コンピュータビジョン 85－2
（1993．9．30）

集合演算による濃淡画像の解析とその記述

浅田 卓哉　井宮 淳　市川 熹

千葉大学 工学部 情報工学科

数理形態学によって数式として構成された形状解析アルゴリズムを実際に計算機で実行する場合，集合演算を計算機でどのように扱うかが問題となる．数理形態学の演算は集合から集合への写像として表現される．一方，関数型言語の演算は，入力から出力への写像として定義される．従って，関数型言語において集合をデータとして定義すれば，数理形態学の演算を簡潔に記述することが可能となる．また，ラムダ記法による関数型言語を用いることにより，形状解析アルゴリズムの構成的な作成や数学的な証明が可能となる．

本論文ではまず，濃淡画像を2値画像の集合として表すことによって数理形態学表現を行ない，数理形態学による形状解析アルゴリズムの記述法を濃淡画像に適用する方法を提案する．また，2値画像に対応する平面上の点集合をリストによって表現し，リストに対するラムダ算法を基本として数理形態学の演算の記述子を新たに定義する．さらに，濃淡画像の連結成分の間の位相構造もリスト表現し，濃淡画像の位相構造を抽出する関数を設計する．

Set Theoretical Method for Topological Analysis of Gray-scale Images

Takuya ASADA,　Atsushi IMIYA,　and　Akira ICHIKAWA

Dept. of Information and Computer Sciences, Chiba University
1-33 Yayoi-cho, Inage-ku, Chiba 263 JAPAN
e-mail imiya@ics.tj.chiba-u.ac.jp

Mathematical morphology clarifies what we must compute in binary digital image analysis. The theory, however, does not clarify how to compute set theoretical operations for shape analysis. This paper first introduces a functional programming language for morphological operation by describing set theoretical operations of binary morphology as functions from lists to lists. Furthermore, we extend the method of the classical binary mathematical morphology to gray-scale digital images. Since each level of quantized gray-scale determines a set of points on the plane, we describe a quantized gray-scale digital image as a set of binary digital images. Moreover, we also introduce algorithms to derive the connected graph and the comprehension tree of connected components of gray-scale images through the morphological expression of gray-scale images and functional expressions of set theoretical operations.

1. まえがき

従来の画像解析手法では，データ構造が画像の形状情報と対応していないため，系統的にアルゴリズムを構成することが難しい．またデータおよびプログラムの記法が統一されておらず，再利用性が乏しい等の欠点があった．さらに，記述されたアルゴリズムの意味が分かりづらいなどの問題点がある．

数理形態学では，集合演算を用いることにより，集合を平面上の図形と直接的に対応させて行うことができる．従って，数理形態学によって形状処理アルゴリズムを視覚的に構成することができ，操作の幾何学的な意味も分かりやすく効率的なアルゴリズム設計が可能になる [1]．しかし，構成されたアルゴリズムを計算機で実行するとき，数理形態学の集合演算を計算機でどのように扱かが問題となる [2,3]．

数理形態学の演算は集合から集合への写像として表現される．一方，関数型言語の演算は入力データから出力への写像として数学的に表すことができる．従って，関数型言語において，集合をデータとして定義することによって，数理形態学の演算を簡潔に記述することが可能となる．また，データおよびプログラムの記法が統一され，全ての処理は入力から出力への写像として取り扱うことができる．そのため，プログラムの再利用性が高く，さらに，ラムダ記法による関数型言語を用いることにより，アルゴリズムの構成的な作成や数学的な証明が可能となる [4,5,6]．

本論文では，まず，濃淡画像を2値画像の集合族として表すことによって数理形態学の演算の対象となるように表現する．そして，濃淡画像の形状解析アルゴリズムを数理形態学の枠組みで記述する手法を提案する．次に，数理形態学の体系を見直し，関数型言語の記述を意識しながら再構築を行なう．そして，画像を現す平面上の集合族をリスト [7] によって表現し，数理形態学の演算を記述するラムダ算法を基本とした記述子によって表現する [4]．最後に，これらの記法を用いて，集合演算に基づいた濃淡画像に対するラベル付けのアルゴリズム，連結成分の隣接グラフの構成アルゴリズム，および，濃淡画像の連結成分の決める包括木の導出アルゴリズム [8] を入力画像から出力リストへの関数対応として記述する．

2. 離散画像の表現

2次元ユークリッド空間 R^2 の直交座標 (x,y) のベクトルを $\boldsymbol{x}=(x,y)^T$ とする．そして，$\boldsymbol{x}$ の要素がすべて整数である点を格子点と呼び，R^2 の格子点全体の集合を Z^2 で表すことにする．次に，R^2 において定義された関数を $f(x,y)$ によって表すことにする．本論文では $f(x,y)$ が各点において零以上の有限の値を持つ場合を考えることにする．特に，f の値が0か1かのどちらかである関数を2値関数と呼ぶことにする．

格子点 $\boldsymbol{x}=(m,n)^T$ を中心とする R^2 上の単位正方形領域を $\boldsymbol{P}_{mn}$ とする:

$$\begin{aligned}\boldsymbol{P}_{mn} \;=\; \{\boldsymbol{x}=(x,y)^T \\ |m-\frac{1}{2}\le x\le m+\frac{1}{2}, \\ n-\frac{1}{2}\le y\le n+\frac{1}{2}\ \}\end{aligned} \tag{1}$$

Z^2 の任意に固定した部分集合を $\boldsymbol{F}$ とする．そして，$\boldsymbol{F}$ の点を中心とする単位立方体領域の和集合を $\boldsymbol{V}$ とする:

$$\boldsymbol{V}=\bigcup_{\boldsymbol{x}\in\boldsymbol{F}}\boldsymbol{P}_{mn} \tag{2}$$

このとき，P_{mn} の上で値が一定値 f_{mn} であり，$\overline{\boldsymbol{V}}$ の上で値が0である関数 f:

$$f(x,y)=\begin{cases} f_{mn}: & \boldsymbol{x}\in\boldsymbol{P}_{mn} \\ 0: & \boldsymbol{x}\in\overline{\boldsymbol{V}}\end{cases} \tag{3}$$

を離散画像と呼ぶことにする．以下では，画素の濃度値 f_{mn} が量子化された量子化離散画像を考える．そして，量子化された濃淡値を正の整数で表すことにする．

さて，多値画像から濃度値の等しい点のみを取りだせば，それは2値画像として表わすことができる:

$$\boldsymbol{F}_\alpha=\{(m,n)^T|f_{mn}=\alpha\} \tag{4}$$

すなわち，多値画像を等しい濃度値を持つ点によって作られる2値画像を要素として持つ集合族と考えることができる:

$$\boldsymbol{F}=\{\boldsymbol{F}_\alpha|\alpha\in\mathrm{Z}_0,\alpha>0\} \tag{5}$$

ただし，Z_0 は零以上の整数の集合である．

特に，2値離散画像の場合には

$$\boldsymbol{F}_1=\{(m,n)^T|f_{mn}=1\} \tag{6}$$

によって濃度値1の点集合 $\boldsymbol{F}_1$ だけが画像から決まる．この場合は集合属 $\boldsymbol{F}$ と集合 $\boldsymbol{F}_1$ とを同一視することにする．

以上より，格子点集合と離散画像とが1対1に対応していることがわかる．離散画像に対応する点集合族 $\boldsymbol{F}$ を離散図形と呼ぶことにする．

図1に2値離散画像の点集合による表現の例を示す．また，図2に多値離散画像の2値離散画像の集合族としての表現の例を示す．

本論文では，点の間の代数構造を利用して画像 $\boldsymbol{V}$ の性質を解明する．以下では，離散図形だけを取り扱うので，離散図形を単に図形と呼ぶことにする．また，断わらない限り図形が有限集合である場合だけを取り扱う．従って，$\boldsymbol{F}_\alpha$ は2次元格子点集合 Z^2 の有限部分集合である．

3. 離散画像と集合演算

3.1 2値図形の集合演算

本節では以下で必要となる範囲で，2値画像の決める点集合の間の演算についてまとめる．集合 $\boldsymbol{F},\boldsymbol{G},\boldsymbol{H}$ を Z^2 の有限部分集合とする．$\boldsymbol{F}$ の補集合 $\overline{\boldsymbol{F}}$ は2値離散関数 f の0と1とを反転した2値離散関数 $\overline{f}$:

$$\overline{f}(x,y) = 1 - f(x,y) \tag{7}$$

の決める図形である．従って，一般には$\overline{F}$は無限集合となる．また，

$$\check{F} = \{-x | x \in F\} \tag{8}$$

をFの反転という．$a+b$をベクトル$a, b \in \mathrm{R}^2$の和とする．ベクトル$x \in \mathrm{Z}^2$によるFの平行移動集合F_xを

$$F_x = \{x + y | \forall y \in F\} \tag{9}$$

とする．

図形Fと図形Gとの和$F \cup G$，積$F \cap G$はそれぞれ，fとgとの合弁図形，fとgとの共通図形に対応する．そして，集合FとGとのミンコフスキー和を

$$F \oplus G = \bigcup_{x \in G} F_x \tag{10}$$

とする．

3.2. 多値画像の集合演算

集合族Fと集合族Gの和集合，差集合 $F \cup G$，$F \setminus G$を，次のようにして定義する．

$$F \bigcup G = \{F_\alpha \bigcup G_\alpha : \alpha \in \mathrm{Z}\} \tag{11}$$
$$F \setminus G = \{F_\alpha \setminus G_\alpha : \alpha \in \mathrm{Z}\} \tag{12}$$

また同様にして，FとGの共通部分$F \cap G$を

$$F \bigcap G = \{F_\alpha \bigcap G_\alpha : \alpha \in \mathrm{Z}\} \tag{13}$$

と定義する．すなわち，集合族どうしの演算を，その要素のうち等しい添え字を持つものどうしの演算結果の集合族として表わす．

集合族Fに対する平行移動・回転は，それぞれFの要素であるF_αを平行移動・回転した結果の集合として表わすことができる．すなわち，

$$F_x = \{(F_\alpha)_x : \alpha \in \mathrm{Z}\} \tag{14}$$
$$\check{F} = \{\check{F}_\alpha : \alpha \in \mathrm{Z}\} \tag{15}$$
$$\tag{16}$$

である．また，反転を

$$\overline{F} = \{\overline{F_\alpha} : \alpha \in \mathrm{Z}\} \tag{17}$$

によって定義する．

4. 位相構造の解析

4.1. 近傍と連結性

原点と，原点の回りの8つの格子点の決めるベクトルを

$$\begin{aligned} e_1 &= (1,0)^T, \quad e_3 = (0,1)^T \\ e_5 &= -e_1, \quad\quad e_7 = -e_3 \end{aligned} \tag{18}$$

$$\begin{aligned} e_{2m} &= e_{2m-1} + e_{2m+1}, \\ & e_9 = e_1,\ m = 1,2,3,4 \end{aligned} \tag{19}$$

と置くことにする．

$y \in \mathrm{R}^2$の回りの8つの格子点の集合$N^{[8]}(y)$:

$$N^{[8]}(y) = \{x | x = y + e_i : i = 1,2,\cdots,8\} \tag{20}$$

を，点yの8近傍という．また同様に，

$$N^{[4]}(y) = \{x | x = y + e_i : i = 1,3,5,7\} \tag{21}$$

を，点yの4近傍という．以下では，$N^{[8]}(y)$, $N^{[4]}(y)$に共通の性質を述べる場合は代表して2つの集合を$N(y)$で表すことにする．また，$N((0,0)^T)$をNと書くことにする．

$x \in F_\alpha$に対して，$y \in N(x)$であるとき，xとyとは接続しているという．そして，xと接続している総ての点からなる集合を点xの連結成分という．

4.2. 連結成分のラベル付け

画像のそれぞれの連結成分に対して固有のラベルを割り当てる処理をラベル付けという．ラベルには普通1以上の整数が用いられる．また，背景にはラベル値0を割り当てることがある．ラベルを割り当てられた画像をラベル画像という．

ラベル画像はラベルを濃度値として持つ多値画像であり，1つの濃度値に対して1つの連結した図形が対応する．従って，ラベル値ℓを持つ連結成分を，

$$L(\ell) = \{(m,n)^T : \ell_{mn} = \ell\} \tag{22}$$

とすれば，ラベル画像を集合族L:

$$L = \{L(\ell) : \ell \in \mathcal{L}\}, \quad \mathcal{L} \subset N \tag{23}$$

と表すことができる．添付集合$\mathcal{L}$が1から始まる連続する整数であれば，$\mathcal{L}$の最大値が連結成分の個数となる．

2値画像に対するラベル付けは手順

1. 原画像からラベル付けされていない点を取り出す．
2. 取り出した点の属する連結成分を取り出す．
3. 取り出した連結成分を原画像から削除する．
4. 取り出した連結成分にラベルを付けてラベル画像に加える．
5. 原画像が空集合となれば停止する．
   そうで無ければ1に戻る．

によって行うことができる[2]．

原画像をF，取り出された一点をpとする．取り出される連結成分をCとし，初期値として点pのみを持つ集合とする．このとき，取り出した点の属する連結成分は

1. $\boldsymbol{C} := \{\boldsymbol{p}\}$ とする.

2. $\boldsymbol{F} \cap (\boldsymbol{C} \oplus \boldsymbol{N})$ が空であれば終了.

3. 2 の結果を $\boldsymbol{C}$ に加える.

4. $\boldsymbol{C}$ を $\boldsymbol{F}$ から削除する.

5. 2 へ戻る.

なる処理によって抽出できる [2]. 図 3 に近傍の伝搬による連結成分の抽出の動作原理を示す. この場合近傍 $\boldsymbol{N}$ は 4 近傍を考えている.

ミンコフスキー和の定義より, $\boldsymbol{C} \oplus \boldsymbol{N}$ は, $\boldsymbol{C}$ に隣接する総て点からなる集合を $\boldsymbol{C}$ に付け加えた集合となる. $\boldsymbol{C}$ の初期値は一点 $\boldsymbol{p}$ のみの集合であるから, 1 の結果が空集合になるまでこの過程を繰り返すことにより, 点 $\boldsymbol{p}$ を含む連結成分を取り出すことができる.

4.3. 多値画像のラベル付け

次に, 多値画像に対するラベル付けを考える. 連結性の定義より, 濃度値の異なる点は連結成分にはならない. 従って, 多値画像のラベル画像は画像 $\boldsymbol{F}$ の各濃度値を表す 2 値画像 $\boldsymbol{F}_\alpha$ に対するラベル画像 $\boldsymbol{L}_\alpha$ の和集合となる.

$$\boldsymbol{L} = \bigcup_{\alpha \in Z} \boldsymbol{L}_\alpha \tag{24}$$

ここで, 各 $\boldsymbol{L}_\alpha$ のラベル値をそれぞれ ℓ_α, その最大値を $\mathcal{L}_\alpha$ とする. ラベル画像 $\boldsymbol{L}$ のラベル値の最大値 $\mathcal{L}$ は

$$\mathcal{L} = \sum_{\alpha \in Z} \mathcal{L}_\alpha \tag{25}$$

となる. そこで, 各連結成分のラベル値が一意に決まるためには, $\ell \in \boldsymbol{L}$ と $\ell_\alpha \in \boldsymbol{L}_\alpha$ との関係を,

$$\ell = \ell_\alpha + \sum_{n=1}^{\alpha-1} \mathcal{L}_n \tag{26}$$

とすればよい.

4.4. 連結成分の位相構造

$\boldsymbol{A}_\alpha, \boldsymbol{B}_\beta$ を, それぞれラベル値 α, β を持つ連結成分とする. 連結成分の位相構造には,

(1) $\boldsymbol{A}_\alpha$ のどの要素の近傍にも $\boldsymbol{B}_\beta$ の要素が存在しない. すなわち, 離れている状態,

$$separate(\boldsymbol{A}_\alpha, \boldsymbol{B}_\beta) \equiv \forall x \in \boldsymbol{A}_\alpha,\ \boldsymbol{N}(x) \notin \boldsymbol{B}_\beta \tag{27}$$

(2) $\boldsymbol{A}$ の要素の近傍に $\boldsymbol{B}_\beta$ の要素が存在する点がある. すなわち, 接している状態,

$$tangent(\boldsymbol{A}_\alpha, \boldsymbol{B}_\beta) \equiv \exists x \in \boldsymbol{A}_\alpha,\ \boldsymbol{N}(x) \in \boldsymbol{B}_\beta \tag{28}$$

(3) $\boldsymbol{B}_\beta$ のすべての要素が $\boldsymbol{A}_\alpha$ の内部に含まれている状態,

$$include(\boldsymbol{A}_\alpha, \boldsymbol{B}_\beta) \equiv \{hole(\boldsymbol{A}_\alpha) = hole(\boldsymbol{A}_\alpha \cup \boldsymbol{B}_\beta) + 1\} \tag{29}$$

の 3 つの場合がある. ここで, 関数 $hole(\cdot)$ は, 集合・の中に存在する穴の個数である. 図 4 に上の (1)(2)(3) の場合の例を示す.

図形 $\boldsymbol{A}_\alpha$ の中に存在する穴の数が, 図形 $\boldsymbol{A}_\alpha \cup \boldsymbol{B}_\beta$ の中に存在する穴の数に比べて, ちょうど 1 だけ大きい場合, すなわち, $\boldsymbol{A}_\alpha \cup \boldsymbol{B}_\beta$ から $\boldsymbol{B}_\beta$ を取り除いた場合, その部分が $\boldsymbol{A}_\alpha$ の穴となる場合が (3) の状態である.

さて, 図形

$$\mathcal{E}(\boldsymbol{A}_\alpha) = (\boldsymbol{A}_\alpha \oplus \boldsymbol{N}) \setminus \boldsymbol{A}_\alpha \tag{30}$$

は $\boldsymbol{A}_\alpha$ に隣接する総ての点からなる集合である. 従って, $\mathcal{E}(\boldsymbol{A}_\alpha)$ と $\boldsymbol{B}$ との共通部分が空集合であれば, $\boldsymbol{A}_\alpha$ の要素の近傍に $\boldsymbol{B}_\beta$ の要素は存在しない.

また, $\boldsymbol{A}_\alpha$ が n 個の穴を持つ図形であれば, $\mathcal{E}(\boldsymbol{A}_\alpha)$ は $n+1$ 個の連結成分からなる. 従って, $\mathcal{E}(\boldsymbol{A}_\alpha)$ のラベル画像のラベルの総数 $n+1$ から図形 $\boldsymbol{A}_\alpha$ の穴の個数 n を計数できる.

さらに, 図形 $\mathcal{E}(\boldsymbol{A}_\alpha)$ を, 他の画素値を持つ画素と接する図形の要素と考えれば, ある領域の中の画素値の違う連結成分の個数を同じ方法で計数できる.

4.5. 位相構造の抽出

ラベル画像において等しいラベル値で表わされる 1 つの連結成分を対象と呼ぶことにする. 対象間の隣接関係は, 対象のラベル値を節点の符号とし, 接合関係を枝とする隣接グラフによって表すことができる. また, ある対象に接しており, かつ含まれている対象を子として持つ包括木によって包括関係を表わすことができる. 包括木は背景である対象 0 を根とし, 他の対象に含まれていない対象は全て対象 0 の子となる.

包括木は隣接グラフの部分グラフであり, 隣接グラフの背景を根として, 隣接要素同士の隣接関係を横の枝として取り除き, 縦の枝を取り出したものである. 従って,

1. ある要素 x に隣接している要素のリスト
   $Tang_x = \langle x'_1, x'_2, \cdots, x'_n \rangle$
   を作成する.

2. $\langle x, x'_1 \rangle, \langle x, x'_2 \rangle, \cdots, \langle x, x'_n \rangle$
   を出力し, 隣接グラフから削除する.

3. $-\{\langle x, y \rangle \mid x, y \in Tang_x\}$
   であるような対を隣接グラフから削除する.

4. $Tang_x$ の要素それぞれについて, 同様の操作を行なう.

なる処理を根となる背景要素を起点として再帰的に繰り返せば, 隣接グラフから包括木を抽出できる. 図 6 に濃淡画像の連結成分の作る隣接グラフの例を示す. また, 図 7 に隣接グラフからの包括木の導出原理を示す.

5. 集合演算の関数表現

R^2 の点 $\boldsymbol{x} = (m, n)^T$ は座標値のリストによって表現できる:

$$x = \langle\, m,\, n \,\rangle \tag{31}$$

そこで, 格子点集合を点のリストとして表すことにする:

$$\boldsymbol{F}_\alpha = \langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \cdots, \langle x_n, y_n\rangle\rangle \tag{32}$$

ただし, リスト内の各点の出現の順序は任意である.

同様に, 多値画像は2値画像の集合として次式のように表すことができる.

$$\boldsymbol{F} = \langle\, \boldsymbol{F}_1, \boldsymbol{F}_2, \cdots, \boldsymbol{F}_n \,\rangle \tag{33}$$

ここで, $f_{mn} = \alpha$である集合がない場合には, $\boldsymbol{F}_\alpha = \emptyset$ とする.

次に, 集合演算の記述に使われるいくつかの演算子を定義する.

$null(x)$ は, x が空リストであれば真, それ以外のとき偽となる:

$$\begin{aligned} \text{Def} \;:\; & \lambda x.null \\ & \equiv (x = \emptyset) \rightarrow \mathrm{T}; \mathrm{F} \end{aligned} \tag{34}$$

$equal(x, y)$ は, x, yが等しいリストであれば真, それ以外のとき偽となる.

$$\begin{aligned} \text{Def} \;:\; & \lambda x.\lambda y.equal \\ & \equiv (x = y) \rightarrow \mathrm{T}; \\ & (car(x) = car(y)) \\ & \rightarrow equal(cdr(x), cdr(y)); \mathrm{F} \end{aligned} \tag{35}$$

定義より, 明らかに $x = y$ならば $equal(x, y)$ であるから, 以後, $equal(x, y)$ を $x = y$と略記する.

$in(x, y)$ は, リスト x が要素として yを持てば真, 持たない場合偽となる.

$$\begin{aligned} \text{Def} \;:\; & \lambda x.\lambda y.in \\ & \equiv (x = \emptyset) \rightarrow \mathrm{F}; \\ & (car(x) = y) \rightarrow \mathrm{T}; \\ & in(cdr(x), y) \end{aligned} \tag{36}$$

また, 2次元画像処理の記述を簡略化するため, 以下の演算を定義する. $X(x), Y(x)$ は x を点とみなし, それぞれ x 座標, y座標を取出す:

$$\text{Def} \;:\; \lambda x.X \equiv car(x) \tag{37}$$

$$\text{Def} \;:\; \lambda x.Y \equiv cadr(x) \tag{38}$$

$point(x, y)$ は, 座標 (x, y) の点を表す:

$$\text{Def}: \lambda x.\lambda y.point \equiv cons(x, cons(y, \emptyset)) \tag{39}$$

上の演算子によって, 集合演算を関数型言語によって記述する. 集合の反転は,

$$\text{Def} \;:\; \lambda a.inv \equiv invsub(\boldsymbol{U}, a) \tag{40}$$

$$\begin{aligned} \text{Def} \;:\; & \lambda u.\lambda a.invsub \\ & \equiv (null(u)) \rightarrow \emptyset; \\ & (in(a, car(u)) \rightarrow invsub(cdr(u), a); \\ & cons(car(u), invsub(cdr(u), a)) \end{aligned} \tag{41}$$

となる. ただし, $\boldsymbol{U}$は画像定義域のすべての点の集合である.

和, 差は,

$$\begin{aligned} \text{Def} \;:\; & \lambda a.\lambda b.add \\ & \equiv (null(b)) \rightarrow a; \\ & (in(a, car(b)) \rightarrow add(a, cdr(b)); \\ & cons(car(b), add(a, cdr(b))) \end{aligned} \tag{42}$$

$$\begin{aligned} \text{Def} \;:\; & \lambda a.\lambda b.sub \\ & \equiv (null(a)) \rightarrow \emptyset; \\ & (in(b, car(a)) \rightarrow sub(cdr(a), b); \\ & cons(car(a), sub(cdr(a), b)) \end{aligned} \tag{43}$$

となる.

$$\boldsymbol{A} \bigcap \boldsymbol{B} = \boldsymbol{A} \setminus \overline{\boldsymbol{B}} \tag{44}$$

より, 共通部分は

$$\text{Def}: \lambda a.\lambda b.intsec \equiv sub(a, inv(b)) \tag{45}$$

と記述できる.

平行移動, 回転はそれぞれ,

$$\begin{aligned} \text{Def} \;:\; & \lambda a.\lambda x.move \\ & \equiv (null(a)) \rightarrow \emptyset; \\ & cons(point(X(car(a)) + X(x), \\ & Y(car(a)) + Y(x)), \\ & move(cdr(a), x)) \end{aligned} \tag{46}$$

$$\begin{aligned} \text{Def} \;:\; & \lambda a.check \\ & \equiv (null(a)) \rightarrow \emptyset; \\ & cons(-X(car(a)), -Y(car(a))), \\ & check(cdr(a))) \end{aligned} \tag{47}$$

となる.

ミンコフスキー和は

$$\begin{aligned} \text{Def} \;:\; & \lambda.a\lambda b.Madd \\ & \equiv (null(b)) \rightarrow \emptyset; \\ & add(move(a, car(b)), madd(a, cdr(b)) \end{aligned} \tag{48}$$

となる.

以上の演算は, 3次元以上の画像に対しても同様に定義できる.

6. 関数表現による算法の記述

さて, 多値画像のラベル付けのための関数 $Label$は, 2値画像に対するラベル付けのための関数 $LabelBin$ を用いて

$$\begin{aligned} \text{Def} \;:\; & \lambda f.Label \\ & \equiv (null(f)) \rightarrow \emptyset; \\ & add(LabelBin(car(f)), Label(cdr(f))) \end{aligned}$$

$$
\begin{aligned}
\text{Def} \ : \ & \lambda f.LabelBin \\
& \equiv (null(f)) \rightarrow \emptyset; \\
& cons(closure(f, get(f)), \\
& LabelBin(Sub(f, closure(f, get(f))))) \\
\text{Def} \ : \ & \lambda f.\lambda p.closure \\
& \equiv (null(Sub(f, IntSec(f, Madd(p, N))))) \\
& \rightarrow p; \\
& closure(f, IntSec(f, Madd(p, N))) \\
\text{Def} \ : \ & \lambda f.get \equiv cons(car(f), \emptyset)
\end{aligned}
$$

と記述できる．ここで，関数 $closure$ は，$\boldsymbol{N}$ とのミンコフスキー和を繰り返すことによって連結性分を取出す．また関数 get は，画像を表わすリストの先頭の要素を戻り値とする関数である．またさらに，Nは近傍$\boldsymbol{N}$のリストによる表現である．

a, bをそれぞれ対象$\boldsymbol{A}_\alpha$,$\boldsymbol{B}_\beta$ を表わす集合とすれば，$\boldsymbol{A}_\alpha$ と$\boldsymbol{B}_\beta$ とが隣接しているがどうかを，

$$
\begin{aligned}
\text{Def} \ : \ & \lambda a.\lambda b.tangentp \\
\equiv \ & not(null(intsec(sub(madd(a, N), a), b)))
\end{aligned}
$$

によって調べることができる．ただし，ラベル画像 L の中で対象 A と隣接した対象と Bとの対の集合を返す関数 $Tangent1$ は

$$
\begin{aligned}
\text{Def} \ : \ & \lambda L.\lambda a.Tangent1 \\
& \equiv null(L) \rightarrow \emptyset; \\
& tangentp(car(L), a) \\
& \rightarrow add(cons(car(L), a), \\
& Tangent1(cdr(L), a)); \\
& Tangent1(cdr(L), a)
\end{aligned}
$$

となる．

隣接グラフは無向グラフであるから，$\langle x, y \rangle$ が分かれば $\langle y, x \rangle$ は必要が無い．従って，隣接グラフを求める関数 $Tangent$ は

$$
\begin{aligned}
\text{Def} \ : \ & \lambda L.Tangent \\
\equiv \ & null(L) \rightarrow \emptyset; \\
& add(Tangent1(cdr(L), car(L)), \\
& Tangent(cdr(L)))
\end{aligned}
$$

となる．

さらに，隣接グラフに背景との隣接を含めた関数 $Tangent'$は

$$
\begin{aligned}
\text{Def} \ : \ & \lambda L.Tangent' \\
& \equiv Tangent(cons(GetBG(L), L))
\end{aligned}
$$

となる．ただし，$GetBG$ は背景を得るための関数，

$$
\begin{aligned}
\text{Def} \ : \ & \lambda f.GetBG \\
\equiv \ & null(f) \rightarrow inv(\emptyset); \\
& intsec(inv(car(f)), GetBG(cdr(f)))
\end{aligned}
$$

である．

以上より，隣接グラフから包括木を

$$
\begin{aligned}
\text{Def} \ : \ & \lambda x.\lambda t.TangTo \\
\equiv \ & null(t) \rightarrow \emptyset; \\
& car(car(t)) = x \\
& \rightarrow cons(cdr(car(t)), TangTo(x, cdr(t))); \\
& cdr(car(T)) = x \\
& \rightarrow cons(car(car(t)), TangTo(x, cdr(t))); \\
& TangTo(x, cdr(t)) \\
\text{Def} \ : \ & \lambda x.\lambda t.CutVPath \\
& \equiv null(t) \rightarrow \emptyset; \\
& in(car(car(t)), x) \wedge in(cdr(car(t)), x) \\
& \rightarrow CutVPath(x, t); \\
& cons(car(t), CutVPath(x, cdr(t))) \\
\text{Def} \ : \ & \lambda x, \lambda y.\lambda t.OutHPath \\
& \equiv null(t) \\
& \rightarrow \emptyset; \\
& in(cons(x, car(y)), t) \\
& \rightarrow cons(cons(x, car(y)), \\
& OutHPath(x, cdr(y), t)); \\
& OutHPath(x, cdr(y), t) \\
\text{Def} \ : \ & \lambda x.\lambda t.HPath \\
& \equiv OutHPath(x, TangTo(x), t) \\
\text{Def} \ : \ & \lambda x.\lambda t.Remain(x, t) \\
& \equiv Sub(CutVPath(TangTo(x), t), \\
& OutHPath(x, TangTo(x), t)) \\
\text{Def} \ : \ & \lambda x.\lambda t.GIT \\
& \equiv null(x) \\
& \rightarrow \emptyset; \\
& atomp(x) \\
& \rightarrow union(HPath(x, t), \\
& GIT(TangTo(x), Remain(x, t))); \\
& union(GIT(car(x), t), GIT(cdr(x), t))
\end{aligned}
$$

によって抽出できる．

7. むすび

2値画像処理として定義された数理形態学に多値画像の表現を追加した．多値画像を 2 値画像の集合として定義することより，多値画像に対する処理を 2 値画像処理に分解することが可能である．従って，既に 2 値画像に対して定義された形状解析アルゴリズムが存在するならば，多くの場合これを多少拡張するだけで容易に多値画像に対する解析アルゴリズムが記述可能であることを示した．

点集合のリストによる表現とラムダ計算に基づく集合演算の計算手順の記述法を導入し，関数型言語による

数理形態学記述法を構築した．このようにして記述されたアルゴリズムは，他の演算子と同様に入力画像から出力画像への関数となるため，関数をさらに複雑な処理のための演算子として用いることが可能となる．すなわち，関数型の記述子はプログラムのモジュール化が可能となり再利用性が高くなることが期待できる．

関数型記述子の適用として，多値画像に対するラベル付けのアルゴリズムを作成した．さらに，ラベル画像の解析を行い，ラベル画像から連結成分の隣接関係を表す隣接グラフを抽出し，その結果の隣接グラフから包括関係を表す包括木の抽出を行なうアルゴリズムを提案し，関数型記述子を用いて記述を行なった．

本研究の一部は文部省からの科学研究費補助金，實吉奨学会研究助成金，ならびに，電気通信フロンティア研究開発によるものである．

文献

[1] J. Serra, *Image Analysis and Mathematical Morphology,* Academic Press: London, 1982.

[2] A. Imiya and T. Nakamura, "Morphological operation via convolution," *Proc. of 6th Scandinavian Conference on Image Analysis,* vol. 2, pp. 878-881, Oulu, Finland, June 1989.

[3] A. Imiya, K. Wada and T.Nakamura, "Coded morphology for labelled pictures," IEICE Trans. Infoemation and Systems, Vol. E76-D, pp. 1-9, 1993.

[4] J. Backus, "Can programming be liberated from the von Neumann style? A functional style and its algebra of programs," *Communication of ACM,* Vol. 21, pp. 613-641, 1978.

[5] R. Bird, *Programs and Machines, An Introduction to the Theory of Computation,* John Wiley & Sons: London, 1976.

[6] M. Hušek, "Categories and mathematical morphology," in *Categorical methods in computer science with aspects from topology,* H. Ehring et al. Eds. *Lecture Notes in Computer Science,* Vol. 393, Springer-Verlag: Berlin, 1989, pp. 295-301.

[7] K. Voss, *Discrete images, objects and functions in* Z^n, Algorithms and combinatrics 11, Springer-Verlag:Berlin 1993.

[8] J. -I. Toriwaki, *Digital image processing for image understanding,* (in Japanese), vol.1 and vol.2, Syokodo:Tokyo, 1988.

付録

本論文の関数型言語は．ラムダ算法を基本としたものであり．リスト処理に適したいくつかの組込み関数を持つ．この言語の仕様のうち．集合演算を定義するために用いるものについて．簡単に説明する．

- T は真，F は偽の論理値を表す．
- $\emptyset$ は空集合を表す．
- 取り扱うデータにはアトム (任意の文字列および数字) と対がある．対は $\langle x.y \rangle$ のように書き表わされ．x, y はそれぞれアトム．または対である．
- リスト $\langle x_1, x_2, \cdots, x_n \rangle$ は．対

  $\langle x_1.\langle x_2.\cdots.\langle x_n.\emptyset\rangle\cdots\rangle$ の略記法である．
- $car(x), cdr(x)$ はそれぞれ対 x の第 1 要素．第 2 要素を表す．x がリストであるとすると．それぞれリストの先頭の要素．残りの部分を表す．
- $cons(x, y)$ は $\langle x, y\rangle$ なる対を表す．
- $ep(x, y)$ は．x, yが等しいアトムであれば真．それ以外のとき偽となる．また．$ep(x, y)$ を $x = y$と略記する．
- 数値アトムの比較演算子 $lt(x, y)$ は x が y よりも大きい場合真．それ以外の場合偽となる演算子であり．$x < y$ と略記する．同様に $gt(x, y)$ は x が y よりも小さい場合真となり．$x > y$ と表わす．
- 数値アトムに対する $plus(x, y), minus(x, y)$ などの基本数値演算演算を $x + y, x - y$などと略記する．
- $c_1 \rightarrow e_2; e_3$

  は，c_1の値が真であれば e_1．それ以外のとき e_2となる．

  $c_1 \rightarrow e_1; (c_2 \rightarrow e_2; (c_3 \rightarrow e_3; (\cdots e_n)\cdots)$

  を

  $c_1 \rightarrow e_1; c_2 \rightarrow e_2; c_3 \rightarrow e_3; \cdots e_n \cdots$

  と表す．
- Def $\lambda x.f \equiv$ *Equation* は，$f(x)$ なる関数を *Equation* によって定義する．
- $\lambda x.(\lambda y.f)$ を$\lambda x.\lambda y.f$．$(f(y))(x)$ を $f(x, y)$ と書きあらわす．

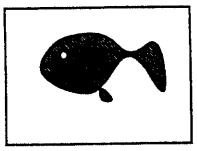


図 1)

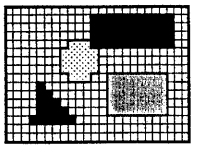


図 2)

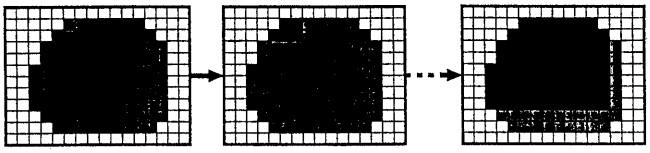
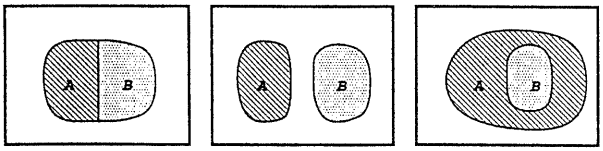


図 3)



(a) *tangent* (*A,B*) (b) *separate* (*A,B*) (c) *include* (*A,B*)

図 4)

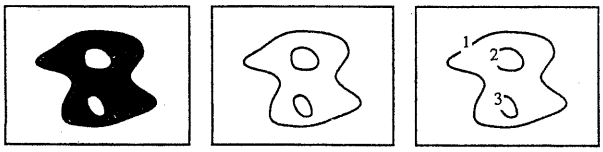
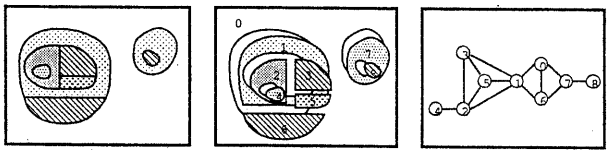
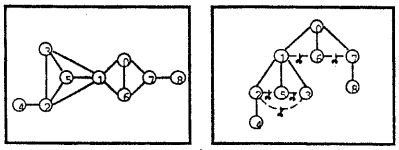


図 5)



(a) 原画像 (b) ラベル画像 (c) 隣接グラフ

図 6)



(c) 隣接グラフ (d) 包括木

図 7)

図 1 2 値離散画像の点集合による表現の概念図を示す.
図 2 多値離散画像の 2 値離散画像の集合族としての表現の概念図を示す.
図 3 近傍の伝搬による連結成分の抽出の動作原理を示す. この場合近傍 N は 4 近傍を考えている.
図 4 濃淡画像の 2 つの連結成分が (1) 離れている状態, (2) 接している状態, (3) 一方が他方に含まれている状態.
図 5 に輪郭抽出とラベル付けによる連結成分中の穴の計数の手順を示す.
図 6 濃淡画像の連結成分の作る隣接グラフの例を示す.
図 7 隣接グラフからの包括木の導出原理を示す.