7G-03

Parallella アーキテクチャを用いたユークリッド距離変換の並列化

東村　将志 †　　森本　康彦 ‡
広島大学大学院　工学研究科　情報工学専攻 †
広島大学　大学院工学研究院　情報部門 ‡

1. はじめに

近年のマルチコアプロセッサ技術の向上やプラットフォームの小型化に伴い，Parallella と呼ばれる並列・並行コンピューティングプラットフォームが Adapteva 社 [1] によって開発された．Parallella は 16 コアの演算用アクセラレータを搭載し，Mesh ネットワークによる並列処理を可能とする．

ユークリッド距離変換とは，白黒の 2 値画像を入力とし各画素についてそこから最も近い黒画素への距離を求める処理で，ディジタル画像処理における基本的な処理である．

本研究においては，画像処理アルゴリズムの 1 つである 2 次元ユークリッド距離変換アルゴリズムを Parallella 上に実装し，処理の並列化を図る．実験では，複数のプラットフォームによる逐次処理と Parallella による並列処理の実行時間及び消費電力を計測・比較を行い，Parallella アーキテクチャの性能を評価する．

2. Parallella

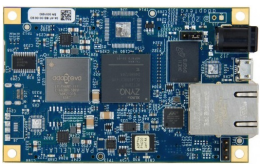

図 1: Parallella Board

2.1. Parallella Architecture

Parallella アーキテクチャは図 2 のようになっている．Parallella Board にはホストプロセッサとしての Xilinx Zynq-7010 チップ，コプロセッサとしての Epiphany チップが組み込まれている．1GB DRAM チップには 32MB のホストプロセッサ及び Epiphany コプロセッサの両方がアクセスできる Shared Memory 領域が存在する．Epiphany チップには 16 個の演算用アクセラレータが Mesh ネットワークで接続されており，DMA エンジンを用いて高速ネットワークを可能にする．また，各コアが 32KB の Local Memory を保持している．

Parallel Computation of Euclidean Distance Transformation using the Parallella Architecture
Masashi Higashimura†, Yasuhiko Morimoto‡
Department of Information Engineering, Hiroshima University†
Institute of Engineering, Division of Information Engineering ‡

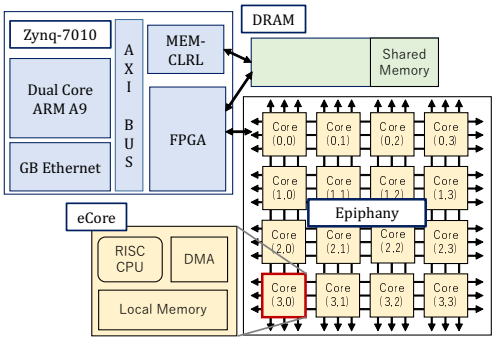

図 2: Parallella Architecture

3. 2 次元画像のユークリッド距離変換

この節では $n \times n$ サイズの白黒の 2 値画像を入力とし，各画素 $p_{i,j}(0 \leqq i, j < n)$ についてそこから最も近い黒画素への距離を求めるユークリッド距離変換について説明する．最も近い黒画素の位置を $p'_{i',j'}(0 \leqq i', j' < n)$ とすると，求める距離 $d_{i,j}$ は以下の式によって求められる．

$$d_{i,j} = \sqrt{(i-i')^2 + (j-j')^2}$$

この処理を実行すると，図 3 のような出力結果を得る．

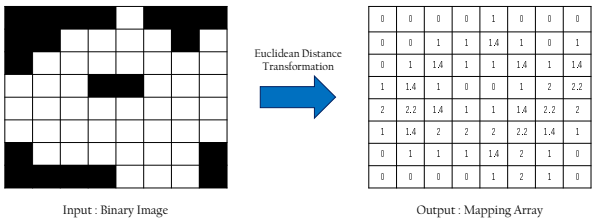

図 3: 入力画像および出力配列

距離 $d_{i,j}$ を求めるために [2] で用いられたアルゴリズムを用いる．

Algorithm Euclidean Distance Transportation (EDT)

EDT 1.
各画素 $p_{i,j}$ に対して列方向における最も近い黒画素との距離 $d_{i,j}$ を求める．まず上から下に向かって黒画素との距離 $dub_{i,j}$ を計算し，次は下から上に向かって距離 $dbu_{i,j}$ を計算する．そして，$d_{i,j} = min\{dub_{i,j}, dbu_{i,j} | 0 \leqq i, j < n\}$ を用いて，距離 $d_{i,j}$ を求める．(図 4)

EDT 2. 各 $j, (0 \leqq j < n)$ において $P_j = \{p_{i,j} = (i, d_{i,j}) | 0 \leqq i < n\}$ を定義する．そして各 P_j の最近点の集合 $E(P_j)$ を求める．

EDT 3. $E(P_j)$ 内の各点 p において図 5 のように P_j の

表 1: 実行時間 [ms]

Platform	16^2	32^2	64^2	128^2	256^2	512^2	消費電力 [W](64^2)	消費電力 [W](512^2)
Desktop PC	0.019	0.064	0.249	1.023	4.297	18.826	37.2	45.8
Raspberry Pi 2	0.267	1.033	3.435	13.853	57.971	257.654	1.7	1.7
Parallella	0.043	0.146	0.790	1.772	9.933	40.368	5.8	5.8

最近点領域を決定する．

EDT 4. 各 $i, (0 \leqq i < n)$ において P_j 内の点 $p_{i,j}$ が対応する最近点領域を決定し，距離 $d_{i,j}$ を更新する．

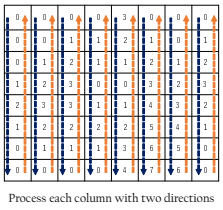

図 4: EDT - 1

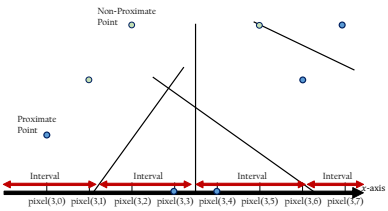

図 5: 最近点領域

4. Parallella への実装手法

Parallella での実装について説明する．Parallella は搭載されているメモリが非常に小さいため，3. で示したアルゴリズムを画像サイズに応じて 2 つの手法に分ける．画像サイズが 64×64 以下の場合（Algorithm1）とそれ以上の場合 (Algorithm2）に分ける．

Algorithm 1

Step 1. ホスト側から Shared Memory に転送した画像を列方向に 16 分割し，各コアの Local Memory に転送

Step 2. 各コアで EDT-1 を行い，列方向における最も近い黒画素への距離を算出

Step 3. Mesh ネットワークを利用して，Step2 の出力結果を隣接コアに転送し行方向のデータを取得

Step 4. 行方向において EDT-2,EDT-3,EDT-4 を行い，最終的な結果を Shared Memory に転送し出力

Algorithm 2

Step 1. ホスト側から Shared Memory に転送した画像を列方向に 16 分割し，Local Memory に収まるサイズだけを各コアの Local Memory に転送

Step 2. 各コアで EDT-1 を行い，列方向における最も近い黒画素への距離を算出し，Shared Memory に転送

Step 3. 各コアの担当領域内の全画素において EDT-1 の処理を終えるまで，Step1，Step2 を反復

Step 4. Shared Memory に転送された EDT-1 の出力結果を行方向に 16 分割し，Local Memory に収まるサイズだけを各コアの Local Memory に転送

Step 5. 行方向において EDT-2,EDT-3,EDT-4 を行って最終的な結果を取得し，Shared Memory に転送

Step 6. 各コアの担当領域内の全画素において最終結果を出力するまで Step4，Step5 を反復

5. 実験結果

実験環境

- Desktop PC : Intel Core i5 - 4440
  —動作周波数 : 3.1 GHz
- Raspberry Pi 2 : ARM Cortex - A7
  —動作周波数 : 900MHz
- Parallella : Epiphany Ⅲ
  —動作周波数 : 600 MHz

画像サイズが 64×64 以下は Algorithm1，それ以上は Algorithm2 を使用する．表 1 では黒画素の個数を全画素の 20%として実験を行う．

表 1 は各プラットフォームの実行時間及び消費電力を示す．この表によると Algorithm1 を使用した場合は Core i5 に比べ約 2.02 倍，Algorithm2 を使用した場合は約 3.683 倍のエネルギー効率を達成している．結果として Parallella を用いることにより Core i5 より消費電力を抑え且つ高いエネルギー効率を達成した．

6. まとめ

本研究では，小型並列コンピューティングプラットフォームである Parallella を用いて画像処理アルゴリズムの 1 つであるユークリッド距離変換の並列化を行い，Parallella アーキテクチャの性能を評価した．エネルギー効率の比較では Core i5 と比較し，512×512 の画像サイズの時に約 3.683 倍のエネルギー効率を達成した．

参考文献

[1] Adapteva, http://www.adapteva.com/

[2] Duhu Man, Kenji Uda, Yasuaki Ito and Koji Nakano, Accelerating computation of Euclidean distance map using the GPU with Efficient memory access, International Journal of Parallel, Emergent and Distributed Systems, Vol. 28, No. 5, pp. 383406, 2013.