画像解析における戦略の表現†

松山隆司†† 村山直樹†† 伊藤貴康††

複雑な画像を対象とした画像解析においては，単純な画像処理演算では十分満足のゆく結果を得ることはむずかしい．このため画像解析の精度や信頼性の向上を図るための戦略がこれまでにいくつか考案されてきた．たとえば，領域解析とエッジ解析の組合せや，ピラミッド構造を利用した多重解像度での解析などがその代表例である．ここでは，そうした画像解析のための戦略の多くが基本的な画像処理演算の高度な組合せ（画像処理演算の複合的合成）として表現できることを明らかにし，画像解析戦略の表現について画像処理ソフトウェア，知識表現の2つの観点から考察を加える．論文の前半では，まず画像解析のための代表的な戦略の実行形式を分析し，それらを表現するための3つの基本的な画像処理演算の複合的合成法を提案する．つぎに，画像処理演算の複合的合成を柔軟に記述する機能を持った関数型プログラミング言語のインタプリタを設計，試作し，複雑な画像解析過程が簡潔に記述できることを示す．論文の後半では，筆者らが先に提案した画像処理エキスパートシステム LLVE における戦略的知識の表現法について検討し，超グラフによる知識表現を導入することによって従来のシステム構成をほとんど変更することなく高度な画像解析が実現できることを示す．

1. はじめに

ディジタル画像処理研究の歴史も 20 年を越え，その間多くのアルゴリズムが考案され，我が国においては SPIDER[4]や SLIP などの画像処理ソフトウェア・パッケージとしてその体系化が行われている．最近では，こうした画像処理ソフトウェア・パッケージを基にして，目的に応じた画像処理手順の合成やアルゴリズム，パラメータの選択を自動的に行う画像処理エキスパートシステムがいくつか提案されている[1)-3)]．

一般に複雑な画像を解析し，意味のある特徴（線や領域）を抽出するには，様々な画像処理演算（オペレータ）をうまく組み合わせる必要がある．たとえば，領域分割とエッジ抽出を組み合わせ領域分割の誤りを修正したり，ピラミッド構造を用いることによりエッジ抽出の信頼性と処理効率の向上を図る，といった方法の有効性が明らかにされている．ここでは，こうした高度な画像解析の手法を画像解析のための戦略と呼ぶことにする．

画像処理エキスパートシステムの目的は，これまでに蓄積された様々な画像処理，画像解析のためのノウハウをシステムに蓄積，利用することによって高度な画像処理・解析システムを実現することにある．しかし，これまでに提案された画像処理エキスパートシステムでは，雑音除去→エッジ検出→2値化→エッジの連結などといった演算の逐次的な連結しか許されておらず，上で述べたような画像解析のための戦略を表現・利用することができないのが現状である．

本論文では，画像解析のための戦略の多くが基本的な画像処理演算の高度な組合せ（以下では画像処理演算の複合的合成と呼ぶ）として表現できることを明らかにする．論文の前半では画像処理ソフトウェアの観点から画像処理演算の組合せ方式について考察を加える．論文の後半では画像処理エキスパートシステムにおける知識表現の観点から画像解析戦略の表現法について検討する．まず2章で，画像解析のための代表的な戦略の実行形式を分析し，それらを表現するための3つの基本的な画像処理演算の複合的合成法を提案する．3章では，画像処理演算の複合的合成を柔軟に記述する機能を持った関数型プログラミング言語のインタプリタを設計，試作し，複雑な画像解析過程が簡潔に記述できることを示す．4章では，筆者の一人が先に提案した画像処理エキスパートシステム LLVE[1)] における戦略的知識の表現法について検討し，超グラフによる知識表現を導入することによって従来のシステム構成をほとんど変更することなく高度な画像解析が実現できることを示す．

2. 画像解析戦略における処理の実行形式

ここではまず画像解析における戦略をソフトウェア的な観点から分析する準備として，関数を用いた画像処理演算の表現について議論し，複数の画像処理関数の合成によって複雑な画像解析戦略が表現できることを示す．

† On Representation of Image Analysis Strategies by TAKASHI MATSUYAMA, NAOKI MURAYAMA and TAKAYASU ITO (Department of Information Engineering, Faculty of Engineering, Tohoku University).

†† 東北大学工学部情報工学科

2.1 関数型言語による画像解析過程の記述

基本的な画像処理演算を組み合わせ，高度な画像解析を実現するには，まず基本的な画像処理演算をどのように表現するかが大きな問題となる．

従来，画像処理プログラムはFORTRANやCなどの手続き型言語で記述されることが多かったが，文献1)でも述べたように，画像処理演算を，ある画像特徴（濃淡画像，微分画像，線，領域など画像データから抽出できる様々な特徴）を別の画像特徴に変換する関数と考えるのは自然であろう．画像処理演算を関数として表現することには次のような利点がある．

（1） 関数の逐次的合成による処理過程の記述

1章で例示したような逐次的な画像処理過程は，与えられた入力画像に対して次々と関数を適用すること（関数の逐次的な合成）として簡潔に表現できる．すなわち，画像処理演算を表す関数を f_i $(i=1\sim n)$ としたとき，これらを逐次的に連結して得られる画像処理過程は，$f_n(f_{n-1}(\cdots f_2(f_1(x))\cdots))$ と表現される．

（2） 中間結果の暗黙的表現

画像処理演算を関数として表現すると，画像特徴 D に対して演算 f を施した結果は $f(D)$ と表され，演算の出力を出力変数によって明示的に示さなくてもよい．これによって，プログラミングの効率や，デバッグの容易性，プログラムの可読性が大幅に向上される．たとえば，平滑化，二値化，ラベル付けといった処理過程を SPIDER[4] のルーチンを用いて記述すると，

```
CALL EGPR (IN, JP1, ISX, ISY)
CALL SLTH1 (JP1, JP2, ISX, ISY, THRE)
CALL CLAB (JP2, LAB, ISX, ISY, ……)
```
（1）

となる．ここで，IN は入力画像，JP1 は平滑化された画像，JP2 は二値化された画像，LAB はラベル画像，ISX と ISY は画像の大きさ，THRE はしきい値を表す．同じ処理過程を関数によって表現すると

```
LAB=CLAB (SLTH1 (EGPR (IN, ISX,
        ISY), ISX, ISY, THRE), ISX, ISY, ……)
```
（2）

と書け，JP1 や JP2 といった不要な中間結果を表す必要がない．ここで＝は代入を表す．

さらに，処理系におけるメモリ管理の立場から見ると，(2)のような関数による記述を用いることにより，メモリの効率的な利用が可能となる[5]．

（3） 型を用いた整合性の検証

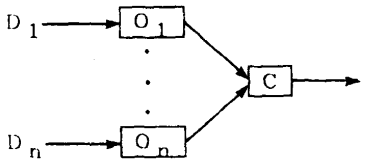

図 1 結果合成型演算
Fig. 1 Combination of multiple analysis results.

画像処理演算を関数として表現した場合，各関数の型（関数の引数のデータ型と出力のデータ型）を明示的に宣言し，処理データと関数の型との整合性を調べることによって意味のない処理を検出できる．このほか，構造を持った型を利用することにより，画像の大きさなどといったデータの構造を表す関数の引数を省略することや，微分演算のように複数のデータ（勾配の大きさと方向）を出力とするような演算を1つの値を返す関数として表現できる．たとえば，GRAY_PICTURE というデータ型を

```
TYPE GRAY_PICTURE=RECORD
    IMAGE_DATA : ARRAY [1…256], [1…256]
                 OF INTEGER;
    ISX  : 256;
    ISY  : 256
    END;
```

と定義し，その型の変数として LAB, IN を宣言すれば(すなわち，VAR LAB, IN : GRAY_PICTURE ;)，(2)は

```
LAB=CLAB (SLTH1 (EGPR (IN), THRE), …)
```
（3）

と表せ，本質的に意味のある引数のみを用いた簡潔な形で記述できる．(注：ここでの型宣言は MODULA-2 による表記法を用いた．)

2.2 画像処理演算の複合的合成の実行形式

ここでは，画像解析戦略が複数の画像処理演算の高度な組合せとして表現できることを示し，画像処理演算の複合的合成の基本形式として，（1）結果合成型（2）処理制御型（(2-1) マスク処理型 (2-2) パラメータ最適化型）を考え，それぞれの方式について考察を加える．

（1） 結果合成型

図1に示すように，画像処理演算 $O_1\sim O_n$ の処理結果を結果合成演算によって1つの結果に合成する複合的合成を結果合成型と呼ぶ．その出力は

$$C\,(O_1\,(D_1),\ \cdots,\ O_n\,(D_n)) \qquad (4)$$

で定義される．ここで，$O_i\,(D_i)$ は画像データ D_i を画像処理演算 O_i で処理した結果を表す．

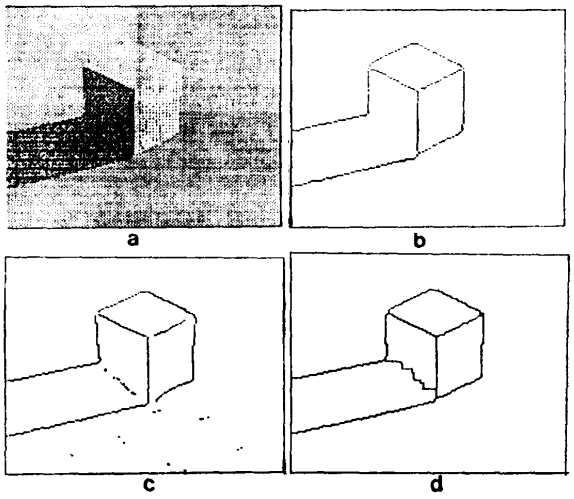

図 2 領域分割とエッジ検出の合成
Fig. 2 Integration of region-based and edge-based analysis.

結果の合成演算 C の代表例としては，(a) 画素ごとの算術・論理演算 (b) 部分結果の連結 (c) 処理結果の検証がある．

合成演算Cは形式的には複数の画像データを入力とする関数であるが，その処理内容は様々な評価判断機能を持った認識処理を含むため，通常の画像処理演算と区別している．

処理結果の検証の例としては次のようなものが考えられる．図2(a)のような画像の領域分割を行う場合，その方法としては画素の濃度値の類似性に基づく領域拡張法や微分オペレータによるエッジ検出が考えられるが，いずれの方法によっても完全な結果は得られない (図2(b), (c))．そこで，領域成長法の結果の妥当性をエッジ検出の結果を利用して検証し，誤った処理結果を修正する．具体的には，図2(b)の各領域内に含まれる (図2(c)中の) エッジ点の数を求め，その値が大きい領域を抽出し再分割する (図2(d))．ここで，抽出された領域の再分割には次に述べるマスク処理型の演算を用い，再分割された領域と図2(b)の他の領域とを空間的に合成して図2(d)の画像が得られる．(この解析過程の関数型言語による記述が3章で与えられる．)

(2) 処理制御型

図3に示すように，この方式ではまず画像処理演算 O_1 の処理結果を求め，その結果を用いて他の画像処理演算 O_2 の実行の制御を行う．処理制御型は，(2-1) 演算 O_2 の空間的な適用範囲を制限するマスク処理型 (2-2) 演算 O_2 に必要なパラメータの選択を行うパラメータ最適化型，に分けられる．

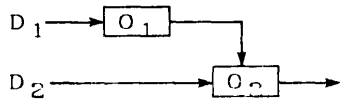

図 3 処理制御型演算
Fig. 3 Controlling analysis process.

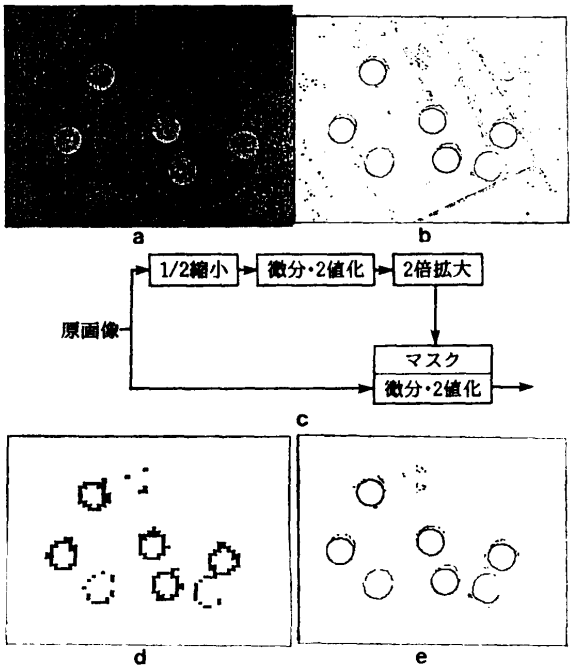

図 4 多重解像度を利用した画像解析
Fig. 4 Image analysis using multiple resolutions.

(2-1) マスク処理型

マスク処理型では，演算 O_1 の処理結果 (マスク画像) を演算 O_2 の処理制御用マスクとし，マスクの値が真の範囲に対してのみ演算 O_2 を施す．$M=O_1(D_1)$ とすると，O_2 の出力 D は，

$$D=O_2(D_2)\ (m_{ij}=\text{真}) \qquad (5)$$
$$\text{undefined}\ (m_{ij}=\text{偽})$$

となる．m_{ij} はマスク M の i 行 j 列の要素である．

マスク処理型の合成演算を用いるとピラミッド構造を用いた階層的な画像処理が容易に実現できる．たとえば，図4(a)の画像からエッジを抽出したい場合，単純な微分オペレータを用いると図4(b)のように多くの雑音が検出されてしまう．そこで図4(c)のように演算を組み合わせ，解像度の低い画像から求めたエッジ画像をマスクとして元の画像を処理する (図4(d))．図4(c)の演算を何段も組み合わせることにより，ピラミッド・データ構造を用いた処理が実現で

きる．

(2-2) パラメータ最適化型

一般に画像処理演算には多くのパラメータが含まれ，処理対象に応じてその値をうまく設定する必要がある．パラメータ最適化型の合成演算では，**図5**に示したように，O_2 に含まれるパラメータを逐次変化させながら処理を行い，その処理結果の中から O_1 の処理結果と最も整合性のあるものを選び出力とする．この合成演算の結果を D とすると，

$$D=O_2(D_2, p^*) \qquad (6)$$

と表せる．ここで p^* は2つの処理結果の間の整合性を評価する関数 $E(O_1(D_1), O_2(D_2, p))$ を最大にするパラメータ p を表す．

パラメータ最適化型の代表例として，2値化におけるしきい値の決定がある．**図6**(a)のようなコントラストの悪い画像を2値化する場合，通常のしきい値決定法では適切な値が求められない（図6(b)）．そこで，まず画像を微分・2値化して得られたエッジ画像を2値化の基準画像とする（図6(c)）．つぎに，しきい値を逐次変化させながら元の画像を2値化し，得られた2値画像における領域の境界線とエッジ画像との整合性を調べ，最も整合性の高い2値画像を出力とする（図6(d)）．

3. 画像処理演算の複合的合成機能を持つ対話型画像処理システム

2章での考察を基に画像処理演算の複合的合成機能を持った対話型画像処理システムの設計・試作を行った．その設計方針はつぎのとおりである．

(1) すべての画像処理演算を関数として表現し，画像処理用の関数型プログラミング言語のインタプリタを作成する．

(2) 上で述べた3つのタイプの合成演算を行う(メタ)関数を用意し，複雑な画像処理過程が記述できるようにする．

(3) 処理によって得られる各種の画像特徴に型を与えるとともに，関数に対しても型宣言を行い型の不一致によるエラーの自動検出を行う．

(4) 画像データを記憶するメモリ領域の自動管理機能を実現し，処理の中間結果や作業領域の割り当てをシステムが自動的に行う．

システムは UNIX ワークステーション USTATION-E15（カラービットマップ付）上でC言語を用いてインプリメントされている．

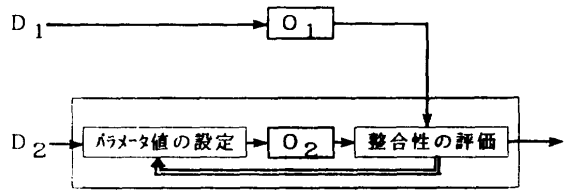

図5 パラメータ最適化型演算の処理構造
Fig. 5 Analysis process of the parameter optimization.

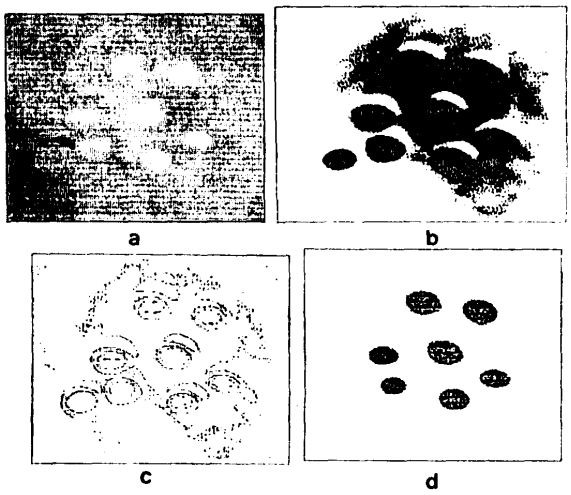

図6 しきい値の最適化
Fig. 6 Optimization of the threshold value.

3.1 システムの基本機能

現在，システムには前処理，線画抽出，領域分割に関連した画像処理関数が約40用意されている．

(代入演算子)

本システムでは関数型言語によって画像処理過程を記述するため，

FUNCTION (DATA, PARAMETERS)! (7)

と入力した場合，実際に処理は行われるがその結果は残らない．(!は入力文の終了記号) すなわち，システムは，関数 FUNCTION による処理結果を記憶するために確保したメモリ領域を処理の終了とともに解放してしまう．これを防ぐには代入演算子を用いて処理結果に名前を付ける．すなわち，

NAME=FUNCTION (DATA,
PARAMETERS)! (8)

とすればこの文の評価が終わってもNAMEによって処理結果を参照できる．

(複数の文の逐次的実行)

2章で述べたように，画像処理演算の逐次的連結

は，ある関数による処理結果を他の関数の引数とすることによって表現できる．しかし，非常に複雑な画像処理過程を関数の埋め込みのみによって表現するとプログラムの可読性が損なわれる．そこで本システムでは，つぎのような構文によって複数の文の逐次的実行が行えるようにしている．

```
OUT1=FUNCTION1 (DATA);
OUT2=FUNCTION2 (OUT1);          (9)
・・・・・・・・・・・・・・・・ !
```

当然のことながら，こうしたプログラムをファイルに書いておきそれを実行させることもできる．

3.2 データタイプ

本システムでは，以下に示す3つの項目によって画像データのタイプを定めている．

（1） データ形式

GRAY_PIC：多値画像　BINARY_PIC：2値画像
REAL_PIC：実数画像　TABLE：テーブルデータ

（2） データサイズ（解像度）

（3） 情報形式

GRAY_VAL：明度値　GRAD_VAL：1次微分値
GRAD_DIR：1次微分方向
LAPL_VAL：ラプラシアン値　LABEL：ラベル
REGION：領域　LINE：線　POINT：点

本システムでは，

画像のデータタイプ＝(データ形式)×(データサイズ)×(情報形式)

と考え，いずれかの項目が異なっておれば，異なったタイプであると考える．これは同じ多値画像でも各画素の持つ値の意味によって適用できる演算が異なるからである．たとえば，細線化のための関数では，その入力データのタイプは，BINARY_PIC かつ LINE (REGION) であることが要求され，このタイプ以外のデータを与えると，タイプエラーとなる．

3.3 複合的合成用関数

システムには結果合成型，マスク処理型，パラメータ最適化型の3種類の画像処理演算の複合的合成を行うための関数として以下のものが組み込まれている．

(結果合成型)

結果合成型の複合的合成用関数は，

$$\text{combine}\,(D_1, D_2, \cdots, D_n \text{ by } C) \qquad (10)$$

と表す．ここで，D_1〜D_n は画像データ，C は結果合成演算名を表す．現在利用できる結果合成演算は，相加・相乗平均，論理演算，代数演算，部分結果の合成など10種類である．

(マスク処理型)

マスク処理型用の関数は，

$$\text{mask}\,(O(D) \text{ by } M) \qquad (11)$$

と表す．ここで O は画像処理用関数，D は処理対象画像，M は O の処理範囲を定めるマスク画像である．ただし，画像処理関数 O としては，マスク制御型の処理ができるものでなければならない．結果合成用，マスク処理用関数を用いると，図2の処理過程は

```
mask(region (D)
    by binarym(combine(region(D), sobel(D)
            by edge_count), threshold)))
                                        (12)
```

と記述できる．ここで，D は図2(a)の入力画像，region，binarym，sobel はそれぞれ領域分割，2値化（真偽値を返す），微分を行う画像処理関数，edge_count は各領域内のエッジ点の数を数える関数を表す．また，図4の場合は，

```
mask(binary(sobel(D), thre)
    by enlarge(binarym(sobel(compact(D)),
      thre)))                           (13)
```

と書ける．ここで，compact，enlarge はそれぞれ画像を1/2に縮小，拡大する関数，binary は通常の2値化である．((13)式において，binary(sobel(D), thre) の演算はマスク画像の値が真の画素のみに対して行われることに注意)

(パラメータ最適化型)

パラメータ最適化用の関数は，

$$\text{optimize}\,(O(D, \cdots, *, \cdots),\ n_1, n_2, n_3 \text{ by } De \text{ at } E) \qquad (14)$$

と表し，画像 De を評価基準として評価関数 E によって画像処理 $O\,(D, \cdots, *, \cdots)$ におけるパラメータを最適化することを意味する．ここで * は最適化すべきパラメータを表す．また，パラメータの値は初期値 n_1 から上限値 n_2 まで，増分 n_3 ずつ変化される．たとえば，図6の2値化におけるしきい値の最適化処理は，

```
optimize(binary (D, *), 40, 60, 2
        by binary (sobel (D), 3) at efunc1)
                                        (15)
```

と表される．ここで，D は図6(a)の画像，efunc1 は2値画像中の領域の境界線と微分画像から求めたエッジ点との一致度を評価する関数である．

以上いくつかの例で示したように，この関数型プログラミング言語を用いると，かなり複雑な画像解析過

程が簡潔に記述できる.

4. 画像処理エキスパートシステム LLVE における戦略的知識の表現

文献 1) で報告した画像処理エキスパートシステム LLVE は, ゴールに記述された制約条件を満たす画像特徴を画像から自動的に抽出するものであった. しかし, 1章で述べたように, このシステムにおいても実際に実行される画像処理過程は基本的な画像処理演算を逐次的に連結したもののみであった. ここでは, LLVE において2章で考察した3種類の画像処理演算の複合的合成を実現するための拡張について検討する.

4.1 超グラフを用いた知識の表現

LLVE における基本知識は図7のような有向グラフによって表現されていた. グラフの節点が画像特徴の型, 弧がある画像特徴を別の画像特徴に変換する画像処理演算を表す. システムはゴールで指定された画像特徴を表す節点と入力データを表す節点とを結ぶグラフ上の経路を探索し, 経路中の弧が表す画像処理演算を実行する.

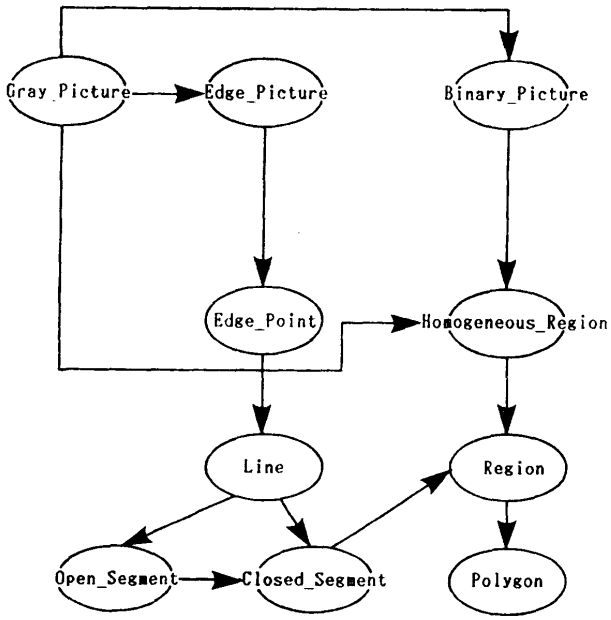

図 7 画像処理に関する知識を表すグラフ構造
Fig. 7 Graph structure for representing knowledge about image processing.

こうした知識表現法では, 図2や図4, 図6で示したような複数の異なった型の画像特徴を入力とする画像処理演算は全く表現できない. すなわち, 多くの画像解析戦略では, エッジ画像や, 領域分割画像, マスク画像といった型の異なった複数の画像特徴を入力としており, 図7のような単純なグラフではそうした演算を表現することができない.

この問題を解決するには, これまでのグラフ構造を超グラフに拡張すればよい. 超グラフでは, ある節点と節点の集合を結ぶものとして超弧が定義され, 複数の画像特徴を入力とする複合的な画像処理演算をそうした超弧によって表現することができる. たとえば, 図2, 6での演算を超グラフとして表現すると図8のようになる. ここで, 1本の超弧は円弧で結ばれた複数の線として表している. また, これまでの1入力1出力の画像処理演算を表す弧は, 1つの節点と要素が1の節点の集合とを結ぶ超弧として表現でき, 従来の知識表現の自然な拡張となっている.

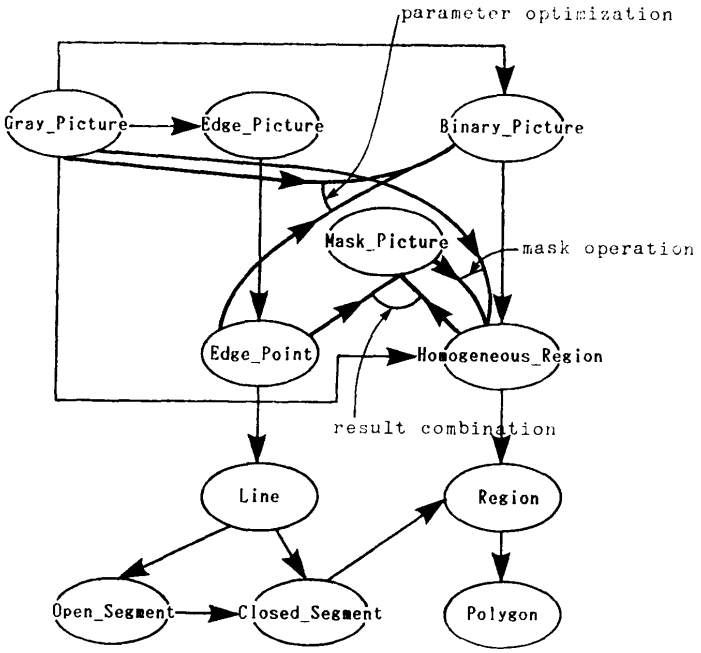

図 8 超グラフによる知識の表現
Fig. 8 Knowledge representation by hyper graph.

4.2 多重解像度の利用

上の議論では, 図4のような解像度の異なった画像を利用した処理は考えていなかった. LLVEにおいてはすべての画像特徴の解像度は同じであるとしていた

が，多重解像度の利用は画像解析の非常に有力な戦略の1つである．LLVEにおいて多重解像度を利用した解析を実現するには，画像特徴を拡張する必要がある．すなわち，図7の節点として表現される画像特徴は，2章での考察からデータの型を表すものと考えられ，解像度の異なるデータは異なった型のデータであるとみなすのが自然である（3.2節参照）．

以上の考察から，LLVEにおける知識構造を図9のように解像度軸に沿った階層的な超グラフとし，各階層では同じ解像度の画像特徴が超弧によって結ばれるとする．解像度の異なる層は，(13)で用いたenlargeやshrinkといった解像度を変化させる画像処理演算を表す超弧によって結ばれる．

ここで注意しなければならないことは，ほとんどの画像処理演算はデータの解像度と独立に定義されており，データの解像度が異なっても同一の画像処理演算が適用できるということである．その結果，各階層での超グラフの構造は同一のものとなり，図9の構造は多くの冗長な記述を含むことになる．この問題に対処するには，各節点の表す画像特徴の型をGray_Picture(s)のように，解像度を表すパラメータsを含んだ表現とし，同一の画像処理演算が任意の解像度において実行できることを表現すればよい．パラメータsは，同一解像度での画像処理演算では変化せず，enlargeやshrinkなどの解像度変化を引き起こす演算によって変化する．たとえば，エッジ検出演算（を表す関数）の型はGray_Picture(s)→Edge_Picture(s)，enlargeの型はGray_Picture(s)→Gray_Picture($2s$)となる．こうした表現によって，概念上の知識構造を表す階層的な超グラフが1層の超グラフによって記述できる（図10）．

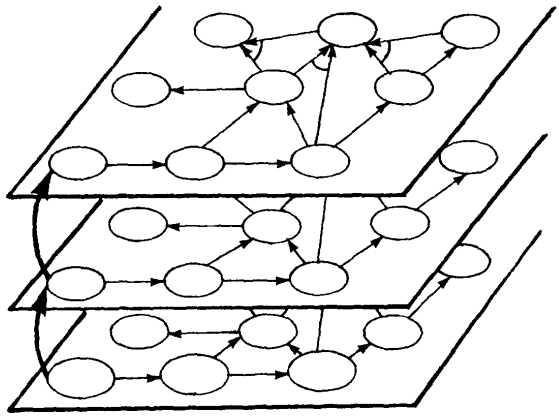

図9 階層的な超グラフ
Fig. 9 Hierarchical hyper graph.

4.3 探索過程の制御

従来のLLVEでは，ゴールが与えられた場合，図7のグラフ上での最適な経路の探索を行い，それを初期の処理方針とした．拡張されたシステムにおける処理方針の決定では，超グラフ上での経路の探索を行うことになる．こうした探索は，超グラフをAND/ORグラフとみなし，人工知能の分野でよく知られているAND/ORグラフの探索を行えばよい[6]．

最適な経路を選択するには処理コストの計算が必要であり，LLVEではそのためのコスト計算規則が各弧に付加されていた．拡張された知識構造を表す超グラフにおいても各超弧にコスト計算規則を付加しておけば，これまでと同様の方法によってコスト計算ができる．このとき，1本の超弧によって表現される複合的画像処理演算は実際には複数の基本演算から構成されており，それらの演算コストの総和として超弧のコストが計算される．

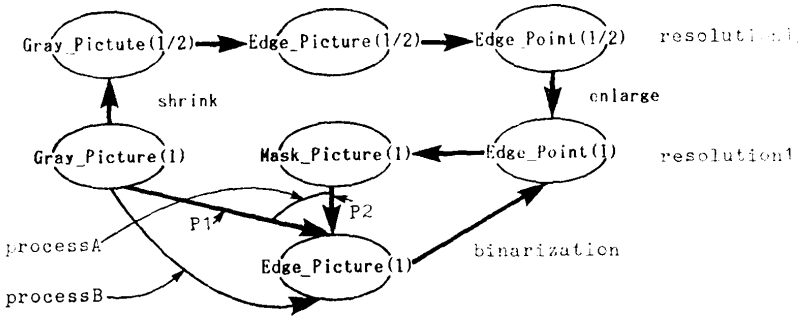

図10 多重解像度を利用した画像解析戦略の表現
Fig. 10 Representation of an image analysis strategy using multi resolution.

図4の画像解析過程は，図 10 のような超グラフ上の経路（太線の部分）として表現される．システムが処理方針決定のために経路探索を行う場合，図 10 のような経路がうまく選択できるようにするには，画像処理演算の選択や依存関係を表す規則を用いればよい[1]．

今ゴールとして抽出したい画像特徴が Edge_Point (1)，入力データが Gray_Picture (1) であるとする．システムは，ゴールである Edge_Point (1) の節点から逆向きにグラフの探索を開始する．図 10 では，enlarge と binarization の演算がこの節点に入ってくる弧であるので，この両者，あるいはいずれか一方を選択する．ここで“enlarge を行うにはマスク演算を行う process A がすでに選択されている必要がある”という演算の選択規則を Edge_Point (s) の節点に書いておくことにより，binarization の演算のみが選ばれる．これによって探索が Edge_Picture (1) の節点に進む．この節点に入る弧は process A と process B があるが，画像データの性質やゴールの条件からこの節点に付けられた演算選択規則によって process A が選ばれたとする．process A は入力を2つ持つ複合演算であるため，その部分演算を表す p1 と p2 の弧が探索される．p1 の弧をたどると入力データである Gray_Picture (1) に到着し，探索は終了する．一方，p2 をたどると再びゴールである Edge_Point (1) に戻るが，今度は探索中の経路に process A があるため，enlarge が選択される．(binarization を選ぶと，探索中の経路に同一の演算が重複して含まれることになるため，この弧は選ばれない．) このとき，enlarge の弧に“この演算を行うには shrink をまず実行しておく必要がある”という演算の依存関係規則を付けておくと，Gray_Picture (1)→Gray_Picture(1/2) という部分的な経路が探索中の経路に取りこまれ，Edge_Point(1/2) と Gray_Picture(1/2) を結ぶ経路の探索が起動される．

以上のように，演算の選択や依存関係を記述した経路を利用することによって，様々な画像処理の複合的演算を含む複雑な画像解析過程を実現することができる．また，LLVE では，処理が失敗した場合に処理をやりなおすための制御用規則が各弧に付加されていたが，これらに関してもそのまま利用できる．

5. 考　　察

本論文では，複雑な画像を効果的に解析するための画像解析戦略が基本的な画像処理演算の高度な組合せとして表現できることを示し，関数による画像処理演算の表現，画像処理演算の複合的合成のための3種類の基本形式を提案した．また，これらの考察に基づき，簡単な関数型言語を用いた対話型画像処理システムを試作した．さらに，画像処理エキスパートシステム LLVE において，そうした画像処理の複合的合成が実行できるようにするための拡張として超グラフを用いた知識表現法を検討した．

プログラム・モジュールの組合せによる複雑なソフトウェアの作成という観点から考えると，今回行った考察や試作したシステムの機能は非常に原始的なものであると言わざるを得ない．今後の検討課題としては次のようなものが挙げられる．

（1） 関数合成に関する統一的な枠組の開発

今回のシステムでは，combine や mask，optimize といった関数によって基本関数の組合せを記述したが，そうしたマクロ命令的なものではなく，より体系的な関数合成の記述法を開発する必要がある．その際，多様な関数合成演算子を持った FP[7] や高階の関数型言語 PEBBLE[8] などにおける関数合成の考え方が参考になる．

（2） データタイプの体系化

今回の検討では，画像データのタイプを考えたが，それらはごく基本的なものに限られていた．今後は，画像処理の分野において用いられる多種多様なデータに対する系統的なタイプ構造を考える必要がある．このことは，画像処理エキスパートシステムにおける知識の構造的表現にとっても重要な問題である．

（3） 画像処理演算の複合的合成法の拡充

今回は，画像処理演算の複合的合成法として3つの基本形式を提案したが，これらによってすべての画像解析過程が記述できるのかどうかについては，今後さらに検討を進める必要がある．

画像処理エキスパートシステムの機能向上という観点から考えると，4章で検討した拡張によって LLVE の能力が以前より向上することは確かであるが，実際に複雑な画像を解析するには多くの問題がある．その代表が処理結果の評価法である．今回の検討によって，システム内に種々の機能を持った画像処理結果の評価関数を導入する枠組が実現されたが，具体的にどのような評価関数を利用すればよいのか，あるいは評価の基準を何に求めればよいのか[2]といった問題を解決しなければ真に役立つ画像処理エキスパートシステ

ムの実現は望めない．そのためには，画質の評価法，形の類似性，処理結果のパラメータ依存性の解析などに関する具体的な画像処理の知識の蓄積がまだまだ必要であろう．

参考文献

1) 松山，尾崎：LLVE：トップダウン・セグメンテーションのための画像処理エキスパートシステム，情報処理学会論文誌，Vol. 27, No. 2, pp. 191-204 (1986).
2) 久保田，長谷川，鳥脇：サンプル図形提示による線図形および面図形抽出手順の自動構成方法の実現，情報処理学会研究会資料，CV42-5 (1986).
3) 坂上，田村：処理モジュールの構造的知識を利用した画像処理プログラム自動生成システム，情報処理学会論文誌，Vol. 26, No. 4, pp. 652-661 (1985).
4) 田村ほか：ポータブル画像処理ソフトウェア・パッケージ SPIDER の開発，情報処理学会論文誌，Vol. 23, No. 3, pp. 321-328 (1982).
5) 松山，村山，伊藤：画像処理演算の複合的合成，情報処理学会研究会資料，CV43-4 (1986).
6) 白井，辻井，佐藤（共訳）：人工知能の原理，日本コンピュータ協会，東京 (1983).
7) Backus, J.: Can Programming be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs, *CACM*, Vol. 21, No. 8, pp. 613-641 (1978).
8) Burstall, R.: Programming with Modules as Typed Functional Programming, *Proc. of Int. Conf. of 5th Generation Comp. Sys.*, pp. 103-112 (1984).

（昭和62年6月4日受付）
（昭和62年9月9日採録）

松山 隆司（正会員）

昭和51年京都大学工学部修士課程了．昭和51年4月～60年7月京都大学工学部助手．昭和60年8月より東北大学工学部情報工学科助教授．工学博士．昭和57～59年米国メリーランド大学客員研究員．画像理解，人工知能，並列処理に興味を持っている．昭和55年情報処理学会創立20周年記念論文賞．

村山 直樹（正会員）

昭和36年生．昭和59年東北大学工学部電子工学科卒．61年同大学院修士課程修了．同年日本電気(株)入社．以来，C＆C共通ソフトウェア開発本部にてプログラム開発環境の研究開発に従事．

伊藤 貴康（正会員）

昭和37年京都大学工学部電気卒．同年三菱電機入社．昭和40～43年米国スタンフォード大学計算機科学科．昭和43～52年三菱電機中研．昭和53年以来，東北大学工学部．現在，情報工学科教授．工学博士．現在の研究の関心は，ソフトウェア基礎論，人工知能と意味論，並列計算構造と LSI など．昭和39年度稲田賞．