Cell Broadband Engine を用いた輪郭抽出処理の高速化

中村 哲平　荒井 大輔　上原 寛樹
千葉大学大学院工学研究科 人工システム科学専攻

X 線 CT などで撮影された断層画像から注目部位を立体的に認識することは困難である．そこで，複数枚の断層画像に対して輪郭抽出を行い，ボリュームレンダリングを用いて 3D 表示させる研究が行われている．しかし複数の画像に対して処理を行うため，汎用コンピュータでは膨大な演算時間がかかってしまう．そこで本研究では，ヘテロジニアス型マルチコア・プロセッサの Cell Broadband Engine（以後 Cell）搭載の PLAYSTATION3（以後 PS3）を用いて演算の高速化を図った．その結果，汎用コンピュータに対して SPE1 コアで約 1.7 倍，PS3 16 台で各 6 コアを用い 96 並列処理を行うことで約 30 倍の高速化を実現した．

1　研究背景

近年，医療現場での，画像診断装置の占める割合は大きくなってきている．特に画像診断装置の中で X 線 CT や核磁気共鳴画像法（MRI）といった装置は鮮明な断層画像を撮影することが可能であり，医療現場においてなくてならない存在である．これらのコンピュータ断層撮影（CT）を行っている画像診断装置で得られる撮影写真は，対象物を横からスライスしたような画像であるため，立体的な認識は観測者の経験や技術によるところが大きい．そこで，複数枚の断層写真に輪郭抽出処理を行い，また抽出された画像データを用いて 3D で表示させる技術が開発されている．しかし，現在の汎用コンピュータの逐次処理では演算時間が膨大なものになってしまう．

そこで，本研究では輪郭抽出処理である Snake 法を，並列処理が得意であり，演算用コアが 8 つある，Cell のクラスタ環境を構築することにより演算時間の短縮を図った．

2　輪郭抽出法原理

本章では，実際に用いた輪郭抽出法である Snake 法について述べる．最初に今回用いた手法のフローチャートを図 1 に示す．

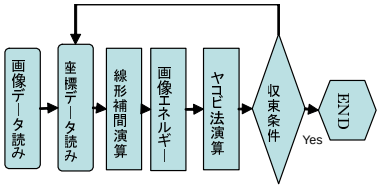

図 1: 輪郭抽出手法フローチャート

Snake 法とは輪郭抽出を直接行う手法として，M.Kass らにより Active Contour Model という名で提案された [1]．その手法は，

1. 初期値として閉曲線（以後 Snake 曲線）を与える．
2. 『輪郭線 = 画素値の勾配が大きい』という特徴を用いて，Snake 曲線の形状や重心間の距離といった評価関数と，画素値の勾配（以後 画像エネルギー）が釣り合うようにそれぞれパラメータを掛け，足し合わせる．
3. その評価関数（以後 Snake 関数）が最小値となる Snake 曲線の位置，形状を求めることによって，輪郭抽出を行う．

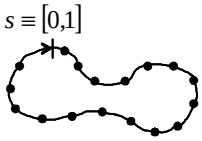

図 2: Snake 曲線略図

図 2 の様に，Snake 曲線が $\boldsymbol{u(s)} = \boldsymbol{(x(s), y(s))}, \boldsymbol{s} = \boldsymbol{[0, 1]}$ であった場合，Snake 関数はぞれぞれ，Snake 曲線の形状を評価する関数（内部エネルギー）と重心間との距離を評価する関数（外部エネルギー），画像エネルギーの 3 つからなる ((1) 式).

$$\begin{aligned} E_{total}\left(u(s)\right) &= E_{inter}\left(u(s)\right) + E_{exter}\left(u(s)\right) \\ &\quad + E_{image}\left(u(s)\right) \end{aligned} \tag{1}$$

また，最小値を求めるため，変分法を用いて積分を微分の形に直し，微分を差分の形に近似することによって，

(2)，(3) 式のような連立方程式となる．

$$\boldsymbol{AX} = \omega \boldsymbol{F_x} \tag{2}$$

$$\boldsymbol{AY} = \omega \boldsymbol{F_y} \tag{3}$$

それぞれ，$\boldsymbol{A}$ はパラメータを要素とした行列であり，$\boldsymbol{X}, \boldsymbol{Y}$ は Snake 曲線の $\boldsymbol{x}$ 座標，$\boldsymbol{y}$ 座標を，$\boldsymbol{F_x}, \boldsymbol{F_y}$ は画像エネルギーを，ω は画像エネルギーの重みを調整するパラメータを示している．

3 開発環境

今回 Cell の開発環境に SONY から発売されている，PS3 を用いて行った．その他，実行環境を表 1 にまとめる．

表 1: 実行環境

CPU	Pentium4 (3.2GHz)	Cell (3.2GHz)
OS	Linux (Fedora8)	Linux (Fedora9)
コンパイラ	g++ 4.1.2	g++ 4.3.0
オプション	-O3 -funroll-loops	-O3 -funroll-loops

4 Cell への実装

4.1 画像データの転送

256KB のローカルストレージ (LS) 上への 512 × 512 画素の画像データの搭載は難しい．また，転送時間の影響が大きくなるなどの問題がある．

そこで，今回は対象部位の輪郭抽出を行うために，初期座標の近傍の領域を予め定めて転送を行った．その際，SPE 上での SIMD 演算を効率的に行うために，単精度にキャストを行った状態で転送を行った．

4.2 SIMD 演算 [3]

線形補間演算部，ヤコビ法演算部共に，SIMD 演算を用いて，Snake 曲線上の隣り合った 4 点に対して，同時処理を行うことで，高速化を目指した．

4.2.1 線形補間演算部

今回，画素間隔よりも詳細な演算を行うために，4 点の線形補間を行った．その手法は図 3 にあるように，ある点の画素値 $\boldsymbol{f}$ を，近傍の 4 点の画素値 $\boldsymbol{f(i,j)}, \boldsymbol{f(i+1,j)}, \boldsymbol{f(i,j+1)}, \boldsymbol{f(i+1,j+1)}$ の 4 つの画素値を用いて求める手法である．実際の式は (4) 式のようになる．

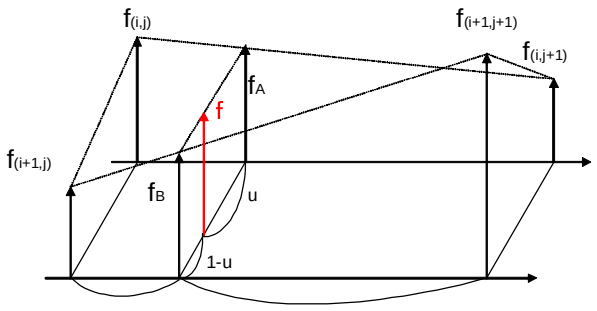

図 3: 線形補間

$$\begin{aligned}\boldsymbol{f} &= (1-u)f_A + uf_B \\ &= (1-u)\left\{(1-v)f_{(i,j)} + vf_{(i,j+1)}\right\} \\ &\quad +u\left\{(1-v)f_{(i+1,j)} + vf_{(i+1,j+1)}\right\} \\ &= (1-u)(1-v)f_{(i,j+1)} + v(1-u)f_{(i,j+1)} \\ &\quad +u(1-v)f_{(i+1,j)} + uvf_{(i+1,j+1)}\end{aligned} \tag{4}$$

4.2.2 Jacobi 法 [2]

$(\boldsymbol{x}, \boldsymbol{y})$ 座標の移動に伴う画像エネルギーの変化に対応するため，通常連立方程式の解法に用いる直接法ではなく，反復法である Jacobi 法を用いて演算を行った．

5 実行結果

5.1 SIMD 演算による計算の高速化

今回演算時間の測定の際に，同条件で演算を行うために Snake 曲線点数が 324 点であり，Jacobi 法の反復回数が 616 回で収束するデータを用いた．

SPE 内部での各演算部の SIMD 演算を用いる前後の実行時間を表 2 に示す．

Pentium4 での演算時間と，SPE 起動コストを含んだ Cell(1SPE) での処理時間を表 3 に示す．

表 2: SIMD 実装 (ms)

処理時間	SIMD 無し	SIMD 有り
線形補間	31.01	3.14
Jacobi 法	19.71	2.89
重心座標	5.80	1.37
DMA 転送	0.05	0.05
合計	56.57	7.45

表 3: Pentium 4 との比較 (ms)

slashbox	Pentium4	Cell
演算時間	18.76	10.66

Cell 内部の 1SPE を用いる際の各処理にかかった時間を表 4 に示す.

表 4: SPE 各実行時間 (ms)

slashbox	実行時間	内容
Image Open	0.19	SPE のプログラム イメージをオープン
Context Create	0.87	使用する SPE の確保
Program Load	0.92	SPE にプログラム イメージをロードする
Thread Create	0.14	複数 SPE を同時に 呼び出すための スレッド作成
Context Run	条件依存	プログラムの実行
Context Destroy	0.57	演算終了した SPE の解放
Image CLose	0.07	SPE のプログラム イメージをクローズ

5.2 SPE 並列化による輪郭抽出精度の向上 [4]

Snake 法の特徴として，パラメータの設定により，同じ初期条件であっても，収束しない場合や誤抽出が発生する (図 4)．また，これらのパラメータは経験的にしか求めることができないという問題がある．

そこで，Cell の搭載されている PS3 を 16 台用いてクラスタ環境を構築し，各 Cell でユーザが使用可能である 6 つ

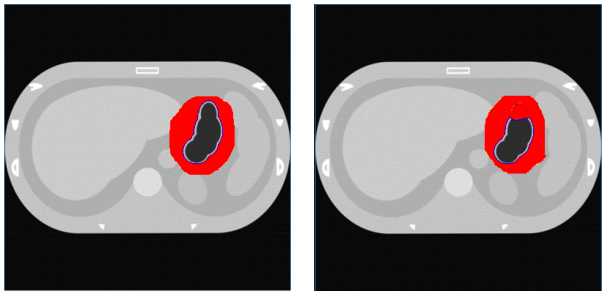

図 4: パラメータによる正抽出と誤抽出

の SPE を用いて，1 枚の画像データに対して，96 組 (PS3-16 台× 6SPE) のパラメータでの並列演算を行った．クラスタ環境構築の際には,MPI(Meggage Passing Interface) を用いて行った [5].

その 96 個の演算結果から，最も正確に抽出されているものを取り出すことで，抽出精度の向上を目指した．

1 枚の画像に対して 1,000 回の反復演算を,96 組のパラメータで行った場合の,1SPE に対する高速化比と演算時間を表 5 にまとめる．

表 5: PS3 クラスタ演算時間 (ms) と高速化比

SPE コア数	演算時間	高速化比
1	1,418.81	1.00
6 (PS3-1 台)	403.43	3.52
96 (PS3-16 台	46.53	30.49

6 まとめと今後の課題

SIMD 演算による Pentium4 に対する高速化比は 1.7 倍に留まった．また，1 台の PS3 での SPE 並列化による高速化比は約 3.5 倍となった．上記の結果 2 つは, 演算時間が数十 ms と短いため, 使用する SPE の個数が増えるごとにコンテキスト作成時間も増加するためと考えられる (表 6).

クラスタ環境での 96SPE での並列演算も MPI 通信時間（約 20ms）の影響で約 30 倍に留まった．

Snake 法の問題点である抽出制度の向上は，複数のパラメータで並列演算を行うことによって, 汎用コンピュータによる逐次演算よりも早く結果を得られるため，効果的であると思われる．しかし, 正確に抽出できたかどうかの判定は, 出力ファイルに対する目視で行っているため,

ノード数増加による判定時間の増加が考えられる．そのため, 機械的に判定できるようなシステムの構築を行う予定である．また、今回行ったのは人工的に作成されたファントムの医療画像であるため, より実用に近い実際の臨床データに対する処理も行う予定である．

表 6: SPE コンテキスト作成時間 (ms)

SPE 起動数	演算時間
1	0.87
2	2.11
3	3.39
4	4.25
5	5.38
6	6.68

参考文献

[1] MICHAEL KASS, ANDREW WITKIN, DEMETRI TERZOPOULOS. Snakes: Active Contour Models, International Jounal of Computer Vision, pp. 321-331, 1988.

[2] パソコンで数値計算. http://pc-physics.com/.

[3] FIXSTARS web page. PLAYSTATIONR3 Linux Information Site. http://www.fixstars.com/.

[4] 坂口嘉之, 美濃導彦, 池田克夫. Snake パラメータの設定についての検討. PRU90-21, pp. 43–49.

[5] MPICH2 インストール・設定. http://ameblo.jp/hamubane/entry-10165568392.html.