Toward Improved Tumor Targeting for Image Guided Neurosurgery with Intra-operative Parametric Search using Distributed and Grid Computing

Andriy Fedorov, Andriy Kot, Yixun Liu, Olivier Clatz, Peter M. Black, Alexandra J. Golby, Ron Kikinis, Nikos Chrisochoides

Abstract Current neurosurgical procedures utilize medical images of various modalities to enable precise location of tumor and critical brain structures for the purposes of planning accurate brain tumor resection. The practical difficulty of using pre-operative images during the surgery is caused by the intra-operative deformation of the brain tissue (brain shift), which introduces discrepancies with respect to the pre-operative configuration. Intra-operative imaging allows tracking of such deformations, but cannot fully substitute for pre-operative data. Non-Rigid Registration (NRR) is a complex time-consuming image processing operation that allows the adjustment of the pre-operative image data to account for intra-operative brain shift. We review computational aspects of a specific method for registering brain MRI to enable its evaluation during image-guided neurosurgery, and consider different strategies for parallelizing this NRR method. We show that the implementation we develop not only allows the delivery of NRR results within the clinical time constraints improving NRR speed, but also provides the potential of improving the accuracy of registration by utilizing distributed Grid resources for distributed search of optimum parameters for the NRR method. In this context, we describe a concept of a dynamic data driven environment for highly distributed non-rigid registration calculations. We present initial results of using national cyberinfrastructure as a platform for such environment, and outline the major challenges in integrating it with the operating rooms of the future.

Andriy Fedorov, Andriy Kot, Yixun Liu, Nikos Chrisochoides
Center for Real-Time Computing, College of William and Mary, Williamsburg, VA USA

Andriy Fedorov, Ron Kikinis, Alexandra J. Golby
Surgical Planning Laboratory, Department of Radiology, Brigham and Women's Hospital, Boston, MA USA

Olivier Clatz
INRIA Sophia Antipolis, France

Alexandra J. Golby, Peter M. Black
Department of Neurosurgery, Brigham and Women's Hospital, Boston, MA USA

1 Introduction

Cancer is one of the leading causes of death both in the USA and around the world. Among the different types of cancer, brain cancer was estimated to claim over 50 thousand new victims in 2008 [1]. Neurosurgical resection is one of the most common and effective treatment options for brain tumor patients. It is crucial that the resection removes as much as possible of the tumor tissue, while maximally preserving the vital structures of the healthy brain. Maximal tumor excision increases time to progression, reduces symptoms and seizures In this Chapter we explore how the concept of Dynamic Data Driven Application Systems (DDDAS) [22], together with the advances in medical image acquisition and distributed computing, can assist in enabling image guidance during neurosurgery and potentially can improve the accuracy of the procedure, allowing more complete tumor resections without additional morbidity.

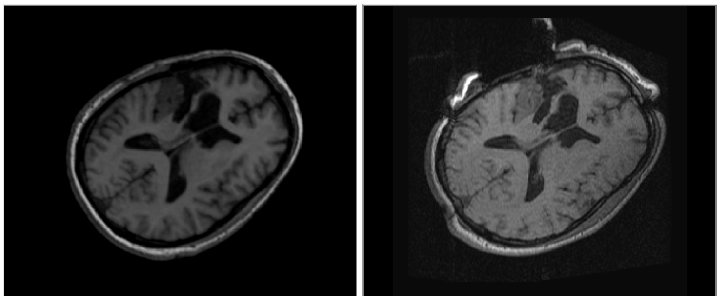

Fig. 1 Intra-operative brain deformation. Left: pre-operative, higher quality image, showing the location of brain tumor. Right: intra-operative image showing brain shift [3].

There are two major challenges in accomplishing the objectives of neurosurgery. First, it is not possible to distinguish between the tumor and non-tumor tissue with the naked eye for certain kinds of tumors. Second, the exact locations of the brain areas that are responsible for the critical brain function, e.g., the motor cortex, are patient-specific, and, again, cannot be identified with the naked eye. This is where medical imaging becomes essential.

Magnetic Resonance Imaging (MRI) is indispensable in demonstrating brain pathologies. Although not distinguishable with the naked eye, neoplastic tissues can be differentiated from brain tissue based on changes in MR signal and corresponding image intensities. MRI has also been shown to be useful in constructing functional mapping of the brain using functional MRI (fMRI) [15]. Both the structural and functional imaging data are used for the purposes of improving the precision of the resection.

Image registration in general is concerned with spatial alignment of corresponding features in two or more images. During image registration, a spatial transformation is applied to one image, which is called *floating*, such that it is brought into

alignment with the *target*, or *reference* position of the object. During rigid image registration, the floating image corresponds to the pre-operative image, which is aligned with the position of the patient using translations and rotations (rigid transformations).

During the course of surgery, opening of the skull and dura causes changes in pressure inside the Intra-Cranial Cavity (ICC). Because of this and other factors, such as drainage of cerebrospinal fluid, induced changes in brain tumor, and the effect of gravity, the brain changes its shape, introducing discrepancies in relation to the pre-operative configuration. *Non-rigid* image registration uses spatially varying transformation to account for this deformation. In general, image registration algorithms are based on optimization of certain similarity criteria between the fixed and floating image under varying parameters of spatial transformation. The complexity of this optimization depends on the number of parameters that describe the transformation. Both rigid and non-rigid registration are open research areas in medical image processing. However, non-rigid registration is a conceptually more difficult problem, which usually requires significant computing resources and time.

Non-rigid registration recovers the deformation of the brain based on the intra-operatively acquired imaging data. Recent advances in medical image acquisition have made it possible to acquire high resolution images, in particular MRI, during the surgery. Intra-operative MRI (iMRI) cannot substitute pre-operative images because of its limited resolution and the high processing time required to obtain functional data. However, iMRI can be used to guide registration of the pre-operative data.

There are three main requirements to non-rigid registration (NRR) [9]. First, NRR should deliver accurate results. Second, the result should be consistently accurate independent of the specific images being registered, and should not be sensitive to small variations in the parameter selection. Finally, a requirement that is specific to IGNS is that registration should meet the time constraints required by the neurosurgical workflow, which is usually 5-10 minutes.

Prospective application of NRR is a dynamic process. iMRI is obtained periodically as requested by the surgeon. Immediately following iMRI, NRR should be used to estimate the deformation of the brain and update the pre-operative images. Usually, hospitals do not have locally available large-scale computational facilities. In this Chapter we describe an infrastructure that enables computation of non-rigid registration using remotely located high performance computing resources, guided by intra-operative image updates.

2 Related Work

The research in NRR for IGNS can be separated into the development of the core registration methods, and design of end-to-end systems, that are capable to support NRR computation and deliver the results intra-operatively. The choice of the NRR method depends mostly on the intra-operative image modality that captures

brain deformation [11]. However, the core computation components of NRR are very similar for different intraoperative imaging modalities.

Registration algorithms are based on optimizing certain similarity measure between the intensities of the reference and floating images. In non-rigid registration, the number of parameters (degrees of freedom) that are being optimized is exceedingly large compared to rigid registration. This contributes significantly to the costs of computing the similarity metric and to the evaluation of gradients required during optimization. However, optimization of the similarity measure alone can lead to unrealistic solutions, since non-rigid registration is an ill-posed problem. Therefore, NRR usually include some form of solution regularization. Biomechanical modeling of the tissue deformation is one of such regularization approaches. Deformation of tissue is usually modeled using the Finite Element Method (FEM) [36], which requires solving a system of equations. The size of this system is proportional to the resolution of the brain biomechanical model.

Timely completion of the core NRR computations is the key component for the efficient end-to-end registration systems. A number of strategies have been proposed to parallelize the time-consuming steps in medical image processing. Christensen and collaborators were some of the first to discuss the use of parallel computing resources for solving time-consuming problems related to brain MRI processing on a massively parallel SIMD architecture [8]. Warfield et al. [32] presented some of the first results in intra-operative processing (segmentation) of iMRI. The authors demonstrate linear speedup of segmentation on a 20-processor workstation, which allows processing of a typical dataset in about 20 seconds. Remarkably, the developed method was subsequently applied and evaluated prospectively during neurosurgeries and liver cryo-ablation procedures [33]. The same group later developed a high-performance method for intra-operative non-rigid registration, which uses linear biomechanical model [31] solved in parallel. Although the authors report clinically acceptable timing results delivered by their implementation, the evaluation was restricted to off-line experimental studies.

Computation of the NRR result within the time constraints of neurosurgery is an essential requirement. In order to facilitate this task, support of the computation on the remote resources may be required. These issues have been recognized by the community, and a number of solutions have been proposed. Stefanescu et al. [29] describe an NRR implementation that is exposed as a web service. Ino et al. developed an end-to-end system for rigid registration computation on a remote cluster [16]. Lippman and Kruggel use a customized grid infrastructure to design an NRR system for IGNS [18].

To the best of our knowledge, none of the systems developed to date was used prospectively during image-guided neurosurgeries. Our approach to the development of such dynamic data-driven NRR system for IGNS (NRR DDDAS) is to adopt an existing NRR method of established accuracy. Next, we parallelize the most time-consuming components of this method, and develop an end-to-end system to facilitate image guidance during neurosurgery.

3 Physics-Based Non-Rigid Registration

The core registration method of our dynamic infrastructure was originally developed by Clatz et al. in [9]. This NRR approach is specifically designed for registering high-resolution pre-operative data with iMRI. The NRR computation consists of preoperative and intra-operative components. Intra-operative processing starts with the acquisition of the first iMRI. However, the *time-critical* part of the intra-operative computation is initiated when a scan showing shift of the brain is available. The basic idea of the registration method is to estimate the *sparse deformation field* that matches similar locations in the image, and then use biomechanical model of brain deformation to discard unrealistic displacements and derive *dense deformation filed* that defines transformation for each point in the image space.

Sparse displacement vectors are obtained at the selected points in the image, where the variability in the intensities in the surrounding region exceeds some threshold. Such *registration* points can be identified prior to the time-critical part of the computation in the floating (pre-operative) image. Once the reference (intra-operative) scan is available, the deformation vector is estimated at each of the selected points by means of block matching. Fixed size rectangular regions (blocks) centered at the registration points are identified in the floating image. Given such a block, we next select a search region (window) in the reference image. The displacement of the block that maximizes intensity-based similarity metric between the image intensities in the block and the overlapping portion of the window corresponds to the vector value of the sparse deformation field at the registration point. The normalized cross correlation (NCC) similarity metric is evaluated as follows:

$$NCC = \frac{\sum_{i \in B}(B_T(i) - \bar{B_T})(B_F(i) - \bar{B_F})}{\sqrt{(B_T(i) - B_T)^2 (B_F - B_F)^2}}.$$

$\bar{B_T}$ and $\bar{B_F}$ correspond to the average intensity values within the block in the reference and floating image respectively. We note the high computational complexity of the block matching procedure. Considering the sizes of three-dimensional block and window are defined in pixels as $B = \{B_x, B_y, B_z\}$ and $W = \{W_x, W_y, W_z\}$, the bound on the number of operations is $O(B_x B_y B_z \times W_x W_y W_z)$ for one registration point.

Estimation of brain deformation is based on the finite element analysis (FEA) using linear elastic model of brain deformation. The finite element mesh of the intracranial volume is constructed from the segmented ICC volume following the methods we evaluated in a separate study [13]. We then iteratively seek such a position of the mesh vertices $\mathbf{U}$ that balances the mechanical forces of the modeled tissue that resist deformation, with the external forces, that correspond to the displacements $\mathbf{D}$ estimated by block matching:

$$\mathbf{F}_i \Leftarrow \mathbf{K}\mathbf{U}_i,\ \mathbf{U}_{i+1} \Leftarrow [\mathbf{K} + \mathbf{H}^T\mathbf{S}\mathbf{H}]^{-1}[\mathbf{H}^T\mathbf{S}\mathbf{D} + \mathbf{F}_i].$$

Here, $\mathbf{K}$ is the mechanical stiffness matrix [10], $\mathbf{H}$ is the interpolation matrix from the mesh vertices to the block matching displacements, $\mathbf{S}$ is the matrix that captures

the confidence in the block matching results. **F** is the force that is increasing between iterations to slowly cancel the influence of the mechanical forces.

Both block matching and iterative estimation of displacements are time critical and should be performed while the surgeons are waiting. Block matching contributes most to the computation costs, because of the exhaustive search for optimum block position. Iterative estimation of mesh vertex displacements based on biomechanical model requires solution of a system of linear equations during each iteration. However, the size of that system is constrained by the number of mesh vertices, which cannot be arbitrarily large due to inherent properties of the NRR algorithm [13].

In the context of the application, we define the *response time* as the time between the acquisition of the intra-operative scan of the deformed tissue and the final visualization of the registered preoperative data on the console in the operating room. These steps performed intra-operatively form the Dynamic Data-Driven Application System steered by the periodic acquisitions of the iMRI data. Our broad objective is to minimize the perceived (end-to-end) response time of the DDDAS component.

4 High Performance DDDAS Infrastructure for Non-Rigid Registration

The baseline code used in the design of the NRR was the implementation developed and evaluated by Clatz et al. [9]. Based on the benchmarking and analysis of this implementation, we identified the following problems:

1. The execution time of the original non-rigid registration code is highly data-dependent. When computed on a high-end 4 CPU workstation, the computation time varies between 30 and 50 minutes. The scalability of the code is poor due to work-load imbalances.
2. The code is designed as a single monolithic component (since it was not evaluated in the intraoperative mode), and a single failure at any point requires restarting the registration from the beginning.
3. The original code is implemented in PVM [5] which is not widely supported as compared to the use of MPI [28] for message passing.

Consequently, we identified the following implementation objectives in the design of the system.

High-performance Develop an efficient and portable software environment for parallel and distributed implementation of real-time non-rigid registration method for both small scale parallel machines and large scale geographically distributed Clusters of Workstations (CoWs). The implementation should be able to work on both dedicated, and time-shared resources.

Quality-of-service (QoS) Provide functionality not only to sustain failure but also to dynamically replace/reallocate faulty resources with new ones during the real-time data acquisition and computation.

Ease-of-use Develop a GUI which automatically will handle exceptions (e.g., faults, resource management, and network outages), and assist in the parameter initialization.

Different strategies can be explored in high performance implementation of the described NRR method. We first explore how this can be done using ubiquitous CoWs. The CoW-based implementation was used prospectively during the recent studies of NRR at BWH [2]. We also describe our recent efforts to further increase the availability of the implementation by developing its components ported on Graphical Processing Units (GPUs) and studying the use of Grid resources.

We develop NRR DDDAS based on the concept of the *computational workflow*. We re-design the core NRR implementation as a coordinated set of processing components that communicate by passing data. Such approach allows to separate time-critical steps, and concentrate on the optimum parallelization strategies for each individual step that requires performance improvement.

4.1 Cluster of Workstations

In the recent years CoWs have become power-plants of ubiquitous computing. Availability of such cluster at the College of William and Mary (CWM, Williamsburg, VA) motivated the development of the implementation of the CoW-based NRR DDDAS. In addition to the dedicated computing cluster, we use the shared resources of a computer lab to boost computing power and reliability of the implemented system. The targeted users of our DDDAS are clinical researches of Brigham and Women's Hospital (BWH, Boston, MA). Our approach is to map the components of the workflow on the computing and communication resources of CWM and BWH, and expose the DDDAS to the clinical researchers by means of a web service interface. The timeline of the interaction with the complete NRR DDDAS is shown in Figure 2. The key component of this system, which requires parallelization, is block matching.

Multi-level distributed block matching In order to find a match for a given block, we need the block center coordinates, and the areas of the fixed and floating images bounded by the block matching window [9]. The fixed and floating images are loaded on each of the processors during the initialization step, as shown in Fig. 2. The total workload is maintained in a *work-pool* data structure. Each item of the work-pool contains the three coordinates of the block center (total number of blocks for a typical dataset is around 100,000), and the best match found for that block (in case the block was processed; otherwise that field is empty).

However, because of the scarce resource availability we have to handle computational clusters that belong to different administrative domains. We address this issue with hierarchical multi-level organization of the computation using master-worker model. A dedicated master node is selected within each cluster. The master maintains a replica of the global work-pool, and is responsible for distributing the work

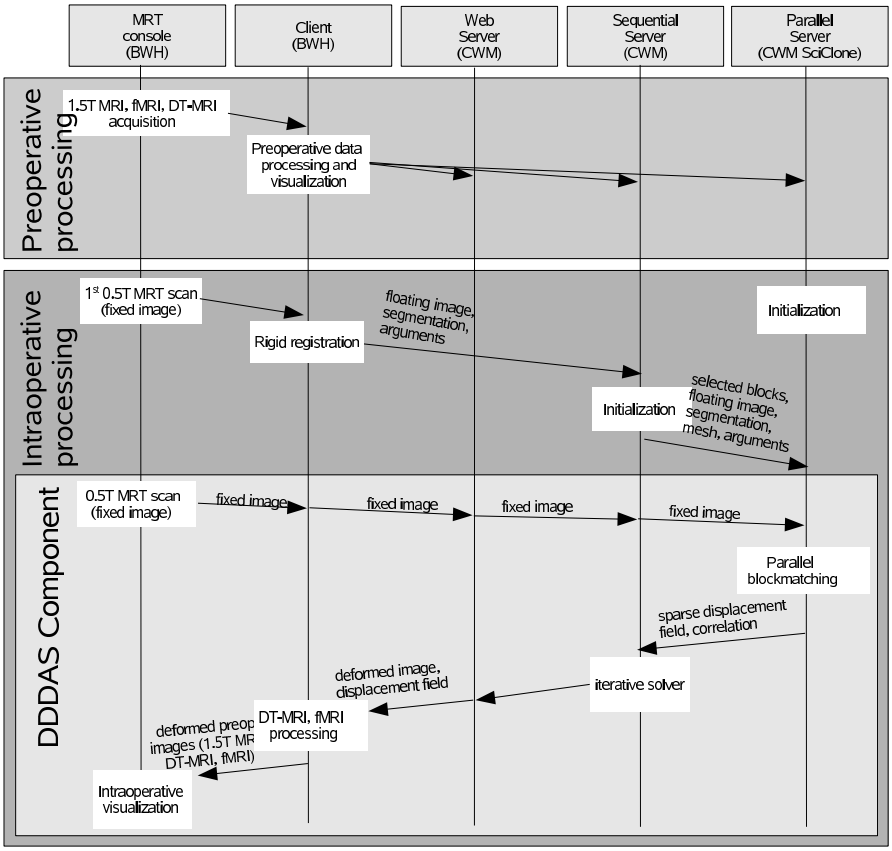

Fig. 2 Timeline of the image processing steps during IGNS (the client is running at BWH, and the server is using multiple clusters at CWM, for fault-tolerance purposes).

according to the requests of the nodes within the assigned cluster, and communicating the execution progress to the other master(s).

Multi-level Dynamic Load Balancing The imbalance of the processing time across different nodes involved in the computation is caused by our inability or difficulty to predict the processing time required per block of data on a given architecture. The main sources of load imbalance are *platform-dependent*. These are caused by the heterogeneous nature of the PEs we use. More importantly, some of the resources may be time-shared by multiple users and applications, which affect the processing time in an unpredictable manner. The (weighted-) static work assignment of any kind is not effective when some of the resources operate in the time-shared mode.

We have implemented a multi-level hierarchical dynamic load balancing scheme for parallel block matching. We use initial rough estimation of the combined computational power of each cluster involved in the computation (based on CPU clock speed) for the weighted partitioning of the work-pool and initial assignment of work.

However, this is a rough "guess" estimation, which is adjusted at runtime using a combination of master/worker and work-stealing [6, 34] methods. Each master maintains an instance of the global work-pool. Initially, all these pools are identical. The portion of the work-pool assigned to a specific cluster is partitioned in meta-blocks (a sequence of blocks), which are passed to the cluster nodes using the master-worker model. As soon as all the matches for a meta-block are computed, they are communicated back to the master, and a new meta-block is requested. In case the portion of the work-pool assigned to a master is processed, the master continues with the "remote" portions of work (i.e., those, initially assigned to other clusters). As soon as the processing of a "remote" meta-block is complete, it is communicated to all the other master nodes to prevent duplicated computation.

Multi-Level Fault Tolerance Our implementation is completely decoupled, which provides the first level of fault tolerance, i.e., if the failure takes place at any of the stages, we can seamlessly restart just the failed phase of the algorithm and recover the computation. The second level of fault tolerance concerns with the parallel block matching phase. It is well-known that the vulnerability of parallel computations to hardware failures increases as we scale the size of the system. We would like to have a robust system which in case of failure would be able to continue the parallel block matching without recomputing results obtained before the failure. This functionality is greatly facilitated by maintaining the previously described work-pool data-structure which is managed by the master nodes.

The work-pool data-structure is replicated on the separate file-systems of these clusters, and has a tuple for each of the block centers. A tuple can be either empty, if the corresponding block has not been processed, or otherwise it contains the three components of the best match for a given block. The work-pool is synchronized periodically between the two clusters, and within each cluster it is updated by the PEs involved. As long as one of the clusters involved in the computation remains operational, we are able to sustain the failure of the other computational side and deliver the registration result.

Ease-of-use The implementation consists of the client and server components. The client is running at the hospital site, and is based on a Web-service, which makes it highly portable and easy to deploy. On the server side, the input data and arguments are transferred to the participating sites. Currently, we have a single server responsible for this task. The computation proceeds using the participating available remote sites to provide the necessary performance and fault-tolerance.

We applied the developed NRR DDDAS for registering seven image datasets acquired at BWH. The computations for two of these seven registration computations were accomplished during the course of surgery (at the College of William and Mary), while the rest of the computations were done retrospectively. All of the intra-operative computations utilized *SciClone* (a heterogeneous cluster of workstations located at CWM, reserved in advance for the registration computation) and the workstations of the student lab (time-shared mode). The details of the hardware configuration can be found in [7]. Data transfer between the networks of CWM and BWH (subnet of Harvard University) are facilitated by the Internet2 backbone network with the slowest link having bandwidth of 2.5 Gbps.

Table 1 Response time (sec) of the intra-surgery part of the CoW-based NRR DDDAS at various stages of development.

Setup	ID						
	1	2	3	4	5	6	7
High-end workstation, using original PVM implementation	1558	1850	2090	2882	2317	2302	3130
SciClone (240 procs), no load-balancing	745	639	595	617	570	550.4	1153
SciClone (240 procs) and CS lab(29 procs), dynamic 2-level load-balancing and fault-tolerance	**30**	**40**	**42**	**37**	**34**	**33**	**35**

The evaluation results are summarized in Table 1. We were able to reduce the total response time to 2 minutes (4 minutes, including the time to transfer the data). We showed, that dynamic load balancing is highly effective in time-shared environment. Modular structure of the implemented code greatly assisted in the overall usability and reliability of the code. The fault-tolerance mechanisms implemented are absolutely essential and introduce a mere 5-10% increase in the execution time.

4.2 Grid Computing Resources

In the last decade, significant effort has been focused on development of the supporting standards and software for Grid computing, deploying production grid systems worldwide and porting applications on those systems. One such production system under continuous improvement and development is USA-based TeraGrid [23]. As of May 2007, TeraGrid was connecting 11 high-end computational sites within the USA, providing "...more than 250 TFLOPS of computing capability and more than 30 petabytes of storage" and therefore making TeraGrid "...the world's largest, most comprehensive distributed cyberinfrastructure for open scientific research" [23]. Currently, TeraGrid connects 11 computational centers providing cumulative peak performance of 1124 teraflops. The capabilities of TeraGrid are continuously growing, providing computational and storage resources otherwise unavailable to any single research institution worldwide.

There are two major advantages of using the Grid infrastructure for NRR DDDAS. First, the implementation is not restricted to run on a specific cluster resource. With the multiple computing centers participating in TeraGrid, temporary resource outages are more feasible to tolerate. Second, complex image processing methods, like NRR, often require proper setting of the large number of parameters in order to achieve optimum accuracy. Identification of such parameter combination is a non-trivial task. One approach to selecting the optimum parameter combination is to use *speculative computation* [16], when multiple instances of NRR are computed in parallel with different parameter settings. In this regard, we have developed initial accuracy assessment solutions [12] to facilitate intra-operative speculative NRR over the Grid. In out Grid NRR DDDAS, we leverage the CoW-based implementa-

tion, augment it with the automatic error estimation, and develop a framework for speculative execution of NRR on the TeraGrid.

While TeraGrid resources can be accessed directly for individual job submission and data transfer, doing this manually on the large scale or as part of workflow execution is not practical. We adopted Swift workflow scripting and management system [35] to implement and deploy NRR workflow. Swift has been developed and evaluated to support grid implementations that are based on Globus Toolkit, which allows to use this system without any modifications to schedule workflows on TeraGrid. SwiftScript, the scripting language used for workflow definition, is a powerful way of abstracting interaction of the processing tasks, which allows to define composite data inputs, dependencies between the processing tasks, and provides familiar control structures like loops and conditional structures, which allow flexible control over workflow definition and execution.

Fault-tolerance and dynamic load-balancing are important characteristics of NRR DDDAS. Swift implements basic fault-tolerance of workflow execution at the individual task level, which is critical for NRR computations. In case a particular task fails to deliver the output, Swift will re-schedule its execution, possibly on a different site. Task-level load-balancing is also provided by the Swift infrastructure. The execution traces for the same computational task are continuously collected and used to dynamically select the best performing site when the task is scheduled again.

Swift provides the means to define and execute the workflow, which consists of individual processing tasks. Each of the processing tasks must be available as an executable at each of the sites, that will be involved in the workflow computation. The details of running a specific task are provided to Swift in the so called *translation catalog* available at the client (submission) site. The translation catalog contains the identifier of the remote site where the executable is installed, together with the optional information on its invocation.

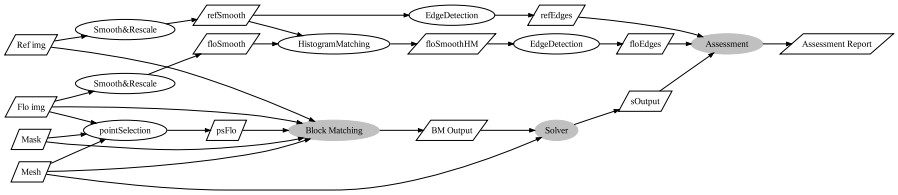

Fig. 3 NRR workflow diagram for single registration execution (shaded are the time-critical components of the workflow).

The NRR workflow diagram of a single NRR procedure together with the accuracy assessment module is shown in Figure 3. The block matching task is parallelized using MPI, and has been deployed on the TeraGrid sites for remote parallel execution. The other components of the workflow are executed on the local resources (single node of the CWM SciClone cluster). This NRR workflow corresponds to the base case for computation supported by our cluster-based implemen-

tation we discussed earlier, augmented with the accuracy assessment module. The accuracy assessment module of the Grid-based NRR DDDAS was developed separately [12]. This module allows to automatically estimate the registration error. The construction of the workflow for speculative execution is straightforward with the scripting capabilities of Swift. This allows us to study the impact of some of the parameters on the registration accuracy.

Table 2 Absolute improvement in accuracy (mm) evaluated at selected landmarks using optimal values of block size and outlier rejection rate.

Case	1	2	3	4	5	6	7	8	9	10	11	12
1	0.2	0.1	0.2	0.2	0.3	0.1	0.3	0.4	0.0	0.2	0.2	0.3
2	0.2	0.0	0.0	0.0	0.2	0.3	0.3	0.2	0.5	0.3	0.1	0.2
3	0.6	1.0	0.2	2.9	0.0	0.1	0.3	–	–	–	–	–
4	0.9	0.8	0.2	0.4	0.7	0.5	0.7	0.4	0.3	0.7	0.7	–
5	0.0	0.3	0.0	0.0	0.8	0.5	0.4	0.0	0.3	0.4	–	–
6	0.2	0.1	0.2	0.1	2.0	0.1	0.0	0.1	–	–	–	–

We considered the impact of varying the block size and outlier rejection rate on the accuracy of NRR on retrospective clinical data. Table 2 summarizes the improvement in accuracy evaluated at the expert-selected anatomical landmarks with the optimum combination of these two parameters, as compared to their default settings. Based on the experimental data, in most cases, good registration accuracy is achieved using the default parameters suggested by Clatz et al. [9]. However, in Case 3 the improvement of registration accuracy was significant. In both cases, however, there were landmark points, where registration error exceeded voxel dimensions. The analyzed data also suggests, that the optimum value of outlier rejection is varied in different locations of the image. For example, if we consider landmarks 5 and 12 in Case 1, the optimal combination of the studied parameters is different in each case, as we show in Figure 4.

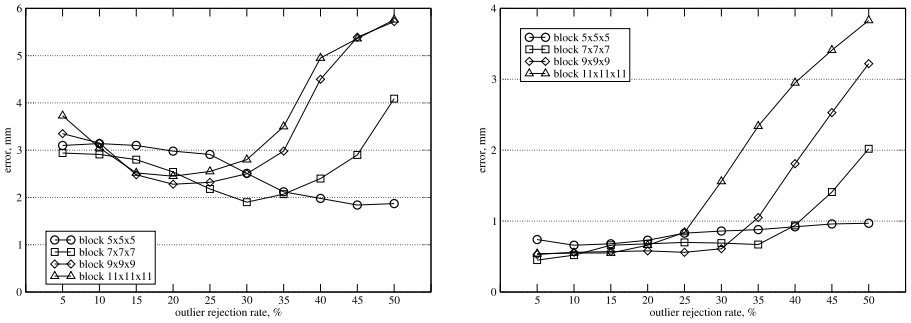

Fig. 4 Influence of the block size and rejection rate on landmark error: case 1, landmark 5 (top), and case 1, landmark 12 (bottom).

4.3 Graphical Processing Units

Graphics Processing Unit (GPU), an inexpensive, single-chip, massively parallel architecture, have shown orders of magnitude higher throughput and performance per dollar than traditional CPUs. In addition, a GPU can be easily deployed in the Operating Room as a co-processor of CPU without hindering routine surgical operations. In recent years, some researchers have made efforts to accelerate NRR using GPU [25, 20, 30, 17]. However, to satisfy the requirement for the accuracy and real-time in clinic a more advanced GPU based NRR is imperative.

The workflow implementation of NRR DDDAS allows us to use best parallelization strategies for individual components. Block matching component is embarrasingly parallel, which makes it highly amenable to GPU parallelization. We use CUDA programming model [24] to develop the GPU implementation of this component. CUDA organizes GPU threads in grid, which is an array of blocks and each block is an array of threads. Kernel is the core code to be executed on each thread, which performs on different sets of data using its ID in a SIMD fashion. CUDA programming model can be treated as two levels loop: block level and thread level. In the following code, the outer loop can be parallelized using GPU on the thread block level. Computation of the similarity metric for an image block (inner loop) is parallelized on GPU thread level, while the similarity metric computation is done on CPU:

1: **for** each image block $fblk$ in floating image **do**
2: define search window *sw* in fixed image
3: **for** each image block $tblk$ in *sw* **do**
4: calculate similarity *s* between $fblk$ and $tblk$
5: **end for**
6: find the maximum *s* and corresponding displacement
7: **end for**

This GPU-based implementation of block matching can gain a speed up of about 10, as we show in Figure 5, compared to CPU. The speed up is measured at seven different image block sizes. Figure 5 clearly shows that GPU running time increases linearly as we increase the block size, but CPU exhibits a super linear behavior.

Optimization of GPU codes is particularly important, since there are numerous parameters of the execution environment, which can affect performance. Significant evidence exists that there can be orders of magnitude performance difference depending on the level of optimization for GPU implementations [4, 26, 27]. The search space generated by the execution configuration is so large that it is not practical to find the optimal parameters by trial-and-error. Several recent studies trackle this problem through empirical search-based approaches [27, 19]. We utilize the method provided in [19] to optimize the GPU execution configuration for block matching and improve speedup further, as shown in Figure 6. We observe speedup of about 30 when comparing optimized and non-optimized implementations.

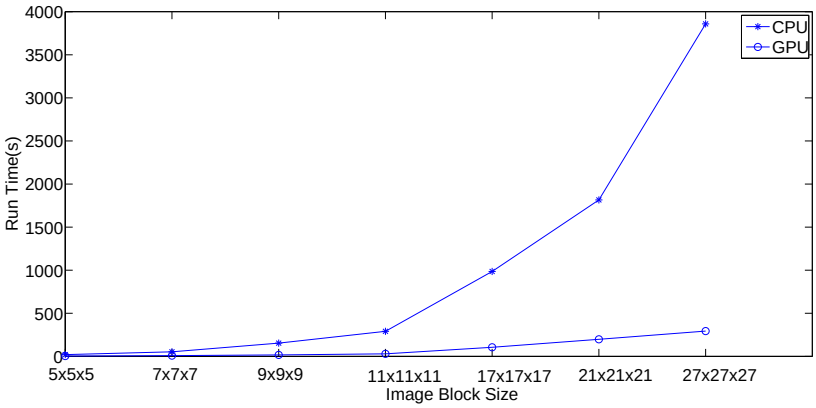

Fig. 5 Parallel block matching execution time on CPU vs GPU.

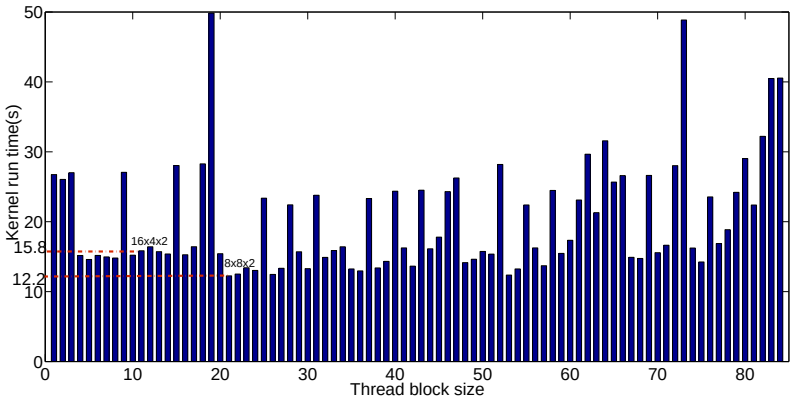

Fig. 6 Effect of GPU thread configuration on the performance of the block matching kernel code.

5 Discussion

We have described the use of Dynamic Data Driven Application Systems for Image Guided Neurosurgery, enabled by the advances in medical image acquisition and parallel/distributed computing. The DDDAS concept, *for the first time ever in clinical practice, helped us to complete and present non-rigid registration results to neuro-surgeons at BWH during tumor resection procedures using image landmark tracking across the entire brain volume.* Using NRR DDDAS we were able to reduce the total response time of the time-critical computation component to about 35 sec, therefore delivering effective speedup of nearly 100, as compared to the original sequential implementation of the code. In order to achieve this, we used remotely (at CWM) several CoWs with the total of 269 processors [7] (the trans-

fer of data from CWM to the operating room in BWH takes between 3 to 5 min). Our preliminary data suggest that we might be able to improve the accuracy of the NRR method by performing speculative execution [14] on the TeraGrid which is capable to deliver about 250 TFLOPS. This computing power translates into tens of thousands of registrations (with different parameters) in almost real-time, if there is a proper coordination with all sides to avoid scheduling conflicts. However, this requires availability of network connection between the operating room and remote computing resources. Also, the imaging data must be anonymized prior to transfer to address the confidentiality concerns.

Our next goal is to meet the real-time constrains of NRR DDDAS using much cheaper hardware which can be located in the operating room. Our preliminary results in the CRTC indicate that it is possible to complete the time critical component of non-rigid registration within a minute —save another 3 to 5 minutes, for the data transfer— using a single (or two, for fault-tolerance) high-end workstations with NVIDIA GeForce 8800 GT GPU and 2 x Intel Core2 Duo CPU 3.16GHz. We believe the use of current and emerging hardware architectures along with the coordinated use of TeraGrid will be the most appropriate platform for NRR DDDAS. Our results show that GPU provides excellent computing capabilities without the need to sacrifice accuracy of the result.

Next generation operating rooms, like Advanced Multi-modality Image Guided Operating (AMIGO) [21] suite, will provide new capabilities to improve intra-operative image guidance. Advances in high performance and distributed tools for image analysis, like the NRR DDDAS we presented in this chapter, will be essential to meet the ever-increasing computational demands of such environments. The use of DDDAS will be critical in health care among other areas, where this concept proved to be successful [22].

Acknowledgments

AF, YL, AK and NC were supported in part by the NSF grants CCS-0719929, CCS-0750901, CCF-0833081, and by the John Simon Guggenheim Memorial Foundation. AF was supported in part by a grant from Brain Science Foundation. RK, AJG and PMB were supported in part by NIH grants U41 RR019703 and P01 CA67165. This research was supported by an allocation through the TeraGrid Advanced Support Program. This work was performed [in part] using computational facilities at the College of William and Mary which were enabled by grants from Sun Microsystems, the National Science Foundation, and Virginia's Commonwealth Technology Research Fund.

References

1. American Brain Tumor Association. Facts & statistics, 2008. `http://www.abta.org/siteFiles/SitePages/5E8399DBEEA8F53CBBBBF212C63AE113.pdf`, accessed 23 Dec 2008.
2. N. Archip, O. Clatz, S. Whalen, D. Kacher, A. Fedorov, A. Kot, N. Chrisochoides, F. Jolesz, A. Golby, P.M. Black, and S.K. Warfield. Non-rigid alignment of preoperative MRI, fMRI, and DT-MRI with intra-operative MRI for enhanced visualization and navigation in image-guided neurosurgery. *NeuroImage*, 35:609–624, 2007.
3. N. Archip and I.-F. Talos. Spl brain tumor resection image dataset, 2007. `http://www.spl.harvard.edu/publications/item/view/541`.
4. M. M. Baskaran, U. Bondhugula, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan. A compiler framework for optimization of affine loop nests for gpgpus. In *ICS08: Proceedings of the 22nd Annual International Conference on Supercomputing*, page 225234, 2008.
5. A. Belguelin, J. Dongarra, A. Geist, R. Manchek, S. Otto, and J. Walpore. Pvm: Experiences, current status, and future direction. In *Supercomputing '93 Proceedings*, pages 765–766, 1993.
6. R. Blumofe, C. Joerg, B. Kuszmaul, C. Leiserson, K. Randall, and Y. Zhou. Cilk: An efficient multithreaded runtime system. In *Proceedings of the 5th Symposium on Principles and Practice of Parallel Programming*, pages 55–69, 1995.
7. N. Chrisochoides, A. Fedorov, A. Kot, N. Archip, P.M. Black, O. Clatz, A. Golby, R. Kikinis, and S.K. Warfield. Toward real-time image guided neurosurgery using distributed and Grid computing. In *Proc. of IEEE/ACM SC06*, 2006.
8. G.E. Christensen, M.I. Miller, M.W. Vannier, and U. Grenander. Individualizing neuroanatomical atlases using a massively parallel computer. *Computer*, 29(1):32–38, 1996.
9. O. Clatz, H. Delingette, I.-F. Talos, A. Golby, R. Kikinis, F. Jolesz, N. Ayache, and S.K. Warfield. Robust non-rigid registration to capture brain shift from intra-operative MRI. *IEEE Trans. Med. Imag.*, 24(11):1417–1427, 2005.
10. H. Delingette and N. Ayache. *Soft tissue modeling for surgery simulation*, volume XII of *Handbook of Numerical Analysis: Special volume: Computational models for the human body*, pages 453–550. Elsevier, Netherlands, 2004.
11. C. DeLorenzo. *Image-Guided Intraoperative Brain Deformation Recovery*. PhD thesis, Yale University, 2007.
12. A. Fedorov, E. Billet, M. Prastawa, A. Radmanesh, G. Gerig, R. Kikinis, S. K. Warfield, and N. Chrisochoides. Evaluation of brain MRI alignment with the robust Hausdorff distance measures. In *Proc. of ISVC 2008*, pages 594–603, 2008.
13. A. Fedorov and N. Chrisochoides. Tetrahedral mesh generation for non-rigid registration of brain MRI: Analysis of the requirements and evaluation of solutions. In *Proc. of the 17th International Meshing Roundtable*, pages 55–72, 2008.
14. A. Fedorov, B. Clifford, S.K. Warfield, R. Kikinis, and N. Chrisochoides. Non-rigid registration for image-guided neurosurgery on the TeraGrid: A case study. Technical Report WM-CS-2009-05, Department of Computer Science, College of William and Mary, 2009.
15. A.J. Golby, R.A. Poldrack, J. Illes, D. Chen, J.E. Desmond, and J.D. Gabrieli. Memory lateralization in medial temporal lobe epilepsy assessed by functional MRI. *Epilepsia*, 43(8):855–863, 2002.
16. F. Ino, K. Ooyama, and K. Hagihara. A data distributed parallel algorithm for nonrigid image registration. *Parallel Computing*, 31(1):19–43, 1 2005.
17. D. Levin, D. Dey, and P. Slomka. Acceleration of 3d, nonlinear warping using standard video graphics hardware: implementation and initial validation. *Comput Med Imaging Graph*, 28(8):471–483, 2004.
18. H. Lippman and F. Kruggel. Quasi-real-time neurosurgery support by MRI processing via grid computing. *Neurosurg. Clin. N. Am.*, 16(1):65–75, 2005.
19. Yixun Liu, Eddy Z. Zhang, and Xipeng Shen. A cross-input adaptive framework for gpu programs optimization. In *23rd IEEE IPDPS*, 2009. accepted.

20. Pinar Muyan-Ozcelik, John D. Owens, Junyi Xia, and Sanjiv S. Samant. Fast deformable registration on the gpu: A cuda implementation of demons. In *ICCSA '08: Proceedings of the 2008 International Conference on Computational Sciences and Its Applications*, pages 223–233, Washington, DC, USA, 2008. IEEE Computer Society.
21. National Center for Image Guided Therapy. Advanced Multimodality Image Guided Operating (AMIGO) Suite, 2009. `http://www.ncigt.org/pages/AMIGO`.
22. National Science Foundation. DDDAS: Dynamic Data Driven Applications Systems, 2009. `http://www.nsf.gov/cise/cns/dddas/DDDAS_Appendix.jsp`.
23. National Science Foundation. The TeraGrid project, 2009. `http://www.teragrid.org/`.
24. NVIDIA Corporation. NVIDIA CUDA programming giude, 2008. Version 2.1.
25. Antonio Ruiz, Manuel Ujaldon, Lee Cooper, and Kun Huang. Non-rigid registration for large sets of microscopic images on graphics processors. *J Sign Process Syst*, 55(1-3):229–250, April 2008.
26. S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu. Optimization principles and application performance evaluation of a multithreaded gpu using cuda. In *PPoPP08:Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, page 7382, 2008.
27. S. Ryoo, C. I. Rodrigues, S. S. Stone, S. S. Baghsorkhi, S. Ueng, J. A. Stratton, and W. W. Hwu. Program optimization space pruning for a multithreaded gpu. In *CGO08: Proceedings of the Sixth Annual IEEE/ACM International Symposium on Code Generation and Optimization,*, page 195204, 2008.
28. M. Snir, S. Otto, S. Huss-Lederman, and D. Walker. *MPI The Complete Reference*. The MIT Press, 1998.
29. R. Stefanescu, X. Pennec, and N. Ayache. Parallel non-rigid registration on a cluster of workstations. In *Proc. of HealthGrid'03*, 2003.
30. C. Vetter, C. Guetter, C. Xu, and R. Westermann. Non-rigid multi-modal registration on the gpu. In *Medical Imaging 2007: Image Processing*, volume 6512, page 651228, 2007.
31. S. K. Warfield, M. Ferrant, X. Gallez, A. Nabavi, and F. A. Jolesz. Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 23, Washington, DC, USA, 2000. IEEE Computer Society.
32. S. K. Warfield, F. A. Jolesz, and R. Kikinis. Real-time image segmentation for image-guided surgery. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–7, Washington, DC, USA, 1998. IEEE Computer Society.
33. S. K. Warfield, A. Nabavi, T. Butz, K. Tuncali, S. G. Silverman, P. Black, F. A. Jolesz, and R. Kikinis. Intraoperative segmentation and nonrigid registration for image guided therapy. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 176–185, London, UK, 2000. Springer-Verlag.
34. I. Wu. *Multilist Scheduling: A New Parallel Programming Model*. PhD thesis, School of Comp. Sci., Carnegie Mellon University, Pittsburg, PA 15213, July 1993.
35. Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde. Swift: fast, reliable, loosely coupled parallel computation. In *Proc. of 2007 IEEE Congress on Services*, pages 199–206, 2007.
36. O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, 2005.