Robust Registration of Multi-modal Images: Towards Real-Time Clinical Applications

Sébastien Ourselin[1,2], Radu Stefanescu[1], and Xavier Pennec[1]

[1] INRIA Sophia-Antipolis, Projet Epidaure, 2004 Route des Lucioles, F-06902 Sophia-Antipolis Cedex
{Radu.Stefanescu, Xavier.Pennec}@sophia.inria.fr
http://www-sop.inria.fr/epidaure/Epidaure-eng.html
[2] CSIRO Telecommunications and Industrial Physics, PO Box 76, Epping NSW 1710 Australia
sebastien.ourselin@csiro.au

Abstract. High performance computing has become a key step to introduce computer tools, like real-time registration, in the medical field. To achieve real-time processing, one usually simplifies and adapts algorithms so that they become application and data specific. This involves designing and programming work for each application, and reduces the generality and robustness of the method. Our goal in this paper is to show that a general registration algorithm can be parallelized on an inexpensive and standard parallel architecture with a mall amount of additional programming work, thus keeping intact the algorithm performance.
For medical applications, we show that a cheap cluster of dual-processor PCs connected by an Ethernet network is a good trade-off between the power and the cost of the parallel platform. Portability, scalability and safety requirements led us to choose OpenMP to program multi-processor machines and MPI to coordinate the different nodes of the cluster. The resulting computation times are very good on small and medium resolution images, and they are still acceptable on high resolution MR images (resp. 19, 45 and 95 seconds on 5 dual-processors Pentium III 933 MHz).

1 Introduction

One major concern in Image-Guided Therapy (IGT) is the simultaneous need for high performance algorithms for planning, targeting, and monitoring, and the time constraints imposed by the operating room [3]. For instance, in neurosurgery, pre-operative guidance using stereotactic systems allows the surgeon to select the best and safest trajectory to penetrate the tissue. This step drastically reduces the surgery time in the operating room. During surgery, the surgeon may use an intra-operative guidance, in order to control his trajectory. However, these image-guided surgery systems are limited by the *static knowledge* of the anatomical brain structures, since cerebrospinal fluid (CSF) leaks or tumor removal deform the anatomical structures [14]. Intra-operative (interventional) imaging is being developed to solve these problems, and also to detect complications during surgery, such as bleeding. Typically, fluoroscopic, sonographic and more recently ultrasound images are used. Concurrently, 3D modalities were developed, such as CT [4] or MRI [13,2] guided interventional procedures.

A typical example is a surgical tracking and visualization system developed at the Brigham and Women's Hospital, based on open-MR image guidance, and used in 45 neurosurgical interventions [1]. During each craniotomy, 3 to 5 intra-operative MR datasets were acquired (with a typical resolution of $256 \times 128 \times 60$) and rigidly registered to the pre-operative image to guide the initial approach. The registration maximizes the mutual information and typically takes five minutes. Since each acquisition lasts from 1 to 5 min, one would like to obtain a registration time of under 1 min to add a minimal overhead to the image acquisition and to remain faster than the medical need.

To decrease the computation time, Netsch *et al.* presented a multi-modal registration algorithm [6] based on local correlation (LC) optimized by a Gauss-Newton technique, using only 10% of image voxels that have the largest local variance. They obtain a registration time of about 1 mn for typical MR and CT images. Nevertheless, the optimization procedure used is not robust to the outliers since it is a least-square minimization [12].

Recently, we presented a multi-modal registration algorithm, also based on a local similarity measure, which explicitly takes into account the presence of outliers [8]. We believe that our algorithm could lead to faster and more robust results while considering more image information. Our goal in this paper is to show that such a registration algorithm can be parallelized on a cheap and standard parallel architecture with a reasonably small amount of additional programming work while keeping intact the algorithm's performance. Other important requirements for the parallel environment are portability, scalability and safety, since the software is intended to be used in a safety-critical environment. Thus, the choice of a mature and well-tested environment is important.

We detail in Section 2 the possible hardware platforms for parallel computing and the chosen software environments, namely OpenMP and MPI (Message Passing Interface). Then, we recall in Section 3 the principles of the registration algorithm, a block matching technique detailed in [9,8], and we detail some improvements. Section 4 focuses on successive parallelizations of the main time consuming steps. Finally, we analyze in Section 5 the gain in computation time with respect to the number of processors and the data size.

2 The Parallel Environment

Hardware Choices. A parallel computer is essentially made of processors, memory, and an *interconnection network* that connects processors between each other and with the memory. In *shared memory computers*, all the processors are connected to the same memory and do not need to explicitly exchange information. However, they are limited to a few processors and in the amount of memory that can be addressed. *Distributed memory computer* are composed of *nodes*, containing processors and memory, and interacting by sending each other messages through the interconnection network. There is virtually no limit on the number of processors but the synchronization and the information exchange takes much longer. Global data structures often have to be replicated on every node for performance reasons, which also leads to larger memory needs.

For medical applications, especially IGT, we believe that a cluster of symmetric multi-processors (SMP), typically PCs, connected by an internal network is a good trade-off between the power and the cost of the parallel platform. The system can be used as a research tool or as an embedded system in a clinical environment. In this article, we used up to 10 PCs of our lab (dual-processors Pentium III 933 MHz), connected by a fast Ethernet Network (100 Mbps using 2 interconnected 1Gbps switches). Such a hardware configuration is already present in many labs and hospitals.

Software Choices. A parallel programming model that seemed appropriate to adapt a sequential algorithm with minimum reprogramming is the Single-Program-Multiple-Data (SPMD) approach, in which all the processors involved execute the same program, but on different data. We used OpenMP to program multi-processor machines and the Message Passing Interface (MPI) to coordinate the different nodes of the cluster. One of their main advantages is the portability over many different kind of SMP clusters: both are standards and do not depend on the machine architecture, operating system, and network topology.

OpenMP is a set of compiler directives and library functions that specifies the behavior of a program when executed on shared memory computers [7]. A large part of the OpenMP C standard is implemented as "pragma" compiler directives. This eases the parallelization of the sequential code and enables a sequential compilation by standard C compilers. The core notions of OpenMP are the *parallel sections* (a piece of code executed by all the processors with shared or replicated variables), and *parallel "`for`" statements* that enable the parallelization of independent iterations of a loop on multiple processors.

MPI is a standard for communication libraries between the nodes of a cluster [5]. Among the most powerful functions, we can `send` a message to a node, `receive` a message from a node, `broadcast` a message to all other nodes, `scatter` subparts of a "list" to different nodes, or `gather` the subparts of a "list". An important difference between MPI and OpenMP is that each MPI process has its own data and variables, as the memory is not shared.

Using MPI, we run a UNIX *process* on each PC of the cluster. Each process uses OpenMP to start one *thread* per processor on its machine. To coordinate the different processes, a *master* process does everything that cannot be done in parallel, such as input/output (I/O) operations and tasks that have be done sequentially. All the other processes are called *slaves*. In our case the master is *not dedicated*, which means that it also can do everything that regular slaves do.

3 The Sequential Algorithm

The algorithm we parallelize in this article computes a parametric transformation (rigid, similarity, or affine) from correspondences between very similar areas in both images, with a block matching strategy. This procedure is extensively described in [9] for rigid registration of anatomical sections, in [8] for multi-modal rigid registration of medical images, and in [10,11] to compute the mid-sagital plane of the brain. In order to quickly approach the desirable optimum and to extend the capture range of the search, a multi-resolution scheme is used. We previously used a coarse to fine strategy for the block sizes. Here, we use a small and constant block size (4^3 voxels) and we subsample the original images by (at most) a factor of two in each axis at each level of the pyramid. At each resolution level, the correspondences are searched using a block matching strategy around the current position. This is done by minimizing the simple but efficient local linear correlation criterion, which is well suited to multi-modal registration [10]. Then, a robust transformation is computed by minimizing the distance between matched points using Least Trimmed Squares (LTS) [12]. This process is iterated until convergence. As we will discuss in Section 2, an acceptable registration can be obtained before the highest resolution.

To further improve the robustness and speed-up the algorithm, we only select points with a high local variance. However, unlike [6], we use all the points for the low level, and we halve the number of relevant voxels at each pyramid level, with a lower bound of 20%. Thus, all the image information is taken into account for large displacements, while the algorithm adaptively focuses on relevant image parts when estimating a more precise motion. The accuracy and the high robustness of the algorithm (100% of the CT/MR registrations within the voxel size) has been validated [10] using the Vanderbilt database [15].

4 Parallel Implementations

A profiling of the sequential version of the algorithm showed that the program spent most of its computation time (about 93%) in the vector field computation. The remaining time was shared between image resampling, LTS minimization, and I/O operations. Hence, our first concern was the vector field computation.

MPI / OpenMP Implementation of the Vector Field Computation. At each iteration, one has to compute the block $\mathcal{B}'_i$ that best matches each selected block $\mathcal{B}_i$, compute the new transformation and resample the image. For the parallel algorithm, each process has a copy of the initial images, and locally resamples it to limit the communication time. The master divides the list of N_{blocks} blocks into N_{procs} sub-lists and `scatter` them to the processes. Each process computes N_{blocks}/N_{procs} block displacements, and returns its local sub-list Sub_i of correspondences to the master (`gathering` step). Then, the master computes the new transformation and broadcasts it to all the slaves. We observed an acceleration of 48% on two processors, which is close to the 50% theoretical gain. We may further reduce the communication time by using OpenMP to drive dual-processors workstations. In practice, this only decreased the necessary amount of memory by avoiding the replication of large floating point images.

OpenMP Implementation of Image Resampling. As the number of processors used becomes high, the resampling of images takes a higher percentage of the total computation time (see Fig. 1). An MPI implementation of the resampling would ask each process to take a part of the image, resample it and then send it to every other process, thus requiring much communication time. With an OpenMP implementation, each process (i.e. machine) does the resampling of

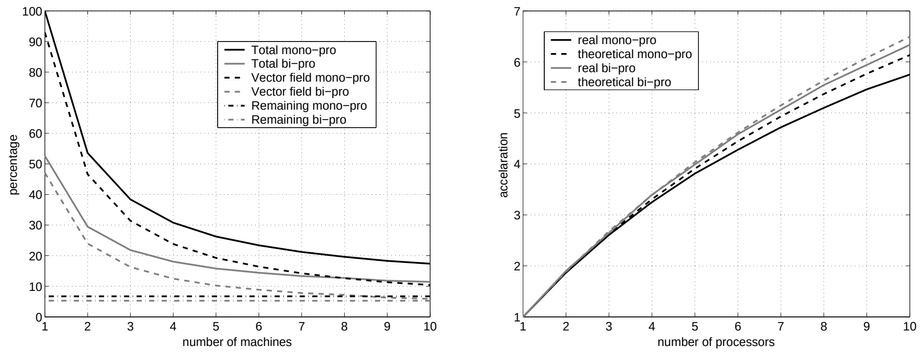

Fig. 1. Left: Computation time of parallel part (vector field computation), sequential part (remaining), and total computation time, using 1 to 10 processors on mono- and dual-pro workstations. **Right:** Speedup of the implementation, normalized by the execution time on a single CPU Pentium III 933 MHz, with the theoretical Amdahl's value.

all the points, but it shares the load among all the processors available on the machine with a "`for`" loop on all the image points. We expected an acceleration of about 50%, and obtained 40%. The difference between theory and experimental results are essentially due to the bounded memory access speed (in our case 133 MHz for 933 MHz Pentium III dual-processors).

5 Experiments

We illustrate our algorithm with two data sets. The first one is an MR/CT multi-modal case from the Vanderbilt database [15]. These low resolution images[1] represent a typical multi-modal registration in a clinical application. To simulate an IGT application, we also used high resolution[2] sets of pre- and post-operative MR images, acquired by La Pitié Salpêtrière Hospital (Paris) in the context of the treatment of Parkinson's disease by deep brain stimulation (DBS).

We used up to 10 PCs of our lab (dual-processors Pentium III 933 MHz), connected by a fast Ethernet Network (100 Mbps). We applied our algorithm with mono-processor and dual-processors architecture independently, from one to ten machines. For each case, we made 100 registrations to estimate the mean time of the vector field computation and the mean time for rest of the program.

Computation Times of the Parallel Section. We present the wallclock time (Fig. 1, left) for a typical registration problem, as a percentage of the computation time with one single CPU workstation. The total computation time is drastically reduced with the first machines (82% of gain for 4 dual-processors), and decreases much slower with each additional machine (only a 10% gain for the next 6 machines). The computation time of the vector field drops from 93% of the total computation time (for a mono-processor machine) to 51% (for a cluster of 10 dual-processors), reaching the constant computation time of the

[1] T1 MR: $256^2 \times 26$ voxels of $1.25^2 \times 4$ mm^3, CT: $512^2 \times 28$ voxels of $0.65^2 \times 4$ mm^3.
[2] T1 MR: $256^2 \times 124$ voxels of size $0.9375^2 \times 1.4$ mm^3.

sequential part. This means that we need to investigate the parallelization of the "constant time operations" to further improve the time performance.

Performance Analysis. (Fig. 1, right) shows the speedup of the parallel implementation w.r.t. the number of CPUs, normalized by the execution time on a single CPU. This experiment was done using 1 to 10 mono-processor machines, and 1 to 10 dual-processors. To evaluate the quality of the performance, we also plot the theoretical speedup factors provided by Amdahl's law: if $\alpha\%$ of the code is sequential, and the remaining code is parallelized on N processors, the maximum speedup factor (with no communication delays) is $S = 1/(\alpha + \frac{1-\alpha}{N})$. Using "constant time" values (Fig. 1, right), we estimated that $\alpha \approx 7\%$ for the mono-processor case, and a slightly inferior value of 6% for dual-processors, due to the OpenMP parallelization of the image resampling. As expected, the measured speedup is higher for dual-processors machines (the sequential part takes a smaller time) but it is also relatively closer to its theoretical curve. This can be explained by a lower communication overhead du to the shared memory (OpenMP part). Therefore, it is definitely cheaper, faster, and more efficient to use dual-processor machines.

Influence of the Data Size. In the previous sections, we were only interested in the speedup factor, but verified on many datasets that the computation time was directly proportional to the volume of the data being registered. We report in the following table the registration times (in seconds) for different cluster configurations and the mean computation time per million of voxels. Even though the computation times are excellent for small image volumes with a small cluster, we still need at least 10 machines for comparable computation times on high resolution images. However, extrapolating our measurements to an intermediate image size typical for IGT($256 \times 256 \times 60$ voxels [1]) gives acceptable computation times for only a few machines.

Data type	Sequential	One dual-pro	5 dual-pro.	10 dual-pro.
Time for 10^6 voxels	72 s	38 s	11.5 s	8.3 s
Vanderbilt (1.7 10^6 voxels)	118 s	63 s	19 s	14 s
Typical IGT (3.9 10^6 voxels)	*280 s*	*150 s*	*45 s*	*33 s*
Salpêtrière (8.1 10^6 voxels)	600 s	316 s	95 s	69 s

Trade-Off between Precision and Computation Time. Even if we finally obtained registration times of the order of one minute, which was our aim for IGT applications, one could think of applications where we need to register higher resolution images and/or reduce the number of machines. In this case, it seems difficult to further improve the computation times without modifying the parallel algorithmic scheme. However, the pyramidal approach used in the algorithm provides a multi-scale transformation estimation where the last level takes about 73% of the total computation time. This suggests that we could drastically speed-up the registration by stopping the algorithm before this level.

To estimate the loss in precision with respect to the pyramid level, we used the transformation T_0 obtained with the high level as the reference (our "ground truth" for this experiment), and we computed the mean localization error of representative points for the transformation T_i at each level i. We report in the following table this relative precision for the three high resolution datasets. As these are pre- and post-operative MR images, there are some important deformations between the images due to the surgical operation and artifacts (distortion due to air-brain interfaces, presence of electrodes...). In all cases, the relative precision of the transformation at level 1 is still below the voxel size, and corresponds to a relative precision of 0.2 mm at the center of the brain. Thus, an optimal trade-off between precision and computation time seems to be obtained by stopping the algorithm at pyramid level 1; we obtain a one minute registration with only two dual-processors machines. With low resolution images (the Vanderbilt database), we obtain even more impressive computation times: from 21 seconds for 1 dual-processors PC to 7 seconds for 5 dual-processors PCs.

	Data set 1	Data set 2	Data set 3
Voxel size	$0.94^2 \times 1.3$ mm	$0.90^2 \times 1.0$ mm	$0.98^2 \times 1.4$ mm
$RMS(T_3)$	3.54 mm	10.7 mm	8.96 mm
$RMS(T_2)$	2.80 mm	1.08 mm	2.60 mm
$RMS(T_1)$	**0.92** mm	**0.68** mm	**0.71** mm
$RMS(T_0)$	reference	reference	reference

6 Discussion and Conclusion

The registration algorithm we chose to parallelize computes at each step a sparse vector field by block matching, which is used to estimate a robust parametric (rigid to affine) transformation using Least-Trimmed-Squares, and embedded in a multi-scale framework [10]. We proposed in this article parallel MPI and OpenMP implementations of the two main time-consuming steps: the vector field computation, with a resulting acceleration of approximately 48% on two processors, and the image resampling, with a speedup of a bit less (40%), mainly due to memory access limitations. The speedup results on more than two processors closely follow the theoretical bound given by Amdahl's law. For the same number of processors, dual-processors machines are cheaper than mono-processor ones, and the parallelization is slightly more efficient (thanks to memory sharing in OpenMP). The computation times themselves are excellent on small images (19 seconds for 5 dual-processors on Vanderbilt images), and still very good on typical intra-operative and high resolution MR images: 45 seconds and 1min35 for 5 dual-processors, 33 and 70 seconds for 10 dual-processors workstations. Thus, we can conclude that a small cluster of dual-processors PCs is an optimal choice for real-time registration in IGT.

One way to further accelerate the algorithm (or reduce the number of machines) is to slightly decrease its accuracy by stopping one level before the end in the multi-scale pyramid: the relative precision of the transformation is still below the voxel size, for only 27% of the total computation time. The registration of high resolution MRI now takes only 1 min on a cluster of only 2 dual-processors PCs. With lower resolution images (the Vanderbilt database), we can even obtain a registration time of 15 seconds.

The results obtained in this article, both in terms of computation time for registration and methodology for parallelization, open research avenues for performing huge computational tasks on large medical image databases, such as the quantification of disease evolution on a large number of patients or information retrieval and exploration in large image collections.

Acknowledgments: This work was done during the PhD Thesis of S. Ourselin in the Epidaure Project, INRIA Sophia-Antipolis. The authors would like to thank Pr. D. Dormont from the Neurological Dpt. of La Pitié Salpêtrière Hospital (Paris) for the high resolution MR images. We would also like to thank Briony Doyle and Bhautik Joshi for proofreading of this article.

References

1. D.T. Gering, A. Nabavi, R. Kikinis, N. Hata, L.J. O'Donnell, E.L. Grimson, F.A. Jolesz, P.M. Black, and W.M. Wells. An integrated visualization system for surgical planning and guidance using image fusion and open MR. *JMRI*, 13:967–975, 2001.
2. D.F. Kasher, S.E. Maier, H. Mamata, Y. Mamata, A. Nabavi, and F.A. Jolesz. Motion robust imaging for continuous intraoperative MRI. *JMRI*, 13:158–161, 2001.
3. R. Kikinis. IGT: Today and tomorrow. Technical Report 192, Surgical Planing Laboratory, Brigham and Women's Hospital, Boston, MA, USA, 2000.
4. L.D. Lunsford, R. Parrish, and L. Albright. Intraoperative imaging with a therapeutic computed tomographic scanner. *Neurosurgery*, 15:559–561, 1984.
5. Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, May 1995. http://www.mpi-forum.org/.
6. T. Netsch, A. Van Muiswinkel, and J. Weese. Towards real-time multi-modality 3D medical image registration. In *Proc. of ICCV'01*, pages 718–725, 2001.
7. OpenMP Architecture Review Board. *OpenMP C and C++ Application Program Interface Version 1.0*, October 1998.
8. S. Ourselin, A. Roche, S. Prima, and N. Ayache. Block Matching: A General Framework to Improve Robustness of Rigid Registration of Medical Images. In *Proc. of MICCAI'00*, pages 557–566, Pittsburgh, Penn. USA, October 11-14 2000.
9. S. Ourselin, A. Roche, G. Subsol, X. Pennec, and N. Ayache. Reconstructing a 3D Structure from Serial Histological Sections. *Im. Vis. Comp.*, 19(1-2):25–31, 2001.
10. Sébastien Ourselin. *Recalage d'images médicales par appariement de régions. Application à la construction d'atlas histologiques 3D.* PhD thesis, Nice, France, Jan. 2002. http://www-sop.inria.fr/epidaure/BIBLIO/Author/OURSELIN-S.html.
11. S. Prima, S. Ourselin, and N. Ayache. Computation of the Mid-Sagittal Plane in 3D Brain Images. *IEEE Transactions on Medical Imaging*, In Press.
12. Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection.* Wiley Series in Probability and Mathematical Statistics, first edition, 1987.
13. J. Schenk, F. Jolesz, and P. Roemer. Superconducting open-configuration MR imaging system for image-guided therapy. *Radiology*, 195:805–814, 1995.
14. S.K. Warfield, M. Ferrant, X. Gallez, A. Nabavi, F.A. Jolesz, and R. Kikinis. Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery. In *SC 2000: High Performance Networking and Computing Conf.*, volume 230, pages 1–16, Dallas, USA, Nov 4–10 2000. SPL Technical Report 188.
15. J. West and al. Comparison and evaluation of retrospective intermodality brain image registration techniques. *J. of Comp. Assist. Tomography*, 21:554–566, 1997.