コンピュータビジョン 43－4
（1986. 7. 17）

画像処理演算の複合的合成

松山　隆司　　村山　直樹　　伊藤　貴康

（東北大学工学部情報工学科）

一般に複雑な画像を解析し、意味のある特徴を抽出するには、様々な画像処理演算（オペレータ）をうまく組合せる必要があり、これまでに様々な手法が考えられてきた。これは、多くの場合単独の画像処理演算では完全な結果を得ることがむずかしいため、領域解析とエッジ解析を組合せたり、ピラミッド構造を利用した多重解像度での解析を行うことにより処理精度や信頼性の向上を図ろうというものである。ここでは、こうした画像処理演算の高度な組合せを画像処理演算の複合的合成と呼ぶ。本報告では、まず画像処理演算の複合的合成を考える基礎として、関数を用いた画像処理過程の記述の有用性について議論する。つぎに、種々の複合的合成の実行形式を分析し、それらを表現するための３つの基本形式を提案する。さらに、画像処理演算の複合的合成機能を持った簡単な関数型プログラミング言語のインタープリタを設計、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。

Combination of Image Processing Operators

Takashi MATSUYAMA　Naoki MURAYAMA　Takayasu ITO

Department of Information Engineering
Tohoku University
Sendai, Miyagi 980 Japan

It is well known that we have to combine various image processing operators to analyze complex images. This is because we usually cannot obtain reliable analysis result by a single operator. To increase the reliability and accuracy of the analysis, many methods of combining operators have been devised: integration of region-based and edge-based analyses, image processing using the pyramid (multi-resolution) data, and so on.

In this paper we first discuss the utility of describing image analysis processes in terms of functions. Then, we introduce three fundamental types of combinations of image processing operators, and describe a simple functional programming language with programming constructs to represent the three combination methods. Several examples will be given to demonstrate how compactly we can describe complex image analysis processes by using this language.

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た。我国においては、ＳＰＩＤＥＲやＳＬＩＰなどの画像処理ソフトウェア・パッケージが開発され、画像処理実験のためのソフトウェア環境の改善に役立っている。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（プログラム）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）をうまく組合せる必要がある。たとえば、複数のエッジ抽出オペレータを組合せることにより雑音に強いエッジ抽出を実現したり、領域分割とエッジ抽出を組合せ領域分割の誤りを修正したり、あるいはピラミッド・データ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多くの実験例によって明らかになっている。ところが、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、演算の逐次的な連結しか許されておらず、柔軟な画像処理演算の組合せが実現できないのが現状である。

本報告では、上で述べたような高度な画像処理手法の組合せを画像処理演算の複合的合成と呼び、それらの実行形式を分析し、複合的合成の３つの基本形式を提案する。さらに、画像処理演算の複合的合成機能を持った簡単な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。

以下では、まず２章において、画像処理演算の複合的合成を考える準備として、関数を用いた画像処理演算の表現について議論する。３章では、画像処理演算の複合的合成の３つの基本形式を、それぞれ実例を示しながら説明し、複雑な画像処理過程がこれらの基本形式の組合せによって表現できることを示す。４章では、画像処理演算の複合的合成機能を持った簡単な関数型プログラミング言語のインタプリタの機能を説明する。

２．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合せ、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。

[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もの（関数）と考えるのは自然であろう。画像処理演算を関数と考えると、通常の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。

多くの場合、画像処理プログラムはＦＯＲＴＲＡＮやＣなどの手続き型言語で書かれるが、ＳＰＩＤＥＲやＳＬＩＰなどに含まれる画像処理用サブルーチンのレベルで考えると、手続き的に記述された内部のプログラムが隠され、個々のサブルーチンを関数と見做すことができる。以下では画像処理演算の複合的合成という観点から、従来サブルーチンとして表現されていた画像処理演算を関数として表現することの意義とその有用性について議論する。

２．１．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)と表される。こうした関数表現の第一の利点は、演算の出力を（出力用変数によって）明示的に示さなくてもよいということにある。すなわち、画像解析においては、いくつかの演算を逐次的に組合せることが多いが、関数表現を用いることによりそうした場合の中間結果をいちいち明示的に記述する必要がなくなる。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)          (1)
CALL CLAB(JP2,LAB,ISX,ISY,・・・・・・・・)
```

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。(1)のプログラムにおいてEGPR、SLTH1の出力用変数JP1，JP2は処理の中間結果を表しており、処理過程の記述という観点からは不要のものである。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、不要な中間結果を表す必要がない。ここで=は代入を表す。

このように関数型言語を用いると、処理過程の途中で現れる中間結果に一々適切な名前を付ける必要がなく、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を大幅に向上できる。すなわち、関数型言語では演算をどのように適用していくかという処理過程自身が簡潔に表現できる。（同様のことが論理型言語ＰＲＯＬＯＧに対する関数型言語の優位性の１つして議論されている。）

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。たとえば、(1)のプログラムでは、中間結果を格納する配列JP1はSLTH1の処理が終了すれば不要であるにもかかわらず、そのことを処理系に知らせる手段がなく、無駄な記憶領域を処理が終わるまで確保しておかねばならない。一方、(2)の関数表現では、EGPRの出力結果を記憶するために一時的に確保したメモリは、必要がなくなれば直ちに解放され、新たなデータの記憶のために使われる。具体的には、１つの画像の大きさを64KBとすると、(1)では256KBのメモリが必要であるのに対し、(2)では192KBのメモリで済むことになる。

２．２．型の導入

（型を用いた整合性の検証）

一般に画像処理演算は、ある特定のタイプの画像特徴を別のタイプの画像特徴に変換する。たとえば、通常、上のCLABは２値画像を入力とし、ラベル画像を出力とする。つまり、各画像処理演算の入力として許される画像特徴のタイプは特定のものに限られており、それ以外のタイプの画像特徴を入力とすることは意味がない。したがって、画像処理演算を関数として表現した場合、その入力のデータ型と出力のデータ型を明示的に宣言しておくことにより、意味のない処理の指定を型の不整合（プログラムのエラー）として検出できる。([3]ではこうした観点からＳＰＩＤＥＲに含まれるサブルーチンの引数の型を分類、整理し、画像処理プログラムの自動生成に利用している。）

（構造を持つ型の利用）

上の(2)を見た場合、関数の引数にISX,ISYといった画像の大きさを表すものが含まれている。こうしたデータの属性を表す引数は、データ型の定義を行う際に宣言すればよく、関数の引数として与える必要はない。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;
```

のように定義し、(2)の関数EGPRの中で引数INの型を

```
VAR  IN : GRAY_PICTURE
```

と宣言すれば、EGPRのプログラムにおいてIN.ISXおよびIN.ISYによって画像データINの大きさを調べることができ、関数の引数としてISX,ISYを与える必要はない。（注：ここでの型宣言はMODULA-2による表記法を用いた。）こうした構造化されたデータ型を利用することにより、(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。

（多出力関数の表現）

一般に関数の出力（返す値）は１つに限られている。しかし、微分演算では勾配の大きさと方向を表す２枚の画像を出力する必要がある。こうした複数の出力を返す関数をうまく表現するためにも構造を持つデータ型が役立つ。たとえば、

```
TYPE
DIFFERENTIAL_PICTURE = RECORD
GRADIENT_PICTURE:ARRAY[1..256],[1..256]
                  OF INTEGER;
DIRECTION_PICTURE:ARRAY[1..256],[1..256]
                  OF REAL
END;
```

としてDIFFERENTIAL_PICTUREという型を宣言し、微分演算を行う関数がこの型のデータを返すようにすれば、関数の出力は１つでよいことになる。

（Polymorphicな関数）

(1)で用いたSLTH1という２値化ルーチンは、濃淡画像だけでなく微分画像やテンプレートとの相関を求めた結果を表す画像など多くの型のデータに適用できる。一般にこうした関数はPolymorphicな型を持つといわれる。こうした場合は、まずGRAY_PICTUREやGRADIENT_PICTUREといった型の上位の型としてMULTI_VALUED_PICTUREを定義し、関数SLTH1の入力としてMULTI_VALUED_PICTUREという型を宣言すればよい。こうした型の間に上位下位関係を定義し、上位の型が許される場合には下位の型も許されると考えることは、オブジェクト指向プログラミングにおける属性の継承と同じ意味を持つと考えられる。

２．３．関数の合成

一般的な関数型プログラミング言語を考える場合、関数自身を通常の数値や文字列と同じく一種の値として考え、関数を他の関数のパラメータとして用いることを許す高階の関数型言語がよく用いられる。こうした高階の関数型言語では、関数の合成が簡潔に表現でき、多くの基本関数の合成によって複雑なプログラムを作成することができる。事実、[4]では、プログラム・モジュールの組合せによる大規模ソフトウェアの開発のために型付きの高階の関数型言語を利用することを提案している。画像処理の分野でも、プログラム・モジュールの柔軟な組合せを考えるには、こうした高階の関数型言語が有効であると考えられる。

３．画像処理演算の複合的合成の実行形式

１章で述べたように、画像処理の分野では複数の画像処理演算をうまく組合せることにより信頼性の高い結果を得るための方法が多く提案されている。２章では、逐次的な画像処理演算の連結が、関数表現を利用することによって簡潔に記述できることを示した。ここでは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。

３．１．結果合成型

図１に示すように、画像処理演算Ｏ１～Ｏｎの処理結果を結果合成演算Ｃ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（Ｏ１(Ｄ１)，．．．．，Ｏｎ(Ｄｎ)）　(4)

で定義される。ここで、Ｏｉ(Ｄｉ)は画像データＤｉを画像処理演算Ｏｉで処理した結果を表す。

結果の合成演算Ｃの代表例としては、

(a)画素ごとの算術演算(相加平均、四則演算など)
(b)画素ごとの論理演算(AND，ORなど)
(c)部分結果の連結
(d)処理結果の検証

がある。

（演算自身の合成）

合成演算Ｃは基本的には複数の画像データを入力とする画像処理演算であるが、(a)や(b)の画素ごとの演算においては、結果的にＯ１やＯｎといった元の画像処理演算自身の合成を行ったことになる場合がある。すなわち、

Ｄ'　＝　Ｃ（Ｏ１(Ｄ)，Ｏ２(Ｄ)）　(5)

においてＯ１、Ｏ２を３ｘ３の空間フィルタリング、Ｃを画素ごとの相加平均とすると、

Ｄ'　＝　{$\overline{C}$(Ｏ１，Ｏ２)}(Ｄ)

と表せる。ここで$\overline{C}$（Ｏ１，Ｏ２）は演算Ｏ１、Ｏ２の$\overline{C}$による合成を表し（すなわち、Ｏ１、Ｏ２で用いる２つの３ｘ３の加重マトリクスの平均を取ったマトリクスによる空間フィリタリング）、その合成された演算をデータＤに適用することを表す。同様のことが２値画像を対象とした論理演算に対してもいえる。

こうした画素ごとの演算の組合せは、新たなフィルタの設計、特徴抽出、異なったセンサからのデータの組合せなど種々の目的に利用できる。たとえば、図２(a)(b)(c)は、照明の方向を変えて撮った画像、図２(d)(e)(f)はそれぞれの画像から抽出したエッジを表す。これらの図から明らかなように、いずれの画像においても積木の稜に対応するエッジには過不足がある。そこで、つぎのような合成演算を行う。

ＯＲ（　ＡＮＤ(図２(e)，図２(f)），図２(d)）

この合成演算によって影線が除かれ、積木の稜に対応した全てのエッジがうまく検出される(図２(h))。

（部分結果の合成）

画像をいくつかの範囲に分割し、それぞれの範囲で異なった処理を行い最後にそれらの結果を合成するという処理が有効なことがよくある。このような場合、各範囲で求まった部分的な結果を（空間的に）合成するためにも結果合成型の演算が利用でき

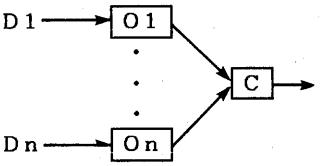


図１　結果合成型演算

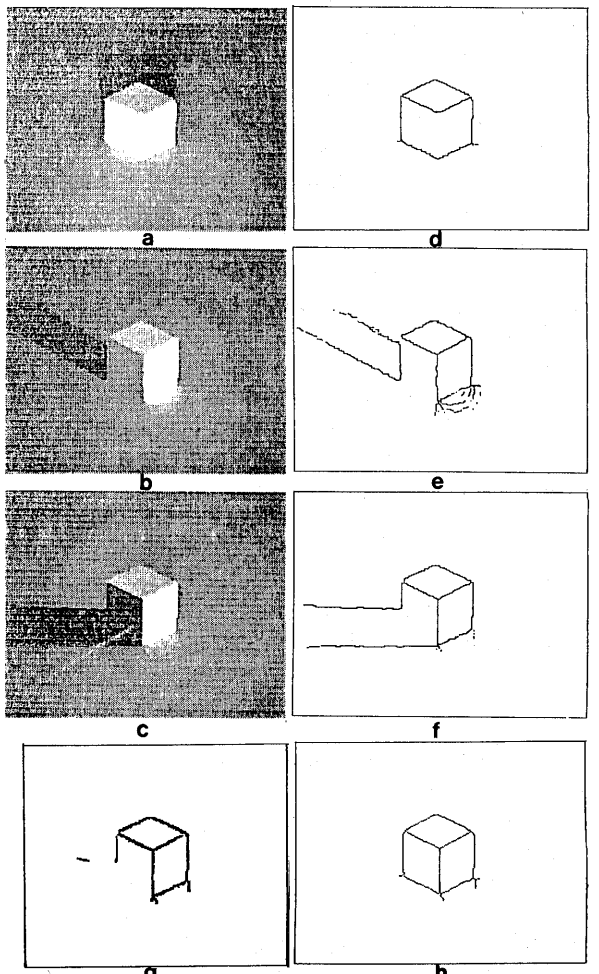


図２　多方向からの照明を用いた画像の解析

る。このタイプの合成演算は一種の"モザイク"処理を表し、３．２．のマスク処理型の合成演算と組合せて用いる。

（処理結果の検証）

ある１つの画像処理演算による処理結果の中には抽出したい特徴だけでなく不要なものが含まれていたり、抽出したい特徴が欠けていたりすることが多い。このような場合、別の画像処理演算による処理結果を用いて元の処理結果に含まれる特徴の信頼性を調べたり、未検出の特徴の発見を行うことが考えられる。こうした処理結果の検証は、画像データからの特徴抽出において重要な役割を果たす。

たとえば、図３(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。図３(b)(c)にそれらによる処理結果を示す。図から明らかなように、いずれの方法によっても完全な結果は得られない。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３(b)の各領域の中に含まれる（図３(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３(b)の例では、画面左下の折れ曲がった領域が２つに分割され、積木の面に対応する領域がうまく検出できた（図３(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３(b)の他の領域とを空間的に合成して図３(d)の画像を作った。

このほか、処理結果の検証の例としては、ラプラシアン(ゼロ・クロッシング)によるエッジの信頼性を１次微分の結果を用いて評価することなど多くのものが考えられる。

３．２．処理制御型

図４に示すように、この方式ではまず画像処理演算Ｏ１の処理結果を求め、その結果を用いて他の画像処理演算Ｏ２の実行の制御を行う。この処理制御型は、以下に述べるように(1)演算Ｏ２の空間的な適用範囲を制限するマスク処理型　(2)演算Ｏ２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。

３．２．１．マスク処理型

マスク処理型では、演算Ｏ１の処理結果（２値画像）を演算Ｏ２の処理用マスクとし、マスクの値が１の範囲に対してのみ演算Ｏ２を施す。Ｍ＝Ｏ１(Ｄ１)とすると、Ｏ２の出力Ｄは、

$$D = \begin{cases} O2(D2, M) & (mij = 1) \\ \text{undefined} & (mij = 0) \end{cases} \quad (6)$$

となる。ここで、ｍｉｊはマスクＭのｉ行ｊ列の要素である。

ここで注意しなければならないのは、この合成演算が、Ｃ(Ｍ，Ｏ２(Ｄ２))とは異なるということである。ここで演算Ｃは３．１．の結果合成型演算で画像Ｏ２(Ｄ)からマスクＭの１の部分のみを抜き出す操作を表す。たとえば、図３の例を考えてみると、その処理は注目した（エッジ点を多く含む）領域をマスクとしてその範囲内のみに対して（再帰的に）元の領域併合演算を施すというものである。すなわち、この領域併合演算では、処理に必要なパラメータを処理対象領域から決定しており、(6)のＯ２(Ｄ２，Ｍ)においてマスクＭに依存して処理結果（パラメータ）が変化する。

マスク処理型の合成演算を用いるとピラミッド・データ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５(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５(b)のように多くの雑音が検出されてしまう。そこで図５(c)のように演算を組合せ、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５(d)）。ここでもエッジ抽出演算は単なる画素毎の微分操作だけでなく、処理対象範囲の微分値ヒストグラムから求めたしきい値によって微分画像を２値化するというもので、処理対象範囲（マスク）が変われば処理結果が異なる。図５(c)の演算を何段も組合せることにより、ピラミッド・データ構造を用いた処理が実現できる。このほ

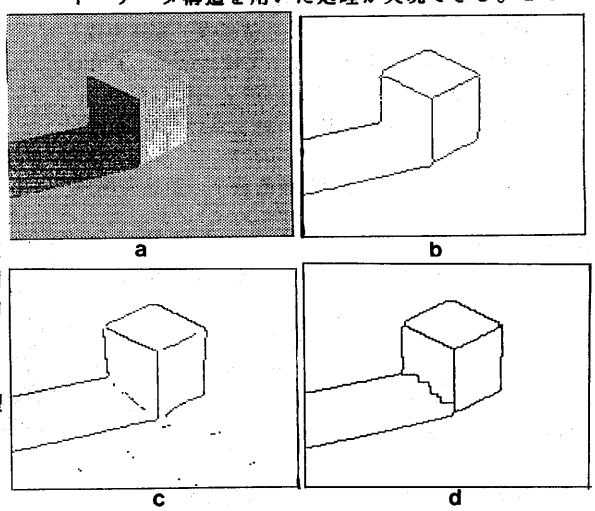


図３　領域分割とエッジ検出の組合せ

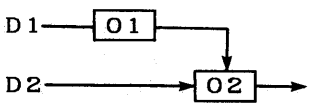


図４　処理制御型演算

か、再帰的しきい値処理による領域分割もこのマスク処理型の演算を用いることにより実現できる。

３．２．２．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においては、図４の画像処理演算Ｏ１の結果を評価基準として、演算Ｏ２に必要なパラメータの最適化を行い、最適なパラメータを用いた演算Ｏ２の結果を出力する。具体的には、図６に示したように、Ｏ２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Ｏ１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　＝　Ｏ２（Ｄ２，ｐ＊）　(7)

と表せる。ここでｐ＊はＥ（Ｏ１(Ｄ１)，Ｏ２(Ｄ２，ｐ)）を最大にするパラメータｐを表す。またＥは２つの処理結果の間の整合性を評価する関数である。

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７(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７(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７(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値画像と基準画像との整合性を次の評価関数によって調べ、最も整合性の高い２値画像を出力とする（図７(d)）。

$$\text{評価値}=\frac{\#\{(i,j)\mid \langle\text{基準画像において点}(i,j)\text{の値が1}\rangle \text{ かつ } \langle\text{点}(i,j)\text{が2値画像中の領域の輪郭線上に存在する}\rangle\}}{\#\{(i,j)\mid \langle\text{点}(i,j)\text{が2値画像中の領域の輪郭線上に存在する}\rangle\}}$$

ここで＃は集合の要素数を表す。

こうした基準画像との整合性を調べることによりパラメータの最適化を行うという考え方は[2]でも用いられている。

４．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、３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通りである。

（１）全ての画像処理演算を関数として表現し、画像処理用の（簡単な）関数型プログラミング

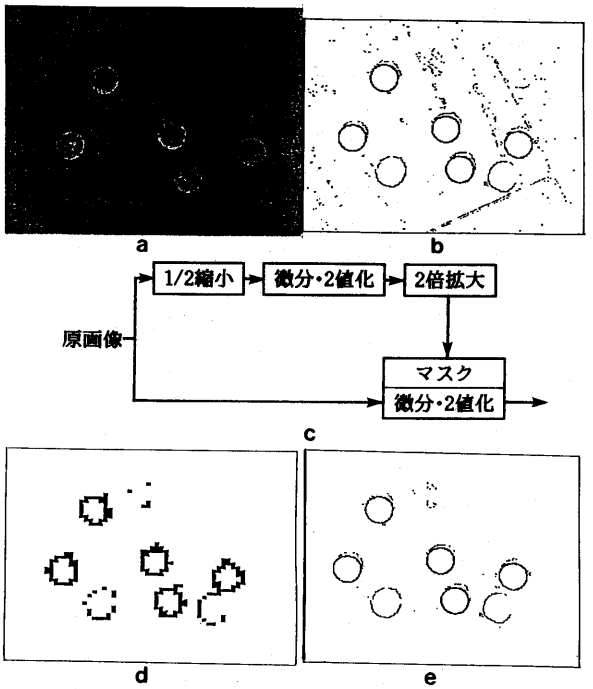


図５　多重解像度を利用した画像解析

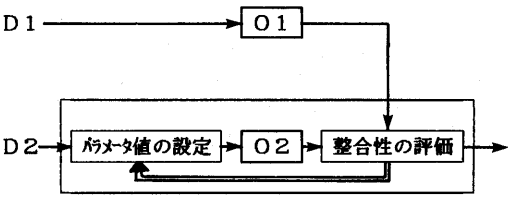


図６　パラメータ最適化型演算の処理構造

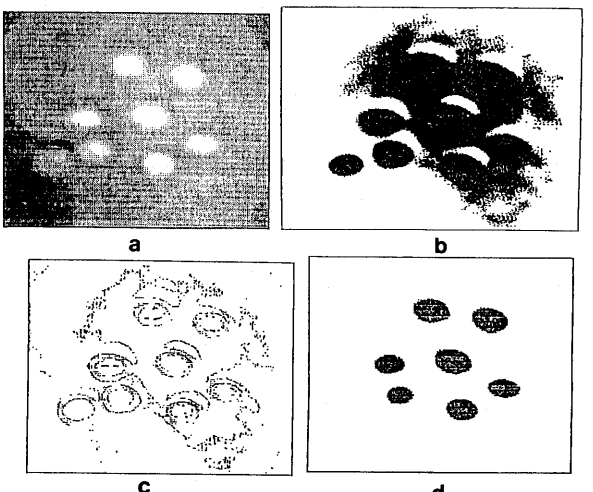


図７　しきい値の最適化

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ようにする。

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おり、システムの規模は約６０００行である。

以下では試作したシステムの機能について説明する。

４．１．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。このほか、システム関数としては、カラービットマップ・ディスプレイに画像を表示するdisplay、プリンタに画像を印刷するhardcopy、システムの画像データ領域に登録されている画像データのリストを表示するlist、画像データや関数の型を表示するtypeなどがある。

（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)！　　(8)

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)！　(9)

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);　　(10)

．．．．．．．．．．．．！

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。

４．２．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

TABLE :テーブルデータ

（２）画像サイズ

（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)Ｘ(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。また、複合的合成演算においても、合成できるデータの型の組合せが決められており、システムがタイプチェックを行う。

４．３．複合的合成用関数

３章での議論に基づき、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、

combine(D1,D2,...,Dn by C)　　(11)

と表す。ここで、D1~Dn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。図２に示した画像処理過程をこの関数を用いて記述すると

combine(combine(D2, D3 by AND),D1 by OR)　(12)

となる。ここで、D1~D3はそれぞれ図２(d)~(f)に示した画像である。

（マスク処理型）

マスク処理型用関数は、

mask(O(D) by M)　　(13)

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出来るものでなければならない。このマスク処理用関数を用いると、図３の処理過程は

```
mask( region(D)                                    (14)
      by extract_ununiform_region( region(D)))
```

と記述できる。ここで、Dは図３(a)の入力画像、regionは領域分割を行う画像処理関数、extract_ununiform_regionは画像Dの領域分割結果から内部に多くのエッジ点を含む領域を抽出する関数を表す（この関数は処理結果の検証を行う結果合成型演算を用いて記述される。）。このプログラムでは、まず図３(b)の左下の領域のみを表すマスクが生成され、その部分のみを対象として再び領域分割が行われる。また、図５の場合は、

```
mask( binary( sobel(D), 5)                         (15)
by enlarge( binary( sobel( compact(D)), 5)))
```

と書ける。ここで、compact、enlargeはそれぞれ画像を1/2に縮小、拡大する関数である。

（パラメータ最適化型）

パラメータ最適化用関数は、

```
optimize(O(D,..,*,..),n1,n2,n3 by De at E)(16)
```

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,..,*,..)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n1から上限値n2まで、増分n3ずつ変化される。たとえば、図７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)  (17)
```

と表される。ここで、Dは図７(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。

５．考察

本報告では、複雑な画像処理過程を簡潔に記述するための関数による画像処理演算の表現、および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。さらに、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。今後の検討課題としては以下のようなものがある。

［１］画像処理用高級プログラミング言語の開発

画像処理用のプログラム・モジュールの組合せによる複雑な画像処理過程の記述（画像解析用高級プログラミング言語の開発）という観点から考えると、今回行った考察、試作したシステムの機能ともまだまだ不十分なものであると言わざるを得ない。こうした観点からの検討課題としては次のようなものが挙げられる。

（１）評価式の直接的記述

combineやoptimizeの引数として用いられる評価用関数は、与えられた問題によっていろいろなものが考えられ、システムに予め登録されているものだけでは不十分であることが予想される。これを解決するには、評価用関数名にほかに評価式を直接書くことが考えられる。すなわち、

```
combine( D1, D2 by mean)
```

の代わりに

```
combine( D1, D2 by (D1+D2)/2)
```

と書ければ、プログラミング言語としてより柔軟なものとなる。

（２）データタイプの体系化

今回の検討では、主として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。

（３）言語機能の高度化

本稿での検討では、一応関数にもタイプ付けを行い、タイプチェック機能を実現した。しかし、[4]で述べられているように型付の高階の関数型言語による柔軟なプログラミングを考えるには、タイプ変数の導入、関数の抽象的な仕様記述法、関数定義用関数やマクロ関数の導入などより深い考察が必要である。

［２］画像処理エキスパートシステムの開発

今回は画像処理エキスパートシステム（画像処理技術に関する知識の表現と利用）の観点からは考察を行わなかったが、個々の画像処理演算の利用法、それらの有効な組合せ法に関しては多くの知見が得られている。ここで述べた画像処理演算の複合的合成においても、合成演算の選択や評価用関数の決定のために画像処理に関する知識やノウハウが利用できる。そうした知識やノウハウは画像解析における戦略的知識ということができ、それらをどのように表現、利用すればよいかは今後の大きな課題である。

［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、情報処理学会研資、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]R.Burstall:Programming with modules as typed functional programming, Proc. of Int. Conf. of 5th Generation Comp. Sys., pp.103-112, 1984

東北大学工学部情報工学科
伊藤研究室の概要

伊藤　貴康　　松山　隆司
（東北大学工学部情報工学科）

｛これは、昭和６１年７月に東北大学工学部で開催の情報処理学会コンピュータビジョン研究会において計画された研究室見学用の資料である。｝

伊藤研究室は、昭和５３年に通信工学科の研究室として始まり、昭和５９年からは新設の情報工学科の研究室として基礎情報学講座を担当している。

研究室の発足以来、ソフトウェアの基礎理論、ソフトウェアの自動作成、人工知能ソフトウェア、パターン認識と画像理解、計算機アーキテクチャ、ＬＳＩの設計と試作などを主なテーマとして研究を行ってきている。研究室は、現在、教授（伊藤）、助教授（松山）の他に、技官（大友）、事務補佐１人の職員と大学院および学部の学生からなり、大学院生および学部学生はそれぞれ毎年６人位である。昭和５６年５月から昭和５９年６月までは助教授として江刺正喜氏（現在、電子工学科助教授）が居り、カスタムＬＳＩの設計と試作に関する研究を行った。松山は、昭和６０年８月にこの研究室に着任後、画像理解の研究に加え、並列処理システム、知的プログラミング環境に関する研究を行っている。

Ｉ．研究の状況

研究テーマごとに、研究の状況を簡単に説明しておこう。

（Ｉ）ソフトウェアの基礎理論

並列／並行プロセスのモデル、並列／並行プログラムの意味論や公理系、プログラムの検証法とそのシステムなどに関する研究を行ってきている。

（ＩＩ）ソフトウェアの自動作成

定理証明に基づく自動合成、例題系列からの自動合成、プログラム変換、日本語による記述からのプログラム生成などの研究を行ってきたが、現在は、相補的プログラミングの考え方に基づく総合的プログラミング環境の実現に関心がある。｛研究室のSymbolics 3670は、この研究のために文部省の科学研究費によって設置したものである。｝

（ＩＩＩ）人工知能ソフトウェア

ＬＩＳＰのインタプリタとコンパイラを、｛庄内、岸本、藤本、今といった｝学生諸君に何回も作成してきて貰っており、その最新版がUSTATION上にMC68000のアセンブラで今君に作成して貰ったＬＩＳＰインタプリタ"ｔｕｌｉｓｐ"であり、研究室での実用システムとして使われている。｛"ｔｕｌｉｓｐ"は、言語仕様は、Ｆｒａｎｚｌｉｓｐと同じであるが、かなり高速な処理系となっている。｝ＰＲＯＬＯＧの処理系や（Ｐｏｐｌａｒ流の）関数型言語の処理系、項書き換え系による定理証明、ＦＯＬの試作やＬＩＳＰプログラム用対話型検証システムの試作なども試み、推論ソフトウェアに関する基礎的研究を行っている。ＰＲＯＬＯＧやＳＮＯＢＯＬのようなバックトラックを内蔵した言語の実行時における無駄なバックトラックの自動消去法やユニフィケーションの拡張についても研究している。

（ＩＶ）パターン認識と画像理解

カラー画像の領域同定および形状解析、線画の解析と理解、３次元物体の認識などの研究を毎年のように研究室で行ってきたが、今年は門馬君にステレオ視による物体認識の研究を始めて貰っている。また、画像理解の研究に関しては、画像理解と並列処理、knowledge-based vision に関する基礎的な研究を深めたいと思っている。

（Ｖ）計算機アーキテクチャ

ＬＩＳＰマシンを念頭に置いた高級言語マシン、マイクロプログラミング方式に関する研究を行ってきたが、将来の並列処理システムに関する研究を行うために、昭和５９年からは並列処理システム開発のための並列処理システムについての研究にも着手している。MC68000を用いた"PAI-68K"と名付けた並列処理システムが黒川君と（技官の）大友君の努力で動き始めている。このPAI-68Kは、現在、並列に動作する４台のMC68000が各々５１２ＫＢのローカルメモリを持ち、更に、それらが４ＭＢのメモリを共有する構成になっている。現在、プロセッサを８台にする努力が４年生の協力も得て進められている。また、PAI-68K上での並列ＬＩＳＰインタプリタを今君に作成して貰っており、ＬＩＳＰの並列実行方式の研究に使われつつある。門馬君のステレオ視の研究にもこの並列システムを使いつつある。さらに、英国ＩＮＭＯＳ社のトランスピュータを用いた人工知能用並列システムについても研究を始めている。

（ＶＩ）ＬＳＩの設計と試作

将来のコンピュータは、すべてＶＬＳＩ技術に基づく高度並列システムになることが予想される。ＶＬＳＩに関する知見を得ること、およびカスタムＬＳＩ製作環境を学内に作り、企業ではやれないａｃａｄｅｍｉｃ　ｃｈｉｐの研究ができないものかと考え、昭和５６年から昭和５９年にかけて、東北大学電気情報系

の協力を得て、設計試作環境を作ることを江剏助教授を中心に行って貰った。大変貧弱な環境ではあるが、小規模なＬＳＩであればｃｈｉｐが作れるようになり、電気情報系の一部で利用するようになっている。並列アーキテクチャの研究や画像理解の研究に基づく新しいＶＬＳＩ　ｃｈｉｐの可能性についても検討したいと考えている。

ＩＩ．研究室の計算機環境

研究室の主な計算機環境は、図に示すごとくである。

現在のところ、ハードウェア的には、

MC68000、Symbolics3670、（トランスピュータ）

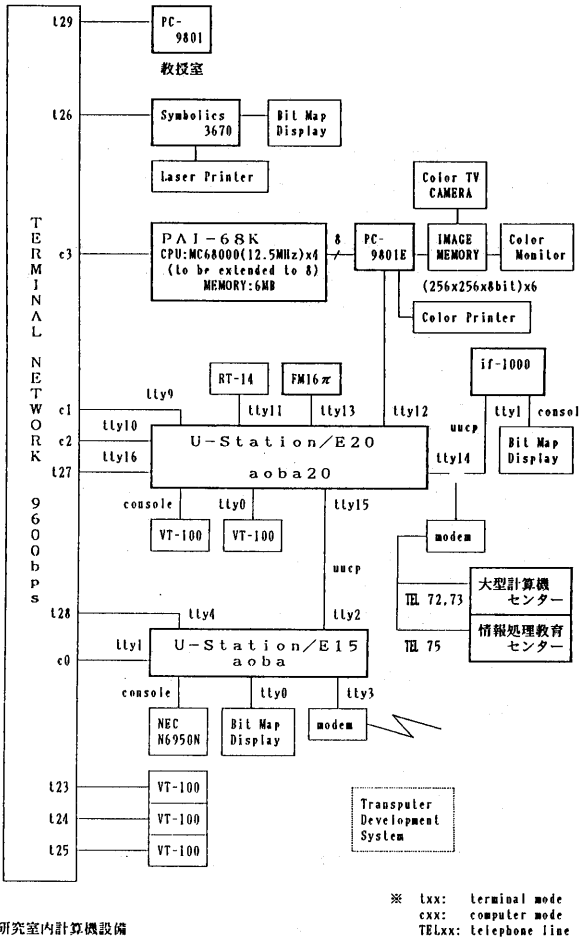
ソフトウェア的には、

LISP、UNIX、MC68000アセンブラ、（OCCAM）

および、いくつかのパーソナルコンピュータが主なツールである。

図に示したように、これらのコンピュータが同軸ケーブルを用いて研究室内でのＬＡＮを形成しており、それぞれの端末から研究室内のいろいろな計算機が利用できるようになっている。

研究室内にある画像処理用の装置としては、パーソナル・コンピュータPC9801を制御装置とするカラーテレビカメラ、画像メモリ、カラーモニタ、カラープリンタがある。これらは主に画像入出力のために用いており、実際の処理はUSTATIONやSymbolics3670で行っている。また、PC9801とPAI-68Kとの間は８ビット幅の線で結ばれており、PAI-68Kを画像処理に用いる場合は、PC9801を画像表示装置として直接利用することができる（256x256の画像で転送時間が約2秒）。この他、電気情報系学科のTOSPIX-II、東北大学大型計算機センターにある画像処理システム（NEC）も利用している。



研究室内計算機設備

研究室の主な計算機の構成

計算機	プロセッサ	メモリ	ディスク	端末装置
Symbolics3670	—	8MB	470MB	Bitmap Display,Laser Printer
USTATION E20	MC68020	4MB	160MB	
USTATION E15	MC68000	2MB	80MB	Color Bitmap Display
IF1000	MC68010	2MB	40MB	Bitmap Display
トランスピュータシステム	T414	2MB	—	
PAI-68K	MC68000x4	6MB	—	