Enhanced Surgical Simulation Sandbox for the daVinci System

Anand Malpani
Mentors: Anton Deguet Prof. Taylor, Simon DiMaio, Ashwin Suresh

May 14, 2013

Abstract

We are developing a software framework to enable robotic surgery simulation across multiple systems. Such a *sandbox* would enable researchers to develop new simulations for enhancing the training of a surgeon. We have used open source libraries like H3DAPI, CISST to implement the core of our sandbox. The component based framework of CISST makes it possible to have any manipulator based system controlling the simulation environment. We have developed a teleoperation component that acts as a communication server between the master device and the slave simulation. Additionally, we have also implemented the slave side kinematics for sending desired motion commands to the simulation. H3DAPI allows us to get access to the open source graphics and physics rendering engines under a unified framework. Developers create their own object models and plug them in the H3DAPI framework or can choose to use existing models provided. We demonstrate the framework using simple geometries and controlling complex objects like tools will be done in the future.

1 Background

1.1 Robotic Surgery

Since, it was first introduced in 2000, robotic surgery has spread widely across the hospitals in the world. It has been effective in certain procedures like prostatecomies, hysterectomies where it has become the 'gold standard' way of doing for these surgeries. The most commonly used system is the daVinci®Surgical System [8] (daVinci) from Intuitive Surgical Inc. (ISI), Sunnyvale, CA, USA. There are currently over 2500 systems in circulation around the world performing over 2 million procedures every year. The daVinci has been prevalent in urology and gynecology since its introduction. However, in recent years it has gained importance in the head and neck procedures also.

The daVinci system is a teleoperation based robot and has three important components - surgeon console (masters), patient side cart (slaves), and the vision cart (Fig.1). The master console had two Master Tool Manipulators (MTM's) and a stereo viewer with 3D display. The slave side has four teleoperated robotic arms. Three of these hold tools and are referred as Patient Side Manipulators (PSM's) and the fourth (Endoscope Control Manipulator (ECM)) holds a full HD capturing stereo endoscope. The vision cart processes the images captured from the endoscope and sends them to the stereo viewer on the master

Figure 1: daVinci S System setup - (from left to right) Surgeon Console (Masters), Patient Cart (Slaves), Vision Cart.

console. The vision cart also contains the *Core*, which performs all the state-management and teleoperation.

There are some clear advantages of using the daVinci system as compared to traditional surgery that it is minimally invasive and thus, the patient recovery time, blood loss, post surgery trauma are greatly reduced. Compared to laproscopic surgery, which is also a minimally invasive way of doing surgery, it has some advantages for the surgeons performing the operation like less stress, no inversion of motion, negligible damage at port of entry, depth perception, high dexterity and precise control of instruments.

1.2 Training

With the rapid growth in robotic surgery, training surgeons how to use and develop skills on the robot has been a challenge. There is a lack of standardization in training protocols. Additionally, unlike traditional or laproscopic surgery, the trainee and operating surgery are not adjacent to each other. ISI has recently launched the dual-console system which tries to overcome this limitation, wherein two surgeons can use the robot from two consoles and one can supervise or learn from the other. This is limited however, due to space and financial constraints. Thus, currently, training in robotic surgery is still in its earlier stages.

Institutions have developed training protocols (robotic fellowships) so that trainees can perform certain bench-top exercises to develop their robotic skills. However, as the robot

is a scarce resource and real-surgery usage is of higher priority, availability of the robot for such training is not always feasible. Additionally, to evaluate the skill of the trainees and to teach them the procedure, an expert surgeon is required to supervise them during the training. This is not preferred by most of the experience surgeons. Thus, program directors and trainees have recognized the need for an alternate way of training.

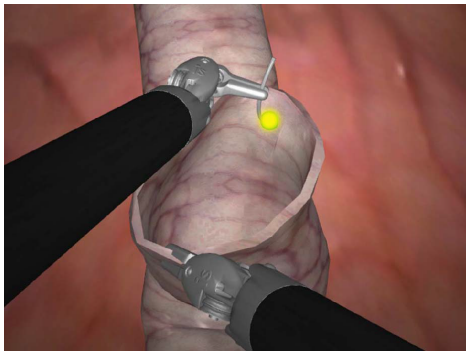
1.3 Simulation in Surgery

The need for a standardized training framework for robotic surgery was answered to by a variety of developers that came up with table-top or console based virtual reality simulators. This removed the need for recycling and a constant need for bench-top phantoms. Another advantage of simulators was the automated assessment of skills using information from the simulation. Thus, there was no or little need of supervision to access the improvement in performance of the trainees. However, the simulations are limited to certain anatomical exercises only. Developing anatomical simulation for procedural training is currently an active area of research for most of the simulation developers.

1.4 daVinci Skills Simulator [9]

There was a need for a simulator similar to the actual daVinci system and thus ISI launched the daVinci Skills Simulator (Fig. 2a). The Skills Simulator consists of the master console and the *Skills Simulator* backpack as seen in the figure. The patient side tools and endoscope are simulated (Fig. 2b) from the backpack onto the stereo viewer of the console. The console is the same from the daVinci system. Thus, the simulator occupies only a smaller space and can be dedicated for training as well as actual surgery. The system reports some skill metrics for the users based on their performance of the simulated task. Overall, the system has the advantage of repeatability and provides a standard platform for skill evaluation.

(a) Console with the backpack simulator



(b) Simulation Exercise - Tubes

Figure 2: daVinci Skills Simulator

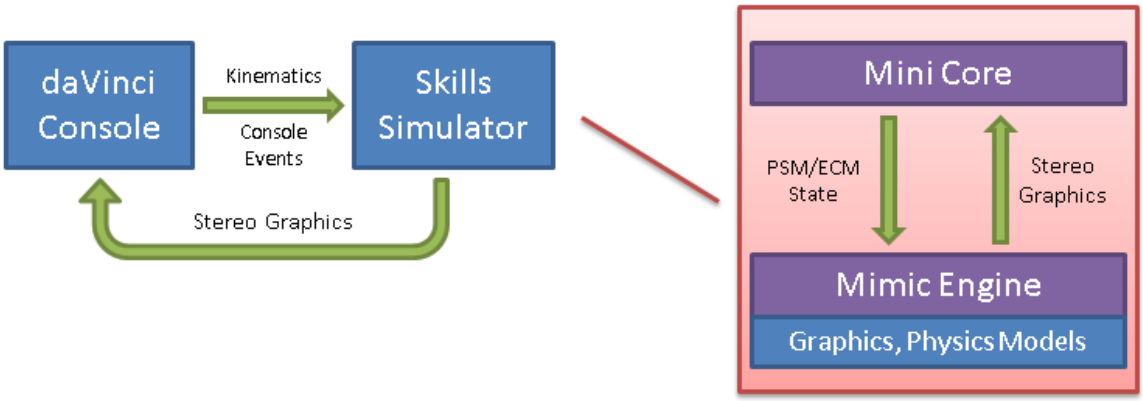


Figure 3: daVinci Skills Simulator Block Diagram

We will talk about the motivation and goal of our project in Section 2. Section 3 will explain the different components of our sandbox framework. Specifically, the different libraries are explained in Sec. 3.2. Following this, we will explain the information flow in the sandbox in sec. 3.6. Section 4 deals with the projects management details.

2 Need for Simulation Sandbox

2.1 Motivation

The skills simulator from ISI does a good job at simulating the actual robot and instruments. It can prove to be a great resource for many wider applications like surgical procedural planning, developing new instruments, user interfaces, modeling task performance for human machine collaboration, etc.

The skills simulator uses the *MSim2.0* simulation framework developed by Mimic Technologies Inc. [12] (who had developed a desktop simulator *dvTrainer* for robotic surgery). Thus, the skills simulator effectively relies on the Mimic engine for the graphics and physics simulation of instruments and other objects in the environment. As shown in Fig. 3, the master console sends the user inputs to the simulator backpack. Internally, the simulator backpack has a *Mini Core* (instead of the actual *Core*) to manage the state of the system and the kinematics. This core sends the desired configuration of the virtual tools and endoscope to the Mimic engine, based on which the graphics and physics are rendered back to the stereo viewer of the console.

However, the Mimic engine is a black box to a developer i.e. one does not have access to the models in the engine or the rendering pipeline. Thus, one cannot add their own simulation onto the skills simulator. This restricts the usage of the simulator to the training tasks developed by Mimic only. And, so the applications mentioned above cannot be implemented using this simulator. This is the motivation for our project which is defined below.

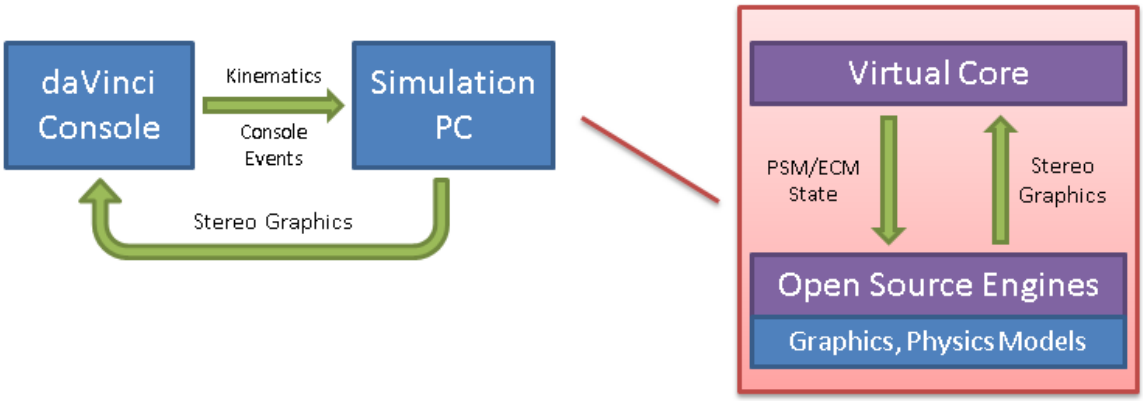


Figure 4: Simulation Sandbox for daVinci System

2.2 Goal

As mentioned above, the current daVinci Skills Simulator is a restricted platform for simulation development owing to the closed Mimic engine. However, one can imagine creating their own rendering engine to perform new simulations. We aim to create a *Simulation Sandbox* which can be used by developers to create their custom object models and environments using these new or existing models. The sandbox already would contain some models for a range of instruments used in robotic surgery, along with simple objects like rings, cubes, pegs, needles, etc. In order to be useful for other developers, we have used open source libraries and packages for the implementation of this sandbox (Sec. 3.2). Figure 4 shows the system block diagram for our proposed sandbox. It is similar in structure to the actual simulator. Instead of the simulator backpack we have our own computer running the sandbox application which contains a *Virtual Core* for mimicking the *Mini Core* from the daVinci simulator or the *Core* from the daVinci system.

2.3 Significance

As mentioned in Sec. 2.1, there are large number of applications for such a simulation sandbox related to developing newer technologies. A framework like this would enable the following applications, to name a few.

- Researchers can design their own simulation exercises for robotic surgery training using the existing library of object models.
- Developers can create and add new objects to the library as well as implement new applications using the same.
- Surgeons can plan procedures beforehand, using patient specific anatomical models.
- Surgeons can identify possible difficulties arising in a planned road-map for a procedure and improvise before performing the actual case in the operating room.

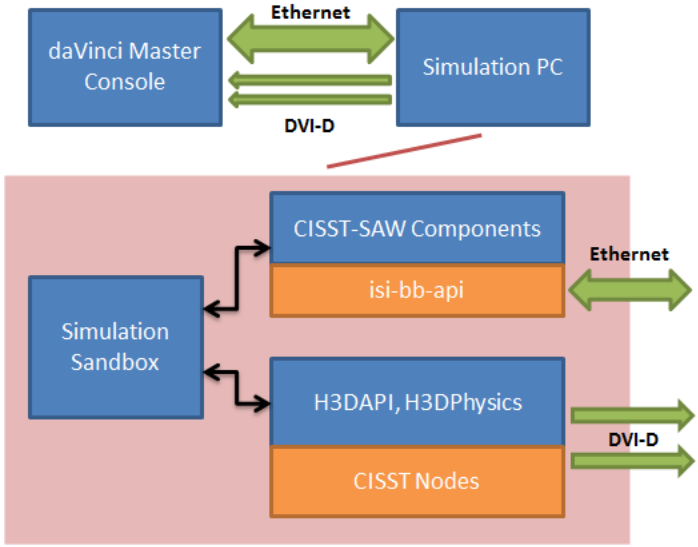


Figure 5: Sandbox System Block Diagram

- Sandbox can be a prototyping platform for testing new user interfaces, instruments, surgical procedures, image guidance methods for the daVinci.
- Researchers can have large source of reliable ground truth available from simulations due to access to the rendering pipeline for developing algorithms for modeling surgical tasks, tool tracking, and so on.

3 Technical Approach

3.1 System Block Diagram

3.2 Software Dependencies

The simulation sandbox aims at creating a platform for surgical simulation of the daVinci system using open source libraries. In order to create the simulation of the slave side of the daVinci system, we would need to be able to communicate with the daVinci master console. Such an access is provided to the framework using the ISI-API [1] which is a research interface to the robot (Sec. 3.2.1). However, one should not be restricted to the use of the daVinci master console. Thus, we chose to use the CISST-SAW (Sec. 3.2.2) libraries to be able to communicate to the sandbox using a variety of master devices, each having a component in the SAW framework. The sandbox was initially developed by Ashwin from ISI. He created models for one of the instruments and some objects, and used them to create a simulation task similar to the one present on the actual simulator. For this, he used the H3DAPI (sec. 3.2.3) framework for the graphics and physics rendering. We shall be building on top of these models, and thus, use the H3D framework for the rendering pipeline.

3.2.1 ISI-BBAPI [1]

This is the only component of the framework which is not open for user development. It is available in binaries from ISI on an agreement basis. There are three different types of API available from ISI for communicating to the daVinci system. We have a daVinci S console (S is the version of the system). The ISI-BBAPI provides a read and write access to the daVinci master console which is required for such a simulation framework. The sandbox can be used on the actual daVinci Skills Simulator also. In this case, we would use the ISI-SIM-API (present on the simulator backpack) to communicate with the master console.

3.2.2 CISST-SAW Framework [2, 3]

The CISST libraries have been developed for computer assisted interventions, while the SAW framework aims at enhancing the surgeon's capabilities through its components that interface with many of the computer integrated surgical devices (e.g. `sawIntuitiveDaVinci` for the daVinci system). The key aspect of the CISST-SAW framework is the component-interface model it is built on. The `cisstMultiTask` library contains the `mtsComponent` class that is the base class for creating new components. Additionally, `mtsTaskPeriodic` is a component that owns its own thread and runs at a given frequency. We would be using this component for the communicating across the components of the sandbox at regular intervals. A component can contain two types of interfaces to talk to other components - *provided interface* and *required interface*. Thus, a component "provides" information and event handlers for the other component's "required" information and event generators and vice versa. So, as long as one can develop a component for a master device containing certain interfaces required by the component for the virtual slaves, any master device can be used to communicate to the virtual slaves. We will talk about the different components developed for the sandbox in Sec. 3.4.

3.2.3 H3DAPI [5]

The H3DAPI is an open source scene graph API. It uses OpenGL (graphics library) for graphics rendering and HAPI (open source haptics library) for the haptics rendering. The H3DAPI can be interfaced at three levels - X3D, Python, C++. X3D is an ISO standard XML based file format for representing 3D computer graphics. The H3DAPI contains *fields* and *nodes*. Fields are basic building blocks of X3D, and are data containers. They are connected in a directed graph and update their values based on events passed through the graph. Their connections are called *routes*. Fields take care of the event handling and pass information over the outgoing connections on the graph upon receiving events from these connected fields. Nodes are containers of fields and their network, and help manage the entire field network and routes in an easier manner. The H3DAPI already contains X3D based nodes for graphics rendering as well as HAPI based nodes for haptics rendering. Along with this, the H3DPhysics toolkit is another package distributed with H3DAPI. The H3DPhysics toolkit is an implementation of the common physics engines like Bullet, ODE, Physyx, etc. using H3D nodes and fields. We talk about some custom nodes developed for the physics toolkit in Sec. 3.5

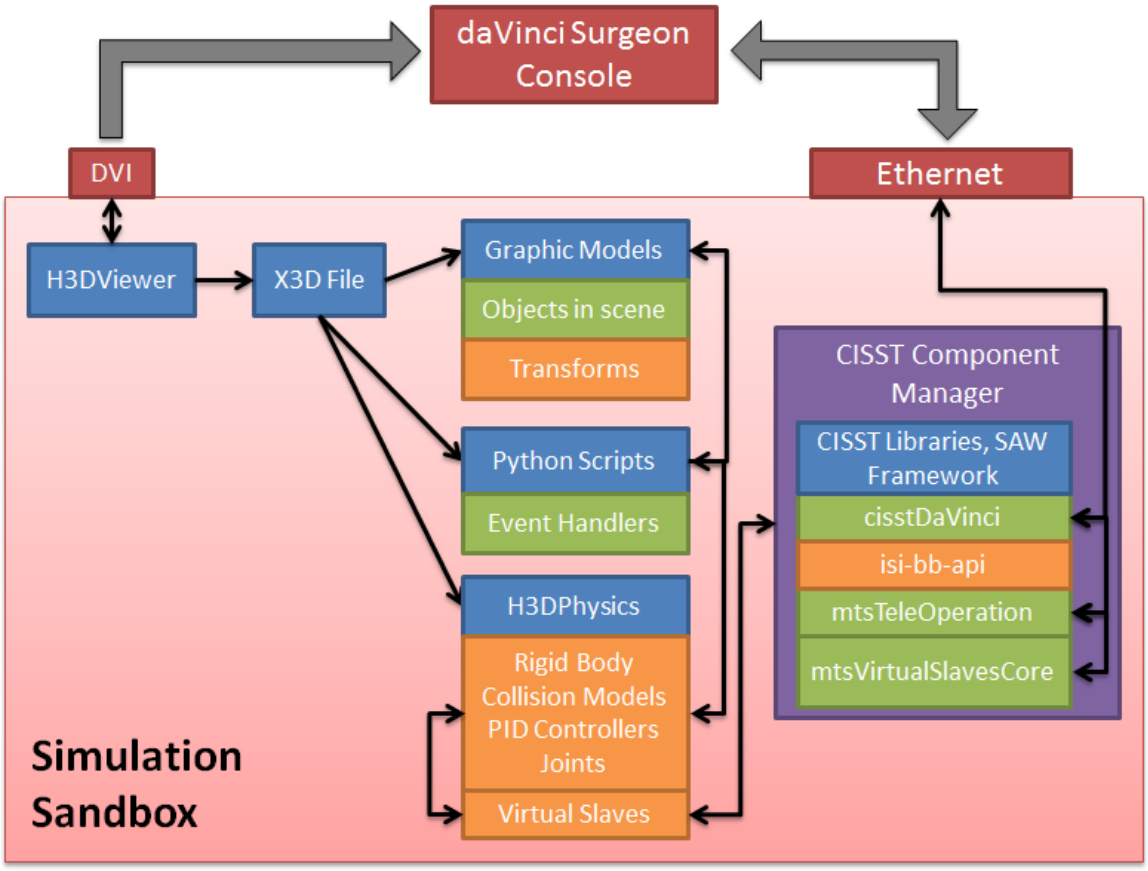


Figure 6: Sandbox System Architecture Diagram

3.3 System Architecture

The system architecture diagram for the simulation sandbox is shown in Fig. 6. We will explain the different components of the diagram below:

H3DViewer This is a GUI-based loader distributed along with the H3DAPI. It is able to parse the X3D files containing the scene graph information, and creates the nodes and field network needed for the requested scene-graph. The H3DViewer displays the rendering by the H3DAPI, and is connected to the daVinci master console via DVI cables to render on the stereo viewer.

X3D File This is the file describing the different nodes within the scene-graph. The nodes present in the scene graph serve different functions like graphic models, collision models, rigid body models, joints, PID controllers, etc.

Graphics Nodes These nodes describe the different objects in the scene and their rendering properties. X3D files for models can be developed from CAD files using some conversion

softwares (like CADExchanger). These X3D files can, then be used in the definition of the graphics nodes. We have existing X3D files for the different parts of the tools and simple objects like rings, letters, numbers, cubes, etc.

Python Script Nodes H3DAPI contains a special node for communicating with a python script. Different functions related to the simulation can be implemented in such a Python script. Values of the fields routed to such a script node can act as inputs for a function to determine certain attributes of the simulation, for example, completion of a surgical task.

H3DPhysics Nodes Nodes related to the physics simulation are present in the X3D file for the scene-graph for specifying various properties of the objects rendered using the graphics nodes. These nodes would be related to describing the collision properties of the objects as well as their surfaces, etc. There are nodes for describing the joints of the instruments and other objects present in the simulation, as well. Finally, our custom node for the virtual slaves implementation is also derived from the H3DPhysics nodes.

3.4 CISST-SAW Custom Components

For the sandbox to be able to talk to different devices it was important to build it using the CISST component framework. Following are the components that were developed for the sandbox:

cisstDaVinci(cdv)ReadWrite

- This is an existing CISST component derived from the `mtsTaskPeriodic` for communicating with the daVinci system using the ISI-BBAPI.
- We extended this component to include methods specific for simulation. These were needed so as to disengage the "real" slave from the master console and allow us to use the master as a stand-alone console.
- This component contains commands in the *provided* interfaces to read the state of the robot. These are triggered upon a function call from the *required* interface of another component.
- The component sends the current master cartesian positions as well as any console events by the user.
- The component also contains write commands to set the initial configuration of the master based on the slave simulation.

mtsTeleoperation

- This was developed using the `mtsTaskPeriodic` so as to talk to a master device using *required* interfaces to fetch the information about the motion of the masters and other events on the console.

- This component contains *provided* interfaces to send the desired cartesian positions to the slaves component whenever a function call is made on the slaves component side.
- This component contains *requried* interfaces to request the current cartesian positions of the masters as well as any console events.
- The component also receives information about the initial configuration of the master and slaves for the teleoperation to be aligned.
- The teleoperation can have certain states like CLUTCHED, FOLLOW, CAMERA, LOCKED, etc. and the state management for the teleoperation is also performed here.

mtsVirtualSlavesCore

- Again, this component is derived from mtsTaskPeriodic component. This was developed to emulate the *Core* on the slave side of the actual system.
- Like the *Core*, this component manages the slave side kinematics and sends information related to the desired joint positions of the virtual slaves to the H3DAPI nodes for simulation. The component uses the cisstRobot library to perform the forward and inverse kinematics.
- This component has *required* interfaces to request the slave side desired cartesian positions from the teleoperation component.
- The component also sets up the initial configuration of the tools and sends the information to the teleoperation component.
- This component will also be able to load different tools for different simulation tasks.

3.5 H3DAPI Custom Nodes

In order to model the slaves of the daVinci system we had to develop some custom nodes in the H3DPhysics toolkit. Specifically, the `VirtualSlaves` node was developed by us while the other were part of the initial sandbox from ISI (written by Ashwin). We describe these nodes below:

SevenDOF This is a simple node just for storing the different joint values of the slave side of the daVinci.

PID Nodes These nodes were developed to perform PID control of the joints of the slave side instruments in the simulation. These are: `PIDCollection`, `JointPID`, `PrismaticJointPID`, `RevoluteJointPID`. Both, the prismatic and revolute joint nodes are derived from the base class of `JointPID`. `PIDCollection` node contains instance of the joint PID nodes.

VirtualSlaves

- This was developed by us for communicating between the CISST components and the H3DAPI nodes. Thus, it was derived from both a H3DAPI node as well as a mtsComponent.
- This has *required* interfaces to request the desired joint positions of the virtual slaves from the `mtsVirtualSlavesCore` component.
- This also contains the `SevenDOF` nodes to pass the information to the `PIDCollection` nodes in the scene-graph.
- This component contains an instance of each of the custom CISST components and creates them when it gets constructed. It also adds all the components including itself to `mtsComponentManager`. The component manager performs and handles all the connections between the different components contained within it.

3.6 Information Flow

Now that we have described all the components and nodes involved in the simulation sandbox, we present the overall information flow that take place within the simulation sandbox here (Fig. 7).

1. The X3D file is loaded through the H3DViewer application which parses through the XML tags and creates all the *nodes* requested along with the routing between the fields of the different nodes.
2. Nodes for the graphics rendering of the object models like instruments are created along with the transform nodes to be applied to them.
3. Nodes describing the physics rendering of the object i.e. rigid body collections, collision models, joints are created and connected to each other as required. These nodes are also connected to the transform nodes of the graphics objects for rendering the appropriate changes.
4. Nodes for the PID control of the instruments are also created and routed to the `VritualSlaves` node created to fetch desired joint values for the instruments.
5. Finally, the custom `VritualSlaves` node is also created. Upon creation, this node internally creates the `cdvReadWrite`, `mtsTeleOperation`, `mtsVirtualSlavesCore` components and adds them to an instance of the component manager. The connections between all the components are also made here.
6. Once, the node and component creation is done, the virtual core sends the initial master and slave configuration to the teleoperation unit, which in turn sends the master pose to the master component. Upon successful initialization of the components to the initial configuration the individual components start running.

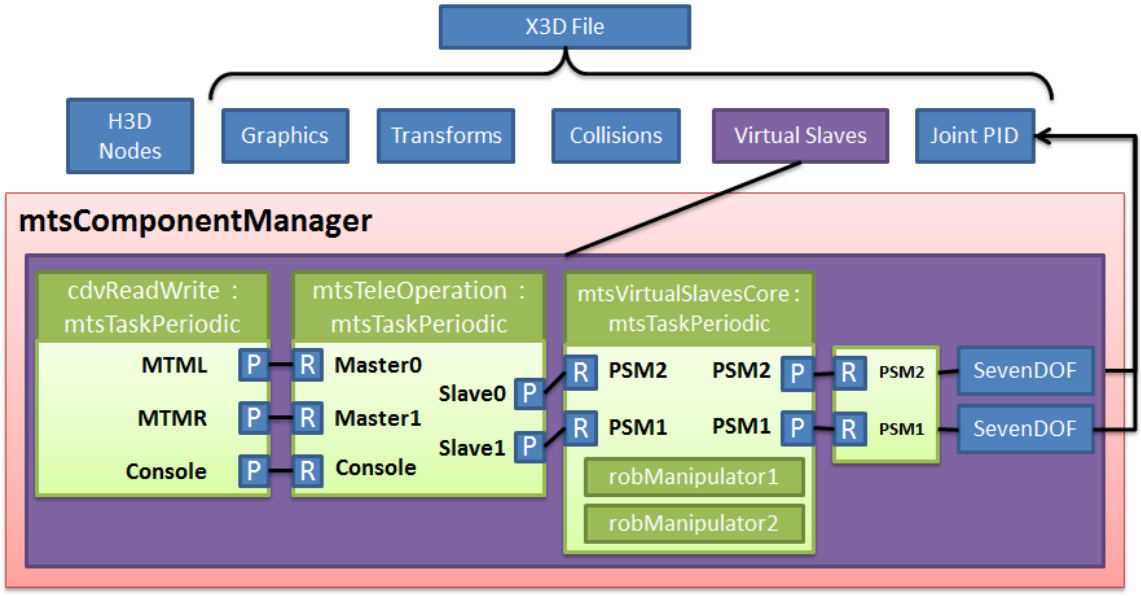


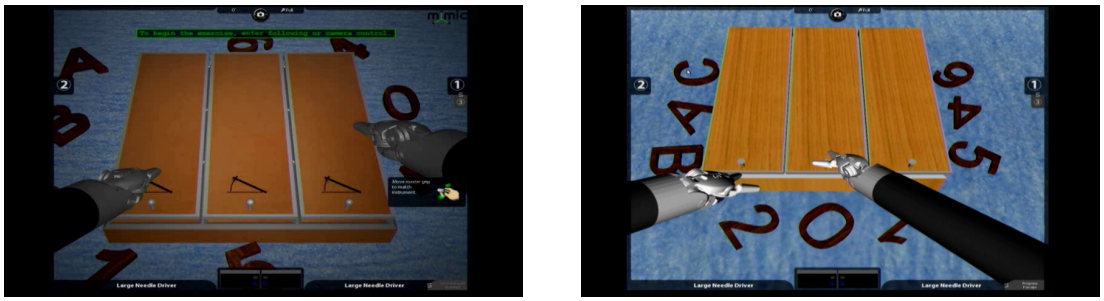
Figure 7: Sandbox Information Flow: 'P' - *provided interface*, 'R' - *required interface*, blocks in green are derived from CISST, blocks in blue are derived from H3D, blocks in purple are derived from both

7. Now, the `traverseSG` method of the `VirtualSlaves` node gets called at the refresh rate of the H3DViewer. Inside this, the required interfaces for the PSM's call their functions to fetch the latest values of the desired joints from the `mtsVirtualSlavesCore`.

8. Also, the other components have their `Run` methods which are called periodically at a pre-set rate. Thus, the virtual core component's requried interface calls the function to fetch the desired cartesian positions of the slaves from the teleoperation unit. Then, it uses the cisstRobot objects and stored DH parameters to compute the desired joint values for the virtual slaves.

9. Similarly, the teleoperation unit's `Run` method calls the function from the required interfaces to fetch information about the latest cartesian positions of the MTM's from the daVinci component. It then, uses this information and the current state to compute the desired slave positions and updates them. The teleoperation also handles events from the console and uses event handlers to change the state of the teleoperation from FOLLOW to CLUTCHED, for example.

10. Finally, the daVinci component, which uses the ISI-BBAPI to talk with the robot, gets the current positions of the MTM's and console events at a preset rate for the API.



(a) (b)

Figure 8: Snapshot of the Match Board exercise (a) Mimic simulation (b) Simulation Sandbox

4 Project Status and Management

4.1 Current Status

So far, we have been successful in creating the framework described above, except that the final step of sending the desired joints to the PID seems to be not working. Thus, we are unable to control the simulated tools. However, we have tested the components using simple geometries like cylinders and moving them around using the cartesian pose for the virtual slaves obtained from the teleoperation unit. The clutch and follow modes are currently functioning in the teleoperation unit. The camera control would be implemented in the future as it needs better understanding of the force constraints related methods of the ISI-BBAPI.

However, we were able to use the existing models developed by Ashwin on the daVinci Skills Simulator using the ISI-SIM-API. Figure 8 shows a comparison of the same exercise from the actual simulation on the daVinci simulator from Mimic (on the left) and the simulation using the sandbox models on the H3DViewer (on the right).

4.2 Deliverables

At the beginning of the project we had promised to deliver the following features in the project. Some of which we were able to meet and the rest will be met in the future.

- Minimum: (Expected by 14th April) **[DONE]**
  - Extend and develop the existing CISST-SAW component for interfacing with the ISI-BBAPI to include additional functions related to setup of the console for use as a stand alone simulator console.
  - Create a simple virtual slaves *Core* for emulating the actual *Core* of the slave side of the robot.
  - Create a new CISST component for interfacing to the simulation library (H3D).
  - Demonstrate the sandbox using a basic example with basic features.

- Expected: (Expected by 28th April)
  - Extend the virtual slave *Core* to incorporate other features of the daVinci console e.g. clutching, swapping instruments, camera control, etc. **[PARTIAL]**
  - Mimic the existing sandbox sample tasks from ISI to interface to the CISST components (modifying according to available features in the sandbox) **[DONE]**
  - Demonstrate the modified task developed by ISI using this extended sandbox on the daVinci S console in the Mock OR.
- Maximum: (Expected by 9th May) **[INCOMPLETE]**
  - Develop a new application involving new models using the extended simulation sandbox, perhaps demonstrating the implications of this framework.
  - Extend the sandbox for including features needed in the new applications developed.

4.3 Other Management Items

We were able to meet with all the dependencies for the project. We got delayed in implementation of the components of the project due to issues related to compilation of the software dependencies. We had trouble getting the correct version of the H3DAPI earlier for testing the existing models on the simulator. Once, all of this had been worked out, we hit another roadblock that ISI-BBAPI binaries were compatible for Microsoft Visual Studio 2008 (VS 2008) and not 2010. Thus, we had to switch to VS 2008 which meant recompilation of the dependencies using VS 2008. This led to another series of delays while getting the packages compiled and running. The important take-away was that one should always find the common denominator in projects like this which depend on a number of libraries and packages before starting to work one of the parts.

We met with our mentors from ISI every fortnightly and contacted via email in case we had questions for them. Meeting with Anton were on a weekly basis. We made many modifications to our timeline and had to cut down on our maximum deliverables as it was out of the timeframe for the project.

4.4 Project Resources

The wiki page for the project can be found here. The source code for the project is under version control here.

Acknowledgments Anton Deguet, our mentor from Johns Hopkins University has been very helpful in building this framework. It was his knowledge of system engineering that led to the current structure of the sandbox. We would like to thank our mentors, Ashwin and Simon from ISI for all the help, they have provided, and without their valuable insight of the daVinci simulator we would not have been able to have success in the project.

This project is funded through the Swirnow Family Foundation and Intuitive Surgical Inc. This would have not been possible without the gift from Richard A. Swirnow (JHU alumnus) which was utilized in building the Mock Operating Room where the daVinci

system is located. The daVinci S Surgical System and the daVinci Skills Simulator are on loan from Intuitive Surgical Inc. for research purposes.

References

[1] S. DiMaio and C. Hasser, *The da Vinci Research Interface*, MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Sep. 2008

[2] A. Deguet and R. Kumar and R. Taylor and P. Kazanzides, *The cisst libraries for computer assisted intervention systems*, MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Sep. 2008

[3] B. Vagvolgyi and S. DiMaio and A. Deguet and P. Kazhanzides and R. Kumar and C. Hasser and R. Taylorl, *The Surgical Assistant Workstation*, MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Sep. 2008

[4] Intuitive Surgical Inc., *ISI API User Guide*

[5] H3D.org, Open Source Haptics Library, http://h3dapi.org/

[6] ERC CISST, https://trac.lcsr.jhu.edu/cisst/

[7] Intuitive Surgical Inc., http://www.intuitivesurgical.com/

[8] daVinci Surgical System, http://www.intuitivesurgical.com/products/davinci_surgical_system/

[9] daVinci Skills Simulator, http://www.intuitivesurgical.com/products/skills_simulator/

[10] Intuitive Surgical Inc., *daVinci S System User Manual*

[11] Intuitive Surgical Inc., *daVinci Skills Simulator Manual*

[12] Mimic Simulation, http://www.mimicsimulation.com/