の画素からなる画像が与えられたとき，K を全等高線の長さの和として，$O(n+K)$ の時間と $O(n)$ の記憶領域だけですべての等高線を求めることができる．

4. コンパクトな表現方法

前章では，すべての等高線を出力サイズに比例する時間だけで求めるアルゴリズムとデータ構造を示した．しかしながら，等高線の長さの総和 K は一般に

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	1
4	5	6	7	8	9	1	0	2
3	0	0	0	0	0	2	0	3
2	0	2	3	4	0	3	0	4
1	0	1	0	0	0	4	0	5
9	0	9	8	7	6	5	0	6
8	0	0	0	0	0	0	0	7
7	6	5	4	3	2	1	9	8

図2 等高線の長さの総和が最大になる例 ($L=10$ の場合)
Fig. 2 An example of an image in which the total length of contours is maximum (in the case of $L=10$).

n に比べて非常に大きく，すべての等高線を蓄えようとすると大量のメモリー領域を必要とし，現実的でない場合が多い．いくつかの標準的な画像 (階調数各色8ビットで 640×500 程度の画像) で実験をしたところ，画像行列の約 10 ～ 40 倍程度のメモリーを要した．もちろん，等高線の長さの総和は与えられた画像によってまちまちであるが，たとえば，隣の画素とのレベル差が平均的に階調数の p% 程度であったとすると，各格子辺は階調数 L の p% 程度の等高線に含まれることになるから，等高線の長さの総和は，ほぼ $2nL\times p/100$ 程度ということになる．$p=10, L=256$ とすると，約 $50n$ となり，画像行列の 50 倍程度の容量が必要であることになる．

最悪の場合，すなわち，白画素 (最高レベル) と黒画素 (最低レベル) が交互に並んだ市松模様の場合，すべての格子辺がすべてのレベルの等高線として現れるから，画像行列に比べて階調数倍のメモリーを要するということになる．もちろん，市松模様の場合には，すべてのレベルで等高線がまったく同じであるからデータ圧縮が可能である．しかしながら，$1, 2, \ldots, L-1$ というレベルが渦巻き状に連鎖し，その間がレベル 0 の画素で埋められているような画像 (図2に $L=10$ の場合の画像行列を示す) では，どのレベルについても他のレベルと異なる等高線を持ち，しかもレベル i 以上の領域はほぼ $n/2L$ 個の連結領域に分割され，各連結領域の境界は，ほぼ $2(L-i-1)+2=2L-2i$ 以上の長さを持つ．したがって，2以上のレベルの等高線の長さの総和は，少なくとも，$(n/2L)\sum_{i=2}^{L-1}(2L-2i)=\Omega(nL)$ ということになる．

4.1 シード辺集合による表現

上に述べたように，等高線をすべて蓄えることは現実的ではない．我々が提案するのは，等高線全体を記憶する代わりに，各等高線について少なくとも1辺を含むような格子辺の集合と画像行列を持つという方法である．画像行列を持つことになるので，記憶スペースは必ず増加するが，格子辺集合の蓄え方を工夫する

と，全体で画像行列の2倍以内の記憶スペースにおさえることが可能である．レベル i の等高線を追跡するのに，追跡の出発点となる格子辺と画像行列が与えられていれば，レベル i 以上の画素を右手に見るように格子辺を順にたどることができる．しかも，各格子辺の次の格子辺の候補は3辺だけであるから，この作業はたかだか2回のif文の実行で済ませることができる．

このように，各等高線について出発点となる格子辺だけを蓄えておけば，任意の等高線をその長さに比例する時間で求めることができる．このような出発点となる格子辺のことを**シード辺**ということにする．しかしながら，この方法でも多量の記憶スペースが必要になる場合がある．すなわち，上で述べた市松模様の場合，外枠の等高線（レベル0以上に対応）を除くすべての等高線は1個の画素を取り囲む長さ4の辺列であり，連結な等高線の個数は全部で $O(nL)$ 個ということになり，シード辺の個数も $O(nL)$ ということになる．しかしながら，この場合には同じシード辺が何度も重複して含まれており，非常に冗長な表現となっている．この点を改善できれば全体の記憶スペースを改善できる．

冗長性をなくすために，シード辺の番号だけを蓄えるのではなく，各シード辺がどれだけのレベルで現れるかの情報も付加して蓄えることにするというのが提案する方法である．すなわち，番号 k のシード辺 e_k が，レベル i から j までの等高線のシード辺であるとき，(k,i,j) という3項組を記憶することにする．このような3項組を**区間表現されたシード辺**という．この表現を用いると，全体の記憶スペースを $O(n)$ におさえることができる．まず，この事実を示そう．

4.2 NWコーナーを用いる方法

先にも述べたように，等高線は領域の内部を右に見るように方向づけられた格子辺の系列として表現される．等高線は閉じているから，必ずどこかには

(a) 垂直に上に進んだ後水平に右に進む角か，

(b) 水平に左に進んだ後下に降りる角

のいずれかがあるはずである．山領域の等高線は時計回りであるから，必ず上の(a)のタイプの角を含んでおり，谷領域は(b)のタイプの角を含んでいる．このような角をNWコーナーと呼ぶことにする．このとき，画像行列を走査していって，何らかの等高線のNWコーナーとなる箇所を検出するのは容易である．

このようにしてすべてのNWコーナーを求めた後，各コーナーを構成する水平辺 e_k について，その上下の画素のレベル a,b（ただし，$a<b$）を用いて，(k,a,b) という3項組を出力する．この操作をすべてのNWコーナーについて行うと，3項組の集合 T が得られる．この操作は明らかに画像を1度スキャンするだけで実行できる．この方法を**NW法**と呼ぶことにする．

NW法で求めた区間表現されたシード辺集合には，1つの等高線に複数個のNWコーナーが含まれていることは十分に考えられるので，冗長性はまだ残っている．しかし，この3項組の集合を画像行列と組み合わせて用いるとすべての等高線を生成することができる．これを示すには，任意のレベル i について，レベル i の等高線（レベル i 以上の領域の境界）が生成できることを示せればよい．下にその概略を示す．

レベル q の等高線の出力

(0) この手続きを始める時点では，どの格子辺もマークされていないものとする．

(1) 区間表現されたシード辺集合 $T=\{(k_i,a_i,b_i)\}$ において $a_i \le q \le b_i$ を満たすようなすべての3項組を求め，T_q とする．

(2) T_q が空でない限り以下の操作を繰り返す．

(3) T_q から任意の要素 (k_i,a_i,b_i) を取り出す．このとき，格子辺 e_{k_i} がマークされていれば，(2)に戻る．

(4) 格子辺 e_{k_i} からレベル q の等高線をたどり，途中の格子辺を出力するとともにマークをつけておく．

(5) T_q が空になったとき，レベル q の等高線はすべて出力されている．ただし，次のレベルの処理を考慮して，出力された等高線を再びたどり，境界線上の格子辺のマークを外していく．

上記の方法で正しく等高線が求まることは明らかであろう．また，区間表現されたシード辺の集合 T のサイズは画像の複雑度によってまちまちであるが，重要なことは，どの格子辺もたかだか1度しか T には含まれることはないという事実である．したがって，集合 T のサイズはレベル数 L に関係なく，$O(n)$ である．実際の画像について実験すると，T のサイズは画像サイズの約40～50%といったところであるので，3項組として蓄えても画像行列の記憶スペースと大差はない．

補題2 NW法で求めた区間表現されたシード辺の集合 T はどの等高線とも共通部分を持ち，そのサイズは画像のサイズを超えない．この表現自身は画像サイズの線形時間で求めることができ，さらに，この表現を用いると，任意のレベル q について，レベル q の等高線をその長さに比例する時間だけで出力すること

ができる．

厳密には上記の補題は正しくない．なぜなら，レベルを指定しても，そのレベルの等高線に含まれるシード辺をすべて求めるための時間が必要だからである．これについては次章で述べる．

3章においてすべての等高線をたどるアルゴリズムについて述べたが，NWコーナーの概念を用いると記憶スペースを節約することができる．3章のアルゴリズムでは，レベル i の等高線を求めようとするとき，その等高線に含まれるすべての格子辺をリスト $RS[i]$ に蓄えた．しかしながら，リスト $RS[i]$ は等高線の各連結成分について少なくとも1つの格子辺を含んでいるだけでよいから，すべての格子辺を蓄えるのではなく，NWコーナーの格子辺だけを管理するように改めることができる．先のアルゴリズムでは，格子辺 e_j をたどるとき，e_j はリスト $RS[i]$ に必ず含まれていたが，上のように修正したアルゴリズムでは，e_j が $RS[i]$ に含まれているとは限らない．しかしながら，e_j が $RS[i]$ に含まれていれば単に $RS[i]$ から削除するだけでよいので，手続きが面倒になることはない．

4.3 区間表現されたシード辺の蓄え方

ここでは，区間表現されたシード辺の集合をどのような形で蓄えておけば，任意に指定されたレベルを含む区間を持つすべてのシード辺を効率良く列挙できるかを考える．この問題はいわゆる区間検索と呼ばれるもので，計算幾何学の分野で初期の頃からよく研究されてきた問題である．区間検索には，区間の集合が静的な場合と，更新を許す動的な場合が考えられるが，本研究の場合には，区間表現されたシード辺集合をすべて求めた後の処理であるから，データの更新を許さない静的な場合に対応する．このような場合に対しては，Chazelle[7]によって提案された方法が最も有効であろう．

Chazelleの方法では，画像のレベルを0から $L-1$ までとしたとき，以下に述べるように区間 $[0, L-1]$ を部分区間 $I_1, I_2, \ldots, I_p$ に分割する．ただし，以下では区間表現されたシード辺の集合を $T = \{(k_i, a_i, b_i)\}$（ただし，k_i は辺の番号で，$[a_i, b_i]$ はレベルの区間）として表す．

各区間 I_j に対して，この区間と共通部分を持つ，すなわち，$I_j \cap [a_i, b_i] \neq \emptyset$ であるようなシード辺の集合を区間 I_j に対応するウィンドウと呼び，W_j という記号で表す．また，あるレベル x に対して，このレベルを含む区間を持つシード辺の集合，すなわち，$\{(k_i, a_i, b_i) : a_i \leq x \leq b_i\}$ を $S(x)$ という記号で表す．このとき，すべての部分区間 I_j について，

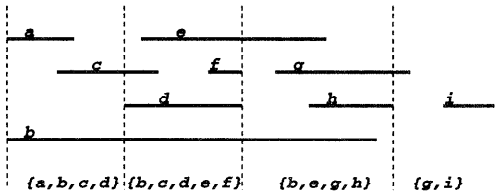

図3 9つの区間のウィンドウへの分割（$\delta = 2$ の場合）
Fig. 3 Partition of nine intervals into windows (for $\delta = 2$).

条件： 任意の $x \in I_j$ に対して，$S(x) \subseteq W_j$ かつ
$0 < |W_j| \leq \delta \max(1, |S(x)|)$
が成り立つように部分区間 $I_0, I_1, \ldots, I_p$ を定める．ただし，δ は，1より大きい整数（定数）のパラメータである．実際にこのような分割を求めるのはかなり簡単である．詳しくは，Chazelleの原著論文[7]を参照されたい．**図3** に9つの区間 $a, b, \ldots, i$ を $\delta = 2$ として分割する簡単な例を示す．この例では，最初の区間 I_0 に対応するウィンドウ W_0 は a と b を含む．x が区間 e の左端より左にあれば，$|S(x)| \geq 2$ であるから，条件より，W_0 のサイズはたかだか4でなければならない．したがって，$W_0 = \{a, b, c, d\}$ と定まる．次に $|S(x)|$ の値が小さくなるのは区間 f と g の間に x があるときで，そのとき $|S(x)| = 1$ である．したがって，ここに次の区間の境界がくる．このようにして，順次ウィンドウを決定していくことができる．

さて，このような分割ができると，質問のレベル x を含む部分区間 I_j は $O(\log p) = O(\log n)$ 時間で求めることができる．部分区間 I_j を同定できれば，後は，対応するウィンドウ W_j の要素を順に調べるだけでよい．質問レベル x を含む区間を持つシード辺の集合は $S(x)$ であるが，ウィンドウ W_j の要素数は $S(x)$ の要素数のたかだか δ 倍以内であるので，ウィンドウ内の探索の無駄はたかだか必要なコストの定数倍以内であることが保証されている．すなわち，$S(x)$ の要素を列挙するのに必要な時間は，$O(|S(x)| + \log n)$ である．

先にも述べたように，等高線データを正直に蓄えるとレベル数と画素数の積に比例する時間とメモリーが必要である．しかしながら，すべての等高線データが一度に必要になることは少ない．そこで，任意にレベルが指定されたときに，対応する等高線をなるべく少ないオーバヘッドで求めたい．上に述べた方法を用いると，画像データと同程度のメモリー量を使うだけで，任意に指定されたレベルの等高線を，等高線の個数の対数に比例する時間程度のオーバヘッドと出力に比例

表1 実験結果
Table 1 Experimental results.

画像の名称	大きさ（画素数）	境界木に基づく方法 シード辺集合のサイズ			NW法 シード辺集合のサイズ		
		赤	青	緑	赤	青	緑
ボートレート	512×640	78977	75518	77000	163885	157489	163596
カフェテリア	512×640	61832	61617	61025	155388	155281	158369
果物籠	640×512	60225	57714	57640	147638	149348	150640
食器とナイフ	640×512	61118	58314	62294	142961	137171	144102
自転車	512×640	63290	58599	62529	161497	151858	154445
蘭	640×512	59482	58677	50249	131605	128496	109585

する時間だけで求めることができる．

4.4 最小シード辺集合を求める方法

NWコーナーを列挙してシード辺の集合を求める方法では，1つの等高線に多数のシード辺が含まれる可能性があるので，メモリー効率の面からは望ましくない．そこで，冗長性を取り除くことが望まれる．もちろん，すべての等高線をたどってみれば，冗長なシード辺を取り除くことも可能であろうが，すべての等高線をたどるのに時間がかかり，しかも冗長な格子辺を取り除いて最小の集合を得ることも簡単ではない．

de Berg ら[8]が地形図における点位置決定に用いた**境界木（contour tree）**を用いると最小のシード辺集合を画素数 n の2乗に比例する時間で求めることができる．さらに，彼らは同文献で $O(n\log n)$ 時間の近似アルゴリズムも提案している．実際，我々も $O(n\log n)$ 時間で境界木を求める過程で同時にシード辺集合を求めるより簡単な近似アルゴリズムをプログラム化した．NW法との違いは，得られたシード辺集合に冗長性がないことである．すなわち，NW法の場合には1つの等高線が複数のシード辺を含むことがありうるので，同じ等高線を重複して通らないように，等高線をたどるときに各格子辺にマークをつける必要がある．しかしながら，上記の近似アルゴリズムで得られたシード辺は，各等高線に1つずつしか含まれないのでマークビットは不必要である．このように，$O(n)$ 時間で実行できるNW法ではシード辺集合が冗長であるのに対して，$O(n\log n)$ 時間を許すと，冗長性のないシード辺集合を求めることができる．さらに $O(n^2)$ 時間と記憶スペースを許すと，最小サイズでしかも冗長性のないシード辺集合を求めることができる．前2者の方法をプログラム化して様々な画像に対して実験を行ったが，その結果からすると，性能（シード辺集合のサイズ）の面においては2者に極端な差はなかった．

5. 計算機実験

前章ではシード辺集合を求めるためのアルゴリズムを2通り説明した．画素数の2乗の時間と記憶スペースを許すと最小のシード辺集合を求めることができるが，メモリー上の制約からこの方法を実行することは困難である．そこで，本文で述べた2通りの近似解法（4.2節で述べたNW法と，4.4節の境界木に基づく方法）のみをプログラム化し，印刷用の標準画像（日本規格協会画像処理技術標準化委員会監修の高精細カラーディジタル標準画像データ）を縦横とも1/4に縮小した画像について計算機実験を行った．その結果を**表1**に示す．最初の方法は境界木に基づく方法で，各等高線につき，正確に1本のシード辺を与えるという意味で冗長性のないシード辺集合を求めるものである．もう1つの方法は，NWコーナーを求めるもので，高速ではあるが，得られるシード辺集合は上記の意味において冗長である．実験の結果をまとめたのが表1である．実験には，PentiumMMX（166 MHz）を積んだパソコンを用いた．

表には示していないが，境界木に基づく $O(n\log n)$ 時間の方法はいずれも約2～3秒を要したが，$O(n)$ 時間のNW法は非常に高速で，いずれも1秒未満であり，計測できなかった．一方，シード辺のサイズに関しては後者が前者の約2倍であった．

6. 等高線表現の応用

本章では，画像の等高線表現の応用例をいくつか紹介する．これらの例は，いずれも従来の画像行列による表現では困難であるが，等高線に基づいた幾何学的表現の下では問題の自然な定式化，および特徴づけが可能になるものである．

6.1 キズ領域の修復

宣伝用の写真では微細なキズも取り除くことが要求される．また，風景画から電柱を取り除いた後の景観を求めたいというような場合もある．このような場合

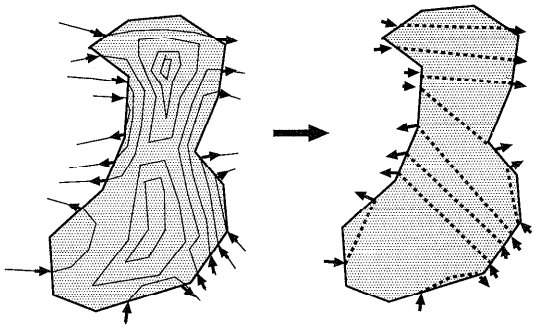

図4 キズ領域によって非連結になった等高線をつなぎ直す
Fig. 4 Reconnecting contour lines disconnected by a flaw region.

には，取り除くべき領域をキズ領域として指定し，その部分にコピーブラッシと呼ばれる機能を用いて画像中の似たような部分の色を移植する．このような処理は人手で行われることが多いが，意外に手間のかかる処理であり，従来から自動化が求められてきた．

画像が等高線表現されている場合，キズ領域を指定するということは，キズ領域を通過する等高線を（消しゴムなどで）消去することに対応している．キズ領域に含まれる等高線が消去されると，この領域に全体が含まれていた等高線は完全に消滅するが，一部がキズ領域を通過していた等高線は非連結になる．ここで，これらの非連結になった等高線を“自然に”つなぎ直すことができれば，キズ領域を目立たないようにすることができるであろう．図4はこれを模式的に示したものである．

さて，キズ領域の境界には非連結になった等高線の端点が存在するが，もちろん連結するときには，同じ高さ（レベル）に対応する端点どうしを連結しなければならない．その意味ではVLSIやプリント基板における配線問題に似ている．ただ，我々の場合には，等高線が方向づけられているので，端点にもキズ領域内部に向かう方向を持つものと，逆の方向を持つものがあるので，方向が合うように連結しなければならない．

自然な等高線のつなぎ方がどのようなものかは十分に分かっていないが，1つの候補として，等高線の包含関係を保ちながら，連結するのに用いた曲線の長さの総和を最小にするという基準が考えられる．そのような意味での最適な連結方法を求める問題は，動的計画法により領域のサイズ（面積）と端点の個数の和に関する多項式時間で求めることができる．実際にプログラム化して実験を行ったが，キズ領域があまり大きくなければ妥当な結果が得られている．方法と実験結果の詳細については別の機会に報告したい．

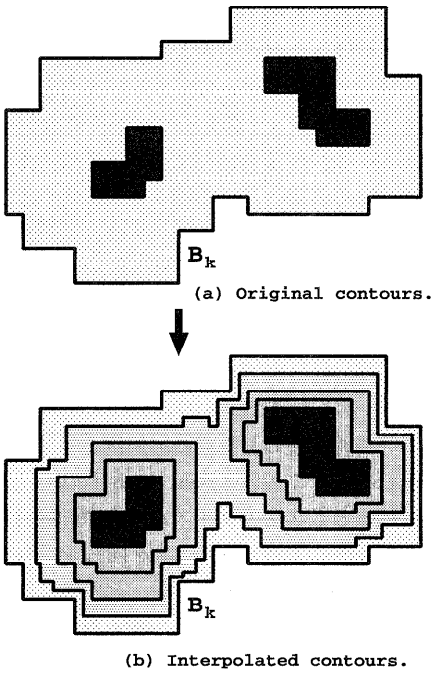

図5 2つの連続する等高線に囲まれた領域の等距離線による分割
Fig. 5 Partition of an region bounded by two consecutive contour lines.

6.2 階調補間による画質改善

入力画像の階調補間によって画質を改善することは画像処理の重要な目標の1つである．したがって，画像処理の分野で様々な方法が提案されてきたが，局所オペレータを用いた場合には画像のぼけをおさえるのが難しく，フーリエ変換などの方法を用いた場合には独特の模様の混入を避けるのが困難である．この問題を等高線表現された画像について考えると，階調補間によって画質を改善する問題は，連続する等高線の間を自然に補間する問題として特徴づけられる．たとえば，連続する等高線によって囲まれた領域を自然に2等分するには，両方の等高線からの距離が等しい曲線によって分割するのが自然である．この作業を繰り返すことにより，連続する等高線によって囲まれた領域を任意の個数に等しく分割することができる（図5参照）．

より一般的には，2つの等高線によって囲まれた領域の任意の点 p について，その点から外側の等高線上で点 p に最も近い点までの距離を $d_{out}(p)$ とし，その点から内側の等高線上で点 p に最も近い点までの距離を $d_{in}(p)$ とするとき，ちょうど両等高線の中央を通る曲線は，$d_{out}(p) = d_{in}(p)$ を満たす点集合として特徴づけることができる．また，両等高線の間を $1:r$ に内分する曲線は，

$\{p : d_{out}(p)/d_{in}(p) = r\}$

として特徴づけることができる．このような点集合は，一般に計算幾何学で（線分の）ボロノイ図と呼ばれるものであるが，ボロノイ図を正確に計算できなくても，各画素がどの2つの等高線に挟まれる領域にあるかが分かればよいだけなので，簡易的な方法が可能である．

この方法の最大の特徴は，いつでも元の情報が残っているという点である．したがって，いつでも原状に復帰できるという利点がある．また局所オペレータをいっさい用いないので，画像にぼけが生じることは少ない．

6.3 滑らかさを保つ画像の拡大

画像の等高線表現の3番目の応用例として，画像の拡大処理をあげることができる．あるサイズの画像を，たとえば縦横とも2倍のサイズに拡大する場合を考えよう．最も単純な方法は，1画素を同じレベルを持つ4画素で置き換えることであるが，その場合には斜め方向のエッジにジャギが目立ち，醜い画像となってしまう．コンピュータグラフィックスでは，この問題はアンチエリアシング問題と呼ばれ，様々な方法が提案されている．

この問題も我々の等高線表現では非常に分かりやすく特徴づけることができる．画像を拡大するということは等高線を拡大することに対応する．したがって，等高線をもともと滑らかな曲線で近似しておけば，拡大のみならず，回転などを含む幾何学的変換を施しても滑らかさは失われず，後処理としてのアンチエリアシングも必要ではない．

7. おわりに

本文で提案した画像の等高線表現は従来の画像行列による表現とまったく異なり，画像の全体的な構造を表現できるという点において優れている．画像行列表現では局所オペレータを用いることが多いが，局所オペレータでは画像の構造を反映した処理は困難である．また，本文では等高線表現に基づく新たな画像処理の例について述べたが，他の応用についても検討中である．等高線表現の最大の利点は，画像行列では単なる画素の集まりであった画像を幾何学的対象物としてとらえることができるので，画像の構造を反映したグローバルな処理が可能となると期待できることであろう．

謝辞 本研究の基本的なアイディアは大日本スクリーン製造の三塚郁夫氏によるものである．同氏には様々な有益な示唆をいただいた．ここに記して感謝する．また，計算機実験で多大な協力をいただいた同社の中井氏に感謝する．さらに，大阪電気通信大学大学院博士前期課程の修士論文の一環としてプログラムの作成から計算機実験に至るまで尽力してもらった平間氏にも感謝する．

参考文献

1) Asano, T. and Kimura, S.: Contour Representaiton of an Image with Applications, *Proc. SPIE's International Symposium on Vision Geometry IV*, San Diego, pp.14–22 (July 1995).
2) Asano, T., Katoh, N., and Tokuyama, T.: A unified scheme for detecting fundamental curves in binary edge images, *Proc. 2nd European Symp. on Algorithms*, Utrecht, pp.215–226 (1994).
3) Brandstädt, A., Chepoi, V.D. and Dragan, F.F.: The algorithmic use of hypertree structure and maximum neighborhood ordering, *Proc. 20th International Workshop Graph Theoretic Concepts in Computer Science* (*WG'94*), Lecture Notes in Computer Science, Vol.903, pp.65–80, Springer-Verlag, Berlin (1995).
4) Burnikel, C., Mehlhorn, K. and Schirra, S.: How to compute the Voronoi diagram of line segments, *Proc. 2nd European Symp. on Algorithms*, Utrecht, pp.227–239 (1994).
5) Breu, H., Gil, J., Kirkpatrick, D. and Werman, M.: Linear time Euclidean distance transform algorithms, *IEEE Trans. Pattern Analysis and Machine Intell.*, Vol.17, No.5, pp.529–533 (1995).
6) Capson, D.W.: An Improved Algorithms for the Sequential Extraction of Boundaries from a Raster Scan, *Computer Vision, Graphics, and Image Processing*, Vol.28, pp.109–125 (1984).
7) Chazelle, B.: Filtering Search: A New Approach to Query-Answering, *Proc. 24th IEEE Annual Symp. on Foundations of Computer Science*, Tucson, Arizona, pp.122–132 (1983).
8) de Berg, M. and van Kreveld, M.: Trekking in the alps without freezing or getting tired, *Proc. 1st Annual European Symp. on Algorithms* (*ESA'93*), Lecture Notes in Computer Science, Vol.726, pp.121–132, Springer-Verlag (1993).
9) Hirata, T. and Katoh, T.: An algorithm for Euclidean distance transformation, SIGAL Technical Report of IPS of Japan, 94-AL-41-4, pp.25–31 (Sep. 1994).
10) Kweon, I.S. and Kanade, T.: Extracting Topographic Terrain Features from Elevation Maps, *CVGIP: Image Understanding*, Vol.59, No.2, pp.171–182 (1994).

11) Preparata, F.P. and Shamos, M.I.: *Comoutational Geometry: An Introduction*, Springer-Verlag, New York (1985).
12) Sugihara, K.：画素によらない画像処理技術の開発，平成7-8年度文部省科学研究費研究成果報告書 (1997).
13) van Kreveld, M., van Oostrum, R., Bajaj, C., Pascucci, V. and Shikore, D.: Contour Trees and Small Seed Sets for Isosurface Traversal, *Proc. ACM Symp. on Computational Geometry*, Nice, pp.212–219 (1997).
14) Wilkins, L.C. and Wintz, P.A.: A contour tracing algorithm for data compression of two dimensional data, Purdue University, School of Engineering, Report TR-EE-69-3 (Jan. 1969).
15) Glassner, A.S. (Ed.): *Graphics Gems*, Academic Press (1990).
16) Arvo, J. (Ed.): *Graphics Gems II*, Academic Press (1991).
17) Kirk, D. (Ed.): *Graphics Gems III*, Academic Press (1992).
18) Heckbert, P.S. (Ed.): *Graphics Gems IV*, Academic Press (1994).
19) Paeth, A.W. (Ed.): *Graphics Gems V*, Academic Press (1995).

(平成10年 6 月22日受付)
(平成10年10月 2 日採録)

浅野 哲夫（正会員）

1949年生．1977年大阪大学大学院基礎工学研究科情報工学専攻博士課程修了．同年大阪電気通信大学工学部応用電子工学科講師．1997年4月より北陸先端科学技術大学院大学情報科学研究科教授．計算幾何学の理論と応用に関する研究に従事．工学博士．著書に「計算幾何学」（朝倉書店）等．IEEE，ACM，SIAM，電子情報通信学会，応用数理学会等の会員．平成1993～1994年本学会アルゴリズム研究会主査．

木村 宗市

1967年生．1990年大阪電気通信大学工学部応用電子工学科卒業．同年，大日本スクリーン製造（株）入社．技術研究所を経て現在同社グラフィックアーツ事業本部開発部勤務．主に図形処理および色彩画像処理の研究開発に従事．

嶋津 茂昭

1959年生．1984年静岡大学大学院電気工学専攻修士課程修了．工学修士．同年大日本スクリーン製造（株）に入社．技術研究所を経て現在同社グラフィックアーツ事業本部開発部勤務．主に図形処理および色彩画像処理の研究開発に従事．現在製版用画像処理ソフトウェアの開発に従事．色彩学会会員．