再帰トーラス結合アーキテクチャにおける並列対象認識のためのデータレベル並列プロセスの構成

松山隆司[†,☆] 浅田尚紀[†,☆☆]
青山正人[†] 浅津英樹[†]

本論文では，我々が開発中の並列画像理解用計算機RTA/1上でデータレベル並列処理を実現する方法について述べ，ボトムアップ解析による対象認識過程を効率的な並列プロセスの組み合わせとして構成できることを示す．RTA/1は先に提案した再帰トーラス結合アーキテクチャに基づいて設計したMIMD型分散メモリ並列計算機で，PE（Processing Element）群によって構成される2次元トーラスの再帰的分割・統合を最大の特徴としている．論文の前半では，データセット（要素データの集合）に対する演算を五つの基本演算パターンに分類整理し，その並列化がデータセットと処理関数の分割および複写，SPMD（Single Program Multiple Data）型およびMPSD（Multiple Programs Single Data）型の並列関数適用および並列パイプライン処理の5種類の機能によって実現できることを示す．すなわち，ここでは五つの基本演算パターンを並列実行することによりデータレベル並列処理が実現でき，並列化された演算パターン（並列プロセス）を組み合わせることにより，複雑な並列処理過程が構成できると考える．こうした考えに基づき論文では，RTA/1上でのデータレベル並列処理の実現法を検討し，エッジ検出，Hough変換，Geometric Hashingを用いた対象認識過程をデータレベル並列プロセスの組み合わせとして設計する．

Designing Data Level Parallel Processes for Object Recognition on Recursive Torus Architecture

TAKASHI MATSUYAMA,[†,☆] NAOKI ASADA,[†,☆☆] MASAHITO AOYAMA[†] and HIDEKI ASAZU[†]

This paper proposes a scheme of data level parallel processing on an MIMD parallel computer RTA/1 and demonstrates its utilities by designing complex parallel processes for bottom-up object recognition. RTA/1 is designed based on the recursive torus architecture (RTA) which we proposed before. Its most distinguishing characteristic is the recursive partition/composition of two dimensional torus networks of processing elements. In the paper, we first propose five types of fundamental operation patterns for data-sets, sets of data elements. Then we show that these operation patterns can be efficiently executed in parallel by incorporating five types of parallelization mechanisms: data/processing-function partitioning and replication, SPMD (single program multiple data) and MPSD (multiple programs single data) processings, and parallel pipeline processing. That is, we consider that data level parallel processing can be realized by parallel execution of five fundamental operation patterns and that complex parallel processes can be fabricated by combining such parallel operations. Based on these ideas, we discuss implementation methods of data level parallel processing on RTA/1, and design parallel object recognition processes including edge detection, Hough transform, and geometric hashing.

† 岡山大学工学部情報工学科
Department of Information Technology, Faculty of Engineering, Okayama University
☆ 現在，京都大学大学院工学研究科電子通信工学専攻
Presently with Department of Electronics and Communication, Kyoto University
☆☆ 現在，広島市立大学情報科学部知能情報システム工学科
Presently with Department of Intelligent Systems, Faculty of Information Sciences, Hiroshima City University

1. はじめに

一般に，並列処理の方式には次の二つの考え方がある[1)]．

データレベル並列処理: 互いに独立処理可能な大量のデータがある場合，それらを分割しProcessing Element（以下PEと略す）に割り当て，並列処理を行う[1)~5)]

コントロールレベル並列処理: プログラムにおける命令実行の制御構造を分析し，独立に実行可能な命令を複数のPEで並列実行させる

従来，単純な一様処理で実現可能な画像処理では，主としてSIMD型並列計算機によるデータレベル並列処理が行われてきた[6)~9)]．これに対して，画像理解では，多様なデータ構造およびアルゴリズムを柔軟に組み合わせ，制御することが必要であり，その効率的な並列化は容易ではない．この問題に関して最近では，SIMD型およびMIMD型並列処理機構を組み合わせた並列計算機を用いて，データレベル，コントロールレベル並列処理の両方式を実現することが検討されている[10)~12)]．しかし，どのシステムも全体として統一性のある処理方式を確立するには至っていないのが現状である．

本論文では，画像理解システムにおける最も基本的な機能であるボトムアップ解析による対象認識過程を対象とし，データレベル並列処理の観点から処理過程全体を統一的に記述表現する方式について検討した．これは以下の理由による．

ボトムアップ解析による対象認識は，

画像処理レベル: 空間フィルタリングによる入力画像からのエッジ画像の作成

画像解析レベル: Hough変換[13)]によるエッジ画像からの直線検出

認識レベル: Geometric Hashing[14),15)]による図形モデルと画像中の直線群との照合

という一連の処理過程によって構成でき，処理の対象となるデータは，画像やパラメータ空間といった一様な幾何学的空間上で定義された図形であるため，幾何学的空間の一様性を生かしたデータレベル並列処理が有効にはたらく．

本論文では，我々が開発を進めている**再帰トーラス結合アーキテクチャ**（Recursive Torus Architecture，以下RTAと略す）[16)]に基づく並列画像理解用計算機RTA/1上で，効率的なデータレベル並列処理を構成する方法について述べ，それに基づいて上記のボトムアップ解析による並列対象認識過程を設計する．

論文の前半では，データレベル並列処理の統一的な記述表現方式を考えるために，まず多数の要素データからなるデータセットを対象とした演算を五つのパターンに分類整理する．次にこれら五つの基本演算パターンを並列実行するには，データセットや処理関数の分割，複写および効率的なパイプライン処理が重要となることを指摘する．

データの分割に基づく並列処理とは，一様なデータ空間を分割し，CPUと局所メモリからなるPEごとに異なる部分データを割り当て，各PEで同一のプログラムを並列に実行するSPMD（Single Program Multiple Data）型の並列処理である．

一方，データの複写に基づく並列処理とは，各PEがそれぞれ同一のコピーデータを持ち，同じデータを異なるプログラムで並列処理するMPSD（Multiple Programs Single Data）型の並列処理である．この並列処理方式は，各PEが同じデータを重複して持つというメモリ使用効率の悪さはあるものの，データの持つ並列性（冗長性）を利用した多彩な並列処理が可能となるため，分散データベースシステムでは早くからその有効性が議論され[17)]，最近では画像処理においてもその有効性が示されている[18)]．

最後にパイプライン処理とは，まずPEごとに異なる部分データを割り当て，次に各PEがデータの持つ順序性に従って通信を行いながら，部分データに対する処理を行う並列処理である．最大値選択やヒストグラム生成のように，データセット内の大局的な情報を求める処理を並列に行うには，パイプライン処理をいかに効率的に行うかが焦点となる．

論文の後半では，データセットに対する基本演算パターンを並列実行することによってデータレベル並列処理が実現されるという考え方に基づき，エッジ検出，Hough変換，Geometric Hashingを用いた具体的な並列対象認識過程を，RTA/1上でのデータレベル並列処理過程として構成し，その有効性を示す．

2. 並列画像理解用計算機RTA/1

RTA/1は，2次元非同期式RTA[16)]に基づいて設計されたMIMD型分散メモリ並列計算機である．RTA/1は，**図1**に示すようにPEを配置した**PEトーラス**，システム全体を制御する**制御プロセッサ**，そしてPEトーラスを構成する通信線上に配置したスイッチ群を動的に切り換える**スイッチ制御機構**[19)]の三つの要素から構成される．

PEトーラスは$\sqrt{N}\times\sqrt{N}(\sqrt{N}=2^{2L}, L$は$L\geq 1$の正整数）個のPEを2次元トーラス状に結合したもので，スイッチ群の動的な切り換えにより，PEトーラスの再帰的な4分割・統合を行うことができる動的結合網となっている（図3，6参照）．

PEトーラスの再帰的な分割・統合を実現するには，PE間の同期をとってスイッチ群を切り換える必要がある．このためスイッチ制御機構は，PE間の接続関係を切り換える機能だけでなく，PE間の同期機構としての役割も果たす．

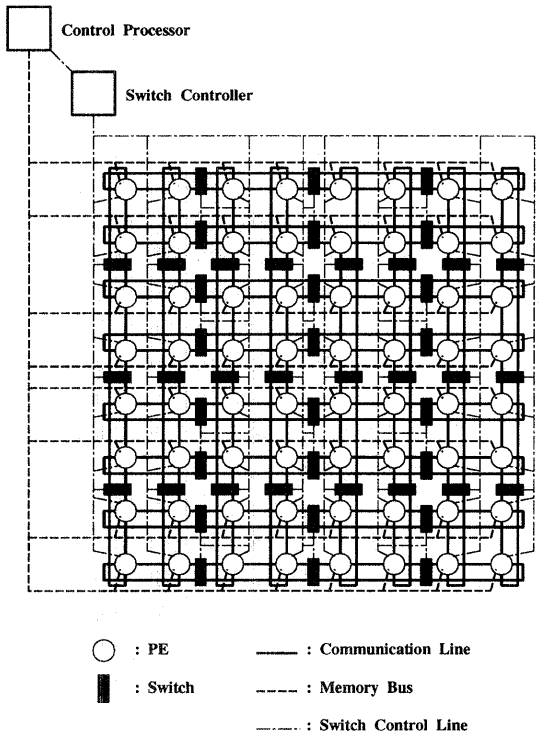


図1 並列画像理解用計算機 RTA/1 の構成
(設計上は $32 \times 32 = 1024$ 台構成，現在試作中の実験機では $4 \times 4 = 16$ 台の PE を持つ)

Fig. 1 Architecture of RTA/1.
($32 \times 32 = 1024$PEs in its specification, $4 \times 4 = 16$PEs in the currently developed prototype machine)

またRTA/1では，制御プロセッサとPE間の双方向高速通信のために共有メモリを配置している．この共有メモリは，各PEの持つ局所メモリの一部を制御プロセッサのメモリバス（図1の破線）と結合することによって実現される．共有メモリは，「制御プロセッサ ⇔PE」という1対1の双方向通信だけでなく，「制御プロセッサ ⇒ 特定の部分トーラスに属するすべてのPE」といったブロードキャスト通信が行えるようにメモリマッピングが行われている．これら共有メモリを介した通信は，制御プロセッサ ⇔PE間の割り込みによって起動される．

3. データレベル並列処理の実現法

3.1 基本演算パターン

データレベル並列処理が有効となるのは，「多数のデータ」を処理する場合である．そこでまず，処理の対象となるデータをデータセットとして次のように定義する．

[データセット]: データセットは1個以上有限個の要素データから構成されるデータの集合である．ここで要素データは数値や文字といった構造を持たない単純データまたはデータセットである．また，要素データ間の関係（例えば全順序関係）のことをデータセットの構造と呼び，種々の構造を用いることにより，配列や木構造のようなデータ構造を表現することができる．ここでは単純化のため，$\mathcal{D} = [d_1, \cdots, d_n]$ のようなリストとしてデータセットを表記する．なお，一つの要素データのみで構成されると考えたデータセットのことを特に単一要素データと呼び，$\mathcal{D} = d$ と表記する．

基本演算パターンはデータセットに関数を適用する際の入出力構造に着目して演算のタイプを分類整理したもので，ここでは以下に示す5種類のものを考えた．

(a) 一括型演算: この演算は，

input: $\mathcal{D}_I = [d_1, \cdots, d_n]$
function: $\mathcal{F} = f$
output: $\mathcal{D}_O = [f(d_1), \cdots, f(d_n)]$

と表記でき，各要素データが関数 f の入力となり，入力データセットと同じ構造を持ったデータセットが出力となる．この演算は一般に同一の構造を持つ複数のデータセットを入力にとることができ，そのときには対応する要素データ同士が多引数の関数 f の入力となる．

(b) 分散型演算: この演算は，

input: $\mathcal{D}_{I_1} = [d_1, \cdots, d_n]$
$\mathcal{D}_{I_2} = d$
function: $\mathcal{F} = f$
output: $\mathcal{D}_O = [f(d_1, d), \cdots, f(d_n, d)]$

と表記でき，データセットと単一要素データが入力となる．データセットの要素データと単一要素データが関数 f の入力となり，入力データセットと同じ構造を持つデータセットが出力となる．

(c) 集約型演算: この演算は，

input: $\mathcal{D}_I = [d_1, \cdots, d_n]$
function: $\mathcal{F} = f$
output: $\mathcal{D}_O = f(f(\cdots f(f(d_1, d_2), d_3), \cdots), d_n)$

と表記でき，データセットの持つ順序構造に従って順次関数 f を適用することによって，入力データセットの各要素データが統合され，単一要素データとして出力される．

(d) スキャン型演算: この演算は，

input: $\mathcal{D}_I = [d_1, \cdots, d_n]$
function: $\mathcal{F} = f$
output: $\mathcal{D}_O = [f(d_1), f(d_1, d_2), \cdots,$
$f(f(\cdots f(f(d_1, d_2), d_3), \cdots), d_n)]$

と表記でき，集約型演算と同様の処理を行うが，統合

過程のそれぞれの中間結果を要素データとした，入力データセットと同じ構造を持つデータセットが出力となる．

(e) **展開型演算**: この演算は，

input: $\mathcal{D}_I = d$

function: $\mathcal{F} = [f_1, \cdots, f_n]$

output: $\mathcal{D}_O = [f_1(d), \cdots, f_n(d)]$

と表記でき，単一要素データに異なる関数を適用し，各関数の出力を要素とするデータセットが出力となる．

3.2 基本演算パターンの並列化

基本演算パターンの並列化においては，ある要素データに関数 f を適用する際，その関数適用が他の要素データに対して独立性および順序性を持っているかどうかが重要となる．

この観点から基本演算パターンを見ると，一括型演算と分散型演算は，データセットの構造に関わりなく各要素データに対して独立に関数が適用でき，関数適用の独立性に基づいて並列化を実現できる．まず一括型演算（**図 2** (a)）では，入力データセットを PE 群に分割して割り当て，関数 f を各 PE に複写することによって SPMD 型の並列処理が実現できる．処理の結果得られた出力データセットは入力データセットと同じ形式で分割され，PE 群に分散配置される．一方，分散型演算（図 2 (b)）では，入力データセットは分割，入力単一要素データおよび関数 f は複写を行うことにより，SPMD 型の並列処理が行える．

集約型演算とスキャン型演算は，データセットの持つ順序構造に基づいた処理を行うことから，関数適用の順序性に基づいて並列化を実現する．まず集約型演算（図 2 (c)）では，入力データセットを PE に分割して割り当て，適切な順序で関数 f の適用と通信を行うパイプライン型の並列処理となり，最終的に得られた単一要素データが出力となる．一方，スキャン型演算（図 2 (d)）は，集約型演算と同様のパイプライン型の並列処理となるが，この演算では統合の各中間結果を要素データとする分散配置されたデータセットが出力される．後述するように，こうしたパイプライン処理が効率よく並列実行できるためには，関数 f が associative であることあるいは，データセットの持つ順序構造が PE 間の並列通信パターンとうまく対応付けられることが必要である．

最後に展開型演算は，関数適用の独立性に基づいて並列化できる．すなわち，図 2 (e) に示すように各 PE に単一要素データを複写し，関数群を分散して割り当てることにより MPSD 型の並列処理ができる．この処理の結果，各関数の出力を要素データとするデータセットが PE 群に分散配置される．

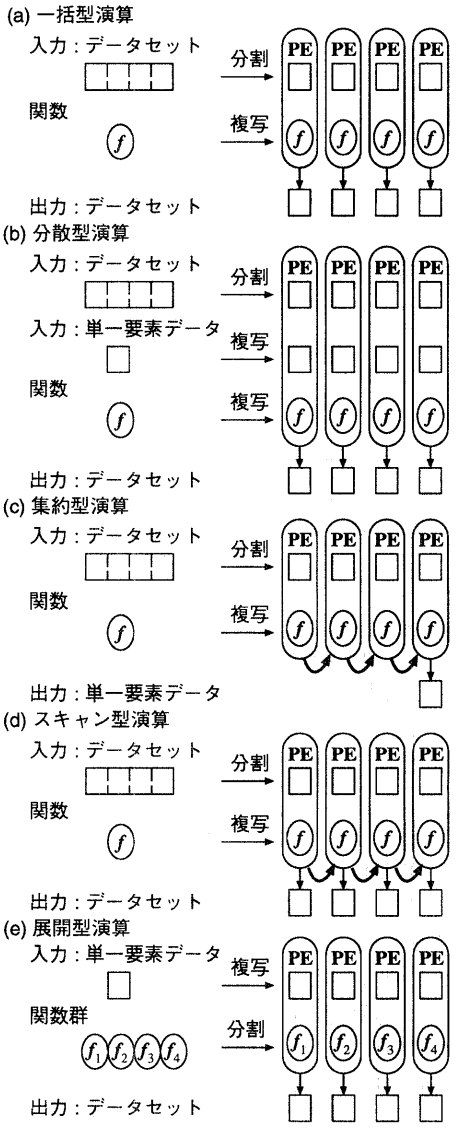


図 2 基本演算パターンの並列化

Fig. 2 Parallelization of fundamental operation patterns.

以上のことから，基本演算パターンの並列化，すなわち本研究で考えるデータレベル並列処理は，「関数適用の並列化」として「SPMD 型」「パイプライン型」「MPSD 型」の 3 種類，データの分散記憶方式を表す「分散データ型」として「分割型」「複写型」の 2 種類によって構成できることが分かる．我々は「分割型」「複写型」に加えて，データセットが任意の PE あるいは制御プロセッサ上に存在している記憶状態を表現する「集中型」を含めた 3 種類を分散データ型として定義し，それらに対する種々のタイプの並列関数適用としてデータレベル並列プロセスを記述する．

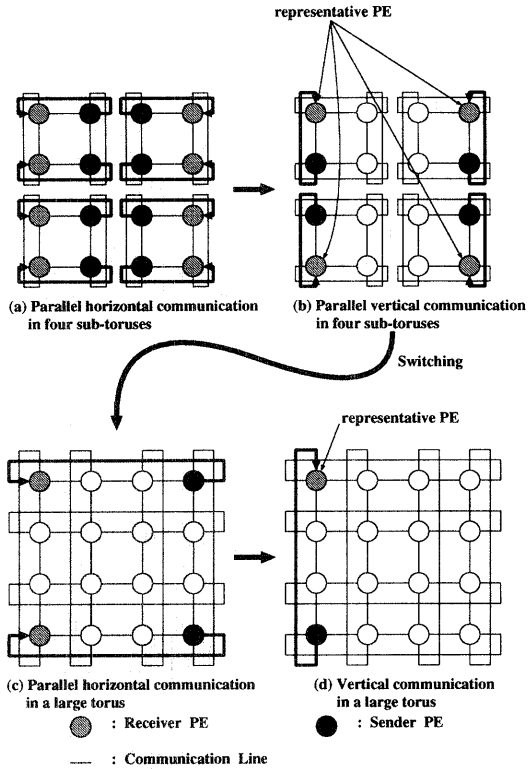


図3 集約型並列データ転送手順

Fig. 3 Scheme of parallel concentration procedure.

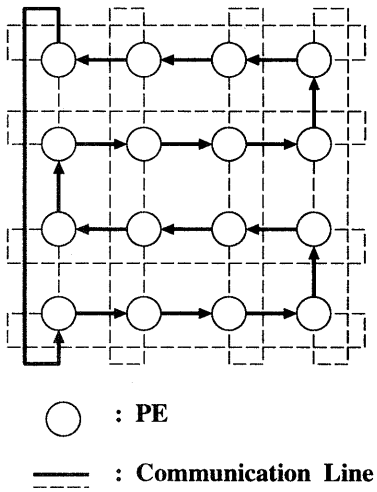


図4 並列シフトデータ転送手順

Fig. 4 Scheme of parallel shift procedure.

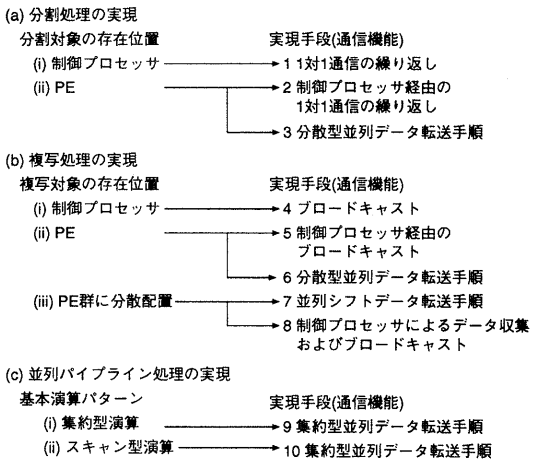


図5 RTA/1での基本処理機能の実現

Fig. 5 Realization of fundamental processing functions on RTA/1.

3.3 RTA/1での実現

ここでは，図2に示したデータレベル並列処理を実現する上で必要となる，データセットや関数といった対象の分割処理，複写処理および並列パイプライン処理の3種類の基本処理機能をRTA/1上で実現する方法について述べる．なお，SPMD型やMPSD型の並列関数適用は，以下に述べる分割，複写処理によって，データセットや関数がPE群に分散配置されれば容易に実現できる．

上記の3種類の基本処理機能はRTA/1の持つ4種類の通信機能を用いて実現する．4種類の通信機能とは，3種類の組織的並列データ転送手順と2章で述べた制御プロセッサ ⇔ PE間の共有メモリを用いた通信である．**図3**は，組織的並列データ転送手順の一つである集約型並列データ転送手順[16),19)]の考え方を示したもので，各PEに分散配置されたデータが階層的並列通信によってトーラスを代表する一つのPEに集約される．この手順を逆順に実行することによって，代表PEが持つデータをトーラス内のPE群に分散させる分散型並列データ転送手順が実現できる．**図4**はPE群に分散配置されたデータを循環シフトする並列シフトデータ転送手順[20)]の考え方を示したものである．

図5は，上記の3種類の基本処理機能とその実現手段であるRTA/1での通信機能との対応関係をまとめたもので以下これに基づいて説明する．

分割処理には分割対象（データセット，関数セット）の存在位置によって二つの場合がある．まず，制御プロセッサ上に分割対象があるときには，共有メモリを用いた制御プロセッサ ⇒ PEの1対1通信を反復することによって分割処理が実現できる．一方，あるPE上に分割対象があるときには2種類の方法が考えられる．一つは，まずPE ⇒ 制御プロセッサ通信によって，分割対象を制御プロセッサに移したのち，上で述べた方法を行うもので，他方は，分散型並列データ転

送手順を用いる方法である．前者は，制御プロセッサ⇔PE 間の割り込み，後者は PE⇔PE 間の通信およびスイッチ切り換えといった異なったハードウェア機能が使われ，いずれの方法が有利かは，実際使用するハードウェアによって変わる．現在試作中の実験機では，前者の方が高速であると考えられる．

複写処理には複写対象の存在位置によって三つの場合がある．まず，複写対象が制御プロセッサあるいは PE に集中して記憶されている場合は，図 5 (b) (i) (ii) に示す方法で複写処理が実現できる．(ii) の場合は一般に制御プロセッサ経由でブロードキャストを行う方が，分散型並列データ転送手順より有利である．次に複写対象が PE 群に分散配置されている場合（図 5 (b) (iii)）には，並列シフトデータ転送手順による実現法と，制御プロセッサによるデータ収集およびブロードキャストによる実現法の二つが考えられる．前者は，各 PE がデータを並列シフトしながら受信データを局所メモリに蓄積することによって実現される．後者は，各 PE が自分の持つ部分データを共有メモリに置き，それらを制御プロセッサが収集して一つのデータセットとした後，ブロードキャストするものである．先の図 5 (a) (ii) と同様，いずれの方法が有利かはハードウェアによって変わるが，実験機では前者の方が高速であると考えられるため，5 章の例では前者の方法を用いている．

最後にパイプライン処理では，関数適用の順序性を考慮する必要があり，その効率的な並列実行には工夫が必要である．我々は，関数適用の順序性を組織的並列データ転送手順における PE 間での通信の順序性と対応付け，パイプライン処理の効率的な並列化を図ることを考えた．すなわち集約型演算を実現するためのパイプライン処理は，集約型並列データ転送手順に関数 f の適用を埋め込んだものとして実現する．具体的には，まず処理対象となるデータセットを前述の分割処理によって PE トーラスに分散配置する．次に，図 3 に示した手順に従って通信を行う過程において，データを受信した PE が受信データと自分が持つ部分データに対して関数 f を適用し，その結果を他の PE に送信する．

分散配置されているデータセットが何ら構造を持たない場合や関数 f が associative な場合は，データセットの分割法は任意で良く，上記の方法で集約型演算を効率的に並列実行できる．一方，データセットが構造を持ち，関数 f が associative でない場合は，通信手順によって規定される関数適用の順序構造とデータセットの構造をうまく対応付けるデータ分割法を考える必要がある．**図 6** に示すように，RTA/1 では，トーラスの再帰的 4 分割を行うことから，PE と 4 分木構造のノードを対応付けることが容易にできる．また，図 6 右のトーラスの再帰的 4 分割と図 3 に示した集約型並列データ転送手順を比較すると，この手順に埋め込まれた関数 f の適用順序が 4 分木構造と対応することが分かる．このことから，データセットの持つ構造が 4 分木構造の場合は，関数 f が associative でなくても集約型演算を効率的に並列実行できる．さらに RTA/1 は，縦あるいは横方向の再帰的 2 分割可能な 1 次元トーラスによって構成されているとも考えられるので，データセットの持つ構造が 2 分木構造の場合でも集約型演算を効率的に並列実行できる．

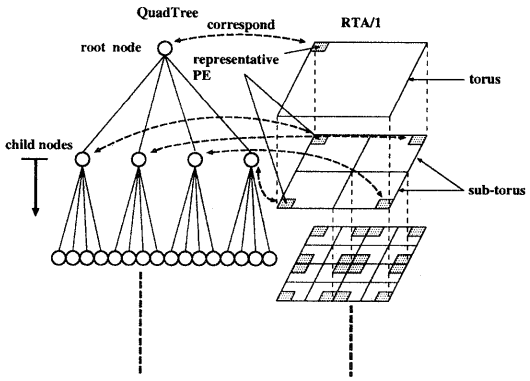


図 6 4 分木構造と RTA/1 の構造の対応付け
Fig. 6 Correspondence between a quadtree and the structure of RTA/1.

スキャン型演算におけるパイプライン処理は，各 PE が関数 f を適用した結果を局所メモリに記憶することによって，集約型演算と同様に効率的並列化ができる．

以上述べた分割，複写，並列パイプライン処理という基本処理機能は，見方を変えると分散データ型の型変換を実現していることになる．**図 7** は図 5 に示した 1~10 の通信機能を型変換として図示したものである．

以上本章で述べた議論をまとめると [**計算モデル**:5 種類の基本演算パターン] $\xrightarrow{\text{並列化のための要素機能の分析}}$ [**並列化法**:5 種類の基本処理機能（分割処理，複写処理，並列パイプライン処理，SPMD 型，MPSD 型並列関数適用）] $\xrightarrow{\text{RTA/1 のハードウェア機能の分析}}$ [**RTA/1 での実現法**:図 5 の 10 種類の通信機能および並列関数実行] となり，RTA/1 におけるデータレベル並列処理の実現法が示せたことになる．

4. データレベル並列処理過程の構成

ここでは，前章で述べた五つの基本演算パターンを

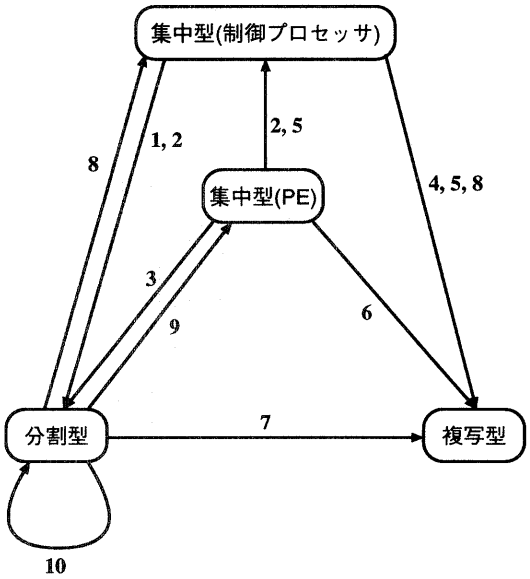


図7 基本処理機能による型変換（図5参照）
Fig. 7 Distributed data type transform by the fundamental processing functions (see Fig. 5).

並列化して得られる並列プロセスを組み合わせることにより，複雑な並列処理過程を構成する問題を考える．

4.1 並列分割・統合演算の実現

並列プロセスを組み合わせることによって様々なマクロ演算が実現できるが，その最も汎用的なものとして並列分割・統合演算がある．

よく知られているように，逐次処理では，再帰計算の枠組みの中で分割・統合という処理を行うことによって効率的なアルゴリズムが構成できる．これに対し，我々が考えている並列分割・統合演算では，先の並列集約型演算を用いて統合演算を実現する．すなわち，まず後述する分割演算で得られた結果を，PE群に分散配置されたデータセットとして表し，それに対して集約型並列データ転送手順に基づいた関数適用を行うことによって，効率的な並列統合演算を実現する．

一般に，このような並列統合演算が可能となるためには，逐次計算における分割・統合演算の持つ再帰構造と，集約型並列データ転送手順の持つ構造の対応が取れなければならない．この対応付けに関しても，前章で述べたデータセットの持つ構造と通信パターンとの対応付けに関する議論がそのまま成り立つ．

分割演算については，(1) データセットの分割，(2) 処理関数の分割，という二つの観点がある．

データセットの分割演算は並列一括型演算（図2(a)）を用いて実現する．この並列分割演算と，先に述べた並列統合演算を図8(a)のように組み合わせることにより，データセットの分割に基づいた効率のよい並列分割・統合演算が構成できる．こうした並列分割・統合演算を用いることによって文献21)で提案されているsplit-and-mergeに基づく並列処理がすべて実現できる．

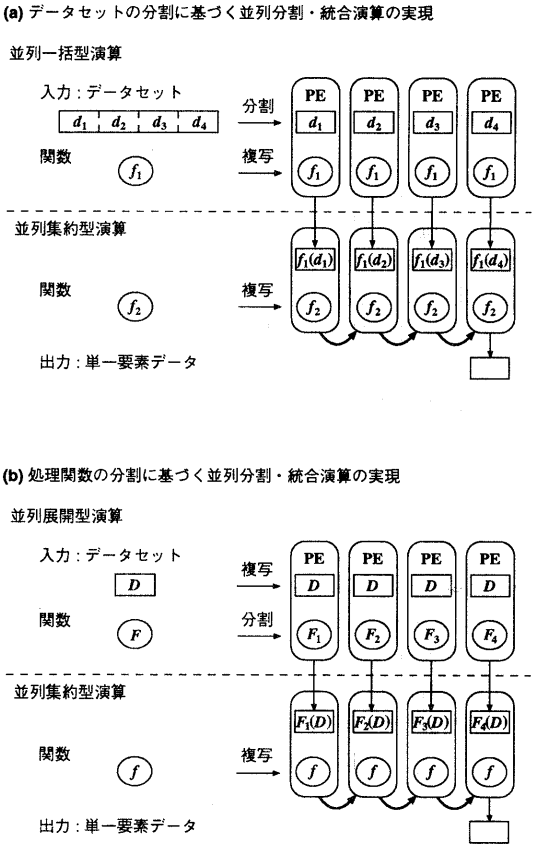


図8 並列分散・統合演算の実現
Fig. 8 Realization of parallel divide-and-combine operations.

一方，多様な処理の効率的並列化のためには，処理関数の分割に基づいた並列分割・統合演算が必要となる．ここでいう処理関数の分割とは，意味のあるまとまった処理が関数 F をデータ D に適用して実現されるとき，F を多数の部分関数 $F_1, \cdots, F_n$ に分割し，各 F_i を D に適用して処理を行うことを意味する．処理関数の分割が意味を持つには，$F(D) = G(F_1(D), \cdots, F_n(D))$ となる結合関数 G が存在するように F を分割する必要がある．先に述べたように，ここで考える並列分割・統合演算では，並列集約型演算によって結合演算すなわち G を実現する．したがって，処理関数の分割に基づく並列分割・統合演算（図8(b)）は，まずMPSD型並列処理による並列展開型演算を行い，その結果得られる分散配置されたデータセット $[F_1(D), \cdots, F_n(D)]$

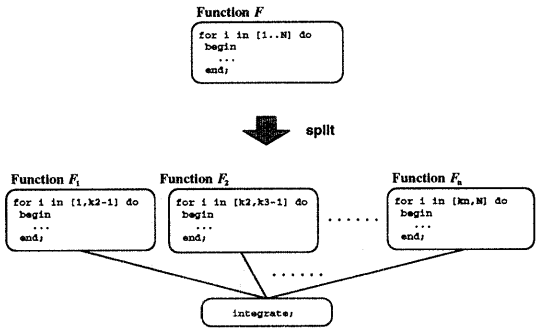


図9 反復計算に着目した処理関数の分割
Fig. 9 Dividing a processing function based on iterative computation.

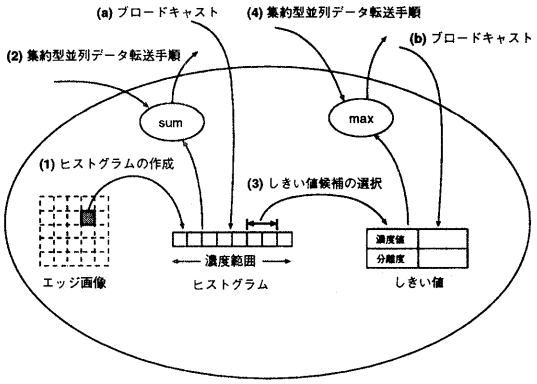


図10 並列しきい値決定
Fig. 10 Parallel threshold determination.

に並列集約型演算を適用することによって実現される．この際，関数 F の分割および統合関数 f は $F(D) = f(f(\cdots f(f(F_1(D), F_2(D)), F_3(D)), \cdots), F_n(D))$ を満たすように設計されなければならない．

図9 は，処理関数の分割の例を示したもので，関数 F のプログラム中に存在する反復計算ループの範囲 $[1..N]$ を分割し，各部分関数 F_i が $[k_i..k_{i+1}-1]$ の範囲の処理を行うというものである．反復計算ループの範囲といった値は，各部分関数の引数として与えることができる．そうした場合，各 F_i のプログラムはすべて同一のものとなり，それらに与える引数の値のみが異なるということになる．関数 F をそうした部分関数群 $F_1, \cdots, F_n$ に分割し，MPSD型並列処理を行うことをPMPSD（Parameterized MPSD）型並列処理といい，5章においてその具体例を示す．

なお関数分割の方式に関する図9以外の有効な例については，現在検討中である．

4.2 並列プロセス間の整合性

並列プロセスを基に並列処理過程を構成していく上で問題となることとして，ある並列プロセスの出力データセットが表現されている分散データ型と，次のプロセスが入力として要求する分散データ型の間に整合性があるかどうかということがある．

図8に示した並列分割・統合演算では，並列一括型演算/展開型演算の出力データセットはいずれも分割型となっており，それを入力として直ちに並列集約型演算を適用することができる．このことは，集約型演算を単独に並列化する場合（図2(c)）に必要であった入力データセットの分割処理が不要となることを意味し，その意味でも並列分割・統合演算が効率的であると言える．

このように二つの並列プロセス間で受け渡しされるデータセットを表現する分散データ型を揃えること，すなわち，処理対象であるデータセットをどのような分散データ型として表現するかが，効率のよい並列処理過程を構成していく上での基本的方針となる．しかし，実際に複雑な並列処理過程を構成していくと，型の整合性が取れない場合もでてくる．そうした場合には図7で示した型変換機能を用いてデータセットの型変換を行う必要があり，5章においてその例を示す．

5. 並列対象認識過程の設計

本章では，これまでの議論を踏まえ，1章で示したボトムアップ解析による対象認識を，データレベル並列処理に基づく並列処理過程として設計する．

5.1 画像処理過程（二値エッジ画像の作成）

この過程では，画像入力 →Sobel フィルタ → 大津のしきい値決定法[22]→ 二値化，を行う．

Sobel フィルタは，図7の型変換1を用いて画像配列をブロック状に分割しPEトーラスに分散配置することで，並列一括型演算として効率的並列化が実現できる．次に分散配置された出力エッジ画像からヒストグラム生成，最適しきい値の計算を行う．**図10** は，この処理の様子を一つのPE内でのデータ配置，関数適用および通信手順として描いたもので，以下この図に基づいて説明する．

まず分割型配列として表されたエッジ画像から，ヒストグラムといった大局的特徴量を計算するには，以下のように並列分割・統合演算を行えばよい．

(1) SPMD型並列処理により部分画像中の局所ヒストグラムを作成する．

(2) 分散配置された局所ヒストグラム群を入力データセットとした並列集約型演算による大局的ヒストグラムの作成．このとき，集約型並列データ転送手順に埋め込まれる関数 f（図10のsum）は，ヒストグラム配列の要素ごとの和を計算するものでassociativeと

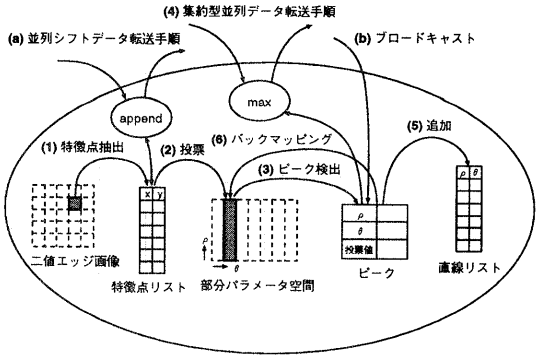


図 11 並列 Hough 変換
Fig. 11 Parallel Hough transform.

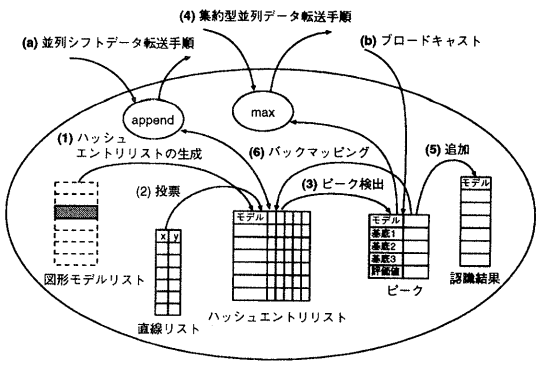


図 12 並列 Geometric Hashing
Fig. 12 Parallel Geometric Hashing.

なっている.

次に，最適しきい値計算は，各濃度値ごとに，しきい値としての妥当性を評価するという反復計算を含んでおり，以下のような処理関数の分割に基づいた並列分割・統合演算として実現する.

(a) 型変換：(2) で得られた大局的ヒストグラムはトーラスの代表 PE に格納されているため，図 7 の型変換 5 によってすべての PE にそのコピーを作る.

(3) 濃度範囲（ヒストグラム配列の長さ）を部分範囲に分け，各部分範囲をパラメータとして PMPSD 型並列処理を行い，各 PE で部分範囲内での最適しきい値を計算する.

(4) 最大値選択関数 max を埋め込み関数とした並列集約型演算により，大局的最適しきい値を求める.

(b) 型変換：代表 PE に格納された大局的最適しきい値を図 7 の型変換 5 によってすべての PE にコピーする.

(b) の型変換によって，各 PE が自分の持つ部分エッジ画像を大局的最適しきい値によって二値化できる．すなわち，二値化を SPMD 型並列処理として実現できる.

5.2 画像解析過程（Hough 変換）

この過程では，特徴点抽出 → パラメータ空間への投票 → ピーク検出 → 直線リストの作成 → バックマッピング[23),24)]，を行う．**図 11** は，この処理の様子を前節と同様の方法で描いたもので，以下この図に基づいて説明する.

(1) SPMD 型並列処理で部分二値エッジ画像中の特徴点を抽出する.

(a) 型変換：各 PE に分散配置された特徴点リスト群を図 7 の型変換 7 で複写型に変換する.

(2) 角度パラメータ（$0 \leq \theta < \pi$）を部分範囲に分け，各部分範囲をパラメータとして PMPSD 型並列投票処理を行う．すなわち，ρ–θ パラメータ空間は図 11 のように短冊状の部分空間に分割され，各 PE は部分空間内に投票を行う.

(3) SPMD 型並列処理で部分パラメータ空間中の局所ピークを検出する.

(4) 最大値選択関数 max を埋め込み関数とした並列集約型演算による大局的ピークの検出.

(b) 型変換：代表 PE に格納されている大局的ピークを，図 7 の型変換 5 を用いてすべての PE にコピーする.

(5)(6) SPMD 型並列処理によって，大局的ピークを直線リストに追加するとともに，部分パラメータ空間に対するバックマッピングを行う.

必要な回数だけ，(3)～(6) を繰り返すことにより，画像中の直線群を表す ρ–θ パラメータが直線リスト（複写型）として得られる.

5.3 認識過程（Geometric Hashing）

対象認識は，文献 15) で提案された線分を基にした Geometric Hashing に基づいて行われ，図形モデル集合の入力 → ハッシュエントリリストの作成 → ハッシュエントリリストへの投票 → モデル候補の選択 → モデル名リストの作成 → バックマッピング，という処理過程によって実現される（**図 12**）.

(0) 制御プロセッサが持つモデル集合を図 7 の型変換 1 により部分モデル集合に分割し，PE 群に分散配置する（図 12 の図形モデルリスト）.

(1) SPMD 型並列処理により，各 PE が部分モデル集合から部分ハッシュエントリリストを作る.

(a) 型変換：分散配置された部分ハッシュエントリリスト群に図 7 の型変換 7 を適用し複写型の全ハッシュエントリリストを作る．これにより，すべての PE は，先に求めた画像中の全直線リストとすべてのモデルに対するハッシュエントリリストを持つことになる.

図12の(2)(3)(4)(b)(5)(6)の一連の並列処理は図11の並列Hough変換の場合と同様であるので省略する.

6. まとめ

本論文では,データセットに対する処理を五つの基本演算パターンとして分類整理し,それらを並列化することによってデータレベル並列処理が実現できると考え,RTA/1上での具体的な並列プロセスの構成法を示した.ここで考えた並列処理方式は,文献1),21),25),26)で提案されたほとんどすべてのデータレベル並列処理方式を内包している.さらに,データの複写に基づく新たな並列処理方式としてPMPSD型並列処理を提案し,ボトムアップ解析に基づく対象認識をRTA/1上での並列処理過程として設計することによってその有効性を明らかにした.

今後は,以下の項目についての検討を行い,RTA/1を用いた並列画像理解システムの構築を目指す.

(1) 本論文で示したデータレベル並列処理方式に関する理論的検討およびそれに基づいた並列プログラミング言語の設計,処理系の試作.後者の言語については,文献27)においてその基本的考え方を示し,現在処理系の試作を進めている.

(2) 現在製作中のRTA/1の実験機を用いて,図5右に示した通信機能の定量的性能評価を行い,本論文で設計した並列対象認識過程の処理効率を評価する.

(3) 新たな分散データ型および通信機能の考案による,より柔軟なデータレベル並列処理の実現.

(4) 本論文では,PE間の同期の問題は,組織的並列データ転送手順内部の問題として特に議論しなかった.また,図8のような並列プロセスの組み合わせにおいては,図中の破線部で同期が取られるものと考えた.しかし,並列画像理解システムを実現するには複雑な処理過程の制御が必要となり,同期の問題およびコントロールレベルの並列処理が重要となる.今後は,今回提案したPMPSD型の並列処理をコントロールレベル並列処理実現のための出発点と考え,検討を進める.

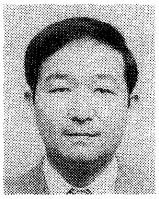
本研究は,文部省科学研究費一般研究(B)(05452-359)の補助を受けて行った.

参考文献

1) Introduction to Data Level Parallelism, Thinking Machines Technical Report 86.14 (1986).
2) Hillis, W.D. and Steele, G.L. Jr.: Data Parallel Algorithms, *Comm. ACM*, Vol.29, No.12, pp.1170–1183 (1986).
3) Hillis, W.D.: *The Connection Machine*, MIT Press (1989).
4) 松田秀雄,金田悠起夫:データ並列論理型言語とその応用,情報処理学会研究会資料,93-PRG-13-14, pp.105–112 (1993).
5) 郷田 修:High Performance Fortran,情報処理,Vol.34, No.9, pp.1179–1186 (1993).
6) Duff, M.J.B.: CLIP 4: A Large Scale Integrated Circuit Array Parallel Processor, *Proc. IEEE Int. Joint Conf. Pattern Recognition*, pp.728–733 (1976).
7) Cypher, R.E. et al.: Algorithms for Image Component Labeling on SIMD Mesh-Connected Computers, *IEEE Trans. Comput.*, Vol.39, No.2, pp.276–281 (1990).
8) Maresca, M.: Polymorphic Processor Arrays, *IEEE Trans. on Parallel and Distributed Systems*, Vol.4, No.5, pp.490–506 (1993).
9) Tucker, L.: *Data Parallelism: Image Understanding and the Connection Machine System, Parallel Architecture and Algorithms for Image Understanding*, pp.473–497, Academic Press (1991).
10) Weems, C.C. Jr.: The Second Generation Image Understanding Architecture and Beyond, *Int. Conf. on Computer Architectures for Machine Perception*, pp.276–285 (1993).
11) Choudhary, A.N. et al.: NETRA: A Hierarchical and Partitionable Architecture for Computer Vision Systems, *IEEE Trans. on Parallel and Distributed Systems*, Vol.4, No.10, pp.1092–1103 (1993).
12) Cantoni, V. and Ferretti, N.: *Pyramidal Architectures for Computer Vision*, Plenum Press (1994).
13) 松山隆司,輿水大和:Hough変換とパターンマッチング,情報処理,Vol.30, No.9, pp.1035–1046 (1989).
14) Lamdan, Y. and Wolfson, H.J.: Geometric Hashing: A General and Efficient Model-Based Recognition Scheme, *Proc. of 2nd ICCV*, pp.238–249 (1988).
15) Tsai, F.C.D.: Robust Affine Invariant Matching with Application to Line Features, *Proc. of CVPR*, pp.393–399 (1993).
16) 松山隆司,青山正人:再帰トーラス結合アーキテクチャ,情報処理学会論文誌,Vol.33, No.2, pp.212–222 (1992).
17) Stamos, J.W. and Young, H.C.: A Symmetric Fragment and Replicate Algorithm for Distributed Joins, *IEEE Trans. Parallel and Distributed Systems*, Vol.4, No.12, pp.1345–1354 (1993).

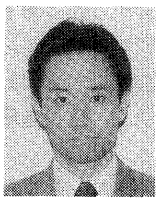
18) Narayanan, P.J. and Davis, L.S.: Replicated Data Algorithms in Image Processing, *Comput. Vision Gr. Image Process.: Image Understanding*, Vol.56, No.3, pp.351–365 (1992).
19) 山下敦也，青山正人，浅田尚紀，松山隆司：再帰トーラス結合アーキテクチャにおけるスイッチ制御機構，信学技報，CPSY93-17, pp.33–40 (1993).
20) 青山正人，浅田尚紀，松山隆司：再帰トーラス結合アーキテクチャを用いた並列画像解析アルゴリズム (I)，信学技報，PRU92-71, pp.103–110 (1992).
21) Webb, J.A.: Steps toward Architecture-Independent Image Processing, *IEEE Comput.*, Vol.25, No.2, pp.21–31 (1992).
22) Otsu, N.: A Threshold Selection Method from Gray-Level Histograms, *IEEE Trans. Systems, Man, Cybernetics*, Vol.SMC-9, No.1, pp.62–66 (1979).
23) Gerig, G. and Klein, F.: Fast Contour Identification through Efficient Hough Transform and Simplified Interpretation Strategy, *Proc. of 8th ICPR*, pp.498–500 (1986).
24) Gerig, G.: Linking Image-space and Accumulator-space: A New Approach for Object-recognition, *Proc. of 1st ICCV*, pp.112–117 (1987).
25) Steele, G.L. Jr., 井田昌之訳：設計者が語る CM-5 の本当の遊び方 (1)，*bit*, Vol.26, No.7, pp.15–22 (1994).
26) Steele, G.L. Jr., 井田昌之訳：設計者が語る CM-5 の本当の遊び方 (2)，*bit*, Vol.26, No.8, pp.40–44 (1994).
27) 松山隆司，浅田尚紀，青山正人：再帰トーラス結合アーキテクチャを用いた並列画像解析アルゴリズムの構成，情報処理学会研究会資料，93-CV-84-8, pp.55–62 (1993).

(平成 6 年 10 月 31 日受付)
(平成 7 年 6 月 12 日採録)



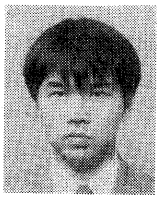
松山 隆司（正会員）

1976 年京都大学大学院工学研究科修士課程修了．京都大学工学部助手，東北大学工学部助教授，岡山大学工学部教授を経て 1995 年 4 月より京都大学大学院工学研究科電子通信工学専攻教授．1982～1984 年米国メリーランド大学客員研究員．工学博士．情報処理学会コンピュータビジョン研究会主査．画像理解，人工知能，並列処理に興味を持っている．1980 年情報処理学会創立 20 周年記念論文賞，1990 年人工知能学会論文賞，1993 年情報処理学会論文賞受賞，1994 年電子情報通信学会論文賞受賞．1995 年第 5 回 ICCV Marr Prize 受賞．著書：「A Structural Analysis of Complex Aerial Photographs」(Plenum)，「SIGMA: A Knowledge-Based Aerial Image Understanding System」(Plenum)，「パターン理解」(オーム社) など．



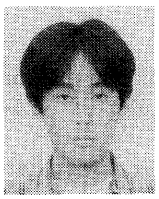
浅田 尚紀（正会員）

1979 年京都大学工学部電気工学科卒業．同大学院博士課程を経て 1984 年福井医科大学助手，1987 年京都大学工学部助手，1990 年岡山大学工学部助教授，1995 年広島市立大学情報科学部教授，現在に至る．1989 年 7 月から 1 年間シカゴ大学カートロスマン研究所客員研究員．工学博士．コンピュータビジョン，画像理解，並列処理，医療診断支援の研究に従事．1989 年医用画像工学会論文賞，1993 年情報処理学会論文賞．IEEE Computer Society，電子情報通信学会，人工知能学会，医用画像工学会各会員．



青山 正人（学生会員）

1991 年岡山大学工学部情報工学科卒業．1993 年同大学大学院工学研究科（修士課程）修了．現在，同大学大学院自然科学研究科（博士課程）在学中．電子情報通信学会学生会員．



浅津 英樹

1993 年岡山大学工学部情報工学科卒業．1995 年同大学大学院工学研究科（修士課程）修了．現在，同大学大学院自然科学研究科（博士課程）在学中．