Distributed Modular Computer-Integrated Surgical Robotic Systems:

Implementation using Modular Software and Networked Systems

Andrew Bzostek[1,4], Rajesh Kumar[1,4], Nobuhiko Hata[2,4], Oliver Schorr[2,3,4], Ron Kikinis[2,4], Russell H. Taylor[1,4]

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

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

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

[4]Engineering Research Center for
Computer Integrated Surgical Systems and Technology

Abstract. We seek to build CIS research systems within a flexible, open architecture. In this paper, we outline our solutions to the problems of system design, construction, and integration in this environment: building distributed, modular systems for sensing, control, and processing. Based on our experience building these systems, we utilize distributed network architectures, modular software components, and intelligent object distribution to maximize flexibility while complying with the particular needs and interfaces of specific components. While a work in progress, these approaches have been integrated into several systems in development and have demonstrated significant utility, based on construction time and system flexibility. This paper will discuss the first two aspects: modular software and basic network architectures, while intelligent object distribution is presented in a companion paper. Based on the applications we have targeted and technologies we have used, this paper introduces our architecture and its components. It then presents the system's current state and finally, discusses plans for future improvements and extensions.

1 Introduction

One of the goals of our groups is to build an open architecture for CIS research, which will allow the integration of a variety of components into systems targeting various surgical applications. Software modularity and distribution of components across a standard network are two cornerstones of our work. Using modular architectures has allowed: **reuse** of research, **speed-up** of application development, and **encapsulation** of proprietary technologies. Distribution of system components on standard networks has: improved **computational capabilities** and **robustness**, as well as facilitating the support of **specialized devices**.

Based on concrete system examples, this paper presents efforts within the Center for Computer Integrated Surgical Systems and Technologies to design and implement components which use these concepts. It focuses on two major aspects of our architecture: modular software components and a basic system distribution. Presented in a companion paper [1] are our efforts to build an intelligent object distribution architecture. While at varying levels of maturity, these techniques are complementary and their integration into systems has demonstrably reduced system implementation overhead and increased flexibility.

1.1 Examples

1.1.1 Steady Hand Cooperative Manipulation

Cooperative ("Steady Hand") manipulation offers an attractive alternative to the popular master-slave teleoperation systems. Our LARS [2] test bed robots are equipped, and the "steady hand" robots [3] designed to explore this approach.

The current primary application being investigated is therapy delivery in the eye to treat vein occlusion, though future applications include other ophthalmic applications, as well as a variety of ENT and spinal surgeries. Our modular robot control (MRC) library implements control for both the LARS and Steady hand robots, as well as integrating low-level sensing.

1.1.2 Fluoroscopy-Guided Robotic Needle Placement

Under development by one of our groups, this system is designed to place needles into soft tissue organs (see fig. 1), initially targeting hepatic tumors for localized therapy delivery [4, 5]. It uses fluoroscopic imaging for guidance. Originally developed on the LARS [2] robotic platform, it currently uses the CART(Constrained Access Robotic Therapy) System [4] for manipulation. Its configuration integrates the robot and imaging, as well as force sensing and visualization. Like the steady hand system, the system's robotic platform has changed as the system has evolved. In contrast, however, further significant changes in this component are anticipated. Also unlike the previous system, this approach has required the integration of imaging and a significant visualization capability. The modular construction of our robot software has allowed this system to move seamlessly between a variety of robots. Distributing the system components across a standard network has not only separated robot control from the visualization and image processing, but has also allowed the robot and imaging host computers to be specialized for their respective devices.

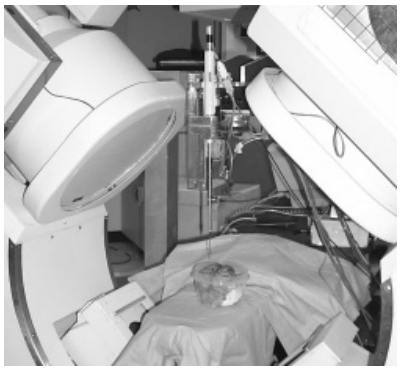

Figure 1. The Fluoroscopy-Guided Robot Needle Placement system.

1.1.3 IMRI-Guided Robot-Assisted Needle Placement

An intraoperative MRI (IMRI)-guided robotic system is in development, currently targeting prostate brachytherapy needle guidance. It is discussed more thoroughly elsewhere [1, 6]. Like the previous example, this system includes a robot, an imager, and a visualization system. Integrated into the IMRI system [7] is also a tracking device which provides position information. Because of the system can acquire volumes during the procedure, a separate high-performance planning host is also used. To support these,

we have extended our modular software and distributed system architectures to support trackers and to include distributed planners as well as physical devices such as robots and imagers.

1.2 Our Approach

In general, modularity and distribution are desirable in any engineered software system. Development of CIS Systems is a particularly iterative, application driven process, however. System specifications and targets are often short to medium-term and tied fairly closely to the particular task at hand. While modularity and distribution are worthy goals in abstract, their associated overhead can outweigh their benefit when only applied to isolated cases. Our approach has been to find commonality among our systems, and then try to implement this commonality reusably in second-generation systems. We've found that only after building systems which address a number of often different targets, using a variety of technologies, have we found useful patterns.

1.3 Previous Work

Complex sensing and manipulation systems are becoming common in improving surgical outcomes. These include navigation aids, including Northern Digital's Polaris optical tracking Systems and ISG's viewing wand arm-based localizer; imaging from a variety of vendors; and robotic systems such as Aesop and Zeus from Computer Motion, daVinci from Intuitive Surgical, and Robodoc from Integrated Surgical Systems. Several research groups are working on systems for a variety of clinical applications. These include Davies [8-10], and Troccaz [11], Peshkin[12], Salcudean [13], Das [14], and Sheridan [15]

As these systems evolve the need for building modular architectures is also becoming evident. Modular architectures have been proposed for industrial automation [16, 17], and integrated, propriety frameworks, such as Picker's Venue system and Surgical Navigation Network's system for component integration are becoming available.

Some open systems, such as VRPN from UNC, have also been developed, though for different domains and thus with different design specifications.

1.4 Components

Our systems use a variety of technologies to address our clinical targets. These include, but certainly are not limited to robots, imagers, trackers, and computational modeling.

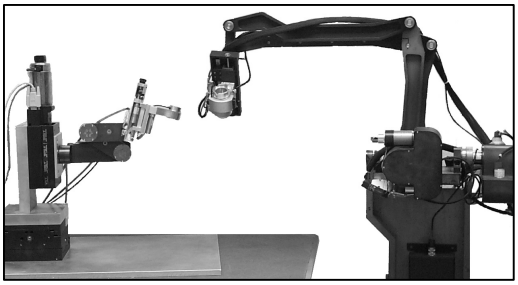

Figure 2. LARS *(right)* and the Steady Hand robot *(left)*: two generations of robots for CIS Research

Robots: Our first generation systems for cooperative manipulation and fluoroscopically-guided needle placement were built on the LARS platform [2] (see fig. 2). Next generation robots developed within the center include the "steady hand robot," [3] designed for cooperative manipulation; the CART (Constrained Access Robotic Therapy) Robot [4], a platform for needle placement and experimental orthopedic work; and the RCM-PAKY System [18] designed for needle

insertion for renal surgery. We are also using the MRT robot, designed for an open-configuration MR scanner, for needle guidance [19].

Within the CISST Engineering Research Center, there are also several new robotic systems suitable for particular imaging/manipulation environments are under development.

Imagers: One of our groups has focused on iMRI-guidance, the other has focused on fluoroscopy. Intraoperative MR techniques hold significant promise, [6, 20], providing unique capabilities in terms of intraoperative planning and monitoring of a procedure. Real-time fluoroscopy provides a more light-weight, real-time imaging solution. Additionally, other groups within the CISST center have now begun to use CT [21] and ultrasound as well.

Trackers: We have developed systems based on three commercially available tracking systems: Northern Digital's Optotrak, Ascension's Flock of Birds, and Image Guided Technology's Pixsys system. Integration of NDI's Polaris as well as other tracking technologies is in the initial phases.

Computational Modeling: While significant computation takes place on the computers within the OR, for very demanding tasks, it is necessary to move this work to a more powerful platform. This is particularly relevant when intraoperative dosemetry and planning is required, e.g. in IMRI-guided localized therapy. A variety of software packages, both commercial and noncommercial, are available for many of these tasks. Our systems currently use non-commercial, research packages, but this will certainly change as applications diversify, and systems move toward clinical use.

2 Overview

In this paper, we present two categories of architectures for dealing with these components: Modular software abstractions, and network distribution of devices and processing.

2.1 Modular Software

In using commercially available systems, the lack of standards for interfaces poses a major difficulty. When using experimental components, similar capabilities are required for control or processing among a class of devices. We resolve the first problem by abstracting programming interfaces common to a devices class. For the second, the control for classes of components are abstracted in layered structures (see fig. 3). These two techniques complement each other. Vendor or hardware-specific software is often available at levels below our general device abstractions. The appropriate layers can then be implemented around this software, and the remaining layers of our architecture are shielded from vendor implementations. We follow the same approach for our robots, trackers, planners, and to a lesser extent, general sensors.

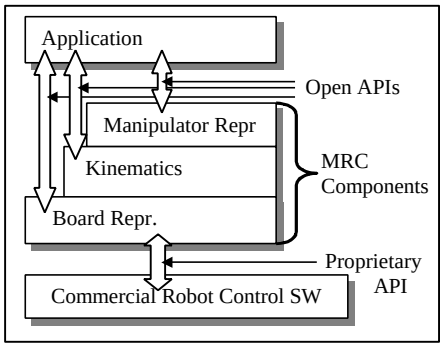

Figure 3. A portion of the modular architecture within the Modular Robot Control (MRC) Library

2.2 Networked System Distribution

Monolithic systems suffer from problems of fault-sensitivity and lack of scalability in terms of computation, component integration, and system monitoring. Additionally, components of our systems have often had hardware or software-related platform restrictions. A local, OR network of computers solves many of these problems.

2.2.1 General Architecture

Our general networked system architecture uses several PC-class machines connected via a standard Ethernet LAN, though, in cases where more significant computational power is needed, it can be extended to a larger class of machines located more remotely. Within the OR, different machines are directly connected to (and responsible for) various other components (e.g. robots or trackers), though more than one component may be connected to the same host machine if architectures/processing allow. In general, we then call the host connected directly to device X, the X Server. Other hosts which want to use X's capabilities will then connect to the X Server as X Clients (see fig. 4). In addition to hosts for controlling application(s), clients can also include additional hosts for safety monitoring and data collection.

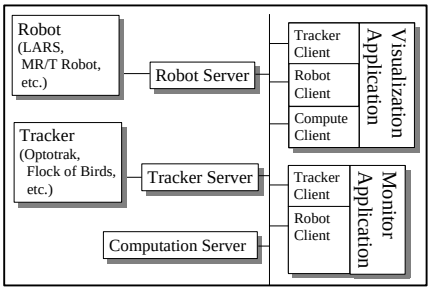

Figure 4. A prototypical implementation of our network architecture

2.2.2 Requirements

The general architecture is not specific to a particular software implementation. Potential implementations include simple message-based communication via TCP or UDP, RCP style invocation, and distributed objects (e.g. via CORBA).

Any of these, however, will need to provide certain capabilities, and can be selected based on how well they meet these needs.

Transparent Distribution: In order to maintain maximum software flexibility the network implementation needs to make the distribution as transparent as possible. A client should not have to differentiate local vs. remote devices and should have to know as little as possible about their actual locations.

Network Performance: The network implementation should provide not only adequate bandwidth, but should minimize latency as well. Ensured quality of service will also be increasingly important as systems become more distributed.

Synchronous &. Asynchronous Communication: Depending on the distributed device and its mode of use, primary communication can be either synchronous or asynchronous. An implementation should allow for either and for their mixing within an interface.

Multi-Priority Client Support: A network implementation must thus provide the ability to not only support multiple clients, but support different levels of functionality among them. For example, a robot server should support only a single controlling client while still allowing multiple clients to connect as observers.

Safety: Although system distribution can enhance safety by adding redundancy and monitoring capabilities to the system, the components necessary to achieve this and to ensure that the distributed system maintains a high overall level of safety are not yet

developed. A network implementation should at least facilitate, and ideally enhance these capabilities

3 Component Implementations

We have taken these patterns and implemented them into several different components.

3.1 Robot Control

We have chosen to abstract programming interfaces for our robots. The alternatives to this approach, including developing/using programming languages poses problems of learning curve and user acceptance. Implementations of interfaces in acceptable programming languages provide libraries that can be shared, swapped, and developed independently of each other. Furthermore, it allows the programming language to be changed, while preserving the interfaces. Our modular robot control (MRC) library provides interfaces for all of our test bed robots. The interface can be easily extended include additional manipulators and allows high level control independent of the robotic platform actually implementing the commands.

The MRC structure abstracts interfaces in layers. The lowest layers wrap hardware implementations (vendor specific sensor implementations, servo control for robots etc). This functionality is then used by logical abstractions (joints for robots, sensor interface for all sensors). The higher layers contain functionality for Cartesian control, command scheduling, and network support. This layering allows new robots and new configurations of existing robots to be quickly supported, often requiring only a change in run-time loadable configuration.

The library provides Cartesian control, forward and inverse kinematics, and joint control interfaces for our robots. It also includes implementations of force sensor, and digital and analog i/o device interfaces.

Our first implementations of robot control used remote procedure calls for network communication. However this is cumbersome, and not platform independent. The current implementation includes remote execution functionality as a part of the class interface. This also allows us to separate network implementations (socket style TCP or UDP or direct cable serial/parallel connections) from the programming interface for robot control. CORBA, an emerging standard for distributed networking, is also under active investigation [1].

Current MRC implementations support four robots (the CART robot, the Steady Hand robot, the LARS robot and the MRT robot). Implementations for several others are under development. Two motion control interfaces, the MEI motion controller, and the proprietary motion controller for LARS are supported. Support for other motion controllers is being developed. Cross platform support for Solaris and linux is being tested. We plan to support clients on most operating systems.

3.2 General Sensors

For general sensors, we have established a simple architecture which shields an application from device specific API's. In general, it uses two device representations (see fig. 5). The first, a "Physical Device Representation" (PDR), represents the device itself and, except for a very small common API, is specific to the device. The application, however, can also interact with a "Logical Device" (LD). The abstraction for LD's is that each "senses" only a single kind of data (e.g. Points, or 2D images). Specific implementations then map a common interface to specific PDR's. This abstraction layers above MRC and other device abstractions, but provides the application layer with the ability, for example, to use either a robot or a tracker, (or other position sensor) transparently as a source for position information.

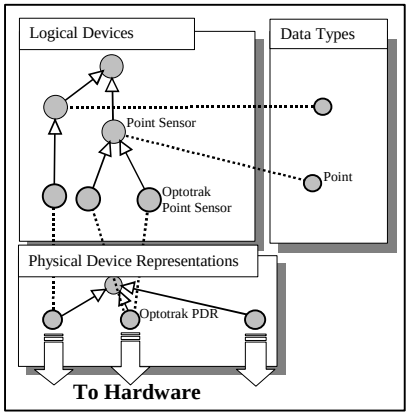

Figure 5. Relationships between Data Types, Physical Device Representations and Logical Devices within the General Sensor Architecture

PDR's for position sensing have been implemented for Optotraks, Flocks of Birds, and robots via MRC, as well as for 2D imaging via Matrox's family of image capture cards. These have been integrated into LP/PDR's and are currently in use in several systems in development. The Optotrak PDR supports network distribution using CORBA. This implementation has not yet been integrated into the architecture discussed in [1], however.

3.3 Trackers

Our architectures for robots and trackers share some significant parallels. While we view robots as maintaining information both in joint and Cartesian space, using kinematics to move between them, we view trackers as maintaining both 3D point and 6D frame (position + orientation) information, using a solver to go between. However, while the kinematics for a robot can be viewed as static, the rigid body configurations necessary for a solver are dynamic (at least run-time configurable). Additionally, solvers for pure 6DOF trackers are inherently different (frame to points, rather than points to frame), than those for point-based systems. This being said, however, many of the general principals of MRC apply here as well.

The abstract tracker library has not yet been fully implemented, though its layered structure has been used in the implementation of the Optotrak PDR.

3.4 Planners

For localized therapy, general planning systems can be posed as dosemtry planners. We have layered planners' internal information into targets and dose volumes. The Dosemtry layer takes targets to labeled dose volumes, the Optimizer layer takes prescribed dose volumes to targets. Only a prototype interface between 3D Slicer [22] and the planning software used in our fluoro-guided system [4] has been implemented.

4 Evaluation

Individual systems built using these components have been evaluated individually based on the requirements of their particular applications. Within such systems, the components have worked well, providing the required levels of performance. Direct evaluation of these methods is more difficult. However, two metrics which can be used to evaluate the architecture are integration time and the flexibility of the resulting system. A controlled comparison with other possible implementation architectures has not been done, nor is implementation of a system for the sole purpose of performance comparison likely. But, comparing these attributes to a base-line and measuring their incremental improvement as system implementations evolve are both options.

Implementation time can be easily quantified, though interpretation of the measure is dependant on environmental factors. Flexibility is harder to pin down, and is better as an incremental measure.

In two cases, the use of the techniques and components described here has demonstrably improved flexibility and/or integration time.

Prior to the integration of MRC, the fluoro-guided system was closely tied to the LARS robotic platform. With MRC's integration, however, it is able to control any of our robotic systems, clearly increasing the system's flexibility. This flexibility has sped development of the system itself, and allowed application to new imaging and manipulation environments.

The MRT robotic system provides another example. Slicer, which it uses for visualization, predates the development of the MRT robot significantly and, while it has significant 3D visualization, segmentation, registration, and measurement capabilities, it had no notion of robotics. With two days of work from two of us, however, robot control was added to Slicer using MRC. In a similar time, MRC components were implemented to control the novel MRT robot. Thus, with four day's work, three components: Slicer, MRC, and the MRT robot, which had never been connected were integrated into a working prototype. While there is no prior measure to compare this to directly, four day's work of two people is clearly significantly less than what is usually associated with the integration of robot control into new applications. Once this was accomplished, Slicer also inherited all of the flexibility associated with MRC and any application using MRC gained the ability to control the MRT robot.

5 Discussion

These architectures and implementations are a work in progress. We continue development and integration. In the long term, Slicer will serve an expanded role in systems using various kinds of image guidance, serving as a central visualization and user interaction platform for a variety of research systems.

The MRC library is being extended to include more robots, and servo controllers. The command scheduling, and networking support is also being upgraded.

The integration of imaging should also be extended. The distribution of imaging devices across the network is straightforward, although network performance will be a greater issue. An MRC-like abstraction over multiple imaging devices would also be useful, though less trivial.

Currently, device servers' functionality has been restricted—only providing computation directly related to their devices. Application specific information and

processing has generally been centralized on a single host, a machine which will often also provide visualization to the user. This division is not necessarily optimal, as more distributed, agent-based techniques might better utilize the system's resources, but these are also left as future extensions.

6 Conclusion

In our efforts to build an infrastructure for CIS research which is open and flexible, modularity and distribution have proven useful, when applied appropriately. We have applied these techniques based on our experience with a variety of robotic, sensing, and computational technologies for surgical assistance. Our groups have created layered architectures for robots, trackers, planners, and general sensors as well as network implementations which allow these components to be distributed to machines connected via a standard network. Several of these components have then been implemented and integrated into systems targeting different clinical applications, including retinal surgery and cancer therapy in the liver and prostate. While the development of these systems is ongoing, our approaches have demonstrably added to the flexibility and integration time for the components we have developed. Additionally, they are enabling technologies—creating an environment which will foster integration of new technologies into the next generation of safer, smarter, more capable CIS systems.

Acknowledgements

The authors gratefully acknowledge the support of the National Science Foundation under grant #IIS9801684, the Engineering Research Center grant #EEC9731478, the NSF/Whitaker Foundation for Cost Reducing Health Care Technology Grant #5T32HL07712, and the NIH under grants P41 RR13218-01 and P01 CA67165-03. This work was also supported in part by Johns Hopkins University internal funds.

References

[1] O. Schorr, N. Hata, A. Bzostek, R. Kumar, C. Burghart, R. H. Taylor, and R. Kikinis, "Distributed Modular Computer-Integrated Surgical Robot Systems: Architecture for Intelligent Object Distribution," proc. of Medical Image Computing and Computer Assisted Interventions, Pittsburgh, PA, USA, 2000.

[2] R. H. Taylor, J. Funda, B. Eldridge, K. Gruben, D. LaRose, S. Gomory, M. Talamini MD, L. Kavoussi MD, and J. Anderson, "A Telerobotic Assistant for Laparoscopic Surgery," in IEEE EMBS Magazine Special Issue on Robotics in Surgery, 1995, pp. 279-291.

[3] R. H. Taylor, A. Barnes, R. Kumar, P. Gupta, P. Jensen, L. L. Whitcomb, E. d. Juan, D. Stoianovici, and L. Kavoussi, " A Steady-Hand Robotic System for Microsurgical Augmentation," proc. of MICCAI'99, Cambridge, UK, 1999.

[4] A. Bzostek, A. C. Barnes, R. Kumar, J. H. Anderson, and R. H. Taylor, "A Testbed System for Robotically Assisted Percutaneous Pattern Therapy," proc. MICCAI '99, pp. 1098-1107, Cambridge, UK, 1999.

[5] S. Schreiner, J. Anderson , R. Taylor , J. Funda , A. Bzostek, and A. Barnes, "A system for percutaneous delivery of treatment with a fluoroscopically-guided robot," proc. of Joint Conf. of Computer Vision, Virtual Reality, and Robotics in Medicine and Medical Robotics and Computer Surgery, Grenoble, 1997.

[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 e. al, "Superconducting open-configuration MR imaging system for image-guided therapy," Radiology, v. 195, pp. 805-14, 1995.

[7] J. Schenck, F. Jolesz, P. Roemer, H. Cline, W. Lorensen, R. Kikinis, S. Silverman, C. Hardy, W. Barber, and E. Laskaris, "Superconducting open-configuration MR imaging system for image-guided therapy," Radiology, vol. 195, pp. 805-814, 1995.

[8] B. L. Davies, S. J. Harris, W. J. Lin, R. D. Hibberd, R. M. R, and J. C. Cobb, "Active Compliance in robotic surgery - the use of force control as a dynamic constraint," Proc Instn Mech Engrs, vol. 211, pp. 285-292, 1997.

[9] B. L. Davies, K. L. Fan, R. D. Hibberd, M. Jakopec, and S. J. Harris, "ACROBOT - Using Robots and Surgeons Synergistically in Knee Surgery," proc. of 8th International Conference on Advanced Robotics, California, USA, 1997.

[10] S. J. Harris, W. J. Lin, K. L. Fan, R. D. Hibberd, J. Cobb, R. Middleton, and B. L. Davies, "Experiences with robotic systems for knee surgery," proc. of Proc. First Joint Conference of CVRMed and MRCAS, pp. 757-766, Grenoble, France, 1997.

[11] J. Y. Delnondedieu and J. Troccaz, "PADyC: A Passive Arm with Dynamic Constraints - A two degree-of-freedom prototype," proc. of Proc. 2nd Int. Symp. on Medical Robotics and Computer Assisted Surgery, pp. 173-180, Baltimore, Md., 1995.

[12] J. E. Colgate, W. Wannasuphoprasit, and M. A. Peshkin, "Cobots: Robots for Collaboration with Human Operators," proc. of International Mechanical Engineering Congress and Exhibition, pp. 433-39, Atlanta, GA, USA, 1996.

[13] S. E. Salcudean, G. Bell, S. Bachmann, W. H. Zhu, P. Abolmaesumi, and P. D. Lawrence, "Robot-Assisted Diagnostic Ultrasound - Design and Feasibility Experiments," proc. of MICCAI'99, pp. 1062-1071, Cambridge, UK, 1999.

[14] H. Das, H. Zak, J. Johnson, J. Crouch, and D. Frambach, "Evaluation of a Telerobotic System to Assist Surgeons in Microsurgery," Computer Aided Surgery, vol. 4, pp. 15-25, 1999.

[15] T. B. Sheridan, "Human factors in Tele-inspection and Tele-surgery:Cooperative manipulation under Asynchronous Video and Control Feedback," proc. of MICCAI'98, pp. 368-376, Cambridge, MA, USA, 1998.

[16] A. C. Sanderson and G. Perry, "Sensor Based Robotic Assembly Systems: Research And Applications in Electronic Manufacturing," Proceedings of the IEEE, vol. 71, pp. 856-871, 1983.

[17] R. H. Taylor and D. D. Grossman, "An Integrated Robot Systems Architecture," IEEE Proceedings, 1983.

[18] D. Stoianovici, L. L. Whitcomb, J. H. Anderson, R. H. Taylor, and L. R. Kavoussi, "A Modular Surgical Robotic System for Image Guided Percutaneous Procedures," proc. of MICCAI'98, pp. 404-410, Cambridge, MA, USA, 1998.

[19] K. Chinzei, R. Kikinis, and F. A. Jolesz, "MR Compatibility of Mechatronic Devices: Design Criteria," proc. of MICCAI'99, Cambridge, UK, 1999.

[20] F. A. Jolesz, "Image-Guided Procedures and the Operating," Radiology, vol. 204, pp. 601-612, 1997.

[21] R. C. Susil, J. H. Anderson, and R. H. Taylor, "A Single Image Registration Method for CT Guided Interventions," proc. of MICCAI'99, pp. 798-808, Cambridge, UK, 1999.

[22] D. Gering, A. Nabavi, R. Kikinis, W. E. L. Grimson, N. Hata, P. Everett, F. Jolesz, and W. W. III, "An Integrated Visualization System for Surgical Planning and Guidance using Image Fusion and Interventional Imaging," proc. of MICCAI'99, Cambridge, UK, 1999.