2P−6

構成要素のランレングス表現による数理形態学演算の高速化

川田　雄一　　東海林　健二
宇都宮大学

1．はじめに

数理形態学演算を画像処理に応用する際の問題点として計算コストが大きいという問題がある。文献(1)では、この問題点を解決するために、演算の対象となる2値画像をより小さな画像のミンコウスキー和に分解する方法を示した。しかし、この分解が存在する画像のクラスは限られているという欠点がある。本研究では、ランレングス表現を用いることにより、任意の2値画像に適応可能な高速化の手法を提案する。

2．数理形態学演算

本研究では、2値画像に対する数理形態額演算を用いる。また、用いる演算は2項演算で表され、その第二項を構成要素(structuring element)と呼ぶ。次に、以下の2つのよう基本演算を定義する。

1）ミンコウスキー和：図1

図形Aと図形Bにおいて、図形Aを図形Bの全ての要素で平行移動させて、それらの和をとったもの。

$$A \oplus B = \bigcup_{y \in B} (A + y) \qquad (1)$$

$$= \{x + y : x \in A、y \in B\}$$

ここで、(A＋y)は図形Aと図形Bの要素の位置ベクトルyの和、つまり図形Aをベクトルy移動する事を意味する。

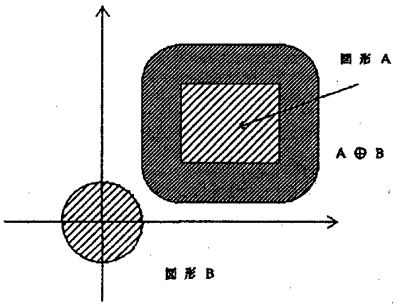

図1　ミンコウスキー和

2）ミンコウスキー差：

図形Aと図形Bにおいて、図形Aを図形Bの全ての要素で平行移動させて、それらの積をとったもの。

$$A \ominus B = \bigcap_{y \in B} (A + y) \qquad (2)$$

$$= \{x : B_x{}^t \subset A\}$$

以上の二つの定義から分かるようにこの演算のコストは構成要素Bの面積によることになる。

3．ランレングス表現を用いる手法

1）ランレングス表現

図形を長さr_nの線分の和によって表現することをランレングス表現という。但しここで、$r_n = 1$とは画素1つのことであり、$r_n = m$とはm個の画素を結ぶ線分の長さである。

ここで$r_n = m$の線分をレベルmと呼ぶことにする。そして各線分の左端の位置ベクトルをxとしてL(r_n)はレベルr_nの直線を表すとすると、構成要素Bは、

$$B = \bigcup_{n=1}^{N} (L(r_n) + x_n)$$

と表現される。

2）ランレングス表現による⊕・⊖演算の分解

上のランレングス表現を用いてミンコウスキー演算における構成要素を分解すると以下のようになる。

$$① \quad A \oplus B = A \oplus \left(\bigcup_{n=1}^{N} (L(r_n) + x_n) \right)$$

$$= \bigcup_{n=1}^{N} (A \oplus (L(r_n) + x_n))$$

$$= \bigcup_{n=1}^{N} ((A \oplus L(r_n)) + x_n)$$

$$② \quad A \ominus B = A \ominus \left(\bigcap_{n=1}^{N} (L(r_n) + x_n) \right)$$

$$= \bigcap_{n=1}^{N} (A \ominus (L(r_n) + x_n))$$

$$= \bigcap_{n=1}^{N} ((A \ominus L(r_n)) + x_n)$$

A High-speed Algorithm for Morphological Operation Using Run-length Expressins of Structuring Elements

Yuichi KAWADA , Kenji SHOJI
UTSUNOMIYA University

以上のことより、ミンコウスキー演算は、(A⊕L(r n))または、(A⊖L(r n))の演算を行い、それをx n移動させることにより計算できるということになる。

4．アルゴリズム

以下に、A⊕Bの演算アルゴリズムを示す。

① 図形Bのランレングス表現を求める。

ここで求められた各要素をβ_k＝{x k、r k}とする。

② ランレングス表現によって生成された各要素をラン長r nの昇順にソートしたβ_nの系列S(β_n)を求める。ここで、

S(β_n)＝β_0、β_1、β_2、・・・、β_N

となる。ただしここで

β_i＝(x i、r i)とすると、r i-1＜＝r iである。

③ 演算の開始

ここで実際に演算を行っていくわけだが、(A⊕L(r n))の演算を行う際、S(β_n)が昇順になっていることから、

(A⊕(L(r)))=(A⊕(L(r n-1)))∪(A⊕(L(r n−r n-1)))

で計算を行う。これによって、(A⊕L(r n))の計算を行うときに(A⊕L(r n-1))を用いることで和をとる回数の削減ができる。またここで、(r n／2)＞r n-1のときには、

$$(A\oplus(L(r_{n-1})))\cup((A\oplus(L(r_{n-1})))+\overline{(r_{n-1},0)})$$

を行うことにより、レベルをr n-1からr＝2＊r n-1にする。ここでも和をとる回数の削減ができる。これを((r n／2)＜r)になるまで行い、その後

$$(A\oplus(L(r_n)))=(A\oplus(L(r)))\cup((A\oplus(L(r)))+\overline{(r_{n-r},0)})$$

として(A⊕L(r n))の計算を行う。

④ 以上β_nを1、2、・・・Nまで同様の操作を繰り返して演算を行っていく。

またA⊖Bについては3・2）で示したことから、上のアルゴリズムで⊕が⊖になり、∪が∩になることがわかる。

5．実 験

1） 計算コストの評価法

通常、ミンコウスキー演算を行う際の基本的な操作は、図形を移動させて、論理演算を行うことと考えることが出来る。そこで、本手法を評価する手段として、上の2つの操作を1ステップとし、計算コストの評価のための実験を行うことにする。

2） 実験

ここで本手法について考えてみると、本手法は一概に図形の面積に影響されるわけでなく、図形の形状にも影響されることになると思われる。そこで、複雑な図形、普通の図形、簡単な図形の3種類についてそれぞれ縮尺を6，5，4，3，2，1，0.8，0.6，0.4倍の9段階に分けて本手法ではどの程度計算コストがかかるかを実験してみる。実験は以下の3つの図形について行い、図2が複雑な図形で、図3が普通の図形、図4が簡単な図形とする。

そしてそれぞれの形状・縮尺の図形について、本手法を実行した。その結果は図5のグラフに表わす。グラフの横軸は図形の面積で、縦軸は本手法による計算コストである。またグラフ中の式(1)の定義通りの手法による計算コストであり、面積とコストが一致する。

図2．複雑図形(◇)

図3．普通図形(□)

図4．簡単図形(△)

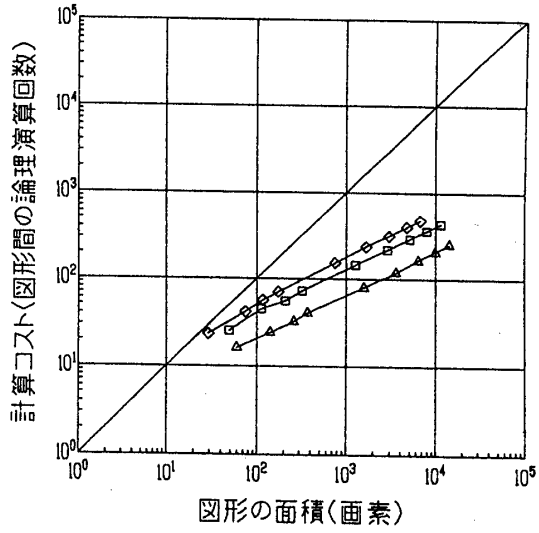

図5．計算コストと図形の面積のグラフ

6．考 察

上のグラフから計算コストと図形の面積の関係を求めると、傾きはほぼ1／2なので、

コスト∝√面積

となり、同じ形状であれば、本手法は面積の平方根にほぼ比例する。また、3つの図形のうち図4の簡単な図形が常に一番下にあるということは、本手法の計算コストが多少の面積の違いよりは、図形の形状に左右されやすいことを表わしていると思われる。

参考文献

(1) X.Zhuang and R.M.Haralick: "Morphological structuring element decomposition", Computer Vision, Graphics, and Image Processing, 35, pp.370-382(1986).