A Grid Service for the Interactive Use of a Parallel Non-Rigid Registration Algorithm of Medical Images

Radu Stefanescu, Xavier Pennec, Nicholas Ayache
{Radu.Stefanescu,Xavier.Pennec,Nicholas.Ayache}@sophia.inria.fr
INRIA Sophia, Epidaure, 2004 Rte des Lucioles, BP93, F-06902 Sophia-Antipolis Cedex, France

Preprint. To appear in the special HealthGrid 2004 issue of Methods of Information in Medicine

Summary

Objective: The goal of this work is to improve the usability of a non-rigid registration software for medical images.

Method: In order to use the interactivity of a visualization workstation and the computing power of a cluster, we have build a registration grid service. On the user side, the system is composed of a graphical interface that interacts in a complex and fluid manner with the registration software running on a remote cluster.

Conclusion: Although the transmission of images back and forth between the computer running the user interface and the cluster running the registration service adds to the total registration time, it provides a user-friendly way of using the registration software without heavy infrastructure investments in hospitals. The system exhibits good performances even if the user is connected to the grid service through a low throughput network such as a wireless network interface or ADSL.

Keywords:

grid service, non-rigid registration, cluster, medical imaging.

1. Introduction

Non-rigid registration recovers the deformation between two images. As an example, Figures (1a) and (1b) present two-dimensional slices of three-dimensional magnetic resonance images (MRI) of the brain coming from different subjects. The algorithm recovers a deformation, presented in Fig. (1c), that morphs one of the images (Fig.1a) into an image (Fig.1d) that is as close as possible to the other (Fig.1b). Since the two brains are not identical, this correspondence can not be perfect. Moreover, it is necessary that the recovered deformation preserve the anatomic coherence of the objects in the images: the deformation has to be smooth and invertible.

This non-rigid registration procedure is necessary to compare and/or fuse images in many domains of medical imaging, ranging from building and usage of medical atlases [1] to pre-operative planning [2]. Generally, non-rigid registration is a computationally intensive, and hence time-consuming procedure.

In order for registration to be really used in clinical environments, physicians need software tools that are precise and robust. Furthermore, care must be taken regarding the usability of such tools. One primary issue is the execution time, which should be lower than a few minutes. However, since financial pressure makes it easier for healthcare organizations to invest in medical equipment than in high-performance computing, the software should not

require hardware that is too expensive to be acquired by a hospital (more expensive than medical equipment), or that is too hard to administrate.

In a previous work [3], we developed a non-rigid registration algorithm able to reliably recover large deformations and that admits an efficient parallel implementation on a cluster of personal computers. The algorithm solves a two-part partial differential equation: if we denote by $U=(u_1,u_2,u_3)$ the deformation that links the source image J to the target image I , then the first term minimizes the square distance $\|I(p)-J(p+U(p))\|^2$ between the target image and the deformed source image. The second term is based on two assumptions: the deformation is supposed to be regular, and the degree of regularity spatially varies according to the deformability of the imaged objects. Hence, the *regularity term* has the form of a diffusion term, where the conductivity d represents the local ability to deform. The two terms are weighted by a *confidence factor k(p)* that estimates at each point the reliability of the matching process:

$$\frac{\partial u_i}{\partial t} = \sum_{p\in\Omega(images)} \left[k\nabla\|I(p)-J(p+U(p))\|^2 + (1-k)\nabla(d\nabla u_i) \right]$$

The parallel implementation of this algorithm, using the Message Passing Interface (MPI) and a cluster of personal computers, lowers the registration process time for typical 256x256x124 MR images to less than five minutes on a cluster of 15 2GHz Pentium IV PC's linked by a 1GB/s Ethernet network.

Though accurate, robust and fast, our registration software suffers from several issues regarding its usability. First, it is command line based, which means that once the algorithm is started on the two input images, there is no mean to *interact* with the algorithm until it ends. Desirable interactions are: stopping a registration process either because it is about to diverge or because the level of convergence is sufficient for the user's needs; modifying parameters; visualizing the intermediate result. In the near future, we would also like to enable the user to add some medical expertise in order to guide the registration process. All this requires the creation of a *graphical user interface* (GUI) that allows the user to master the registration software.

Our system is composed of two interacting modules: the registration software and the graphics user interface. The registration module performs the computation. It is run on a parallel computer that usually needs special conditions to operate, such as air-conditioned rooms. Therefore, such a machine cannot lie next to the user and is more likely to find a suitable place as a shared resource in a data center than in an operating or pre-operative planning room. The GUI provides the interaction between the registration module and the user. It also deals with the visualization of the three dimensional images involved. For visualization performance reasons, the GUI must be executed on the user's computer. The computers running the two components of our system must be connected through a network. Due to the different conditions in which these two computers must operate, such a network may be a long distance one. Moreover, it may be difficult to install additional wires in the operating room for intra-operative registration. Thus, we want to enable our service to run through a wireless, hence low bandwidth, network interface.

The purpose of our work is to implement the registration as a *grid service* able to connect the clinically useful user interface to a computational center. This interaction has to be sufficiently refined to allow the user to dynamically manipulate the state of the registration software.

A previous work on a non-rigid registration service was done by Ino *et al.* [4], who used a high-end cluster connected to the visualization workstation through a high-bandwidth Wide Area Network (WAN). We believe that the infrastructure these authors used is not representative of the one available in most hospitals. Lower end to average clusters and low bandwidth WAN's are in our opinion more realistic hardware platforms for any medical grid service. Moreover, this paper does not address the interactive usability issue that we believe essential in a clinical environment.

In this paper, we present a system that enables the interactive use of parallel registration software running on a remotely located cluster. The presentation emphasizes the mechanisms used to connect our user interface to the grid service, and the security and usability concerns that are raised by such a system.

2. Objectives

To summarize, we want to build a registration grid service implemented on a cluster of PC's on one side, and a graphical interface able to control it on the user side. In order to connect them, we need a communication library that fulfills several conditions:

1) The communication library has to be as flexible as possible. If the users makes two requests A and B to the GUI, and request A is transmitted to the grid service before request B, it is important that no constraint be imposed to the responses of the two request. The response to B may arrive before or after the response to A, and one or both responses may be never sent at all.

2) Communications have to pass through firewalls.

3) The access to the grid service should be secured. A public key authentication such as RSA fulfills our needs regarding access permissions to the grid service.

4) Communications should be encrypted. Since confidential medical data is transmitted through a WAN that is not always under the hospital's full control, this data should be encrypted.

5) The whole system should preserve the anonymity constraints imposed by current regulations [5]. Image files usually contain information about the subject's identity. This information should never make its way to the non-controlled WAN. The solution is to send only the data necessary for the registration: the image sizes and the image intensity data.

6) In order to maximize the accessibility to the grid service, the communication library should spare bandwidth. Therefore, each message transmitted through the WAN should be compressed before sending.

3. Technological choice

During the recent years, a vast amount of work has been invested in distributed computing environments providing standardized and secure communications between remotely located computers. We review below some of these environments in view of our application.

3.1. Client/server

Several standardized communication methods between a client and a server have emerged. Recently, older standards, such as RPC and CORBA, have been replaced by protocols based on XML, which ensure the interoperability between implementations. The

SOAP protocol, used by Web Services (WS) and the Open Grid Services Infrastructure (OGSI) [6,7] has lately become the communication standard for client/server distributed systems.

Generally, the server interface is described in a dedicated interface description language (IDL in CORBA, WSDL in Web Services, GWSDL in OGSI), and the communication interface is generated automatically. The main advantage of such a system is that, when the client sends a service request to the server, the identification of the request and the decoding of its parameters is done in a transparent manner. We feel that several aspects of client/server systems makes them unsuited for our task:

First, a client cannot simultaneously communicate with more than one server. This means that when the server is a parallel software, the client would have to communicate exclusively with the master node. However, when the master node receives a request, the identification of the request is done in a transparent manner, outside the user's control. Thus, further action would have to be taken in order to forward the request type to the other nodes. From this point of view, client/server would not bring additional functionality to the system.

The client/server system has another major drawback in our case. In this model, the server's sole purpose is to answer requests from the client. It cannot send requests to the client, and it remains idle between processing two successive requests. In our system, this constraint imposed on the server is too hard.

3.2. Message passing

This model, used by basic networking protocols (TCP/IP) and low level grid layers (Globus I/O), and generally adopted by computation libraries (MPI, PVM), provides flexible high-performance communications. In TCP/IP, the user has to explicitly describe the data transfers and the message encoding. Higher level libraries (MPI, PVM) facilitate the message encoding and also provide collective communications (e.g. broadcast). However, these libraries were designed for non-secured communication between the nodes of a parallel computer. Furthermore, they are generally unable to pass through firewalls.

We prefer the message passing communication model, since it does not impose any constraints on the possible actions of the service. Our goal is to create a message passing library that can provide secured communications and is compatible with firewalls.

4. Method

We designed a message passing library that is able to transmit messages back and forth between the GUI and each of the nodes running the grid service. Thanks to compression and encryption, the transmission of messages is fast and secure.

4.1. Communications

In our program, we distinguish between computation messages (exclusively exchanged between the nodes of the cluster) and control messages (exchanged between the parallel program and the GUI). Computation messages are directly communicated using MPI's point-to-point or collective communication subroutines, and they do not concern this article. Control messages are programmed as C++ classes containing a *message tag* that identifies the message, and a *message handler* describing the action to execute upon reception. Each message has two methods, *pack* and *unpack*, that describe the way the message can be packed into a buffer for sending, or unpacked from a received buffer.

Control messages are sent through *communication channels*, that encapsulate the communication protocols. They are C++ classes that provide:

- packing and unpacking basic data types (integer and real numbers and arrays) into buffers. These functions are called by the message's pack and unpack procedures upon the sending and the receipt of the message.
- sending and receiving the packed buffer.

Until now, we have implemented two channels:

- The *long distance channel*, currently based on TCP/IP, is used for communicating between the GUI and the master node. For performance reasons, this channel transparently compresses the messages before sending. It also takes care of the endianism difference that might exist between the machines running the GUI and the cluster.
- The *local channel*, based on MPI, is used to forward the messages received by the master node from the GUI to the other nodes of the cluster.

The communication of a message using a channel is done as follows: the message tag and the message itself are packed into a buffer of the appropriate size and then sent. Upon receipt, the message tag is decoded and an object of the appropriate message class is instantiated. The message is unpacked and its associated message handler is invoked.

In the case where the GUI sends a message to the cluster, the message is first sent to the master node using the long distance channel, and then distributed by the master node to the entire cluster using the local channel (see Figure 2). The message handler is subsequently invoked in parallel by all the nodes of the cluster. The sending of a message by the service to the GUI is performed by the master node of the cluster.

The architecture we have presented here has the advantage of being modular. Adding message types or communication channel can be easily done by inheriting the appropriate C++ class.

4.2. Security issues

Both the GUI and the grid service are isolated from the rest of the world by firewalls. Regarding the long distance channel, three security issues are important:

- Communications must be able to pass through the two firewalls.
- The user must be authenticated before being allowed to use the service.
- Communications should be encrypted.

We use *ssh tunneling* in order to fulfill these three requirements. If a ssh tunnel is not created the registration service is not visible by the user. In order to create the ssh tunnel, the user has to authenticate using a user name and a password delivered by the service administrator. The tunnel also provides the encryption of the data flow between the GUI and the service.

Despite all these security mechanisms, a possible anonymity breach can come from the facial reconstruction that can be done on some MRI images. The solution would be to modify the images before transmission through the network in a way that does not affect registration but makes facial reconstruction impossible. We have not tackled this problem.

5. Results

The final system is the following: After user authentication, the GUI enables the loading and comparative visualization of the images to register. A user interface button enables the user to contact the grid service and start the registration. During the process, the user is informed in real time about the status of the algorithm (small size messages). An intermediate result image (several megabytes of data) is regularly sent to the GUI which displays it. At any time the user has the option to abort the registration. Upon termination of the algorithm, the registration result is sent back to the GUI.

We have tested our system by running the GUI on a 600MHz Pentium III laptop equipped with a wireless network interface. The grid service runs on 15 2GHz Pentium IV PC's linked through a 1GB/s Ethernet network and protected by a firewall running on a 2GHz Pentium IV PC. All systems run the Linux operating system. The resolution of the input images is 256x256x124, and their size is 47.2MB. After compression, they reach a size of only 3.3MB. The transmission of the input images takes 39 seconds, including the additional compression and decompression times. For comparison, if compression is not used, the transmission takes 2 minutes and 26 seconds. During the execution of the algorithm, each update of the intermediate result takes 18 seconds. The extra time took by the transmission of images through the network is of about 2 minutes, thus bringing the total registration time to about 7 minutes. However, thanks to the user interface and the grid service, the registration software becomes usable in the clinical environment. Thus, we believe that the two extra minutes in computation time are tolerable.

We also tested our system in a worst case scenario by using a home Asymmetric DSL connection between the user interface and the grid service. The maximum transfer rate is 16kB/s from the GUI to the service and 64kB/s from the service to the GUI. The image upload time is about 3min30, and the result update time is about 1min30. In this extreme case, the total registration time reaches ten minutes, including image transfer time. However, this proves that our non-rigid registration service can be used from almost everywhere with a very modest equipment.

6. Conclusion and future work

We have presented a system that enables a clinician to transparently use a computationally intensive but powerful non-rigid registration algorithm run by a parallel computer physically located at a large distance. The system combines the speed and precision provided by the parallel computer to the ease of use of the physician's workstation. A graphics interface enables the user to supervise the execution of the algorithm in real time. In the future, we will modify the system so that physicians will be able to use their clinical expertise to guide the registration. We will add an integrated authentication, the automatic creation of the ssh tunnel and the remote execution of the service by the user. A test in a clinical setup is also in preparation.

Our non-rigid registration service opens up numerous advanced medical image analysis applications to the clinical practice, such as the usage of brain atlases or image guided therapy. However, while medical imaging algorithmic issues have mostly been dealt with, grid management problems still have to be addressed. The most important problem is the standardization. Indeed, while the long distance TCP/IP channel fulfills its functionality requirements, its integration with standard security and authentication mechanisms and discovery services is not ensured. These aspects have been addressed in existing grid environments, such as the Globus Toolkit. In the future, we intend to replace the TCP/IP-

based long distance channel with one based on Globus I/O. This would also enable us to replace SSL certificates and SSH tunneling with the similar but more standardized functionality provided by the Globus Security Infrastructure.

References

[1] DLG. Hill, JV. Hajnal, D. Rueckert, SM. Smith, T. Hartkens, and K. McKleish. **A Dynamic Brain Atlas.** In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, Tokyo, pp. 532--539, September 2002. Springer.

[2] S.K. Warfield, F. Talos, A. Tei, et al. **Real-Time Registration of Volumetric Brain MRI by Biomechanical Simulation of Deformation during Image Guided Neurosurgery**. Computing and Visualization in Science, Springer, 5:3-11, 2002.

[3] R. Stefanescu, X. Pennec, and N. Ayache. **Grid Enabled Non-Rigid Registration with a Dense Transformation and A Priori Information**. In *Proc. of , Medical Image Computing and Computer-Assisted Intervention(MICCAI'03)*, LNCS, November 2003. Springer Verlag.

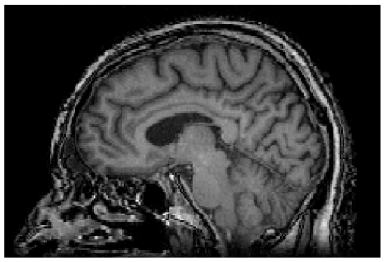
[4] F. Ino, K. Ooyama, Y. Kawasaki, et al. **A High Performance Computing Service over the Internet for Nonrigid Image Registration**. In *Proceedings of Computer Assisted Radiology and Surgery 17th International Congress and Exhibition (CARS 2003)*, International Congress Series 1256, Elsevier Science, pp. 193--199, London, UK, June 2003.

[5] JAM. Herveg, and Y. Poullet. **Directive 95/46 and the Use of Grid Technologies in the Healthcare Sector: Selected Legal Issues.** In Sofie Norager, editor, *Proc. of HealthGrid'03*, Lyon, January 2003. European Commission, DG Information Society.

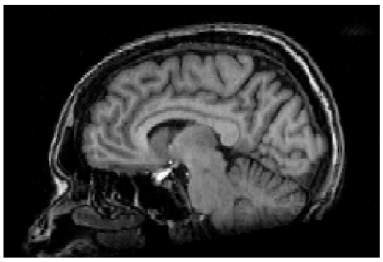
[6] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling. **Open Grid Services Infrastructure (OGSI) Version 1.0**. Global Grid Forum Draft Recommendation, 6/27/2003.

[7] I. Foster, C. Kesselman, J. Nick, S. Tuecke. **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration**. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

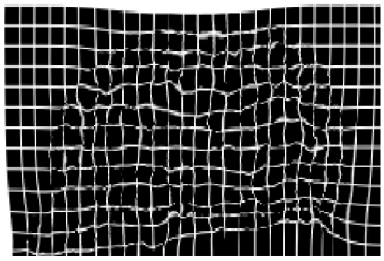
Figure 1: *The algorithm recovers the deformation (c) that morphs the 3D source image (a) into the 3D target image (b). The morphing result (d) should be as close a possible to the target (b).*



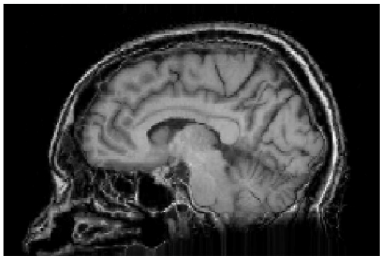
(a) Source image



(b) Target image



(c) Estimated deformation transforming the source into the target



(d) Transformation result

Figure 2: *A message sent by the GUI is compressed, encrypted and sent through the ssh tunnel to the master node, which broadcasts it to all nodes of the parallel computer running the registration service.*