画像の境界値表現を効率よく求める算法

1 J－4

平間尊[1]、浅野哲夫[2]
[1] 大阪電気通信大学大学院工学研究科情報工学専攻
[2] 大阪電気通信大学情報工学部

画像処理の分野では、多階調のディジタル画像を表現するのに、各要素が各画素の階調値を表す２次元配列の形式を用いるのが一般的である。これに対して、画像の境界値表現とは、各画素の階調値を対応する地点の標高と見なして、画像を階調値の等高線で表現するものである。階調値が整数で表される場合、階調値を表す各整数 k に対して、階調値が k 以上の領域を等高線として求めれば、それらの等高線の集合によって画像を表現することができる（あるいは、等高線の集合から画像を再構成することができる）。本文では、このような画像の等高線表現を効率よく求めるアルゴリズムを提案する。

1　諸定義

まず、表現を簡潔にするために、画像を１次元配列で表現する。すなわち、画像を表す $v \times h$ のサイズの２次元配列 $G = (g_{ij}),\ (0 \leq i \leq v-1, 0 \leq j \leq h-1)$ が与えられたとき、ラスタースキャンの順序に従って１次元配列に変換する。すなわち、i 行 j 列の要素 g_{ij} を、１次元配列 $G[\ ]$ の $i \times h + j$ 番目の要素 $G[i \times h + j]$ に対応させる。

このように各画素に通し番号をつけたとき、番号 k の画素の上下左右の画素の番号は、それぞれ、$k-h, k+h, k-1, k+1$ で与えられる。

次に、隣り合う画素を隔てる線分（辺）にも通し番号をつけて管理する。たとえば、番号 k の画素の下と右の辺に、それぞれ、番号 $2k$ と $2k+1$ をつける。このようにすると、番号 k の画素の上の辺には $2(k-h)$、左の辺には $2k-1$ という番号がつけられることになる。以下では、番号 j の辺を e_j と表すことにする。

また、階調値が k 以上の領域の辺には、その領域の内部が辺の右側にあるように方向をつけるものとする。したがって、山の領域では時計回りに、谷の領域では逆に反時計回りになる。

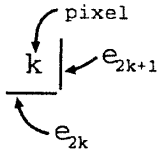

図 1: 辺の番号付け

2　等高線を求める算法

ここでは、各階調値に対する等高線を求めるアルゴリズムについて考える。最も単純な方法は、各階調値 k に対して、まず階調値が k 以上の画素にラベルをつけ、連結領域を求めた後で、連結領域の境界を追跡するというものである。画素の数を $n = v \times h$、階調数を L とすると、この素朴な方法では $O(nL+K)$ の時間が必要になる。ただし、K は等高線の長さ（辺の数）の総和である。

連結領域を求めなくても、１回のラスタースキャンで領域の境界を求める効率の良いアルゴリズムも知られているが [1]、この方法でも $O(nL+K)$ の時間がかかる。これに対して、本文で提案するのは、最適な $\Theta(n+L+K)$ の時間ですべての等高線を求めるというものである。以下にその算法を示す。

[等高線を求める算法]

この算法では、基本的に各階調値 $k = 1, 2, \ldots$ に対して k 以上の階調をもつ領域の境界を追跡していく。このとき、無駄な探索がなければオーダーとして最適な算法となる。そのために、階調値が k 以上の領域 R_k の境界に現れる辺を、k 番目のリスト list[k] に持っておく。

最初は、すべての辺を調べて、その辺がどの階調値から領域の境界として現れるかを求める。すなわち、階調値 $p, q (p \leq q)$ の画素の間の辺 e_i は、$p = q$

Efficient Algorithm for Computing the Contour Representation of an Image, T. Hirama and T. Asano, Osaka Electro - Communication Univ.,18-8 Hatucho, Neyagawa, Osaka, 572, Japan

でない限り、階調値$p+1, p+2, \ldots, q$の領域の境界に現れる。したがって、この辺e_iが階調値qまで境界に現れることを示す為に、辺の番号と最後の階調値のペア(i, q)を、最初にこの辺が境界に現れる階調値$p+1$のリスト list$[p+1]$に蓄えておく。

さて、階調値kの処理では、リスト list$[k]$に蓄えられている辺を取り出して、そこから境界を辿って行く。このとき、境界の進む方向は各地点で定数時間で求めることができる。境界を辿るときに、辺の情報も同時に管理する。すなわち、辺e_iを辿るとき、リスト内での対応するペア(i, q)を求め、$q > k$ならば、この辺は次の階調値$k+1$でも現れるから、リスト list$[k+1]$に挿入する。

上記のリストを双方向連結リストで実現し、さらに、各辺e_iに対応するペア(i, q)を蓄えているレコードへのポインタを配列で蓄えておけば、リストを管理するための上記の操作 (find, insert) は定数時間で実現できる。境界の追跡において間違った方向に進んでしまってバックトラックをするというような事態は起こり得ないので、計算時間は、すべての境界線を追跡するための時間だけで済むことになる。

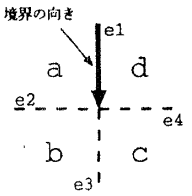

図 2: ある境界の例

図2は、ある境界追跡の例を示したものである。a, b, c, d は画素の階調値、e1〜e4 は辺の番号を示す。追跡している階調値をkとすると、

(i)$b \geq k, c \geq k$の場合、境界は右 ($e4$) へ進む。

(ii)$b \geq k, c \leq k$の場合、境界は下 ($e3$) へ進む。

(iii)$b < k, c < k$の場合、境界は左 ($e2$) へ進む。

(ix)$b < k, c \geq k$の場合、境界が上 ($e1$) から進んできた場合は左 ($e2$) へ下から進んできた場合は右 ($e4$) へ進む。そして、$k = l$ならば、配列 list$[k]$ からeを取り除き、$k < l$ならば配列 list$[k+1]$ にeを挿入する。

(4) まだ list$[i]$ に要素が残っていれば、(3) の作業を繰り返し、配列 list$[i]$ 空になるまで繰り返す。

図3に1以上の境界を求めた簡単な例を示す。

上記の例は、5×5 の大きさで、階調値は6段階

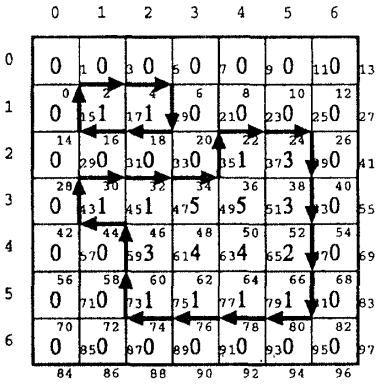

図 3: 1以上の境界の探索

であり、1以上の辺を追跡したものである。図3の中の大きな数字は階調値を、小さい数字は辺の番号を示し、矢印で辺の向きを示したものである。

3 コンパクトな表現方法

等高線表現を用いると、画像の階調補間や拡大縮小など様々な処理を行う事ができるが、これらの応用では等高線データをすべて蓄えておかないといけないわけではない。実際、等高線データは画像データの20倍ほどのメモリーを消費することがあるので、等高線データをそのままの形で蓄えるのは得策でない。追跡すべき等高線に属する1つの辺が指定されれば、元の画像行列上で無駄なく境界追跡ができることを考慮すると、各等高線について1辺だけを保持しておけばよい。このように、画像行列と各等高線について1つの辺を管理することにすると、ほぼ画像行列の2倍以内のメモリーだけで十分であることがわかる。

4 結論

本文では、画像を階調値の等高線で表現する境界値表現を効率良く求めるアルゴリズムを提案した。今後は、この表現を用いた画像処理について研究を進めて行きたい。

参考文献

[1] D. W. Capson: "An Improved Algorithms for the Sequential Extraction of Boundaries from a Raster Scan," *Computer Vision, Graphics, and Image Processing*, 28, pp.109-125 (1984).