Gesture Controls for the Raven Robot

CS 446 Computer Integrated Surgery II

Project Final Report, May 13, 2013

Group 7: Alan Chancellor and Kristine Sarnlertsophon

Mentors: Kelleher Guerin and Anton Deguet

Project Summary

We will write software to control the Raven Surgical Robot with hand gestures. We will do this by integrating the 3Gear System, CISST libraries, the Robot Operating System (ROS), and the Raven itself. This will not only provide a new input to the Raven robot (which currently can only be controlled using Phantom Omnis, a specific piece of hardware) but also join many key libraries used in working with surgical robots.

Motivation and Significance

The Raven Robot is a surgical robot with open-source software, created by the University of Washington in Seattle. It has been sent to research labs across the country as a platform for researchers

to experiment with surgical robots. We aim to increase its usability by integrating the CISST libraries and a gesture control input; in doing so, we will create interfaces for:

1. 3Gear to CISST communication,
2. CISST library to ROS, and
3. ROS to the Raven robot (or Raven simulator).

We are using the Raven Robot as a vehicle to integrate these software libraries, and the goal of our project is to demonstrate this integration by controlling the Raven Robot using hand gestures. The integration of these systems will be of use to researchers working with surgical robots, even if they don't use the Raven robot. By enabling the 3Gear device to communicate using the CISST library, we open the doors for this more natural form of input to be used in other surgical robot systems. In linking the CISST library to ROS, we allow the users of ROS access to a well-developed library specifically for the development of computer assisted intervention systems. Finally, we are refining the Raven Robot's software to better integrate with ROS so that other researchers may further develop using the Raven.

Background

3Gear

The 3Gear system is a commercial product, currently in beta, which utilizes two Kinects (for their 3D cameras) to track a user's hand gestures, accurate to millimeters. 3Gear provides Java and C++ libraries which allow us to write our own hand-tracking applications. Their software passes

messages about the position, orientation, and pose of each hand, which we can use to control the hands on the Raven.

The 3Gear system (image source: futuristicnews.com)

CISST Library

The CISST package is an open-source collection of libraries designed to be used in computer assisted intervention systems. We will be extensively using the CISST Multitask library, which provides the component-based framework for the CISST package. Objects in the CISST Multitask framework include “provided interfaces” and “required interfaces,” which allow for communication between different components in a client-server manner. This allows for our code to be far more reusable; ideally, the CISST libraries we write will be able to plug into any input device with an appropriate SAW/CISST Multitask wrapper (such as the one we are writing for the 3Gear) and transmit the

appropriate data to ROS.

ROS

The Robot Operating System is an open-source operating system for robots. It provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. ROS is a distributed framework of modular "nodes," which can be put together to form an entire system. As such, ROS code is extremely reusable; we aim to take advantage of the nature of ROS and CISST Multitask to make the software we write for the 3Gear to Raven system generalizable for many inputs and robots.

Raven Robot

The Raven is a robotic surgery research system that uses open-source software based on ROS. It was developed by the BioRobotics Laboratory in the University of Washington in Seattle, and it has been sent to multiple research laboratories to enable further research in tele-operative and minimally invasive surgery. Although it has not yet been approved by the FDA, research laboratories have ambitious plans for the Raven, including operating on a beating heart (moving in sync with the heartbeats) and having the robot perform autonomously by imitating surgeons.

The Raven Robot (image source: University of Washington)

Goals

The overall goal of this project was to control a Raven II using hand gestures. This would be done in a manner that provides a proof-of-concept of integration of the university's CISST Multitasks [MTS] library with the popular Robot Operating System [ROS]. Our goals are broken down into the following milestones:

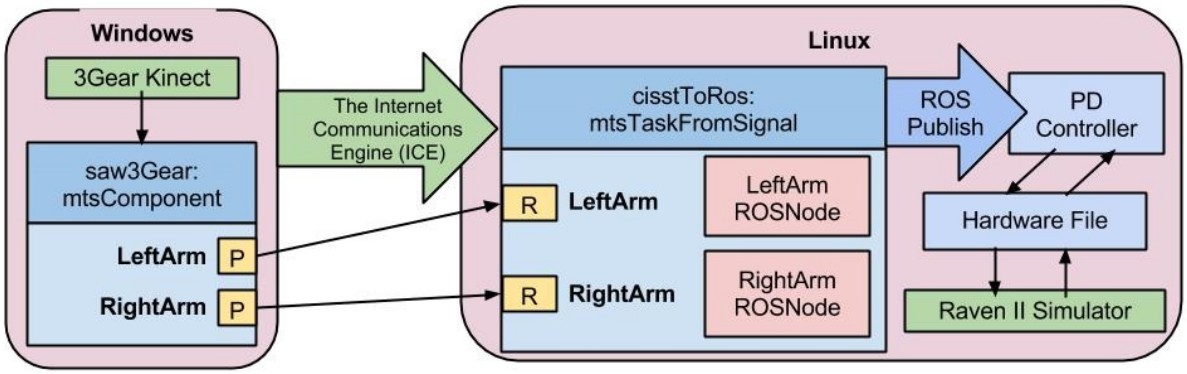
- Create a SAW wrapper for the 3Gear, which takes hand gesture data received by 3Gear and translates it into CISST data structures. (minimum deliverable)
- Create a CISST-ROS interface that takes CISST data structures and translates them into ROS data structures, and "publishes" them using a ROS command. (minimum deliverable)

- Demonstrate the 3Gear-CISST-ROS system by showing simple frames in a ROS visualizer controlled using hand gestures. (minimum deliverable)
- Integrate a Raven visualizer by showing a model of the Raven Robot controlled using hand gestures. This would involve writing a controller for the robot itself. (expected deliverable)
- Finally, control the Raven II robot using hand gestures. This should follow naturally from controlling the Raven visualizer, with the only additional work being dealing with any unexpected difficulties involving the Raven robot's actual hardware. (maximum deliverable)

Technical Approach

The components will be made to interface with each other via a component-based approach. This means writing a SAW (Surgical Assist Workstation; CISST package) wrapper for 3Gear that contains a provided interface which passes the CISST data to the next component, a CISST-ROS bridge that contains a required interface to receive the data. It was necessary to make CISST communicate between an iteration running on a Microsoft Windows machine, which runs the 3Gear input system, and an Ubuntu machine, which runs the ROS and the Raven or robot simulator; communication between the two computers was achieved using ICE, the Internet Communications Engine from ZeroC. The CISST-ROS component would "publish" the hand-tracking data it received to any subscribed ROS nodes. In order to move a Raven robot or a Raven simulator, a PD controller would have to subscribe to the data published by the CISST-ROS node and calculate the velocities at which to move each of the robot's joints (from the desired position data given to the system). The PD controller would communicate this information to a hardware file, which ultimately controls the robot or robot simulator

and keeps track of the robot's current joint states.



Our overall system-block diagram

saw3Gear

The SAW wrapper for the 3Gear acts as both a CISST Multitask component and a 3Gear hand tracking listener. The component itself has two provided interfaces, a left arm and a right arm. Each of these interfaces passes data about the hand's position and pose (the CISST data types used are prmPositionCartesianGet and prmEventButton, which only keeps track of whether the hand is "pressed" or "released." These data types correspond directly to data passed by 3Gear, which keeps track of the hands' positions and poses, in an event-based manner.

We created two versions of the SAW wrapper: one which simply logs the data into a CVS file using CISST (eventCollector), and another which passes the data to CISST's Global Component Manager. The second version is the one we use to connect to the CISST-ROS component. CISST's Global Component Manager tracks all of the CISST Multitask components that connect to it and allows components with matching provided and required interfaces to connect with each other. This networking layer is run using ICE, the Internet Communications Engine, which allows us to communicate between different machines (even remotely, although we did not utilize that capability in this project).

This enables us to go from the Windows machine running 3Gear's hand tracker and our saw3Gear program to the Ubuntu machine running the ROS portion of the code.

CISST-ROS

The CISST-ROS bridge connects to CISST's Global Component Manager over ICE. It converts the CISST data structures it receives (positions and button events) to the appropriate ROS library data types (ROS Transform "TF" positions and ROS boolean messages corresponding to button events). In this step, we not only converted data types from CISST to ROS but also changed the units of measure; ROS expresses positions in meters, while 3Gear expresses them in millimeters. The CISST-ROS bridge then publishes on a ROS topic (a string that specifies what data is being published) that broadcasts the TF frames and boolean messages to any subscribed ROS nodes.

When creating the CISST-ROS bridge, we chose to develop a new bridge rather than expand on an older bridge program. The older program used ROS Services rather than ROS Topics. ROS Services are a one-to-one communication method between ROS nodes, whereas ROS Topics are one-to-many. Another goal the CISST-ROS bridge accomplishes is bridging this difference in communications. CISST prefers one-to-one communications, in contrast to ROS, which default to the one-to-many topics.

ROS-Rviz

As the CISST-ROS bridge is broadcasting, one or mode ROS nodes have subscribed to the topic (ROS can control more than one robot with one controller). To visualize our system in a simple manner, we used ROS's built-in Rviz visualizer, which enables us to display published TF frames using a GUI. This visualization does not display the state of the buttons (our desired hand gripper positions), although

they are still being published.

ROS-Controller

In order to control a Raven robot or a Raven simulator, we had to write a robot controller which subscribes to the topics published by the CISST-ROS bridge. The controller is a ROS node that receives the desired position of the hand grippers and uses inverse kinematics packages to determine the required joint angles needed to correctly move the end effectors. It communicates (publishing on a ROS topic) with a hardware file, which sends messages to the robot's individual joints. The hardware publishes joint velocities to the robot (or robot simulator) and publishes current joint positions back to the controller. The controller uses the difference between the current joint positions and its calculated desired joint positions to calculate and publish the required joint velocities to the hardware file. Thus, the robot or simulator is controlled using a PD controller. This control scheme is identical for both the robot and the robot simulator.

Results

3Gear-CISST

The saw3Gear wrapper is fairly lightweight and sends position (both translation and rotation) data and "button pressing" data to the CISST Multitask provided interface. This is sufficient to move a surgical gripper, but a more robust program might send more types of poses tracked by 3Gear, rather than simple "mouse click" events. Regardless, the saw3Gear wrapper is a useful addition to the SAW libraries, allowing researchers to experiment with a new type of data input.

CISST-ROS

The CISST-ROS bridge relays positions and button events from the 3Gear wrapper to the ROS world in a timely manner. In its current state, the required interfaces are hard-coded, so it will only work with the saw3Gear wrapper's provided interfaces. Nonetheless, it serves as an early prototype for future more robust bridges, and it demonstrates the usability of CISST-ROS integration.

ROS-Raven II

We were not able to use the 3Gear to control the Raven or the Raven simulator. The provided model of the Raven (which we needed to use to write the controller for either the simulator or the robot itself) had multiple issues with the inverse kinematics calculations and incorrect or nonexistent transformations between the robot's different linkages and the robot base.

For our demonstration, we will show our minimum deliverable, moving simple frames in Rviz, and attempt to substitute a different robot (the Merlin) to control. Although we were not able to use the Raven, the ability to easily substitute a different ROS-controlled robot further demonstrates the merits of CISST-ROS integration.

Conclusion

Overall, the project was fairly successful. We met all of our minimum deliverables, adding a new and unique SAW wrapper to the 3Gear library and demonstrating the feasibility and usefulness of the CISST-ROS integration. Although we were not able to move the Raven or Raven simulator, we would easily be able to move other robots or robot simulators that correctly communicate with ROS. Future work on this project would expand upon the CISST-ROS bridge, making it more flexible by dynamically populating its required interfaces and publishing the appropriate ROS data structures, rather

than hard-coding the required interfaces as we have done here.

The challenges we faced in completing this project stemmed mainly from the many diverse libraries we were attempting to integrate. In writing the programs, we learned the to write interfaces for 3Gear, CISST, and ROS, and we also learned to use tools such as CMake and ICE. One significant issue we faced was the compilation and linking of all these libraries (3Gear, CISST and ICE on Windows, and CISST, ICE, and ROS on Ubuntu) on different machines. Only when these issues were resolved could we run and test the system.

Future Work

One potential area for expansion is a more robust CISST-ROS bridge that accepts multiple masters and slave robots. Currently, the 3Gear-CISST wrapper is hardcoded in the bridge, as is the use of the TF and Boolean Message data types.

This project opens up an avenue for studying gesture control interfaces for surgical robots. The most obvious area of follow-up is studying the use of the 3Gear's ability to replicate a more established control system's functionality on simulated surgical procedures. This is valuable as these basic gestures are an extremely intuitive interface. Another potential avenue is partial autonomy. Previous masters require a physical controller which limits the amount of input commands, and restricts the mode to full manual control. A hand gesture interface could be used to give partial autonomy to the robot by having the surgeon input in gestures rather than control commands. The program would match the gesture against a set of known gestures, a functionality the 3Gear already possesses, and move and complete simple commands autonomously.

Management Summary

Working with Anton Degeut and Kelleher Guerin, we were able to achieve our minimum deliverables. Kristine made the Surgical Assistant Workstation (SAW) wrapper for the 3Gear with Anton's guidance. Alan learned to run the ROS simulator with the assistance of Kel. Alan and Kristine collaborated on the 3Gear bridge, again with Anton's guidance. In addition, we were able to partially achieve our maximum deliverable of controlling a robot with our system. While we were not able to control the Raven robot or Raven simulator, we will be able to control the Merlin Robot with our system. The PD controller and hardware files for the Merlin robot have already been written for us by Kel.

Throughout most of the project, any code written was pushed to a private repository on Kristine's Bitbucket account. At Professor Taylor's request, we switched to using the Robotorium's SVN to keep track of our programs. Our source code and documentation can be found here: https://svn.lcsr.jhu.edu/robotorium/trunk/apps/3gear/

Documentation of progress and presentations concerning this project are on our course webpage: https://ciis.lcsr.jhu.edu/dokuwiki/doku.php?id=courses%3A446%3A2013%3A446-2013-07%3A446-2013-07. Of note, we have created User Guide and Installation Manual, available on our course webpage under "Reports and Presentations."

Reading List

- http://www.threegear.com/technology.html

- http://www.ros.org/wiki/
- https://trac.lcsr.jhu.edu/cisst
- M.J.H. Lum, J. Rosen, T.S. Lendvay, M.N. Sinanan, B. Hannaford, 'Effect of Time Delay on TeleSurgical Performance,' IEEE International Conference on Robotics and Automation (ICRA), 2009.
- M.J.H Lum, J. Rosen, H. King, D.C.W. Friedman, G. Donlin, G. Sankaranarayanan, B. Harnett, L. Huffnam, C. Doarn, T. Broderick, B. Hannaford,'Telesurgery Via Unmanned Aerial Vehicle (UAV) with a Field Deployable Surgical Robot,' Proceedings, Medicine Meets Virtual Reality (MMVR), Long Beach, CA, 2007.
- Staub, Christoph, et al. "Implementation and evaluation of a gesture-based input method in robotic surgery." Haptic Audio Visual Environments and Games (HAVE), 2011 IEEE International Workshop on. IEEE, 2011.

Appendices

- User's manual and Installation Guide: https://ciis.lcsr.jhu.edu/dokuwiki/lib/exe/fetch.php?media=courses:446:2013:446-2013-07:usersmanualandinstallationguide.pdf
- Source Code: https://svn.lcsr.jhu.edu/robotorium/trunk/apps/3gear/