計算機アーキテクチャ 113−12
（1995. 8. 24）

再帰トーラス結合アーキテクチャに基づく並列画像理解用計算機RTA/1の設計・試作と性能評価

青山 正人 山下 敦也 浅津 英樹 山本 秀彦
小川 敬介 浅田 尚紀* 松山 隆司**

岡山大学工学部情報工学科
* 広島市立大学情報科学部知能情報システム工学科
** 京都大学大学院工学研究科電子通信工学専攻

本論文ではまず、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開発を進めている並列画像理解用計算機 RTA/1(PE:1024 台) のハードウェア構成を述べ、性能評価のために試作した実験機 RTA/0(PE:16 台) の基本性能を示す。次に RTA/1 上でのデータレベル並列処理に基づいた並列化の枠組を示し、その枠組に基づいて、画像理解システムの最も基本的な機能であるボトムアップ解析に基づく対象認識過程を RTA/0 上で実際にインプリメントし、それを基に RTA/1 における性能を評価した結果について報告する。

Design, Development, and Performance Evaluation of Parallel Image Understanding Machine RTA/1 based on Recursive Torus Architecture

Masahito AOYAMA, Atsuya YAMASHITA, Hideki ASAZU, Hidehiko YAMAMOTO, Keisuke OGAWA, Naoki ASADA*, and Takashi MATSUYAMA**

Department of Information Technology, Faculty of Engineering,
Okayama University
* Department of Intelligent Systems, Faculty of Information Sciences, Hiroshima City University
** Department of Electronics and Communication, Kyoto University

This paper first gives a brief overview of the hardware design of RTA/1 (with 1024 PEs), a parallel machine designed based on the Recursive Torus Architecture. We developed a small-scale prototype machine with 16 PEs, RTA/0, to evaluate its performance. Then, we propose a scheme of data level parallel processing on RTA/1 and demonstrate its utilities by showing performance evaluation on RTA/1 based on implementing complex parallel processes for bottom-up object recognition on RTA/0.

1 はじめに

画像理解、特に実時間処理を目指したシステムの高速化を実現するためには、並列処理の導入が必要不可欠となっている。われわれは、先に提案した再帰トーラス結合アーキテクチャ(Recursive Torus Architecture, RTA)[1] に基づいて、並列画像理解用計算機 RTA/1 を設計・開発している [2]。

画像理解における最も基本的な機能であるボトムアップ解析に基づく対象認識は、

画像処理レベル：データ構造: 配列; アルゴリズム: 空間フィルタリングおよびしきい値選択による二値化

画像解析レベル：データ構造: パラメータ空間; アルゴリズム: Hough 変換 [3] による直線検出

認識レベル：データ構造: ハッシュエントリリスト; アルゴリズム: 直線ベースの Geometric Hashing[4][5] によるモデルと直線群とのマッチング

という一連の処理過程によって構成できる。ここで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、その一様性を生かしたデータレベル並列処理 [6]~[8] が有効にはたらく。

本論文では、RTA/1 上でのデータレベル並列処理に基づいた並列化の枠組を示すとともに、実験機 RTA/0 上で、実際にボトムアップ解析に基づく対象認識過程をインプリメントする。その結果から、本論文で示した枠組に基づく RTA/1 上でのデータレベル並列処理の有効性について検証する。

以下 2 章では、まず RTA/1 のハードウェア構成の概要を述べ、性能評価のために試作した RTA/0 の構成および基本性能を示す。3 章では、RTA/1 上でのデータレベル並列処理に基づいた並列化の枠組を示し、4 章では 3 章で示した枠組に沿って、RTA/1 上でデータレベル並列処理を実現するために必要な機能および、その性能を明らかにする。5 章ではそれまでの議論に基づいて、上記のボトムアップ解析に基づく対象認識過程を RTA/0 上で実際にインプリメントし、その実験結果を基に RTA/1 の性能評価を行った結果について述べる。

2 RTA/1

2.1 RTA/1 のハードウェア構成

n 次元 RTA とは、局所メモリを持った PE(Processing Element) を n 次元トーラス状に結合したものである。n 次元 RTA の最大の特徴は、PE 間を接続する通信線上に配置されたスイッチを動的に切り換えることによって、n 次元トーラス結合の再帰的分割/統合を行うことができるという点である。図 1は 2 次元非同期式 RTA(2D-ARTA) で可能なトーラス結合状態を示したものである。

RTA/1 は、2D-ARTA 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 (図 2)。図 2は 8×8PE 構成を示した図であるが、設計では 32×32PE 構成を採っている。RTA/1 は以下の 3 つの構成要素から成る。

制御プロセッサ (CP)：システム全体を制御・管理するプロセッサ。

スイッチ制御機構 (SC)：通信線上に配置されたスイッチの切り換えを行うとともに、PE 間の同期機構としての役割を果たす。スイッチ切り換えアルゴリズムの詳細については文献 [9] を参照して頂きたい。

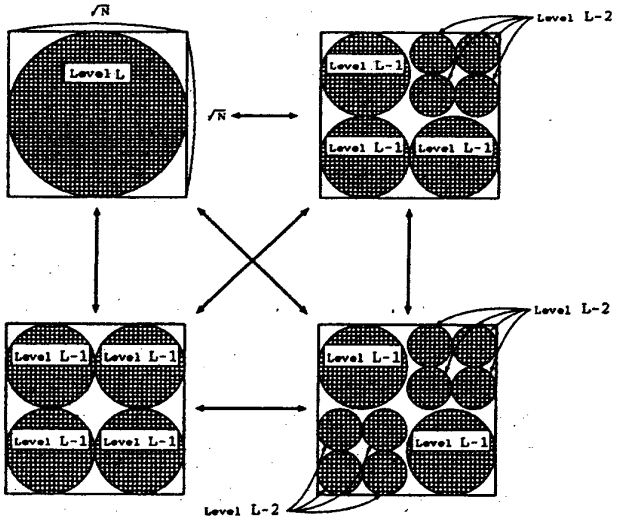

図 1: 2D-ARTA で可能なトーラス結合状態

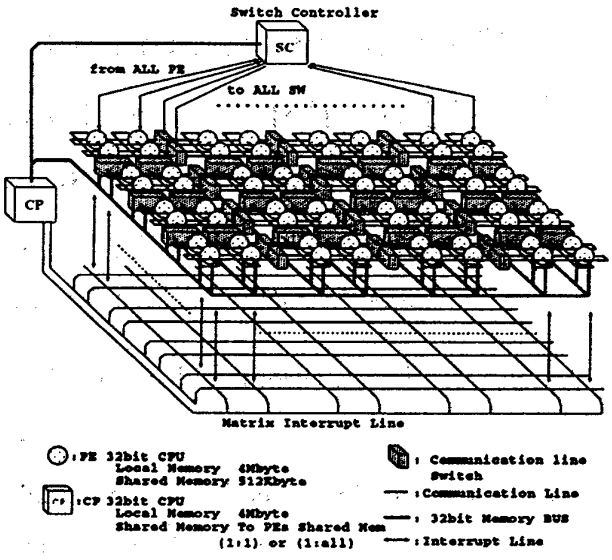

図 2: RTA/1 の概要

表 1: RTA/0 の基本性能

Function	From⟷To	Speed
Effective Memory Access Speed	CP⟷Local/Shared Memory PE⟷Local/Shared Memory	0.203μsec/4Byte 0.203μsec/4Byte
Effective Communication Speed	PE⟷PE (via serial communication line) CP⟷PE (via shared memory)	1.15μsec/Byte (=6.93Mbit/sec) 0.203μsec/Byte + 15.7μsec

PE トーラス：2D-ARTA に基づいた 2 次元トーラス網である。各 PE が持つ局所メモリの一部は CP と共有されており、後述する様々な CP↔PE 間通信に利用される。

2.2 RTA/0 の構成

われわれは RTA/1 の性能を評価するために、4×4PE で構成される実験機 RTA/0 を試作した。RTA/0 は、試作の容易性からトランスピュータ (T805、25MHz、4MB の局所メモリ) を用いて CP および PE を構成した。また安定な PE 間通信を実現するために、トランスピュータが持つシリアル信号線の通信速度を 10Mbit/sec に落としている。RTA/0 の基本的な性能を示したものが表 1である。RTA/1、RTA/0 の詳細なハードウェア設計は文献 [2] で示されている。

3 データレベル並列処理

3.1 対象認識の並列化

1 章で述べたように、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基づく対象認識は、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るデータ構造を扱うことから、データレベル並列処理として、容易に、かつ効率的に実現できる。

ボトムアップ解析に基づく対象認識の一連の処理過程の中で、例えば空間フィルタリング等、配列の各要素に対して独立に処理を適用できるような場合には、それらを PE に分割して割り当てることで高い並列効果を得ることができる。しかし、一連の処理過程を実現するためには、濃度ヒストグラムの計算やしきい値選択など、大局的な情報を得る処理も必要となり、データの表現法、並列処理の手法に工夫を要する。

われわれは、データセットに対する処理を 5 つの「基本演算パターン」として分類整理する。さらにこれらの基本演算パターンが 5 つの「基本処理機能」によって並列実行可能であることを示す。5 つの基本処理機能とは、データ/関数の分割・複写処理、SPMD(Single Program Multiple Data) 型並列関数適用、MPSD(Multiple Program Single Data) 型並列関数適用、および並列パイプライン処理である。次章では RTA/1 上でこれらの基本処理機能を実現する「基本通信機能」について述べ、RTA/0 での実験結果に基づいて RTA/1 上での基本通信機能の性能を評価した結果を示す。われわれの考えを要約すると、データレベル並列処理 $\xrightarrow{\text{モデル化}}$ 5 つの基本演算パターン $\xrightarrow{\text{並列化のための要素機能の分析}}$ 5 つの基本処理機能 (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) $\xrightarrow{\text{RTA/1 での実現}}$ 基本通信機能となる。

3.2 基本演算パターン

データレベル並列処理が有効にはたらくのは、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。

[定義] 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ぶ。データセットの構造としては様々なものを考えることができ、配列や木構造等の多様なデータ構造を表現することが可能である。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、1 つの要素データのみから構成される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
入力: $\mathcal{D}_I = [d_1, \cdots, d_n]$
関数: $\mathcal{F} = f$
出力: $\mathcal{D}_O = [f(d_1), \cdots, f(d_n)]$

ここで入出力のデータセットの構造は同じである。

(b) 分散型演算
入力: $\mathcal{D}_{I_1} = [d_1, \cdots, d_n]$
$\mathcal{D}_{I_2} = d$
関数: $\mathcal{F} = f$
出力: $\mathcal{D}_O = [f(d_1, d), \cdots, f(d_n, d)]$

(c) 集約型演算
入力: $\mathcal{D}_I = [d_1, \cdots, d_n]$
関数: $\mathcal{F} = f$
出力: $\mathcal{D}_O = f(f(\cdots f(f(d_1, d_2), d_3), \cdots), d_n)$

ここで関数 f の適用順序は入力データセットの構造によって決まる。この演算において $\mathcal{D}_I$ の要素データ $d_1, \cdots, d_n$ をデータセットと考えると、出力 $\mathcal{D}_O$ は $\mathcal{D}_I$ の要素データのデータセットとなる。具体例については 5 章で示す。

(d) スキャン型演算
入力: $\mathcal{D}_I = [d_1, \cdots, d_n]$
関数: $\mathcal{F} = f$
出力: $\mathcal{D}_O = [f(d_1), f(d_1, d_2), \cdots, f(f(\cdots f(f(d_1, d_2), d_3), \cdots), d_n)]$

(e) 展開型演算
入力: $\mathcal{D}_I = d$
関数群: $\mathcal{F} = [f_1, \cdots, f_n]$
出力: $\mathcal{D}_O = [f_1(d), \cdots, f_n(d)]$

出力のデータセットの構造は関数の構造によって決まる。

3.3 基本演算パターンの並列化

各基本演算パターンがいかに並列化されるかを示したものが図 3である。

(a) 並列一括型演算
データセット: 分割
関数 (プログラムコード): 複写
処理: SPMD 型並列関数適用

入出力のデータセットは同じ形式で分割される。

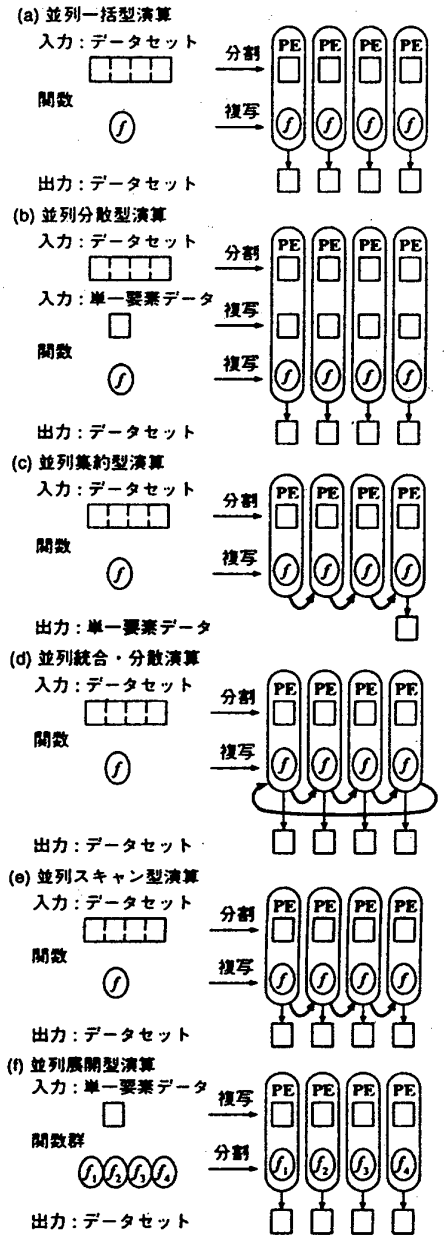

図 3: 基本演算パターンの並列化

(b) 並列分散型演算
データセット: 分割
単一要素データ: 複写
関数 (プログラムコード): 複写
処理: SPMD 型並列関数適用

(c) 並列集約型演算
データセット: 分割
関数 (プログラムコード): 複写
処理: 並列パイプライン処理

パイプラインの構造は入力データセットの構造によって決まる。

(d) 並列統合・分散演算
データセット: 分割
関数 (プログラムコード): 複写
処理: 並列パイプライン処理

この演算は、入力データセットの要素データがデータセットである場合の集約型演算を実現したもので、演算の結果得られる出力データセットは各 PE に分散して配置される。

(e) 並列スキャン型演算
データセット: 分割
関数 (プログラムコード): 複写
処理: 並列パイプライン処理

(f) 並列展開型演算
単一要素データ: 複写
関数 (プログラムコード): 分割
処理: MPSD 型並列関数適用

上記から 5 つの基本処理機能で基本演算パターンの並列化が実現できることが分かる。

データセットの分散表現の観点から見ると、データレベル並列処理の実現のためには、以下の 3 つの分散データ型が必要となる。

集中型: 全データセットをある 1 つの PE の局所メモリに配置する。

分割型: データセットをいくつかの部分データセットに分割し、各部分データセットをそれぞれ異なる PE の局所メモリに配置する。

複写型: 各 PE に全データセットのコピーを配置する。

4 RTA/1 での基本通信機能の実現

本章では、前章で述べた基本処理機能を RTA/1 の基本通信機能によって実現する方法について述べ、またその性能を RTA/0 上での実験結果に基づいて予測する。

4.1 基本通信機能

5 つの基本処理機能のうち、SPMD 型並列関数適用と MPSD 型並列関数適用は、一旦データや関数を分割/複写して与えることができれば容易に実現できる。そこで本節では RTA/1 上でのデータや関数の分割/複写処理および、並列パイプライン処理の実現方法について述べる。

RTA/1 では「組織的並列データ転送手順」と「共有メモリを介した通信」の 2 つの通信手段が用意されている。

1. 組織的並列データ転送手順 [10]

   組織的並列データ転送手順は部分トーラス単位で組織だった通信パターンにしたがって通信を行うため、通信線上でデータが衝突することがない上、CP と独立に機能する。

   (a) **集約型並列データ転送手順:** ある部分トーラス内に分散されているデータをその部分トーラスの代表 PE に集約する。

   (b) **分散型並列データ転送手順:** ある部分トーラスの代表 PE に存在するデータをその部分トーラス内の他のすべての PE に分散する。

   (c) **並列シフトデータ転送手順:** 2 次元トーラス結合をリング構造とみなして各 PE が自分の持つデータを循環シフトする。

2. 共有メモリを介した通信

   RTA/1 では、共有メモリを介することによって、CP↔PE の 1 対 1 通信や CP→任意の部分トーラスへのブロードキャストといった様々な通信が実現できる。

これらの通信手段を用いて以下の13種類の基本通信機能を実現する。

データセットの分割　データセットの分割は分割対象となるデータセットの存在位置によって次の2つの場合に分けられる。

場合1：分割対象となるデータセットがCPにおける集中型である場合。

PD1：CPがデータセットの分割を行い、CP→PEの共有メモリを介した1対1通信を繰り返す。

PD2：CPがデータセットの分割を行い、各PEの共有メモリに部分データセットを書き込む。その後、各PEが並列に部分データセットを読み出す。

場合2：分割対象となるデータセットがPEにおける集中型である場合。

PD3：共有メモリを介したPE→CP通信によってデータセットをCPに転送した後PD1を行う。

PD4：共有メモリを介したPE→CP通信によってデータセットをCPに転送した後PD2を行う。

PD5：分散型並列データ転送手順によって、データセットを分割する。このとき、データセットの分割は通信手順の埋め込み演算[10]として実現する。

データセットの複写　データセットの複写は複写対象のデータセットがどのような分散データ型として表現されているかによって次の2つの場合に分けられる。

場合1：複写対象のデータセットが集中型のとき。これはデータセットの存在位置によってさらに2つに分けられる。

場合1-1：CPにおける集中型である場合。

RP1：共有メモリを介したブロードキャスト通信によって目的の部分トーラスにデータセットを複写する。

場合1-2：PEにおける集中型である場合。

RP2：共有メモリを介したPE→CP通信によってデータセットをCPに転送した後RP1を行う。

RP3：分散型並列データ転送手順によって、データセットを複写する。

場合2：複写対象のデータセットが分割型のとき。このときは以下の3つの方法がある。

RP4：すべてのPEが部分データセットを並列に共有メモリに書き、それをCPが読み出して統合した結果を、目的の部分トーラスにブロードキャストする。

RP5：並列シフトデータ転送手順I(2次元トーラスをリングとみなして循環シフトする)を行う[10]。

RP6：並列シフトデータ転送手順II(2次元トーラスを水平・垂直方向の1次元トーラスの組み合わせと考え循環シフトする)を行う[10]。

表2: RTA/1における基本通信機能の性能評価(100Mbit/sec)

Mechanism	From—To	Function	Time(μsec)
Data partition and Distribution	CP—PE Torus	PD1	$0.203D + 22038$
	CP—PE Torus	PD2	$0.102D + 5880$
	PE—PE Torus	PD3	$0.407D + 22059$
	PE—PE Torus	PD4	$0.305D + 5901$
	PE—PE Torus	PD5	$0.110D + 132$
Data Replication	CP—PE Torus	RP1	$0.203D + 45$
	PE—PE Torus	RP2	$0.407D + 68$
	PE—PE Torus	RP3	$1.1D + 132$
	PE Torus —PE Torus	RP4	$0.305D + 5925$
	PE Torus —PE Torus	RP5	$0.301D + 62452$
	PE Torus —PE Torus	RP6	$0.301D + 3928$
Data Collection and Concentration	PE Torus —PE	CC1	$0.110D + 132$
	PE Torus —PE	CC2	$0.305D + 5887$

並列パイプライン処理　一般に、並列集約型演算、並列スキャン型演算における関数適用の順序性は、処理対象となるデータセットの構造に依存する。文献[10]で示したように四分木や二分木構造はRTA上に容易に埋め込むことができる。したがって、関数適用の順序が木構造で表現できるときには、集約型並列データ転送手順で並列実行可能である。さらに適用関数fがassociativeな場合には、データ構造に関わらずこの手順が利用でき、この手順によって最大値・最小値選択、総和、ソーティングなどの様々なアルゴリズムが構成できる。この場合、二分木構造に沿った通信を行うことにより1次元構造を持ったデータに対するscan operationの効率的な実現が可能であることが知られており[11]、集約型並列データ転送手順により、スキャン型演算の並列化を行うことができる。並列統合・分散演算については並列シフトデータ転送手順IIによって実現できる。

この関数fをappendやmergeとすると、集約型演算は分散データ型を分割型から集中型へ変換する機能を持つことになり、その方法として以下の2種類が考えられる。

CC1：集約型並列データ転送手順を用いる。

CC2：各PEが共有メモリ上に並列に部分データセットを書き込み、それをCPが逐次的にmergeした後、代表PEに共有メモリを介した通信で転送する。

CC1は埋め込み演算によって、並列効果が得られるが、CC2はCPが逐次的にmerge処理を行っているため並列処理になっていないことに注意する。

4.2 RTA/1の性能評価

実験機RTA/0での実測値を基に、RTA/1における基本通信機能の性能予測を行ったのが表2である。ここでシリアル信号線の通信速度は現在市販されているトランスピュータで実現されている100Mbit/secと想定している。表2中のDは通信対象の全データ量をバイト単位で示したものである。表2中の+の後の数字は命令実行時間などのソフトウェアによるオーバーヘッドである。表2から以下のことが分かる。

1. CPに分割対象のデータセットが存在するときには、PD2を用いるのが良い。また、PEに分割対象のデータセットが存在するときには、常に組織的並列データ転送手順であるPD5を用いるのが良い。
2. PEに複写対象のデータセットが存在するときには、常にRP2を用いるのが良い。
3. 分割型から複写型へ変換する機構としては、常に組織的並列データ転送手順であるRP6を用いるのが良い。
4. 分割型から集中型へ変換する機構としては、常に組織的並列データ転送手順であるCC1を用いるのが良い。

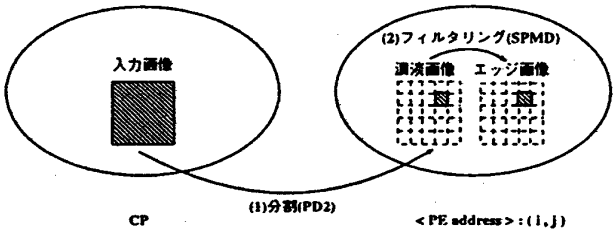

図 4: 空間フィルタリングのデータと処理の流れ

5 並列対象認識

本章では、これまでの議論に基づいて実際に並列対象認識過程を RTA/0 上にインプリメントし、その結果に基づいて RTA/1 の性能評価を行う。

5.1 並列対象認識過程の実現

以下、本論文で行った並列対象認識過程の概要を述べる。

1. 画像処理
   (a) 空間フィルタリング (図 4): 初期状態では入力画像は CP が保持している。その画像を分割し、PD2 で PE に分散し、SPMD 型並列関数適用でフィルタリングを行う。
   (b) しきい値選択 (図 5): まず各 PE は部分エッジ画像からその部分エッジ画像の局所ヒストグラムを作る。その局所ヒストグラムを並列パイプラインで統合・分散する。次に大津のしきい値決定法 [12] に基づいて各 PE で異なる濃度範囲についての評価値を並列に求める。この処理では、各 PE が異なるパラメータ (濃度範囲) について同じプログラムを実行する。このような並列処理を PMPSD(Parameterized MPSD) 型並列関数適用と呼ぶ。大津のしきい値決定法は 3 つの過程からなる。(1) 自分の担当する濃度範囲の 0 次/1 次モーメント、ω/μをそれぞれ求める。(2)(1) で求めたモーメントに対し、並列パイプラインによるスキャン型演算を適用する。(3)PMPSD 型並列関数適用により、自分の担当する濃度範囲の局所的な評価値およびしきい値を決定する。その後、最適しきい値を決定し、RP2 で全 PE にブロードキャストする。
   (c) 二値化: 全 PE がしきい値を持っているので、並列に自分の保持する部分エッジ画像に対し並列に二値化を行う。
2. 画像解析
   この過程における ⟨ データ ⟩ と処理の流れを図 6に示す。⟨ 部分二値エッジ画像 ⟩→特徴点抽出 (SPMD)→⟨ 局所特徴点リスト ⟩→RP4 による統合・複写→⟨ 全特徴点リスト ⟩→部分パラメータ空間への投票 (PMPSD)→⟨ 部分パラメータ空間 ⟩→局所的ピーク検出 (SPMD)→⟨ 局所ピーク ⟩→大局的ピーク検出および RP2 による複写→⟨ 大局ピーク ⟩→ピークリストへの記憶 (SPMD)→⟨ 検出直線リスト ⟩→バックマッピング (PMPSD)[13]。直線を複数本検出するときには、ピーク検出からバックマッピングまでを繰り返す。ここでわれわれはρ-θHough 変換ではなく、パラメータ空間にひずみのないγ-ωHough 変換 [14] を用いて直線検出を行った。ただし、続く認識段階のために、検出された直線はρ,θパラメータ表現で保持している。
3. 認識

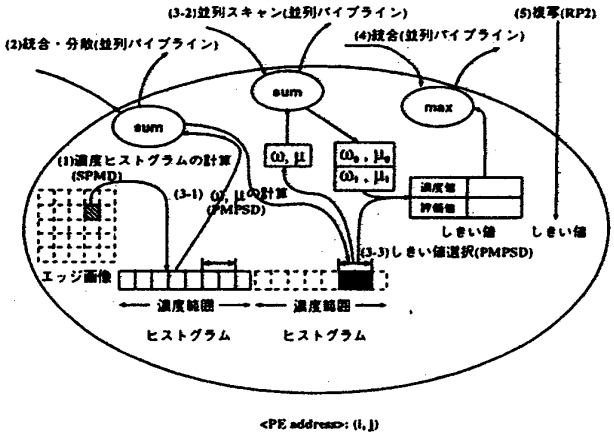

図 5: しきい値選択のデータと処理の流れ

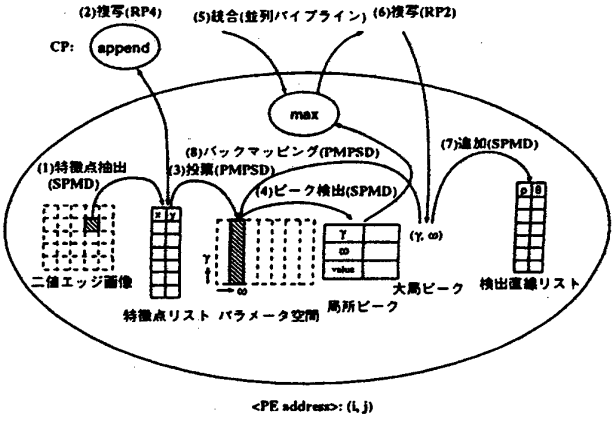

図 6: 直線検出のデータと処理の流れ

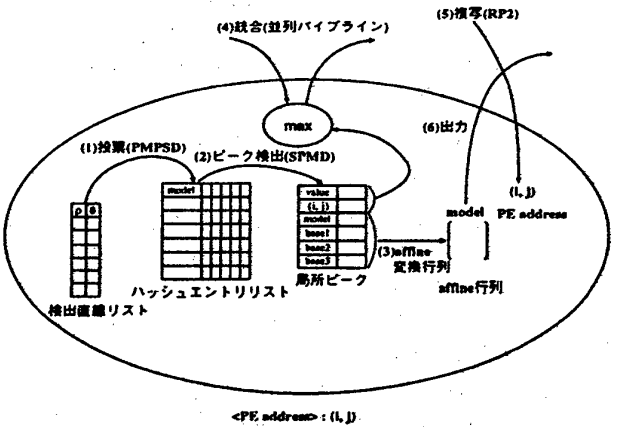

図 7: Geometric Hashing のデータと処理の流れ

モデルマッチングは直線ベースの Geometric Hashing を用いて行う。この過程における (データ) と処理の流れを図 7に示すが、この処理過程はほぼ Hough 変換と同様に実現できる。まずオフラインでモデルのコンパイルを行っておく。(モデル)→ハッシュエントリリストの生成→(ハッシュエントリリスト)→複写。そして続く認識部分を並列処理する。(検出直線リスト)→ハッシュエントリリストへの投票 (PMPSD)→(ハッシュエントリリスト)→局所ピークの検出および affine 変換行列の計算 (SPMD)→(局所ピーク)→大局的ピークの検出および RP2 による PE アドレスの複写→(PE アドレス)→対応する PE が認識モデルと affine 変換行列を出力。直線ベースの Geometric Hashing では 3 本の直線組を基底として残りの直線の affine 不変量を計算し、投票を行う。そこで、われわれは各 PE が異なる基底組について計算を行う PMPSD 型並列関数適用を用いて並列化を図っている。

5.2 性能評価

本節では、試作機 RTA/0 上で C 言語を使ってインプリメントした並列対象認識過程における実験結果と、表 2に基づいて RTA/1 の性能を評価した結果について述べる。表 3は画像処理についての結果である。実験画像は 512×480 画素、8bit の濃淡画像である。表 3を見ると RTA/1 は 1024PE 構成であるにも関わらず、しきい値選択以外では PE の台数倍以上の速度向上が見られる。これは、画像を分割することで 1PE で処理する場合に比べて、各 PE がアクセスする配列が小さくなり、C コンパイラがより最適なコードを生成するからである。微分演算において得られた値の濃度ヒストグラムは、0~1530 の範囲の値をとる。したがってしきい値選択においては各 PE が、1 か 2 濃度値しか処理しないことになり並列化による通信のオーバーヘッドの問題が表面化している。また、全処理時間に対して、画像分割・分散が全処理時間の約 83%を占めており、実時間処理を実現するためには、入力画像が CP を介することなく直接 PE に分割される画像入出力機構の検討が不可欠であることが分かる。

直線検出処理の性質を明らかにするために、いくつかの実験画像から直線を検出した結果を表 4に示す。表 4から以下のことが分かる。

1. 画像処理と同じ理由から投票とピーク検出で PE の台数倍以上の速度向上が得られる。
2. ピーク検出 (最大値選択) の方が投票 (投票先のパラメータ計算) より計算量が少ないため、配列のアクセスによる並列効果がより高いまま維持されている。
3. 特徴点数が増えるとバックマッピングの性能は低下する。この処理は検出されたピークに投票した特徴点を除去するものであるが、ピークに投票した特徴点 (バックマッピングで除去する特徴点) を抽出する処理は各 PE が全く同じことを行っており、並列効果は減少する。しかし、一度除去した特徴点は複数本の直線を検出するときには再計算されないので、検出直線数が増えるに連れて、バックマッピングの速度向上比は徐々に向上していくという結果が得られている。
4. 特徴点が偏っている場合と均一の場合の性質も実験した。その結果、特徴点の分布によっては並列効果が低下することが分かった。このことから特徴点数が均一になるような画像の動的分割による負荷分散を実現する方法についても検討の余地があることが分かる。
5. 逐次処理、並列処理ともに全体の計算時間は、特徴点数に比例する。

表 3: 画像処理の性能評価 (単位:秒)

処理	Single PE	RTA/1	Speedup Ratio
画像分割・分散 (PD2)	—	0.0802	—
ソーベルフィルタ (SPMD)	7.21	0.00692	1041.91
ヒストグラム (SPMD)	1.72	0.00165	1042.42
統合・分散 (並列パイプライン)	—	0.00652	—
しきい値選択 (並列パイプライン & PMPSD)	0.0308	0.00051	60.39
統合 (並列パイプライン)	—	0.000141	—
複写 (RP2)	—	0.000070	—
二値化 (SPMD)	1.38	0.00116	1189.66
計	10.34	0.097	106.60

表 4: 画像解析処理の性能評価 (単位:秒)

10 lines(2535 points)	Single PE	RTA/1	Speedup Ratio
特徴点抽出 (SPMD)	0.237	0.000234	1012.82
複写 (RP4)	—	0.00603	—
投票 (PMPSD)	21.253	0.0206	1031.70
ピーク検出 (SPMD)	1.203	0.00111	1083.78
統合 (並列パイプライン)	—	0.000150	—
複写 (RP2)	—	0.000070	—
バックマッピング (PMPSD)	2.150	0.00267	805.24
計	24.843	0.031	930.42
20 lines(5020 points)	**Single PE**	**RTA/1**	**Speedup Ratio**
特徴点抽出 (SPMD)	0.248	0.000266	932.33
複写 (RP4)	—	0.00906	—
投票 (PMPSD)	42.054	0.0404	1040.94
ピーク検出 (SPMD)	1.204	0.00111	1084.68
統合 (並列パイプライン)	—	0.000150	—
複写 (RP2)	—	0.000070	—
バックマッピング (PMPSD)	2.186	0.00327	668.50
計	45.692	0.054	846.15
30 lines(6892 points)	**Single PE**	**RTA/1**	**Speedup Ratio**
特徴点抽出 (SPMD)	0.258	0.00281	918.15
複写 (RP4)	—	0.01134	—
投票 (PMPSD)	59.414	0.0570	1042.35
ピーク検出 (SPMD)	1.204	0.00111	1084.68
統合 (並列パイプライン)	—	0.000150	—
複写 (RP2)	—	0.000070	—
バックマッピング (PMPSD)	2.218	0.00375	591.47
計	63.094	0.076	830.18
24 non-uniform lines(6000 points)	**Single PE**	**RTA/1**	**Speedup Ratio**
特徴点抽出 (SPMD)	0.252	0.000344	732.56
複写 (RP4)	—	0.01025	—
投票 (PMPSD)	50.415	0.0485	1039.4
ピーク検出 (SPMD)	1.204	0.00111	1084.68
統合 (並列パイプライン)	—	0.000150	—
複写 (RP2)	—	0.000070	—
バックマッピング (PMPSD)	2.195	0.00339	647.49
計	54.066	0.064	844.78
24 uniform lines(6000 points)	**Single PE**	**RTA/1**	**Speedup Ratio**
特徴点抽出 (SPMD)	0.253	0.000234	1081.20
複写 (RP4)	—	0.01025	—
投票 (PMPSD)	50.259	0.0482	1042.72
ピーク検出 (SPMD)	1.203	0.00111	1083.78
統合 (並列パイプライン)	—	0.000150	—
複写 (RP2)	—	0.000070	—
バックマッピング (PMPSD)	2.195	0.00338	649.41
計	53.910	0.063	855.71

表 5: 認識処理の性能評価 (単位:秒)

モデル: 5 角形 直線数: 13 本	Single PE	上段:RTA/1(単純分割) 下段:RTA/1(負荷分散)	Speedup Ratio
投票 & ピーク検出 (PMPSD & SPMD)	79.71	0.090 0.085	885.67 937.76
統合 & 複写 (並列パイプライン & RP2)	— —	0.00025 0.00025	— —
計	79.71	0.090 0.085	885.67 937.76

モデル: 山形 直線数: 12 本	Single PE	上段:RTA/1(単純分割) 下段:RTA/1(負荷分散)	Speedup Ratio
投票 & ピーク検出 (PMPSD & SPMD)	48.47	0.060 0.051	807.83 950.39
統合 & 複写 (並列パイプライン & RP2)	— —	0.00025 0.00025	— —
計	48.47	0.060 0.051	807.83 950.39

モデル: 蝶形 直線数: 14 本	Single PE	上段:RTA/1(単純分割) 下段:RTA/1(負荷分散)	Speedup Ratio
投票 & ピーク検出 (PMPSD & SPMD)	106.10	0.118 0.109	899.15 973.39
統合 & 複写 (並列パイプライン & RP2)	— —	0.00025 0.00025	— —
計	106.10	0.118 0.109	899.15 973.39

モデル: 星形 直線数: 12 本	Single PE	上段:RTA/1(単純分割) 下段:RTA/1(負荷分散)	Speedup Ratio
投票 & ピーク検出 (PMPSD & SPMD)	38.30	0.052 0.043	736.54 890.70
統合 & 複写 (並列パイプライン & RP2)	— —	0.00025 0.00025	— —
計	38.30	0.052 0.043	736.54 890.70

表 5にモデルマッチングの性能評価結果を示す。モデルとしては 5 角形、山形、蝶形、星形の単純なものを利用した。それぞれのモデルは 5 本、5 本、6 本、5 本の直線で表現されている。画像処理・画像解析で得られた直線数は表 5の通りである。この結果から以下のことが分かる。

1. 直線数が増えると基底組が指数的に増えるため、計算時間も増大する。
2. 直線ベースの Geometric Hashing は 3 本の直線で構成される基底組が平行であったり、1 点で交わる場合には投票を行わないため、データに依存して負荷分散が変化する。
3. 表 5の上段は単純に基底組の組み合わせを分割することによって並列処理を行った場合の結果である。このときは、2 に示した理由から、PE ごとに投票を行う基底組と行わない基底組の数がばらつくために、表のような速度向上になる。

上記 2 に着目し、簡単な負荷分散を行った場合についての実験も合わせて行った。ここで考えた負荷分散は以下のものである。平行でない基底組 (投票を行う基底組) の組み合わせを各 PE ごとに生成する。各 PE は生成した基底組の組み合わせと自分のアドレスから自分が処理するべき基底組の範囲を求め、PMPSD 型並列関数適用によって投票を行う。ここで、平行でない基底組の組み合わせを各 PE ごとに生成する処理は、各 PE が同じ処理を行っており、一般にはオーバーヘッドになると考えられるが、表 5の下段に示すように負荷分散の効果が得られていることが分かる。

6 おわりに

本論文では、RTA/1 のハードウェア設計とデータレベル並列処理の並列化の枠組を示し、その枠組に基づいて実際に RTA/0 上に対象認識過程をインプリメントした。そこで得られた実測値に基づき RTA/1 における性能評価を行った。性能評価の結果は、RTA の基本的な考え方、ハードウェア設計、および本論文で述べたデータレベル並列処理の並列化の枠組の有効性が示されたが、真の有効性を実証するためには、今後、以下の点について検討を進める必要がある。

(1) 今回試作した実験機 RTA/0 は試作の容易性および、安定な動作の観点からトランスピュータを用いた。今後はより高速な CPU、通信方式について検討を行う必要がある。
(2) 実時間処理を目指した性能を得るために必要となる画像の高速な入出力機構の検討。
(3) 新たな基本演算パターンについての検討。
(4) より一般的な負荷分散の方式についての検討。
(5) われわれが提案したデータレベル並列処理の並列化の枠組を容易に記述するための並列言語の設計。

この研究は文部省科学研究費一般研究 (B)(No.05452359) の援助を受けて行った。

参考文献

[1] 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.

[2] 山下 敦也 : 再帰トーラス結合アーキテクチャに基づく並列画像理解用計算機 RTA/1 の設計と試作, 岡山大学大学院工学研究科修士論文, 1995.

[3] 松山 隆司, 奥水 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.

[4] Lamdan, Y. and Wolfson, H.J. : Geometric Hashing: A General and Efficient Model-Based Recognition Scheme, Proc. of 2nd ICCV, pp.238-249, 1988.

[5] Tsai, F.C.D. : Robust Affine Invariant Matching with Application to Line Features, Proc. of CVPR, pp.393-399, 1993.

[6] Introduction to Data Level Parallelism, Thinking Machine Technical Report 86.14, 1986.

[7] Hillis, W.D. and Steele, G.L., Jr. : Data Parallel Algorithms, Communications of the ACM, Vol.29, No.12, pp.1170-1183, 1986.

[8] Hillis, W.D. : The Connection Machine, MIT Press, 1989.

[9] 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.

[10] 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, 情処研資, 93-CV-84-8, pp.55-62, 1993.

[11] Steele, G.L. Jr., 訳 井田昌之 : 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.

[12] Otsu, N. : A Threshold Selection Method from Gray-Level Histogram, IEEE Trans., Vol.SMC-9, pp.62-66, 1979.

[13] Gerig, G. : Linking Image-space and Accumulator-space: A New Approach for Object-recognition, Proc. of 1st ICCV, pp.112-117, 1987.

[14] 和田 俊和, 藤井 高広, 松山 隆司 : γ-ωハフ変換 — 可変標本化によるρ-θパラメータ空間のひずみの除去と投票軌跡の直線化 —, 電子情報通信学会論文誌, Vol.J75-D-II, No.1, pp.21-30, 1992.