Interfacing the da Vinci Research Kit (dVRK) with the Robot Operating System (ROS)

Zihan Chen*, Anton Deguet*, Steve Vozar*, Adnan Munawar†, Greg Fischer† and Peter Kazanzides*
* Johns Hopkins University, Baltimore, USA 21218
† Worcester Polytechnic Institute, Worcester, USA 01609

***Abstract*—To interface the da Vinci Research Kit (dVRK) with the Robot Operating System (ROS), we developed a software stack dvrk-ros, exposing the cisst-based robot control API to the ROS environment, providing Unified Robot Description Format (URDF) files for visualization, and offering kinematic as well as dynamic simulation. We also present some medical and non-medical use cases.**

I. Introduction

Robot Operating System (ROS) has become prevalent among robotics researchers and industry. It provides a large set of libraries and utility tools and enables inter-process communication between robot control code in one computer or across multiple computers [1]. To leverage the tools available in the ROS environment and allow ROS users fast prototyping of high level control and application logic in ROS nodes, we developed components that publish the robot state in ROS messages and accept commands by subscribing to ROS messages (topics). This paper gives a short overview of the da Vinci Research Kit and supporting cisst-SAW software stack (Section II), describes the cisst-to-ROS bridge, support for the catkin build tool system, and the ROS API for dVRK and simulation (Section III), and reports several use cases based on the dVRK ROS interface (Section IV).

II. System

The da Vinci Research Kit is based on proprietary mechanical hardware provided by the da Vinci Surgical System, coupled with open-source electronics and software (see Fig. 1). The mechanical system consists of two 7 Degrees of Freedom (DOF) Master Tool Manipulators (MTMs), two Patient Side Manipulators (PSMs), a Foot Pedal Tray and a High Resolution Stereo Viewer (640 x 480). The Foot Pedal buttons can be pressed to trigger different events that change the control behavior. Each manipulator arm is powered by one controller box with two sets of custom electronics boards, where each set consists of an IEEE-1394 (FireWire) FPGA control board and a Quad Linear Amplifier (QLA). All control boxes are daisy chained on an IEEE-1394 bus and connected to a Linux control computer. The FPGA board merely gathers sensor data, transmits them to the control PC, receives motor torque commands from the PC and latches them to the hardware. All computation, including servo-level control, occurs on the control PC. The component-based software system is based on the open-source cisst/SAW package [2], and includes components for low-level I/O, servo-level control, Cartesian mid-level control and teleoperation (see Fig. 2).

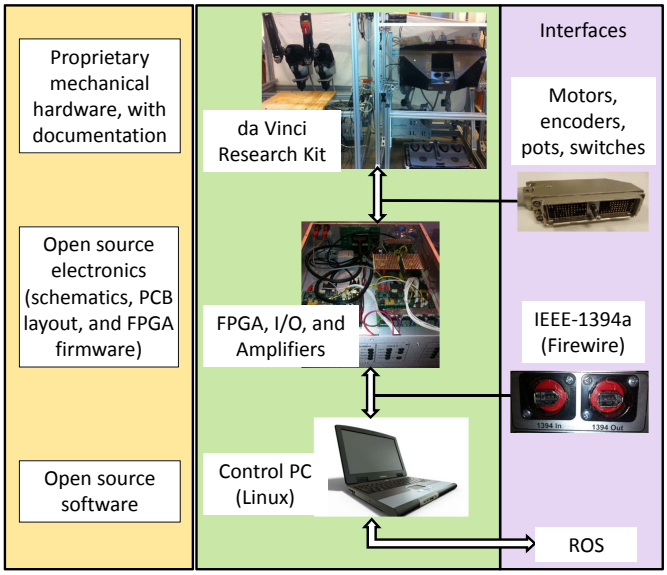


Fig. 1: Overview of telerobotic research platform: Mechanical hardware provided by da Vinci Surgical System, electronics by open-source IEEE-1394 FPGA board coupled with Quad Linear Amplifier (QLA), and software by open-source cisst/SAW package with ROS interfaces.

III. ROS Interface

To interface the dVRK with ROS, we developed a cisst-ros stack with components that publish the robot state as ROS messages and accept commmands from ROS messages. This implementation contains: (1) a set of global data type conversion functions (e.g. cisst Cartesian velocity to ROS geometry_msgs::Twist and vice versa), (2) a templated cisst publisher that fetches, converts, and then publishes the data, (3) a templated cisst subscriber with a ROS subscriber callback function, and (4) a cisst-to-ROS bridge component that serves as a container for cisst publishers and subscribers. The bridge runs periodically at a frequency specified by the developer. A separate ROS thread is created to handle the subscribers. In addition, MTM, PSM and ECM models have been generated in Unified Robot Description Format (URDF) and can be used for visualization and simulation (Fig. 3).

A. Build instructions and ROS API

To further leverage the ROS build system and ease the build process of the dVRK project, we use the catkin build system to build the entire cisst and dVRK software, as described in

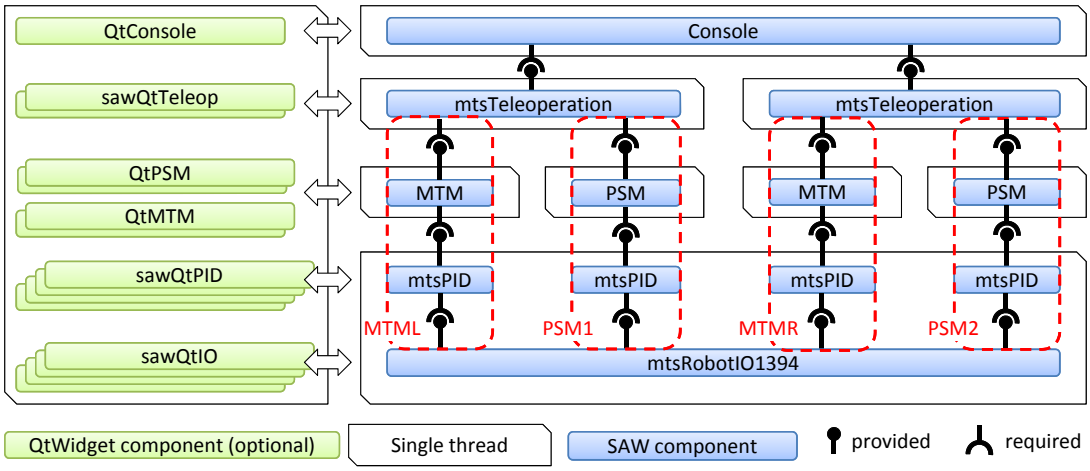


Fig. 2: Robot tele-operation control architecture with two MTMs and two PSMs, arranged by functional layers and showing thread boundaries

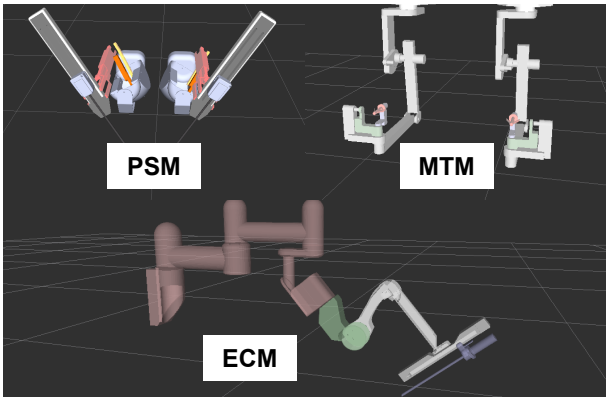


Fig. 3: Robot models displayed in RViz

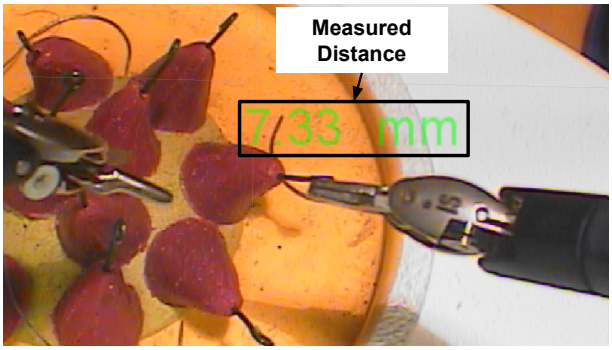


Fig. 4: Distance measured with PSM overlaid on live video

the catkin tools build instructions wiki page. Three groups of API have been exposed by the dVRK ROS API: (1) Servo level API with both position and torque read/write interface; (2) Cartesian (robot) level API providing Cartesian position read/write interface with or without trajectory interpolation; and (3) Foot Pedal API that publishes foot pedal events.

B. Simulation

In addition, the dvrk-ros stack provides a kinematics simulation for debugging and application simulation (e.g., ECM control). It also includes a Gazebo simulation for the MTM, capable of simulating its dynamic behavior, which is intended to develop various types of force controllers (impedance/admittance and stiffness).

IV. Use Cases

A. Augmented Reality: 3D Measuring Application

During minimally invasive surgery, measurements of tissue (e.g., during tumor resection) can be challenging due to magnification effects. But, this distance information is available via robot forward kinematics and can be overlaid on the live video. As shown in Fig. 4, a pinch event on the master triggers the measurement function and indicates the measuring start position. When enabled, a line from the start position to the current robot tooltip position is displayed close to the PSM instrument tip.

We focus on the vision pipeline here as the robot system was covered in Section II. The stereo vision system comprises two SONY lipstick cameras, connected to a SONY DXC-LS1 camera control unit, then a Hauppauge 610 USB-Live 2 Analog Video Digitizer in a control computer. The gscam ROS package, leveraging the gstreamer library, serves as the driver for these two cameras, which provides a ROS image pipeline compatible interface including the raw image topic and camera information. The augmented reality aspect of this application requires the camera intrinsic parameters through camera calibration using ROS camera_calibration as well as extrinsic parameters through hand-eye calibration using ROS aruco_ros and visp_hand2eye_calibration. After calibration, raw images from the cameras are rectified with image_proc and then displayed in RViz with a Camera display type. The distance text display moves with the patient side instrument tool, thus its position is defined with reference to the tool, transformed to the slave base frame and then to the camera frame. A visualization marker message is then created with distance information and published to RViz for visualization. This example will be released in the future as a ROS package in the dvrk-ros repository.

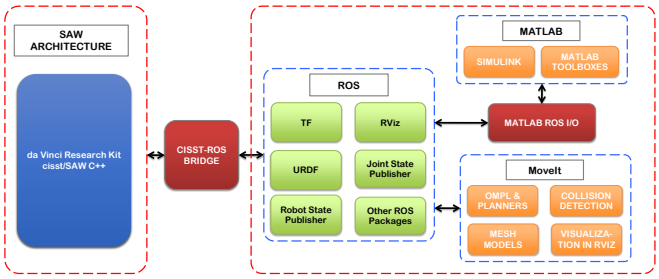


Fig. 5: Flowchart of the extended dVRK setup including MATLAB, Gazebo and MoveIt!

B. Motion Planning Framework for Surgical Assistance

Expanding on the capabilities of the cisst-ROS bridge, two additional components have been added using the cisst-ROS bridge. The first component is the MoveIt interface that allows motion planning for the dVRK manipulators [3]. The MoveIt! [4] package is a standalone package that is supported within ROS and contains state-of-the-art planning algorithms derived from the Open Motion Planning Library (OMPL). Using MoveIt, it is possible to use motion planning in static environments, as in Fig. 6 where the PSM manipulators are positioned in between the ribs of a model skeleton. The motion planners are used to compute a collision-free trajectory between specified start and goal poses and the computed path is graphically shown as a trail of PSM motions. The surgeon uses the MTMs to choose the start and goal poses, aided by the visual feedback of simulated PSMs and the surgical area.

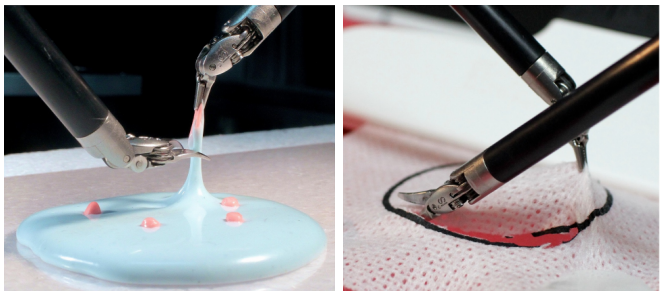
The second component is the newly developed Matlab-ROS interface, which expands the capabilities of the dVRK to researchers familiar with Matlab. As shown in Fig. 5, the MATLAB interface communicates over the Matlab-ROS IO, a newly added functionality by Mathworks, and can read/write data. Since development and testing is much easier in Matlab, this interface opens up many possibilities for researchers who are not familiar with C++, Python, or ROS.

C. Learning by Observation for Surgical Subtasks

Researchers at the University of California, Berkeley developed a system for learning by observation research [5] for surgical subtasks, including multilateral cutting of 3D viscoelastic and 2D orthotropic tissue phantoms (see Fig. 7). They explored a "Learning By Observation" approach where motion sequences and sensor conditions are identified, segmented and parameterized to build a finite state machine for each subtask. As shown in Fig. 8, the dVRK ROS system provides a Cartesian level API that takes a Cartesian pose command from the high level controllers implemented in the ROS environment.

D. Satellite Servicing

The dVRK is also used for non-medical robotics applications, such as a ground-based demonstration of telerobotic satellite servicing tasks, including refueling [6]. In the envisioned scenario, human operators on Earth would use the dVRK to control a remote on-orbit robot. One critical task is



(a) (b)

Fig. 7: Autonomous multilateral surgical subtasks with the da Vinci Research Kit (dVRK): (a) Debridement of 3d viscoelastic tissue phantoms in which small target fragments are removed from a phantom. (b) Pattern cutting of 2D orthotropic tissue phantoms in which the objective is to cut out a specified circular area. Figure courtesy of authors of [5]

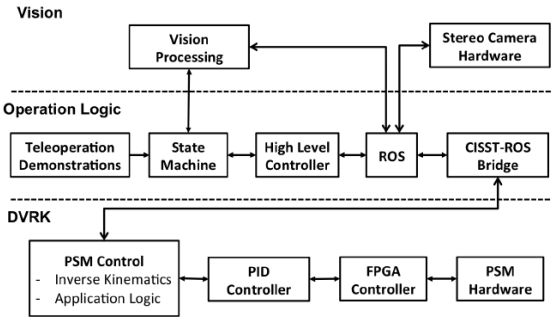


Fig. 8: Software Architecture. The software consists of three components: vision, operation logic, and the DVRK system software. Figure courtesy of authors of [5]

to cut the tape that secures a patch of multilayer insulation (MLI) over the fuel access port. We use the dVRK and cisst-to-ROS bridge extensively for developing and testing methods for improving the capabilities of human operators to perform this task under time delays of several seconds [7].

Our ground-based setup uses the right MTM of the dVRK to control a 7-DOF Whole Arm Manipulator (WAM, Barrett Technology, Inc, Newton MA) that contains a six-axis force-torque sensor (JR3 Inc., Woodland, CA) mounted between the WAM wrist and the titanium cutting blade (NASA Goddard Space Flight Center), as shown in Fig. 9. The force sensor provides cutter contact force information and enables active force-control in the direction normal to the MLI. A stereo camera pair is also mounted on the robot end-effector to provide visual feedback. Augmented reality overlays, generated with ROS rviz, are added to the stereo video feedback and displayed to the user via the dVRK master console using the "Camera" display plugin (see Fig. 10).

The master robot is controlled using the ROS and cisst/SAW open-source robot software systems, while the slave robot is controlled using the Orocos Real Time Toolkit [8] and ROS. Communication between the master and slave robots, including video and data, is accomplished over a local area network via ROS messaging. We induce constant delays on

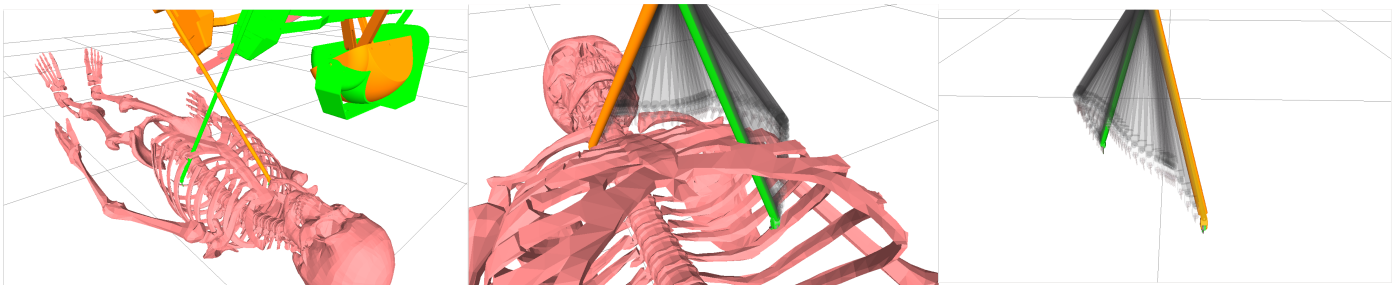


Fig. 6: Figures shows different views of the start (orange) and goal (green) poses of the PSMs positioned in between the model skeleton. The trail shows the resulting collision-free path produced by the motion planning algorithm.

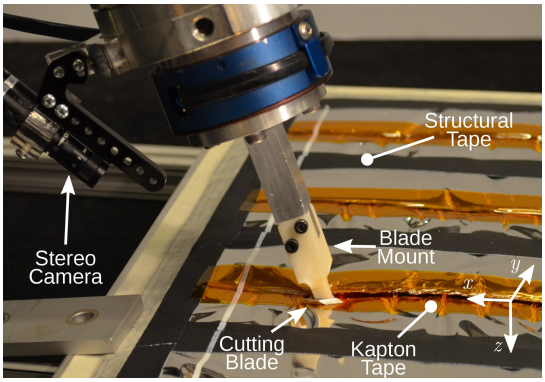


Fig. 9: Cutting test setup for satellite servicing demonstration

Fig. 10: Augmented reality display for satellite servicing, showing cutter goal position (green chevron) and virtual fixture path constraint (blue line)

the order of seconds in our experiments via a ROS message filter to simulate space teleoperation conditions.

V. Dicussion and Conclusion

We presented a software stack that helps to interface the da Vinci Research Kit with the Robot Operating System (ROS) and showed a few research uses cases developed in different universities and for both medical and non-medical applications. During the course of the work, we learned that as the community grows larger, the compatibility and stability of the system, including both hardware and software, have become critically important. On the hardware side, we found that due to chipset design issues, some FireWire cards drop packets and can potentially reduce controller reliability, especially when using a broadcast protocol for both reading and writing. Another issue is support for new releases of the Linux operating system, ROS, and other software dependencies. For example, Ubuntu 14.04 LTS uses Qt5 by default; this includes some API changes and required updates to our CMake files. ROS adopts a similar release protocol and can introduce compatibility issues in the future. The ROS stack presented above was developed and tested under the ROS Hydro distribution. Successful use of the ROS Indigo distribution has been reported in the community. Going forward, we plan to support Ubuntu LTS releases and compatible ROS distributions.

Acknowledgment

This work is supported by NSF NRI 1208540, NASA NNG10CR16C, and by the dVRK Consortium.

References

[1] 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.

[2] M. Y. Jung, M. Balicki, A. Deguet, R. H. Taylor, and P. Kazanzides, "Lessons learned from the development of component-based medical robot systems," *Journal of Software Engineering for Robotics*, vol. 5, no. 2, pp. 25–41, 2014.

[3] Z. Zhang, A. Munawar, and G. S. Fischer, "Implementation of a motion planning framework for the davinci surgical system research kit," in *The Hamlyn Symposium on Medical Robotics*, 2014, p. 43.

[4] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 1, pp. 18–19, 2012.

[5] A. Murali1, S. Sen, K. B., G. A., M. S., P. S., B. W.D., L. S., A. P., and G. K., "Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms," in *IEEE Intl. Conf. on Robotics and Auto. (ICRA)*, May 2015.

[6] National Aeronautics and Space Administration Goddard Space Flight Center, "Robotic refueling mission," 2015. [Online]. Available: http://ssco.gsfc.nasa.gov/robotic_refueling_mission.html

[7] S. Vozar, S. Leonard, L. L. Whitcomb, and P. Kazanzides, "Experimental evaluation of force control for virtual-fixture-assisted teleoperation for on-orbit manipulation of satellite thermal blanket insulation," in *IEEE Intl. Conf. on Robotics and Auto. (ICRA)*, May 2015.

[8] 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)*, vol. 2, Sep 2003, pp. 2766–2771.