Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

CUDA-based Acceleration and Algorithm Refinement for Volume Image Registration

Abstract

In this paper, we propose a GPU-based acceleration method to speed up volume image registration using Compute Unified Device Architecture(CUDA). An novel CUDA-based method for joint histogram computation is introduced in this paper, which is also valuable for 2D image registration and other general graphics applications. Additionally, an algorithm refinement is proposed to improve the widely used FMRIB's Linear Image Registration Tool (FLIRT). Although extra time is taken by applying that algorithm refinements, our implementation showed the ability to perform a full 12 DOF (Degrees of Freedom) registration of two brain volume image in nearly 35 seconds, which is about 10 time faster than the native FLIRT implementation. Experimental results showed that our implementation with algorithm refinement can avoid some mis-registration.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Image Registration—GPU-based acceleration

1. Introduction

Image registration, as the process of finding the mathematical relationship between two images, plays an important role in many image processing and computer vision applications [MV97]. Especially in the medicine area, medical image registration is considered to be one of the major components of medical image processing with segmentation and visualization, and is commonly a base task of other high level operations, such like brain volume gain or loss calculation [SSM01], visualization of anatomy and tumors [AWM95], and treatment verification by comparison of pre- and post-intervention images [AVEH04].

Histogram-based techniques are wildly used in both 2D and 3D image registration solutions with or without deformation. While implementing histogram computation on CPUs is a trivial work, efficiently computing histogram on GPUs is quite difficult.

FMRIB's Linear Image Registration Tool (FLIRT) developed by Mark et [JS01] is the preferred tool for solving linear image registration problems [Kle09]. It employs local optimization methods together with multi-resolution techniques to find the global minima, and global optimization methods are also involved to improve its reliability. However, speed is an issue for FLIRT. As the author reported, it took about one hour to register two brain volumetric images when this algorithm was first proposed at 2001, and according to our latest tests with top-end workstations (4x3.2G Xeon processors, 16GB RAM), it still took nearly 5 minutes to perform a 12 DOF registration.

The GPU-based solution for joint histogram computation will be presented in this paper. By utilizing proposed acceleration technique, we entirely reimplemented the FLIRT algorithm together with an algorithm refinement which can avoid some mis-registration. Our GPU-accelerated implementation of this tool takes only 35 seconds to perform a fully 12 DOF linear registration without any algorithm simplification. This accelerated technique may introduce opportunities for time-critical applications.

2. Related Work

GPUs has been widely used for general purpose computing in various applications, beyond the original target of computer graphics and gaming industry. Attempts of using GPU to accelerate image registration can be often seen from 2004, when programmable vertex and fragment shaders were added to the graphics pipeline of modern GPUs. In 2005, Kim presented a 2D-3D image registration method [JKM05] , which used GPUs for digitally reconstructed radiographs (DRR) generation as preprocessing for registration, other several DRR-based 2D-3D image registration al-

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

gorithms were also presented at 2006 [FIH06] [AKS06]. All these earlier algorithms doesn't have the capacity of registering two volume images.

Compute Unified Device Architecture(CUDA) [Cor08] is a parallel computing architecture developed by NVIDIA in 2007. Comparing with traditional GPGPU techniques, CUDA has several advantages, such as scattered reads, shared memory, faster downloads and readbacks to or from the GPU, and fully support for integer and bitwise operations.

Since the histogram calculation and mutual information computation are considered as the performance bottleneck for linear registration and some non-linear registration algorithms, researches has been conducted to accelerate this process by GPUs, especially after CUDA was available. Podlozhnyuk presented his work of calculating histogram of a single image with CUDA at [Pod07]. Based on Podlozhnyuk's method, Shams proposed a method to compute the joint histogram of two image [SB07] and applied this method to mutual information calculation. However, Shams' method still has a disadvantages: each thread block of its CUDA kernel has to traverse all voxels in the volume, to compute the corresponding part of the joint histogram, although most of traversed nodes will be finally skipped.

Our CUDA-accelerated method for joint histogram calculation is distinct, which first employs a fast CUDA-based sorting algorithm as data pre-processing, and then distributes the computation work to different thread blocks to avoid cumbersome computation.

Beyond this acceleration work, we implemented the full linear image registration based on our efficient mutual information computation method. And while implementing different optimization methods to our system, some algorithm refinement tips were found and experimental results showed these change allowed us to obtain better registration results.

3. Linear Registration of Volume Images

By naming the baseline image as reference image and the other as registering image, the linear registration problem can be described as: seeking a affine transformation, which maximizes the similarity between the transformed registering image and the reference image when it's applied to the registering image. Mathematically this problem then can be divided into problems:

1), Define a cost function, which measures the dissimilarity of the reference image and transformed registering image.

$$C = f(A, I_{ref}, I_{reg}) \tag{1}$$

while I_{ref} and I_{reg} represent the reference image and registering image, and A is a 4x4 matrix represents the affine transformation:

$$A = C_{I_{ref}} \times T \times R \times G \times S \times (-C_{I_{reg}}) \tag{2}$$

While

$$C_I = \begin{pmatrix} 1 & 0 & 0 & I.center.x \\ 0 & 1 & 0 & I.center.y \\ 0 & 0 & 1 & I.center.z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad I \in \{I_{ref}, I_{reg}\} \tag{3}$$

The matrix $-C_{reg}$ transforms the coordinate system of registering image to center based, $I_{reg}.center$ denotes the center of its mass and will be set a candinate of $(0,0,0)$, T,R,G and S denote the matrixs of translation, rotation, skew and scaling transformation under this shifted coordinate system, the matrix C_{ref} finally restores the coordinate system to left-top based.

2), Optimization methods to find the minimum of this cost function at allowable domain.

$$A^* = \{A^* | \forall A \in D : f(A, I_{ref}, I_{reg}) \geq f(A^*, I_{ref}, I_{reg})\} \tag{4}$$

While D denotes the domain of all the allowable transformation.

3.1. Cost Function

Mutual information as the classical similarity measurement was employed in our implementation. Studies and experiments of different methods indicated that this intensity-based is much more accurate and reliable than the geometrically-based methods [MJS02], and is most commonly used in both the linear and nonlinear image registration fields.

According to the information theory, the mutual information of the two images is defined as:

$$C^{MI} = H(I_{ref}, I_{reg}) - H(I_{ref}) - H(I_{reg}) \tag{5}$$

while $H(I_{ref})$ and $H(I_{reg})$ denotes the entropy of reference image and registering image.

$$H(I) = -\int_I p(x) \log p(x) dx \qquad I \in \{I_{reg}, I_{reg}\} \tag{6}$$

$H(I_{ref}, I_{reg})$ represents the joint entropy of these two registering images, and $p(x)$ denotes the probability density of variable x.

$$H(I_{ref}, I_{reg}) = -\int_{I_{reg}} \int_{I_{ref}} p(xy) \log p(xy) dxdy \tag{7}$$

Our experiments showed that more than 90% computation work of an image registration problem is about cost function computation. After briefly introducing the FLIRT algorithm, we'll then show the difficulties of computing this mutual information in a parallel programming model, and present our solution to resolve that problem.

3.2. The FLIRT Algorithm

Once the cost function is specified, the registration problem then becomes an optimization problem. Multi-resolution

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

technique is employed in FLIRT to avoid trapping in the local minima. The entire algorithm can be divided into three stages:(1)low resolution with 8mm spacing, (2)middle-level resolution with 4mm spacing and (3)high resolution with 2mm or 1mm spacing.

In practice, the reference image and registering image are initially resampled to 1mm spacing, images at 2mm, 4mm, 8mm spacing are then produced by half-sampling the finer images.

3.2.1. 8mm Stage: Coarse Search

Search in this stage is restricted to the rotation parameters, as they are the most difficult to find [JS01]. A coarse 3D grid with $M \times M \times M$ dimension is formed, in which each cell represents a rotation with specified angles to 3 different axes. A full local optimization of translation and global scaling is then performed for each specified rotation. After obtaining the result of translations and global scalings, a finer grid is then formed by interpolating that coarse grid. Cost function is then evaluated again for each cell, and finally several local minimums (has lower value than its 8 neighbors) with lowest cost function value are selected for next stage.

3.2.2. 4mm Stage: Perturbations

The several minimums found at previous search stage are then considered as the candidate transformation at this stage. Additionally, several perturbations of the rotation angles for this candidates will be taken and local optimization will be performed for these perturbations. A single best solution is selected and passed to higher resolution stage.

3.2.3. 2mm and 1mm Stage: High Resolution Optimization

With the single solution obtained at previous 4mm stage, a 7 DOF optimization with rotation, translation and global scaling is performed at 2mm stage. Considering the anisotropic scalings and skews start to become significant in 2mm stage, 9 DOF and 12 DOF optimizations are then performed for this solution. Finally, a single pass of 12 DOF optimization is performed at 1mm spacing, and the optimized transformation will be returned as the final result.

3.3. Histogram Computation

Histogram is used to compute mutual information in our implementation, since the other method which utilizes Parzen window [LM08] as an approximation method usually introduces volatility due to it's heavily sampling sensitive.

For the two images I_{ref} and I_{reg} with a affine transformation A, their joint histogram can be mathematically described as:

$$J(i,j) = \sum_{x \in I_{reg}} \delta(x) \tag{8}$$

while

$$\delta(x) = \begin{cases} 1 & I_{ref}(x) \in BIN_{ref}.i, I_{reg}(Ax) \in BIN_{reg}.j \\ & \text{And the point } (Ax) \text{ locates in referrence volume} \\ 0 & \text{Others} \end{cases} \tag{9}$$

Once the joint histogram is obtained, the marginal histogram of each single images can be calculated.

4. Joint Histogram Computation with CUDA

As is mentioned in section 2, in Shams' method, each thread block of its CUDA kernel has to traverse all voxels in the volume and skip most.

The operation that tests all voxels in a voxel and skips the undesired ones may not slow down the program seriously for CPU-based applications, but does make a strong impact to performance of GPU-based ones as it will greatly scale the circulation and produce more branches of program stream, which are very expensive on GPUs.

In order to circumvent the shortcomings of Shams' method, a new volume V_{reg}, in which each voxel stores the value and the 3D coordinate of registering image is built.

$$V_{reg} = \{(coord, val) | coord = x, val = I_{reg}(x), x \in I_{reg}\} \tag{10}$$

Our developed CUDA-based sorting algorithm [Ref to our work needed] is then employed to sort this volume by each voxel's value. After that sorting operation is done, the voxels V_{reg} is continuously distributed due to their value, and all the desired voxels for a same histogram part calculation have been put together. Pivots in V_{reg} representing the boundaries for different bins can be calculated, and we can then reform the Eq 8. as:

$$J(i,j) = \sum_{x \in V_{reg}(j)} \delta(x.coord) \tag{11}$$

while

$$V_{reg}(j) = \{v | v \in V_{reg}, v.val \in BIN_{reg}.j\} \tag{12}$$

By applying this method, each block will be working with specified part of the V_{reg}, this can not only avoid cumbersome computation, and also greatly reduce the expensive global memory access, and accordingly makes our implementation much more efficient than Shames'. Fig 1 shows a diagram of our method.

Note that although V_{reg} might be used for thousands of times, it just need to be sorted one time at the initialization for each resolution stage. The CUDA-based sorting algorithm we employ is very efficient that only takes 120ms to sort a 3M voxels.

We'll then show the techniques and strategies we used in our implementation.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

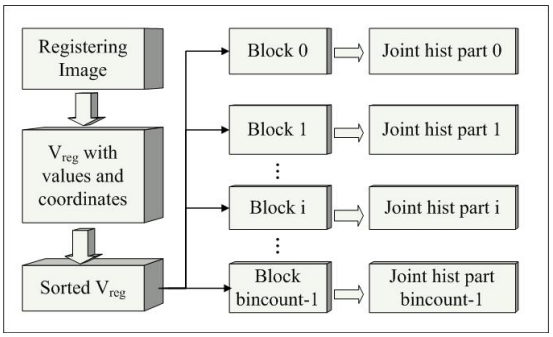


Figure 1: *the coarse-grained flowchart of our algorithm.*

4.0.1. Joint Histogram Part Computation in One Block

As is implied at Fig 1, one thread block is allocated for computing the corresponding part of the joint histogram. For example, the k block is responsible for the computation of $J(i,k)$ while $0 \leq i < BIN_{ref}.count$

How to deal with the intra-block memory access conflict remains a problem. We allocate a sub-part-histogram in the shared memory for each warp so that the memory access collision can only happen at intra-warp, due to the fact that in each hardware cycle only a warp of threads actually execute simultaneously on a stream multi-processer [Cor08]. After each warp completing their sub-part computation, an in-block reduction operation is performed to gather the distributed sub-parts. Fig 2 shows the flowchart of in-block histogram part calculation.

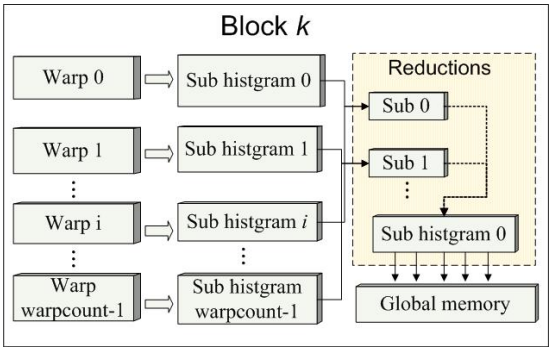


Figure 2: *the joint histogram part calculation process in one block.*

The technique of simulating atomic operation to shared memory unit is employed, more information about this simulated atomic technique can be found at [Pod07].

4.0.2. Interpolation

The transformed coordinate Ax commonly doesn't coincide with any primary points, so an interpolation method is required to fetch the value $I_{reg}(Ax)$. In particular, a trilinear interpolation is chosen as it's efficient and the most commonly used for 3D applications.

CUDA 2.0 started to support 3D texture, together with the capacity of hardware built-in trilinear interpolation. This new feature performs a 9-bits calculation to apply the trilinear interpolation. Our algorithm employs this new technique when at 2mm and 1mm resolution stages, while full 32-bits software-based interpolation is applied at lower resolution stages due to the accuracy.

4.0.3. Adaptive Strategy

Commonly, there is lots of blank space with intensity 0 in a volume image, a brain MRI image for example, has 0 intensity outside of the skull as there is nothing but air.

In this case, the first block might be over-loaded as it's charging for computing $J(i,0), 0 \leq i < BIN_{ref}.count$, which may account for one-third of the computation. This issue may finally reduce the performance as other blocks will be just waiting while the first block is busy computing.

To resolve this unbalanced loading problem, a adaptive strategy is used. The first bin of registering image will be particularly treated if:

$$count(BIN_{reg}.0) > BIN_{reg}.count^2 \qquad (13)$$

Solution for this problem is to more fatherly distribute the computation work. $BIN_{ref}.count$ blocks will be allocated for the first bin calculation, and the block k will be only responsible for computing $J(k,0)$.

4.0.4. Performance Evaluation

With fully implementing the transformation and interpolation, our CUDA-based method showed an efficiency improvement of mutual information calculation by a factor of 10 to 50. Table 1 and 2 shows the performance comparison of our algorithm with CPU-based method and Shams' method, all tests are run at a workstation computer equipped with 4×2.4G Xeon Processers, 16G RAM and a NVIDIA FX 5800 graphics card.

The author's implementation was employed when evaluated Shams' method, we can find Shams' method achieves desirable performance for MI computation on large images, but shows low-efficiency (even slower than CPU) on small images. Although higher acceleration ratio can be obtain on large images, our method also shows a striking performance improvement over CPU on small images.

Joint histogram partition with 256×256 bins is necessary for image registration when at high resolution stages. As can be found at 2, the performance of CPU-based implementation shows little change from 128×128 bins to 256x256 bins, while GPU-based ones take $1/3$ more time.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

Table 1: *Performance comparison of 128×128 bins (time:ms)*

Volume Size	CPU[1]	Shames[2]	Our[3]	Our[4]
29×29×31	2.5	5.9	0.53	0.45
58×58×62	18.7	6.3	1.23	0.96
115×115×120	141.6	14.9	6.9	4.1
230×230×239	1071.7	38.5	26.5	13.5

[1] CPU-based method
[2] Shams' method without transformation, and no interpolation.
[3] Proposed method with software-based interpolation
[4] Proposed method with hardware-based interpolation

Table 2: *Performance comparison of 256×256 bins (time:ms)*

Volume Size	CPU[1]	Shames[2]	Our[3]	Our[4]
29×29×31	2.6	Ø	0.82	0.62
58×58×62	18.7	Ø	1.6	1.1
115×115×120	145.3	Ø	8.9	5.7
230×230×239	1107.7	Ø	33.5	16.5

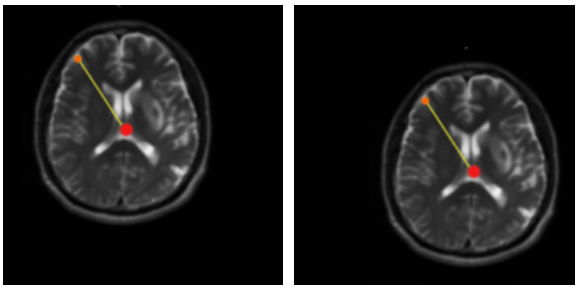
[2] Shams' implementation can't support 256x256 bins, so its result is not presented here.

5. Algorithm Refinement: *Z*-first and *Z*-perturbation

According to the practical medical image registration problems, we here present an algorithm refinements to improve the stability of FLIRT algorithm.

Due to the fact that most medical images at each slice are commonly not full-filled(has blank around), the translation of two volume image in direction of X- and Y-axis are usually very small, as we have shifted the coordinate system to be center-based. Fig 3 gives an example.

On the contrast, the mass center in Z-axis is commonly very closed to the middle in depth of entire volume. Experimental results also showed the two images usually have a more than 10mm translation in Z, while only have less than 3mm translation in X and Y.



(a) (b)

Figure 3: *Although a big translation happens, the coordinate of the left-top point related to the mass center doesn't take a big change.*

We addressed this problem by using Z-first and Z-perturbation strategy.

5.1. *Z*-first

Since linear image registration is a high-dimensional optimization problem with up to 12 DOF, the local optimizations are actually performed at each dimension. In these N 1D optimization methods, the order of different dimensions to be optimized could affect the convergence rate and correctness of final result. The most influential dimension should be performed first as it usually sets the tone of the final result and will lead the later optimizations.

Z-first means the optimization for dimension Z will be first performed, to make sure that the result in dimension is minimally to be affected by optimization result on X and Y.

5.2. *Z*-perturbation

In addition, a Z-perturbation is performed at 2mm stage to avoid Z-translation mis-registration.

Once 4mm stage is done, a single best solution $\{(r_x, r_y, r_z), (t_x, t_y, t_z), s\}$ is determined representing the rotation, translation and scaling under center-based coordinate system. Then, t_z and its neighbors $\{t_z, t_z \pm 1mm, t_z \pm 2mm, t_z \pm 3mm\}$ will be completely traversed, at which optimizations with other 6 parameters will be performed.

This technique is implemented to correct the Z mis-registration, which commonly accounts for the majority of registration errors. Note that this Z-perturbation might be time consuming for CPU-based implementations, but it doesn't make a big impact to the performance, since it takes only 5.7 ms for a cost function calculation at 2mm stage.

Fig 5. shows the mis-registration of FLIRT algorithm, and our correct registration with our refined FLIRT algorithm.

6. Experimental Results

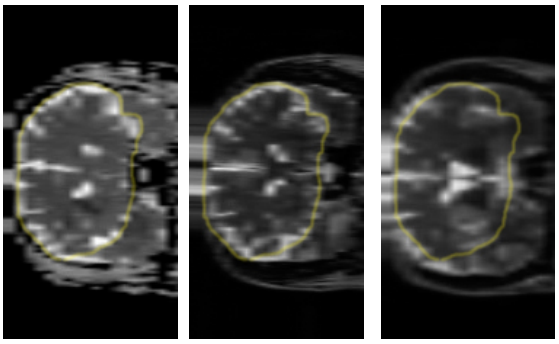
With each 6 experiments for ADC-T2, T1-T2, CT-T1 image registration conducted on the workstation mentioned above, mean errors and obvious mis-registrations for our implementation and native FLIRT were manually evaluated, and reported at Table 3.

Which can be found from Table 3 is that our algorithm has shown large performance improvement over native FLIRT and can obtain results with smaller error while registering MR-MR images (ADC, T1 and T2 represents different types of MR images). However, our algorithm is also found that

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

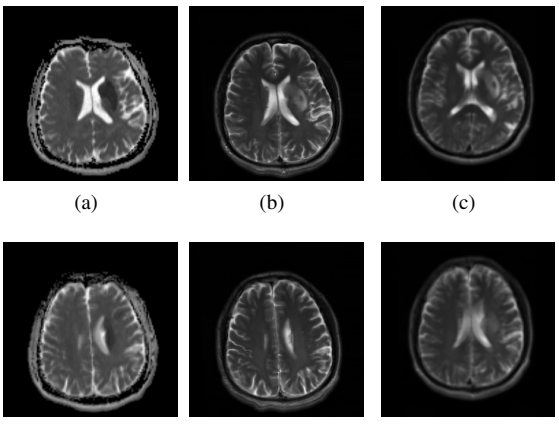
Table 3: *Experimental results*

Type	FLIRT error	FLIRT mis-reg	FLIRT time	Our method's error	Our method's mis-reg	Our method's time
ADC-T2	4.4 mm	1	325 s	2.5 mm	0	32 s
T1-T2	2.8 mm	0	365 s	2.3 mm	0	38 s
CT-T1	6.8 mm	2	372 s	6.9mm	2	39 s



(a) (b) (c)

Figure 4: *Cross-section of the result of aligning T2 image to ADC image(a), while our implementation(b) gives a desirable result with Z-perturbation technique, the native FLIRT(c) raises a mis-registration*



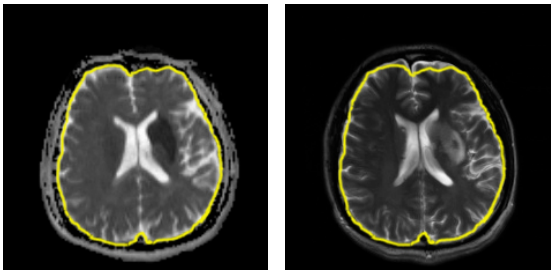
(a) (b) (c)

(d) (e) (f)

Figure 5: *mis-registration of native FLIRT algorithm, and correct result with our refined FLIRT algorithm. The left column is the reference MR-ADC image, while middle and right column are the MR-T2 images registered with our algorithm(middle) and FLIRT(right).*

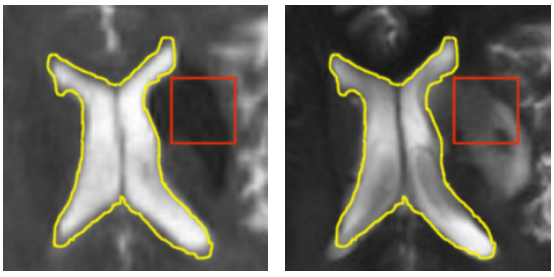
it doesn't address the CT-MR image registration problem. This indicates that multi-modal image registration, as an under-determined problem, requires more techniques than optimization methods.

Fig 6. and detail-enlarged Fig 7. then show a result of typically ADC-T2 image registration using our refined algorithm.



(a) (b)

Figure 6: *Result of MR-T2 image (b) registered to MR-ADC image (a). Note that the front-head of ADC images are commonly incomplete, so the profile there doesn't well match the registered T2-image's.*



(a) (b)

Figure 7: *Part-enlarged comparison of the lateral ventricle cut from above two images, red-marked region as lesion area (cerebral hemorrhage or ischemia) is aligned.*

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

7. Conclusion and Future Work

A CUDA-based acceleration method together with fast mutual information calculation techniques are proposed in this paper to speed up volume image registration. Some algorithm refinements are also presented to improve registration stability of FLIRT. Implementation of proposed method allow us to perform a full 12 DOF registration of two brain volume image in nearly 35 seconds as 10 time faster than the native FLIRT implementation. Experimental results show our refined algorithm has the error less than the native FLIRT's.

CUDA-based acceleration for image registration validation and correction will be focus of our future work. An project for acceleration non-linear image registration will also be kicked off.

References

[AKS06] A. KHAMENE P. BLOCH W. W. M. S., SAUER. F.: Automatic registration of portal images and volumetric ct for patient positioning in radiation therapy. *Medical Image Analysis* (Feb 2006), 96–112.

[AVEH04] ANN VAN ESCH T. D., HUYSKENS D. P.: The use of an asi-based epid for routine absolute dosimetric pre-treatment verification of dynamic imrt fields. *Radiotherapy and Oncology 71*, 2 (May 2004), 223–234.

[AWM95] ARNE WAGNER OLIVER PLODER G. E. M. T., MATTHEWS P.: Virtual image guided navigation in tumor surgery ąl technical innovation. *Journal of Cranio-Maxillofacial Surgery 23*, 5 (Oct 1995), 271–273.

[Cor08] CORPORATION N.: Nvidia cuda compute unified device architecture programming guide 2.0.

[FIH06] F. INO J. GOMITA Y. K., HAGIHARA K.: A gpgpu approach for accelerating 2-d/3-d rigid registration of medical images. *Parallel and Distributed Processing and Applications(ISPA), 4330*, 2 (2006), 939–950.

[JKM05] J. KIM S. LI Y. Z., MOVSAS B.: Real-time intensitybased deformable fusion on pc graphics hardware. In *International Conference on the Use of Computers in Radiation Therapy* (2005).

[JS01] JENKINSON M., SMITH S.: A global optimisation method for robust affine registration of brain images. *Medical Image Analysis 5*, 2 (June 2001), 143–156.

[Kle09] KLEIN A.: Evaluation of 14 nonlinear deformation algorithms applied to human brain mri registration. *Medical Image Analysis 46*, 3 (July 2009), 786–802.

[LM08] LIN Y., MEDIONI G.: Mutual information computation and maximization using gpu.

[MJS02] M. JENKINSON PETER BANNISTER M. B., SMITH S.: Improved methods for the registration and motion correction of brain images. *Technical report* (2002), 143–156.

[MV97] MAINTZ J. B. A., VIERGEVER M. A.: A survey of medical image registration. *Medical Image Analysis 2*, 1 (March 1997), 1–36.

[Pod07] PODLOZHNYUK V.: Histogram.

[SB07] SHAMS R., BARNES N.: Speeding up mutual information computation using nvidia cuda hardware. *Digital Image Computing Techniques and Applications* (2007), 555–560.

[SSM01] S.M. SMITH N. DE STEFANO M. J., MATTHEWS P.: Normalised accurate measurement of longitudinal brain change. *Journal of Computer Assisted Tomography 25*, 3 (May 2001), 466–475.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.