ORIGINAL ARTICLE

ROS-IGTL-Bridge: an open network interface for image-guided therapy using the ROS environment

Tobias Frank[1] · Axel Krieger[2] · Simon Leonard[3] · Niravkumar A. Patel[4] · Junichi Tokuda[5]

Received: 10 January 2017 / Accepted: 18 May 2017 / Published online: 31 May 2017

Abstract
Purpose With the growing interest in advanced image-guidance for surgical robot systems, rapid integration and testing of robotic devices and medical image computing software are becoming essential in the research and development. Maximizing the use of existing engineering resources built on widely accepted platforms in different fields, such as robot operating system (ROS) in robotics and 3D Slicer in medical image computing could simplify these tasks. We propose a new open network bridge interface integrated in ROS to ensure seamless cross-platform data sharing.
Methods A ROS node named ROS-IGTL-Bridge was implemented. It establishes a TCP/IP network connection between the ROS environment and external medical image computing software using the OpenIGTLink protocol. The node exports ROS messages to the external software over the network and vice versa simultaneously, allowing seamless and transparent data sharing between the ROS-based devices and the medical image computing platforms.
Results Performance tests demonstrated that the bridge could stream transforms, strings, points, and images at 30 fps in both directions successfully. The data transfer latency was <1.2 ms for transforms, strings and points, and 25.2 ms for color VGA images. A separate test also demonstrated that the bridge could achieve 900 fps for transforms. Additionally, the bridge was demonstrated in two representative systems: a mock image-guided surgical robot setup consisting of 3D slicer, and Lego Mindstorms with ROS as a prototyping and educational platform for IGT research; and the smart tissue autonomous robot surgical setup with 3D Slicer.
Conclusion The study demonstrated that the bridge enabled cross-platform data sharing between ROS and medical image computing software. This will allow rapid and seamless integration of advanced image-based planning/navigation offered by the medical image computing software such as 3D Slicer into ROS-based surgical robot systems.

Keywords ROS · OpenIGTLink · Interface · Surgical robot · Image-guided therapy

✉ Tobias Frank
tobifr@freenet.de

[1] Institute of Mechatronic Systems, Gottfried Wilhelm Leibniz Universität Hannover, Appelstrasse 11 a, 30167 Hannover, Germany

[2] Sheikh Zayed Institute for Pediatric Surgical Innovation, Childrens National Health System, 111 Michigan Avenue Northwest, Washington, DC 20010, USA

[3] Department of Computer Science, Johns Hopkins University, 3400 North Charles Street, Baltimore, MD 21218, USA

[4] Automation and Interventional Medicine (AIM) Laboratory, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609, USA

[5] Department of Radiology, Brigham and Womens Hospital and Harvard Medical School, 75 Francis Street, Boston, MA 02115, USA

Introduction

The use of robotic systems in image-guided therapy (IGT) has expanded in many medical fields leading to a continuous rise of technology in modern medicine [1]. Robot-assisted laparoscopic surgery has become common in radical prostatectomies in the USA and other developed countries [2] and expanding its use in other procedures. Robotic catheter systems have been used for a wide variety of endovascular procedures [3] and cardiac arrhythmia procedures [4]. Robotic radiosurgery systems have been used to treat tumors of the lung, liver, pancreas, spine, kidney, head, and neck and prostate [5].

While most of todays clinical robotic systems are designed to assist surgeons by following their commands and plans, there has been a growing interest in autonomous assistance, where robotic systems take over some of the surgeons routine tasks, such as cutting and suturing, to let the surgeons focus on high-level surgical decisions [6,7]. With this growing interest in autonomous technologies, researchers and surgeons face a vast variety of technology in the operating room. This growing interest motivates the medical robotics community to take advantage of a wide variety of features incorporated in the robot operating system (ROS) [8], such as computer vision, sensing, kinematics, simulation, and motion planning. Robotic research systems including medical robots Raven II [9] and the da Vinci Research Kit (dVRK) [10] and the KUKA light weight robot (LWR) [11] have used ROS as a software platform.

However, the ROS platform is not an ideal environment to perform certain clinical tasks. Specifically, modern surgical planning and guidance heavily rely on medical images to identify and localize diseased areas and critical structures. Furthermore, such navigational information must be mapped onto the physical space in order to achieve safe and accurate treatment. While ROSs versatility allows researchers to implement such features by themselves, many of them have already been available in some research platforms that are specifically developed for medical image computing and image-guided therapy, such as 3D Slicer [12], IGSTK [13], MITK [14] or NifTK [15], OsiriX [16], the XIP-Builder [17], and MeVisLab [18]. Therefore, bridging a robotics research platform such as the ROS with medical image computing platforms is becoming important for the development of advanced medical robotics systems. By bringing popular platforms extensively developed in the two research fields together, researchers can take advantage of rich engineering resources from both fields.

Bridging a popular platform like ROS with medical image computing platforms would further benefit the medical robotics research. Because of the wide variety of robot hardware supported by ROS, ranging from hobby-oriented products to industry-grade high dexterity robots, one can switch hardware easily, or scale-up the system from a proof-of-concept prototype to a fully functional system for animal and human studies, without significantly changing the software architecture. Therefore, the bridge will make the iteration of prototyping and testing easier and faster.

The goal of this study was to develop a new software interface that bridges ROS and popular medical image computing software, 3D Slicer, and provide a research and engineering tool that supports the development of image-guided and robot-assisted surgery system. The two software platforms seamlessly share data and commands through a TCP/IP network using the open network communication protocol, OpenIGTLink [19].

In this paper, we describe the system architecture and its implementation, as well as a proof-of-concept scenario. The structure of the implemented bridge in ROS and the conversions between ROS and OpenIGTLink messages are explained in the “Methods” section. Subsequently, the results of network communication performance tests are presented in the “Experiments” section followed by the “Use Cases” section that provides an outlook on the capabilities of the network bridge. The experimental results and use cases are discussed in Discussion section.

Methods

ROS-IGTL-Bridge node

The core component of our software interface is a ROS node named ROS-IGTL-Bridge. The ROS-IGTL-Bridge works as one of nodes in a ROS-based system and establishes a TCP/IP socket connection with 3D Slicer or other external medical image computing software equipped with a socket interface. It can be configured to run either as a TCP/IP server or client. ROS provides a graph architecture consisting of multiple nodes that are connected by the peer-to-peer network and process data together. Nodes can publish a ROS message to a given *topic*, and other nodes can receive the message by subscribing to the topic. The ROS-IGTL-Bridge node translates a ROS message to an OpenIGTLink message and sends it to external software through the TCP/IP connection. It also receives an OpenIGTLink message, translates it to a ROS message, and publishes it in the ROS network (Fig. 1).

The interface supports data types commonly used in the context of image-guided and robot-assisted therapy. Supported data types are listed in the next section.

The ROS-IGTL-Bridge node process consists of two independent POSIX threads allowing simultaneous data sending and receiving (Fig. 2). Methods for message serialization

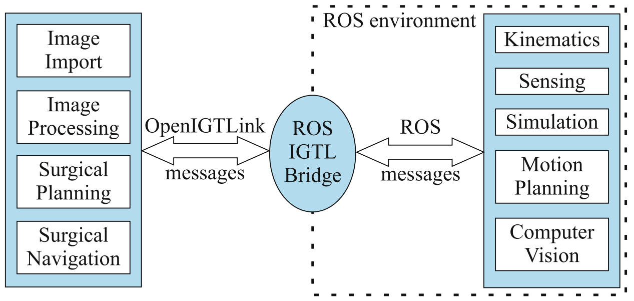


Fig. 1 ROS-IGTL-Bridge works as a message interface between medical image computing software that supports image import, image processing, surgical planning and surgical navigation (*left*), and the ROS environment that offers kinematic calculations, sensing, simulation, motion planning, and computer vision (*right*)

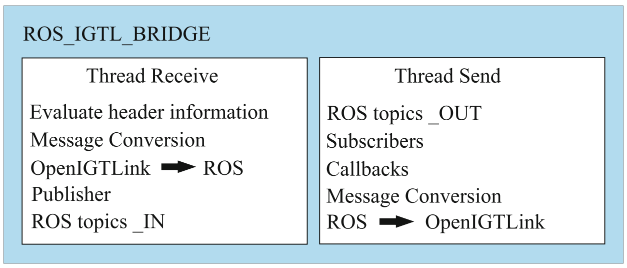


Fig. 2 Data processing by the bridge consisting of two independent POSIX threads. The thread "Receive" (*left*) handles incoming OpenIGTLink messages by evaluating the header information and execute message conversion to subsequently publish the data to the corresponding ROS topic. Simultaneously, the thread "Send" sends outgoing messages after previously triggered callbacks on subscribed ROS topics (*right*)

and deserialization for the OpenIGTLink communication were implemented per message types as callback functions and called by the two threads. Incoming messages received through the OpenIGTLink connection are evaluated by the header information of the message and published to their corresponding ROS topic in the ROS network.

The created topics in the ROS network are marked with _OUT for outgoing data and _IN for incoming data. The ROS-IGTL-Bridge node can easily be configured by using launch files and setting the parameter server IP, port and client/server flag. Furthermore, an autonomous testing node ROS-IGTL-Test can be used to evaluate the functionality by sending dummy data and visualizing received data on subscribed ROS topics. During the test routine, a random transform, a point, a pointcloud containing 20 points, a string, and a sample vtkPolyData model are sent.

Message types supported in ROS-IGTL-Bridge

The ROS-IGTL-Bridge supports various types of data that are frequently used in IGT applications. Those supported types include string, transform, image, poly data, and point. The following paragraphs detail the supported OpenIGTLink messages along with corresponding topics in ROS. To determine the conversion method from an OpenIGTLink message to a corresponding ROS message, the ROS-IGTL-Bridge uses the header section of an OpenIGTLink message, which provides meta-data including type name, device name, time stamp, body size, and pack status (Fig. 3).

String

The exchange of string messages between the device and external image computing software allows sending and receiving commands or status updates. Thus, the graphical user interfaces of the external software can be extended to display information like acknowledgments about received data, or to give instructions for controlling used devices (Fig. 4).

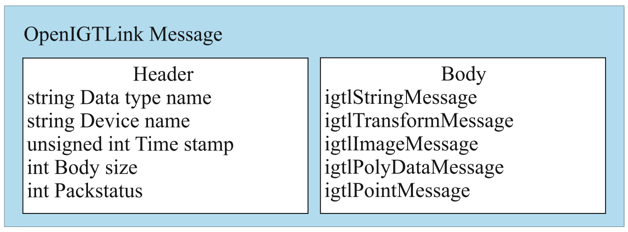


Fig. 3 Basic structure of an OpenIGTLink message consists of generic header information including data type, name, time stamp, size and pack status, and a data type specific body section

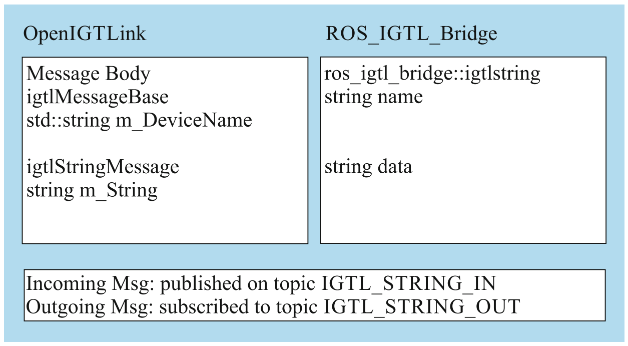


Fig. 4 Corresponding data fields in string messages for OpenIGTLink and ROS-IGTL-Bridge

Transform

The transform messages can contain a linear transform representing the positions and orientations of devices, objects of interests, etc, measured by sensors connected to ROS, or generated on the external image computing software. The messages can be used to monitor the positions and orientations of tools, devices, and objects, or providing planning data to the devices. The OpenIGTLink igtlTransformMessage is represented by a 4×4 matrix and converted to the ROS message type geometry_msgs/Transform containing a vector for translation and a quaternion for orientation (Fig. 5).

Image

Figure 6 shows the attributes of the corresponding image messages. An image consists of sizing and spacing parameters for each dimension in 3D space. The data are stored in an 8-bit array with the size of image dimensions. It is possible to send a 2D video stream from ROS using the IGTL_VIDEO_OUT topic which supports the common ROS message type sensor_msgs/Image. Thus, camera data can directly be forwarded to the bridge and no additional message conversion is required.

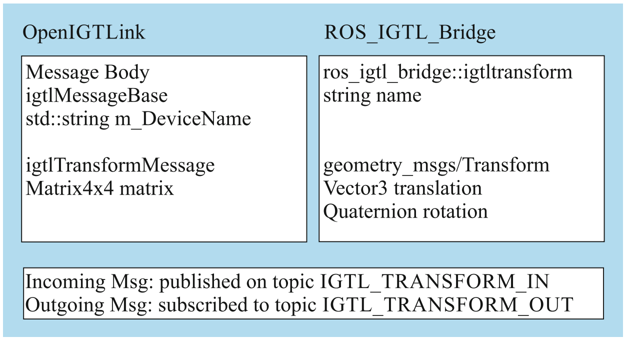


Fig. 5 Corresponding data fields in transform messages for OpenIGTLink and ROS-IGTL-Bridge. ROS standard data type *geometry_msgs/Transform* contains translation as a 3-element vector and rotation as a quaternion

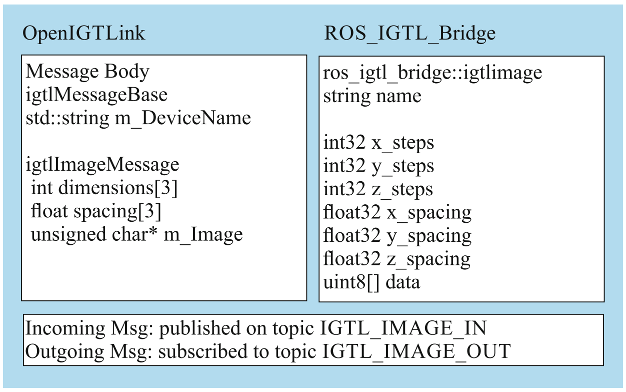


Fig. 6 Corresponding data fields in image messages for OpenIGTLink and ROS-IGTL-Bridge. The messages contain meta-information of the volume including the size and spacings as well as the pixel data

Poly data

The poly data message allows the transfer of 3D models composed of points and additional surface information. Polygons, triangle strips, lines, or vertices represent the structure of the mesh. Due to the lack of an explicit equivalent to the vtkPolyData message in ROS, methods for converting vtkPolyData to the ros_igtl_bridge::igtlpolydata message are provided.

Point

The point message consists of 3D point data allowing to send and obtain target points as a robot's movement destination, the manipulator's position or landmark coordinates for registration purposes or moreover sensor data in the form of point clouds and point lists (Figs. 7, 8). Additionally, the bridge is able to send point clouds in the form of geometry_msgs/Point published on the IGTL_POINTCLOUD_OUT topic.

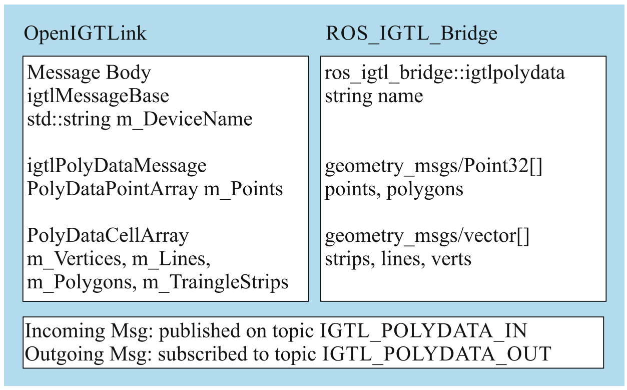


Fig. 7 Corresponding data fields in poly data messages for OpenIGTLink and ROS-IGTL-Bridge. The considered attributes include *name*, *points*, *polygons*, *strips*, *lines* and *vertices*

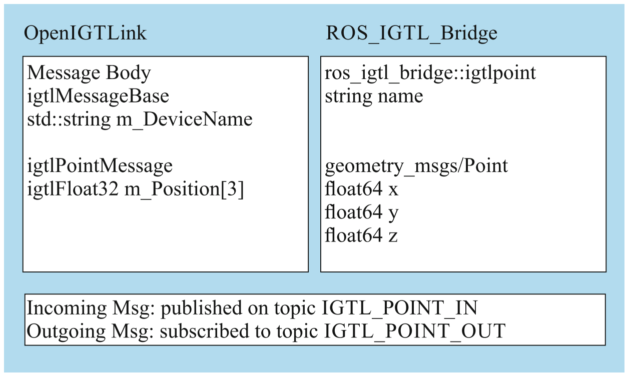
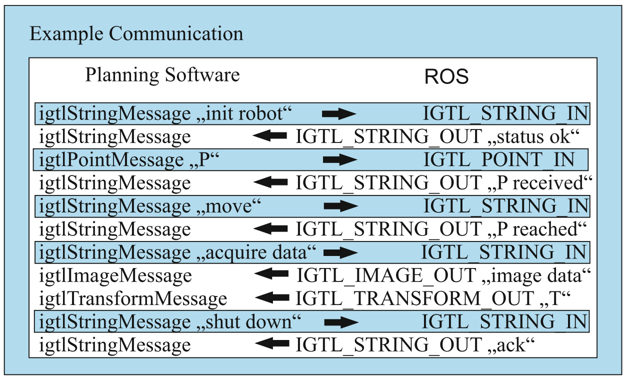


Fig. 8 Corresponding data fields in point messages for OpenIGTLink and ROS-IGTL-Bridge. *geometry_msgs/Point* contains the *x*-, *y*-, and *z*-coordinates of the point

System configuration and message scheme

The use of different message types depends on the system configuration and the required message scheme to operate the IGT setup. Figure 10 shows two representative system configurations as example applications for using the ROS-IGTL-Bridge.

The first example IGT setup consists of robot and tracking systems both of which are operated by ROS using already existing ROS nodes. The tracking system is used to track the end-effector of the robot. An external imaging device, for instance optical coherence tomography (OCT), ultrasound, or MRI, is connected to a medical image computing platform like 3D Slicer that provides methods for processing and visualization of such medical data (Fig. 10a). By bridging the ROS message environment and the medical image computing platform the function sets can efficiently complement each other. Commands and status updates are exchanged using the string message conversion of the bridge, so the user interface of the surgical planning tool can be used to operate the system. The transformation message allows for getting con-



Example Communication

Planning Software		ROS
igtlStringMessage „init robot“	➡	IGTL_STRING_IN
igtlStringMessage	⬅	IGTL_STRING_OUT „status ok“
igtlPointMessage „P“	➡	IGTL_POINT_IN
igtlStringMessage	⬅	IGTL_STRING_OUT „P received“
igtlStringMessage „move“	➡	IGTL_STRING_IN
igtlStringMessage	⬅	IGTL_STRING_OUT „P reached“
igtlStringMessage „acquire data“	➡	IGTL_STRING_IN
igtlImageMessage	⬅	IGTL_IMAGE_OUT „image data“
igtlTransformMessage	⬅	IGTL_TRANSFORM_OUT „T“
igtlStringMessage „shut down“	➡	IGTL_STRING_IN
igtlStringMessage	⬅	IGTL_STRING_OUT „ack“

Fig. 9 Possible communication protocol between a surgical planning software and a ROS operated robot using the ROS_IGTL_Bridge as network interface. The robot is initialized, commanded to move to a point and acquire image data with additional transform information. Finally, the robot is shutdown

tinuously refreshed tracking data or to command a certain position in six degrees of freedom. Robot model data could be transferred as poly data model to be visualized in 3D Slicer. Raw point cloud or image data of the tracking device can be forwarded to the image processing platform in order to localize robot position or perform path planning Fig. 9.

The second example represents a robot-guided imaging device possibly an ultrasound probe attached to the robots end-effector (Fig. 10b). The robot as well as the imaging device are controlled by downloadable standard ROS nodes allowing simple integration in the desired setup. To access or publish data in the resulting ROS message environment, the communication with an external surgical planning platform like 3D Slicer is established using the ROS-IGTL-Bridge node. Therefore, robot control as well as image device commands and responses can be transferred as string messages via the network bridge. Additionally, a positioning command as point or transformation message can be sent to the ROS environment. Acquired image data are forwarded to the planning software controlling the robot as image or point cloud messages, and thus, the image processing and visualization methods can be used to compare intraoperative with preoperative data. Figure 9 shows a possible communication protocol between the ROS environment and a surgical planning platform while performing image acquisition after being commanded to a specific point location.

Experiments

We evaluated the performance of network communication between ROS and 3D Slicer using a mock image-guided surgical robot system. In addition, we demonstrated the feasibility of the software in two representative use-case scenarios: educational/rapid-prototype image-guided surgical navigation system based on Lego Mindstorms, and autonomous suturing robot.

Experimental setup

The mock image-guided surgical robot system consists of two computers: a Linux-based computer (Precision M3800, Quad-Core Intel Core i7-4712HQ 2.3 GHz, 16 GB 1600 MHz DDR3 memory, Ubuntu Linux 14.04LTS, Dell Inc., Round Rock, TX, USA) that mimics a robot controller, and a Mac-based workstation (Mac Pro, Dual 6-Core Intel Xeon 2.66 GHz and 40 GB 1333 MHz DDR3 memory, Mac OS X 10.10, Apple Inc., Cupertino, CA, USA) that mimics a workstation for surgical planning and navigation interface. The two computers are connected to a 8-port gigabit Ethernet switch (SG100D-08, Cisco Systems Inc., San Jose, CA,

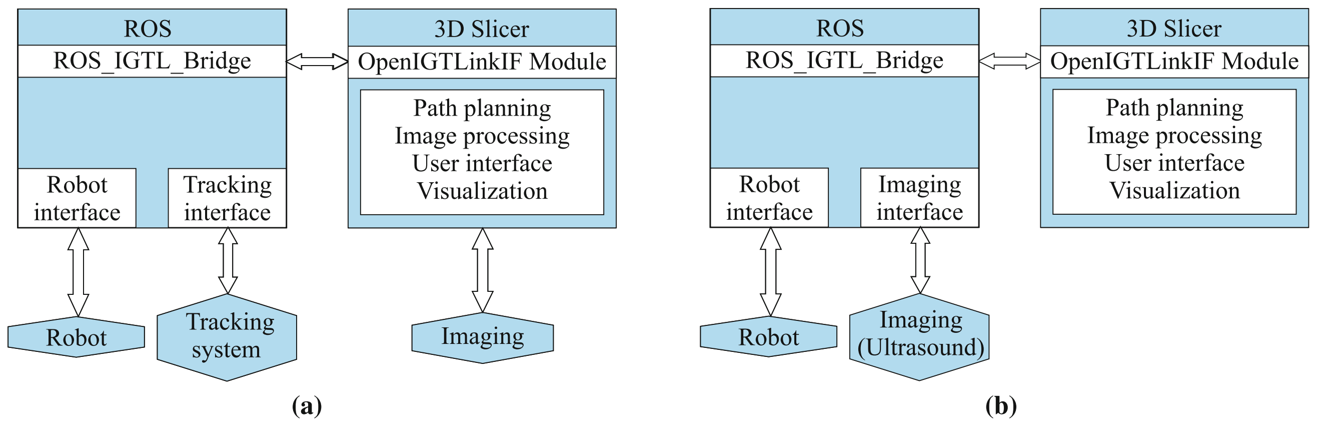


Fig. 10 Example setups showing the capabilities of ROS-IGTL-Bridge. **a** Robot and tracking system are connected to ROS and controlled by 3D Slicer through the ROS-IGTL-Bridge. The imaging device is connected directly to 3D Slicer using the OpenIGTLink interface. **b** The robot and imaging device (ultrasound) are connected to the robot operated by ROS and controlled by 3D Slicer through the ROS-IGTL-Bridge

USA) via Cat 5e cables. ROS Indigo was installed on the Linux-based computer with the ROS-IGTL-Bridge node. On the Mac workstation, 3D Slicer 4.6 was installed for visualizing data transferred from the ROS.

In order to evaluate the latency, the sender embedded the time of data generation in the message header based on the internal clock of the sender. Once the message is received by the receiver, the receiver extracts the time of data generation, and compare it with the internal clock of the receiver to calculate the time spent for data transfer between the two simulators. For accurate determination of the latency, the internal clocks of the two computers were synchronized using PTPd [20]. PTPd synchronizes the internal clock to the master clock through the network using the Precision Time Protocol (PTP) defined in the IEEE 1588-2008 standard. Unlike the widely used network time protocol (NTP) [21], PTP is designed for more accurate clock synchronization between computers connected to the local area network (LAN).

Evaluation of data transfer performance

Using this setup, we evaluated the performance of data transfer for transforms, strings, points, and images. These data types are often used for real-time data sharing, such as tool tracking, video streaming, and status monitoring, and thus, it is crucial to ensure the data delivery with appropriate frame rate and latency.

We deployed two custom software simulators, namely *ROS test node* and *IGTL test server*, on the Linux computer and Mac workstation, respectively. ROS test node is a ROS node that communicates with the ROS-IGTL-Bridge through the ROS network, whereas IGT test server is a TCP/IP server that communicates with the ROS-IGTL-Bridge over the LAN using the OpenIGTLink protocol. IGTL test server simulates the behavior of 3D Slicer or any other surgical planning and navigation software. Both simulators work as either a *sender* or a *receiver*; when the sender role is assigned to one simulator, the receiver role is assigned to the other. The sender generates a random message with given type and message size and sends it to the receiver via the ROS-IGTL-Bridge node. Two sets of tests were performed:

Qualitative evaluation of data transfer latency. We measured the data transfer latency while streaming the messages from the sender to the receiver at 30 frames per second (fps). Each transform message contains one linear transform that represents position and orientation. The size of each string message was fixed at 100 bytes, which is sufficient for commands for devices. For the point messages, multiple data sizes were used ranging from 1 point per message to 10,000 points per message, considering different use-case scenarios including tool tracking, landmark tracing and point cloud transmission. For image messages, we consider 2D color image (RGB) in five image formats: VGA (640 × 480 pixels), SVGA (800 × 600 pixels), XGA (1024 × 768 pixels), HD (1280 × 720 pixels), and Full HD (1920 × 1080 pixels).
Demonstration of High Frame Rate Data Transfer. Additionally, we tested high frame rate data transfer up to 1000 fps using the transform data type considering applications where sensory data (e.g., tracking sensors, encoders) are transferred between two computers over the local area network.

In both tests, the data were transferred from ROS test node to IGTL test server, and vice versa.

Demonstration of polygon data sharing

Additionally, transfer of poly data was demonstrated using the setup. For this qualitative evaluation, a simulator program called the ROS-IGTL-Test node was used as a ROS node, whereas 3D Slicer was used as external software that generated a 3D poly data model based on MRI data. The ROS-IGTL-Test node comes with the ROS-IGTL-Bridge software for testing purposes with a 3D model of the human head. After setting up the connection parameters in the file *test.launch*, the ROS-IGTL-Bridge and the integrated OpenIGTLinkIF module in 3D Slicer established a connection and subsequently, exchanged the poly data message.

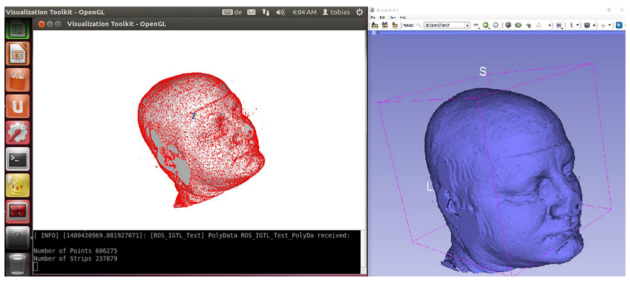
Results

The mean and standard deviation latency for 1000 frames are shown in Table 1. There were cases where some of the

Table 1 Means and standard deviations (SD) of message transfer latency for transform, point, string, and image messages based on measurements in 1000 message transfers

Message type	From ROS test node to IGTL test server (ms)	From IGTL test server to ROS test node (ms)
Transform	1.1 ± 0.5	1.1 ± 0.4
String (100 bytes)	1.1 ± 0.5	1.2 ± 0.4
Point (1 point)	1.2 ± 0.5	1.2 ± 0.4
Point (10 points)	1.2 ± 0.5	1.4 ± 0.5
Point (100 points)	17.2 ± 12.6	3.8 ± 5.4
Point (1000 points)	24.8 ± 12.3	20.9 ± 26.7
Image (VGA)	25.2 ± 4.3	23.6 ± 3.5
Image (SVGA)	33.6 ± 4.6[a]	33.6 ± 3.7[a]
Image (XGA)	58.8 ± 10.6[a]	58.5 ± 6.0[a]
Image (HD)	76.3 ± 10.5[a]	67.2 ± 6.1[a]
Image (Full HD)	NA[a]	144.5 ± 12.2[a]

[a] Some of the messages were not delivered to the recipient



```
[ INFO] [1480420969.081927071]: [ROS_IGTL_Test] PolyData ROS_IGTL_Test_PolyDa received:

Number of Points 606275
Number of Strips 237879
```

Fig. 11 Exchanged vtkPolyData model generated from sample data visualized in the vtkRenderWindow on the ROS side (*left*) and in comparison in the 3D Slicer scene (*right*)

messages were not delivered because the data transfer latency is larger than the interval of message transfers. During the measurement, the clock offset between the two computers was maintained at 0.00 ± 0.28 ms (mean ± SD).

For the demonstration of high frame rate data transfer, the transform messages were transferred from the ROS test node to the IGTL test server at 1000 fps successfully. When the messages were transferred in the other direction, the data transfer was successful up to 900 fps.

In the additional demonstration of poly data transfer, the poly data model consisting of more than 600,000 points and 230,000 faces was successfully transferred from ROS to 3D Slicer and vice versa. The transferred poly data were successfully visualized in both environments (Fig. 11).

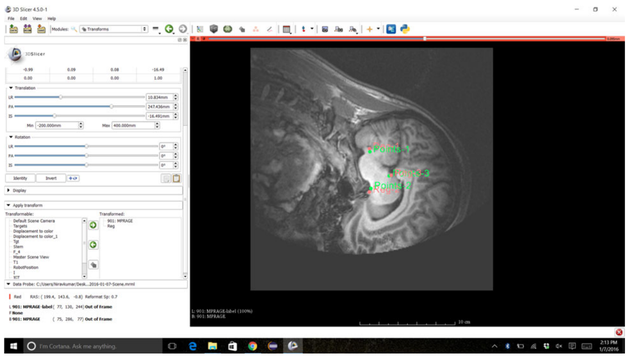
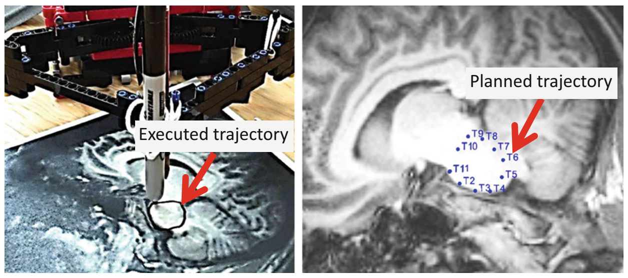


Fig. 12 Matching defined and physical landmarks on the 2D sample data using point registration algorithm in 3D Slicer



(a) **(b)**

Fig. 13 A robotic arm built with Lego Mindstorms is following a trajectory on the 2D phantom (**a**) that was previously planned in 3D Slicer (**b**) after a successful image-to-patient registration

Use cases

Rapid prototyping/educational platform for IGT

A proof-of-concept image-guided manipulator system was prototyped using 3D Slicer, ROS, and Lego Mindstorms EV3. Lego Mindstorms has been used for IGT-related projects in the context of education [22] as well as hardware/software testing platform [23]. The goal of this project was twofold: (1) create a brainstorming tool for medical robotics with a scalable software system that facilitates the seamless conversion from prototypes into research- and commercial-grade systems; and (2) create an educational tool for students, engineers, and scientists to learn image guidance and medical robotics.

The system mimics a surgical robot that actuates its end-effector to follow a trajectory in the physical space defined on a 3D medical image (e.g., CT, MRI) on surgical navigation software. The process can be monitored through 3D graphics. The system consists of an active 3-degree-of-freedom (DoF) parallel-link manipulator, control brick, ROS master computer, and navigation computers that ran the ROS master server and navigation software, 3D Slicer. The control computer module (Lego EV3 Programmable Brick) communicates with the navigation software via a ROS master server over the wireless network. We performed a mock procedure using an MR image of the brain, and a 2D phantom created from the image. The manipulator was registered to the phantom by recording the coordinates of predefined landmarks by physically touching them with its end-effector, and match them with the corresponding points on the image (Fig. 12).

Afterward, the drawing process on the 2D phantom was executed (Fig. 13a) using previously defined trajectory points transferred to the control brick from 3D Slicer (Fig. 13b). This mock procedure demonstrated that this Lego-based system can be used to build a robotic system with research- or commercial-grade architecture and mimic a realistic clinical workflow, making it an ideal tool for rapid prototyping and education.

Autonomous suturing with KUKA LWR

The goal was to test the feasibility of the ROS-IGTL-Bridge for planning and guiding autonomous surgical robots. Using the developed ROS-IGTL-Bridge, we started to improve the software system for the smart tissue autonomous robot (STAR), which uses the KUKA LWR robot. STAR consists of a robotic suturing tool based on a commercially available laparoscopic Endo360° (EndoEvolution, Raynham) tool,

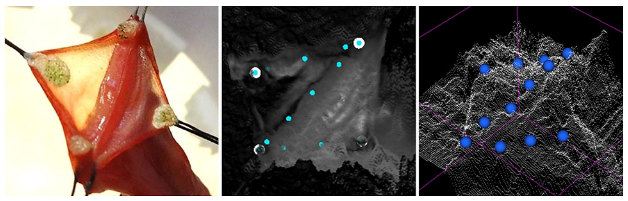


Fig. 14 Picture of porcine bowel staged for anastomosis (*left*), current STAR suture plan (*middle*), and suture plan using 3D Slicer enabling 3D adjustments (*right*)

custom control implemented using ROS for graphical user interface (GUI) and camera integration, and open robot control software (OROCOS) for real-time control [11,24]. While we demonstrated superior consistency and burst pressure using STAR, 42.2% of all sutures required manual operator adjustments, leading to longer procedure times compared to manual and teleoperated robotic surgery [7]. Most missed suture placements requiring operator adjustments were caused by noisy point cloud data from the plenoptic camera, in particular along the depth axis, that was not apparent on the two-dimensional suture plans provided by ROS. By upgrading STAR with the ROS-IGTL-Bridge, we were able to transfer 3D point clouds from STAR to Slicer for point cloud visualization (Fig. 14). This could lead to improve the workflow for 3D planning of suture locations, greater autonomy and shorter procedure times.

Discussion

In this work, a ROS-IGTL-Bridge node was implemented to extend ROS by a generic open network interface based on the OpenIGTLink protocol. This interface enabled seamless sharing of data frequently used in IGT applications, including strings, transforms, points, poly data, and images, between the ROS-based system and external medical image computing platforms. Conversion methods for the matching message types were generated to ensure compatibility. While the OpenIGTLink protocol provides two distinct message types for transmitting linear transforms, namely TRANSFORM and TDATA, we chose to use TRANSFORM in the current implementation because of its simplicity and generalness. A TRANSFORM message represents an affine transformation matrix, which can be easily converted to a ROSs *geometry_msgs/Transform* message, and can be used for many purposes, from tool tracking to coordinate transformations. TDATA could be supported in the future, as it provides convenient features for tool tracking, such as transmission of multi-channel tracking. The created messages and conversions can be easily adapted or new message types can be additionally included to fulfill task-specific needs.

The performance of the data transfer using the ROS-IGTL-Bridge was sufficient for many real-time IGT applications. Our test demonstrated that the data transfer latency of incoming and outgoing messages at 30 fps was <1.2 ms for string, points and transform messages and 25.2 ms for images (VGA). Additionally, a high frame rate data transfer up to 900 fps was achieved for the transform message. Furthermore, large poly data models with more than 600 thousand points and 230,000 faces were successfully exchanged. Transfer of images larger than the XGA format could not achieve full 30 fps. We are currently working on the extension of the OpenIGTLink protocol to provide video compression for better streaming performance.

The mock IGT system using 3D Slicer and Lego Mindstorm demonstrated that the ROS-IGTL-Bridge allowed building a prototype system and validate its functionality in a mock IGT procedure in a limited time. We were able to build this demo setup during the 22nd NA-MIC Winter Project Week 2016, a week-long hackathon event focused on open-source medical image computing software infrastructures [25]. The ROS-IGTL-Bridge was also demonstrated in an upscale IGT setup consisting of the industry-grade high dexterity robots with autonomous controlling system. The ROS-IGTL-Bridge's ability to transfer point cloud data enabled incorporating advanced visualization and 3D planning of suture location offered by 3D Slicer into the system.

The study demonstrated that the ROS-IGTL-Bridge enabled cross-platform data sharing between ROS and image-guidance software with sufficient data transfer performance. The bridge benefits IGT setups by combining the specific methods such as robot control, motion planning and sensing within ROS with the image processing and visualization function set of surgical planning tools like 3D Slicer. This will allow rapid and seamless integration of advanced image-based planning/navigation offered by image-guidance software such as 3D Slicer into ROS-based surgical robot systems. By bridging two different software platforms, the researcher can benefit from state-of-the-art engineering resources developed in the different research fields, including robotics and medical image computing to develop advanced image-guided surgical robot systems.

The idea of using OpenIGTLink to bridge two research platforms has been demonstrated in several studies. In particular, OpenIGTLink has been extensively used to bridge data grabbing software and visualization/user interaction software for medical image computing research. Papademetris, Tokuda et al. used OpenIGTLink to bridge a commercial navigation system with external research platforms including 3D Slicer [19] and BioImage Suite [26]. The Image-Guided Surgery Toolkit (IGSTK) [13], an open software platform that provides connectivity with tracking and imaging devices, supported OpenIGTLink to export tracking and imaging data to other platforms such as 3D Slicer and MITK [27,28]. More recently, Lasso et al. used OpenIGTLink to bridge their data grabbing software, PLUS, with external visualization software to stream tracking and ultrasound image data [29]. Clarkson et al. developed a messaging library, NiftyLink, based on OpenIGTLink to integrate data grabbing software and end-user application for visualization/user interaction [15]. Other open-source packages, such as CustusX [30], IBIS [31] and MeVisLab [18], have adapted to OpenIGTLink as well, to take advantage of existing software infrastructures available in the community.

OpenIGTLink has also been used to bridge platforms beyond the medical image computing field. A medical robotics platform, the CISST library, offers an OpenIGTLink bridge to integrate image visualization software into medical robotics applications [32]. OpenIGTLink is also used with ROS-based robotic system for laparoscopic interventions [33]. The ROS-IGTL-Bridge is a generalization of those prior works aiming to extend the successful model found in the medical image computing field and has the potential to facilitate the sharing of engineering resource between the medical robotics and medical image computing.

The source code, instruction, and test program of the ROS-IGTL-Bridge is available as open-source software at GitHub [34]. The code can be compiled and installed with Catkin, a CMake-based build system.

Acknowledgements This study was supported in part by the National Institutes of Health (R01EB020667, R01CA111288, R01EB020610, P41EB015898). The authors thank Mr. Longquan Chen of Brigham and Women's Hospital for his technical support. The authors also thank Ms. Christina Choi for evaluating the rapid prototyping platform with Lego Mindstorms.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Animal or human rights No animal or human study was performed in this study.

References

1. Beasley RA (2012) Medical robots: current systems and research directions, pp 1–14
2. Trinh QD, Sammon J, Sun M, Ravi P, Ghani KR, Bianchi M, Jeong W, Shariat SF, Hansen J, Schmitges J, Jeldres C, Rogers CG, Peabody JO, Montorsi F, Menon M, Karakiewicz PI (2012) Perioperative outcomes of robot-assisted radical prostatectomy compared with open radical prostatectomy: results from the nationwide inpatient sample. Eur Urol 61(4):679–685
3. Shurrab M, Schilling R, Gang E, Khan EM, Crystal E (2014) Robotics in invasive cardiac electrophysiology. Expert Rev Med Devices 11(4):375–381
4. de Ruiter QMB, Moll FL, van Herwaarden JA (2015) Current state in tracking and robotic navigation systems for application in endovascular aortic aneurysm repair. J Vasc Surg 61(1):256–264
5. Dieterich S, Gibbs IC (2011) The CyberKnife in clinical use: current roles, future expectations. Front Radiat Ther Oncol 43:181–194
6. Moustris GP, Hiridis SC, Deliparaschos KM (2011) Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature. Int J Med Robot Comput Assist Surg 7(4):375–392
7. Shademan A, Decker RS, Opfermann JD, Leonard S, Krieger A, Kim PCW (2016) Supervised autonomous robotic soft tissue surgery. Sci Transl Med 8(337):337ra64
8. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. ICRA workshop on open source software 3(3.2):5
9. Hannaford B, Rosen J, Friedman DW, King H, Roan P, Cheng L, Glozman D, Ma J, Kosari SN, White L (2013) Raven-II: an open platform for surgical robotics research. IEEE Trans Biomed Eng 60(4):954–959
10. Kazanzides P, Chen Z, Deguet A, Fischer GS, Taylor RH, DiMaio SP (2014) An open-source research kit for the da Vinci® surgical system. In 2014 IEEE international conference on robotics and automation (ICRA), 6434–6439. IEEE
11. Leonard S, Wu KL, Kim Y, Krieger A, Kim Peter CW (2014) Smart tissue anastomosis robot (STAR): a vision-guided robotics system for laparoscopic suturing. IEEE Trans Biomed Eng 61(4):1305–1317
12. Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin JCC, Pujol S, Bauer C, Jennings D, Fiona F, Sonka M, Buatti J, Aylward S, Miller JV, Pieper S, Kikinis R (2012) 3D slicer as an image computing platform for the quantitative imaging network. Magn Reson Imaging 30(9):1323–1341
13. Enquobahrie A, Cheng P, Gary K, Ibanez L, Gobbi D, Lindseth F, Yaniv Z, Aylward S, Jomier J, Cleary K (2007) The image-guided surgery toolkit IGSTK: an open source C++ software toolkit. J Digit Imaging 20(Suppl 1):21–33
14. Nolden M, Zelzer S, Seitel Al, Wald D, Müller M, Franz AM, Maleike D, Fangerau M, Baumhauer M, Maier-Hein L, Maier-Hein KH, Meinzer HP, Wolf I (2013) The medical imaging interaction toolkit: challenges and advances. Int J Comput Assist Radiol Surg 8(4):607–620
15. Clarkson MJ, Zombori G, Thompson S, Totz J, Song Yi, Espak M, Johnsen S, Hawkes D, Ourselin S (2015) The NifTK software platform for image-guided interventions: platform overview and NiftyLink messaging. Int J Comput Assist Radiol Surg 10(3):301–316
16. Rosset A, Spadola L, Ratib O (2004) OsiriX: an open-source software for navigating in multidimensional DICOM images. J Digit Imaging 17(3):205–216
17. Paladini G, Azar FS (2009) An extensible imaging platform for optical imaging applications. In: SPIE BiOS: biomedical optics, International Society for Optics and Photonics, p 717108
18. Egger J, Tokuda J, Chauvin L, Freisleben B, Nimsky C, Kapur T, Wells W (2012) Integration of the OpenIGTLink network protocol for image-guided therapy with the medical platform MeVisLab. Int J Med Robot 8(3):282–290
19. Tokuda J, Fischer G, Papademetris X, Yaniv Z, Ibanez L, Cheng P, Liu H, Blevins J, Arata J, Golby AJ, Kapur T, Pieper S, Burdette EC, Fichtinger G, Tempany CM, Hata N (2009) OpenIGTLink: an open network protocol for image-guided therapy environment. Int J Med Robot Comput Assist Surg 5(4):423–434
20. Correll K, Barendt K, Branicky M (2005) Design considerations for software only implementations of the IEEE 1588 precision time protocol. In Conference on IEEE 1588, pp 11–15. https://repo.eecs.berkeley.edu/
21. Mills DL (1991) Internet time synchronization: the network time protocol. IEEE Trans commun 39(10):1482–1493

22. Pace D, Kikinis R, Hata N (2007) An accessible, hands-on tutorial system for image-guided therapy and medical robotics using a robot and open-source software. Int Conf Med Image Comput Comput Assist Interv 10(WS):122–141
23. Jomier J, Ibanez L, Enquobahrie A, Pace D, Cleary K (2009) An open-source framework for testing tracking devices using Lego Mindstorms. In: SPIE Medical Imaging, International Society for Optics and Photonics, p 72612S
24. Leonard S, Shademan A, Kim Y, Krieger A, Kim PCW (2014) Smart tissue anastomosis robot (star): accuracy evaluation for supervisory suturing using near-infrared fluorescent markers. In: IEEE international conference on robotics and automation, (ICRA 2014)
25. 22nd NA-MIC Winter Project Week. http://www.na-mic.org/Wiki/index.php/2016_Winter_Project_Week (2016)
26. Papademetris X, Jackowski MP, Rajeevan N, DiStasio M, Okuda H, Constable RT, Staib LH (2006) BioImage suite: an integrated medical image analysis suite: an update. Insight J 2006:209
27. Lu T, Liang P, Wu WB, Xue J, Lei CL, Li YY, Sun YN, Liu FY (2012) Integration of the image-guided surgery toolkit (IGSTK) into the medical imaging interaction toolkit (MITK). J Digit Imaging 25(6):729–737
28. Klemm M, Kirchner T, Grhl J, Cheray D, Nolden M, Seitel A, Hoppe H, Maier-Hein L, Franz AM (2017) MITK-OpenIGTLink for combining open-source toolkits in real-time computer-assisted interventions. Int J Comput Assist Radiol Surg 12(3):351–361
29. Lasso A, Heffter T, Rankin A, Pinter C, Ungi T, Fichtinger G (2014) PLUS: open-source toolkit for ultrasound-guided intervention systems. IEEE Trans Biomed Eng 61(10):2527–2537
30. Askeland C, Solberg OV, Bakeng JBL, Reinertsen I, Tangen GA, Hofstad EF, Iversen DH, Vpenstad C, Selbekk T, Lang T, Hernes TAN, Leira HO, Unsgrd G, Lindseth F (2016) CustusX: an open-source research platform for image-guided therapy. Int J Comput Assist Radiol Surg 11(4):505–519
31. Drouin S, Kochanowska A, Kersten-Oertel M, Gerard IJ, Zelmann R, Nigris DD, Briault S, Arbel T, Sirhan D, Sadikot AF, Hall JA, Sinclair DS, Petrecca K, DelMaestro RF, Collins DL (2017) IBIS: an OR ready open-source platform for image-guided neurosurgery. Int J Comput Assist Radiol Surg 12(3):363–378
32. Deguet A, Kumar R, Taylor R, Kazanzides P The *cisst* libraries for computer assisted intervention systems. In: MICCAI workshop on systems and arch. for computer assisted interventions, (2008)
33. Bihlmaier A, Beyl T, Nicolai P, Kunze M, Mintenbeck J, Schreiter L, Brennecke T, Hutzl J, Raczkowsky J, Wörn H (2016) ROS-based cognitive surgical robotics. In Koubaa A (ed) Robot operating system (ROS), vol. 625 in studies in computational intelligence. Springer, Heidelberg, pp 317–342. doi:10.1007/978-3-319-26054-9_12
34. Frank T (2016) ROS-IGTL-Bridge source code GitHub repository https://github.com/openigtlink/ros-igtl-bridge