効率の良いモルフォロジー演算が可能なフィルタ形状について

櫻井敦史[†,☆] 平田富夫[†]

モルフォロジー演算は画像の特徴抽出やノイズ除去など様々な画像処理に用いられる基本的処理である．2 値画像入力に対するその時間計算量は，入力画像のサイズを $n \times n$，フィルタのサイズを $r \times r$ とすると $O(n^2r^2)$ となり，処理時間がフィルタサイズに大きく依存する．しかし距離変換を用いることで処理時間がフィルタサイズに依存しないモルフォロジー演算を行うことが可能である．本研究ではフィルタ形状のあるクラスに対しては，$O(n^2)$ 時間でモルフォロジー演算ができることを示す．このクラスに入るフィルタ形状の例をあげると，円，長円形，正三角形，長方形，台形などであり，画像処理で用いられるフィルタのほとんどが含まれる．

On a Class of Efficiently Computable Morphological Filters

Atsushi Sakurai[†,☆] and Tomio Hirata[†]

Mathematical morphology is used for feature extraction and noise elimination in image processing. Morphological operation for a binary image of size $n \times n$ with a filter of size $r \times r$ is performed in $O(n^2r^2)$ time, and thus the computation time depends heavily on the filter size. By using distance transformation, morphological operation can be done in time independent of the filter size. In this paper, we show that morphological operation can be done in $O(n^2)$ time for some class of filter shapes. This class contains most of filter shapes which appear in image processing, such as circle, rectangle, equilateral triangle, trapezoid, etc.

1. はじめに

モルフォロジーとは一般には形態学を意味する言葉であるが，画像処理の分野では数理形態学（mathematical morphology）のことを指す．1960 年代後半に G. Matheron や J. Serra らによって提起され，当初は鉱石の顕微鏡写真の解析手段として応用された．その後，1 つの学問体系として発達をとげた．一般的には多次元空間における集合論として展開されるものであるが，実際に応用される場合は 2 次元画像を対象とすることが多い．特に 2 値画像に対するモルフォロジー演算は画像の特徴抽出やノイズ除去などを行う際に用いられる画像処理の 1 つである．サイズ $n \times n$（縦横がそれぞれ n 画素）の 2 次元ディジタル画像を入力とし，フィルタ画像のサイズを $r \times r$ とすると，モルフォロジー演算に要する時間計算量は $O(n^2r^2)$ となり，処理時間がフィルタサイズに大きく依存するという問題がある．これを解決する手段として，大きなフィルタを小さなフィルタに分割して繰り返し処理を行う方法[1),2)]やフィルタを 1 次元フィルタに分解して処理する方法[7)]が提案されている．しかし分割繰返し型フィルタリングの場合は小フィルタのサンプリング誤差が累積して結果に影響を及ぼしたり，1 次元分解型フィルタリングはフィルタ形状が対称でないと十分な効果を発揮しないことなどが指摘されている[3)]．これらとはまったく異なる手法として，距離変換アルゴリズムを用いてモルフォロジー演算を行うことができる．円や正方形といった形状のフィルタはそれぞれ Euclid 距離や chessboard 距離の単位円に相当する．これらの距離については効率の良い距離変換アルゴリズムが与えられており[4),5)]，それを利用して円や正方形のフィルタに関するモルフォロジー演算が可能である．しかしモルフォロジー演算のフィルタとしては，これら以外にも様々な形状が要求される．本研究では，多くの形状を含むフィルタのクラスを定義し，そのクラスに属するフィルタに関するモルフォロジー演算がすべて $O(n^2)$ 時間で計算可能であることを示す．このクラスは，円，長円形，正三角形，長方形，台形など，画像処理で用いられるフィルタのほとんどを含む．

† 名古屋大学工学研究科電子工学専攻
Graduate School of Engineering, Nagoya University
☆ 現在，CSK 総合研究所
Presently with CSK Research Institute

2. 準　　備

2.1 ディジタル画像

$\mathbf{I}$をサイズ $n \times n$ の 2 次元ディジタル画像とする．画素の位置は xy 整数座標系で表すものとし，x，y 座標がそれぞれ i，j の画素を (i,j) で表す．画素 (i,j) の濃度値（画素値）が I_{ij} のとき，この画像を $\mathbf{I}\{I_{ij}\}$ と表記する．このとき x 座標のインデックス i と y 座標のインデックス j の集合をそれぞれ z_x，z_y とする（$|z_x| = n$，$|z_y| = n$）．$\mathbf{I}\{I_{ij}\}$ が 2 値画像の場合は $I_{ij} \in \{0,1\}$ である．また，y 座標を $c1$ に固定した画素 $(x, c1)$ の集合のことを行 $c1$，x 座標を $c2$ に固定した画素 $(c2, y)$ の集合のことを列 $c2$ と呼ぶ．画素 (i,j) は点 (i,j) とも呼び，位置ベクトルとして考えることもある．

2.2 モルフォロジー演算

モルフォロジー演算は一般論としては N 次元空間における集合論として展開されるものである．しかし実際には 2 次元空間つまり画像が処理の対象となることがほとんどである．モルフォロジー演算の利用例としては画像のノイズ除去やエッジ検出，テクスチャ方向性検出，パターンスペクトルによる特徴抽出などがあげられる[6]．

ここでは 2 次元 2 値画像を対象とした場合のモルフォロジー演算についてその基本演算を定義する．

モルフォロジー演算には**ディレーション（dilation）**と**イロージョン（erosion）**という 2 つの基本演算があり，これら単体もしくはその組合せによって処理を行う．入力をサイズ $n \times n$ の 2 値画像 $\mathbf{I}\{I_{ij}\}$ とし，$F = \{(i,j) \mid I_{ij} = 1\}$ を前景，$B = \{(i,j) \mid I_{ij} = 0\}$ を背景と呼ぶ．フィルタはサイズ $r \times r$ の 2 値画像 $\mathbf{G}\{G_{ij}\}$ である．$G = \{(i,j) \mid G_{ij} = 1\}$ をフィルタ領域と呼び，これを単にフィルタと呼ぶこともある．本論文では簡単のためフィルタ画像の原点 $(0,0)$ はフィルタ領域 G に含まれるものとする．この仮定は後に述べる平行移動不変性のため一般性を失うものではない．2 値画像のモルフォロジー演算では，入力画像およびフィルタ画像のそれぞれの前景画素集合について演算を行った結果が出力画像の前景画素集合になると考える．ディレーション演算は $\mathbf{I}$ と $\mathbf{G}$ から $n \times n$ の画像 $\mathbf{D}\{D_{ij}\}$ を得る演算で，$\mathbf{I} \oplus \mathbf{G} = \mathbf{D}$ と表記し，次の式で定義される．

$$D_{ij} = \max_{(s,t)\in G, i-s\in z_x, j-t\in z_y} \{I_{i-s,j-t}\}$$

つまり $\mathbf{I} \oplus \mathbf{G}$ は F と G のミンコフスキー和に対応する．

イロージョン演算は $\mathbf{I}$ と $\mathbf{G}$ から $n \times n$ の画像 $\mathbf{E}\{E_{ij}\}$ を得る演算で，$\mathbf{I} \ominus \mathbf{G} = \mathbf{E}$ と表記し，次の式で定義される．

$$E_{ij} = \min_{(s,t)\in G, i+s\in z_x, j+t\in z_y} \{I_{i+s,j+t}\}$$

ディレーション処理は**ずらし重ね**または**膨張**とも呼ばれ，入力画像の前景を広げる効果がある．一方，イロージョン処理は**掻き取り**または**収縮**とも呼ばれ，入力画像の前景を狭める効果がある．なお，ディレーションおよびイロージョンでは以下の性質が成り立つ[6]．ただし，A，B，C は 2 値画像であり，$(A)_z$ は z を位置ベクトルとして，A の前景画素を z だけ平行移動したものを表す．また，$\cup$ と $\cap$ はそれぞれ前景の和集合，積集合である．

平行移動不変性　$A \oplus (B)_z = (A \oplus B)_z$ および $A \ominus (B)_z = (A \ominus B)_z$

これは原点座標の位置を平行移動したフィルタによるディレーションおよびイロージョンの出力は，平行移動する前のフィルタによるディレーションおよびイロージョンの出力を平行移動したものに等しいということである．

分配則　$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$ および $A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C)$

これは複数のフィルタの和集合によって表されるフィルタによるディレーションの出力はそれぞれのフィルタによるディレーションの出力の和集合に等しく，また，同様のフィルタによるイロージョンの出力はそれぞれのフィルタによるイロージョンの出力の積集合に等しいことを表す．

ディレーションとイロージョンを組み合わせた演算としては**オープニング（opening）**と**クロージング（closing）**などがある．オープニングは $\mathbf{I} \circ \mathbf{G} = (\mathbf{I} \ominus \mathbf{G}) \oplus \mathbf{G}$ と定義され，クロージングは $\mathbf{I} \bullet \mathbf{G} = (\mathbf{I} \oplus \mathbf{G}) \ominus \mathbf{G}$ と定義される．オープニング処理は前景を内側から平滑化する効果を持っており，前景の微小突起や孤立点などを消去する．また，クロージング処理は前景を外側から平滑化する効果があり，前景の微小凹部分や前景内の欠損スポットなどを埋め合わせる．

モルフォロジー演算を単純に定義どおりに計算すると，入力画像の各画素についてサイズ $r \times r$ のフィルタを重ね合わせて出力を計算する必要があるため，その時間計算量は $O(n^2 r^2)$ となる．

2.3 距 離 変 換

入力画像をサイズ $n \times n$ の 2 値画像 $\mathbf{I}\{I_{ij}\}$ とする．画素 $p_1 = (i_1, j_1)$ と画素 $p_2 = (i_2, j_2)$ の間の距離を $d(p_1, p_2)$ で表す．2 値画像 $\mathbf{I}$ の距離変換とは，各画素

(i,j) について，前景画素との間の距離の最小値を求めることであり，その出力 $\mathbf{T}\{T_{ij}\}$ は $n \times n$ の画像で，次のように定義される．

$$T_{ij} = \min_{s \in z_x, t \in z_y} \{d((i,j),(s,t)) \mid I_{st} = 1\}$$

画像処理の分野でよく使われる距離関数 $d(\cdot\,,\cdot)$ としては Euclid 距離，cityblock 距離，chessboard 距離，octagonal 距離などがある．

距離変換は単純に定義どおりに計算すると，入力画像サイズが $n \times n$ のとき，その時間計算量は $O(n^4)$ である．距離変換は図形の骨格抽出などに利用される．

2.4 距離変換アルゴリズム（UDT）

文献 5) によって，Euclid 距離を含む様々な距離関数について統一的手法により $O(n^2)$ 時間で距離変換を行うアルゴリズムが与えられている．以下ではこの距離変換アルゴリズムを UDT と呼ぶことにする．この節では UDT の概要について，Euclid 距離変換を例として簡単に紹介する．詳しくは文献 4)，5) を参照されたい．アルゴリズムは 2 段階の処理からなる．

【変換 1：列方向処理】入力画像の各列 t ごとに列方向のみを参照して，各画素について最も近い前景画素までの距離値を求める．すべての列についてこの処理を行った後に得られる画像を $\mathbf{C}\{c_{ij}\}(i \in z_x, j \in z_y)$ とする．ここで，

$$c_{ij} = \min_{0 \le p \le n-1} \{|j-p| \mid I_{ip} = 1\}$$

である．また列 t に前景画素が 1 つもない場合は $c_{tj} = \infty (j \in z_y)$ とする．このアルゴリズムは 1 列処理するのに $O(n)$ 時間でできるので，すべての列についての処理は $O(n^2)$ 時間でできる．

【変換 2：行方向処理】変換 1 の結果 $\mathbf{C}$ の各行 e ごとに行方向のみを参照して，各画素 (i,e) について次のようにして距離値 t_{ie} を求める．列 k 内で最も行 e に近い前景画素（つまり $c_{ke} = |e-p|$ となる画素）を (k,p) とする．画素 (k,p) と画素 (i,e) との間の距離の 2 乗は $(i-k)^2 + c_{ke}^2$ である．これを i についての関数として $\ell_k^e(i) = (i-k)^2 + c_{ke}^2$ と表し，この関数を行 e に関する列 k の距離グラフと呼ぶ．このとき，t_{ie} の 2 乗は次のようになる．

$$t_{ie}^2 = \min_{k \in z_x} \{\ell_k^e(i)\}$$

したがって，関数の集合 $L_n^e = \{\ell_k^e(x) \mid k \in z_x\}$ の下側包絡線 $L_{env}(L_n^e) = \{\ell_t^e(x) \mid \exists x_0\ \ell_t^e(x_0) = \min_{k \in z_x} \ell_k^e(x_0)\}$ を求めればよい．この処理はスタックを用いることによって $O(n)$ 時間で実行できる[4),5)]ので，すべての行についての処理は $O(n^2)$ 時間でできる．

よって UDT アルゴリズム全体は $O(n^2)$ 時間で処理が終了する．上の説明では Euclid 距離についてアルゴリズムを紹介したが，他の距離関数について距離変換を行う場合は $\ell_k^e(i)$ の形を変えてやればよい．具体的には，cityblock 距離なら $\ell_k^e(i) = |k-i| + c_{ke}$，chessboard 距離なら $\ell_k^e(i) = \max(|k-i|, c_{ke})$ などとすればよい．また，このアルゴリズムが適用可能な距離関数については，次の定理が得られている．

定理 1[5)] 距離が軸対称ノルムに基づいているならば，距離変換は $O(n^2)$ 時間で実行可能である．

距離がノルム☆に基づく，またはノルムで定義されるとは，点 p_1 から点 p_2 までの距離が，$d(p_1,p_2) = \|p_2 - p_1\|$ として与えられることとする．また，軸対称ノルムとは $\|(x,y)\| = \|(|x|,|y|)\|$ を満たすノルムである．つまり軸対称ノルムの単位円は x 軸，y 軸に対称な図形となる．

ノルムに基づく距離では 2 点間の距離はその相対位置関係から決定され，また，三角不等式が成立する．さらに軸対称ノルムに基づく距離の場合は，距離が x 座標，y 座標それぞれについて単調かつ対称である．距離が x 座標に単調かつ対称とは，$\forall y$ について，$|x| < |x'|$ ならば，$\|(|x|,|y|)\| \le \|(|x'|,|y|)\|$ となることである．同様に y 座標に単調かつ対称とは，$\forall x$ について，$|y| < |y'|$ ならば，$\|(|x|,|y|)\| \le \|(|x|,|y'|)\|$ となることである．

Euclid 距離，chessboard 距離，cityblock 距離，octagonal 距離はすべて軸対称ノルムで定義できる．このように UDT アルゴリズムは画像処理の分野でよく用いられるほとんどの距離関数について適用できる．

2.5 距離変換によるモルフォロジー演算

モルフォロジー演算に用いるフィルタの形状が円や正方形の場合には，距離変換を用いてモルフォロジー演算の結果が得られる．入力画像 $\mathbf{I}$ に対し半径 b の円形フィルタでディレーションを行うには，まず入力画像に対し Euclid 距離変換を施し，その出力 $\mathbf{T}$ の各要素 T_{ij} をしきい値 b によって 2 値化（b より大きい値なら 0，b 以下なら 1 に）すればよい．イロージョンの場合は，入力画像の画素値を反転させて距離変換を施し，その結果の各要素を 2 値化（b より大きい値なら 1，b 以下なら 0 に）すればよい．

つまりモルフォロジー演算のフィルタ形状が距離関

☆ ベクトル空間 X の各元 x に対して次の性質を持つ実数 $\|x\|$ が対応しているとき，$\|x\|$ を x のノルムという．i) $\|x\| \ge 0$, $\|x\| = 0$ iff $x = 0$. ii) $\|\alpha x\| = |\alpha|\|x\|$. iii) $\|x\| + \|y\| \ge \|x+y\|$.

数の単位円と相似である場合には，上のようにして距離変換を施した結果に対して2値化することでモルフォロジー演算を行うことができる．円形フィルタの場合はEuclid距離，正方形フィルタの場合はchessboard距離，ダイヤモンド型フィルタの場合はcityblock距離を用いて距離変換を施せばよい．これらの距離の場合は前節で紹介したように $O(n^2)$ 時間で距離変換を行うことができる．しきい値処理や画像反転処理は入力画像のサイズに比例する時間（$O(n^2)$）で行えるので，距離変換が効率良く実行できる場合には，処理時間がフィルタサイズに依存しないでモルフォロジー演算を実行できることになる．したがってこの方法は大きなサイズのフィルタの場合に特に有効である．

3. モルフォロジー演算の高速化

3.1 従 来 法

大きなフィルタを用いたときのモルフォロジー演算の処理時間短縮のために過去に以下のような方法が提案されている．ここではこれら従来法の処理方式およびその性質について紹介する．

3.1.1 1次元分解型フィルタリング

フィルタを複数の1次元フィルタに分解し，それぞれの処理結果からもとのフィルタ全体の演算結果を求める方法を1次元分解型フィルタリング[7]という．この原理に基づいたハードウェアの開発なども行われている[8]．1次元分解型フィルタリングでは，もとのフィルタを1次元の帯状フィルタに分解し，分割したそれぞれのフィルタについてモルフォロジー演算を行う．この演算結果を分解した方向と直行する1次元フィルタでモルフォロジー演算を行い，それぞれの結果を組み合わせることで，もとのフィルタ全体の演算結果を求めることができる．この方法の場合，単純に行うとモルフォロジー演算の計算量は削減されない．しかし分割したフィルタのなかに同じ長さの1次元フィルタが複数含まれる場合は，その1次元フィルタによる演算結果を記憶しておいて利用することで計算量を削減することが可能である．すなわち1次元分解型フィルタリングの計算量はフィルタ形状に依存する．たとえば辺が水平垂直な長方形フィルタならば，水平（または垂直）方向に分解した1次元フィルタの形状（長さ）はすべて等しくなるため，大幅な計算量削減が可能であるが，フィルタ形状によってはほとんど計算量が変化しないものもある．

3.1.2 分割繰返し型フィルタリング

大きなサイズのフィルタによるモルフォロジー演算を，より小さなサイズのフィルタによるモルフォロジー演算の繰返し処理で代用する方法を分割繰返し型フィルタリングという[1),2)]．もととなるフィルタを G とし，

$$G = g_1 \oplus g_2 \oplus \cdots \oplus g_k$$

の式を満たす，G よりもサイズの小さなフィルタ g_i が存在するならば，次の関係が成立する．

$$\begin{aligned} F \oplus G &= F \oplus (g_1 \oplus g_2 \oplus \cdots \oplus g_k) \\ &= (\cdots((F \oplus g_1) \oplus g_2) \oplus \cdots \oplus g_k) \end{aligned}$$

よってフィルタ G によるディレーション演算を行う代わりに，フィルタ $g_1, \ldots, g_k$ のディレーションを順に行うことで同様の結果が得られる．イロージョンについても同様の処理が可能である．フィルタ G のサイズと比べると個々の g_i のサイズが小さくなっており，計算量を削減することができる．分割した小フィルタのサイズがすべて等しい場合，その計算量は分割したフィルタの個数を k とすると $1/k$ になる．しかしディジタル空間においてはフィルタ分割の式が厳密には成立しないため，小フィルタをかけるたびに誤差が累積する．そのため分割繰返し型フィルタリングではうまくフィルタの分割方法を選択する必要があり，これについては文献3)で分析されている．

3.2 提 案 法

本研究ではUDTアルゴリズムのアイデアを利用して距離変換を用いたモルフォロジー演算の高速化を提案する．距離変換を利用する場合にはフィルタの形状が距離関数によって決定される．そこで本研究ではUDTアルゴリズムの性質の詳細を検討したうえでUDTアルゴリズムに適用可能な距離関数を考え，モルフォロジー演算に用いることのできるフィルタ形状のクラスを広げる．これによって円や正方形など以外にも様々な形状のフィルタについて高速なモルフォロジー演算が可能となる．提案法の時間計算量は $O(n^2)$ であり，フィルタのサイズには影響を受けない．提案法において使用可能なフィルタ形状のクラスについては次章で詳しく述べる．

4. 効率良くモルフォロジー演算可能なフィルタのクラス

4.1 x 軸片側単調凸なフィルタ

本研究ではUDTアルゴリズムのアイデアを利用してモルフォロジー演算を行うことを考える．なお，以下ではフィルタは画像としてではなく xy 平面上の図形として考える．この場合，フィルタ領域とは図形の外周および内部の領域を指すものとする．また，フィルタ図形は多角形や円，長円形など，定数個のパラメータで表現できる図形とする．定理1より次の定理が得られる．

定理 2 軸対称ノルムに基づく距離の単位円に相似な形状のフィルタについては，モルフォロジー演算が $O(n^2)$ 時間で実行可能である．

この定理により，円形，正方形，ダイヤモンド型のフィルタなどに加え，長方形や長円形のフィルタなどについてもそのサイズに関係なく $O(n^2)$ 時間でモルフォロジー演算が可能であることが分かる．しかし，モルフォロジー演算のフィルタは，これら以外にも様々な形状が要求される場合がある．そこで，より広いフィルタ形状のクラスについて高速にモルフォロジー演算を行うことを考える．

ここで，UDT アルゴリズムが適用できる距離について考察する．UDT アルゴリズムでは，軸対称ノルムに基づく距離の単調性および対称性を利用して距離変換を列の処理と行の処理に分けて実行できた[5]．しかしこの UDT アルゴリズムが必要とする性質について考察すると，距離が y 座標について単調かつ対称であればよいことが分かる．それは以下の理由による．行 e 内の画素 $p_0 = (k, e)$ から，列 t 内の前景画素 $p_1 = (t, e_1)$ と $p_2 = (t, e_2)$ までの距離を考える．ただし $|e_1 - e| > |e_2 - e|$ とする．このとき，距離が y 座標について単調かつ対称であれば，$d(p_0, p_1) \geq d(p_0, p_2)$ がつねに成り立つ．そのため，変換 1 では列 t 内で行 e に最も近い前景画素までの距離値のみ記憶している．定理 1 において距離が y 座標に単調かつ対称であることは，距離が軸対称ノルムに基づくことによって満たされている．

また，UDT アルゴリズムの変換 2 において，距離グラフの下側包絡線が $O(n)$ 時間で求められたのは，行 e の処理における 2 つの距離グラフの交点がたかだか 1 点であるからである．このことは距離が三角不等式を満たすことで保証される（ノルムに基づく距離は三角不等式を満たすことに注意）が，詳細は文献 5) を参照されたい．

以上の考察に基づいて，フィルタが x 軸の上（下）側のみにフィルタ領域を持つ凸図形で，その輪郭線が x 軸に対して単調である場合を考える．このようなフィルタを x 軸上（下）側単調フィルタと呼ぶ．輪郭線が x 軸に対して単調であるとは，輪郭線の x 軸と重ならない部分が，あらゆる垂直線とたかだか 1 点もしくは 1 線分でしか交わらないときをいう．

ここで図形 X に基づく距離を次のように定義する．

定義 1 図形 X に基づく距離 d_{X} とは，2 点 p，q に対して $d_{\mathrm{X}}(p, q) = \min\{a \geq 0 \mid q = p + ax, \exists x \in \mathrm{X}\}$ とする．また $d_{\mathrm{X}}(p) = d_{\mathrm{X}}(0, p)$ とする．

図形 X に基づく距離 d_{X} は座標系の平行移動に関して不変，すなわち $d_{\mathrm{X}}(p - c, q - c) = d_{\mathrm{X}}(p, q)$ である．また図形 X の輪郭線は距離 d_{X} の単位円と考えることができる．この d_{X} は図形 X が凸であるならば三角不等式を満たすことを次の補題で示す．

補題 1 凸図形 X に基づく距離 d_{X} は三角不等式を満たす．

証明：図形 X に基づく距離 d_{X} の三角不等式は

$$d_{\mathrm{X}}(p) + d_{\mathrm{X}}(q) \geq d_{\mathrm{X}}(p + q)$$

と表現できる．$d_{\mathrm{X}}(p) = a$，$d_{\mathrm{X}}(q) = b$ とする．このとき，d_{X} の定義より $x_1 = p/a \in \mathrm{X}$，$x_2 = q/b \in \mathrm{X}$ である．任意の t $(0 \leq t \leq 1)$ に対して，$z(t) = tx_1 + (1 - t)x_2$ を考えると，図形 X が凸なので，$z(t) \in \mathrm{X}$ である．ここで，$t = a/(a + b)$ とおくと，

$$\begin{aligned} z(t) &= \frac{ax_1}{a+b} + \frac{bx_2}{a+b} \\ &= \frac{p}{a+b} + \frac{q}{a+b} \\ &= \frac{p+q}{a+b} \\ p + q &= (a + b) \times z(t) \end{aligned}$$

である．また $a \geq 0$，$b \geq 0$ より $(a + b) \geq 0$ である．したがって d_{X} の定義より，$d_{\mathrm{X}}(p + q)$ は $(a + b)$ で上から抑えられる．すなわち $d_{\mathrm{X}}(p + q) \leq a + b = d_{\mathrm{X}}(p) + d_{\mathrm{X}}(q)$ である． □

図形 X として x 軸上（下）側単調で凸な形状を考えた場合，図形 X に基づく距離について UDT アルゴリズムによる距離変換ができれば，図形 X をフィルタとして用いたモルフォロジー演算も $O(n^2)$ 時間で計算可能となる．

x 軸上（下）側単調で凸な図形 X に基づく距離は y 座標について対称ではないので，UDT アルゴリズムをそのまま適用することはできない．しかしこの図形 X を y 座標について対称な距離の単位円の上（下）側部分のみであると考えることで処理することができる．そのために UDT アルゴリズムの変換 1 の代わりに次の変換 1b を行うことにする．

【変換 1b：一方向列処理】各列 t ごとに列方向のみを参照して，各画素から下にある最も近い前景画素までの画素数（距離値とは異なることに注意）を求める．すべての列についてこの処理を行った後に得られる画像を $\mathbf{C}'\{c'_{ij}\}$ $(i \in z_x, j \in z_y)$ とする．ただし

$$c'_{ij} = \min_{p \in z_y}\{(j - p) \mid I_{ip} = 1, p \leq j\}$$

である．また，列 t において画素 (t, j) より下に前景画素がない場合は $c'_{tj} = \infty$ とする．各画素から上方向について最も近い前景画素までの画素数を求めたい場合も同様に処理する．

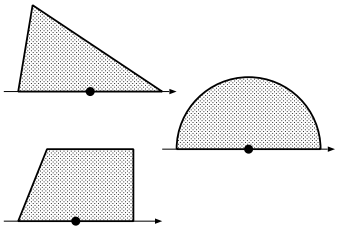

図 1 x 軸上側単調で凸なフィルタの例
Fig. 1 Examples of the upper monotone convex filter.

x 軸上側単調で凸なフィルタについてモルフォロジー演算を行う場合には，次のことに留意しなければならない．すなわち 2 点間の距離が非対称（$d(p_1,p_2) \neq d(p_2,p_1)$）であり，距離公理を満たしていないということである．そのため距離変換を行う際には距離の方向性を考慮しなければならない．距離変換を用いてディレーションを行う場合には，距離変換の結果として各画素について前景画素からの距離の最小値を得る必要があり，その結果を 2 値化することでディレーション結果が得られる．UDT アルゴリズムを用いる場合は以下のように処理を行う．まず，変換 1b（下方向への画素数の計算）を行う．次に変換 2 では，フィルタ形状の点対称図形に基づいて距離グラフを表す関数 $\ell_k^j(x)$ を用いる．このようにして得られた出力を 2 値化すればよい．一方，距離変換を用いてイロージョンを行う場合には，距離変換の結果として，入力画像の反転画像に対して，各画素について前景画素までの距離の最小値を得る必要があり，その結果を 2 値化することでイロージョン結果が得られる．UDT アルゴリズムを用いる場合は以下のように処理を行う．まず，変換 1b（上方向への画素数の計算）を行う．次に変換 2 では，フィルタ形状に基づいて距離グラフを表す関数 $\ell_k^j(x)$ を用いる．このようにして得られた出力を 2 値化すればよい．x 軸下側単調なフィルタについても同様である．以下では x 軸上側単調または x 軸下側単調のことを単に x 軸片側単調と呼ぶことにする．

定理 3 x 軸片側単調で凸なフィルタについてはモルフォロジー演算が $O(n^2)$ 時間で計算可能である．

図 1 に x 軸上側単調で凸なフィルタの例をいくつかあげる．

4.2 フィルタ分解の利用

図 2 (a)，(b) のようなフィルタ形状を考える．これらのフィルタはフィルタ領域を図にあるような水平線で上下に分割すると，上下それぞれの形状が x 軸片側単調で凸となる．

このような場合は，上下に分割したフィルタそれぞれについて変換 1 の代わりに変換 1b を用いた距離変換を行い，得られた画像 **T** を 2 値化してモルフォロジー演算の出力を得，ディレーション（イロージョン）の分配則に基づいて，それぞれの出力を，OR 演算（AND 演算）で組み合わせることで，もとのフィルタのモルフォロジー演算が可能である．またフィルタ原点が上下分割に用いた水平線上にない場合は，水平線上に仮の原点を設定して距離変換を施して 2 値化し，その後に本来の原点位置に合わせて結果を平行移動してやればよい．OR 演算，AND 演算や画像の平行移動操作に要する時間計算量は $O(n^2)$ 時間である．

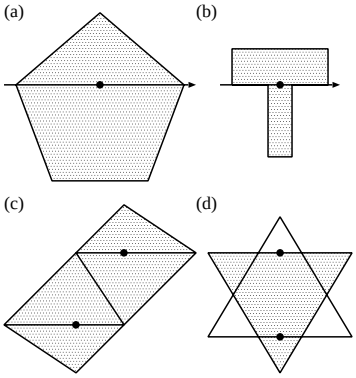

図 2 フィルタの分解を用いる例
Fig. 2 Examples of filter decomposition.

また図 2 (c)，(d) のようなフィルタであっても，それを x 軸片側単調なフィルタに分解（分割もしくはカバー）し，分解して得られたそれぞれのフィルタについてモルフォロジー演算を行い，その結果を組み合わせることによって，分解前のフィルタによるモルフォロジー演算の結果が得られる．これはモルフォロジー演算の平行移動不変性と分配則による．この場合，分解されたフィルタの個数が定数個ならば，処理全体は $O(n^2)$ 時間で実行できる☆．

系 1 定数個の x 軸片側単調な図形に分解できる形状のフィルタはモルフォロジー演算が $O(n^2)$ 時間で計算可能である．

5. 実験結果および考察

単純にディレーション演算を定義どおりに行う場合（以下，単純法）と，UDT アルゴリズムを用いてディレーション演算を行う場合（以下，UDT 法），さらに 1 次元分解型フィルタリングに距離変換を利用する場合（以下，1 次元 DT 法）について計算機による計算時間比較の実験を行った．なお 1 次元分解

☆ 本論文ではフィルタの分解は目視により行うとしている．フィルタが多角形のときは，分割の自動化にはたとえば文献 9) のアルゴリズムを用いることができる．

型フィルタリングを距離変換で処理する方法は文献8) でハードウェア化する際にも利用されている考え方で，1 次元フィルタそれぞれについての処理時間を高速化することができる．1 次元 DT 法では各 1 次元フィルタ（水平方向）の左端を原点として扱う．入力画像に右方向のみへの距離変換（各画素について $t_{ij} = \min_s\{d((i,j),(s,j))|I_{sj} = 1, i \leq s\}$ を求めること）を 1 度実行し，その出力を 1 次元フィルタの長さをしきい値として 2 値化し，その結果を平行移動してすべて組み合わせるという方法である．この場合，距離変換は最初に 1 度実行した結果を記憶しておけばよく，あとは 2 値化するときのしきい値の変更だけ行えばよい．よって 1 次元 DT 法の計算量は $O(n^2r)$ である．なお，1 次元分解型フィルタリングにおいては，同じ長さのフィルタが複数ある場合には結果を記憶しておくことにより計算量の削減が可能であるが，その手法は 1 次元 DT 法にも同様に適用可能である．ただし本実験ではその高速化処理は施していない．実験に用いた円形フィルタの場合，理論的にはさらに約 2 倍程度早くなることが期待される．また，UDT 法および 1 次元 DT 法では演算結果の平行移動処理を行う場合に，入力画像の端に前景画素があると正しい結果が得られないことがある．これは入力画像のサイズよりも適度に大きめの計算領域を確保することで回避可能である．

実験に用いた入力画像はサイズ 1280×960 の 2 値画像（**図 3**）で，入力画像の前景背景比率は約 2 : 5 である．円形フィルタ（UDT 法の場合は Euclid 距離を採用）のサイズを変えて時間計測を行った結果が**表 1** である．表中でフィルタ欄の括弧の中の数字は，フィルタ画像の縦横幅を表す．たとえば「円形 (5)」ならば 5×5 のサイズの円形フィルタという意味である．なお，$r \times r$ のフィルタ画像は直径 r の円を前景として持つものである．また，時間の単位はすべて［秒］である．計算機環境は PC-9821 Lavie（CPU Pentium 133 MHz），Windows95，主記憶 80 MByte，使用言語は VisualC++で，時間計測には Windows が実験用プログラム以外の処理をしていない状態で clock 命令を用いて行った．

ここで注目すべきことは，最初にも述べたように，UDT 法の場合は処理速度がフィルタサイズに依存しないということである．一方，単純法や 1 次元 DT 法はフィルタサイズの増大にともなって処理時間が増大していく．単純法の場合はフィルタサイズ r が 2 倍になると処理時間が約 4 倍に，1 次元 DT 法の場合はフィルタサイズ r が 2 倍になると処理時間も約 2 倍になることが実験結果から見てとれる．フィルタサイズがきわめて小さいときは単純法でも処理時間はあまりかからないが，全体としては単純法よりも 1 次元 DT 法のほうが高速に処理可能であり，またフィルタサイズが大きくなるほどその処理時間の差も広がっていく．しかしフィルタサイズが極度に大きくなった場合（31 × 31）などは 1 次元 DT 法の処理時間は UDT 法よりも長くなる．これは $O(n^2)$ と $O(n^2r)$ の差が現れてくるものである．よってフィルタサイズが大きくなればなるほど，UDT 法の有効性が増すことが分かる．

また図 2 (b) の T 字型フィルタ（UDT 法の場合は 2 個の長方形フィルタに分解．フィルタサイズは前景の T 字型を覆う最少の正方形のサイズ）の場合の実験結果を**表 2** に示す．

フィルタ形状が変わっても処理時間の全体としての傾向は同じである．単純法は r^2 に比例し，1 次元 DT

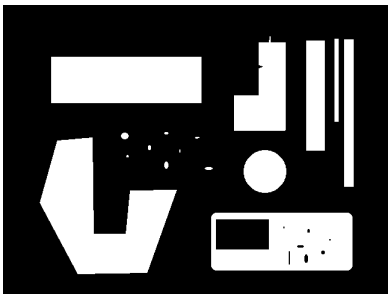

図 3 入力画像（1280 × 960）
Fig. 3 Input image (1280 × 960).

表 1 円形フィルタによるディレーション処理に要する時間比較
Table 1 Computing time for dilation with circle filters.

フィルタ	単純法	UDT 法	1 次元 DT 法
円形 (5)	3.41 [s]	13.25 [s]	2.72 [s]
円形 (11)	14.03 [s]		5.31 [s]
円形 (15)	23.87 [s]		7.03 [s]
円形 (21)	48.12 [s]		9.75 [s]
円形 (25)	66.35 [s]		11.44 [s]
円形 (31)	100.00 [s]		14.18 [s]
円形 (51)	270.12 [s]		23.72 [s]

表 2 T 字型フィルタによるディレーション処理に要する時間比較
Table 2 Computing time for dilation with T-shape filters.

フィルタ	単純法	UDT 法	1 次元 DT 法
T 字型 (5)	4.65 [s]	28.9 [s]	3.75 [s]
T 字型 (11)	13.03 [s]		6.97 [s]
T 字型 (15)	22.78 [s]		8.71 [s]
T 字型 (21)	40.84 [s]		11.35 [s]
T 字型 (25)	57.50 [s]		13.00 [s]
T 字型 (31)	90.94 [s]		15.87 [s]
T 字型 (51)	239.34 [s]		24.87 [s]
T 字型 (75)	511.69 [s]		36.43 [s]

法は r に比例している．1 次元 DT 法は入力画像およびフィルタサイズが同じ場合，円形フィルタでも T 字型フィルタでも処理時間に大きな違いはない．これは各 1 次元フィルタについて 2 値化処理をする際のしきい値および平行移動量がフィルタ形状によって変化するのみであり，今回の場合計算量自体が変わらないためである．UDT 法は T 字型フィルタであってもフィルタサイズに関係なく一定した処理時間となる．ただし，UDT 法は T 字型フィルタを 2 つの長方形に分解してそれぞれ距離変換を行うため，処理時間がほぼ倍になり，$r = 51$ までは 1 次元 DT 法の方が優位である．

UDT 法は本研究で処理可能なフィルタ形状のクラスを拡張したとはいえ，任意のフィルタ形状について利用可能なわけではないので，1 次元 DT 法を併用するなど，使うフィルタの形状やサイズによって用いるアルゴリズムを選択することがよいと考えられる．

6. おわりに

本論文では距離変換アルゴリズムを用いて高速にモルフォロジー演算可能なフィルタの形状について検討した．この方法はフィルタサイズに処理時間が依存しないという特徴を持っている．距離変換アルゴリズムとしては文献 5) の UDT アルゴリズムの考え方を利用した．UDT アルゴリズムを利用して高速にモルフォロジー演算可能なフィルタ形状として，軸対称ノルムの単位円に相似な形状および x 軸片側単調で凸な形状を定義した．またこれらの形状に該当しないフィルタであっても x 軸片側単調で凸な定数個のフィルタに分解できるならば $O(n^2)$ 時間でモルフォロジー演算が可能であることを示した．そして従来法を含めて実験による比較を行い，距離変換を利用する方法がフィルタサイズが大きいときに非常に有効であることを示した．

実際上のモルフォロジー演算のアルゴリズムとしては本研究で示した距離変換による方法のみでなく，利用するフィルタの形状やサイズに合わせて方法を選択することが処理時間の観点からは最適であると思われる．しかし，UDT アルゴリズムを用いる方法は画像処理で用いられる大部分のフィルタについて統一的に効率の良いモルフォロジー演算を与えるものである．

参考文献

1) Zhuang, X. and Haralick, R.: Morphological structuring element decomposition, *CVGIP*, Vol.15, No.3, pp.370–382 (1986).
2) Xu, J.: Decomposition of convex polygonal morphological structuring elements into neighborhood subsets, *IEEE Trans. Pattern Anal. & Machine Intell.*, Vol.13, No.2, pp.153–162 (1992).
3) 仁保 勉，江 浩，山本眞司：Mathematical morphology 演算の高速化アルゴリズムの比較，情報処理学会論文誌，Vol.37, No.10, pp.1751–1759 (1996).
4) 加藤敏洋，平田富夫，斉藤豊文，吉瀬謙二：ユークリッド距離変換アルゴリズムの効率化，電子情報通信学会論文誌，No.12, pp.1750–1757 (1995).
5) Hirata, T.: A unified linear-time algorithm for computing distance map, *Information Processing Letters*, No.58, pp.129–133 (1996).
6) 小畑秀文：モルフォロジー，コロナ社 (1996).
7) 横井茂樹，鳥脇純一郎，福村晃夫：画像処理のための 2 次元フィルタリングの 1 次元分解について，電子情報通信学会論文誌，No.7, pp.512–513 (1978).
8) 小島昭二，海老澤嘉伸，宮川達夫：大きな構造要素が使える画像の高速モルフォロジーハードウェア，電子情報通信学会論文誌，No.6, pp.1106–1113 (1993).
9) Asano, T. and Asano, T.: Minimum partition of polygonal regions into trapezoids, *Proc. 24th Annual IEEE Symposium on Foundation of Computer Science*, pp.233–241 (1983).

(平成 12 年 3 月 27 日受付)
(平成 12 年 10 月 6 日採録)

櫻井 敦史（正会員）

昭和 49 年生．平成 10 年名古屋大学工学部電子工学科卒業．平成 12 年同大学大学院博士課程前期課程修了．同年 CSK 総合研究所入社．在学中，並列アルゴリズム，画像処理アルゴリズムに関する研究に従事．

平田 富夫（正会員）

昭和 51 年東北大学工学部通信工学科卒業．昭和 56 年同大学大学院博士課程修了．昭和 56 年豊橋技術科学大学助手．昭和 61 年名古屋大学工学部情報工学科講師．現在，名古屋大学工学研究科電子工学専攻教授．電子情報通信学会，ACM，IEEE 各会員．工学博士．グラフアルゴリズム，データ構造の研究に従事．