2005－HPC－104（5）
2005／10／7

汎用グラフィクスハードウェアを用いた 2次元／3次元剛体位置合わせの高速化

五味田　遵[†]　合田圭吾[††]　川崎康博[††]
伊野文彦[††]　萩原兼一[††]

画像の位置合わせとは，撮影方法・時刻の異なる画像の組に対し，物体の位置を対応付ける技術である．この技術は手術支援の分野において重要である．しかし，計算量が多いため CPU を用いた実装では位置合わせに十数分を要し，時間制約の強い手術支援の用途に対しては更なる高速化が必要である．一方，PC 用グラフィクスカードが搭載する GPU（Graphics Processing Unit）は近年著しく性能を向上している．そこで本研究は，GPU を位置合わせに用いることで，その実行時間の短縮を目指す．本稿では，2 次元／3 次元剛体位置合わせを構成する主要な 3 つの処理（ボリュームレンダリング，近傍フィルタ，集約演算）を GPU 上に実装し，その性能を測定することで GPU を用いた位置合わせの実用性を検証する．評価実験の結果，提案手法により 3〜5 秒程度の実行時間で位置合わせを行えた．このことから，位置合わせに GPU を用いることで手術支援の要求する時間制約を満たすことを確認した．

Accelerating 2-D/3-D Rigid Registration Using General-Purpose Graphics Hardware

Jun Gomita,[†] Keigo Goda,[††] Yasuhiro Kawasaki,[††]
Fumihiko Ino[††] and Kenichi Hagihara[††]

Image registration is a technique for finding point correspondences between two different images taken at different times and/or in different modalities. This technique plays an important role in computer-aided surgery. However, CPU implementations take more than 10 minutes to complete a registration task due to a large amount of computation. Therefore, some acceleration techniques are required to use this technique for surgical assistances, where response time is strictly limited in a short time. One acceleration technique is to use GPUs (graphics processing units) equipped on PC graphics cards, which are rapidly increasing performance. The objective of our work is to reduce registration time by using GPUs. This paper presents our GPU method that accelerates three key procedures of 2-D/3-D rigid registration: volume rendering, neighbor filtering, and reduction operation. We also investigate the usability of our method from the viewpoint of registration time. The experimental results show that our method completes a registration task within 5 seconds, and thus we find that our GPU method is fast enough to use it for surgical assistances.

1. はじめに

2 次元／3 次元剛体位置合わせ[1)] とは，異なる条件下で同一の物体を撮影した 2 次元画像 I_F および 3 次元画像 V に対し，I_F の座標系に V の座標系を対応付ける，すなわち I_F が V の投影であると見なし，その投影面に対する V の位置姿勢を求める技術である．この技術は，画像処理により外科手術を支援する際に基礎的な役割を果たし，重要である．しかし，位置合わせのための計算量が多いため，短い応答時間を必要とする手術支援においては，その高速化が必要である．

このような時間制約の強い用途に対し，PC クラスタを用いた並列計算によって位置合わせを高速化する手法[2)] がある．この手法では，CPU で逐次実行した場合に約 17 分を要する位置合わせを，プロセッサ 128 個の並列計算により 34 秒に短縮している．しかし，実際の医療現場で用いる場合，PC クラスタの導入や維持管理にかかるコストなどが課題となる．

† 大阪大学基礎工学部情報科学科
Department of Informatics and Mathematical Science, School of Engineering Science, Osaka University
†† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

一方，3 次元グラフィクスの高速処理を目的とする GPU（Graphics Processing Unit）[3] は，近年著しい性能向上を遂げており，単精度であるものの，CPU に対して 6 倍以上の浮動小数点演算性能（FLOPS）を達成している．更に，プログラム可能な GPU が普及するにつれ，従来の描画用途のみならず，汎用計算への応用が注目されている．

そこで本研究では，CPU に比べて高速であり，また PC クラスタに比べて導入の容易な GPU を用い，位置合わせの高速化を目指す．本稿では，2 次元／ 3 次元剛体位置合わせを構成する主要な 3 つの処理（ボリュームレンダリングによるデジタル再構成 X 線画像（DRR：Digitally Reconstructed Radiograph）生成，近傍フィルタによる微分画像生成，正規化相互相関係数（NCC：Normalized Cross Correlation）評価のための集約演算）を GPU 上に実装し，これらによる位置合わせの性能を測定することにより，その実用性を検証する．

2. 汎用プロセッサとしての GPU

GPU は本来，描画の高速化を目的とするプロセッサであり，一般にパイプライン構造を持つ（**図 1**）．このパイプラインは，プログラム可能な 2 つの演算器で主に構成されており，それらは VP（Vertex Processor）および FP（Fragment Processor）と呼ばれる．これらはラスタライザと呼ばれるユニットを挟んで連結される．

VP および FP は，それぞれ MIMD 型および SIMD 型の演算器であり，データを並列処理できる．演算対象として IEEE754 準拠[4] の 32 ビット浮動小数点数を扱え，ベクトル演算により 4 個のスカラ値を 1 度に処理できる．これらの演算器が参照できるビデオメモリはメインメモリとは独立していて，より高速である．しかし，その容量は，現在の GPU で多くとも 512MB であり，メインメモリよりも少ない．

GPU を描画処理以外の用途に用いる場合，処理対象のデータをテクスチャとして保持し，これを参照しながらビデオメモリへの描画を繰り返すという方法をとる．この際，VP と比較して FP の方が一般に高性能であるため，FP のみを用いる実装が多い．なお，VP はパイプラインの上流に位置するため処理結果をビデオメモリに直接出力できないが，下流の FP は結果をビデオメモリへ直接出力できるという相違点もある．

FP は SIMD 型の演算器であるため，FP 上で動作するアルゴリズムはテクスチャ上の各画素を互いに独立に処理できる（データ並列性を持つ）必要がある．

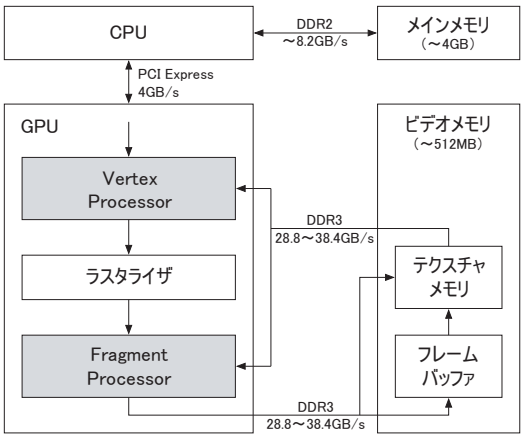

図 1 GPU アーキテクチャ

さもなければ，アーキテクチャの特性を活かせないため，十分な性能を得られない．

描画処理では GPU の処理結果をビデオメモリ上のフレームバッファ（画面）に出力すればよいが，非描画処理では処理結果の取り出しが必要である．そのため，CPU から GPU へのデータ転送に加え，GPU から CPU へのデータ転送（リードバック）が必要となる．このデータ転送は AGP もしくは PCI Express といったバスを経由するため，頻繁な転送は性能を低下させる．更に，現在の GPU は転送の間パイプライン処理を停止させることも，性能を低下させる要因の 1 つである．

3. 2-D/3-D 位置合わせの概要

2 次元／ 3 次元位置合わせ[1] の手法には，特徴点の抽出に基づくもの[5),6)] やボクセル値の比較に基づくもの[1),7),8)] などがある．後者は位置合わせ問題を 3 次元データの描画問題に帰着する解法であり，GPU の高速化対象と合う．ゆえに，本研究では後者を用いる．

この手法は，位置合わせ問題を最適化問題に帰着する．具体的には，3 次元画像 V の位置姿勢 P に関するコスト関数 C を最適化することにより，位置合わせを実現する（**図 2**）．ここで，コスト関数 C としては，V の投影である DRR と 2 次元画像 I_F との類似度で与える．この類似度を最大化する P を決定する．図中のステップサイズ D は P の評価 1 回当たりの移動量を表し，P を階層的に最適化するために用いる．D が最下層に至るまで P の移動および評価を繰り返し，最下層で得た結果を解とする．

類似度の指標には勾配相関係数（GC：Gradient Correlation）を用いる．GC は比較画像の画素値補正を

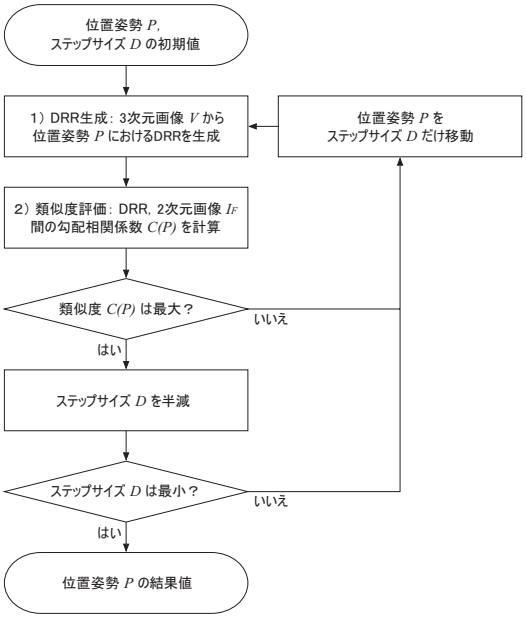

図 2　2 次元/3 次元位置合わせのフロー図

必要とせず，また，補正を必要としない方法の中で最も高い頑健性を持つことが知られている[8]．2 枚の画像 A，B 間の GC の値 $G(A,B)$ は次式で与える．

$$G(A,B)=\frac{1}{2}\left[N\left(\frac{\partial A}{\partial x},\frac{\partial B}{\partial x}\right)+N\left(\frac{\partial A}{\partial y},\frac{\partial B}{\partial y}\right)\right] \tag{1}$$

なお，関数 N は NCC である．また，$\partial A/\partial x$，$\partial B/\partial x$，$\partial A/\partial y$ および $\partial B/\partial y$ は画像 A，B それぞれの水平方向および垂直方向に対する微分画像である．

2 枚の画像 A，B 間の NCC の値 $N(A,B)$ は次式で与える．

$$N(A,B)=\frac{S_{AB}-S_AS_B/n}{\sqrt{S_{A^2}-(S_A)^2/n}\sqrt{S_{B^2}-(S_B)^2/n}} \tag{2}$$

なお，n は画像の画素数である．また，S_A，S_{A^2}，S_B および S_{B^2} は画像 A，B それぞれの画素値の和および 2 乗和であり，S_{AB} は画像 A，B の画素値の積の和である．

水平方向および垂直方向の各微分画像の生成にはガウシアン 1 次微分フィルタを用いる．このフィルタは画像のノイズを軽減する特性を持ち，位置合わせの頑健性を向上する．

画像 I に対する，ガウシアン 1 次微分フィルタによる水平方向微分画像 $\partial I/\partial x$ および垂直方向微分画像 $\partial I/\partial y$ は次式で与える．

$$\frac{\partial I}{\partial x}(x,y)=\sum_{i,j}\frac{-i}{2\pi\sigma^4}e^{-\frac{i^2+j^2}{2\sigma^2}}I(x+i,y+j) \tag{3}$$

$$\frac{\partial I}{\partial y}(x,y)=\sum_{i,j}\frac{-j}{2\pi\sigma^4}e^{-\frac{i^2+j^2}{2\sigma^2}}I(x+i,y+j) \tag{4}$$

なお，フィルタ半径を R とすると $(-R\le i,j\le R)$ である．また，σ はガウス関数の標準偏差である．

4. GPU を用いた 2-D/3-D 位置合わせ

本研究における GPU への実装の指針は，次の 2 点である．第 1 に，CPU 上の逐次実行で性能ボトルネックとなる部分を GPU へ移動すること．第 2 に，性能ボトルネックを生じ得る CPU–GPU 間の転送量を削減することである．

前者に対しては，GPU で DRR 生成および微分画像生成を行うことで，これらを高速化する手法を提案する．2 次元／3 次元位置合わせを CPU 上で逐次実行した場合，その実行時間のほぼすべて（9 割強）が DRR 生成および，類似度評価における微分画像生成に占められる．このことから，これら 2 つの処理を GPU を用いて高速化できれば，位置合わせ実行時間の大きな短縮が見込める．

後者に対しては，GPU で NCC 計算の前処理を行うことで，GPU からの転送量を削減する手法を提案する．NCC 計算はデータ並列性を持たないため，GPU での効率的な実現は難しい．しかし，微分画像生成を GPU で行う場合，NCC 計算も GPU 上に実装しなければ性能は低下する．その理由は，CPU–GPU 間のデータ転送が性能ボトルネックになるためである．NCC 計算を CPU で行う場合，類似度評価のたびに GPU から CPU へ微分画像を転送する必要がある．この際の転送量は画像サイズ相当である．一方，GPU で NCC 計算のための集約演算を行えば，転送量は 5 つの値に削減できる．

以上より，本研究における GPU を用いた位置合わせの設計概要は**図 3** のとおりである．

4.1 DRR の生成

本手法は，3 次元テクスチャを用いた TBVR（Texture-Based Volume Rendering）[9] により DRR を生成することで，処理時間を短縮する．

GPU を用いてボリュームレンダリングを行う手法には，Ray-Casting および TBVR の 2 つがある．これら 2 つを比較すると，一般的に，Ray-Casting は生成される投影像の精度に優れるが処理時間が長い．一

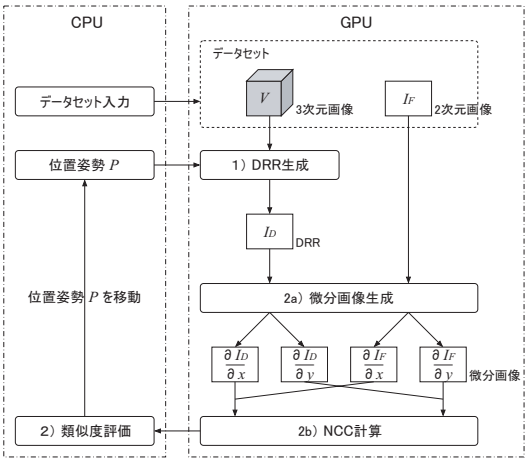

図 3　GPU を用いた 2-D/3-D 位置合わせ

方，TBVR は精度でやや劣るが処理時間が短い．

TBVR はさらに 2 次元テクスチャを用いる手法と 3 次元テクスチャを用いる手法の 2 つがある．ただし，前者は生成される投影像の精度で劣り，かつメモリ消費量が大きい．

3 次元テクスチャとは，通常のテクスチャが 2 次元の平面画像であるのに対し，3 次元のボリュームデータをテクスチャとして使用するものである．通常の 2 次元テクスチャは頂点データの表現する物体の表面に平面画像を貼り付ける機能しか持たないが，3 次元テクスチャはボリューム内の任意の断面を物体の表面にマップできる．視線方向に垂直な断面画像を投影面から遠いものから順に描画することで，投影像を得る．

4.2　微分画像の生成

本手法は，2 次元フィルタを 2 つのフィルタに分解することで計算量を削減し，また，2 枚の画像に対する処理をベクトル化することで全体のスループットを向上する．

ガウシアン 1 次微分フィルタは近傍参照型のフィルタであり，データ並列性を持つ．このため，FP 上での実装が容易であり，かつ FP による並列計算により高い性能を達成できる．

ガウシアン 1 次微分フィルタを FP 上で実装する場合，単純な方法は式 (3) および式 (4) をフラグメントプログラムとして実装する方法である．しかし，この場合，計算量はフィルタ半径 R の 2 乗に従い，実際の位置合わせでは入力画像のノイズを軽減させる目的で R をある程度大きくとる場合があるため，望ましくない．そこで 2 次元フィルタを，水平方向 1 次元フィルタおよび垂直方向 1 次元フィルタの 2 つのフィルタに分解し，計算量を削減する手法を用いる．また，この方法は，隣接する画素同士で重複して行われていた入力画像の参照を含む計算を省くため，テクスチャからの画素値の取得回数，すなわちビデオメモリに対するアクセス回数を削減する効果もある．

ここで，水平方向微分画像，垂直方向微分画像各々に対する垂直方向 1 次元フィルタおよび水平方向 1 次元フィルタの参照する入力画像の画素に着目すると，これらは一致している．これらの計算はベクトル化可能である．これら 2 つのフィルタをベクトル化してまとめると次のようになる．

$$\boldsymbol{P}_{xy1} = \sum_j \left[\begin{pmatrix} e^{-j^2/2\sigma^2} \\ \dfrac{-j}{2\pi\sigma^4} e^{-j^2/2\sigma^2} \end{pmatrix} * \boldsymbol{p}(x, y+j) \right] \tag{5}$$

$$\boldsymbol{P}_{xy2} = \sum_i \left[\begin{pmatrix} \dfrac{-i}{2\pi\sigma^4} e^{-i^2/2\sigma^2} \\ e^{-i^2/2\sigma^2} \end{pmatrix} * \boldsymbol{p}(x+i, y) \right] \tag{6}$$

なお，$*$ はベクトル要素ごとの積を表し，$\boldsymbol{P}_{xy1}$，$\boldsymbol{P}_{xy2}$ および $\boldsymbol{p}$ はベクトル値である．

微分画像生成において扱われる 2 次元画像 I_F および DRR はいずれも濃淡画像であり，各画素はスカラ値で表現できる．

4.3　NCC の計算

本手法は，ベクトル演算を用いて複数の集約演算を一括して行うことで演算回数を削減し，また，静的な 2 次元画像 I_F に対する処理を 1 回に省略することで計算量を削減する．

NCC 計算は，画素値の総和や 2 乗和などを算出する集約演算に帰着できる．集約演算は FP を用いた手法[3] が既に知られている．

FP 上での集約演算は並列総和計算と呼ばれる手法で実現される．並列総和計算とは，集約演算を部分和計算の集合体として構成し，個々の部分和計算同士を並列に実行する手法である．この手法はデータ並列性を備えており，かつ各段階の演算が SIMD 処理可能である．このため，FP 上への実装が可能となる．

NCC の値を求めるためには，先述のように 5 つの総和値を求める必要がある．しかし，FP はベクトル演算が可能であるので，集約演算を個数分である 5 回行う必要はない．入力画像の画素値（4 要素ベクトル）の各要素に異なる値を入れておき，集約演算を行うと，ベクトル演算により，部分和画像の画素値の各要素は要素ごとの部分和となる．このことから，2 回の集約

演算で必要な 5 つの総和値を求めることができる．このように演算回数を削減することは，リードバック回数の削減と処理時間の短縮を同時に達成する．

位置合わせにおいては，2 次元画像 I_F および DRRI_D の微分画像間の NCC の値を求めることになる．ここで，一方の I_F は処理全体を通じて静的であり，その画素値の和および 2 乗和は一定である．したがって，I_F に対しては初めの 1 回のみ結果を得れば，以降は I_D に対してのみ集約演算を行えば NCC の値を算出できる．

5. 評 価 実 験

GPU を用いた 2 次元／3 次元剛体位置合わせの実用性を速度面から検証するため，前章の通り実装を行った 2 次元／3 次元位置合わせに対し，その実行時間を GPU および CPU を用いて測定した（**表 2** および**表 3**）．

実験に用いた PC は，GPU として NVIDIA Quadro FX 3400，CPU として 2.8GHz 駆動の Pentium 4 を装備している．データには椎骨（**表 1** および**図 4**）を用いた．

なお，今回の実験では正解の位置姿勢を特定するため，あらかじめ特定の位置姿勢における DRR を生成し，これを 2 次元画像 I_F として用いた．また，ステップサイズの初期値は 2.0（voxel あるいは度），終了条件である最小値は 0.1 とする．

実験の結果，GPU を用いることで，2 次元／3 次元剛体位置合わせを 3〜5 秒程度で実行できた（表 3）．このことから，CPU で逐次実行した場合に十数分を要する位置合わせは，GPU を用いることで，手術支援の要求する時間制約を満たすと考える．

また，その実行時間の内訳（表 2）を見ると，CPU の逐次実行でその大部分を占めていた DRR 生成および微分画像生成の処理時間を削減できている．

続いて，GPU から CPU へ微分画像を転送して NCC 値を計算する場合（手法 A）に比べ，本研究で用いた GPU で総和を求めておく手法（手法 B）がどの程度の処理時間を短縮する効果を持つのかを確認するため，GPU で生成した微分画像を CPU へ転送する時間を測定した．CPU で微分画像から NCC 値を求める処理時間は測定の結果 9 ms 程度であり，この時間に測定した転送時間を加算することで，手法 A の処理時間を算出できる．

測定の結果，GPU から CPU へ 4 枚の微分画像を転送する時間は 6.9 ms である．すなわち，手法 A の処理時間は 16 ms，手法 B では 11 ms であるから，後者が有効であることがわかる．

表 4 および**表 5** に，本研究の実装と，PC クラスタ（64CPU）に実装した位置合わせとの比較を示す．ここで比較に用いた PC クラスタ版位置合わせは本研究の実装とアルゴリズムが異なるため，厳密な比較はできないが，GPU を用いた位置合わせの処理時間にお

表 1 データセット

	椎骨
3 次元画像	512 × 512 × 204 voxel
ファイルサイズ	102 MB
ROI	300 × 300 × 48 voxel
2 次元画像	300 × 200 pixel
ファイルサイズ	17 KB

表 2 実行時間の内訳（初期残差：8 mm）

内訳	評価 1 回当たり（μs）	評価回数	合計（ms）
DRR 生成	247		56
微分画像生成	241		54
NCC 計算	11385		2584
水平方向微分画像に対して	10018	227	2274
垂直方向微分画像に対して	1367		310
全体	—	—	2758

表 3 初期残差ごとの実行時間

初期残差（mm）	評価回数	実行時間（s）
2〜 4	296	3.7
4〜 6	292	3.6
6〜 8	336	4.2
8〜10	377	4.7
10〜12	398	5.0
12〜14	447	5.7
14〜16	402	5.0
16〜18	434	5.6
18〜20	419	5.5
20〜22	405	5.3

表 4 評価 1 回当たり処理時間（ms）比較

内訳	GPU 版	PC クラスタ版
DRR 生成	0.2	10.7
微分画像生成	0.2	3.8
NCC 計算	11.4	83.6

表 5 位置合わせ成功率（結果残差 0.5 mm 以下，10 回中）比較

初期残差（mm）	GPU 版	PC クラスタ版
2〜 4	1	2
4〜 6	1	0
6〜 8	0	0
8〜10	0	1
10〜12	0	0
12〜14	1	0

(a) 位置合わせ前　(b) 位置合わせ後　(c) 正解（2 次元画像 I_F）

図 4　位置合わせ前後および正解の位置姿勢における DRR （初期残差：8 mm；結果残差：0 mm）

ける優位は確認できる（表 4）．また，表 5 より，位置合わせ精度について大きな劣りは見られない（ただし，この PC クラスタ版は単方向投影によるため精度はよくない）．しかし，GPU 版と PC クラスタ版で得られる結果が異なることも確認できる．この原因が GPU の計算精度によるものか，アルゴリズムの違いによるものなのかは確認できていない．

6. おわりに

本稿では，GPU を用いて 2 次元／3 次元剛体位置合わせを高速化する手法を提案した．提案手法は，実行時間を短縮するため，CPU 上の逐次実行でその実行時間の大半を占める DRR 生成および微分画像生成に GPU を用いて高速化する．更に，NCC 計算のための集約演算を GPU で処理することで，GPU から CPU へのデータ転送量を削減する．

GPU を用いて提案手法の実行時間を測定した結果，3〜5 秒で位置合わせを行えた．この実行時間は手術支援用途に対して十分に短く，GPU を用いた 2 次元／3 次元位置合わせは実用性が高いと考える．

なお，今回の実装では，単方向のみの投影に基づいて位置合わせを行っており，精度に関して改善の余地がある．今後は，二方向投影による位置合わせを実装し，精度の観点から実用性を検討したい．

謝辞　本研究の一部は，平成 17 年度厚生労働省がん研究助成金（15 指-1），科学研究費補助金特定領域研究（16016254）および（17032007）の補助による．

参 考 文 献

1) Lemieux, L., Jagoe, R., Fish, D.R., Kitchen, N. D. and Thomas, D. G. T.: A patient-to-computed-tomography image registration method based on digitally reconstructed radiographs, *Medical Physics*, Vol. 21, No. 11, pp. 1749–1760 (1994).
2) 川崎康博，伊野文彦，田代孝仁，中島義和，佐藤嘉伸，菅野伸彦，田村進一，萩原兼一：術中二次元/三次元剛体位置合せのための並列化手法，電子情報通信学会論文誌，Vol.J88-D-I, No.10 (2005).
3) Fernando, R.(ed.): *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, Addison-Wesley, Reading, MA (2004).
4) Stevenson, D.: A Proposed Standard for Binary Floating-Point Arithmetic, *IEEE Computer*, Vol.14, No.3, pp.51–62 (1981).
5) West, J., Fitzpatrick, J. M., Wang, M. Y., Dawant, B. M., Maurer, C. R., Kessler, R. M. and Maciunas, R.J.: Retrospective Intermodality Registration Techniques for Images of the Head: Surface-Based Versus Volume-Based, *IEEE Trans. Medical Imaging*, Vol. 18, No. 2, pp.144–150 (1999).
6) McLaughlin, R.A., Hipwell, J., Hawkes, D.J., Noble, J.A., Bryne, J.V. and Cox, T.: A Comparison of 2D-3D Intensity-Based Registration and Feature-Based Registration for Neurointerventions, *Proc. 5th Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'02), Part II*, pp.517–524 (2002).
7) Weese, J., Penney, G.P., Desmedt, P., Buzug, T. M., Hill, D. L. G. and Hawkes, D. J.: Voxel-Based 2-D/3-D Registration of Fluoroscopy Images and CT Scans for Image-Guided Surgery, *IEEE Trans. Information Technology in Biomedicine*, Vol. 1, No. 4, pp. 284–293 (1997).
8) Penny, G.P., Weese, J., Little, J.A., Desmedt, P., Hill, D. L.G. and Hawkes, D.J.: A Comparison of Similarity Measures for Use in 2-D–3-D Medical Image Registration, *IEEE Trans. Medical Imaging*, Vol.17, No.4, pp.586–595 (1998).
9) Cullip, T. J. and Neumann, U.: Accelerating Volume Reconstruction with 3D Texture Hardware, Technical ReportTR93-027, University of North Carolina at Chapel Hill (1993).