The Surgical Assistant Workstation (SAW) in Minimally-Invasive Surgery and Microsurgery

Release 1.0

Peter Kazanzides[1], Simon P. DiMaio[2], Anton Deguet[1], Balazs Vagvolgyi[1], Marcin Balicki[1], Caitlin Schneider[1], Rajesh Kumar[1], Amod Jog[1], Brandon Itkowitz[2], Christopher Hasser[2], and Russell H. Taylor[1]

June 15, 2010

[1]Johns Hopkins University, Baltimore, MD
[2]Intuitive Surgical Inc., Sunnyvale, CA

Abstract

The Surgical Assistant Workstation (SAW) provides a modular, open-source software framework to support rapid prototyping of computer-assisted surgery systems, especially those that benefit from enhanced 3D visualization and user interaction. The framework includes a library of components that can be used to implement master-slave or collaborative robot control systems with support for complex video pipelines and a novel interactive surgical visualization environment. SAW includes standardized interface definitions (e.g., command names and payloads), with the goal of making the framework easily extensible so that developers can add support for their own robotic devices and associated hardware platforms. This paper presents an overview of the component-based architecture, followed by applications (use cases) in the areas of minimally-invasive surgery (MIS), microsurgery, and surgical skill assessment.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3179]

Contents

1 Introduction

The Surgical Assistant Workstation (SAW) provides a modular, open-source component-based software framework to support rapid prototyping of medical robotics and computer-assisted surgery systems. It utilizes the *cisst* C++ libraries [4, 8] to provide basic foundation classes (data types such as vectors, matrices, transformations, and tools such as class and object registries, logging, etc.) and a component-based framework that supports different execution models, such as periodic threads, callbacks, and event-based programming. As illustrated in Fig. 1, SAW includes a number of Interface Components (IC), that encapsulate hardware devices, and software components such as robot motion, collaborative control, speech recognition, 3D user interface, and video processing. A key aspect is the definition of interface standards, within the SAW framework, that enable "plug-and-play" configuration of systems. For example, although robots are physically different, their interfaces (e.g., command names and parameters) have been standardized as much as possible. SAW also includes an interface component that supports the OpenIGTLink research standard[11] that has been gaining acceptance in the image-guided therapy (IGT) community.

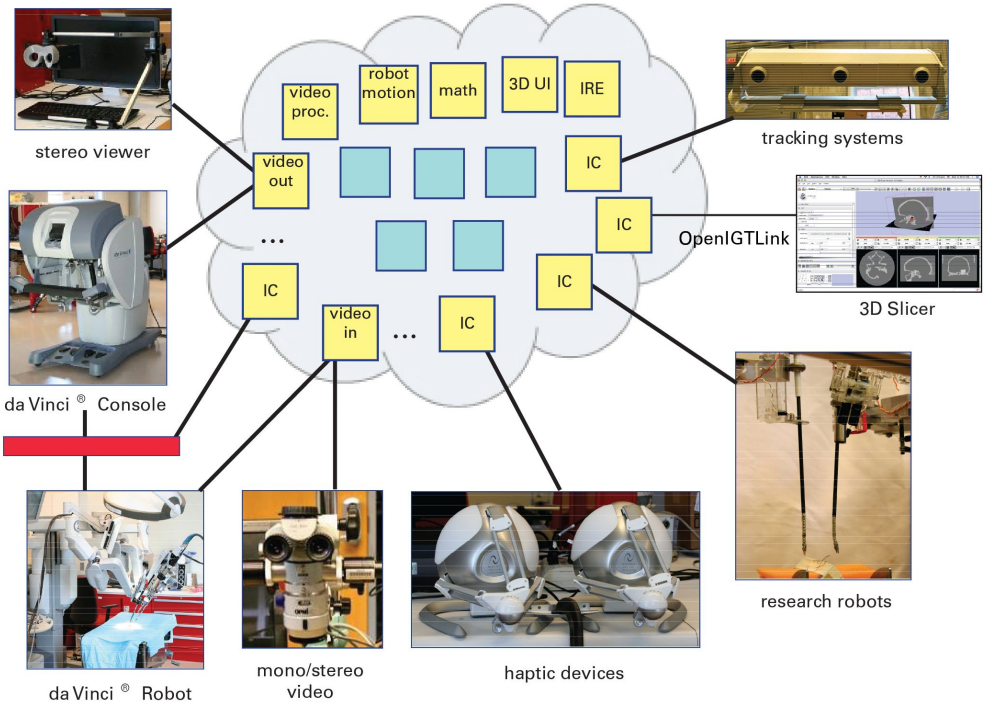

Figure 1: Surgical Assistant Workstation (SAW). New application software (blue boxes) can utilize existing interface components (IC) and software components (yellow boxes) to streamline development.

SAW bears some similarity to component-based frameworks for robotics that have been introduced in other application domains, such as mobile robots. These include Player[6], OROCOS [3], Orca [2], and ROS[9]. Player is a popular set of tools for mobile robot research. Conceptually, it is a hardware abstraction layer for mobile robot devices that includes data communication mechanisms between drivers and control programs.

The Open ROBot COntrol Software (OROCOS) was started in 2001 to develop open source robot control software. It includes real-time C++ libraries for advanced machine-tool and robot control. Orocos components communicate with each other using interfaces which consist of properties, events, methods, commands and data flow ports. Branching from the OROCOS project, the Orca Project aims to provide building-blocks (components) that can be combined together to build arbitrarily complex non-real-time robotic systems. Willow Garage's Robot Operating System (ROS) is a more recent entry to the field, but has undergone rapid development. It is an open-source package that provides operating system types of services, such as hardware abstraction, low-level device control, and communication between processes, as well as numerous tools to facilitate development. The intent is to create a common platform upon which researchers can build, and then share, high-level robotic algorithms in areas such as navigation, localization, planning, and manipulation.

The SAW framework is most similar to OROCOS, as they both define comparable components with features such as lock-free data exchanges that support hard real-time performance. The key differences, however, stem from the different application domains. OROCOS is primarily targeted at control of industrial robots and machine tools, whereas SAW is focused on medical robotics and computer-assisted surgery. Although the underlying robotics functionality is similar, the set of associated components, such as sensors and user interface elements, is application-specific.

2 System Architecture

SAW is a component-based system and thus the architecture is defined by the available components and their interconnections. SAW is built on the component framework provided by the *cisst* libraries[4, 8]. In particular, *cisst* defines a base component class and a number of derived component frameworks that facilitate creation of components with active execution models (e.g., periodic tasks, continuous tasks). All of these component frameworks contain a list of provided interfaces and required interfaces. The system defines a component manager that is used to control components (e.g., start, stop) and to create connections between their provided and required interfaces. For example, the required interface of one component can be connected to the provided interface of another component, as shown in Fig. 2. In this context, it is convenient to refer to first component as the client and the second component as the server, though it should be noted that every component can have provided and required interfaces and therefore act as both client and server. Each provided interface contains a number of command objects that encapsulate the available services, which generally are implemented by class member functions of the server component. The provided interface may also generate events. The client's required interface contains a list of function objects that are bound, at runtime, to the corresponding command objects in the server's provided interface. Similarly, the required interface may specify event handlers for events generated by the provided interface. The binding of function objects (event handlers) to command objects (event generators) is implemented using string comparisons and occurs when the required interface is connected to a provided interface. Subsequent communication is efficiently handled by directly invoking the function objects (on the client side) or event generators (on the server side). The *cisst* component interfaces also include data stream inputs and outputs, which are provided to establish real-time image (e.g., video, ultrasound) streams between components.

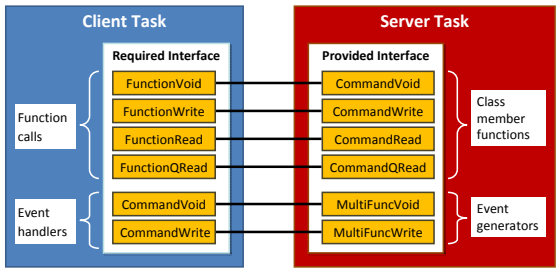

Figure 2: *cisst* component-based framework

While *cisst* provides the basic component framework, the SAW architecture further defines the set of components, interfaces, and interconnections to create an application framework that is suitable for research in advanced telesurgical and cooperatively controlled robots[12, 7]. Fig. 3 shows a typical data flow in such a system and indicates some of the key hardware components, such as robots, video input and output devices, and other sensors and imaging equipment, along with the software components that coordinate and process the associated information. Because this architecture is intended for research with diverse hardware platforms, it is important to precisely define the component interfaces, so that similar components can be substituted. For example, it is necessary to standardize the interface to a robot component, thereby enabling researchers to construct a platform using da Vinci robots, Phantom haptic devices, or any other robot arm. Similarly, the video processing pipeline should enable video capture from different imaging devices and video output, including overlay, on different display devices. The SAW architecture therefore has led to the definition of standardized command names and data types for robots and other devices. The data types are defined in the `cisstParameterTypes` library, which includes all supporting functionality such as serialization, deserialization, and dynamic creation. These data types are available for use by components.

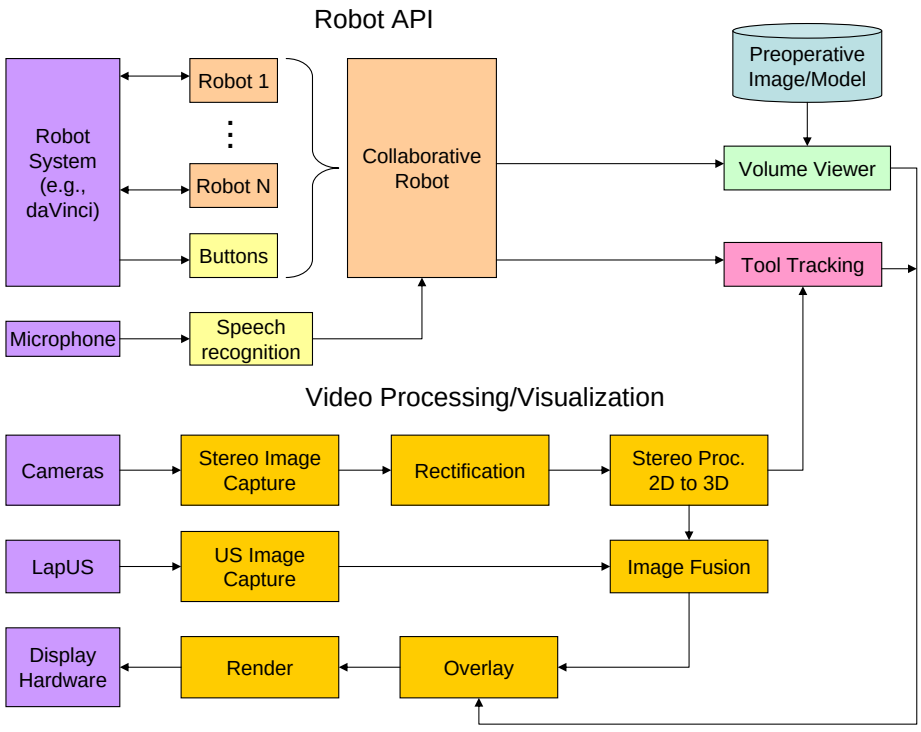

Figure 3: Sample SAW Data Flow

Finally, a component-based framework is most useful when it includes a number of implemented components that can be reused for a new application. Thus, SAW includes a number of components that provide hardware interfaces – the Interface Components (IC) shown in Fig. 1 – as well as the software components shown in this, and other, figures such as robot motion, collaborative control, speech recognition, 3D user interface, and video processing.

The system includes two research interface components to the da Vinci® robot (Intuitive Surgical, Inc., Sunnyvale, CA), as shown in Fig. 4. The first allows user client applications to receive a uni-directional stream of kinematic and event data from the da Vinci surgical system during operation, and is approved for use during human clinical cases [5]. The SAW da Vinci read-only API component communicates with the da Vinci via a TCP/IP protocol and provides interfaces to all data stream fields and events transmitted by the system. A second API component has been developed for bi-directional communication with the da Vinci system. This API provides a rich set of functions for both querying system state and directly controlling motion and behavior of both patient-side and console manipulators. Due to obvious safety concerns and regulatory limitations, this interface is not permitted for use during human surgeries. The bi-directional, read-write API encapsulates a binary library (the BBAPI library) that is used to communicate with the da Vinci via an Intuitive Surgical proprietary interface. In order to use the da Vinci research interfaces, users must develop and sign a research agreement with Intuitive Surgical.

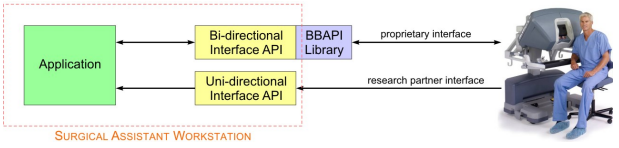

Figure 4: da Vinci® interface components

3 Use Cases

The development of SAW has been motivated by concurrent development of several "use cases" that provide focus for defining and testing the functions of the SAW system. Conversely, the SAW development has proved invaluable in facilitating the research associated with the use cases.

3.1 Augmented Reality in MIS Surgery

SAW is an information-rich immersive environment for planning and executing telesurgical procedures, such as those performed with the da Vinci robot. The underlying principle is that the master console should be used not only to control the patient-side robots, but also to provide a 3D interface that enables the user to interact with other sources of information and provide high-level supervisory control commands to the system.

One use case in the area of MIS surgery is the development of ultrasound-guided liver tumor biopsy, ablation, and resection capabilities for the da Vinci, allowing surgeons to perform minimally invasive liver interventions with the efficacy of open surgery. While the project uses liver surgery as a focusing application, the addition of laparoscopic ultrasound (LapUS) guidance to the da Vinci platform promises much broader benefits in general, cardiac, ob/gyn, and urologic surgery. Fig. 5 shows the current embodiment of the laparoscopic ultrasound tool developed for the da Vinci and illustrates the role played by the SAW software, as well as some typical user interfaces[10].

To support this functionality, the system provides a *masters as mice* mode, where the master manipulators are decoupled from the remote manipulators and used as 3D input devices to manipulate graphical objects in the 3D environment. The LapUS project uses this mode to enable interaction with the ultrasound image; for example, the surgeon can move the displayed US image to a different region of the visual field so that it does not obstruct the view of important anatomy. The *masters as mice* mode also provides a Marker Tool, shown in Fig. 5-b, that enables the surgeon to place markers in the 3D environment and then later realign the robotic instruments with these markers. The markers are referenced to the patient and their positions are updated when the camera moves. Similarly, when activated by the surgeon, a Measurement Tool displays (as a text

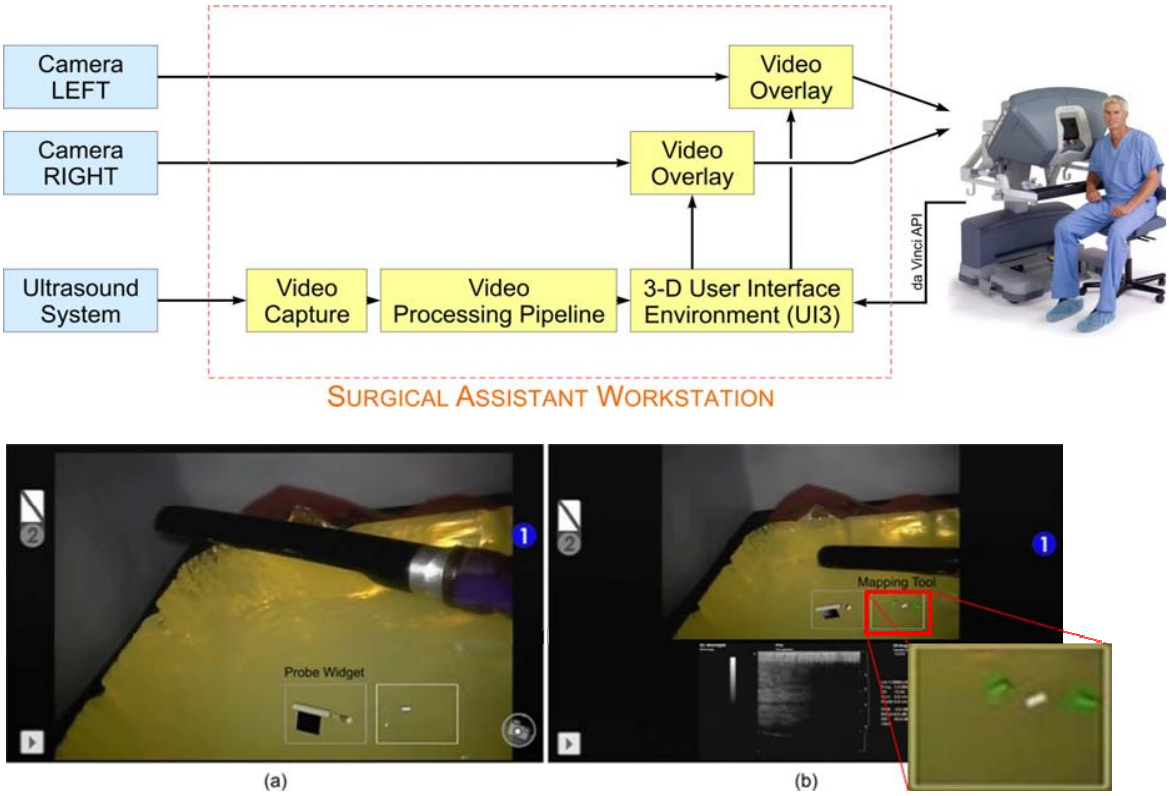

Figure 5: Laparoscopic Ultrasound (LapUS) with Surgical Robots: (Top) A high-level system schematic illustrating the use of SAW for video processing, 3-D user interface, and display within the da Vinci console. (Bottom) (a) the graphical probe widget indicates the transducer and image plane orientation as well as wrist configuration. (b) A mapping tool indicates the current probe position and orientation (white cursor), as well as "bread crumbs" (green markers).

overlay) the 3D distance traveled by the robotic tool; this can be useful for measuring anatomical features.

Other forms of augmented reality for MIS include a Volume Viewer, which enables the surgeon to load a 3D model (such as a preoperative CT or MRI scan) of the patient, as shown in Fig. 6, and interact with it in 3D using the master input devices. The model can appear in a small "3D window" within the field of view (as in Fig. 6), or it can be overlaid onto the stereo video. The latter case requires a registration between the coordinate systems of the 3D model and the stereo video.

The *masters as mice* mode also enables the user to provide high-level control parameters, such as the definition of virtual fixtures or autonomous motion macros. As a simple example, this interface could be used to allow the user to define safety boundaries (boundary virtual fixtures) in the 3D view, which could then be enforced by the system

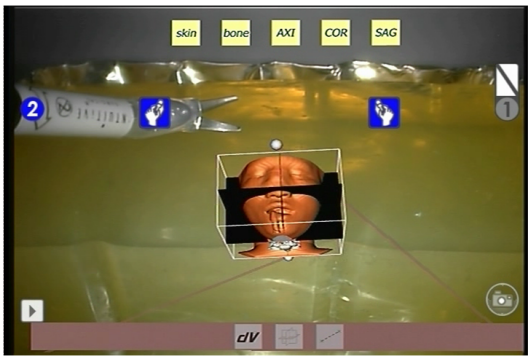

Figure 6: SAW application showing interaction with 3D medical image overlayed on visual field.

during telemanipulation.

3.2 Cooperative Control and Information Fusion for Microsurgery

The goal of this project is to develop technology and systems addressing fundamental limitations in current microsurgical practice, using vitreoretinal surgery as an intitial focus. Vitreoretinal surgery is the most technically demanding ophthalmologic discipline and addresses prevalent sight-threatening conditions in areas of growing need. Retinal surgery is currently performed under a stereo microscope with free-hand instrumentation. Limitations include limited visual resolution, and physiological hand tremor. The surgeon also struggles with a lack of tactile feedback, proximity sensing, and real-time sensing of physiological parameters of the retina. Surgical technique and efficiency would be enhanced by the integration of preoperative images with the intraoperative view. In current practice, poor ergonomics result in surgeon fatigue and potential disability. All of these factors contribute to extended operating times, associated light toxicity, and higher than needed complication rates. At the center of our planned approach is a "surgical workstation" system, shown in Fig. 7, interfaced to a stereo visualization subsystem and a family of novel sensors, instruments, and robotic devices[1]. The capabilities of these components individually address important limitations of current practice; together they provide a modular, synergistic, and extendable system that enables computer-interfaced technology and information processing to work in partnership with surgeons to improve clinical care and enable novel therapeutic approaches.

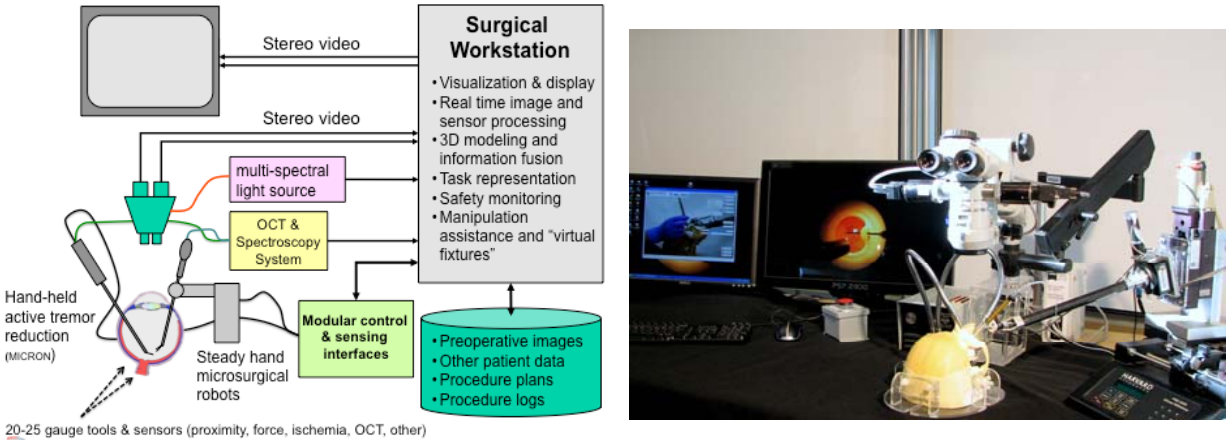

Figure 7: Microsurgery Assistant Workstation: (Left) System block diagram, showing major components; (Right) Typical experimental setup showing stereo video microscope and 3D display, Steady Hand microsurgical robot holding cannulation instrument, and phantom head containing chick embryo.

3.3 Surgical Skill Assessment

We are currently conducting a multi-center study of robotic surgery training focusing on development and longitudinal assessment of surgical skill (see Fig. 8). Such assessment will be key for developing feedback methods to robotic surgery trainees. We aim to separately investigate automated methods for surgical and system skill development and assessment, including creation of ground truth for such methods using manual expert assessment of clinical skill using methods such as Objective Structured Assessment of Technical Skill (OSATS) evaluations. Figure 8,right, shows the motion trajectories of the left and right masters during a

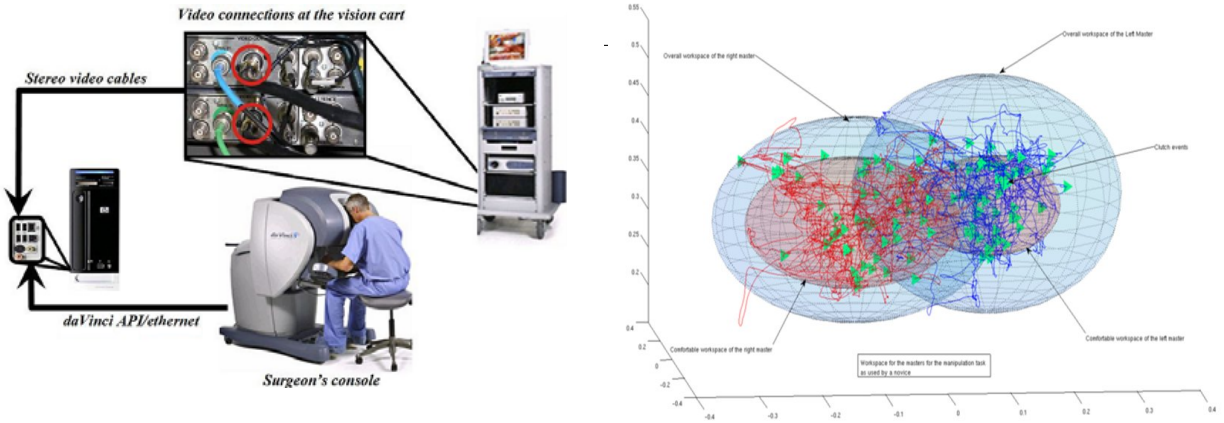

Figure 8: Robotic surgery training data collection infrastructure (left), and motion envelop of the master during a training task (right)

training tasks. The green triangles represent camera or master workspace reconfiguration. Similar data from 24 trainee subjects is currently being processed for development of skill metrics.

4 Conclusions

The Surgical Assistant Workstation (SAW) is an open-source, component-based C++ software framework developed to support research in medical robotics and computer-assisted surgery. It is available at www.cisst.org/saw. Key features of SAW include interface components to a variety of devices, such as robots and sensors, used to build such systems, as well as software libraries for functionalities such as real-time image processing, 3D user interfaces, data collection, and robot control. An important point is the attempt to standardize interfaces to similar components, to enable "plug and play" system construction. Currently, these standards only apply within the SAW framework, as there is not yet an accepted industry standard. It is possible, however, to develop interface components (adapters) to "bridge" between SAW and any standard that may evolve in the future. For example, SAW already includes an adapter for the recently introduced OpenIGTLink research protocol. Use cases in minimally-invasive surgery, microsurgery, and surgical skill assessment demonstrate the versatility of SAW and suggest broader applicability within the field of computer-assisted surgery.

Acknowledgments

This work is supported in part by National Science Foundation grants EEC 9731748, EEC 0646678, MRI 0722943, in part by National Institutes of Health grants R01 EB007969, R42 RR019159, R21 EB009143, in part by the Swirnow Foundation, and in part by Johns Hopkins and Intuitive Surgical internal funds.

References

[1] M. Balicki, J.-H. Han, I. Iordachita, P. Gehlbach, J. Handa, R. H. Taylor, and J. Kang. Single fiber optical coherence tomography microsurgical instruments for computer and robot-assisted retinal surgery. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 108–115, London, Sep 2009. 3.2

[2] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback. Towards component-based robotics. In *Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 163–168, Aug 2005. 1

[3] H. Bruyninckx, P. Soetens, and B. Koninckx. The real-time motion control core of the Orocos project. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 2766–2771, Sep 2003. 1

[4] A. Deguet, R. Kumar, R. Taylor, and P. Kazanzides. The *cisst* libraries for computer assisted intervention systems. In *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions*, Midas Journal: http://hdl.handle.net/10380/1465, Sep 2008. 1, 2

[5] S. DiMaio and C. Hasser. The da Vinci research interface. In *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions*, Midas Journal: http://hdl.handle.net/10380/1464, Sep 2008. 2

[6] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc. Intl. Conf. on Advanced Robotics (ICAR)*, pages 317–323, 2003. 1

[7] M. Y. Jung, T. Xia, R. Kumar, A. Deguet, R. Taylor, and P. Kazanzides. A Surgical Assistant Workstation (SAW) application for a teleoperated surgical robot system. In *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions*, Midas Journal: http://hdl.handle.net/10380/3079, Sep 2009. 2

[8] P. Kazanzides, A. Deguet, and A. Kapoor. An architecture for safe and efficient multi-threaded robot software. In *Technologies for Practical Robot Applications (TePRA)*, pages 89–93, Woburn, MA, Nov 2008. 1, 2

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009. 1

[10] C. M. Schneider, G. W. Dachs II, C. J. Hasser, M. A. Choti, S. P. DiMaio, and R. H. Taylor. Robot-assisted laparoscopic ultrasound. In *Information Processing in Computer-Assisted Interventions (IPCAI)*, Geneva, Jun 2010. 3.1

[11] J. Tokuda, G. S. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, H. Liu, J. Blevins, J. Arata, A. J. Golby, T. Kapur, S. Pieper, E. C. Burdette, G. Fichtinger, C. M. Tempany, and N. Hata. OpenIGTLink: an open network protocol for image-guided therapy environment. *Intl. Journal of Medical Robotics and Computer Assisted Surgery*, 5(4):423–434, Dec 2009. 1

[12] B. Vagvolgyi, S. DiMaio, A. Deguet, P. Kazanzides, R. Kumar, C. Hasser, and R. Taylor. The Surgical Assistant Workstation: a software framework for telesurgical robotics research. In *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions*, Midas Journal: http://hdl.handle.net/10380/1466, Sep 2008. 2