コンピュータビジョンと
イメージメディア 104－10
（1997．3．27）

並列画像処理プログラミング支援システム PRIME

牧山 靖* 鶴田 直之* 谷口 倫一郎* 雨宮 真人*
* 九州大学大学院 システム情報科学研究科 知能システム学専攻

筆者らは，異機種での並列画像処理プログラムの再利用を可能にし，容易なプログラミング環境を提供する並列画像処理プログラミング支援システム PRIME (PaRallel Image Media processing Environment) を開発している．本稿では特に，通信に基づいてデータ並列型画像処理プログラミングを支援する部分の実装を行う．まず，並列画像処理アルゴリズムが通信に基づいて少ないカテゴリに分類できることに注目して，並列画像処理アルゴリズムを分類する．そして，各カテゴリの通信構造をシステムに隠蔽して実装する．これにより，画像処理の局所計算を異機種間で共有することができるようになる．また本稿では，分散メモリ環境と共有メモリ環境にデータ並列支援の PRIME を実装して，高い記述性と高い性能を実現したものであることを確認した．

PRIME : PaRalell Image Media processing Environment

Yasushi MAKIYAMA *, Naoyuki TSURUTA *,
Rin-ichiro TANIGUCHI *, Makoto AMAMIYA *
* Graduate School of Information Science and Electrical Engineering,
Kyushu University

PRIME is a programming support system for parallel image processing. This system makes it possible to share programs of parallel image processing among machines which have different architectures. To share programs, we have classified parallel image processing algorithms into several categories from the viewpoint of intercommunication, and *PRIME* hides the intercommunication patterns of these categories from the programmer. In this paper, we present the method to implement these intercommunication algorithms, and report some experimental results on both a machine with distributed memory and a machine with shared memory.

1 はじめに

画像処理では一つ一つの処理は複雑ではないが，数百万に及ぶ画素に対して計算を行なったり，動画像のような数百枚の画像に対して計算を行うことが普通となってきている今，膨大な計算量による処理時間の増大が問題になっている．しかし，画像処理は一般的に処理が局所的に独立しているものが多く，高い並列性が期待できる．問題は，その高い並列性を実現するためのプログラミングが容易でないことである．

近年，並列画像処理の困難なプログラミングを支援するためのプログラミング言語 Apply[1], Adapt[2] や並列画像処理システム AMP[3][4] が開発されている．しかし，これらは特別なハードウェア専用の言語やシステムなので，異機種間での並列画像処理プログラムの共有·蓄積が不可能である．そのため，ユーザがプログラムを異機種において記述する際に，過去に記述したプログラムの再利用ができず，無駄に多くのプログラムを記述し直さなければならない．また，並列画像処理プログラムの作成経験のないユーザの場合は，さらに困難なプログラミングを行わなければならない．そこで，我々は，異機種間でのプログラムの再利用を可能にし，並列画像処理プログラムの作成経験のないユーザでも容易にプログラミングできることを目指した並列画像処理プログラミング支援システム PRIME(PaRallelized Image Media processing Environment) を作成している．このシステムは，C++という汎用的な言語の枠の中で実現しているため，様々な応用システムに組み込むことができる．

PRIME の主要な目的は，並列画像処理プログラミング記述の支援と異機種間でのプログラムの共有·蓄積を行う環境を提供することであり，以下のような特徴を持つ．

1. **並列画像処理プログラミング支援**

   データ並列及び機能並列による並列画像処理プログラムを容易に作成できるプログラミング環境を提供．

2. **異機種間でのプログラムの共有・蓄積支援**

   共有メモリ環境，分散メモリ環境に実装可能であり，並列画像処理プログラムの部分プログラムを共有できる．

3. **分散環境における画像入出力支援**　ファイル入出力及び画像データの分散を支援する．

本稿では，PRIME が目指しているデータ分割と機能分割の両方における支援のうち，データ並列におけるプログラミング支援及び異機種間でのプログラム共有・蓄積支援を実現する方法を提案する．まず，はじめに支援を実現するための並列画像処理の実装方式を述べ，その後，MPI(Message Passing Interface)[5] が実現する分散メモリ環境と，共有メモリ計算機上に データ並列支援の PRIME を実装し，その有効性を検討する．

2 並列画像処理の実装方式

2.1 データ構造とデータ交換

データ並列型画像処理は，画像データを幾つかの部分画像集合に分割し，各部分画像集合の処理を並行して実行することである．データ並列画像処理の実装は，部分画像集合を割り当てるメモリのタイプにより大きく異なる．並列環境の多数のプロセッサが読み書きするメモリには，多数のプロセッサでそれを共有するか，個別的に所有するかの二つに大別される．前者を共有メモリ，後者を分散メモリと呼ぶ (図 1)．

- **共有メモリ**

  複数のプロセッサからアクセスされるメモリであり，通常バスを介して複数個のプロセッサで共有されている．プロセッサ間のデータ交換は高速に行えるが，メモリアクセスがネックとなりやすい．そのため，各プロセッサは，バスへのアクセス頻度を下げるためのキャッシュメモリを持つ．データ交換に要する時間は，キャッシュメモリへのキャッシングパラダイムに左右される．

- 分散メモリ

  各プロセッサにメモリを分割する方式で，各プロセッサの持つメモリを局所メモリと呼ぶ．プロセッサ間通信は相互結合網を通して行われる．プロセッサからのメモリアクセスがネックとならず，高並列向きであるが，一方プロセッサ間のメッセージ交換などに時間がかかる．プロセッサの粒度[1] に応じて局所メモリのサイズも変わり，その機能も変化する．

分散メモリ型並列環境に一般的な並列画像処理を実装する場合は，その機種固有のメッセージ通信が実装されているため，その通信方法におけるデータ転送のオーバーヘッドをなるべく少なくするようなデータ分散でなければならない．例えば，全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ことや，分割の形態(分割領域の形状，大きさや重なりの有無) といった選択肢がある．その点，共有メモリ型並列環境に実装する場合には，データの格納方式はほとんど問題とならない．

PRIME では，並列画像処理プログラムの共有と蓄積を実現するために上記の機種に依存するプロセッサ間通信を行う部分をシステムに隠蔽する必要がある．そこで，並列画像処理アルゴリズムにおいて，プロセッサ間通信に関する機種依存部を隠蔽する．

2.2 並列画像処理の実装方式

一般にデータ並列画像処理アルゴリズムには，三つの主要な処理がある．以下，各々の処理について説明する．

(1) 画像の分散及び収集に伴うデータ交換

並列処理の前後に実行される処理である．

- **画像の分散**

  画像を部分画像に分割し，各部分画像をプロセッサに割り当てる．分散の種類としては，データ集中型，データ複写型，短冊状の分散，格子状の分散等がある．

- **各プロセッサで得られた処理結果の収集**

  各プロセッサで得られた処理結果を 1 つのプロセッサに集める．

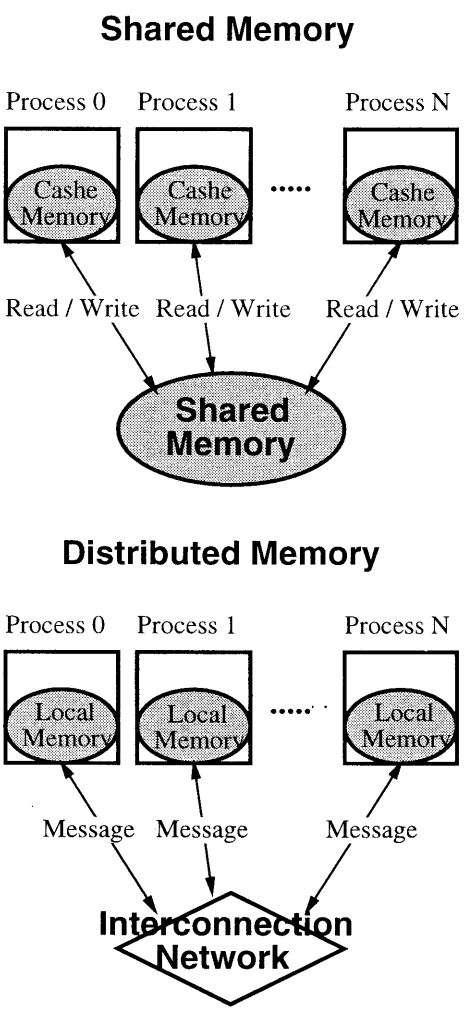

図 1: 並列環境のメモリタイプ

[1] 並列処理の単位，基本プロセッサの規模，性能，複雑度などを意味する．この場合は基本プロセッサの規模のこと．粒度の大小により，粗粒度，中粒度，細粒度と分類されている．

(2) 画像処理の局所計算本体

並列処理中に実行される処理である．各プロセッサは，割り当てられた部分画像に対して画像処理の局所計算を実行する．

(3) 画像処理の局所計算本体同士のデータの交換

並列処理中に実行される処理である．画像処理の局所計算の際，割り当てられた部分画像以外に処理に必要な情報は，プロセッサ間通信により，他のプロセッサから得る．

以上の処理のうち，(2) の部分は画像処理毎に対応する局所計算があるため非常に多様であるのに対し，(1) 及び (3) の部分はそれほど多様でない．例えば，フィルタリングを行う場合は，画像を分散した後，各プロセッサが近傍情報を得るためのプロセッサ間通信を行いつつ，局所フィルタリング計算を実行し，最後に各処理結果を出力画像の形に統合することになる．この処理のうち，局所フィルタリング計算を除いた部分の処理は，どのフィルタリング処理をみても全く同じである．

そこで，本稿で提案する実装方式では，(1) を画像データの型と分散の形態によってタイプ付けされる，データとデータ交換関数の構造体を表す Distributed Image Data (DID と略) と呼ぶクラスとして抽象化し，実装する．(3) は，画像処理の局所計算本体を関数として受け取る高階関数[2] を用いて実装する．(1) 及び (3) をそれぞれシステム内に隠蔽することにより，(2) の画像処理の局所計算を異機種間で共有できる．また，通信に関する難しいプログラミングをすることなく，並列画像処理プログラムを記述することができる (図 2)．以下，(1) および(3) の実装方法について詳細を述べる．

[2]高階関数とは，関数を引数とする関数，あるいは，関数を引数とし関数を返す関数のことをいい，抽象度の高いプログラムを作成することができる機構である

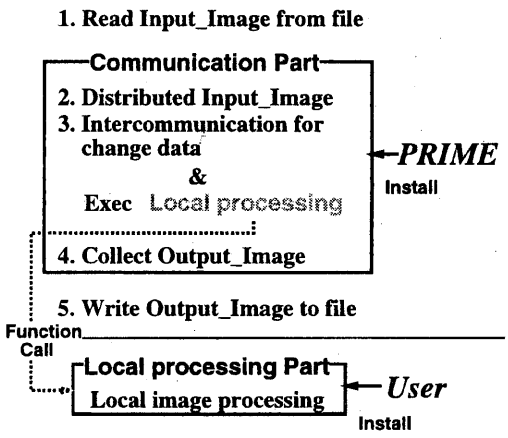

図 2: 並列画像処理の処理手順

2.3 画像の分散及び収集に伴うデータ交換の実装方式

2.3.1 分散データ表現

実装対象の機種が，分散メモリ型環境であれば固有のメッセージ通信方法，また，共有メモリ型環境であれば共有メモリへの Read/Write といった機種依存のプロセッサ間通信方法をを持つため，効率の良い画像分散方法は異なる．また，2 値画像と多値画像では，画素値がとり得る値の範囲が異なるため，画像のデータの型の違いに応じたデータ交換を実装しなければならない．そこで，画像データ型と分散の種類の両方に基づいて，データとデータ交換通信をクラス化し，このクラスに基づいて並列画像処理でよく用いられる分散の種類を分類する．そして，DID(Distributed Image Data) として実装する．以下，並列画像処理において用いられる分散表現を定義する．

集中型 (Integrate)
画像データを一つのプロセッサ上に集めて配置する．

複写型 (Copy)
画像データを全てのプロセッサ上に複写して

配置する．

水平短冊状分散型

(HorizontalSkip,Horizontal)

画像を水平に短冊状に分割し，各部分画像データをプロセッサに割り当てる．分散方法に，画像を 1 行ずつ順番にプロセッサに分配する方法と，プロセッサ数と同数個のほぼ等しい部分画像データに分割する方法の 2 つがある．

垂直短冊状分散型

(VerticalSkip,Vertical)

水平短冊上分割と同様に，画像を垂直に短冊状に分割し，部分画像データをプロセッサに割り当てる．

格子状分散型 (Grid)

画像を $n \times m$ の格子状に分割し，各部分画像データを $n \times m$ 個のプロセッサに割り当てる．

これらの分散データ表現を実現するため，DID は部分画像データへのポインタ及び分散表現を管理するための分散表現情報を持つ．

2.3.2 Distributed Image Data

DID は以下の情報を保持している．

- 部分画像データへのポインタ
- 分散表現を管理するための分散表現情報
  - 部分画像の位置，サイズ
  - 全体画像のサイズ
  - プロセッサ数，プロセッサ ID

実装した主な DID を表1に示す．ユーザが PRIME プログラムを記述する際には，DID の分散構造を意識する必要はなく，画像全体が DID というクラスで表現されていると考えて記述すれば良い．例えば，2 値画像へのポインタをもつ DID オブジェクトを生成する場合，

```
Image2D_Binary
   bimage(Horizontal,''pict1.pbm'');
```

と記述して生成する．上記の記述では，2 値画像ファイル pict1.pbm を読み込み，水平短冊状分散に画像データを分散して，**bimage** を生成する．

データの種類	DID クラス名
2 値画像	Image2D_Binary
濃淡画像	Image2D_Gray
カラー画像	Image2D_RGB
ラベル画像	Image2D_Label
ペア画像	Pair
画像系列	Sequence

表 1: 主な DID クラス

濃淡画像やカラー画像等といった多値画像の場合は，2 値画像と異なり，画素値の範囲が一意に決まっておらず，画素値の型が限定できない．そこで，C++ の機能であるテンプレート機能[3] を利用して，濃淡画像クラスを実装する．濃淡画像クラスを用いて，濃淡画像へのポインタをもつ DID オブジェクトを生成する場合，

```
Image2D_Gray<char>
  gimage(Vertical,''pict2.pgm'');
```

と記述すれば良い．上記の記述では，濃淡画像ファイル pict2.pgm を読み込み，垂直短冊状分散に画像データを分散して，**gimage** を生成する．

2.4 画像処理の局所計算本体同士のデータ交換の実装方式

局所計算同士のデータ交換を，「**並列画像処理アルゴリズムにおいてプロセッサ間通信にはいくつかのよく使用されるパターンがある**」という観点から，通信に基づいて少ないカテゴリに分類できることに注目して，表 2のように通信による並列像処理アルゴリズムを整理する．そして，各カテゴリの通信のパターンを対象とする並列環境に実装する．この通信のパターンを**通信構造**と定義する．通信構造は機種依存なので，機種固有の通信を最大限に利用して実現できる．

[3] どんな型でもあり得る **sort()** のような総称関数を型の族に対し一つ定義することを可能にする．静的な型チェックや実行効率を失うことなしに簡単に定義し実装することができる．

通信関数	対象画像処理の例
分割統治演算型	濃度ヒストグラム生成，連結成分のラベル付け
フィルタリング型	局所オペレータを用いた平滑化やエッジ検出等の処理
条件反復フィルタリング型	弛緩法を用いたエッジ検出やマッチング等の処理
バタフライ演算型	FFT
階層処理演算型	階層型データ構造を利用したエッジ検出

表 2: 通信関数と画像処理例

画像処理の局所演算本体を関数として与えることを考えて，高階関数を利用して実装する．高階関数を用いて実装した，このデータ交換を**通信関数**と呼ぶ．この画像処理の局所演算本体の関数を**計算関数**と呼ぶ．ユーザは，計算関数を作成し，システムが提供する通信関数与えることにより処理アルゴリズムを作成することができる．

例えば，フィルタリング型の通信関数を用いて，3 × 3 平均化フィルタによる画像の平滑化を行う場合を考える．フィルタリング型の処理内容は，

通信関数

入力： 画像，計算関数，マスクサイズ
出力： 画像画素値
振舞： 近傍情報取得通信→計算関数の実行

計算関数

入力： マスク，マスクサイズ
出力： 画素値
振舞： 画素近傍を含むウィンドウ (N×M) に対して行なう計算

である．3 × 3 平均化フィルタ計算関数は次のように記述する．

```
int
  Average3x3(int** mask,
             int rx,int ry,
             void* arg)
{
  return
    (mask[0][0]+mask[0][1]+mask[0][2]
     +mask[1][0]+mask[1][1]+mask[1][2]
     +mask[2][0]+mask[2][1]+mask[2][2])
     /9;
};
```

この計算関数と平滑化処理対象の DID オブジェクトをフィルタリング型通信関数`Filter`に与えることにより，並列画像平滑化アルゴリズムを作成できる．

```
Image2D_Gray<char> gimage2();
Filter(&gimage1,&gimage2,Average3x3,
       (char*)0,3,3,255);
```

ただし，エンドユーザは，通信関数，計算関数等の関数の入出力となるデータの流れを把握しておく必要がある．この高階関数による記述方式は，計算関数という形でプログラムを共有できるだけでなく，プログラムの可読性も良い．

3 記述支援と並列化の検証

データ並列支援の PRIME を

- 共有メモリ計算機の CRAY Superserver 6400，ソラリススレッドを用いた並列処理
- MPI によるワークステーションクラスタ環境

に実装して実験を行った．局所的な情報の参照のみで処理を行えるメディアンフィルタによる画像の平滑化と大局的な情報を参照しなければならない連結成分のラベリングを行った．ただし，CS6400 と MPI において，計算関数は全く同じものを使用した．

(平滑化アルゴリズム)

画像を水平短冊状に分割して並列処理を行う．なお，メディアン値をクイックソートを用いて求める．(フィルタリング型の通信関数を使用．)

(ラベリングアルゴリズム)

画像を水平短冊状に分割し，部分画像に対して反復型のラベリング処理 (図 4) を並列して行う．統合処理では，各部分ラベル画像の分割境界を見て，ラベルの対応テーブルを作成する．その後，テーブルを深さ優先探索して新ラベルのテーブルを作成し，

Laplacian Filter Program

```
1 #include <prime.h>
2 int laplacian(int** mask,int rx,int ry,
                void arg)
3 {
4   return
      (-mask[0][0]-mask[0][1]-mask[0][2]
       -mask[1][0]+8*mask[1][1]-mask[1][2]
       -mask[2][0]-mask[2][1]-mask[2][2]);
5 }
6 void main(int argc,char** argv)
7 {
8   PRM_Init(&argc,&argv);
9   Image2D_Gray<int> image1(Horizontal,
10                           "picture1");
11  Filter(&image1,&image1,laplacian,
12         (char*)0,3,3,255,1);
13  image1.save("picture2");
14  image1.display();
15  PRM_Finalize();
16}
```

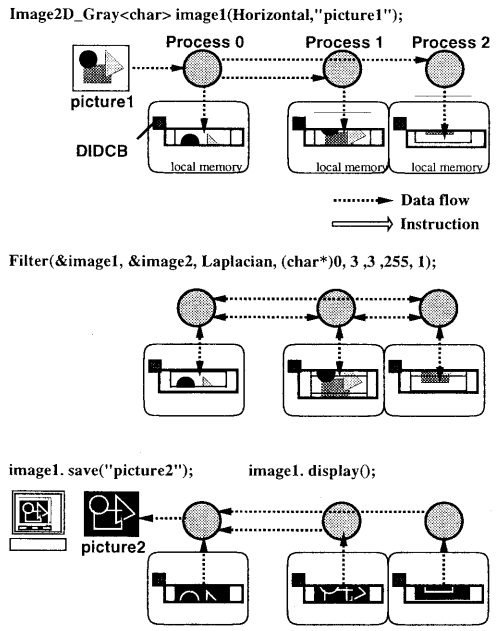

図 3: PRIME を用いたプログラム例

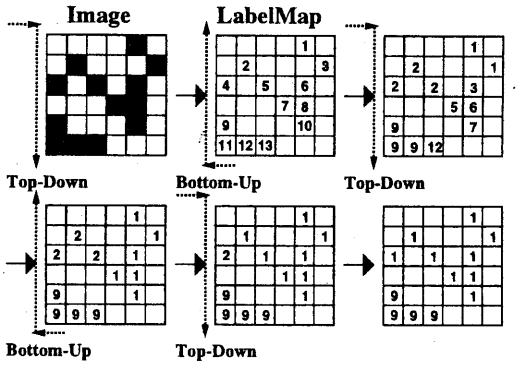

図 4: 反復型処理によるラベリング

最後にラベルを新ラベルに変更する．(分割統治演算型の通信関数を使用．)

図 5 に，PRIME を使用した場合と，MPI のみを使用して記述した場合のプログラムの記述行数を示す．PRIME では，ファイルの入出力，分散及び収集と局所計算本体同士のデータ交換をうまく切り分けし，関数として実装しているため，その部分を関数呼出しで記述できる．そのため，ユーザが記述する部分は画像処理の局所計算本体だけであり，高い記述性が実現できている．

各並列画像処理の処理時間を計測し，逐次プログラムの処理時間と比較して台数効果を求めたものを図 6に示す．平滑化では非常に高い並列性を抽出することができた．ラベリングでは，プロセッサ数が多くなると統合処理に時間がかかり，逆に速度が落ちている．しかし，ピーク時には，CS6400 においてプロセッサ数の約 6 割り，MPI においてはプロセッサ数の約 4 割りの台数効果が得られた．次の図 7には，PRIME を使用した場合と，ソラリススレッドのみ，あるいは，MPI のみを使用して記述した場合の比較グラフを示している．高い記述性を提供する処理を含んでいるため，PRIME の方が少々劣る結果となった．PRIME では，フィルタリング型の計算を行なう際，画素数と等しい回数だけ計算関数を呼び出す．その関数呼び出しのオーバヘッド

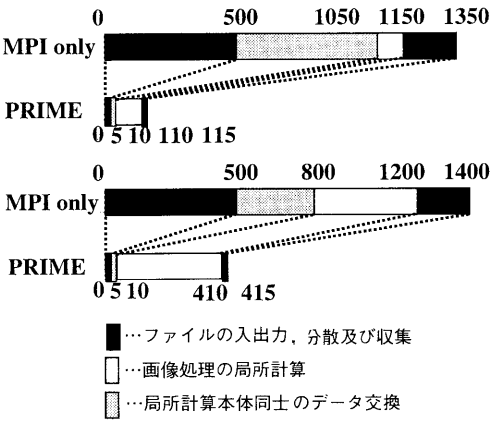

図 5: 各並列画像処理の記述行数

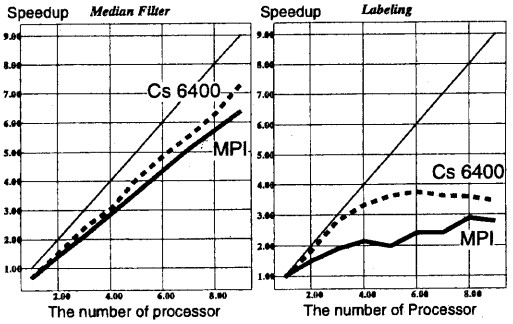

図 6: 各並列画像処理の台数効果

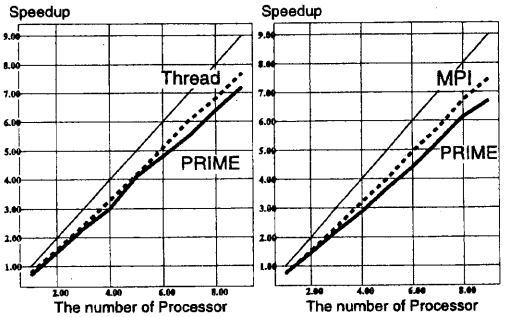

図 7: PRIME の台数効果

が理由と考えられる．

4 おわりに

本稿では，並列画像処理プログラミング支援システム PRIME が提供する支援の内，データ並列におけるプログラミング支援及び異機種間でのプログラム共有・蓄積支援を実現する方法を提案した．提案した方法で実装した PRIME が高い記述性を提供できる上に，高い並列性が抽出できることを，実験により確認した．また，異機種間でプログラムが共有できることも確認できた．

今後は，データ並列のみならず，機能並列画像処理における支援を実現する．そして，データ並列と機能並列により，並列画像処理よりも高レベルな並列画像認識，並列画像理解への応用を考える．

参考文献

[1] Wallace,R.S,Webb,J.A,and Wu,I.: Mashine-independent image processing:performance of Apply on diverse architecture, *Proc. of Image Understanding Workshop*, pp.602-608(1988).

[2] Webb,J.A.: Unifying Concepts in Architecture Independent Parallel Image Processing in Adapt, *Proc. of SPIE Image Processing and Interchange*, pp.228-239(1992).

[3] 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，情報処理学会論文誌，Vol.32，No.7，pp.933-939(1991)．

[4] 山元，堀田，谷口，雨宮：画像処理用超並列プロセッサ AMP のプログラミング言語 Valid-A について，情報処理学会第 40 回全国大会講演論文集 (II)，pp.547-548(1990)．

[5] MPI フォーラム著，MPI 日本語訳プロジェクト訳：MPI：メッセージ通信インタフェース標準 (日本語訳ドラフト)，(1996)．