小特集 「並列・分散協調画像理解」

並列画像理解のソフトウェアアーキテクチャ

Software Architecture for Parallel Computer Vision

谷口 倫一郎*
Rin ichiro Taniguchi

* 九州大学大学院総合理工学研究科情報システム学専攻
Dept. of Information Systems, Grad. School of Engineering Sciences, Kyushu University.

1994年4月27日 受理

Keywords: computer vision, image processing, parallel processing, parallel programming language, software system.

1. まえがき

高速な画像理解システムを実現するためには，システムに並列処理の機構を導入することが不可避となってきている[Computer 92]．実際に並列画像理解システムを実現するためには，並列処理のためのハードウェアだけでなく，並列画像理解のアルゴリズムを記述し，実行するためのソフトウェア環境が不可欠である．また，しっかりしたソフトウェア環境がなければ，並列処理ハードウェアを十分に使いこなすことができないだけでなく，信頼性の高いシステムを構築することもできない．そのような点で，並列画像理解のソフトウェアアーキテクチャは極めて重要なものであり，システムの構造が複雑になるほど，その重要性は高くなってくる．そこで本稿では並列画像理解システムを構築するためのソフトウェアアーキテクチャの現状について概説する．

一般に，並列処理プログラムを記述し，実行するためのソフトウェアとしては，並列プログラミング言語の処理系と並列処理の実行制御を行うオペレーティングシステムの二つが重要となってくる．オペレーティングシステムに関しては，特に画像理解に独自の機能という点からの研究はあまりなく，これまでの並列画像理解のソフトウェアに関する研究は，いかに並列画像理解のアルゴリズムを簡潔に記述するかという点に重点が置かれてきた．最近では，並列処理のハードウェアがかなり一般的なものになってきているため，画像処理のためのさまざまな並列処理プログラミング言語が設計・開発されてきている．したがって，以下では並列画像理解アルゴリズムの記述方式に関してこれまでに行われてきた研究・開発を中心に述べることにする．

2. 並列プログラム記述の枠組み

画像理解を対象とした場合に限らず，一般に並列プログラムの記述方式には，並列性の高いプログラムの効率的な記述を可能にするために考えなければならない重要な点がいくつか存在する．したがって，並列プログラミング言語がこれらの点に対してどのような枠組みを提供するかが重要である．そこで具体的な事例を紹介する前に，まずこれらの基本的な事項について述べる．

〔1〕 データ分割と機能分割

一般にプログラムの並列化の方法としては，大別してデータ分割による並列化と機能分割による並列化がある．データ分割は，データ集合をいくつかの部分集合に分割し，各部分集合の処理を並行して実行することでプログラムを並列化するものである．一方，機能分割は，ある一つのタスクを同時に実行可能な複数のサブタスクに分割し，それらを並行して実行することで並列化を行うものである．

画像理解の並列化について考えると，低レベルの画像処理（画像信号に近いレベルの特徴抽出処理）では，その多くが近傍処理であり画素単位で独立に処理できるため，データ分割に基づく並列処理が有効である．一方，画像理解の高レベル処理（信号を抽象化した，

準記号，記号レベルでの推論・認識処理）では，単純なデータ分割による並列処理だけでなく，機能分割による並列化も有効であると考えられる．特にシステムが高機能で複雑になる場合はその傾向が顕著であると考えられる．

したがって，並列プログラミング言語として，データ分割による並列化を記述するのか，機能分割によるものを記述するのかという二つのアプローチが考えられる．

〔2〕 **分散データ構造**

並列画像処理のプログラム作成には二つの側面がある．一つは，処理アルゴリズムそのものの並列化であり，もう一つは画像データや画像処理の結果得られる特徴データなどをどのように並列プロセッサ上に記憶させるかという問題である．共有メモリアーキテクチャを採用する場合には，特にデータの格納方式は問題とならないが，分散メモリアーキテクチャを利用する場合は，プロセッサ間の通信（データ転送）のオーバヘッドをなるべく少なくするようなデータの分散化を考えなければならない．例えば，すべてのプロセッサで同じデータのコピーを持っておくのか分割して持っておくのかといった問題や，分割の形態（分割領域の形状，大きさや重なりの有無）といった問題が出てくる．

プログラミング言語としても，抽象的なレベルでデータ分割の方法を指定するアプローチ[松山 93]と，プログラムのなかにデータ分割とそれに伴うデータ転送の手順を埋め込んでおくというアプローチがある．前者の場合は，プログラム記述は容易になるが，言語仕様がカバーしていないデータ分割については対応できないという問題があり，後者については自由度は大きいものの，記述が複雑になるという問題が出てくる．

〔3〕 **対象アーキテクチャ**

並列プログラミング言語は，プログラムが実行される並列計算機のアーキテクチャに多かれ少なかれ依存する．画像理解の領域では，低レベル処理のデータパラレルによる並列化に重点が置かれてきており，アーキテクチャもSIMD（各プロセッサが同一の命令を実行する方式）を採用しているものが多かったが[Duff 76, Hillis 85]，高レベルの処理まで並列化しようというシステムではSIMDでは十分ではなく，MIMD（各プロセッサで異なる命令が実行可能な方式）あるいはSIMDとMIMDの組合せを用いている[松山 93, Weems 90, 山元 91]．そのため，画像理解のための並列プログラミング言語の仕様も対象マシンのアーキテクチャによってかなり異なってくる．

また，上記以外の点としても，プログラミングを記述するための枠組みとして，従来よく利用されてきた手続き型言語の延長でよいのか，それとも関数型などの並列プログラム記述向きの言語体系がよいのかといった議論もある．

画像理解の並列システムは，画像理解（画像処理）のために設計・開発された並列プロセッサ上に実現される場合が多く，並列プログラミング方式も，対象とする並列プロセッサアーキテクチャを強く意識したものが多い．また，対象とするアーキテクチャを見れば，画像理解の並列化のポイントをどこに置いているかということも明らかになってくる．そこで，以下ではプログラミング言語が意識としているアーキテクチャという観点から整理して，いくつかの代表的な画像理解(画像処理)のための並列プログラミング言語を紹介する．

3. SIMD型の実行モデルのための記述体系

画像理解における並列処理の多くは低レベル処理のデータパラレルを意識したものであり，その実行モデルとしては各画素に対して同一の処理を並行して行うというSIMD的なものを採用している*1．また，そのためのプログラム記述の枠組みも各種提案されているが，ここでは，Image Algebraという数学的なモデルに基づくプログラム記述体系について紹介する．

Image Algebraは画像とテンプレートという二つのデータを対象とする代数系であり，この二つのデータの上にさまざまな演算を定義しているものである[Ritter 90]．ここでいうテンプレートは，近傍の形と重みの配置を定義したものである．

定義されている演算としては基本的には

（1）〈画像，画像〉→画像

（2）〈画像，テンプレート〉→画像

（3）〈テンプレート，テンプレート〉→テンプレート

の3種類である．基本的な演算としては，（1），（3）に関しては四則演算など，（2）に関しては表1に示すような畳込みなどの演算が定義されている．また，画像の画素単位の特徴や画像全体の特徴を計算する関数も定義されている．

実際に言語としてImage Algebraを実現したものとしては，SIMD型並列計算機MasPar上に実現した

*1 MIMDアーキテクチャ上でも，各プロセッサは同じプログラムを実行するというSPMD（Single Program Multiple Data）モデルを採用しているものが多い．

表 1 Image Algebra における主な演算子

演算子	演算内容
A *+ B	Generalized Convolution
A *\/ B	Multiplicative Maximum
A */\ B	Multiplicative Minimum
A +\/ B	Additive Maximum
A +/\ B	Additive Minimum

```
Procedure Sobel (Var Src, Dst: ImageType);
  Var
    SHorz, SVert: Template of [3, 3] of
                  Integer;
  Value
    SHorz = [[-1, -2, -1]
             [ 0, [0], 0]
             [ 1,  2,  1]];
    SVert = [[-1,  0,  1]
             [-2, [0], 2]
             [-1,  0,  1]];
  Begin
    Dst := abs (Src *+ SHorz)
         + abs (Src *+ SVert);
  End;
```

図 1 IAL によるプログラム記述

もの[Meyer 91]，Transputer を用いた MIMD 型並列計算機上に実装した IAL[Morrow 92]がある．前者は C を，後者は Pascal をベース言語としている．図 1 に IAL で記述した Sobel フィルタの例を示す．このプログラムは，テンプレートの定義とテンプレートと画像の畳込み（演算子＊＋で示している）を行う実行文からなっている．テンプレートの定義中の中央にある[0]は，畳込みの中心の画素を示している．また，さまざまな形のテンプレートを定義できるよう，非参照画素を設定することができる．テンプレート定義のなかに“～”が現れた場合は，その画素は参照しないことになっている．さらに，IAL では新しい演算子を定義して用いる仕掛けも提供している．

上記以外にも，セルオートマトンをベースにした記述言語[Leite 92, Pfeiffer 90]などが提案されているが，紙面の都合上割愛する．

4. MIMD 型の実行モデルを意識した記述体系

画像理解の高レベル処理においては，単純なデータ分割による並列化よりもむしろ機能分割による並列化や協調処理の重要性が高くなると考えられる．したがって，それらを実行するのに適したアーキテクチャのモデルは MIMD 型になるが，いまのところ MIMD 型の実行モデルを意識した記述体系に関する研究はあまりなされていない．このような点から，筆者らのグループでは画像理解の低レベル処理から高レベル処理までの記述を行えるような並列プログラミング言語を目指して，Valid-A とその拡張について研究を進めている[Taniguchi 91, 谷口 93]．ここではそれについて簡単に紹介する．

Valid-A は筆者らのグループで研究を進めている超並列画像処理プロセッサ AMP[山元 91]のためのプログラミング言語であり，関数型をベースにしている．関数型言語は，副作用がなく参照透明性があるため，プログラムの構造がわかりやすい，高階関数を用いた抽象度の高いプログラムが記述できるといった利点があり，画像処理プログラムの記述にも用いられるようになってきている[Kozato 92, Poole 92]．並列処理の観点から見ると，関数型言語で書かれたプログラムからは，プログラムに内在する並列性を自動的に抽出しやすいといった利点があり並列プログラムを記述するのに適している．Valid-A では，上記のような関数型の特徴を生かしつつ，

- データ分割による並列処理，機能分割による並列処理のいずれも記述可能である．
- 単純なメッシュ構造だけでなく，ピラミッド構造も扱うことができる．

といった特徴を持たせている．

まず画像のデータパラレル処理の記述であるが，Valid-A では一つの関数が一つの画素値を計算するものとし，画像を処理するプログラムはその関数の集合体（関数配列と呼ぶ）で表現する．近傍演算を記述する場合は，近傍の関数要素にある値を参照する必要が出てくるが，Valid-A ではリモート変数と呼ぶ関数要素を指定した変数を記述することができるため，陽にプロセス間通信を記述しなくてよく，簡潔な記述が可能になっている．関数配列を用いたエッジ検出記述の例を図 2（a）に示す．図 2（a）中の太字がリモート変数を示している．関数配列では，条件式を用いることによって要素によって異なった演算を行うことが可能になっており，必要に応じて不均質な処理も容易に記述できるようになっている．また，関数配列の要素を実際のプロセッサエレメントにどのようにマッピングするかは，プログラマが指定することもできるようになっている．関数配列を呼び出すための記述は，図 2（b）のようにマクロ的な処理の流れを記述するだけでよい．

一方，機能分割による並列化の記述であるが，これは通常の関数型言語の枠組みを利用して行うことができる．すなわち，変数の依存解析を行うことにより，

```
array of function Edge1[256][256](x:integer)
         return(integer)
---  for PE's in inner region
for each function (i, j)
{
   if (i>0 & i<255 & j>0 & j<255) then
      Edge1[i][j](x) =
       { let
           center = x + Edge1[i-1][j].x +
                    Edge1[i+1][j].x,
           left   = Edge1[i-1][j-1].x +
                    Edge1[i][j-1].x +
                    Edge1[i+1][j-1].x,
           right  = Edge1[i-1][j+1].x +
                    Edge1[i][j+1].x +
                    Edge1[i+1][j+1].x
         in if (center-right) > (center-left)
            then center-left
            else center-right
       }
   --- for PE's on the border
   else if (i=0 & j>0 & j<255) then
     Edge1[i][j](x) = 0
   . . . . . .
}
```

（a） 関数配列の記述

```
function main() return(signal)
 =
{ let
    image1 = Read_image("Camera0"),
    image2 = Edge1(image1),
    image3 = Edge2(image1)
in
    Display_image(Add(image2, image3))
}
```

（b） 関数配列の呼出し

図 2 Valid A によるプログラム記述

並列に実行可能な式（関数，関数配列など）を抽出し，並列実行させることができる．例えば，図2（b）の例でいうと Edge1 と Edge2 に依存関係はなく，並列に実行される．Valid-A の実行対象マシンである AMP はデータフローを用いたプロセッサであり，命令レベルの並列性を自動的に抽出することができるため，このような並列プログラムを効率良く計算することが可能になっている[Yamamoto 93]．

そのほかに MIMD での実行を前提とした処理系としては，Massachusetts 大学の並列画像理解システム IUA 上に Linda[Carriero 89]を実装した例などがある[Bhalla 90]．

5. アーキテクチャに依存しない言語

従来の画像理解（処理）のための並列プログラミング言語は，多くの場合，特定のハードウェアアーキテクチャを対象としたものであった．したがって，同じアルゴリズムを記述したものであっても，他のアーキテクチャ上でそのまま実行させるということは極めて難しい状況にあった．このことは，ソフトウェア資産の共有という点では望ましいことではなく，画像理解の分野での並列処理の普及という点で一つの阻害要因となっている．

この並列画像処理ソフトウェア資産の共有という問題を解決するために，Webb らは新しい画像処理アルゴリズムの記述言語 Apply と Adapt を提案している[Wallace 88, Webb 92]．その背景にある考え方は，「画像処理やコンピュータビジョンの分野でよく用いられる並列アルゴリズムの種類は限られており，それらを表現するためには汎用的なプログラミング言語を利用する必要はない．むしろ問題のクラスに応じた専用記述言語のほうがコンパイラの能力(最適な並列化)やコンパイラの開発コスト，プログラミングの容易性といった点で望ましい．重要な点は，記述言語を対象アーキテクチャから独立させることにより並列画像処理のソフトウェア資産の共有を進めることである」というものである．

Apply は平滑化やエッジ検出などの局所的な低レベル画像処理演算のみを対象としたものであり，そのプログラムは，入力となる局所領域の大きさとデータ型，画像の縁の取扱い方，出力画素のデータ型，出力画素の計算式といった定義からなっている．Apply の対象とする処理は画素単位で完全に独立に計算できるデータパラレルのアルゴリズムであるため，並列化が極めて容易なアルゴリズムである．

一方，Adapt が対象とするのは分割統治アルゴリズムで表現可能なものである．分割統治アルゴリズムは，あるデータの集合に対する演算を，部分集合に対する演算を行った後に統合化するという過程を再帰的に実行することによって計算するアルゴリズムである．こうすることにより，分割された部分集合での演算が並列に実行でき，演算時間の短縮を図ることができる．画像処理としては，画像全体の特徴量計算（濃度ヒストグラム，連結成分のラベル付けなど）が分割統治アルゴリズムで計算するのに適している．

ただし，統合の過程で統合すべき情報の量が多い場合は，統合のための計算時間（プロセッサ間の通信によるオーバヘッドも含む）がかかるため効率の良い並列計算ができない場合もあることに注意する必要がある．また，通信のオーバヘッドの問題から，最適なプロセッサ数が存在する．すなわち，あまり細かく領域を分割しすぎると，統合のための計算時間がかかりすぎることになる．

Adapt を用いて，濃度ヒストグラムを求めるアルゴ

リズムを記述したものを図3に示す．Adaptで記述されたプログラムは四つのセクション（Firstセクション，Nextセクション，Combineセクション，Lastセクション）に分かれる．

- Firstセクション：初期設定を行う．図3の場合はヒストグラム配列の各要素を0にしている．
- Nextセクション：最も細かく分割された領域（最小分割領域）での処理の記述．ヒストグラムを求める場合は，最小分割領域でのヒストグラムを求める．
- Combineセクション：分割領域で求めた特徴量を統合する処理の記述．図3の例では，分割領域で求めたヒストグラム配列の内容を要素単位で加算し，その結果を統合した領域のヒストグラムとする．プログラム中の下線のついた変数(_count)は，分割領域で得られた特徴量を示している．
- Lastセクション：統合の過程がすんだ後の後処理の記述．図3の例では，ヒストグラムを正規化している．

最適実行を行うための分割領域の形状，大きさは並列プロセッサのアーキテクチャに依存するが，Adaptのプログラムテキスト上にはそのような情報はいっさい必要なく，抽象的な計算手順を示すだけでよい．コンパイラが対象アーキテクチャに最適な実行コードを生成する．

```
procedure histogram(
  im:in image byte,
  hist: out array(0..255)of float)
is
  count: array(0..255)of integer;

first begin
  for i in 0..255 loop
    count(i) := 0;
  end loop;
end first;

next begin
  count(im) := count(im) + 1;
end next;

combine begin
  for i in 0..255 loop
    count(i) := count(i) + _count(i);
  end loop;
end combine;

last
  pixels: integer;
begin
  pixels := 0;
  for i in 0..255 loop
    pixels := pixels + count(i);
  end loop;
  for i in 0..255 loop
    pixels := float(count(i)) / pixels;
  end loop;
end last;

end histogram;
```

図 3 Adaptによるプログラム記述

6. その他のツール

ここまでは，主に並列画像理解のアルゴリズム記述という観点からプログラミング言語について述べてきた．実際に，並列プログラムを開発するという側面から考えると，プログラミングの段階の前に，目標とするプログラムをどのように並列化するかという段階があり，実はこのほうが本質的な問題である．人間があるアルゴリズムを並列化するときの一つのアプローチは，与えられた問題と既知の問題の類似性を調べ，既知の問題の解法が利用できる場合はそれを利用することである．この過程を機械的に実現することによって並列アルゴリズムを簡単に開発しようという試みも行われている．

Jamiesonらはこのような観点からCloner，Graph Matcherという並列画像処理プログラム作成ツールの開発を進めている[Jamieson 92]．Clonerはその名のとおり，ある並列アルゴリズムのクローンを作るものであり，すでにある画像処理の並列アルゴリズムの一部を変更して新たな並列アルゴリズムを作成するものである．これは，画像処理の主要な並列アルゴリズムがいくつかの種類に分類され，それぞれの種類でほぼ同じアルゴリズムの構造を持つことに着目したものである．Clonerはすでにある並列アルゴリズムライブラリをもとに，既存のアルゴリズムのデータ構造，制御構造などを再利用することにより新しい並列アルゴリズムを作成するものである．プログラム作成時には，Clonerはユーザにデータ分割・機能分割の指定，画像データのデータ構造，データのアクセスパターン（局所処理のマスクのサイズなど）などを問い合わせて，アルゴリズム作成に必要な情報を対話的に取り込むようになっている．

一方，Graph Matcherは，既存の並列プログラムを直接再利用するというよりは，抽象度の高いレベルで既存の並列アルゴリズムを利用しようとするものである．すなわち，既存の並列アルゴリズムの構造がデータの依存関係で表現されると考え，依存関係の同一性を手掛りにして効率的な（各プロセッサの負荷が均一でプロセッサ間通信の少ない）並列アルゴリズムを作成しようというものである．

Graph Matcherも，Clonerと同様，ライブラリベースのシステムであり，並列画像処理アルゴリズムをデータ依存グラフで表現したものとそれを実際の並列計算機にマッピングしたものをライブラリとして持っている．並列プログラム作成時には，並列化を行おうとする画像処理アルゴリズムをデータ依存グラフで表現したものとライブラリにある既存のデータ依存グラフとマッチングを行い，マッチングしたアルゴリズムと同様の戦略で並列計算機上にアルゴリズムをマッピングする．一般にグラフのマッチング問題は計算量が問題になるため，Graph Matcherでは，ハイパーグラフを利用したマッチング手法を用いている．すなわち，もとのデータ依存グラフの関連するノード集合をハイパーグラフの1ノード（ハイパーノード）で表現し，ハイパーノード間を枝で結ぶことにより，グラフのサイズを小さくし，マッチングの手間を少なくするという工夫を行っている．

また，これまでに述べてきたプログラミング方式だけでなく画像理解の並列プログラムの実行制御方式に関する研究も行われ始めている．例えば，DISCはさまざまな種類のアルゴリズムを利用する複雑なプログラムを効率的に並列計算機上で実行するためのスケジューラであり，ルールベースのエキスパートシステムとして実現されている[Weil 91]．そのほかにも，Rochester大学のAnimate Visionプロジェクトでは，画像理解をベースにロボットの制御を行うといった大規模並列分散システムにおけるプログラミングとその実行制御方式についての研究が行われている[Marsh 92]．

7. むすび

本稿では，並列画像理解のためのソフトウェアアーキテクチャ，主として並列プログラミング言語について紹介した．低レベル（信号レベル）の画像処理に関しては，各種の並列プロセッサが開発されていることもあって，そのソフトウェアについてはかなり研究・開発が進んでいるが，高レベル処理（記号レベルの認識・推論処理）に関しては，「高レベル処理として確立された処理手法が少ない」，「並列プロセッサがまだあまり利用されていない」という点でさほど研究が進んでいないのが実情である．また，動画像処理のような実時間画像処理の記述に関する研究にはほとんど手がつけられていない．実時間処理は実行制御方式とも関連するだけに，単純な問題ではないが，ロボットビジョンや自動走行車のような実時間処理は画像理解の最も重要なアプリケーションの一つであり，今後の研究が待たれるところである．

並列画像理解を実現するための本質的な問題点は，低レベル処理の単純なSIMD型のデータパラレル処理と高レベル処理における機能分割による並列化の両方をいかにうまく取り込むかということである．すなわち，ハードウェア，ソフトウェアの両面において，両者の並列化方式をいかに融合して実現するかという点が極めて重要である．このような処理の枠組みは，単に画像理解に限った特別なものではなく，さまざまな現実的知能処理アプリケーションで必要になってくるものであり，今後，並列処理システムに関する重要かつ中心的な課題になってくるものと考えられる．

また，並列画像理解（処理）プログラムの収集・蓄積も重要な問題である．並列プログラムにはアーキテクチャ依存の部分が多かれ少なかれ存在するため，プログラムの標準化というのは困難な点も多いが，逐次型計算機上の画像処理ライブラリであるSLIP[鳥脇 81]やSPIDER[田村 82]の果たした役割を考えると，並列画像理解研究を進展させるためにも並列プログラムの収集・蓄積を進める必要がある．そのためには，並列アーキテクチャをある程度抽象化して，プログラム記述を行うことも必要であり，MPI[Dongarra 93]のような標準化を目指した並列プログラミングツールを利用することも検討する必要がある．

謝　辞

本稿を執筆するにあたり貴重なコメントをいただいた，岡山大学松山隆司教授に感謝します．また，日頃よりご指導賜る九州大学雨宮真人教授に感謝します．

◇参考文献◇

[Bhalla 90] Bhalla, S. and Weems, C.: A Linda Implementation on the ICAP Level of the Image Understanding Architecture, *Proc. SPIE Parallel Architecture for Image Processing*, pp. 239-249 (1990).

[Carriero 89] Carriero, N. and Gelernter, D.: Linda in Context, *Commun. ACM*, Vol. 32, No. 4, pp. 444-458 (1989).

[Computer 92] Special Issue on Parallel Processing for Computer Vision and Image Processing, *Computer*, Vol. 25, No. 2 (1992).

[Dongarra 93] Dongarra, J., Hempel, R., Hey, A. J. G. and Walker, D. W.: A Proposal for a User-Level, Message Passing Interface in a Distributed Memory Environment, Technical Report TM-12231, Oak Ridge National Laboratory (1993).
[Duff 76] Duff, M. J. B.: CLIP4: a large scale integrated circuit parallel processor, *Proc. 3rd Int. Joint Conf. on Pattern Recognition,* pp. 728-733 (1976).
[Hillis 85] Hillis, W. D.: *The Connection Machine,* MIT Press (1985).
[Jamieson 92] Jamieson, L. H., Delp, E. J., Wang, C., Li, J. and Weil, F. J.: A Software Environment for Parallel Computer Vision, *Computer,* Vol. 25, No. 2, pp. 73-77 (1992).
[Kozato 92] Kozato, Y. and Otto, G. P.: Geometric Transformations in a Lazy Functional Language, *Proc. 11th Int. Conf. on Pattern Recognition,* Vol. IV, pp. 128-132 (1992).
[Leite 92] Leite, N. J. and Bertrand, G.: A Parallel Image Processing Language Based on Computational Models, *Proc. 11th Int. Conf. on Pattern Recognition,* Vol. IV, pp. 181-184 (1992).
[Marsh 92] Marsh, B., Brown, C., LeBlanc, T., Scott, M., Becker, T., Das, P., Karlsson, J. and Quiroz, C.: Operating System Support for Animate Vision, *J. Parallel and Distributed Computing,* Vol. 15, pp. 103-117 (1992).
[松山 93] 松山，浅田，青山：再帰トーラス結合アーキテクチャ上での並列画像解析アルゴリズムの構成，情処学研報，93-CV-84, pp. 55-62 (1993).
[Meyer 91] Meyer, T. and Davidson, J. L.: Image algebra preprocessor for the MasPar parallel computer, *Proc. SPIE Image Algebra and Morphological Image Processing* II, Vol. 1568, pp. 125-136 (1991).
[Morrow 92] Morrow, P. J. and Crookes, D.: Parallel Languages Issues in Transputer-Based Image Processing, *Image Processing and Transputers,* pp. 27-46, IOS Press (1992).
[Pfeiffer 90] Pfeiffer, J. J.: HCL: A Language for Low-Level Image Analysis, *J. Parallel and Distributed Computing,* Vol. 8, pp. 231-244 (1990).
[Poole 92] Poole, I.: A Functional Programming Environment for Image Analysis, *Proc. 11th Int. Conf. on Pattern Recognition,* Vol. IV, pp. 124-127 (1992).
[Ritter 90] Ritter, G. X., Wilson, J. N. and Davidson, J. L.: Image algebra: an overview, *Computer Vision, Graphics and Image Processing,* Vol. 49, pp. 297-331, 1990.
[田村 82] 田村，坂根，富田，横矢，金子，坂上：ポータブル画像処理ソフトウェアパッケージSPIDERの開発，情処学論，Vol. 23, No. 2, pp. 321-328 (1982).
[Taniguchi 91] Taniguchi, R., Yamamoto, N., Tsuruta, N. and Amamiya, M.: Valid-A: Parallel Functional Language for Image Processing—A programming language of AMP (Autonomus Multi-Processor)—*Proc. 7th Scandinavian Conf. on Image Analysis,* pp. 1178-1187 (1991).
[谷口 93] 谷口，日下部，雨宮：超並列処理記述言語Vによる画像処理アルゴリズムの記述，信学技報，PRU93-99, pp. 81-88 (1993).
[鳥脇 81] 鳥脇，福村：画像処理サブルーチンライブラリSLIPについて，情処学論，Vol. 22, No. 4, pp. 353-359 (1981).
[Wallace 88] Wallace, R. S., Webb, J. A. and Wu, I.: Machine-independent image processing: performance of Apply on diverse architectures, *Proc. Image Understanding Workshop,* pp. 602-608 (1988).
[Webb 92] Webb, J. A.: Unifying Concepts in Architecture-Independent Parallel Image Processing in Adapt, *Proc. SPIE Image Processing and Interchange,* pp. 228-239 (1992).
[Weems 90] Weems, C. C., Rana, D., Hanson, A. R., Riseman, E. M., Shu, D. B. and Nash, G.: An Overview of Architecture Research for Image Understanding at the University of Massachusetts, *Proc. 10th Int. Conf. on Pattern Recognition,* Vol. II, pp. 379-384 (1990).
[Weil 91] Weil, F. J., Jamieson, L. H. and Delp, E. J.: Dynamic Intelligent Scheduling and Control of Reconfigurable Parallel Architectures for Computer Vision/Image Processing, *J. Parallel and Distributed Computing,* Vol. 13, pp. 273-285 (1991).
[山元 91] 山元，鶴田，谷口，雨宮：画像処理用超並列プロセッサAMPのプログラミングと性能評価について，情処学論，Vol. 32, No. 7, pp. 933-940 (1991).
[Yamamoto 93] Yamamoto, N., Tsuruta, N., Taniguchi, R. and Amamiya, M.: Massively Parallel Image Processing on AMP (Autonomous Multi-Processor), *Proc. Asian Conf. on Computer Vision '93,* pp. 705-709 (1993).

著 者 紹 介

谷口 倫一郎(正会員)

1980年九州大学工学研究科修士課程情報工学専攻修了．同年，九州大学大学院総合理工学研究科助手．1989年同助教授．工学博士．画像理解，並列処理，画像処理システムに関する研究に従事．電子情報通信学会篠原記念学術奨励賞，情報処理学会論文賞受賞．情報処理学会，電子情報通信学会各会員．現在，情報処理学会コンピュータビジョン研究会幹事．