A Software Framework for the Development of Collaborative Robotic Surgery : Robotics Toolkit for Collaborative Work.

Gunjan Bhandari, Sanskar Ghosh, Dimpal Saini and Anmol Jain

November 20, 2022

A software framework for the development of collaborative robotic surgery : Robotics Toolkit for Collaborative Work.

Gunjan Bhandari
Computer Science And Technology
Quantum University ,Roorkee
Saharanpur,Uttar Pradesh
Bhgunjan.05@gmail.com

Sanskar Ghosh
Computer Science And Technology
Quantum University ,Roorkee
Patna,Bihar
prince990ghosh@gmail.com

Dimpal Saini
Computer Science And Technology
Quantum University ,Roorkee
Saharanpur,Uttar Pradesh
Sainidimpal760@gmail.com

Anmol Jain
Computer Science And Technology
Quantum University ,Roorkee
Uttar Pradesh
anmoljainshab@gmail.com

***Abstract*—In the past few decades, robot-assisted minimally invasive surgery has made a significant impact in operating rooms due to its high dexterity, small tool size, and impact on the adoption of minimally invasive techniques. Medical robotics endeavors at numerous academic institutions and leading surgical robot companies have contributed to the development of intelligence and different levels of autonomy in surgical robots in recent years. To accelerate interaction within the research community and prevent repeated development, we propose the Collaborative Robotics Toolkit (CRTK), a common API for the RAVEN-II and da Vinci Research Kit (dVRK), which are both open surgical robot platforms installed in over 40 institutions globally. Other robots and devices have been added to CRTK, including industrial robots and simulations of robotic systems. In areas such as semiautonomous teleoperation and medical robotics, this API provides a community software infrastructure that can be used for research and education. The purpose of this paper is to present the concepts, design details, and integration of CRTK with physical robot systems and simulation platforms.**

Keywords—dVRK, CRTK, AMBF, MTMs, PSM

I. Introduction (*Background*)

Telerobotics and cooperatively-controlled robots both have compelling applications in surgery. In telerobotic systems, one or more robots are controlled by a master console by the surgeon. These systems can (1) For minimally invasive surgery, provide high dexterity through small incisions.(2) When acting on a patient while they are exposed to ionizing radiation, such as when performing computed tomography (CT) or x-ray imaging,(3) An MRI scanner's bore, for example, enables it to fit into confined spaces (4) Remote operations are performed by expert surgeons. As a result of cooperatively-controlled robots, the surgeon can share control of the surgical instrument with the robot. An embedded force sensor is usually used to measure the surgeon's intent to guide the surgical instrument. While many search robots and commercial surgery systems have been put in place to provide one or more of the greater than the advantages, to date, the da Vinci Surgical System (Intuitive Surgical, Inc., Sunnyvale, CA) (1) has achieved its greatest success with more than 5,000 robotic systems installed in hospitals worldwide and with more than 6 million surgical procedures executed. The da Vinci, however, only provides direct teleoperation , when a human surgeon individually controls the action of the robots on the patient side, even if the semi-independent teleoperation, in particular prudential supervision, shared supervision, and other co-robotic methods, has been active during decades. Furthermore, the da Vinci does not currently support teleoperation across large distances, though that capability was demonstrated in 2001 with a competing system(2) In the field of research, however, there is a trend towards the integration of semi-autonomous staff trained via device or Learning enhancement (ML/RL) in the surgical workflow. Some notable research in this field include autonomic algorithms for soft tissue suture.(3) a computerized approach to sinus surgery using computerized navigational techniques.(4) characterizing and automating soft tissue suture with a curved needle guide.(5) automation reduction/reduction of sub-tasks while utilizing apprenticeship ascertaining.(6) Additionally,(7) offers a holistic approach simplify the positioning of the handler before surgical interplay,(8) has demonstrated surgical simulation remote manipulator designed for cardiac surgery.(9) An infrastructure trainer is introduced with controllable domination and aggressiveness factors for the automation of repetitive surgical tasks. Finally, a common infrastructure for the training of apprenticeship agents through the decomposition of sub-task movements is developed in.(10) Naturally, these research systems cannot be applied directly to an actual surgical procedure, but rather rely on model configurations with custom robots or commercially available industrial robots. Researchers often have to make the tough decision to:(1) build a realistic experimental robot system (for example a custom robot) that matches the complexity and variety of tasks of the intended application, but is expensive and one of a kind or Choose a "tabletop" platform (e.g. an industrial robot) with less complexity and abundance of tasks, which is easier to develop but reduces the impact of research. . A few years ago it became clear that one of the obstacles to surgical robotics research was the lack of a robust and realistic common research platform. This led to the development and community adoption of two open research platforms: : the Raven-II and the da Vinci Research Kit (dVRK), described in the following sections. Currently these two robotic surgical platforms together have an installed base of about 50 systems in over 40 research centers (several laboratories use both systems) and several of the above publications were developed using one of these systems [5], [6], [9], [10]. In addition, simulations of the da Vinci robot have been developed, including commercial products used for training surgeons. In the field of research, da Vinci simulations include [11] and [12], the former based on Gazebo and the latter on V-REP. These typically simulate the Patient Side Manipulator (PSM) and use a human input device such as a mouse. B. da Vinci Master Tool Manipulators (MTMs) or haptic devices. Below we also summarize the recently developed Asynchronous MultiBody Framework (AMBF) [13] that supports the simulation of Raven II, dVRK and other robots. Simulators of the da Vinci robot have also been created, including products that are sold to train surgeons.Da Vinci simulations are found in [11] and [12], the former based on V-REP and the latter on Gazebo.The Patient Side Manipulator (PSM) is typically simulated by these, which make use of a mouse or other human input device.B. haptic devices or da Vinci Master Tool Manipulators (MTMs).The Asynchronous MultiBody Framework (AMBF) [13], which enables the simulation of Raven II, dVRK, and other robots, is described in detail below.

Fig 1. The Raven-II surgical robot

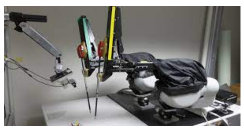


Fig 2. The da Vinci Research Kit

1) Raven II:The Raven-II (Fig.1) was developed in 2012 by the University of Washington and the University of California, Santa Cruz, and initially distributed to seven institutions as an-improvement to the previous Raven surgical robot.It was made to provide the necessary forces and range of motion in a small package with cutting-edge motion control.The Raven-II system's portability and durability have been demonstrated through experiments, including a simulated telerobotic surgery in a tent at a remote location north of Simi Valley, California; a gas-powered generator that powers the wireless teleoperation;also, shipping also, re-gathering the robot at the Aquarius undersea exploration station at a profundity of 19 meters off the bank of Key Largo (as a feature of NASA's NEEMO program).At the end of 2013, production of Raven II systems was outsourced to a new company called Applied Dexterity Inc., which has since installed a number of other systems.

2) dVRK: da Vinci Research KitAdditionally in 2012, Worcester Polytechnic Institute (WPI) and Johns Hopkins University (JHU) released open source software and mechatronics to enable researchers to construct dVRK platforms (Fig.2) [16] and [17] are examples taken from defunct da Vinci Systems of the initial generation .In particular, researchers could connect the cables that connect the da Vinci Master Tool Manipulators (MTMs), Patient Side Manipulators (PSMs), and Endoscopic Camera Manipulators (ECMs) to an open source controller that uses bridge linear amplifiers to drive the motors and field programmable gate arrays (FPGAs) to process the sensor feedback and control signals that are associated with them. These devices are known as da Vinci manipulators . Through the use of IEEE-1394 (FireWire), the FPGAs exchange all data with a control PC, resulting in closed-loop control rates exceeding 1 kHz.

3) Simulator for the Asynchronous Multi-Body Framework (AMBF):We recently proposed a simulation framework based on a front-end description format known as Asynchronous MultiBody Framework (AMBF Format) and an associated robust realtime dynamic simulator to address the requirements for a highly flexible simulation framework in terms of robot definitions, support for a wide variety of disparate input devices, and interactions with the environment [13].The AMBF Format permits:improved readability and editability for humans, distributed definition of the simulation elements, independent constraint handling, controllability of the way forces are applied to the bodies, communicability of all aspects of each body that are independent of one another, and dynamic loading with the capability to add bodies and alter constraints at run-timeBased on this AMBF format, the AMBF simulator offers soft body support, flexible visualization options, asynchronous support for a wide range of input devices used simultaneously without affecting performance, and dynamic body simulation.[18] describes a method for modeling the dVRK's dynamic model parameters, and [11] describes an example of using this framework to implement closed-loop kinematic chain mechanisms, which are difficult in many simulation environments.Section III-C provides additional information about how AMBF has been applied to the dVRK and Raven II.

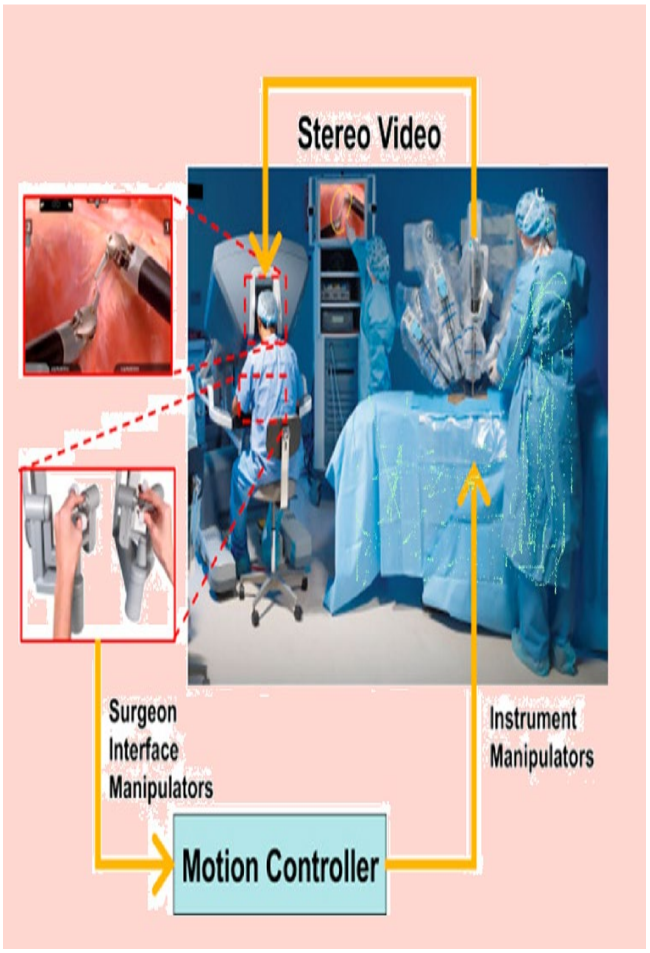


Fig. 3. Concept for Common API to Raven II, dVRK and other systems, and Surgical Tool Class to facilitate sharing of surgical instruments, especially those actuated by the four disks introduced by the da Vinci robot.

II. MOTIVATION

A. Even though the Raven II and the dVRK share a research platform, it became clear that the "one design" advocated by [19] would be even better.Because Raven II and dVRK are based on different hardware designs, it would appear impossible to develop a common software and hardware platform.However, since many Raven II robots drive the four-degree-of-freedom da Vinci instruments, the part of the robot that interacts with the environment is frequently the same for both systems.As depicted in Figure, this served as the initial impetus for the development of a common surgical tool class and software interface for Raven II and dVRK.3.However, it soon became clear that researchers in surgical robotics could use a similar software interface for other robots, so the goal was expanded to include defining a common "language" for component-based robotics software.One particular goal was to make it simple to replicate research on other robot platforms, like the ones mentioned earlier. We take into account two aspects of this universal language:1) the message content and the communication infrastructure that carries messages between the components.With abstract (implementation agnostic) definitions of the message content, our work focuses on the second aspect, presuming that one or more existing middleware packages can meet the first requirement. The widely used robotics operating system Robot Operating System (ROS) [20] is an obvious choice for the communication infrastructure.However, despite the fact that ROS provides common message types and standard middleware (such as topics and services), there is no welldocumented consensus standard for how these messages and interfaces should be used.Beginning users are able to quickly become proficient in ROS-based applications due to the low barrier to entry provided by ROS topics and the ROS communication structure as a whole. ROS is now regarded as the "community standard" middleware due to its widespread adoption and lack of enforcement of a messaging payload standard.However, the absence of a payload standard leads to the creation of redundant "wrapper" or "adapter" nodes for connecting ROS applications that have been independently developed.This was discovered at an early stage, and efforts were made to create some unofficial, but broadly accepted, payloads for particular applications.These messaging payload specifications have been widely adopted by the community, particularly when utilized by well-known ROS open source packages.However, despite their usefulness, these standards do not take into account the complexity of modern robots' hierarchical control structure.

III. . PROJECT OUTLINE AND SOFTWARE ARCHITECTURE

A. Project Outline The Collaborative Robotics Toolkit (CRTK) is a community-based software infrastructure made for research and education in cutting-edge fields like semi-autonomous teleoperation and medical robotics.The project development process consists of two steps:Technical Implementation and Community ParticipationCommunity engagement for medical robotics communities worldwide includes both in-person and online discussion forums.The technical implementation defines the CRTK structure and hierarchy based on user feedback and takes into account the community's comments and suggestions.

1) Community Participation:A major objective of the project is to involve the global community of software developers and researchers in medical robotics.We held workshops on "Shared Platforms for Medical Robotics Research," "Supervised Autonomy in Surgical Robotics," and "Open Platforms for Medical Robotics Research" at IROS 2017, "ISMR 2019," and "CRTK" and "Open Platforms for Medical Robotics Research" tutorials at IROS 2018. Joint editing of collaborative documents defining a set of use cases, naming conventions, and functionalities was the outcome of these workshops and tutorials.Some of the examples described in Section III were demonstrated during hands-on demonstrations, and Raven-II and dVRK robots were physically present at some of the events.

2) Technical Implementation: The authors gathered community ideas and defined use cases (Section II-B) through these community workshops and events, which they then used to define and modify the CRTK infrastructure.Weekly teleconferences were held by the authors, including developers of Raven-II and dVRK, to ensure that the ROS message payloads, such as frames, units, the API, and namespace usage, were consistently implemented across both robotic platforms.Python and C++ were used to implement example tests and interface scripts.Implementation status and solutions to devicespecific technical issues were discussed during these weekly meetings.

B. Use Cases The medical robotics research use cases we identified during our collaborative design process will serve as the foundation for the API's development.The use cases were divided into five themes by the authors.

1) Working remotely:Allow force information to be incorporated through bilateral teleoperation or force reflection and support teleoperation across various communication channels with various master and slave devices. 2) Motion by Oneself:interfaces that let researchers use Cartesian and joint space autonomous robot motion planners. 3) Custom Control and Kinematics:Allow researchers to implement advanced controllers that simultaneously solve kinematics and control, such as constrained optimization, for applications like optimizing kinematic redundancy or enforcing virtual fixtures. 4) Compliant or cooperative control:Provide capabilities for custom cooperative or compliant control, such as by attaching a force sensor to the wrist of the robot and driving it with measured forces. 5) Individual instruments:Facilitate the integration of custom instruments that provide capabilities like increased dexterity or additional sensing with the RavenII or dVRK for researchers.

A. SOFTWARE ARCHITECTURE

Standard conventions for the flow of command and feedback messages within a robotic system are the objective of CRTK.We believe that each message has a unique name and a payload attached to it.The payload is specified in a message description file (msg file, for example) in ROS, and the name is related to the name of the topic or service.ROS provides tools for parsing message files and creating software to convert messages to and from the target programming language's (C or Python) data types.However, other middleware, such as OpenIGTLink [22], could be utilized because CRTK is not restricted to ROS.

These two interfaces—the Robot Motion Interface and the Robot State Interface— were the primary focus of the initial development of CRTK.

1) Interface for Robot Movement:A robot's ability to move is arguably its most important feature, making it an obvious target for any standardization efforts.High-level motion primitives were traditionally used to program industrial robots, such as moving in a straight line to a desired pose.Medical robots that are teleoperated or cooperatively controlled (as noted in the use cases in Section II-B) also require a low-level motion interface, despite the fact that highlevel motions are relevant to medical robotics.A teleoperated robot, for instance, might require a continuous stream of position or velocity commands from the master manipulator to the slave manipulator.In a similar vein, some forms of cooperative control make use of a force sensor that is attached to the wrist of the robot and converts the forces it detects into a stream of commands for the desired velocity (called admittance control).The rate of command streaming must also be taken into account, as it may affect the slave robot's assumptions regarding motion smoothness and interpolation. As a result, as depicted in Figure, we define three levels of motion commands in CRTK.4.In a nutshell, the servo level is designed for low-level, high-rate robot control.This includes numerous use cases for cooperative control and teleoperation. In most cases, the robot should respond to the servo command as quickly as possible, preferably after performing some safety checks.Similar to the interpolate level, the robot should perform a straightforward interpolation to ensure smooth motion because the rate of setpoints may be slow or unreliable. Last but not least, the move level is for routine, high-level motion commands like moving into a certain pose.In this instance, the robot ought to have the capability of planning its trajectory.Table I's naming convention for CRTK motion commands must be followed, indicating whether the motion is in Cartesian or joint space and which motion parameter is being controlled (position, velocity, or force, for example). Additionally, the robot's motion-related information is depicted in Figure 4.It is possible to inquire about the current setpoint as well as the ultimate objective of the current motion in addition to the measured (sensor) feedback.

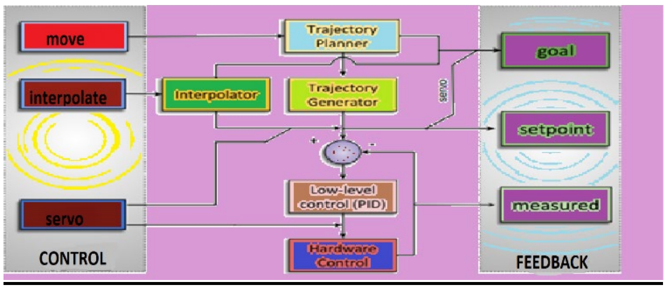


Fig. 4. CRTK motion commands: high-level move commands, mid-level interpolate commands, and low-level servo commands, which move robots in joint or Cartesian space based on various desired quantities, as defined in Table I. The arrows in the diagram represent data flow. Three types of inquiry are supported: measured, setpoint, and goal.

Keep in mind that the goal will be the same as the setpoint in the case of a low-level servo motion. It is vital to take note of that robots are not expected to support a wide range of movement orders in any case, on the off chance that an order is executed, it should follow the naming show in Table I. Likewise, it should likewise utilize the recommended payload (message type).The payloads for those commands are documented on the project website because we initially focused on the servo interface [23].The payloads for the move and interpolate commands are currently being defined by us. An interesting observation is that the higher levels could be implemented by generic software modules that interact with that level, so a standard servo interface might be sufficient.In point of fact, having only the joint space servo commands might be sufficient.This is comparable to ROS's strategy, which typically involves joint interactions with robots. However, the fact that it disregards any existing high-level implementations is the approach's drawback.For instance, all industrial robots offer the same move command in joint space and Cartesian space.Researchers can "wrap" these vendor-supplied capabilities, which may have been optimized for the particular robot, according to the CRTK naming convention and prescribed message type using the CRTK approach.In situations where the high-level functionality may not exist (such as custom robots) or when the wrapped vendor-supplied solution is deemed inadequate, a software-based solution can still be utilized.

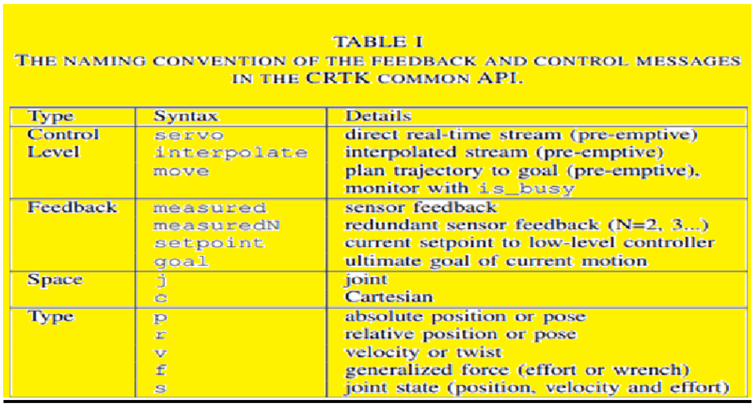


TABLE I
THE NAMING CONVENTION OF THE FEEDBACK AND CONTROL MESSAGES IN THE CRTK COMMON API.

Type	Syntax	Details
Control Level	servo interpolate move	direct real-time stream (pre-emptive) interpolated stream (pre-emptive) plan trajectory to goal (pre-emptive), monitor with is_busy
Feedback	measured measuredN setpoint goal	sensor feedback redundant sensor feedback (N=2, 3...) current setpoint to low-level controller ultimate goal of current motion
Space	j c	joint Cartesian
Type	p r v f s	absolute position or pose relative position or pose velocity or twist generalized force (effort or wrench) joint state (position, velocity and effort)

2) The Robot State Interface:Even though most, if not all, robot systems have operating states, it is impossible to try to create a state diagram that is the same for all of them.As a result, our primary focus is on creating a high-level "metastate" diagram that, as depicted in Figure 5, provides a summary of the operating states and the commands necessary to move between them. It's possible that a robot's various internal states correspond to these meta-states. Additionally, we define two operating modes that can be applied to any one of the meta-states.

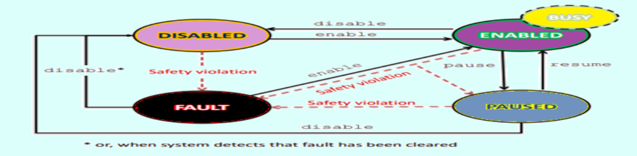


Fig. 5. CRTK robot meta-states and modes. The four meta-state queries are: is_disabled, is_enabled, is_paused, is_fault. The two operating mode queries are: is_busy and is_homed (not shown).

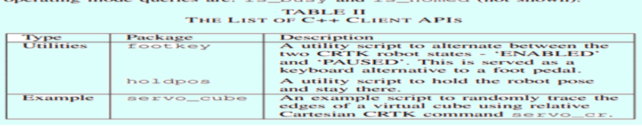


TABLE II
THE LIST OF C++ CLIENT APIS

Type	Package	Description
Utilities	footkey holdpos	A utility script to alternate between the two CRTK robot states - 'ENABLED' and 'PAUSED'. This is served as a keyboard alternative to a foot pedal. A utility script to hold the robot pose and stay there.
Example	servo_cube	An example script to randomly trace the edges of a virtual cube using relative Cartesian CRTK command servo_cr.

For example,the is_homed working mode shows whether the robot has been homed and applies to all of the meta-states.The is_busy operating mode, on the other hand, only applies to the is_enabled meta-state and indicates that the robot is currently executing a motion command.

B. *Client APIs*

On Raven-II, dVRK, the AMBF simulator, and other robots and devices in our laboratories, the authors implemented the lowest-level (servo) CRTK interface in 2018–19.Example interfacing scripts, also known as the Client APIs, are provided for users to modify or test on their robots in order to lessen the learning curve for new users.

- ROS Client API for C++:The authors created the crtk-cpp repository [24] to demonstrate how to use the CRTK interface. This repository includes (a) a library, (b) examples, (c) utilities, and (d) functionality tests.For improved readability and code compression, the rest of the package makes use of the library's fundamental CRTK API robot state and motion helper functions.The two types of C++ client APIs are utilities and examples.Section II-E will provide a more in-depth description of Client Test Scripts. Software packages called utilities are intended to be useful in a variety of robot control scenarios.As part of their research applications that enable robots to demonstrate CRTK functionalities, the authors envision community users directly downloading and utilizing the utilities (Table II), which are likely to be modified to meet future user requirements.

- ROS Client API for Python:The provision of a CRTK API that enables users to communicate with a ROS CRTK-compliant robot is the primary objective of the Python client API.The rospy package can be used directly in Python, but it can be difficult to learn.The ROS publishers and subscribers are hidden by the Python client API, the payloads are converted to more convenient data types like PyKDL frames and Numpy vectors and matrices, and blocking commands (for state changes and move commands) are implemented by using Python thread events.

- The Python client module provides methods to instantiate only portions of the CRTK standard because CRTK devices may implement different subsets of the CRTK specifications and the application may only require some of the CRTK features.Add_operating_state() and add_measured_cp(), for instance, would be used to only monitor a device's operating state and Cartesian position:

```
import crtk
import PyKDL

# instance of CRTK client
class custom_client:
  # configuration
  def configure(self, namespace):
    # add CRTK features needed
    self.utils = crtk.utils(self, namespace)
    self.utils.add_operating_state()
    self.utils.add_measured_cp()
```

-

- The user can later develop a customized Python client instance and use the CRTK feature measured_cp():

```
client = custom_client()
client.configure('/dvrk/PSM1')
p = client.measured_cp()
```

-

- Additionally, there are methods for waiting for CRTK state events in the Python client API.With wait_while_busy, for instance, the client can wait while the device is busy executing a move command.Examples and the most recent version can be found at [25].

C. *Client Test Scripts*

To demonstrate that a robot system correctly supports the API, we developed a set of standard client test scripts and written descriptions of expected robot behavior in parallel with the design of the CRTK API (Table III).Each test is performed on a for each arm premise.When the test is run, the robot namespace is sent as an input;ROS parameters are then loaded with robot-specific information like joint types, joint numbers, and home poses.The authors envision that all CRTK-compliant robots will be able to use the same test scripts to verify compliance with the CRTK API.

.EXAMPLES

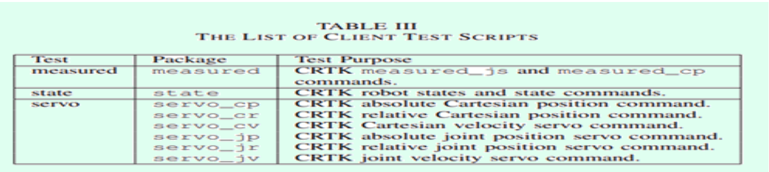


TABLE III
THE LIST OF CLIENT TEST SCRIPTS

Test	Package	Test Purpose
measured	measured	CRTK measured_js and measured_cp commands.
state	state	CRTK robot states and state commands.
servo	servo_cp	CRTK absolute Cartesian position command.
	servo_cr	CRTK relative Cartesian position command.
	servo_cv	CRTK Cartesian velocity servo command.
	servo_jp	CRTK absolute joint position servo command.
	servo_jr	CRTK relative joint position servo command.
	servo_jv	CRTK joint velocity servo command.

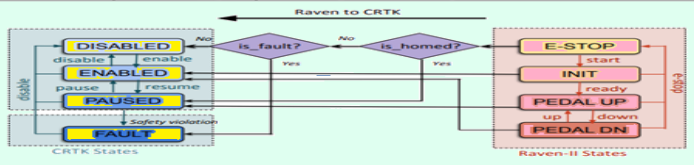


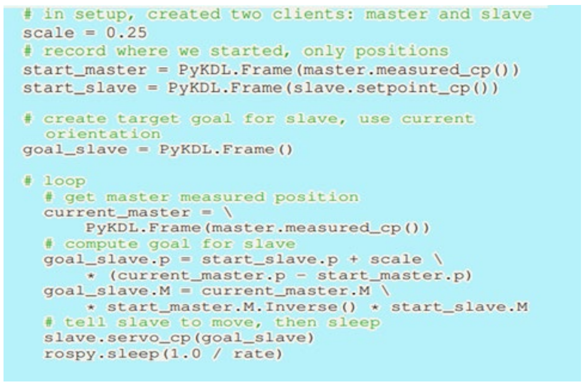
Fig. 6. The correspondence map between Raven-II states and CRTK states.

1. Teleoperation- In order to enable teleoperation of a Raven-II at the University of Washington in Seattle, CRTK servo_cr and robot state transitions were successfully implemented during the hands-on tutorial session at IROS 2018 in Madrid. At a rate of 1000 Hz, ROS topics are used by the Raven-II robot to check for new CRTK commands.As shown in Figure,6, Raven-II automatically maps the desired CRTK states to the internal Raven-II states upon receiving a state transition command and proceeds with the state change.Raven-II responds as follows when a servo_cr CRTK motion command is received:

   1) Determine whether the incremental Cartesian command servo_cr falls within a predetermined safety threshold for step sizes. If true, proceed, and if not, disregard the command. 2) Change the units and spatial transformation of the reference coordinates from the CRTK frame to the Raven-II base frame to transform servo_cr into raven_cr. 3) Attach the raven_cr command to the raven_cp_d command for the desired Raven-II Cartesian pose. 4) If the desired Raven-II pose differs from the current Raven-II pose by more than a safety threshold, cap raven_cp_d. 5) Give the motion command a shot.

Identify applicable funding agency here. If none, delete this text box.

Servo_cp and the Python client interface were used to demonstrate the CRTK-based teleoperation at the ISMR 2019 workshop in Atlanta, where a Phantom Omni was used to teleoperate a dVRK system at Johns Hopkins University in Baltimore, Maryland.A Novint Falcon was used as the slave arm to demonstrate the same code.



```
# in setup, created two clients: master and slave
scale = 0.25
# record where we started, only positions
start_master = PyKDL.Frame(master.measured_cp())
start_slave = PyKDL.Frame(slave.setpoint_cp())

# create target goal for slave, use current
  orientation
goal_slave = PyKDL.Frame()

# loop
  # get master measured position
  current_master = \
      PyKDL.Frame(master.measured_cp())
  # compute goal for slave
  goal_slave.p = start_slave.p + scale \
      * (current_master.p - start_master.p)
  goal_slave.M = current_master.M \
      * start_master.M.Inverse() * start_slave.M
  # tell slave to move, then sleep
  slave.servo_cp(goal_slave)
  rospy.sleep(1.0 / rate)
```

2. Image-Guided Surgery -The CRTK API aims to make it simple to translate its commands to various surgical robot systems with different software architectures.Image-guided robots belong to a different category than teleoperated and cooperatively controlled surgical robots.These robots are frequently used to percutaneously or stereotactically place instruments like needles for therapy and biopsy, instruments for ablation, and electrodes.Medical imaging, such as MRI, ultrasound, or CT, is typically used intraoperatively or registered to an intraoperative tracking system to direct the procedure in this scenario. CRTK command structures were implemented on the WPI NeuroRobot system in order to demonstrate the capability of the proposed framework for this kind of robot [26].This seven-degree-of-freedom MRI-compatible stereotactic surgical robot is used for interstitial needle-based therapeutic ultrasound for the ablation of brain tumors. Its use case is typical of the kind of robot in which one or more targets and optionally an associated trajectory to reach them are defined in medical imaging and the robot is intended to follow that trajectory to align and insert the instrument. Often, real-time imaging is used to update the trajectory on the fly.A modular MRI-compatible robot controller that is used to control a number of surgical robots and has applications in prostate cancer [27] and neurosurgery [28] is in charge of the NeuroRobot system. This control system is a self-contained centralized controller that is housed in the MRI scanner room. It is connected to a robot that is housed with the patient on the scanner bed in the scanner bore.Onboard, a National Instruments sbRIO 9561 module runs a real-time Linux operating system. This system communicates with external devices via a fiberoptic Ethernet network connection with surgical navigation software, such as 3D Slicer [29]. This implementation of CRTK is directly translatable to a wide range of devices that also use the Open Network Interface for Image-Guided Therapy (OpenIGTLink) communication interface, which provides a standardized mechanism for communication among computers and devices in operating rooms for a wide variety of image-guided therapy (IGT) applications [22]. This image-guided surgery robot controller makes use of the Open Network Interface for Image-Guided Therapy (OpenIGTLink). The NeuroRobot controller uses the C++ OpenIGTLink library to communicate with other systems in both directions.The OpenIGTLink interface's packet naming convention was changed to use CRTK notation.The NeuroRobot can now receive a desired Cartesian setpoint with the servo_cp command and a desired joint setpoint with the servo_jp command thanks to this update. The CRTK-defined commands measured_cp, measured_jp, measured_jv, desired_cp, and desired_jp enable the NeuroRobot to also transmit internal robot parameters.The ROS-OpenIGTLink bridge is compatible with this transmit and receive implementation, making ROS-based control simple.

3. AMBF simulator example Using a plugin-based interface for haptic interaction, the physical dVRK MTMs are incorporated into the dynamic simulation by the AMBF Simulator [13].The plugin, which goes by the name dVRK ARM, can be found at "https://github.com/WPI AIM/ambf/tree/master/ambf ros modules/dvrk arm." Class methods that are modeled after the CRTK specification are provided by this plugin, which makes use of the ROS messaging interface that is made available by the dVRK software (Figure 7).Servo_jp,servo_jf,servo_cp, servo_cf, measured_jp, measured_jv, measured_cp, measured_cf, move_jp, and move_cp are just a few of the functions that it supports. The use of the CRTK-proposed hierarchical controller structure specification made the plugin's design and implementation simpler because it satisfied the plugin's requirements for controlling and sensing a simulated environment, which included a variety of control modes and different types of feedback data. A controller for a simulated Raven-II was implemented in AMBF, demonstrating yet another example of CRTK.Yun-Hsuan Su wrote the code, which can be found at [30].Raven-II kinematics calculation and support for a variety of control modes, such as homing, sinusoidal motions, and virtual 3-dimensional cube tracing, are included in this code and are also available in the actual Raven-II system.Simply pressing one of the preset keyboard shortcuts will select one of these modes.

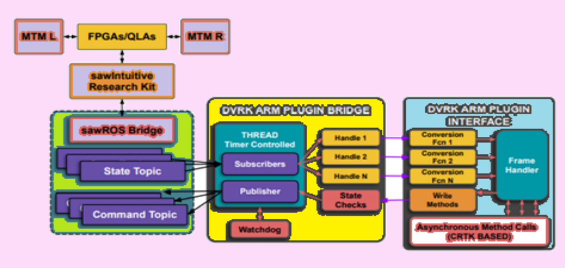


Fig. 7. The design of the dVRK ARM plugin based interface that is used by the AMBF Simulator. The block labeled "sawROS Bridge" provides the ROS Based interface. The sub-block labeled "Asynchronous Method Calls (CRTK BASED)" wraps the CRTK specification for state and command data for the method based interface.

IV. CONCLUSION

The Collaborative Robotics Toolkit (CRTK), a common robot command and feedback interface suitable for complex teleoperation and cooperative control tasks at various control levels, is the focus of this work.The AMBF simulator and the Raven-II and dVRK software have been updated to support CRTK.Download a set of sample client API and test codes at [31]. In the past few years, we have also held workshops and tutorials at a number of international robotics conferences to introduce CRTK to the community, collect user feedback, and encourage community adoption.In the meantime, you can find the user guide, additional documentation, and design details at [23].We hope to continue expanding the user base and making the CRTK infrastructure more userfriendly in future projects.

ACKNOWLEDGMENT

The authors we grateful to Yun-Hsuan Su, Adnan Munawar, Anton Deguet, Andrew Lewis, Kyle Lindgren, Yangming Li, Russell H. Taylor, Gregory S. Fischer, Blake Hannaford and Peter Kazanzides at Y-H. Su is with the Dept. of Computer Science, Mount Holyoke College, South Hadley, MA USA, msu@mtholyoke.edu. A. Lewis is with the Dept. of Mechanical Engineering, K. Lindgren, and B. Hannaford are with the Dept. of Electrical and Computer Engineer ing, University of Washington, Seattle, WA USA, {alewi, kyle509, blake}@uw.edu. A. Munawar and G.S. Fischer are with the Robotics Engineering Pro gram, Worcester Polytechnic Institute, Worcester, MA USA, {amunawar, gfischer}@wpi.edu. A. Deguet, R.H. Taylor and P. Kazanzides are with the Dept. of Computer Science, Johns Hopkins University, Baltimore, MD USA, {anton.deguet, rht, pkaz}@jhu.edu. Y. Li is with the Dept. of Electrical, Computer, and Telecommunication Engineering Technology at Rochester Institute of Technology, Rochester, NY USA, Yangming.Li@rit.edu. This work was supported by NSF National Robotics Initiative Awards IIS 1637789, IIS-1637759, and IIS-16374 for the research work in the paper “Collaborative Robotics Toolkit (CRTK): Open Software Framework for Surgical Robotics Research ” [32].

REFERENCES

[1] Murali, S. Sen, B. Kehoe, A. Garg, S. McFarland, S. Patil, et al., "Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms", IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1202-1209, 2015.

[2] Shademan, R. Decker, J. Opfermann, S. Leonard, A. Krieger and P. C. W. Kim, "Supervised autonomous robotic soft tissue surgery", Science Translational Medicine, vol. 8, May 2016.

[3] Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, et al., "Raven-II: an open platform for surgical robotics research", IEEE Transactions on Biomedical Engineering, vol. 60, no. 4, pp. 954-959, 2012.

[4] E. Messina, "Your mileage may vary", Science Robotics, vol. 4, no. 35, pp. eaay6004, 2019.

[5] F. Proctor, S. Balakirsky, Z. Kootbally, T. Kramer, C. Schlenoff and W. Shackleford, "The Canonical Robot Command Language (CRCL)", Industrial Robot: An International Journal, vol. 43, no. 5, pp. 495-502, 2016.

[6] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendiuelli and B. Siciliano, "A V-REP simulator for the da Vinci Research Kit robotic platform", Intl. Conf. on Biomedical Robotics and Biomechatronics, pp. 1056-1061, 2018.

[7] G. Guthart and J. Salisbury, "The Intuitive™ telesurgery system: Overview and application", IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 618-621, May 2000.

[8] J. MacDonell, N. Patel, G. Fischer, E. C. Burdette, J. Qian, V. Chumbalkar, et al., "Robotic Assisted MRI-Guided Interventional Interstitial MR -Guided Focused Ultrasound Ablation in a Swine Model", Neurosurgery, vol. 84, no. 5, pp. 1138-1148, 2018.

[9] J. Marescaux, J. Leroy, M. Gagner, F. Rubino, D. Mutter, M. Vix, et al., "Transatlantic robot-assisted telesurgery", Nature, vol. 413, no. 6854, pp. 379, 2001.

[10] J. Tokuda, G. S. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, et al., "OpenIGTLink: an open network protocol for image-guided therapy environment", Intl. J. of Medical Robotics and Computer Assisted Surgery, vol. 5, no. 4, pp. 423-434, 2009.

[11] K. Bumm, J. Wurm, J. Rachinger, T. Dannenmann, C. Bohr, R. Fahlbusch, et al., "An automated robotic approach with redundant navigation for minimal invasive extended transsphenoidal skull base surgery", Minimally Invasive Neurosurgery, vol. 48, pp. 159-64, July 2005.

[12] K. Shamaei, Y. Che, A. Murali, S. Sen, S. Patil, K. Goldberg, et al., "A paced shared-control teleoperated architecture for supervised automation of multilateral surgical tasks", IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1434-1439, 2015.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, et al., "ROS: an open-source Robot Operating System", ICRA Workshop on Open Source Software, 2009.

[14] M. Wartenberg, J. Schornak, K. Gandomi, P. Carvalho, C. Nycz, N. Patel, et al., "Closed-loop active compensation for needle deflection and target shift during cooperatively controlled robotic needle insertion", Annals of Biomedical Engineering, vol. 46, no. 10, pp. 1582-1594, Oct 2018.

[15] N. A. Patel, G. Li, W. Shang, M. Wartenberg, T. Heffter, E. C. Burdette, et al., "System integration and preliminary clinical evaluation of a robotic system for MRI-guided transperineal prostate biopsy", Journal of Medical Robotics Research, vol. 04, no. 02, pp. 1950001, 2019.

[16] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor and S. P. DiMaio, "An open-source research kit for the da Vinci® surgical system", IEEE Intl. Conf. on Robotics and Auto. (ICRA), pp. 6434-6439, Jun 2014.

[17] R. A. Gondokaryono, A. Agrawal, A. Munawar, C. J. Nycz and G. S. Fischer, "An approach to modeling closed-loop kinematic chain mechanisms applied to simulations of the da Vinci Surgical System", Acta Polytechnica Hungarica, vol. 16, no. 8, pp. 2019-2048, 2019.

[18] R. Bauernschmitt, E. U. Schirmbeck, A. Knoll, H. Mayer, I. Nagy, N. Wessel, et al., "Towards robotic heart surgery: Introduction of autonomous procedures into an experimental surgical telemanipulator system", The Intl. Journal of Medical Robotics and Computer Assisted Surgery, vol. 1, no. 3, pp. 74-79, 2005.

[19] R. Kikinis, S. D. Pieper and K. G. Vosburgh, 3D Slicer: A Platform for Subject-Specific Image Analysis Visualization and Clinical Support, New York, NY:Springer New York, pp. 277-289, 2014.

[20] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization", IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 4178-4185, 2016.

[21] T. D. Nagy and T. Haidegger, "An open-source framework for surgical subtask automation", ICRA Workshop on Supervised Autonomy in Surgical Robotics 2018.

[22] Y. Li, B. Hannaford and J. Rosen, "The Raven open surgical robotic platforms: A review and prospect", Acta Polytechnica Hungarica, vol. 16, no. 8, 2019.

[23] Y. Wang, R. Gondokaryono, A. Munawar and G. S. Fischer, "A convex optimization-based dynamic model identification package for the da Vinci Research Kit", IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3657-3664, Oct 2019.

[24] Z. Chen, A. Deguet, R. H. Taylor and P. Kazanzides, "Software architecture of the da Vinci Research Kit", IEEE Intl. Conf. on Robotic Computing, April 2017.

[25] 2019, [online] Available: https://github.com/collaborative-robotics/.

[26] 2019, [online] Available: https://github.com/collaborative-robotics/crtk-cpp.

[27] 2019, [online] Available: https://github.com/collaborative-robotics/documentation/wiki.

[28] 2019, [online] Available: https://github.com/WPI-AIM/ambf/tree/master/ambf_controller/.

[29] 2019, [online] Available: https:/github.com/collaborative-robotics/crtk-python-client/.

[30] Krupa, J. Gangloff, M. de Mathelin, C. Doignon, G. Morel, L. Soler, et al., "Autonomous retrieval and positioning of surgical instruments in robotized laparoscopic surgery using visual servoing and laser pointers", IEEE Intl. Conf. on Robotics and Automation (ICRA), vol. 4, pp. 3769-3774, 2002.

[31] A. Munawar, Y. Wang, R. Gondokaryono and G. Fischer, "A realtime dynamic simulator and an associated front-end representation format for simulating complex robots and environments", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[32] Image Referance Fig 1 **https://www.researchgate.net/figure/The-RAVEN-II-Surgical-Robot-is-the-target-system-for-the-cable-research-in-this-paper_fig1_303941918,**Fig2 http://biorobotics.ri.cmu.edu/robots/daVinci.php

[33] Yun-Hsuan Su, Adnan Munawar, Anton Deguet, Andrew Lewis, Kyle Lindgren, Yangming Li, Russell H. Taylor, Gregory S. Fischer, Blake Hannaford, Peter Kazanzides, "Collaborative Robotics Toolkit (CRTK): Open Software Framework for Surgical Robotics Research",IEEE, 2020