Distributed Modular Computer−Integrated Surgical Robotic Systems: Architecture for Intelligent Object Distribution

Oliver Schorr[1,2], Nobuhiko Hata[1], Andrew Bzostek[3], Rajesh Kumar[3], Catherina Burghart[2], Russel H. Taylor[3], Ron Kikinis[1]

[1]Surgical Planning Laboratory,
Brigham and Women´s Hospital and Harvard Medical School
Boston, MA, USA

[2]Institute for Process Control and Robotics
University of Karlsruhe, Germany

[3]Department of Computer Science
Johns Hopkins University, Baltimore, Maryland, USA

Abstract

This paper presents intelligent object distribution architecture to maximize the performance and intelligence of a distributed surgical robotics system and its preliminary implementation in an MR−guided surgical robot system in an open−configuration MRI scanner. The method enables networked integration of a robot control server and multiple clients with minimum engineering overhead but maximum flexibility and performance. The clients in this study include an intraoperative imager, high−performance image processing computer(s), and surgical navigation host. The first contribution of the paper is to propose the use of object distribution by common object request broker architecture (CORBA), in which a robot control object on the robot control server can be remotely but transparently invoked from the clients regardless of their hardware, operating systems, or programming language. Second, we propose a technique to achieve additional flexibility by reporting the robot configuration information, i.e. geometry and kinematics of the robot, to the clients upon connection. Third, we ensure protection against an unauthorized entity by introducing a security control host that authorized the clients´ access to the robot server. In a prototype implementation of an MR−guided surgical robot system, the robot was controlled by surgical navigation software (the 3D Slicer) on a UNIX client by invoking the distributed control object on a robot control server on a PC. The method was evaluated in performance studies; and the result indicated 3.6 milliseconds for retrieving positions of the robot stages and 25.5 milliseconds to send a frame−based motion command, which are satisfactory for surgical robot control. In conclusion, the proposed method shows the potential usefulness of flexibly integrating the legacy software to a surgical robot system with minimum engineering overhead, thereby achieving highly complex and intelligent tasks in robot−assisted surgery.

1

2 Introduction

We have been trying to build an open architecture system for a computer−integrated robot surgery system, which can integrate component technologies into systems targeted at a variety of surgical tasks. This study aims to link surgical robot technology with medical image processing and surgical navigation technology with minimum overhead so that complex and intelligent tasks can be carried out by surgical robot assistance.

The software and hardware for surgical robot systems are developed on a per−clinical−application basis, thus limiting the flexibility of the hardware and software for reuse in multiple clinical applications. These systems usually rely on a generic computer (most likely a PC) to manage all the tasks, thereby limiting their functional expandability. For instance, real−time elastic registration can hardly be incorporated into a PC−based robot controller due to the requirement for computational heavy tasks. To overcome these problems, we present our approach of modular software architecture and networked system distribution in the associated paper [1]. The approach separates the monolithic system into an intra−operative imager host, a user interface host, and heavy computing hosts, so that each host can focus on dedicated tasks while collaboratively communicating through the network. For instance, the robot host (PC) continuously monitors actuators and sensors though real−time motion control hardware, while the user interface host (graphics workstation) interactively visualizes the status of the robot. By selecting the component hosts, one can flexibly adjust the system to application−specific requirements with minimum engineering overhead. Similar concepts can be found in intelligent manufacturing [2, 3].

While we highly appreciated the modularity and flexibility of the distributed systems in our previous studies, we also realized that the complexity of the communication using message−based commands is relatively limited. In addition, the content of commands was still designed on a per−configuration basis, and protocol definition depended on the programming language, network transport, and other factors. Therefore, the cost for updating and synchronizing a command receptor was still significant.

In order to overcome these shortcomings, we propose a method to achieve highly complex and intelligent communication within the distributed system. Specifically, the method provides the following new features: (1) the use of an object distribution mechanism by common object request broker architecture (CORBA) [4] to implement the software objects solely on the robot control server, while enabling clients to calling them in a transparent manner, (2) flexible robot configuration reporting mechanism to flexibly and automatically adjust the visualization and image−processing clients to various surgical robots; (3) a safety control mechanism to ensure security, manageability, and status reporting in the distributed system. The method is implemented and evaluated in a test−bed system, MR−guided surgical robot, in which a PC−controlled 5DOF MR−compatible robot is controlled and monitored by graphic surgical navigation and simulation software 3D Slicer on UNIX workstations.

This study is clinically significant, as it allows a surgical robot system to accomplish highly complex tasks controlled by intraoperative images,

computationally heavy image processing, and graphics visualization. The method is significant technically because it achieves its clinical goal with minimum cost but maximum flexibility. The use of CORBA enables high interoperability of multiple computing hosts over different OS, hardware, and software language platforms, which encourage bringing in expertise from various research institutions by removing the barrier of software and hardware incompatibility.

3 Methods

3.1 System overview

In this study, we attempted prototype system development within the framework of an MR−guided surgical robot system[5]. The robot control objects are developed in a PC−based control server and distributed to a visualization client in a UNIX workstation. The security−monitoring host in the center of the system monitors connection to the robot control server. Other clients may be high−performance computers processing computationally heavy tasks such as elastic registration.

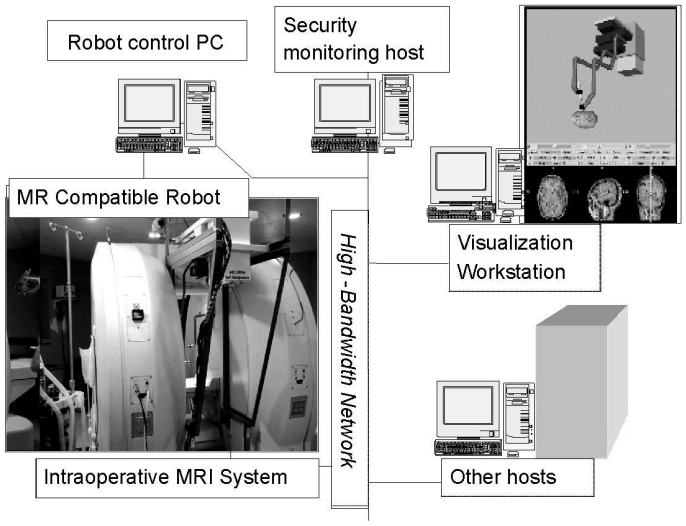

Figure 1: Schematic overview of the system

MR−compatible robot server

The center of the MR−guided surgical robot system is a MR−compatible surgical manipulator, referred as the MR/T robot hereafter, developed jointly by Mechanical Engineering Laboratory of Japan and our institution [5]. The MR/T robot has four horizontal stages and one vertical stage, each driven by ultrasonic actuators. The four stages move two arms independently, enabling the end−effector held by the two arms to have 5 degree−of−freedom DOF movements.

The ultrasonic actuators and linear optical encoders in the MR/T robot are controlled by a PC (CPU Cyrix 300 MHZ, 64MB, Windows NT4.0) using a motion control board (PCX/DSP, Motion Engineering Inc., CA). The card provides on−board

PID control by a 40−MHz Analog Devices ADSP−2105 processor. Software for the robot control is built upon a modular robot control (MRC) C/C++ library providing Cartesian control, forward and inverse kinematics, and joint control interfaces for the MR/T robot. The MRC also includes implementation of a force sensor, and digital and analog input/output device interfaces; thus, expansion of the functionality of the robot can be effectively achieved with minimum cost. The library classes have a layered structure, each new layer inheriting significant functionality from its parents. Further detail on the MRC library including its modularity design can be found in [1].

Intraoperatve MRI scanner

The MR/T robot is designed to perform surgical assistance in an open−configuration intraoperative MRI (IMRI) scanner (Signa SP, GE Medical Systems, Milwaukee, WI)[6], using intraoperative MRI for monitoring and navigation. The host computer for the IMRI is a UNIX workstation (Sparc 20, Sun Microsystems, Moutainview, CA) in which a software offers a service to export real−time IMRI through an Ethernet port. In our system, the 3D Slicer described below establishes the socket connection with the IMRI front−end computer and communicates with the IMRI host to transfer the images.

Visualization client with the 3D Slicer

The 3D Slicer is an interactive surgical navigation and planning software on the UNIX workstation (Ultra30, Sun Microsystems, Moutainview, CA.) It was originally developed for neurosurgical navigation with 3D interactive graphics display. We also utilize its unique functionalities such as automatic multimodality image registration, semi−automatic segmentation, generation of 3D surface models, and quantitative analysis of various medical scans. The Slicer is coded in the Tcl/Tk [7] version of the Visualization Tool Kit [8]. In this study the role of the 3D Slicer also involves displaying the robot to simulate its motion path with respect to patient anatomy. The robot status including the position of all the stages and end−effector is continuously updated through CORBA−based robot control object invocation.

Security monitoring client and other clients

Another significant computer is the security−monitoring host on a UNIX workstation (Ultra80, Sun Microsystems, Moutainview, CA). We can also include multiple hosts for other monitoring and computing purposes, such as elastic registration for updating pre−operative images based on IMRI, which captures the deformed shaped of the target organs, or radiation planning server for planning radioactive seed planting sites for brachytherapy.

3.2 Object distribution by CORBA

Selected robot control objects for MRC are distributed to the 3D Slicer by a middleware MICO [4] which is a freely available and fully compliant implementation of the CORBA standard. MICO intercepts robot control call and finds an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its

programming language, its OS, or any other system aspects that are not part of an object's interface. The MICO provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems. : illustrates the principle of this object distribution mechanism.

Figure 2: Schematic view of the object distribution mechanism

Objects invoked from the 3D Slicer at the robot PC server are robot hardware and software initialization, frame−level movement command, joint−level movement command, and joint−level position report. CORBA defines object interfaces via implementation language−independent hardware−independent specification, using the interface definition language (IDL). The IDL also involves definition of complex data types. In practice, MICO´s IDL compiler writes "wrapper" code that translates between the standardized bus ˆORB core˜ and the objects residing in the CORBA−interface which redirects method calls inside the MRC library.

Another unique feature of our object distribution mechanism is the Dynamic Invocation Interface (DII). Rather than hard coding the handle in the 3D Slicer using the prefixed standard IDL stub, the DII lets the 3D Slicer dynamically incorporate the interface description sent from the robot server and map it to Tcl commands.

We employed TclMico [9] that plugs into MICO´s DII converts the IDL to Tcl script which can be plugged into any Tcl application at runtime. Since the 3D Slicer is coded by script language Tcl/Tk, which does not require compilation, this dynamic invocation mechanism is suitable for the 3D Slicer. In the same manner as DII, Dynamic Skeleton Interface (DSI) allows the robot server to declare accessible objects at runtime instead of fixed IDL skeleton.

3.3 Flexible robot configuration report

Minimizing the robot−dependent information in the client but managing the information in the robot control server and transferring it to the clients is suitable for keeping the flexibility of the client software. It also helps to centralize the maintenance of the robot in the robot server and thus reduce the cost of updating the robot configuration. Therefore, we employed a mechanism that maintains the configuration file of the robot in the robot control server and sends it to the clients upon establishment of the connection.

This configuration file was originally designed for the MRC library to accommodate various configuration settings of the robots. The MRC´s configuration file has authentication or identification block (designating a name, kinematics, and joint controller), followed by one kinematics and joint control block per joint of the robot. The joint block has moving range for each joint and gain parameters for actuators´ PID control.

In addition to parameters for joint control, the configuration file has simple kinematic and geometric descriptions of the current robot. This robot geometry is intercepted by the 3D Slicer to generate a graphic model of the robot (). The geometry model is broken into the simple elements (cylinder, box, and ball) and connected by joints (linear, rotational joints, slide or ball joints) around arbitrary axes.

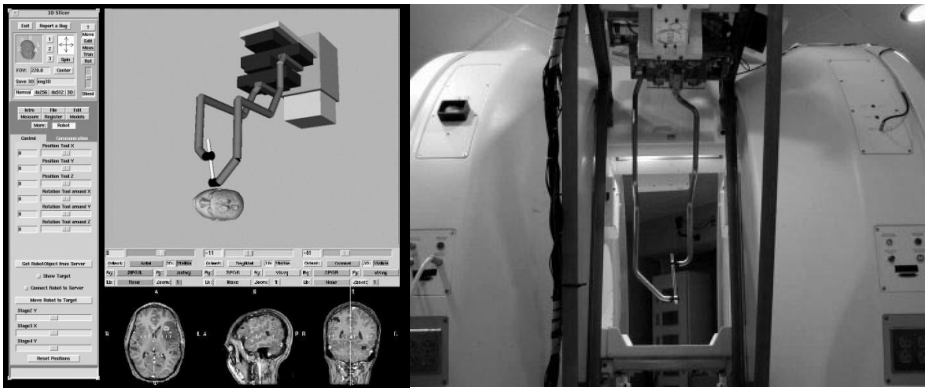

Figure 3: Integrated display of robot and anatomy data from the interventional MR scanner (left) and the robot installed in the scanner. By using object distribution (Sec− tion 2.1) and configuration distribution (Section 2.2), the 3D Slicer can control the MR−compatible robot (right) with minimum implementation cost., thus enabling the highly intelligent robot control to be coordinated with original functionalities of the 3D Slicer, i.e. image processing, navigation, image retrieval from the IMRI scanner, and more.

3.4 Safety−monitoring host

The safety−monitoring host implements a subset of three different tasks dedicated to increase safety and simplify communication initialization. The first task is to supervise the client´s access to robot server by registering the client hosts and authorize the of passive and active control privilege to the robot control. For

instance, MR/T robot limits the access to one client with active control privilege, while unlimited number of hosts can access with passive control privilege. The grounds for this access limit is that the decision for control should be finalized at the centralized in the single authorized client server, while the other client hosts should be devoted to data processor. The second task of the safety−monitoring host is to store Tcl files, interface description files, and configuration files from the robot control server, which can be later transferred to the clients upon connection. Some of the information in these files should be sent to the client when the client ˆgreetsˇ the client but before actual CORBA connection happens.

4 Performance test

In order to evaluate the feasibility of the method, the processing time of the prototype system was measured for the elected tasks. (Task 1) is position inquiry of five joints by one, two or three passive clients, and (Task 2) is Cartesian−level motion command of the end−effector by an active client. Task 2 involves the inverse kinematics computation in the robot control server. (Task 1) and (Task 2) were invoked 100 times from 0, 1, or 5 3D Slicer clients dedicated for (Task 1), and one client for (Task 2). Each client was connected to the 100Mbit/second fast switcher, yet was not isolated from the rest of the hospital network. summarizes the result from the tests.

Table 1: Processing time for invoking distributed objects

# of Task 1 clienents	# of Task 2 clients	Average/STD of Task 1 time (micorosecnds)	Average/STD of Task 2 time (micorosecnds)
1	0	3373/520	N/A
2	0	3636/639	N/A
5	0	3412/399	N/A
0	1	3585/304	24764/1780
1	1	3894/637	24954/1101
5	1	3678/384	26784/1502

Though it doesn´t appear in , we also found several exceptionally long processing time in a few per 100 invocations. Those processes took about 40 milliseconds for (Task 1) and 105 milliseconds for (Task 2), which are 5 to 10 times longer than average processing times.

5 Discussions

The result of the performance test indicated the feasibility of the method in computer−integrated surgical robot systems. It is conceivable that the processing of 3.6 milliseconds for passive position inquiry and 25.5 milliseconds for active motion command are acceptable for robot control using the motion control card with on−

board PID control chip. The result also indicates that the number of clients accessing the robot control server does not influence the processing time.

As of the publication of this report, only the robot control objects are distributed by CORBA. Images are transferred from the IMRI scanner to the 3D Slicer through a socket connection and commanded by prefixed ASCII text, but the plan is to convert the service CORBA−based object distribution so that the image processor and other hosts can access the scanner with more complex command. An example of such task is image scanning with scanning parameter and sequence.

Added value of the object distribution using CORBA is the ability to incorporate advanced image processing software, which, due to software and hardware incompatibility, used to require significant engineering overhead to port into robot control. The currently standarized CORBA has an interface to virtually all programming language including C, C++ , Smalltalk, Java, Ada, Cobol, Visual Basic, Tcl, PL/1, LISP, Python and Perl. In our testbed prototype, we linked C/C++ with Tcl each running on a different hardware and OS platform. There is talk of making IDL compliant with XML; thus, it may be beneficial to design robot configuration description and patient−specific image and model entry to have synergy with XML to prompt the smooth transition to future XML−compliant database management systems.

6 Conclusion

We reported a method to distribute objects from robot control server to multiple clients to achieve highly intelligent and complex task by the collaborative work of automonous visualization, navigation, and image processing hosts. The method was implemented MR−guided surgical robot system in which robot control objects are transparently called from the graphical navigation software on UNIX workstation, and executed C/C++−based robot control software on PC. The evaluation test indicated the distributed objects can be remotely invocated in 3.6 milliseconds for passive monitoring and 25.5 milliseconds for active control, both of which are satisfactory for surgical robot control. The result indicated that the method is feasible to apply in distributed modular computer−integrated surgical robot systems.

7 Acknowledgement

This study is supported by National Science Foundation´s Engineering Research Center (grant #EEC9731478) and NAC grant P41. The authors gratefully acknowledge the support from Dr. Chinzei of MEL for supporting MRI−compatible robot, and Mr. Gering of MIT for supporting the 3D Slicer. We also appreciate the technical contribution of Mr. Daniel Kacher of Brigham and Women´s Hospital.

8 References

[1] A. Bzostek, R. Kumar, N. Hata, O. Schorr, R. Kikinis, and R. H. Taylor, ˆDistributed Modular Computer−Integrated Surgical Robotic Systems: Implementation using modular software and networked systems,˜ presented at Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery, 2000.

[2] M. Lei, X. H. Yang, M. M. Tseng, and S. Z. Yang, ˆA CORBA−based

agent−driven design for distributed intelligent manufacturing systems,˜ *Jounal of Intelligent Manufacturing*, vol. 9, pp. 457−465, 1998.

[3] O. A. Suarez, J. L. A. Foronda, and F. M. Abreu, ˆStandard based framework for the development of manufacturing control systems,˜ *International Journal of Computer Integrated Manufacturing*, vol. 11, pp. 401−415, 1998.

[4] A. Puder and K. Römer, *MICO: An Open Source CORBA Implementation*. San Francisco, CA: Morgan Kaufmann Publishers, 2000.

[5] K. Chinzei, R. Kikinis, and F. A. Jolesz, ˆMR Compatibility of Mechatronic Devices: Design Criteria,˜ presented at Medical Image Computing and Computer Assisted Intervention, Cambridge, UK, 1999.

[6] J. F. Schenck, F. A. Jolesz, P. B. Roemer, H. E. Cline, W. E. Lorensen, R. Kikinis, S. G. Silverman, C. J. Hardy, W. D. Barber, E. T. Laskaris, and et al., ˆSuperconducting open−configuration MR imaging system for image−guided therapy,˜ *Radiology*, vol. 195, pp. 805−14, 1995.

[7] J. K. Ousterhout, *Tcl and the Tk Toolkit*. Massachusetts: Addison−wesley, 1994.

[8] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit. An object−oriented approach to 3D graphics*. Upper Saddle River, New Jersey: Prentice Hall PTR, 1996.

[9] F. Pilhofer, ˆTclMico˜, http://www.informatik.uni−frankfurt.de/~fp/Tcl/tclmico, 1999.