33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

# GPU Implementation of a Deformable 3D Image Registration Algorithm

Hamed Mousazadeh, Bahram Marami, Shahin Sirouspour and Alexandru Patriciu

*Abstract*— **We present a parallel implementation of a new deformable image registration algorithm using the Computer Unified Device Architecture (CUDA). The algorithm co-registers preoperative and intraoperative 3-dimensional magnetic resonance (MR) images of a deforming organ. It employs a linear elastic dynamic finite-element model of the deformation and distance measures such as mutual information and sum of squared differences to align volumetric image data sets. Computationally intensive elements of the method such as interpolation, displacement and force calculation are significantly accelerated using a Graphics Processing Unit (GPU). The result of experiments carried out with a realistic breast phantom tissue shows a 37-fold speedup for the GPU-based implementation compared with an optimized CPU-based implementation in high resolution MR image registration. The GPU implementation is capable of registering 512x512x136 image sets in just over 2 seconds, making it suitable for clinical applications requiring fast and accurate processing of medical images.**

## I. INTRODUCTION

Medical image registration is the process of aligning images so that corresponding features can be easily matched. This process plays a significant role in clinical interventions such as biopsy, image-guided surgery and radiotherapy planing. A fast and reliable registration algorithm allows for surgical plans based on preoperative images to be updated using real-time image feedback to compensate for tissue movements/deformations.

Deformable registration algorithms can account for both rigid and non-rigid tissue transformations. However, the complexity of these methods often significantly increases the runtime in conventional single CPU-based implementations [1]. Recently, GPUs, Field-Programmable-Gate-Array (FPGA) devices and multi-processor systems have been used to accelerate image registration algorithms. In [2], parallel processing with 64 CPUs using a shared memory architecture has resulted in average runtimes of 67 seconds and 89 seconds for non-rigid registration in intraoperative brain deformation analysis and contrast-enhanced MR mammography, respectively. Maintenance and acquisition costs are the main drawbacks of such parallel computing systems [3].

FPGA-based computing architectures are highly customizable and therefore can significantly speed up computations, if properly designed. A study of rigid registration using FPGAs is presented in [4]. GPUs are less expensive, and easier to program than FPGAs, making them widely popular in scientific computing applications in recent years. Near real-time registration can be achieved with recent advances in the GPU technology [5]. A GPU-based implementation of 2D and 3D rigid and nonrigid registration algorithms was presented in [6]. A runtime of about one minute was reported for 3D registration of 256x256x128 data sets.

The Computer Unified Device Architecture (CUDA) provides a powerful and user-friendly programming environment for GPU-based scientific computing applications including those in medical imaging. We have recently developed a new deformable image registration algorithm that employs a dynamic linear elastic deformation model in conjunction with image similarity measures such as sum of squared differences (SSD), mutual information (MI), and correlation ratio (CR) [7]. The algorithm involves computationally expensive tasks such as interpolation, solving a system of second-order differential equations, finding the 3D deformation using linear shape functions on tetrahedral finite elements, and solving a large linear system of equations based on the conjugate gradient method. In this paper, we report an implementation of the algorithm for 3D-3D MR registration based on SSD on a CUDA capable NVIDIA GTX 480 GPU. We have carried out experiments with a realistic breast phantom in order to volumetrically register high-resolution MR images.

## II. METHODS

### A. Mathematical Formulation of the Registration Problem

The registration problem can be formulated as finding the displacement field $u$ that minimizes the following cost function:

$$J(u) = D(T[u], R) + \frac{\alpha}{2} u^T K u; \quad \alpha \in \Re_+ \tag{1}$$

where $D$ is a distance measure between the reference $R$ and the deformed template $T[u]$ images. The second term in (1) is the linear elastic energy of the deformed object which is weighted by a constant value $\alpha$. This regularization term is based on a physical model for the deformation that would ensure that the resulting solution is "reasonable".

We employ the finite element method (FEM) to discretize the deformation model in the spatial domain using a volumetric tetrahedral mesh. Therefore in (1), $K$ is the global stiffness matrix associated with the volumetric mesh, and $u$ is the vector of nodal displacements. The nodal displacements are propagated to any point $x$ of the template image volume using the shape functions of the element containing the point:

$$u_{in}(x) = \sum_{i=1}^{4} N_i^{el}(x) u_i^{el}(x) \tag{2}$$

H. Mousazadeh is with the School of Biomedical Engineering, McMaster University, Hamilton, Ontario L8S4L8 , Canada

B. Marami, S. Sirouspour and A. Patriciu are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario L8S4L8 , Canada. Please send correspondence to sirouspour@ece.mcmaster.ca

978-1-4244-4122-8/11/$26.00 ©2011 Crown

where $u_{in}(x)$ is the displacement for the internal point and $u_i^{el}(x)$ is the displacement of the nodal points of the element. $N_i^{el}(x)$ is the shape function of the elements which is explained in details in [8]. If $J$ in (1) has a minimum at $u$, its first derivative must vanish, i.e.

$$\frac{\partial J(u)}{\partial u} = \frac{\partial D(T[u], R)}{\partial u} + \alpha K u = 0 \tag{3}$$

This equation can be written as a set of nonlinear finite element equilibrium equations for static analysis

$$Ku = f(u) \tag{4}$$

where $f(u) = -\frac{1}{\alpha}\frac{\partial D(T[u],R)}{\partial u}$ is the vector of concentrated nodal forces applied to the volumetric mesh. The solution to (4) will provide the displacement field corresponding to the global minimum of the objective function (1).

### B. Dynamic Finite Element Model

The force vector $f(u)$ in (4) is a nonlinear function of the displacement field $u$. An iterative numerical method has to be employed to solve the nonlinear system of equations in (4). To this end, we consider the following second-order dynamic model for the motion of deformable object [8]:

$$M\ddot{u} + C\dot{u} + Ku = f(u) \tag{5}$$

where $M$ is the mass matrix of the elements concentrated at nodes, and $C = \beta M + \gamma K$ is the damping matrix for constant values of $\beta$ and $\gamma$. The steady-state equilibrium of this dynamic system is the solution to the static system of equations in (4). In (5), the dynamic forces $M\ddot{u}$ and $C\dot{u}$ tend to smoothly drive the system towards this equilibrium. The dynamic system of equations can be solved using existing implicit or explicit numerical integration routines. The solution can be obtained more effectively by using the transformation $u = \phi v$ on the $n$ finite element nodal point displacements, where columns of $\phi$ are eigenvectors of $M^{-1}K$ [8]. With this change, (5) can be written as:

$$\hat{M}\ddot{v} + \hat{C}\dot{v} + \hat{K}v = \hat{f} \tag{6}$$

where $\hat{M} = \phi^T M \phi$, $\hat{C} = \phi^T C \phi$, $\hat{K} = \phi^T K \phi$ are diagonal matrices, and $\hat{f} = \phi^T f$. Assuming that $M^{-1}K$ is full rank, the finite element equilibrium equations are decoupled. The fast modes of (6) can be eliminated without affecting the steady-state solution, resulting in a significant reduction in the computation time.

### C. Algorithm

In each iteration of the algorithm, the nodal forces $f(u)$ have to be computed and then used to drive the original dynamical system in (5) or the decoupled reduced system in (6). The registration solution is obtained when the system reaches its steady state. In this paper, we use the SSD [9] as the distance measure because it is computationally efficient and works relatively well in single modality registration. Therefore, the force vector at any nodal point can be computed as [9]:

$$f(u_i) = -\frac{1}{\alpha}\left(T(p_i + u_i) - R(p_i)\right)\nabla T(p_i + u_i) \tag{7}$$

where $f(u_i)$ is a 3×1 vector, $\nabla T(p_i + u_i)$ is the gradient of the deformed template image at the deformed nodal point $p_i + u_i$. $T(p_i + u_i)$, $R(p_i)$ and $\nabla T(p_i + u_i)$ are computed using the trilinear interpolation algorithm at each iteration. When (6) reaches an equilibrium, the resulting displacement of the finite element nodal points $u$ can be used to calculate the displacement of the regular 3D high-resolution grid of the template image based on the shape function (2).

Equation (7) shows that the force vector is computed based on the pixel values and the gradient of the template image only at the nodal points. Obviously, the registration accuracy is improved by increasing the resolution of the finite element mesh at the expense of a greater computational load. In order to use the whole information of the reference and template images, displacements which are computed in a finer 3D regular grid $x_p$ inside the image volume are used to find the nodal point forces. The displacement of the regular grid are computed according to (7), i.e. $\Delta x = -\kappa\left(T(x_{p+u}) - R(x_p)\right)\nabla T(x_{p+u})$, where $\kappa \in \Re_+$, $x_p$ and $x_{p+u}$ represent the original and the deformed grid points in each iteration. Nodal forces can be approximated based on $\Delta x$ using the inverse of the shape function. This requires solving a linear system of equations of the form $Af = b$, where $A$ is a long matrix derived from the linear shape function of tetrahedral elements and vector $b$ results from $\Delta x$ of the regular grid.

## III. GPU Implementation

We have implemented our registration algorithm on NVIDIA's GeForce GTX 480 GPU which has 15 streaming multiprocessors, each with 32 streaming processors (SPs), i.e. a total of 480 SPs. The hardware provides support for the execution of thousands of threads in parallel with minimum scheduling overhead. The threads are grouped in blocks and the blocks are organized in a grid. The flowchart in Fig. 1 summarizes the algorithm.

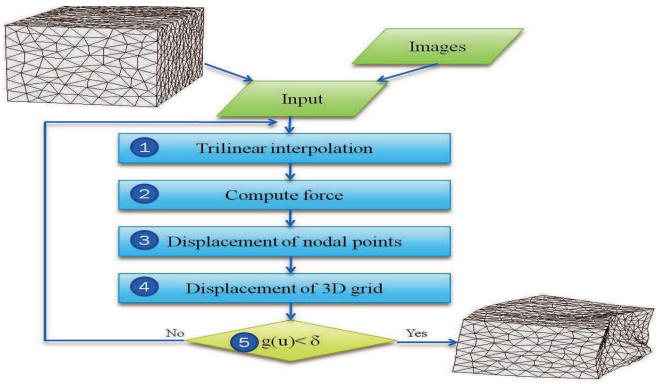



Fig. 1. Registration algorithm flowchart

### A. Implementation Steps

1) Trilinear interpolation: this step requires trilinear interpolations of the template image and its directional gradients at $x_{p+u}$.

2) Compute force: the over-determined system of equations $Af = b$ is solved using the linear least squares and

conjugate gradient method to compute the vector of nodal forces.

3) Displacement of nodal points: having computed $f$, $\hat{f}$ is calculated and the reduced system of equations in (6) is solved for one time step; the original displacement field is computed from $u = \phi v$.

4) Displacement of 3D grids: having computed $u$, the deformed 3D grid in the template image can be obtained from the deformed finite element mesh using the linear shape function of the tetrahedral elements. The final high-resolution deformed image can be computed similarly.

5) Distance measure: the SSD between the deformed template and the reference images is computed. The iterations stop when the relative error in the SSD falls below a threshold, i.e.

$$g(u) = \frac{|SSD(u_t) - SSD(u_{t-1})|}{SSD(u_{t-1})} < \delta \qquad (8)$$

where $u_t$ and $u_{t-1}$ are the displacement vectors at the current and previous iterations, respectively.

### B. Program Optimization

GPU hardware provides support for massively parallel computing but the execution resources are obviously limited and shared between the execution threads. Therefore, the block and gird size and the memory use have a great influence on the execution performance [10]. Due to lack of space, only Step 4 of the algorithm is discussed below while similar arguments apply to the other steps.

Step 4 computes the displacement field using the shape function (2) of a volumetric mesh with 2335 tetrahedral elements and 515 nodes, encompassing a 512x512x136 image and a 128x128x50 regular grid. Fig. 2 shows the execution time for this step based on different host memory configurations. The memory transfer portion, memcpy in Fig. 2, includes both host (CPU) to device (GPU) and device to host transfers. The regular pageable memory, M1, has the highest runtime of 19.23 ms which includes the kernel time plus the memory transfer time. The mapped memory, M2, maps a block of page-locked host memory into the device. This block has two addresses in host and device and thus the programmer does not need to explicitly allocate a block in GPU for memory transfer [10]. In fact this data transfer is implicitly performed by the kernel and hence its time is included in the kernel time. The write-combined memory configuration, M3, exhibits the best performance of all three with a total runtime of 10.43 ms. M3 transfers data across the Peripheral Component Interconnect Express (PCIe) more quickly and increases the performance especially when GPU reads the buffer that is written by CPU.

The performance of a few possible CUDA thread configurations in Step 4 of the algorithm are compared in Table I. Configuration 3 has the best performance in terms of kernel execution time. Configurations 1 and 2 have relatively poor performance due to insufficient use of execution resources such as registers. Similarly, Configurations 4 and 5 are not as efficient because of excessive use of registers and other execution resources.

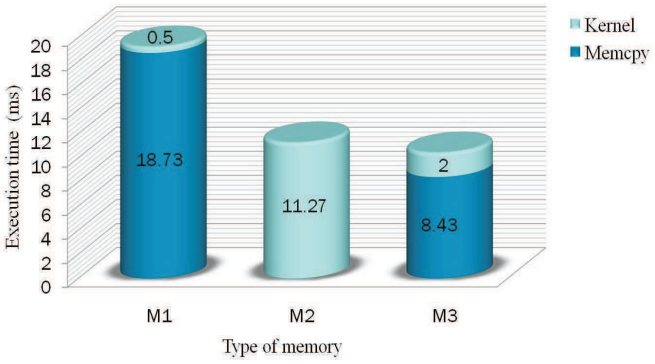



Fig. 2. Total execution time (memcpy + kernel) is shown for displacement. Three types of host memories have been used. M1 is the regular pageable memory, M2 is the mapped memory and M3 is the write-combined memory.

## IV. Results

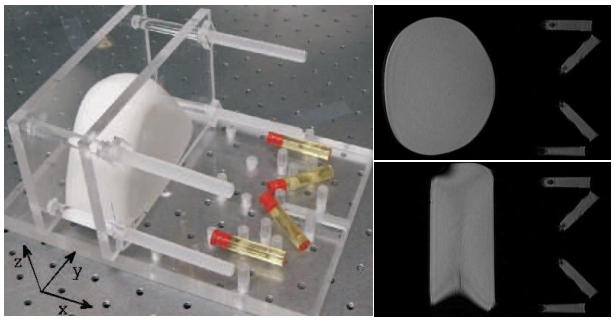
A triple modality biopsy training breast phantom (CIRS model 051) has been used for obtaining 3D volume high resolution (512x512x136) MR images in both the un-deformed and the deformed states. Fig. 3 shows an MR compatible plexiglass structure and four capsules of vitamin E that are mounted on the framework as landmarks to match the coordinates of the deformed and un-deformed images. COMSOL Multiphysics and Simulation software is used just to create a cubic finite element mesh of 7502 linear tetrahedral elements with 1601 nodal points. Using this mesh, we would have 4803 decoupled equations in (6). In our experiments, we only used 500 slowest modes of (6) for solving the dynamic system. The deformation model is used as a tuneable constraint on the registration process and therefore exact material and geometrical properties of the tissue are not required in the model. The cubic FE mesh encompasses the whole volume of deformed and un-deformed data and a 20x30x10 finer regular grid (see Section II-C), with no need for the segmentation of image.

### A. Execution Time

The algorithm converges to a solution after 15 iterations. The CPU execution time for the 3D high resolution image registration is 82.83 sec. compared to only 2.19 sec. for the GPU, i.e., a 37-fold speedup is achieved. The CPU implementation which is written in standard C ran on a

TABLE I
GPU RUNTIME AND GFLOP/S OF THE CUDA KERNEL FOR 3D GRID DISPLACEMENT FIELD

| Config | Block Size (B), Grid Size (G) | GFLOP/s | time (ms) |
|---|---|---|---|
| 1 | B(1,64), G(1,38400) | 25.84 | 1.87 |
| 2 | B(1,128), G(1,19200) | 28.94 | 1.67 |
| 3 | B(1,256), G(1,9600) | 29.65 | 1.63 |
| 4 | B(1,512), G(1,4800) | 26.55 | 1.82 |
| 5 | B(1,1024), G(1,2400) | 22.48 | 2.15 |




(a) (b)

Fig. 3. (a) the apparatus, (b) x-y views of un-deformed (top) and deformed (bottom) images

3.20 GHz Intel Core i5 650 processor with 4GB RAM. The composition of the GPU and CPU execution times are given in Table II. Step 1, which involves 4 trilinear interpolations, exploits the hardware built-in 3D texture trilinear interpolation. Step 2 involves very large sparse matrix operations, with the conjugate gradient method taking 70 ms of this step. CUSPARSE library has been used in Step 2 and 3 for sparse matrix operations. Linear shape function (2) has been implemented in Step 4. SSD is computed in Step 5 and "GPU others" refers to memory transfers and other miscellaneous computations.

### B. Quantitative Study and Quality of Registration

The 3D preoperative image in Fig. 4(top left) and the 3D intraoperative image in Fig. 4(top right) are the inputs of the registration algorithm. Fig. 4(bottom left) and Fig. 4(bottom right) are the outputs of GPU and CPU-based implementations, respectively. The SSD between the intraoperative and the deformed preoperative data are 285.11 and 282.08 for the GPU and CPU implementations, respectively. These numbers are normalized by the image size. The small difference between the GPU and CPU results is due to the double-precision floating point operations in CPU implementation versus the single-floating point operations in the GPU implementation. Some features such as linear filtering are only supported in single-precision floating operations on GPU.

## V. CONCLUSIONS AND FUTURE WORK

We have proposed and implemented a new FEM-based 3D deformable registration algorithm. The proposed method handles large deformations and is based on voxel intensities; therefore feature extraction or image segmentation are not required in our approach. A GPU-based parallel implementation of the algorithm yielded 37-fold speedup over an optimized CPU implementation for 3D MR-to-MR image registration. Future work will involve single and multi-GPU implementations of the new algorithms for 3D to 2D registration of single and multi-modality medical images.

TABLE II
EXECUTION TIMES

| Steps | GPU (ms)/itr | CPU (ms)/itr | Speed Up |
|---|---|---|---|
| 1 | 2.21 | 892 | 403.61 |
| 2 | 120 | 2797 | 23.3 |
| 3 | 12.7 | 1828 | 143.93 |
| 4 | 0.11 | 4 | 36.36 |
| 5 | 0.65 | 1 | 1.53 |
| GPU others | 10.54 | – | – |
| Total | 146.21 | 5522 | 37.76 |

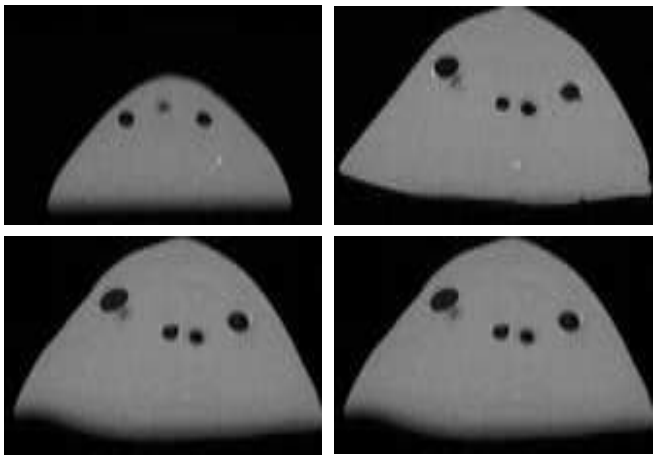


Fig. 4. Visual comparison of GPU and CPU registration. 2D ($y-z$ view) slices of the preoperative (template) image (top left), intraoperative (reference) image (top right), deformed preoperative image after registration on GPU (bottom left) and deformed preoperative image after registration on CPU (bottom right). Note that the phantom has been compressed along x-axis which has caused image elongation in $y-z$ plane and also movements of image features in the image plane.

## REFERENCES

[1] Y. Liu, A. Fedorov, R.Kikinis and N. Chrisochoides, "Real-time non-rigid registration of medical image on a cooperative parallel architecture," *in proc. IEEE Int. Conf. Bioinformatics and Biomedicine*, USA, 2009, pp. 401-404.

[2] T. Rohlfing and C.R. Maurer, "Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts and bees," *IEEE Trans. Information Technology in Biomedicine*, 2003, vol. 7, no. 1, pp. 16-25.

[3] V. Saxena, J. Rohrer and L. Gong, "A parallel GPU algorithm for mutual information based 3D nonrigid image registration," *Lecture Notes in Computer Science, 16th Int. Euro-Par Conf. Parallel Processing*, 2010, vol. 6272, pp. 223-234.

[4] M. Sen, Y. Hemaraj, S. Bhattacharyya and R. Shekhar, "Reconfigurable image registration on FPGA platforms," *in proc. IEEE Conf. Biomedical Circuits and Systems*, 2006, pp. 154-157.

[5] D. Rueckert and P. Aljabar, "Nonrigid registration of medical images: theory, methods and applications," *IEEE Signal Processing*, 2010, vol. 17, pp. 113-119.

[6] A. Köhn, J. Drexl, F. Ritter, M. Knig and H.O. Peitgen, "GPU accelerated image registration in two and three Dimensions," *Bildverarbeitung fr die Medizin*, 2006, Part 3, pp. 261-265.

[7] B. Marami, S. Sirouspour, D. Capson, "Model-Based deformable registration of preoperative 3D to intraoperative low-resolution 3D and 2D sequences of MR Images," *accepted for presentation at MICCAI 2011*, Toronto, Canada.

[8] K.J. Bathe, *Finite Element Procedures*, Englewood Cliffs, NJ, Prentice Hall 1996.

[9] C. Chefd'Hotel, G. Hermosillo, O. Faugeras, "A variational approach to multi-modal image matching," *IEEE Workshop on Variational and Level Set*, 2001, Canada, pp.21-28.

[10] NVIDIA, *CUDA Programming Guide 3.1*, 2010.