

System Requirements for Surgical Assistant Workstation

Rev 2

January 29, 2007

**Johns Hopkins University
Center for Computer-Integrated Surgical
Systems and Technology (CISST)**

Intuitive Surgical Systems, Inc.

System Requirements for Surgical Assistant Workstation (SAW)

1. Objective.....	3
2. References.....	3
2.1. Project-Specific.....	3
2.2. Calibration and Registration Techniques.....	3
2.3. Virtual Fixtures and Constrained Optimization.....	4
3. System Overview.....	4
4. Functional Requirements.....	6
4.1. Robot API.....	6
4.2. Video processing.....	8
4.3. Other device interfaces.....	8
4.4. Calibration and registration.....	8
4.5. Tool tracking.....	10
4.6. User Interface (Visualization).....	10
4.7. Telesurgery application framework.....	11
4.8. Volume viewer.....	12
5. Performance Requirements.....	13
5.1. Robot API.....	13
5.2. Video processing.....	13
5.3. Other device interfaces.....	14
5.4. Calibration and registration.....	14
5.5. Tool Tracking.....	14
5.6. User Interface (Visualization).....	15
5.7. Telesurgery application framework.....	15
5.8. Volume viewer.....	15
6. Safety Requirements.....	15
6.1. Robot API and Interface.....	15
6.2. Video processing.....	15
6.3. Other device interfaces.....	15
6.4. Calibration and registration.....	15
6.5. Tool tracking.....	15
6.6. User Interface (Visualization).....	16
6.7. Telesurgery application framework.....	16
6.8. Volume viewer.....	16
7. Design Constraints.....	16
7.1. Operating System.....	16
7.2. Programming Language.....	16
7.3. Software Libraries.....	16
8. Change History.....	16

1. Objective

This document defines the requirements for the Surgical Assistant Workstation for Teleoperated Surgical Robots (SAWTSR) being developed by Johns Hopkins University and Intuitive Surgical. This document provides requirements for the workstation system and does not include application requirements. It is expected that applications developed on this workstation would define their own requirements (i.e., in a separate document).

2. References

2.1. Project-Specific

- 2.1.1. "Development of a Surgical Assistant Workstation for Teleoperated Surgical Robots," proposal for NSF ERC Supplement, July 2006.
- 2.1.2. "Intuitive Surgical daVinci API v5.0 Reference Manual", generated July 14, 2006.
- 2.1.3. J. Leven, D. Burschka, R. Kumar, G. Zhang, S. J. Blumenkranz, X. Dai, M. Awad, G. Hager, M. Marohn, M. Choti, C. Hasser and R. H. Taylor "DaVinci Canvas: A Telerobotic Surgical System with Integrated, Robot-Assisted, Laparoscopic Ultrasound Capability," in MICCAI, vol. LNCS 3749, J. Duncan and G. Gerig, Eds. Palm Springs, CA: Springer-Verlag, 2005, pp. 811-818.
- 2.1.4. J. Leven, "A Telerobotic Surgical Systems with Integrated Robot-Assisted Laparoscopic Ultrasound Capability", MS Thesis, Computer Science, Johns Hopkins University, Baltimore, 2005.

2.2. Calibration and Registration Techniques

- 2.2.1. B.K.P. Horn, "Closed-form solution of absolute orientation using unit quaternions," J. Opt. Soc. Amer. A, Vol. 4, No. 4, pp. 629-642, Apr. 1987.
- 2.2.2. K.S. Arun, T.S. Huang, S.D. Blostein, "Least-Squares Fitting of Two 3D Point Sets", IEEE PAMI, Vol. 9, No. 4, pp. 698-700, Sept. 1987.
- 2.2.3. S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns", IEEE PAMI, Vol. 13, No. 4, pp. 376-380, Apr. 1991.
- 2.2.4. E. Boctor, A. Viswanathan, M. Choti, R. Taylor, G. Fichtinger, G. Hager, "A Novel Closed Form Solution for Ultrasound Calibration," IEEE Intl. Symp. Bio. Imag. (ISBI), Arlington, VA, pp 527-530, Apr. 2004.
- 2.2.5. P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, pp. 239-256, Feb. 1992.

2.3. Virtual Fixtures and Constrained Optimization

- 2.3.1. A. Kapoor, M. Li, R.H. Taylor, “Constrained Control for Surgical Assistant Robots”, Proc. IEEE Intl. Conf. on Robotics and Automation, Orlando, FL, May 2006, pp 231-236.
- 2.3.2. M. Li, A. Kapoor and R. H. Taylor “A Constrained Optimization Approach to Virtual Fixtures,” in IROS. Edmonton, Alberta, Canada, 2005.
- 2.3.3. M. Li and R. H. Taylor, “Performance of Teleoperated and cooperatively controlled surgical robots with automatically generated spatial virtual fixtures.,” in IEEE International Conference on Robotics and Automation. Barcelona, Spain, 2005.
- 2.3.4. M. Li, “Intelligent Robotic Surgical Assistance for Sinus Surgery”, Ph.D. Thesis, Computer Science, The Johns Hopkins University, Baltimore, Maryland, 2005.
- 2.3.5. M. Li and R. H. Taylor, “Spatial Motion Constraints in Medical Robots Using Virtual Fixtures Generated by Anatomy,” in IEEE Conf. on Robotics and Automation. New Orleans, 2004, pp. 1270-1275.

3. System Overview

The goal is to create a unified assistive environment for surgery that integrates robotic devices; fused information environments combining preoperative images & models, intraoperative images & other sensors; surgical task modeling; and human-machine cooperative manipulation, as shown in Figure 1 (from Reference 2.1.1).

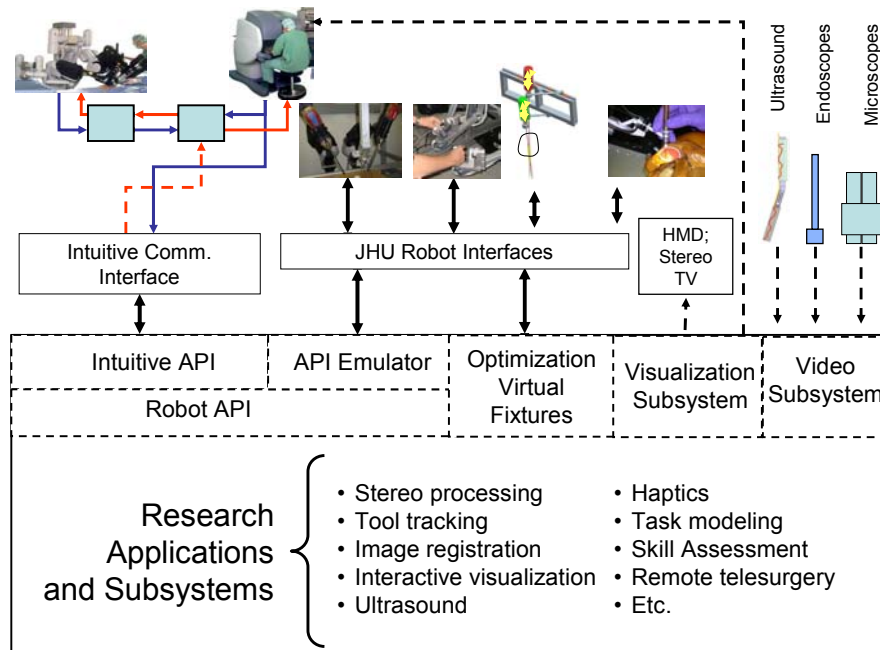


Figure 1: Overview of SAWTSR Architecture

In this document, the system requirements are categorized by the major subsystem:

- **Robot API:** this subsystem corresponds to the interface to the robot or robots; although the design shall be influenced by the existing daVinci research API, it shall be usable for other robot systems.
- **Video processing:** this subsystem provides the processing of 2D (e.g., ultrasound) and 3D (e.g., stereo video) images, using image pipelines.
- **Other device interfaces:** this subsystem provides interfaces to devices other than robots and video systems, including force sensors, foot pedals, tissue oxygenation sensors, etc.
- **Calibration and registration:** this subsystem provides tools for calibrating devices or (at a minimum) for reading the calibration results produced by an external system (e.g., Matlab programs). It also provides methods for computing coordinate transformations (e.g., registration).
- **Tool tracking:** this subsystem provides the capability for tracking the positions of tools using some combination of sensor feedback (e.g., joint encoder positions and stereo video images).
- **User Interface (Visualization):** this subsystem provides the 2D and 3D graphical displays and accepts control information from input devices (including the master manipulators in a telesurgical system).
- **Telesurgery application framework:** this corresponds to a working “skeleton” application, which the researcher can customize. System-level requirements are listed in this category.

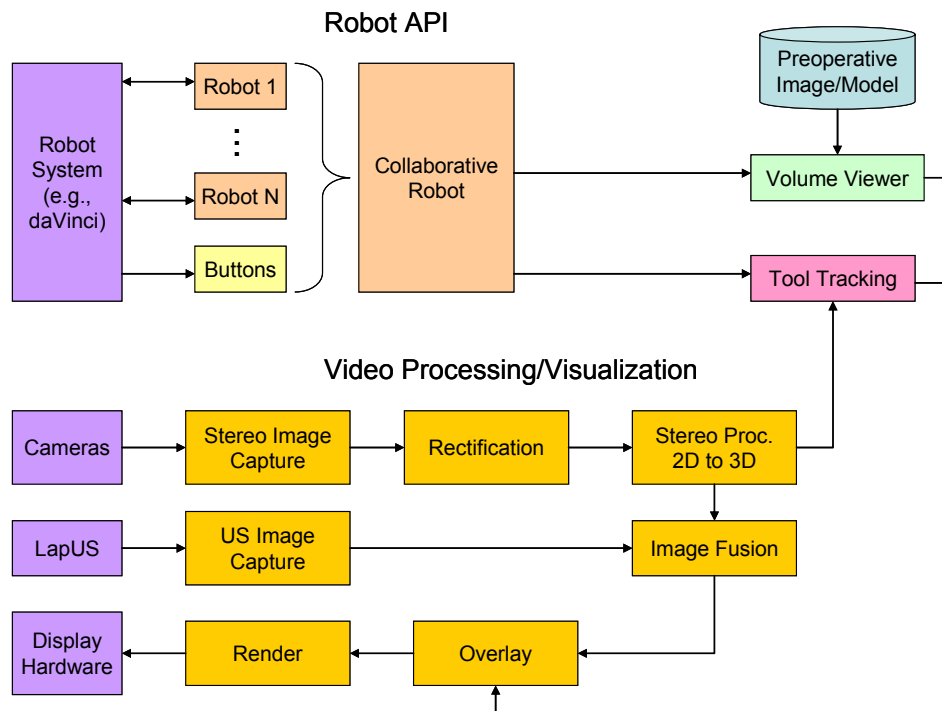


Figure 2: Illustrative data flow

- **Volume Viewer:** this is an application-level “widget” that allows the user to view and manipulate medical images, using the daVinci master manipulators as input devices.

Figure 2 shows an illustrative data flow diagram, focusing on the robot API and the pipeline for the video processing and visualization. This figure also shows the tool tracking and volume viewer subsystems. Although not specifically shown, calibration and registration functions are required. Note that other data flow configurations are possible, depending on the application requirements.

4. Functional Requirements

4.1. Robot API

- 4.1.1. The robot API shall provide an interface to the different robots that may be attached to the system, including:
 - Research daVinci systems, via the read-write research interface (Reference 2.1.2).
 - Clinical daVinci systems, via the read-only research interface (older V4.x interface mentioned in Reference 2.1.2).
 - daVinci master and slave arms, via the JHU controller.
 - JHU robots such as the “snake” robot and the steady hand robot for retinal surgery.
- 4.1.2. The robot API shall provide method(s) to initialize the interface. For the daVinci research API, this would encompass the stream management functions.
- 4.1.3. The robot API shall provide an interface to individual robots (e.g., daVinci PSM or MTM) and collaborations of multiple robots:
 - 4.1.3.1. Each individual robot shall be represented by an instance of a robot object. The methods of that object shall provide the API for a single robot. This is analogous to the “manipulator” commands in the daVinci research API.
 - 4.1.3.2. Collaborative groups of robots (such as master-slave pairs) shall be represented by an instance of a “collaborative robot” object, which shall contain two or more robot objects, as well as other devices, such as surgeon console buttons (see Section 4.3.1). This is analogous to the “supervisor” commands in the daVinci research API.
- 4.1.4. Individual and collaborative robots can be “read-only” (i.e., provide only state information) or “read-write” (i.e., provide state information and allow state changes).

- 4.1.5. The robot API for the individual and collaborative robot objects shall use the CISST fundamental data types (vectors, matrices, transformations), rather than the math support functions in the daVinci research API.
 - 4.1.5.1. This may require translation from CISST data types to ISI data types (e.g., array of floats) and vice-versa.
- 4.1.6. The methods of the individual and collaborative robot objects shall be documented in an external database/document, which shall become the functional specification for the robot API. This database shall contain the following information:
 - 4.1.6.1. Method name
 - 4.1.6.2. Number and types of parameters
 - 4.1.6.3. Functional description
- 4.1.7. The API shall generate events to notify the user application about asynchronous actions detected by the lower level software. The events of the individual and collaborative robot objects shall be documented in an external database/document.
 - 4.1.7.1. Individual robot events shall include: Emergency stop signaled, power amplifier fault, hardware limit reached, etc. (if available for that robot).
 - 4.1.7.2. Collaborative robot events shall include all individual robot events and master console events such as buttons or pedals pressed.
 - 4.1.7.3. The API shall have an extendable architecture to allow new events to be added.
 - 4.1.7.4. Individual and collaborative robots are not required to implement all defined events.
- 4.1.8. Virtual fixtures
 - 4.1.8.1. The API shall include commands for specifying and enabling haptic constraints, such as virtual fixtures.
 - 4.1.8.2. This interface shall be designed to incorporate, as seamlessly as possible, both the JHU constrained optimization implementation (Reference 2.3.1) and the ISI embedded constraint primitives (e.g., infinite planes, triangle patches, detents) described in Reference 2.1.2.

4.2. Video processing

- 4.2.1. The system shall include hardware to capture video images from a number of sources, including stereo cameras and ultrasound.
- 4.2.2. The software shall include a method for 3D surface reconstruction from stereo video images (see 4.3).
- 4.2.3. In addition to video, the software shall accept inputs from other sources, such as prior volumetric images and models.
- 4.2.4. The software should work with recorded data (e.g., video files).
- 4.2.5. The software shall have a method for synchronizing the video frames to other data, such as robot feedback (see 4.7.7). One possibility is to utilize an image format that enables additional information to be stored as meta-data.

4.3. Other device interfaces

- 4.3.1. The software shall provide interfaces to various user input devices, such as foot pedals, switches, and buttons (including those on the daVinci master console).
- 4.3.2. The software shall provide an interface to sensors that can measure forces and torques, including full 6-dof force/torque data.
- 4.3.3. The architecture shall be extensible, allowing the integration of other devices not yet specified.

4.4. Calibration and registration

- 4.4.1. The system shall support calibration of the following:
 - 4.4.1.1. *Ultrasound probe*: position/orientation of the 2D image plane with respect to a 3D reference frame on the probe.
 - 4.4.1.2. *Camera calibration (intrinsic)*: the determination of camera parameters such as focal length, resolution, optical center, and lens distortion.
 - 4.4.1.3. *Stereo camera calibration (extrinsic)*: the transformation between two camera systems (or one camera moved between two positions).
 - 4.4.1.4. *Robot kinematics*: the transformation between a frame on the final link of the robot and a frame on the base of the robot.
 - 4.4.1.5. *Tool tip*: the transformation between a frame on the tip of a (rigid) tool and a frame defined elsewhere on the tool. This calibration may not require a full transformation; for example, a translation vector is sufficient if only the tip position is required.

System Requirements for Surgical Assistant Workstation (SAW)

- 4.4.2. The system shall support each calibration listed in 4.4.1 by providing the following:
 - 4.4.2.1. The system may (optionally) provide functions for collecting calibration data. Otherwise, this can be provided by external software.
 - 4.4.2.2. The system may (optionally) provide functions to compute the calibration parameters. Otherwise, this can be done by external software (e.g., Matlab program).
 - 4.4.2.3. The system shall have a method for using the results of the calibration (e.g., by reading the results from a file).
- 4.4.3. The system shall provide routines to register the following coordinate systems (i.e., find the transformation from one coordinate system to another coordinate system):
 - 4.4.3.1. stereo video image
 - 4.4.3.2. ultrasound image
 - 4.4.3.3. preoperative image (e.g., CT) or model (e.g., segmented volume)
 - 4.4.3.4. robot
 - 4.4.3.5. external measurement device (e.g., Optotrak)
- 4.4.4. Registration between any two of these coordinate systems shall either be computed directly or obtained by composition of two or more known registrations.
- 4.4.5. The system shall provide implementations of the following coordinate computation methods (for calibration and/or registration):
 - 4.4.5.1. A “pivot calibration” method, where the transformation is computed from data obtained by pivoting a tool about a fixed position.
 - 4.4.5.2. Paired-point rigid registration, using a method such as the one proposed in Reference 2.2.1 or in Reference 2.2.2 (enhanced in Reference 2.2.3).

System Requirements for Surgical Assistant Workstation (SAW)

4.4.5.2.1. This method may be used to register the robot to the external measurement device.

4.4.5.2.2. This method may be used to register the preoperative image or model to the robot or external measurement device.

4.4.5.3. Iterative methods for registering sets of points/surfaces. An example is the Iterative Closest Point (ICP) algorithm proposed in Reference 2.2.5.

4.4.5.3.1. This method may be used to register prior models to the anatomy without requiring fiducials

4.4.5.4. Solving the matrix equation $AX = XB$, where A and B are known and X is unknown.

4.4.5.4.1. This method may be used for ultrasound calibration (see Reference 2.2.4).

4.4.5.5. Deformable (non-rigid) registration

4.4.5.5.1. This method may be used for video overlay on organs.

4.5. Tool tracking

4.5.1. The software shall be able to estimate tool positions/orientations using a combination of stereo video images and joint encoder feedback.

4.5.2. The software shall be able to predict the position and shape of tools in the 3D frames (i.e., to provide input to the video rendering).

4.6. User Interface (Visualization)

4.6.1. The system shall contain hardware to display the captured stereo images. At a minimum, the following hardware shall be supported:

4.6.1.1. daVinci master console display

4.6.1.2. Head-mounted display

4.6.2. The software shall include a windowing system to:

4.6.2.1. Manage the display of multiple windows.

4.6.2.2. Render visual objects, including menus, buttons, toolbars, and images (see 4.6.3), in the 3D viewing space.

4.6.3. The system shall be capable of rendering the following images:

System Requirements for Surgical Assistant Workstation (SAW)

- 4.6.3.1. 2D images
- 4.6.3.2. 3D volume models
- 4.6.3.3. 3D surface models
- 4.6.3.4. 2D image projected onto a 3D plane (e.g., laparoscopic ultrasound)
- 4.6.4. The system shall be capable of performing image fusion, i.e., the ability to combine multiple images with different blending/overlay parameters.
- 4.6.5. The system shall support the manipulation of virtual objects in the surgeon's field of view, where the virtual objects may be 2D or 3D images or models, or widgets such as "in volume" menus and virtual push-buttons, and similar functions.
 - 4.6.5.1. The manipulation functions shall be designed to accept position/orientation inputs from a generic user input device. One embodiment shall include the daVinci master telemanipulators and associated surgeon console controls.
 - 4.6.5.2. The manipulation functions shall include controls for manipulating the virtual objects (e.g., repositioning, rotating, scaling) and for turning them on and off.

4.7. Telesurgery application framework

- 4.7.1. The application framework shall integrate all functions listed above.
- 4.7.2. The application framework shall contain an event loop to handle events from the subsystem components.
- 4.7.3. The application framework shall include a real-time data pipeline that can be used by video processing subsystem (4.2).
- 4.7.4. The application framework shall include the CISST Interactive Research Environment (IRE), which is a Python-based shell for interactive development.
- 4.7.5. The application framework shall allow users to dynamically load "plug-in" modules for research purposes.
- 4.7.6. The application framework shall have a modular architecture, allowing it to be implemented on different physical architectures (i.e., different boxes), within the performance (bandwidth and latency) limitations of the interconnections. Supported physical architectures shall include any of the following:
 - 4.7.6.1. Ultrasound acquisition on a separate computer.

System Requirements for Surgical Assistant Workstation (SAW)

- 4.7.6.2. Robot control on a separate computer (e.g., daVinci embedded controller or external JHU controller).
- 4.7.6.3. Collaborative robots controlled by different computers.
- 4.7.6.4. All functions specified in this requirement on a single PC.
- 4.7.7. The application framework shall include a flexible data logging mechanism to allow the recording of relevant state information, including video. This can, for example, be used for research in gesture recognition.
 - 4.7.7.1. The data logging shall support time synchronization, either by ensuring that all data are captured at the same time or by associating a system-wide “timestamp” with each data item.
- 4.7.8. The system shall provide a method to save an occasional “snapshot” of the state for recovery from system restart (e.g., due to power failure, computer crash, etc.).

4.8. Volume viewer

- 4.8.1. The volume viewer shall be implemented using the functions specified in Section 4.6.5.
- 4.8.2. The volume viewer shall provide functions for selecting and loading volume data sets from a menu of choices.
- 4.8.3. The volume viewer shall provide functions for scaling the data set from voxels to physical coordinates and placing it at a specified position within the stereoscopic visualization coordinate system (i.e., in camera coordinates).
- 4.8.4. The volume viewer shall provide functions for turning visualization on and off and providing “fused” visualizations by video blending (see Section 4.6.4).
- 4.8.5. The volume viewer shall provide functions to enable haptic interaction with volumetric data (Phase 2)
- 4.8.6. The volume viewer shall update the visualization to compensate for camera motion, so that the volumetric data set appears to be fixed to the tissue (Phase 2).
- 4.8.7. The interface between the input device(s) and volume viewer shall be as generic as possible to facilitate integration of third-party volume viewer software.

5. Performance Requirements

Note: It is assumed that the system shall have at least one periodic loop (“heartbeat”) that interacts with the hardware devices and/or proprietary device interface software. Many of the following performance requirements depend on the frequency of this periodic loop.

5.1. Robot API

Note: The following performance parameters are highly dependent on the particular robot that is used. The numbers cited below apply to the daVinci robot systems, but it is expected that other robot systems will meet or exceed these minimum requirements.

- 5.1.1. The minimum update rate for receiving state information (e.g., robot positions) from a read-only robot shall be 50 Hz (20 msec).
- 5.1.2. The minimum update rate for receiving state information (e.g., robot positions) from a read-write robot shall be 30 Hz.
- 5.1.3. The minimum update rate for commanding state changes (e.g., providing position goals) shall be 30 Hz, subject to physical constraints (e.g., robot must be able to reach target positions within update cycle).
- 5.1.4. The maximum latency between detection of a physical state change and availability of this information from the robot API shall be 100 msec.
- 5.1.5. The maximum latency between a commanded state change and the corresponding hardware output shall be 100 msec.

5.2. Video processing

- 5.2.1. The video capture frame rate shall be 30 frames per second.
- 5.2.2. The latency due to video capture shall not exceed 2 time frames (depends on video capture hardware).
- 5.2.3. The processed and overlaid video frame rate shall be at least 10 frames per second.
- 5.2.4. The latency due to video processing shall not exceed 1 time frame (e.g., 100 msec at 10 frames per second).
- 5.2.5. The stereo reconstruction resolution is a function of the baseline width (distance between cameras), depth (distance from the cameras), camera resolution, and focal length. For a baseline of 5 mm (worst case for daVinci endoscope) and camera resolution of 640 x 480, the following resolutions are obtained (results in mm):

	Focal length, pixels			
Depth, mm	700	800	900	1000
50	0.70	0.62	0.55	0.50
100	2.78	2.44	2.17	1.96
150	6.16	5.42	4.84	4.37
200	10.81	9.52	8.51	7.69
250	16.67	14.71	13.16	11.90

- 5.2.6. The registration error between 3D anatomic models and the live (video) image shall not exceed 1.5 times (150% of) the stereo reconstruction resolution, at a specified depth and focal length (see 5.2.5), not including errors due to organ deformation.
- 5.2.7. The initial registration shall require no more than 1 second of computation time.
- 5.2.8. The latency due to the visualization shall not exceed 10 msec (depends on video output hardware).

5.3. Other device interfaces

- 5.3.1. The minimum update rate for receiving information from other devices (e.g., force sensor, tracker, etc.) shall be 50 Hz (20 msec).
- 5.3.2. The latency of the force data (e.g., time between physical application of force and software reception of force measurement) shall be no more than 40 msec.
- 5.3.3. The latency of the tracker data (e.g., time between physical motion and software reception of new position) shall be no more than 100 msec.

5.4. Calibration and registration

Note: Calibration and registration performance requirements shall be specified in the application requirements document, rather than in this system (workstation) requirements document.

5.5. Tool Tracking

Note: The following specifications are for the daVinci stereoscopic endoscopes and the Intuitive tool tracking software implementation.

- 5.5.1. The update rate of Tool Tracking shall be no more than 200 ms.
- 5.5.2. The latency of Tool Tracking shall be no more than the update rate (the time to process 1 frame).

- 5.5.3. The position of an instrument shall be determined with an average accuracy of at least 4 mm, at a distance of 75-80 mm.

5.6. User Interface (Visualization)

None

5.7. Telesurgery application framework

- 5.7.1. The time synchronization between different components shall be defined by the minimum (slowest) update rate of the components (e.g., if the slowest component updates every 100 msec, the time synchronization shall be within 100 msec).

- 5.7.2. The system heartbeat shall be 20 msec or less.

5.8. Volume viewer

None

6. Safety Requirements

6.1. Robot API and Interface

- 6.1.1. The read-write Robot API shall include a software command to allow application programs to disable power to the robot motors.
- 6.1.2. The system shall provide a method for disengaging the research interface from any clinical robot capable of operating in a stand-alone manner (e.g., the daVinci) for emergency responses.

6.2. Video processing

None

6.3. Other device interfaces

- 6.3.1. The system shall include an “emergency stop” switch that disables power to the robot motors and any other potentially hazardous device.

6.4. Calibration and registration

- 6.4.1. The computation methods shall indicate the residual error, so that users can determine how much confidence to place in the result.

6.5. Tool tracking

None.

6.6. User Interface (Visualization)

6.6.1. The system shall provide a method for disengaging the research visualization output, so that a clinician can revert to the visualization provided by an unmodified clinical robot (e.g., a clinical daVinci).

6.7. Telesurgery application framework

6.7.1. The application framework shall periodically check all safety-critical subsystems (e.g., by verifying communication integrity) and initiate a safety response (e.g., using function 6.1.1) if a failure is detected.

6.8. Volume viewer

None.

7. Design Constraints

7.1. Operating System

The system shall be designed to operate on Red Hat Enterprise Linux WS 4. It is desirable for it to work with any type of Linux, with future extension to a real-time Linux such as RTAI.

7.2. Programming Language

The software shall be written in C/C++.

7.3. Software Libraries

7.3.1. The software shall use the CISST libraries.

7.3.2. The software shall use the Visualization Toolkit (VTK) for visualization of images. The software may optionally use other toolkits that build on, or extend, VTK.

7.3.3. The tool tracking module shall be based on existing code from Intuitive Surgical.

7.3.4. The daVinci research API shall be jointly evaluated by ISI and JHU to determine whether to add a requirement to port it to use the CISST operating system abstraction and real-time support libraries (for the thread that manages the data stream).

8. Change History

Rev	Date	Author	Description
1	1/3/07	P.K.	Initial version

System Requirements for Surgical Assistant Workstation (SAW)

2	1/29/07	P.K.	Specified performance requirements, other minor changes. Reviewed by project team.