Vol.2019-HPC-172 No.10
2019/12/18

3 次元医用画像データの再構成処理の並列化

中島大地 [1]　大島聡史 [2]　五嶋優詞 [3]　横田達也 [3]　片桐孝洋 [2]　本谷秀堅 [3]
永井亨 [2]　岩本千佳 [4]　大内田研宙 [4]　橋爪誠 [5]

本発表では，異なる染色病理画像をもとに 3 次元データに再構築するための処理を，ハイブリッド MPI / OpenMP による並列化を行い実行時間の短縮を行った．東京大学設置の Reedbush-U スーパーコンピュータシステムを利用し，提案手法による性能評価を行った．性能評価の結果から，オリジナルのプログラム実行に対して，ピュア MPI 実行で 8.3 倍，ハイブリッド MPI/OpenMP 実行で 9.6 倍の高速化を達成できることが明らかになった．

1. はじめに

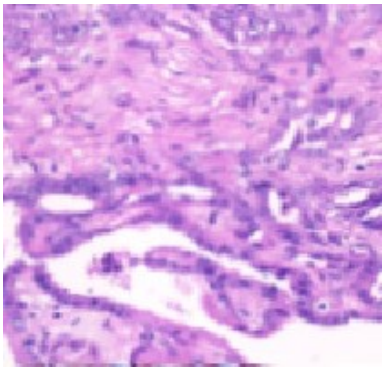
医学分野における医用画像の重要性は年々増している．特に医用画像の一種である高解像度な顕微鏡画像は組織病理学的画像として，疾患の分析や確定診断などの医用行為に広く利用されている[1, 2]. また基礎医療の分野においても，3 次元顕微鏡画像を用いた，癌細胞の構造理解が期待されている．

この様な 3 次元顕微鏡画像を得るための一般的な手法として，連続した 2 次元顕微鏡画像から 3 次元画像を再構成する手法があるが，解剖学的に不自然な変形をする可能性や，細胞の染色が同一であることを想定した方法であるなどの課題が存在した. そこで共著者の本谷らのグループは，以下に記述する新たな手法を採用した 3 次元画像の再構成プログラムの開発を行った[3].

3 次元画像の再構成は，顕微鏡画像による画像の入手過程において組織の位置がずれてしまうため，適切な位置合わせが必要となる．そこで，大きな病理画像の処理に対しメモリ効率が高い特徴を持つ，ランドマークに基づいた位置合わせを採用し，ランドマークの検出には，Normalized Cross Correlation（NCC）を用いたテンプレートマッチングを利用した．NCC を用いることで，異なる染色を伴う組織画像を組み合わせる場合においても，ランドマークの検出を可能とし，同時に折り目やしわなどのアーティファクトに起因する信頼性の低いランドマークを，自動的に無視することも可能とした．上記の提案手法を採用することで，異なる染色を伴う組織病理学的画像を用いたロバストな 3 次元画像の再構成が可能となる．

しかし開発したコードには，多数の容量の大きな顕微鏡画像を利用することで生じる反復計算が存在する．そのため，実行時間が膨大となってしまい，処理時間の短縮が喫緊の課題となっていた．そこで本研究では，この 3 次元画像の再構成プログラムに対し並列化技術を適用することで，1 から並列化を実装し性能向上を図る．利用する癌細胞の顕微鏡画像を図 1 に示す.

図 1. 癌細胞の顕微鏡画像



本プログラムでは標準ライブラリに OpenCV が提供されていること，および，将来 Graphics Processing Unit (GPU) への展開を考慮し，性能評価環境として東京大学設置のスーパーコンピュータ Reedbush-U を用いた．

本論文は以下の構成である．2 章では，3 次元画像の再構成のアルゴリズムについて説明する．3 章では，並列化およびコード最適化についての説明を行う．4 章では，Reedbush-U を用いた性能評価の結果を示す．最後に 5 章で本論文から得られた知見を述べる．

2. 3 次元画像の再構成のアルゴリズム

本研究で利用する 3 次元画像の再構成プログラムは，連続する複数の 2 次元画像より，NCC を用いたテンプレートマッチングによるランドマークの検出を行い，そこからランドマークに基づく位置合わせを実施する．そして位置合わせ結果から，得られた軌跡の最適化を行う．さらに，変形後の座標を導出して画像の変形を行うことで，異なる染色を伴う 2 次元入力画像を用いても，ロバストな 3 次元画像の再構成を可能とする．この章では，この 3 次元画像の再構成のアルゴリズムの詳しい解説を行う．

1 名古屋大学大学院情報学研究科
Graduate School of Informatics, Nagoya University
2 名古屋大学 情報基盤センター
Information Technology Center, Nagoya University
3 名古屋工業大学大学院 工学研究科
Graduate School of Engineering, Nagoya Institute of Technology,
4 九州大学　医学研究院　臨床・腫瘍外科
Department of Surgery and Oncology, Faculty of Medicine Sciences, Kyushu University
5 九州大学　先端医療イノベーションセンター
Center for Advanced Medical Innovation Kyushu University

2.1 画像の再構成

3 次元画像の再構成は，単一の組織から取得した連続する N 個の顕微鏡画像I_i（u）（i= 1,2, … N，ここで，$u = (u_1, u_2)^T$は元画像の座標を示す.）より行われ，初めに以下の 2 つのステップを実施する.

1. 剛体レジストレーションによって画像を大まかに調整し，新たな集合 $\tilde{I}_i(x) = I_i(\rho_i^{-1} \circ u)$（$\rho_i$は剛体変換，x =$\rho_i$（u））を作成する.
2. 連続する$\tilde{I}_i(x)$に対応するランドマークの点となる集合P_i^jの検出を行う．検出方法については，次節で解説を行う.

次にランドマークの点の集合が同じ解剖学的構造のいくつかの断面に沿って検出され，且つこれらの構造が空間的に滑らかで連続していると仮定する．その場合ランドマークを用いた最適な画像変形を行うためには，再構成した画像上において，ランドマークの変形先の点の集合が滑らかな曲線に沿うように，画像変形する必要がある.

画像変形のプロセスを図 2 に示し，解説を行う.

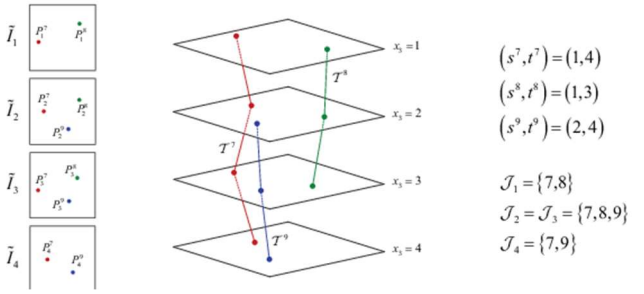


図 2. 画像変形のプロセス:(左図)4 つの入力画像$\tilde{I}_i(\cdot)$, (中央)ランドマークP_i^j及び軌跡T^j, (右図)各軌跡の s^j，t^j及び J_i（参考文献[3]の図 1 より転載）

$P^j = \{P_i^j | i \in [s^j, t^j]\}$は，N 個の画像の部分集合から検出された，ランドマークの j 番目の集合である． s^jとt^jは，P^jが通過する初めの画像と最後の画像を示す．また点の集合は，$1 \le s^j < t^j \le N$ の範囲において，利用可能な N 個の画像の一部分に沿って検出される場合も存在する.

ランドマークの集合P^jは， x_3がs^jからt^jを範囲に持つ，$\tilde{R}(x_1, x_2, x_3)$の多角形の軌跡$T^j = \overline{P_{s^j}^j P_{s^j+1}^j \dots P_{t^j}^j}$を形成する.

ここで，$\tilde{R}(x_1, x_2, x_3) = \tilde{I}_t(x_1, x_2)$（$x_3 = i$）は，$\tilde{I}_i(x)$を積み重ねることによって得られる再構成の開始を示す．組織画像より独立した非剛体変換から，これらの軌跡は，滑らかではないギザギザの形状となる．各ランドマークの最適化後の座標は，こうした各軌跡T^jを滑らかにすることによって導出する.

滑らかな軌跡は$T_Q^j = \overline{Q_{s^j}^j Q_{s^j+1}^j \dots Q_{t^j}^j}$，軌跡の交点は$\{Q_i^j | j \in J_i\}$で示す． J_iは平面 x3 = i を交わる軌跡の指数であり，$\tilde{I}_i(\cdot)$の二相写像Φ_iを定義するために利用する．画像$\tilde{I}_i(x)$は，P_i^jをQ_i^j（$j \in J_i$）に写像するように，取得された写像によって非剛性に変形され，3 次元画像$\tilde{R}(y_1, y_2, y_3) = \tilde{I}_t(y1, y2) = \tilde{I}_i(\phi^{-1} \circ y)$を再構築する．ここで，$y_3 = i$， $y = (y_1, y_2)^T$， $y_i^j = \phi_i \circ x_i^j$， x_i^j，および，y_i^jは，それぞれ画像$\tilde{I}_i(x)$のP_i^jおよびQ_i^jの座標である．新しい点Q_i^jの座標は，$t^j - s^j + 1$個からなる画像$\tilde{I}_t(i = s^j, \dots, t^j)$から導出される．点$Q_i^j$の座標の導出方法については，2.3 節で解説を行う．また変形前の軌跡T^jと，変形後の軌跡T_Q^jを示した図を図 3 に示す.

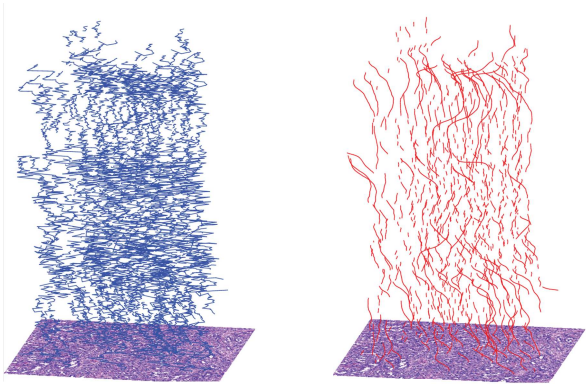


図 3. 軌跡の図:(左図)変形前の軌跡T^j(右図)変形後の平滑化した軌跡T_Q^j

こうした点Q_i^jの正確で安定した検出を行うために，coarse-to-fine の手法を採用している．テンプレートサイズ D は，すべての画像$\tilde{I}_t(i = 1,2, \dots, N)$が写像 φi によって変形された後，D ←γ D として縮小される（0 <γ <1）.

2.2 ランドマークの検出とアーティファクトの処理

NCC を用いたランドマークの検出について，解説を行う.

初めに点P_i^jの集合は次の式(1)で与えられる確率p_i^jによって，$\tilde{I}_t$（・）から繰り返しサンプリングされる.

$$p_i^j(x) = \begin{cases} \dfrac{||\nabla \tilde{I}_i(\cdot)||}{Z}, if(||\nabla \tilde{I}_i(\cdot)|| > T) \wedge (x \in \Pi_i^j) \\ 0, otherwize \end{cases} \dots (1)$$

ここで，Z は正規化係数，T は閾値を示し，指数 j は反復とともに増加する．j 番目の点のサンプリング領域は，点の均一な空間分布を保証し，オーバーサンプリングを回避するために，領域Π_i^jに制限する.

また，NCC から得られたテンプレートマッチングの信頼性に基づいて，後方テンプレートマッチングを適用することにより，マッチングの誤りを抑制する.

まず次の NCC 値が閾値θ_cより小さい場合は，ランドマークを不採用とする．そして$\hat{P}_{i+1}^j$に対応する$\hat{P}_i^j$を検出する

ため，後方テンプレートマッチングを適用する．P_i^jと$\hat{P}_i^j$の間の距離がしきい値θ_Dより大きい場合も，ランドマークP_{i+1}^jを不採用とする．

以上の方法より，折り目，しわ，ぼやけた部分など，組織画像に共通する多くのアーティファクトを自動的に処理することが可能となる．

2.3 Image Warping

点Q_i^jを導出するために行う，軌跡T_jを平滑化する方法について解説する．最適化後の座標y_i^jは，式(2)の様にトレードオフパラメータλから各軌跡の合計変動の2乗と2乗誤差を最小化することで導出することができる．

$$\left\{y_{s_j}^j, \dots, y_{t_j}^j\right\}$$

$$= \underset{\tilde{y}_{sj}^j, \dots, \tilde{y}_{tj}^j}{\operatorname{argmin}} \left(\sum_{i=s_i}^{t_i-1} \left\| \widetilde{y_i^J} - \widetilde{y_{i+1}^J} \right\|^2 + \lambda \left\| \widetilde{y_i^J} - x_i^j \right\|^2 \right) \dots (2)$$

式(2)の関数は凸型であり，分析的に計算できる独自の最小化機能を備えている．点y_i^jを求めることで，B-spline deformation 手法を用いて，二相写像ϕ_iを導出することができる．

2.4 プログラムの構成と実行時間内訳

3 次元画像の再構成プログラムは，以下の 5 つのコードに分けられている．

1. Edge
2. Trace_landmark
3. Optimize_trajection
4. Merge_center
5. NTM2

それぞれ，対応する役割は以下の通りである．

1. 初期テンプレートの生成(軌跡の 1 番目の点を生成)
2. テンプレートマッチングを利用し，軌跡T^jの座標P^jを取得する．
3. 座標の最適化を行い，点Q_i^jの座標を取得する．
4. 軌跡T^jの点P_i^jの座標とそれに対応する点Q_i^jの座標をまとめる．
5. 点P_i^jと点Q_i^jの座標から，画像の変形を行う．

これら 5 つのコードの実行時間について調査するため，全く並列化を実装されていなかったコード(オリジナルコード)に対して，Reedbush-U を用いた前評価を行った．Reedbush-U の計算機構成については，4 章の表 2 で示すためここでは省略する．実行結果を表 1 に示す．

表 1 に示されるように，実行時間は全体で 20696 秒（約 5.8 時間）であり，プログラムの実行時間の大部分を，Trace_landmark 及び NTM2 の 2 つのコードが占めている．2 つのコードの共通点として，連続する顕微鏡画像を読み込んでいることが挙げられる．読み込まれる顕微鏡画像は，1 枚当たりのデータ容量は大きく，本実験で用いられる画像の一つも 1.62GB となっている．

表 1. 3 次元画像の再構成プログラムの実行結果

	実行時間(s)	割合(%)
Edge	30	0
Trace_landmark	2784	13
Optimize_trajection	46	0
Merge_center	173	1
NTM2	17662	85
合計	20696	100

この様に，データ容量の大きな画像を複数回読み込むため，Trace_landmark 及び，NTM2 は実行時間が膨大となってしまう．そこで本研究では，この二つのコードについて並列化を実装することで，実行時間の短縮を試みる．

3. 並列化およびコード最適化

この章では，Trace_landmark 及び NTM2 に対し実装した並列化について，解説を行う．

3.1 Trace_landmark の並列化

Trace_landmark は，Edge より入手した初期テンプレートを利用して，連続した画像に対しテンプレートマッチングを行うコードである．

コード内では，画像の順番を示した i と軌跡の番号を示した id の 2 重ループが実装されている．またループ内はテンプレートマッチングを行うことで生じる，i について依存関係を持つデータ CX,CY が存在する．そこで本研究では，データ依存が存在しない id に関する並列化を行った．

並列化の実装には，MPI (Message Passing Interface) [4]と OpenMP [5]による並列化形態であるハイブリッド MPI/OpenMP 実行の実装を行った．ハイブリッド MPI/OpenMP 実行は，同コア数を用いたピュア MPI 実行と比較し，MPI プロセスによる通信時間を削減できることが知られている．

MPI 並列の実装は以下の手順で示す．

I. MPI の送受信をするためにメモリ確保
II. ループの長さを MPI プロセスごとに調整し，演算処理を実施
III. 送信バッファに計算したデータを格納
IV. MPI_ALLGATHER を用いて，通信するデータの送受信
V. 手順 IV．で取得した通信データを受信バッファに格納し利用する

OpenMP による並列化は，for ループに parallel 構文の適用を行う並列化を行った．以下の図 4，図 5 より，Trace_landmark の並列化を説明する．

```
//ステップⅡ.ループの長さを MPI プロセスごとに調整し, 演算処理を実施
for (i = 1; i < IMG_NUM; i++){
  ~演算処理①~
  #pragma omp parallel for private(…)
  for (id = mystart; id <= myend ; id++){
    ~演算処理②~
  }
~演算処理③~
}
```

mystart,myend は各プロセスのループ開始インデックス，終了インデックスを示している．

① 画像の読み込み(cv::imread(imagename)を利用),
画像のグレースケール変換
② パターンマッチング，CX,CY を導出する
③ 画像のコピー

図 4. Trace_landmark の並列化実装（１）

```
//ステップⅢ: 送信バッファに計算したデータを格納
#pragma omp parallel for private(…)
for (id = mystart; id <= myend; id++){
  int c=(id-mystart)*2*IMG_NUM;
  for (int i = 1; i < IMG_NUM+1 ; i++){
    xsend[c++]=CX[id][i];
    xsend[c++]=CY[id][i];
  }
}
//ステップⅣ:MPI_ALLGATHER を用いて，通信するデータの送受信
MPI_Allgather(xsend,tmp,MPI::INT,xrecv,tmp,MPI::INT,MPI_COMM_WORLD);
//ステップⅤ：手順 IV．で取得した通信データを受信バッファに格納し利用する
#pragma omp parallel for private(…)
for(int pr = 0; pr< numprocs * 2; pr = pr + 2 ){
  int cc=pr/2;
  int c=tmp*cc;
  for (id = order[pr]; id <= order[pr+1] ; id++){
    for (int i = 1; i < IMG_NUM+1 ; i++){
      CX[id][i] = xrecv[c++];
      CY[id][i] = xrecv[c++];
    }
  }
}
```

図 5. Trace_landmark の並列化実装（２）

図 4 では，演算処理①の場所に画像の読み込みがある．そのため，MPI プロセスごとに同一のファイルに対してアクセスが生じ，プロセス数が 2 以上になると，ファイルアクセス時間の増大を招く可能性がある．

図 5 に記されるステップⅢのループについては，ループ i，id 共にデータ依存がないため，どちらのループでも OpenMP の parallel for 構文の適用ができる．

3.2 NTM2 の並列化

NTM2 は，ランドマークP_i^jとそれに対応する点Q_i^jを用いて，連続した画像に対し画像変形を行うコードである．

コードは，最外郭ループに画像の順番を示した i のループの実装を持ち，連続した画像に関して，依存関係を持つデータを利用しないことから，本研究では i に関する MPI 並列の実装を行った．また並列部分処理後に得られる画像は，他のプロセスに影響しない．結果通信によるデータのやり取りを必要とせず，前節 3.1 の MPI 並列の実装手順のステップⅡの実装と同様に，各 MPI プロセスが個別にファイル処理するような実装を行った．実装例を図 6 に示す．

```
//ステップⅡ.ループの長さを MPI プロセスごとに調整し, 演算処理を実施
for (k = mystart ; k <= myend; k++){
        ret = fscanf(fp2,"%s¥n",imgname);
        int t = main_loop(k,imgname);
}
```

図 6 . NTM2 の並列化実装

また関数 main_loop 内において存在するループについては，OpenMP による並列化実装を行った．

4. 性能評価

4.1 問題設定

入力に用いた画像の枚数は，293 枚に設定し実験を行った．例外で，項 4.5.1 については入力画像を 10 枚に設定した

4.2 実験環境

本実験で使用した Reedbush-U の計算機構成を，表 2 に示す．

4.3 実験設定

Trace_landmark 及び NTM2 のハイブリッド MPI/OpenMP 実行の性能評価を行うため，評価方法としてスレッド数比較，ピュア MPI における MPI プロセス数比較，ハイブリッド MPI/OpenMP 実行の比較の 3 種類の方法を用いた．

表 2. Reedbush-U の計算機構成

ハードウェア構成	
全体構成	
総ノード数	**470**
総理論演算	**508.03TFLOPS**
CPU	
プロセッサ	**Intel Xeon E5-2695v4 (Broadwell-EP)**
動作周波数	**2.1 GHz (Turbo boost 時最大 3.3 GHz)**
プロセッサ数(コア数)/ノード	**2 (36)**
演算性能/ノード	**1209.6 GFLOPS**
メモリ帯域幅	**153.6 GB/sec**
ソフトウェア構成	
開発環境	**C++/Python**
MPI 通信ライブラリ	**Intel MPI**
ライブラリ	**OpenCV3.2.0(CUDA 使用無し)**
コンパイラ	**Intel コンパイラ**

4.4 実験結果（Trace_landmark の並列化）

4.4.1 スレッド数比較

1 ノードの環境下で，スレッド数を 1 -36 に変化させた場合の性能評価を行った．スレッド数 1 の実行時間を基準とした，スレッド数増加時の速度向上率を図 7 に示す．

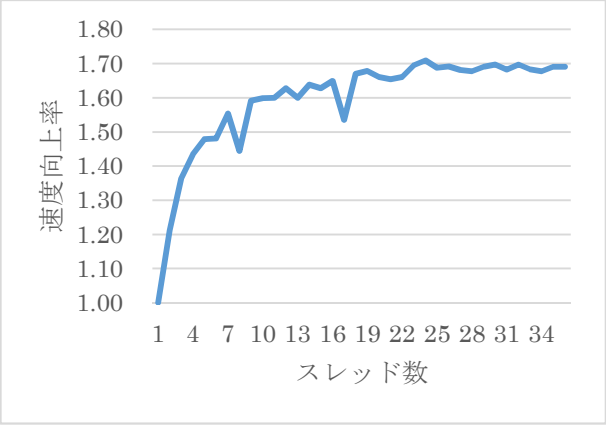


図 7. スレッド数の増加による速度向上率(Trace_landmark)

図 7 より，スレッド数増加に対して速度向上が見られた．ただし一定以上のスレッド数では，比例した性能向上は見られなかった．またスレッド数 24 の時，最大で 1.71 倍の高速化が確認された．

4.4.2 ピュア MPI 実行(MPI プロセス数比較)

ピュア MPI 環境下で，MPI プロセス数(ノード数)を 1- 18 に変化させた場合の性能評価を行った．MPI プロセス数 1 の実行を基準とした，MPI プロセス数増加の速度向上率を図 8 に示す．

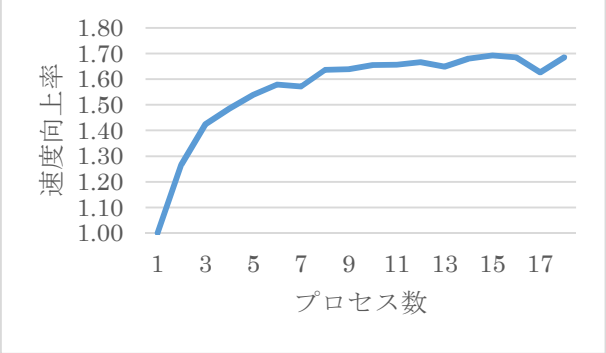


図 8. ピュア MPI 実行時における
MPI プロセス数の増加による速度向上率(NTM2)

図 8 より，MPI プロセス数増加に対して速度向上が見られた．また MPI プロセス数 15 の時，最大で 1.69 倍の高速化が確認された．

4.4.3 ハイブリッド MPI/OpenMP 実行(MPI プロセス数比較)

Reedbush-U におけるノード内の最大スレッド数 36 に固定した際のハイブリッド MPI/OpenMP 実行における，MPI プロセス数(ノード数)を 1-18 に変化した場合の性能評価を行った．MPI プロセス数 1 の実行時間を基準に，MPI プロセス数増加時の速度向上率を図 9 に示す．

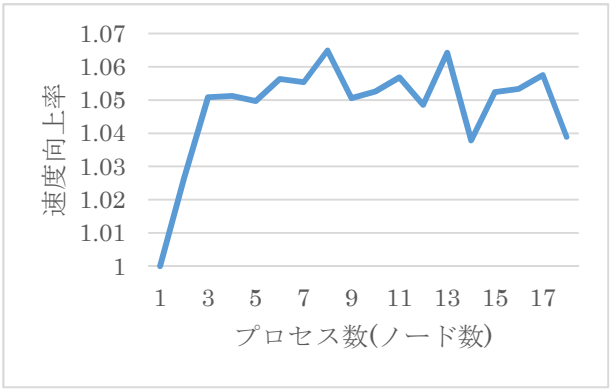


図 9. ハイブリッド MPI 実行時における
MPI プロセス数の増加による速度向上率(Trace_landmark)

図 9 より，MPI プロセス数増加に対して速度向上が見られたが．2 プロセス以上において大きな性能向上は見られなかった．また MPI プロセス数 8 の時，最大で 1.06 倍の高速化が確認された．

4.5 実験結果（NTM2 の並列化）

4.5.1 スレッド数比較

1 ノードの環境下で，スレッド数を 1 -36 に変化させた場合の性能評価を行った．スレッド数 1 の実行時間を基準

とした，スレッド数増加時の速度向上率を図 10 に示す．

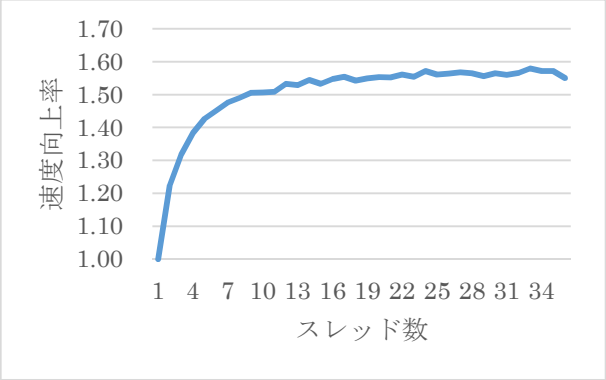


図 10. スレッド数の増加による速度向上率(NTM2)

図 10 より，スレッド数増加に対して速度向上が見られた．ただし一定以上のスレッド数では，比例した性能向上は見られなかった．またスレッド数 33 の時，最大で 1.58 倍の高速化が確認された．

4.5.2 ピュア MPI 実行(MPI プロセス数比較)

ピュア MPI 環境下で，MPI プロセス数を 1-36 に変化させた場合の性能評価を行った．プロセス数 1 の実行を基準とした，プロセス数増加の速度向上率を図 11 に示す．

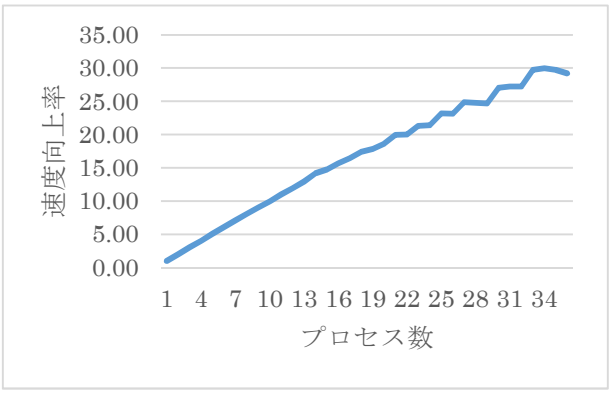


図 11. ピュア MPI 実行時における
MPI プロセス数の増加による速度向上率(NTM2)

図 11 より，MPI プロセス数増加に対して速度向上が見られた．また MPI プロセス数 34 の時，最大で 30.0 倍の高速化が確認された．

4.5.3 ハイブリッド MPI/OpenMP 実行(MPI プロセス数比較)

Reedbush-U におけるノード内の最大スレッド数 36 に固定した際のハイブリッド MPI/OpenMP 実行における，MPI プロセス数(ノード数)を 1-36 に変化した場合の性能評価を行った．MPI プロセス数 1 の実行時間を基準に，MPI プロセス数増加時の速度向上率を図 12 に示す．

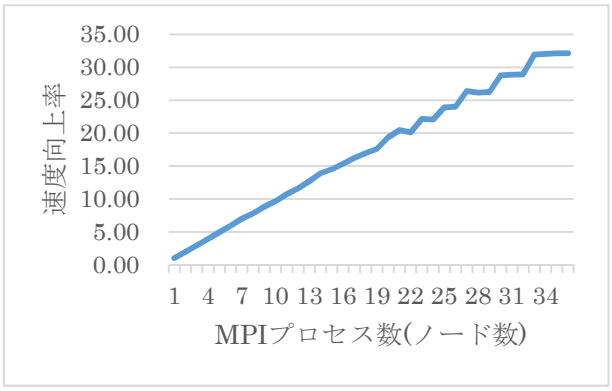


図 12. ハイブリッド MPI 実行時における
プロセス数の増加による速度向上率(NTM2)

図 12 より，MPI プロセス数増加に対して速度向上が見られた．また MPI プロセス数 35 の時，最大で 32.1 倍の高速化が確認された．

4.6 考察

まず Trace_landmark の並列化の結果についてまとめる．オリジナルコードでは実行時間が 46 分であったが，ピュア MPI 実行(MPI プロセス数 15)を用いることで，実行時間 27 分(1.69 倍の高速化)，ハイブリッド MPI/OpenMP (8 ノード，8MPI プロセス，36 スレッド)を用いることで，実行時間 26 分(1.79 倍)となる高速化が実現した．また OpenMP 並列化及び MPI 並列化は，それぞれ一定以上のコア数(OpenMP 並列化は 18 コア，MPI 並列化は 9 コア)の利用について，効果的な高性能化は望めなかった．この理由の一つとして，Trace_landmark は MPI 並列化に伴い，画像読み込みを行う cv::imread を，プロセスそれぞれで実行している(3.3 節図 4 の演算処理①)ことが考えられる．各プロセスにおいて同一画像に対し，同時に読み込み処理を行うことは，ハードウェアの観点で適する実装とは言えないためである．そこで，たとえば，ランク 0 のプロセスのみがファイルアクセスを行い読み込み後，そのデータをその他のランクに放送するような実装が考えられる．また，ファイルアクセスの方法自体を，MPI-IO のようなプロセス並列向きの I/O 方式に変更することがさらに望ましい．

次に NTM2 の並列化の結果についてまとめる．オリジナルコードでは実行時間 4.9 時間であったが，ピュア MPI 実行(MPI プロセス数 34)を用いることで，実行時間 9.8 分(30.0 倍の高速化)，ハイブリッド MPI/OpenMP (35 ノード，35 MPI プロセス，36 スレッド)を用いることで，実行時間 5.6 分(52.2 倍)となる高速化が実現した．また MPI 並列化は，通信を必要としないことから性能向上が著しく，OpenMP 並列化は 8 スレッドまでの利用が効果的であった．ハイブリッド MPI/OpenMP の利用は，再構成に必要な画像が少なく，MPI プロセス増加の影響が少ない場合に，スレッド 8 以下の利用が好ましいといえる．

5. おわりに

本研究では，医用画像処理の一種である 3 次元画像の再構成プログラムに，ハイブリッド MPI/OpenMP の実装を行い有効性の評価を行った．

一定以上のコアの利用について，効果的な性能向上が発揮できないことに対し，コード最適化を実装することで，さらなる速度向上を図る必要がある．

3 次元画像の再構成プログラムは，オリジナルコードと比較して，並列化後のプログラムではピュア MPI 実行時 8.3 倍，ハイブリッド MPI 実行時 9.6 倍の性能向上を達成した．その結果，オリジナルコードの実行時間が 5.8 時間だった処理が，36 分に短縮ができると予想される．このことは，高性能計算技術を医療画像処理分野に展開することで，新たなイノベーションが起こることを示した一例といえる．

謝辞 本研究の一部は JSPS 科研費 JP18H03262 の助成を受けたものです．

参考文献

[1]Pichat, J. et al.. A survey of methods for 3D histology reconstruction. Medical Image Analysis. 2018, vol46, p73-105.
[2]Gurcan, MN. et al.. Histopathological image analysis: A review. IEEE Rev Biomed Eng. 2009, vol2, p147-171.
[3]Mauricio,K. et al.. Robust 3D image reconstruction of pancreatic cancer tumors fromhistopathological images with different stains and its quantitative performance evaluation, IJCARS, 2019.
[4]“MPI forum:” .https://www.mpi-forum.org/,(参照 2019-11-13).
[5]“OpenMP forum:”.http://forum.openmp.org/forum/,(参照 2019-11-13).