一次元プロセッサアレイに基づく超高速画像処理システムの開発環境

許 昭倫[†] 藤田善弘[†] 岡崎信一郎[†]
佐藤 完[††] 天満 勉[†]

ビデオ画像処理アプリケーションの効率的な開発に向けた，SIMDプロセッサに基づく高速画像処理システムの開発環境について述べる．従来の高速画像処理装置では，専用のアセンブラやあらかじめ用意されたライブラリを組み合わせて行う簡易プログラミングが主流であり，高速なプログラムを効率良く開発することは非常に困難であった．本論文では，一次元SIMDプロセッサをターゲットとする高級言語ベースの開発環境について，1）画素更新波の考え方を用いることにより，C言語に対し最小限の拡張を加えた言語仕様のみで，一次元SIMD型の並列性が十分記述可能であること，2）従来の最適化コンパイル技術にSIMD型マシンに特有のPE群選択処理を加えることで，実用に耐えうる高級言語コンパイラが構成できること，そして，3）ビデオ画像処理プログラムの効率的な開発には高級言語コンパイラのみならず，プロファイリングや試行錯誤的パラメタチューニング機能を備えたソースレベルデバッガが有効であることを示す．

A Programming Environment for a Linear Processor Array Real-time Image Processing System

Sholin Kyo,[†] Yoshihiro Fujita,[†] Shin-ichiro Okazaki,[†]
Kan Sato[††] and Tsutomu Temma[†]

This paper describes a programming environment of a high performance SIMD linear array image processing system for efficient developement of video image processing applications. Existing high performance image processing systems usually provide only poor assembler based programming environments, by which efficient program developement is actually very difficult. In this paper, first a briefly extended data parallel C language is shown to be sufficient for programming SIMD linear arrays. Then its compiler developed using mostly conventional compiling techniques is shown to be sufficient for use in real applications. Finally, the source level debugger for it which possesses parameter tuning and profiling facilities is shown to be effective for efficient development of video image processing applications.

1. はじめに

実世界シーンを対象にするビデオ画像処理の技術は，いまだ発展途上にある．現状でのビデオ画像処理アプリケーションの開発ではほとんどの場合，膨大な量の画像サンプルを対象に，パラメータチューニングやアルゴリズム調整を繰り返す必要がある．これらの作業を効率良く行うためには，手軽にプログラム修正が可能な，高級言語をベースとした開発環境が不可欠である．しかし，これまでの高速画像処理装置では，専用のアセンブラやあらかじめ用意されたライブラリを組み合わせて行う簡易プログラミングが主流であり，高速なプログラムを効率良く開発することは非常に困難であった．

従来より，画像処理に向いた高速性とプログラム可能性を有する並列アーキテクチャとして，一次元SIMDプロセッサ（SIMD Linear Processor Array）が提案されている[1)~4)]．しかし既存の一次元SIMDプロセッサ型のシステムも，ほとんどがアセンブリ言語あるいは既定のライブラリの使用を前提としており，実用的な高級言語環境が提供された例はない．それに対し，筆者らはこれまでPCIバスの1ボード型一次元SIMDプロセッサであるIMAP-VISION[5)~7)]を開発し，さらにC言語を拡張したデータ並列言語1DC（One Dimensional C）とその言語処理系，1DCを用いた一次元的並列アルゴリズムの設計手法，そしてビデオ画像処理アプリケーション開発に向けたグラフィカル・ユーザ・インタフェース（GUI）の研究開発を進めてきた．

既存のデータ並列言語は，一次元SIMDプロセッサ

† NECインキュベーションセンター
NEC Incubation Center
†† NEC情報システムズ
NEC Informatec Systems, Ltd.

にとり冗長かつ複雑な言語仕様を有する場合が多い．一方，画像処理をターゲットとする場合では，すでにその分野で広く使われている汎用の高級言語との相違点を最低限におさえた言語仕様であることが望ましい．それは，ユーザの習得の負担を軽減するのみならず，既存のコンパイル技術の流用を十分可能とし，高性能なコンパイラの開発を容易にするからである．それに対し筆者らは，低～中レベルの画像処理の一次元SIMDプロセッサ向き並列化への指針として，PEアレイ上でシストリックに画素更新波を進める手法[10)~12)]を提案している．この画素更新波の考え方を用いれば，高級言語は既存の汎用高級言語の仕様に加え，間接アドレッシングを中心とする，一次元SIMDプロセッサのいくつかの基本動作の記述が可能であれば十分であることが，これまでの検討で明らかになった．

本稿では，C言語に対し最小限の仕様拡張を加える方針で設計された一次元SIMDプロセッサ用高級言語1DC，IMAP-VISIONをターゲットマシンとする1DC最適化コンパイラ，そして1DCソースデバッガについて，

(1) 画素更新波の考え方を用いることにより，Cに対し最小限の仕様拡張を加えるのみで，高級言語による一次元SIMD型の並列画像処理記述が十分に可能であること

(2) 従来のコンパイル技術にSIMD型マシンに特有のPE群選択処理を加えることで，実用に耐えうる最適化コンパイラが構成可能であること

(3) ビデオ画像処理プログラムの効率的な開発には，試行錯誤的パラメタチューニング機能やプロファイリング機能を備えた高級言語デバッガが有効であること

等の点を明確にする．2章では，まずターゲットマシンであるIMAP-VISIONについて簡単に述べる．3章では一次元SIMDプロセッサ用高級言語1DCの設計方針と言語仕様，そして4章では画素更新波の考え方に基づく，1DCによる並列画像処理の記述方法を示す．5章では1DC最適化コンパイラの性能を評価し，さらにそのSIMD型マシンに特有のPE群選択処理について述べる．6章では，ビデオ画像処理プログラムのデバッグに必要な諸機能を有した1DCソースデバッガのGUIについて述べる．

2. 基本一次元SIMDプロセッサ IMAP-VISION

一次元SIMDプロセッサ（**図1**）は，通常全PEを制御する1つのコントローラとPEアレイとで構成される．コントローラとPEアレイ間では，全PEへの命令および共通なデータのブロードキャスト，全PEのステータス値の集計，各PEのローカルメモリへの逐次アクセス等が行えるバスが存在する．各PEは，いっせいに同一の命令を実行するかあるいは休止するかの選択をローカルに行え（=実行の自律性），PEのローカルメモリへのアクセスには互いに異なるアドレスを用いること可能である（=アドレッシングの自律性）．PE間結合は，基本的には各PEはその左右のPEとの間，そして両端のPEの間にのみ，データ転送経路が存在する．ただし，より複雑なPE間結合を有するケースも存在する[2)]．以降，図1に示す構成を有するものを基本一次元SIMDプロセッサと呼ぶとする．

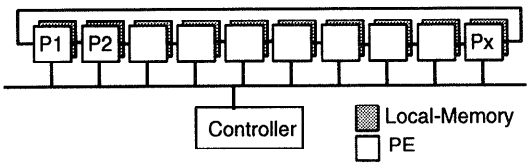


図1 基本一次元SIMDプロセッサの構成
Fig. 1 Composition of a basic SIMD linear processor array.

IMAP-VISIONは，8個のIMAP-VISIONチップ（計256PE），外部画像メモリ（計16MBのSDRAM），制御プロセッサ，そしてビデオインタフェース回路をフルサイズのPCIバスボード上に実装した超高速画像処理システムである．IMAP-VISIONチップは，IMAP（Integrated Memory Array Processor）アーキテクチャ[8),9)]に基づき，32個の8ビットプロセッサ（PE），1PEあたり1KBで計32KBの画像メモリ（SRAM），画像入出力用シフトレジスタ，そしてSDRAMインタフェースを集積したLSIであり，PEアレイ部は40MHz，SDRAMインタフェースは80MHzで動作する．**図2**にボードの構成，そして図**図3**にボード写真を示す．制御プロセッサ部（網掛け部分）は，制御プロセッサ（CP），64Kワードのプログラムメモリ，64Kワードのデータメモリ，そしてPCIバスインタフェース回路からなる．CPはPEアレイ全体の制御とともに，単独で16ビットRISCプロセッサとして，PEアレイ上での広域な演算も担当する．

IMAP-VISIONは多くの低～中レベルの画像処理（画像の前処理・特徴抽出処理）をビデオレートの数十から数百分の一の時間で処理可能である．ピーク性能は10GIPS，PEアレイとオンチップメモリ間のデータ転送レートは10GB/秒，PEアレイと外部メモリ（SDRAM）との間も1.28GB/秒という高いデータ供

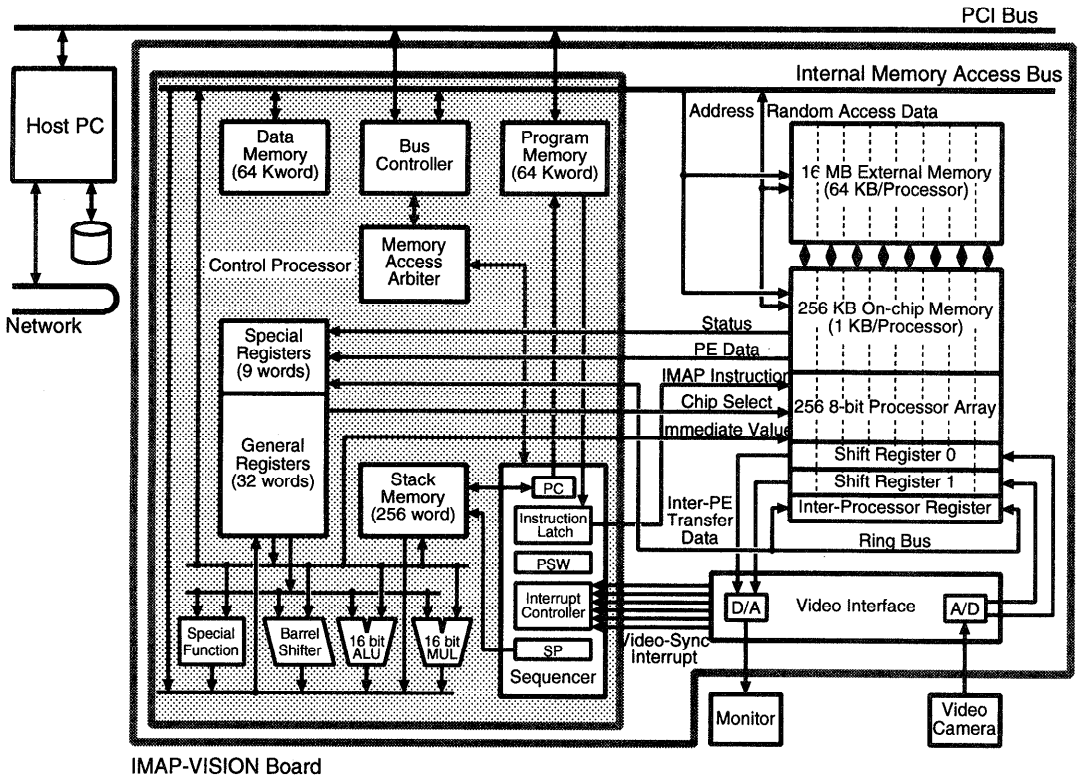


図2 PCIバス用IMAP-VISIONボードの構成
Fig. 2 Composition of the PCI bus IMAP-VISION board.

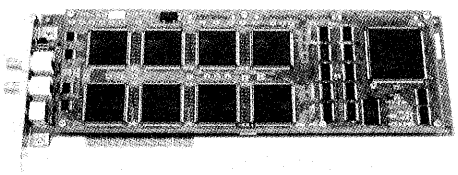


図3 IMAP-VISIONボードの写真
Fig. 3 Picture of IMAP-VISION board.

給能力を有する．また外界との画像入出力には，ビデオインタフェース回路（図2右下）が，1行の画像（行画像）がPEアレイの2本の画像入出力用シフトレジスタに入力し終わるたびに，CPに対し割込み信号を発行する．CPはそこで割込みプログラムを起動し，同シフトレジスタに溜まった行画像を画像入力用メモリ領域に格納すると同時に，画像出力用メモリ領域から行画像を読み，同シフトレジスタに格納する処理を行う．その処理は数マイクロ秒で終了し，これを33ミリ秒の間に画像の行数回だけ行うことで，ユーザプログラムとは独立でかつ並列な画像の入出力を実現している．

3. 1次元SIMDプロセッサ用高級言語1DC

3.1 設計方針

既存のデータ並列言語では，たとえば，コネクションマシンをターゲットとしたものでは様々なデータ集計（リダクション）演算子，またHPFのような汎用データ並列言語では，配列のPEアレイへのマッピング方式の指定が可能である等，高機能化されているものが多い．それに対し筆者らはこれまで，基本一次元SIMDプロセッサのPEアレイ上で，シストリックに画素更新波[10)~12)]を進めることにより，効率良く処理の並列化を行う方法を提案している．この画素更新波の考え方を用いる場合，画像の列ごとを各PEのローカルメモリに均等に割り付ける以外のマッピング方式はほとんど使われない．また画素更新波を進める動作の記述には，間接アドレッシングを中心とする，いくつかの基本一次元SIMDプロセッサの基本動作の記述が可能であれば十分である．

1DCの設計は上記の観点から，基本一次元SIMDプロセッサが行える基本的な並列動作である，1) PE

アレイによる並列演算，2）PE 間通信によるデータ交換，3）アクティブ PE 群の指定，4）アクティブ PE 群のステータス集計値に基づく全体制御，そして5）PE ごとの間接アドレッシング，について，既存の汎用高級言語（C 言語）からの簡単な仕様拡張で記述可能にし，それ以外の高機能化を狙った構文拡張は不要とした．C 言語を選んだ理由は，すでに画像処理研究の分野で広く使われていることによる．このように言語仕様上，既存の汎用高級言語との相違点を最低限におさえることは，ユーザの習得の負担を軽減できるのみならず，既存の最適化コンパイル技術の流用により，高性能なコンパイラの開発を容易にすることが期待できる．

3.2 言語仕様

前節で述べたように，1DC は基本一次元 SIMD プロセッサによる基本並列動作の記述に重点をおき，C 言語に必要最小限の仕様拡張を行う形で設計されたデータ並列言語である．C 言語からみたその拡張は，以下の 3 点に限定した．

変数宣言子 sep の追加 *sep* 付きで宣言された変数（*sep* または *separate* 変数）は各 PE のローカルメモリ領域，それ以外の変数（スカラ変数）は PE アレイ全体を制御するコントローラ上のデータメモリ領域に割り当てられる．

active-PE 指定文の追加 mif･･･melse, mfor, そして mwhile･･･mdo を，PE を複数のグループに分けるための active-PE 指定文として用意した．active-PE 指定文は sep タイプの値を返す式（sep 式）を評価し，その結果が 0 のグループ（非アクティブな PE 群）とそうでないグループ（アクティブな PE 群）に，PE アレイ内の PE を 2 分する．active-PE 指定文内の各文はアクティブな PE によってのみ実行される．active-PE 指定文をネストして記述すれば，さらに細かい PE 群の分割も可能である．その場合のアクティブな PE 群は，上位 active-PE 指定と下位 active-PE 指定の論理積によって指定される．なお関数呼び出しに対しては，呼び出し元の active-PE 指定が関数内で引き継がれる．各 active-PE 指定文は，対応する C の制御文の先頭に m を付けたものをその記法としている．

sep 変数を操作する演算記述子の追加 :(と:) のペア，:[と:] のペア，:>，:<，:||，:&&，:= のように，C の演算子の一部で先頭に:を付けた記法のものを用意した．

- :(と:) のペアは定数記述子である．c0,･･･,cn を定数とすると:(c0,･･･,cn :) とは，最左端 PE からみて 0 番目から n 番目までの PE 上に，それぞれ c0,･･･,cn の値，第 n+1 番目から最右端までの PE 上に 0 値，の要素構成を有する *sep* 定数である．*sep* 定数は *sep* 変数を初期化したり，定数演算の際に用いられる．
- :[と:] のペアは逐次参照記述子であり，*sep* 変数の各 PE 上での要素をコントローラから逐次的に参照する場合に用いられる．E_{sep}，E_{sca} をそれぞれ *sep* 式，スカラ式とすると，E_{sep}:[E_{sca} :] は E_{sep} の第 E_{sca} 番目の PE 上での要素の値を取り出す操作を意味する．
- :>と:< は通信記述子である．:>E_{sep} と :<E_{sep} は，それぞれ自分からみて左と右の PE 上の E_{sep} の要素値への参照を意味する．:>と:<は 2 項演算子としても使用でき，たとえば E_{sep}:>E_{sca} は，自分からみて E_{sca} だけ離れた場所にある PE 上の E_{sep} の要素値への参照を意味する．
- :&&と :|| は集計記述子である．:&&E_{sep} と :|| E_{sep} は，それぞれ現 active-PE 群にわたる E_{sep} の要素値の論理積と論理和を意味する．
- := は全代入記述子である．E_{sep}:=･･･ で記述された E_{sep} への代入処理は，現 active-PE 指定とは無関係に，全 PE を対象に行われる．本代入記述子は，現 active-PE 上に存在するデータに対し，全 PE を活用して処理を行う（後述する折返し型 PUW の生成がその典型例）場合に不可欠である．

1DC では，*sep* データ（*sep* タイプのデータ属性を持つ記述）をオペランドに持つ式（*sep* 式）かどうかにより，並列動作を行うかどうかを指定する．*sep* データをオペランドに持たない記述は，すべて PE アレイを制御するコントローラ上で逐次的に実行される．ユーザは明示的に 1DC プログラム中で *sep* 式を記述することにより，はじめて並列処理が行われる．なお *sep* データを戻値とする関数呼び出しも，*sep* 式と見なされる．**表 1** に，基本一次元 SIMD プロセッサの PE アレイ部での各動作と，1DC の各拡張仕様との対応を示す．

4. 1DC による並列画像処理記述

一次元 SIMD プロセッサの処理性能を最大限に引き出すためには，一次元的に結合された各 PE を，SIMD 動作の枠組み中で積極的に利用するような並列アルゴリズムが必要である．そうした並列アルゴリズムを設計するための有効な方法の 1 つに，画素更新波（Pixel Updating Wave: PUW）の考え方に基づく手法[10)~12)]がある．PUW に基づく並列化方式とは，画像メモリの二次元平面上に，処理可能な画素群をウェーブフロ

表1 各拡張仕様とPE動作との対応
Table 1 Correspondance between each extended language syntax and PE action.

動作	記述方法
PEアレイによる並列演算	*sep* 式をオペランドに使用
PE間通信によるデータ交換	:>, :< を使用
コントローラ経由のデータ交換	:[と:] のペアを使用
active-PE群の指定	active-PE指定文を使用
PEステータス集計値	:&&, :|| を使用
PEごとの間接アドレッシング	*sep* 配列添字に *sep* 式を使用

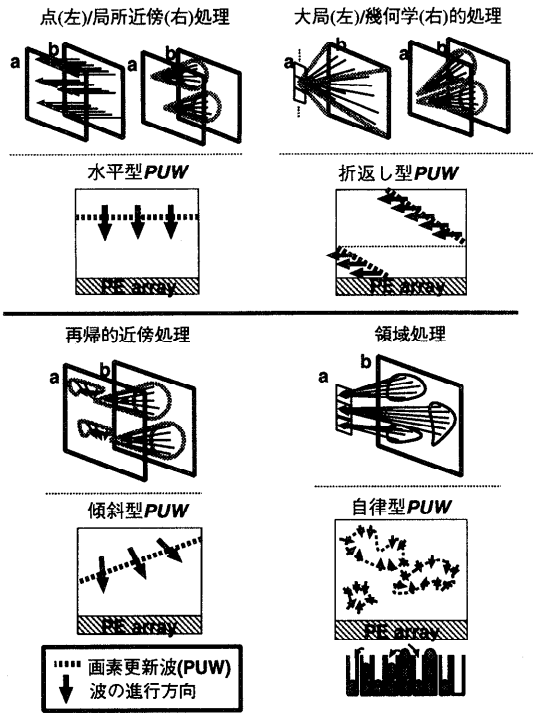


図4 画像処理の各カテゴリとその並列化に使用するPUWの種類
Fig. 4 Image processing task categories and the corresponding PUW for parallelizing them on SIMD linear processor arrays.

ント（WF）として表現し，そのWFを順次進行させ，各時刻ではWF上に位置する各画素を同時に処理対象とすることで，処理の並列性を得ようとする方式である．一次元SIMDプロセッサの各PEが異なるメモリアドレスをアクセスできるアドレッシングの自律性をフルに利用した並列化手法である．図4に，画像処理をその処理の形式によって分類して得られる各カテゴリ[13]（点，局所近傍，大局的，幾何学的，再帰的近傍，そして物体）と，それらを並列化する際に用いられる，水平型，折返し型，傾斜型，そして自律型の4種類のPUWとの対応付けを示す．続く各節では，これら各種PUWを生成するための1DC記述を与える．

4.1 水平型PUWの記述

水平型PUWは全画素を並列に処理可能な点処理（=画素ごとの処理），あるいは局所近傍処理の場合に適用できるPUWであり，PEアレイと平行なWFを有する．一次元SIMDプロセッサにとり，水平型PUWはすべてのPUWの基本形である．水平型PUWでは，繰返し回数はつねに対象画像の処理行数に等しい．画像間演算，各種二値や多値フィルタリング処理は，本PUWで画像を一掃する（全画素を1回ずつ訪れる）ことにより，容易に並列化可能である．

水平型PUWの生成のための1DC記述を以下に示す．WFの行位置は全PEで同一であるため，それを指定する変数にはスカラ変数iを用い，WFの進行はfor文でiを順次増分させる形で記述すればよい．なおWF位置の画素に対する処理はfor文の本体updateで指定し，update関数の内容は処理ごとに異なる．

```
#define uchar unsigned char
void Row_wise_PUW(src,lines)
sep uchar src[];        // 対象画像の sep 配列
int lines;              // 対象画像の処理行数
{
  int i;
  for(i=0;i<lines;i++) // 行数回だけ WF を進行
     update(src+i); // WF 位置の画素情報を更新
}
```

4.2 折返し型PUWの記述

折返し型PUWは，たとえば画素濃度のヒストグラムのように，画像全体から定まる統計的な量の計算や，画像内のラベル付けされた領域ごとの面積/重心/周囲長等を求めるような集計的な処理，そして画像回転のような画素の位置変化をもたらす幾何学的な処理の並列化に有効である．このPUWは，一次元SIMDプロセッサにおいて最右端と最左端のPEが互いに隣接関係にあることを利用したWFを生成する．このWFで画像を一掃するのにかかる反復数はつねにPE数に等しい．

折返し型PUWで右から左へ進むWFを生成する場合の1DC記述を以下に示す．PENOはPE数を表すスカラ定数，PENUMはPE番号（最左端から順に0,⋯,255）を表す *sep* 定数とする．反復制御にはfor文を用いる．WF位置は各PEで異なるため，その指定にはsep変数を用いる．反復ごとのWFの位置更新には，WFを進める方向により，1DCの通信記述子 :<（右から左への場合），:>（同左から右への場合）のいずれかを使用する．for文の反復回数はつねにPENOに等しくする必要があるため，対象の *sep* 配列の行数がPENOより少ない場合は場合分け記述

の追加，PENO より多い場合は 2 つ以上の for 文に分割する等の工夫が必要となる．折返し型 PUW はその性質上，現 active-PE 指定にかかわらずつねに全 PE を動作させる必要がある．それには全 PE 代入記述子 := を使用する．処理ごとに異なる update 関数を作成する際でも，代入記述にはつねに := を使用する必要がある．

```
void wraping_PUW(src)
sep uchar src[];        // 集計対象の配列
{
  int i;
  sep uchar s:= PENUM;// WF 位置の初期化 (全 PE)

  for(i=0;i<PENO;i++){// PE 数回だけ WF を進行
    update(src+s);    // WF 位置の情報を集計 (全 PE)
    s := :<s;         // WF 位置更新 (全 PE)
  }
}
```

たとえば，二値画像をラベル付けした後の各ラベル領域の面積を求めるには，水平型 PUW，折返し型 PUW をこの順で 1 回ずつ発行すればよい．以下にその 1 DC 記述例を示す．まず水平型 PUW の発行により *sep* 配列 wrk に列ごとのラベル面積を格納する．続く折返し型 PUW の発行で，ラベル N の面積は最終的に *sep* 変数 r の N 番目の PE 上の要素値として求まる．なお簡単のため，ラベル値 N の範囲は 0 以上 PENO 以下とした．

```
void wraping_PUW_area(src,lines)
sep uchar src[];     // 対象画像の sep 配列
int lines;           // 対象画像の処理行数
{
   int i;
   sep uchar s,r,wrk[PENO];

// 水平型 PUW:列ごとのラベル面積算出
   for(i=0; i<PENO; i++)  wrk[i]:=0;
   for(i=0; i<lines; i++) wrk[src[i]]++;

// 折返し型 PUW:画像全体のラベル面積算出 (全 PE)
   s := PENUM;        // WF 位置の初期化
   r := 0;            // 面積結果の初期化
   for(i=0; i<PENO; i++) {
      // WF 位置のラベル面積加算後，伝搬
       r:= :<(wrk[s]+r);
      // WF 位置更新
       s:= :<s;
   }
}
```

4.3 傾斜型 PUW の記述

各画素が自分の右上，上，左上，左の画素が処理済みになってはじめて処理が可能となる場合，斜めの WF（傾斜型 PUW）を発生することで，こうした隣接画素どうしでの処理可能となるタイミングのずれを満足させることができる．ラスタスキャン順の処理や，計算方法が再帰的な局所領域での定義として書けるような処理が，このカテゴリ（再帰的近傍処理）に属する．たとえば，画像の左上から右下に 1 回，右下から左上に 1 回の，計 2 パスのスキャンにより二値画像の距離変換を実現する処理[14]の並列化は，傾斜型 PUW で 2 回画像を一掃することにより実現される．

傾斜型 PUW の 1 DC 記述では，x 方向（PE が並ぶ方向）と y 方向（画像の列方向）の両方向への WF の進み具合を制御するための 1 組の sep 変数（sx と sy）を用いる．反復制御には for 文を用いる．各画素が制約を受ける近傍の大きさを M とすると，反復ごとの WF 位置は，sy を M-1 反復おきに増分あるいは減分させながら，かつ sx の非 0 の値を 1 画素ずつ（左から右へ，あるいは右から左へ）伝搬させながら，sx が非 0 の PE 上の sy によって表現する．たとえば，各画素が制約を受ける近傍が自分の右上，上，左上，左の画素である場合の M は 2 となる．また sx の非 0 値の初期位置と sy の初期値，sx を伝搬する方向，そして sy を増分するか減分するかの選択により，PUW が画像を一掃する際の開始位置と方向（左上→右下，左下→右上，右上→左下，右下→左上）を変えることができる．この WF で画像を一掃するのにかかる反復数はつねに M ×（画像行数 −1）＋PENO に等しい．左上→右下方向の傾斜型 PUW を生成する場合の 1 DC 記述を以下に示す．

```
void slant_PUW(src,lines)
sep uchar src[];  // 対象画像の sep 配列
int lines;        // 対象画像の処理行数
{
  int i;
  sep uchar sx=0,sy=0;

  for(i=0;i<M*(lines-1)+PENO;i++){

  // sx の初期非 0 要素の位置
    sx:[0:]= (sy:[0:] < lines);

  // sx が非 0 時，M-1 おきに PE を動作させる
    mif(sx & ((i-PENUM)%M==0)){
       update(src+sy);// WF 位置の画素情報更新
       sy++;          // WF 位置の更新
    }

  // sx を PE 方向 (左→右) へ伝搬
    sx = :>sx;
  }
}
```

4.4 自律型 PUW の記述

輪郭追跡処理等のように，処理の進行状態が画像内容に左右されたり，検出対象の形状に依存したり，い

わゆる物体処理のような場合，PUW を自律的に画像内で進行させる必要がある．これには各 PE のローカルメモリ上に，WF の候補点となる画素位置を溜めておく場所としてのスタック領域を設ける．まず全画素を対象に初期の WF 位置（タネ画素位置）を検出し（seed_detection），それらを当該画素の処理を担当する PE のスタック上にプッシュ（push）しておく必要があるが，これには水平型 PUW で画像を一掃することで実現する．続く自律型 PUW は，以下の処理の繰返しで実現される．すなわち空でないスタックがあるかどうかを調べ（stack_check），あればそこからポップ（pop）した画素位置の集合を現時点の WF としたうえ，WF 位置の画素の情報更新（update）を行うと同時に，WF の周りで，次回以降 WF となることが可能な画素位置があるかどうかを調べ，あればその位置情報を PE のスタックにプッシュ（check_and_push）する．この stack_check → pop → update → check_and_push を，全 PE のスタックが空になるまで繰り返す[12]．自律型 PUW の利用により一次元 SIMD プロセッサ上で効率的に並列化可能な処理には，ラベリング，細線化，輪郭/線分追跡，領域拡張（region growing），そして動的輪郭（snakes）等，多くのものが存在する．

自律型 PUW を生成する 1DC 記述を以下に示す．自律型 PUW における繰返し制御には while 文と mif 文の組合せを使用し，push と pop の動作には *sep* 配列 stack のインデックに *sep* 変数 sp を使用した間接アドレッシングにより実現する．isSeed() はタネ画素位置検出ルーチンであり，処理ごとに異なる．反復ごとの WF 位置は，反復のたびスタックからポップされる画素位置 y で指定する．check_and_push() も処理ごとに異なるが，実行可能画素の検出，検出画素をプッシュするのに必要な active-PE 群の設定，そして実際に検出画素位置を stack にプッシュする処理等を行う．

```
void autonomous_PUW(src,lines)
sep uchar src[];    // 対象画像の sep 配列
int lines;          // 対象画像の処理行数
{
   int i;
   sep uchar sp,r,y,stack[256];

// 水平型 PUW:タネ画素検出と初期 push
   for(sp=0,i=0;i<lines;i++)
      mif(isSeed(src+i)) // 条件を満足すれば
         stack[sp++]=i;  // push を行う

// 自律型 PUW のループ
   while(:||sp) {
      mif (sp) {
      // ポップ：WF 位置更新
           y = stack[--sp];
      // WF 位置の画素情報を更新
           r = update(src+y);
      }
    // 派生する WF 候補位置を検出し push
      check_and_push(src+y,r);
   }
}
```

5. 1DC 言語処理系

図 5 に IMAP-VISION をターゲットマシンとした 1DC 言語処理系の構成を示す．1DC コンパイラ（ccimap）は 1dc1（構文解析・中間コード生成），1dc2（中間コード最適化），1dc3（active-PE 群生成の最適化），1dc4（アセンブリコード生成），そして 1dc5（アセンブリコード最適化），の 5 つの処理パスからなる．1DC 記述を C の記述に変換するパス（1dcc）も存在し，それにより 1DC プログラムを通常の逐次プロセッサ上で実行させることも可能である．その他，リンカ（ldimap），ソースレベルデバッガ（sdbimap）およびアセンブリレベルデバッガ（xdbimap），そして IMAP-VISION を実装しないホスト上でも 1DC プログラムの動作シミュレーションが可能なシミュレータ（simimap），等が存在する．ccimap の処理パスのうち，1dc4 と 1dc5 にターゲットの命令セットに依存する部分があるほかは，ターゲット非依存である．そのため ccimap は，基本一次元 SIMD プロセッサの要件を満たす他の一次元 SIMD プロセッサにも容易に移植可能である．

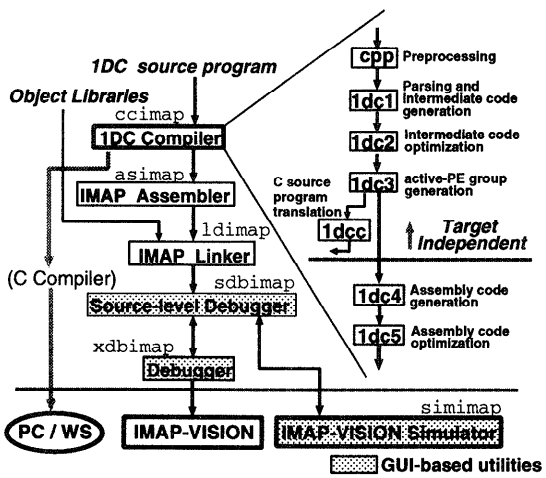


図 5 IMAP-VISION システム用 1DC 言語処理系
Fig. 5 1DC language processor for IMAP-VISION.

5.1 ccimap における最適化処理

ccimap の全処理パスのうち，1dc2 が中間コード，1dc3 が active-PE 群指定コード，そして 1dc5 がアセンブリコードの生成処理を対象に最適化を行う．**表 2** にこれらの最適化パスによる主な処理を示す．

ccimap は，1 DC が持つ C に対する簡潔な拡張仕様と，ターゲットマシンである IMAP-VISION が有する汎用プロセッサに類似した RISC 命令セットにより，1dc2 と 1dc5 では，従来からの汎用プロセッサ向け最適化コンパイル技術[15]が十分に流用できた．それらと比べ，1dc3 による active-PE 群生成にかかわる最適化処理は，SIMD 型アーキテクチャに独特なものである．1 DC では，初期状態では全 PE がアクティブであるが，mif や mfor 等の active-PE 指定文が出現すると，PE アレイはアクティブな PE とそうでない PE とに細分化される．1dc3 では，ます各基本ブロック（Basic Block: BB）を解析し，BB 間の active-PE 群の相違に関する情報を求めることで，互いに異なる active-PE 群によって実行される可能性のある隣接 BB どうしを特定する．次にそのような BB の入口と出口それぞれに，active-PE 群生成，active-PE 群情報保存のための中間コード列を追加することにより，過不足のない active-PE 群の生成を実現している．

5.2 ccimap の性能評価

ここでは，いくつかの 1 DC 記述に対するコンパイラ生成コードと，人手による最適化アセンブラコードとの実行ステップ数を比較する．**表 3** に，それぞれ異なるカテゴリに属する 4 つの簡単な画像処理プログラム，1）単純二値化（点処理），2）3×3 平滑化処理（局所近傍処理），3）ヒストグラム処理（大局的処理），そして 4）90 度回転処理（幾何学的処理），に対する，(a) 最適なアセンブラプログラム，(b) 最適化前のコンパイラコード，(c) 最適化後のコンパイラコードの実行ステップ数とその比較結果，および最適化後のコンパイラ生成コード（c）の実行時間を示す．異なるタイプの基本的な画像処理について，最適化を行った場合のコンパイラ生成コードによる実行では人手によるアセンブラコードの 1.2〜1.5 倍程度の処理時間を達成しており，コンパイラによっても十分に高速な画像処理アプリケーションの開発が可能であることを示している．

表 2 各最適化パス内での主な処理
Table 2 Major operations of the optimizing passes.

1dc2（中間コード）	ビープホール最適化 制御フロー解析 データフロー解析 ポインター追跡 共通部分式削除
1dc3（active-PE 群指定コード）	active-PE 群指定解析 active-PE 群生成最適化
1dc5（アセンブリコード）	ビープホール最適化 遅延スロット充填 レジスタ割付け最適化 ビデオ割込み関連最適化

6. ビデオ画像処理のデバッグ環境

一般に高級言語デバッガの要件は，ソースコードの任意箇所でのブレイクと，ブレイク時の任意変数あるいは配列の値を表示する機能を提供し，それにより，プログラムの任意箇所までの計算結果の正確性の検証を支援することにある．一方ビデオ画像処理の場合ではさらに，膨大な量の画像サンプルを対象に，パラメータチューニングやアルゴリズム調整を繰り返してはじめて，処理結果の正確性を検証できる場合がほとんどである．そこで，ビデオ画像処理を対象とする高級言語デバッガは，前記の一般的な要件に加え，実世界の様々なシーンを対象に，効率良くパラメータチューニングや動作検証を行うための機能，そしてプログラムの動作時間を要求スペック（たとえばビデオレート）内に収めるための性能チューニング機能，等を備えることが不可欠である．

1 DC では通常の C 言語と同様，処理は main という名前の関数から開始する．この main 関数は，ビデオの偶数（even）フレームが入力し終わった直後であり，かつ前の main 関数が実行終了している場合に，そのつど再起動される．フレームごとの処理はこのように main 関数を繰り返し起動することにより実現し

表 3 1 DC コンパイラの性能評価
Table 3 Evaluation of the 1 DC compiler.

ソース名	a	b	c	b/a	c/a	c の実行時間
1）単純二値化	1547	12029	1883	7.78	1.22	0.061 ms
2）3×3 平滑化	5600	17888	6456	3.19	1.15	0.161 ms
3）濃度ヒストグラム	4039	29004	5742	7.18	1.42	0.122 ms
4）90 度回転	20696	48610	24536	2.35	1.19	0.583 ms

ているが，フレーム間を跨る情報のやりとりを実現するために，グローバルな変数領域は，初回の main 関数の起動時を除き初期化されない．プログラム実行中に，これらのグローバル変数に外部から適宜な値を書き込むことで，当該変数内に蓄積された履歴情報をリセットできる．あるいは，プログラムの動作を規定するパラメータとしてのグローバル変数であれば，その値を動的に変更してやることで，変更した時点以降のシーンに対するプログラム動作を容易に変えることができる．そこで，1DC ソースデバッガ sdbimap では，行単位のブレイクポイント設定やブレイク時の任意変数値の表示等，通常の高級言語デバッガと同等な機能のほか，こうしたプログラム実行中でのグローバル変数の参照・定義，それにプログラムの処理時間解析のための機能に向けた GUI として，以下のものを用意した．

グローバル変数（配列）の値を随時表示する機能 各グローバル変数のフレームごとの値を監視できる GUI として，オートリフレッシュウィンドウを用意した．ユーザは変数名をマウスで選択した後，所定のボタンをクリックすれば，当該変数の内容をキャラクタ（*sep* 変数の場合は 256 個の値，*sep* 配列の場合は画像フォーマット，それ以外は 1 個のスカラ値）で随時更新して表示するオートリフレッシュウィンドウを開くことができる．

グローバル変数値をインタラクティブに調整する機能 フレームごとに，処理結果であるビデオシーンを見ながら，変数値をインタラクティブに調整できる GUI として，1 次元スライダ，択一ボタン，そして 2 次元スクロールマップを用意した（**図 6**）．プログラム動作中にユーザが，1 次元スライダではスライダをドラッグする，2 次元スクロールマップではマップ上のシンボルをドラッグする，そして択一ボタンではいずれかのラジオボタンをチェックすると，対応するグローバル変数（2 次元スクロールマップの場合は対応する 1 組のグローバル変数）のメモリ領域へ，対応する値（2 次元スクロールマップの場合はシンボルの x，y 座標値）を書き込む操作が行われ，それにより次フレーム以降の main 関数の動作を変更可能である．

行/関数単位の処理時間計測機能 行/関数単位の処理時間を計測し解析する GUI として，プロファイルウインドウを用意した．ユーザは関数名あるいは行番号をマウスで指定すれば，指定範囲内の各ソース行の動的な処理時間の計測値のキャラクタ表示と，処理時間最大のソース行に対応する箇所を赤で表示した棒グラフを内容とするプロファイルウインドウを開くことができる．本機能は，デバッグオプション付きでコンパイルしたコードに対してのみ正確に動作するが，開発中のアプリケーション内に存在する演算ボトルネックとなる箇所の検出には，十分な効果を発揮できる．

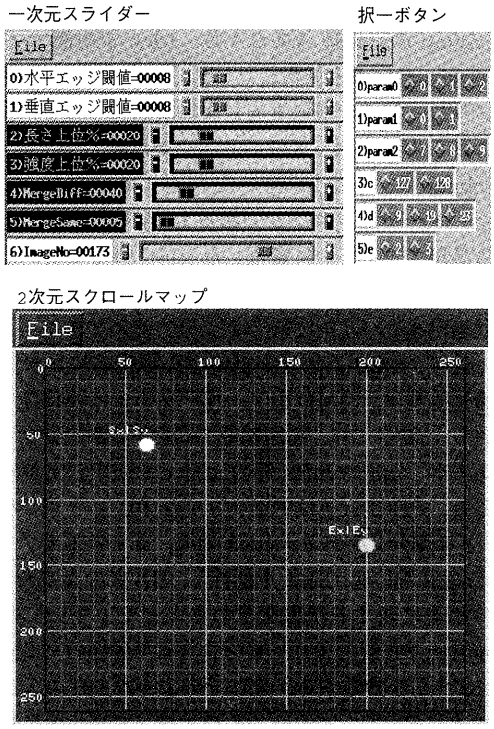


図 6 sdbimap で用意された 3 種類のグローバル変数値調整ツール
Fig. 6 Three kinds of global variable adjusting tools provided by sdbimap.

GUI を用いたパラメータ値の参照・定義操作機能は，ビデオ画像として即座に観測される処理結果と，時々刻々のパラメータ値の変化とが，どのように関連付けされているかをインタラクティブに操作者に伝え，効率的なプログラムの動作検証手段を提供する．また処理時間解析機能は，処理全体のボトルネックとなる箇所の検出を支援する．ユーザはプログラムのそうした箇所を再チェックすることで，処理時間に対する要求スペックを満足したプログラムを効率良く開発できる．こうした試行錯誤的パラメタチューニング機能，およびプロファイリング機能の存在は，特に性能スペックへの要求が厳しく，かつ検証すべきデータ数・シーン数が膨大であるビデオ画像処理アプリケーションを開発する場合，その効果は明確である．sdbimap を用いたビデオ画像処理アプリケーション開発の様子を**図 7** に示す．

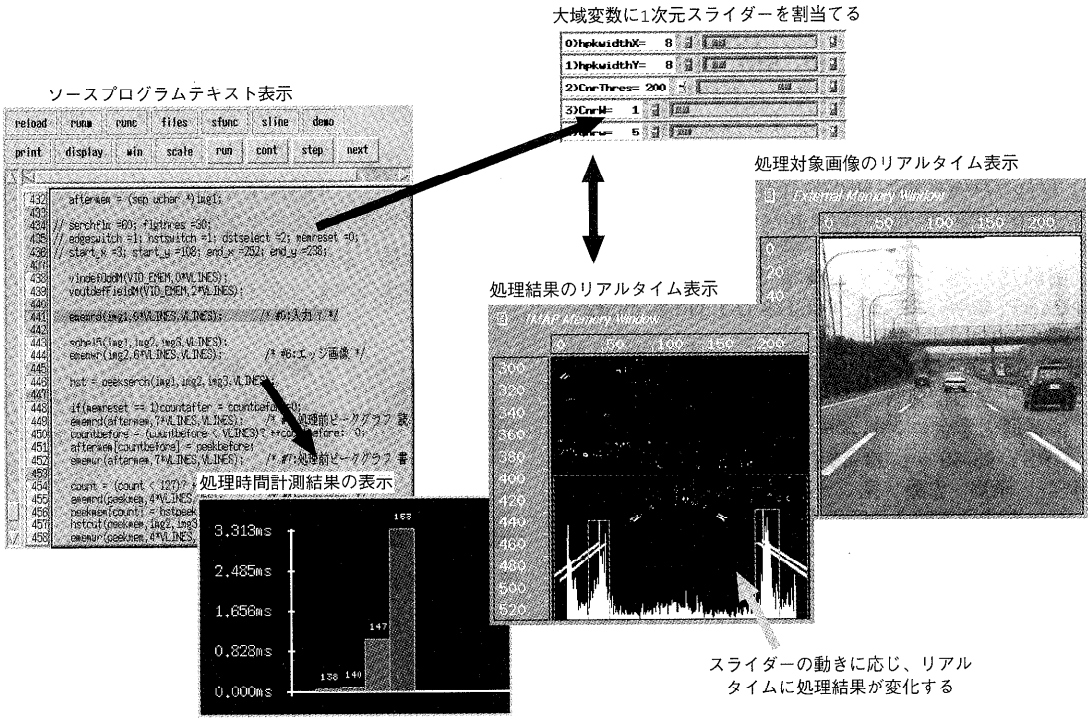


図7 sdbimapを利用したビデオ画像処理プログラム開発の流れ
Fig. 7 Developement of video image processing applications using sdbimap.

7. むすび

従来の高速画像処理装置では，専用のアセンブラやあらかじめ用意されたライブラリを組み合わせて行う簡易プログラミングが主流であり，高速なプログラムを効率良く開発することは非常に困難であるという問題点が存在していた．これに対し我々は，一次元SIMDプロセッサをターゲットマシンとする高級言語ベースの開発環境を提案し，以下の事項を明らかにした．

(1) 画素更新波の考え方を用いることにより，C言語に対し最小限の拡張を加えた言語仕様のみで，一次元SIMD型の並列性が十分記述可能である．

(2) 従来の最適化コンパイル技術にSIMD型マシンに特有のPE群選択処理を加えることで，実用に耐えうる高級言語コンパイラが構成できる．

(3) ビデオ画像処理アプリケーションの効率的な開発には高級言語コンパイラのみならず，プロファイリングや試行錯誤的パラメタチューニング機能を備えたソースレベルデバッガが有効である．

実世界画像を対象とするビデオ画像処理アルゴリズム開発には，多数の画像データを用いたアルゴリズム検証が必要なのに加え，現状では依然として多数のパラメータによる動作チューニングを必要とする場合が多い．これには十分な計算パワーと，効率の良いプログラミング環境を兼ね備えた画像処理システムが不可欠である．提案の高級言語ベースの開発環境を備えた一次元SIMDプロセッサは，低～中レベルの多くの画像処理に対し，高い並列処理性能をユーザに手軽に提供可能である．本開発環境を備えた一次元SIMDプロセッサシステムが今後，様々なビデオ画像処理応用の研究開発に役立つと考えられる．

参考文献

1) Fountain, T.J.: The CLIP7A Image Processor, *IEEE Trans. on PAMI*, Vol.10, No.3. pp.310–319 (1988).

2) Chin, D., Passe, J., Bernard, F., Taylor, H. and Knight, S.: The Princeton Engine: A Real-Time Video System Simulator, *IEEE Trans. on Consumer Electronics*, Vol.34, No.2, pp.285–297 (1988).

3) Astrom, A. and Forchheimer, R.: MAPP2200 Smart Vision Sensor: Programmability and Adaptivity, *Proc. IAPR Workshop on Machine Vision and Applications* (*MVA '92*), pp.17–20 (1992).

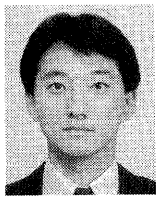
4) Childers, J.: SVP: Serial Video Processor,

IEEE 1990 Custom Integrated Circuits Conference (Mar. 1990).
5) 岡崎信一郎, 藤田善弘, 許 昭倫, 山下信行：1ボード超高速動画像処理システム IMAP-VISION, 第3回画像センシングシンポジウム講演論文集, pp.201–206 (1997).
6) http://www.incx.nec.co.jp/imap-vision/
7) http://image.mitene.or.jp/rvs/
8) 藤田善弘, 山下信行, 木村 亨, 中村和之, 岡崎信一郎：メモリ集積型 SIMD プロセッサ IMAP, 信学論 (D-I), Vol.J78-D-I, No.2, pp.82–90 (1995).
9) 許 昭倫, 岡崎信一郎, 藤田善弘, 山下信行：メモリ型画像処理プロセッサとその応用, 電子情報通信学会コンピュータシステム研究会報告, CPSY-93-50, pp.39–46 (1994).
10) 許 昭倫, 佐藤 完：一次元プロセッサアレイ用データ並列言語1DCによる画像処理アルゴリズムの記述とそのコンパイラ, 情報処理学会計算機アーキテクチャ研究会資料, ARC119-17, pp.95–100 (1996).
11) Kyo, S. and Sato, K.: Efficient Implementation of Image Processing Algorithms on Linear Processor Arrays using the Data Parallel Language 1DC, *Proc. IAPR Workshop on Machine Vision and Applications* (*MVA '96*), pp.160–165 (1996).
12) Kyo, S., Okazaki, S., Fujita, Y. and Yamashita, N.: A Parallelizing Method for implementing Image Processing Tasks on SIMD Linear Processor Arrays, *Proc. IEEE Workshop on Computer Architecture for Machine Perception* (*CAMP '97*), pp.180–184 (1997).
13) Jonker, P.P.: Architectures for Multidimensional Low- and Intermidiate Level Image Processing, *Proc. IAPR Workshop on Machine Vision Applications* (*MVA '90*), pp.307–316 (1990).
14) Borgefors, G.: Distance Transformations in Digital Images, *Computer Vision, Graphics, and Image Processing* (*CVGIP*), Vol.34, pp.344–371 (1988).
15) Aho, A.V., Sethi, R. and Ullman, J.D.: *Compilers: Principles Techniques and Tools.* 原田賢一（訳）：コンパイラ原理・技法・ツール, サイエンス社 (1990).
16) Maya, G. and Pfeiffer, P.: SIMD Optimizations in a Data Parallel C, *Proc. International Conference on Parallel Processing* (*ICPP '93*), pp.2:188–2:191 (1993).

(平成 9 年 10 月 20 日受付)
(平成 10 年 4 月 3 日採録)

許 昭倫（正会員）

1963 生. 1987 年東京大学工学部精密機械工学科卒業. 1989 年同大学大学院修士課程修了. 同年日本電気（株）に入社. 1994 年から 1995 年にかけてオランダ・デルフト工科大学客員研究員. 現在日本電気インキュベーションセンターに所属. 並列画像処理に向けた計算機アーキテクチャ, 言語, アルゴリズム等の研究開発に従事.

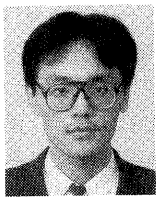


藤田 善弘（正会員）

1960 年生. 1984 年京都大学工学部電子工学科卒業. 1986 年同大学大学院修士課程修了. 同年日本電気（株）に入社. 1993 年から 1994 年にかけてカリフォルニア州立大バークレイ校客員研究員. 現在日本電気インキュベーションセンターに所属. 並列画像処理システムの研究開発に従事.

岡崎信一郎

1959 年生. 1982 年大阪大学工学部電子工学科卒業. 1984 年同大学大学院修士課程修了. 同年日本電気（株）に入社. 現在日本電気インキュベーションセンターに所属. 高速並列画像処理システムの研究開発に従事. 電子情報通信学会会員.



佐藤 完

1962 年生. 1985 年山形大学理学部数学科卒業. 同年日本電気技術情報システム開発に入社. 現在 NEC 情報システムズ技術システム事業部に所属. 画像処理, 画像処理システム用プログラム開発/実行環境, データ並列言語コンパイラ等の研究開発に従事.

天満 勉

1947 年生. 1969 年大阪大学工学部電子工学科卒業. 同年日本電気（株）に入社. 現在日本電気インキュベーションセンター長. パターン認識, 高速並列画像処理システムの研究開発に従事. 電子情報通信学会会員.