Published in final edited form as:

Int J Comput Assist Radiol Surg. 2015 March ; 10(3): 285–292. doi:10.1007/s11548-014-1081-1.

OpenIGTLink interface for state control and visualisation of a robot for image-guided therapy systems

Sebastian Tauscher¹, Junichi Tokuda², Günter Schreiber³, Thomas Neff³, Nobuhiko Hata², and Tobias Ortmaier¹

¹Institute of Mechatronic Systems, Leibniz Universität Hannover, Appelstrasse 11 a, 30167 Hannover, Germany

²Department of Radiology, Brigham and Women's Hospital, Francis Street 75, Boston, MA 02115, USA

³KUKA Laboratories GmbH, Zugspitzstrasse 140, 86165 Augsburg, Germany

Abstract

Purpose—The integration of a robot into an image-guided therapy system is still a time consuming process, due to the lack of a well-accepted standard for interdevice communication. The aim of this project is to simplify this procedure by developing an open interface based on three interface classes: state control, visualisation, and sensor. A state machine on the robot control is added to the concept because the robot has its own workflow during surgical procedures, which differs from the workflow of the surgeon.

Methods—A KUKA Light Weight Robot is integrated into the medical technology environment of the Institute of Mechatronic Systems as a proof of concept. Therefore, 3D Slicer was used as visualisation and state control software. For the network communication the OpenIGTLink protocol was implemented. In order to achieve high rate control of the robot the “KUKA Sunrise. Connectivity SmartServo” package was used. An exemplary state machine providing states typically used by image-guided therapy interventions, was implemented. Two interface classes, which allow for a direct use of OpenIGTLink for robot control on the one hand and visualisation on the other hand were developed. Additionally, a 3D Slicer module was written to operate the state control.

Results—Utilising the described software concept the state machine could be operated by the 3D Slicer module with 20 Hz cycle rate and no data loss was detected during a test phase of approximately 270 s (13,640 packages). Furthermore, the current robot pose could be sent with more than 60 Hz. No influence on the performance of the state machine by the communication thread could be measured.

Conflict of interest. The two co-authors Thomas Neff and Günter Schreiber are with KUKA Laboratories GmbH (Augsburg, Germany). Nobuhiko Hata is a member of the Board of Directors of AZE Technology, Cambridge, MA and has an equity interest in the company. AZE Technology develops and sells imaging technology and software. Nobuhiko Hata's interests were reviewed and are managed by the Brigham and Women's Hospital and Partners HealthCare in accordance with their conflict of interest policies.

Conclusion—Simplified integration was achieved by using only one programming context for the implementation of the state machine, the interfaces, and the robot control. Eventually, the exemplary state machine can be easily expanded by adding new states.

Keywords

Robot control interface; 3D Slicer; Finite state machine; Visualization

Introduction

A seamless integration of surgical assist systems, such as robots, into the surgical workflow is mentioned by Lemke et al. [1] as a growing factor for the acceptance of these systems in the near future. Up to date, this integration process into an image guided therapy (IGT) system is still a barrier for experimental and commercial use, which needs to be overcome. Therefore, an established and accepted standard for the communication between devices under real time conditions would be an important step towards this goal.

A typical IGT system consists of a tracking system, a planning and control machine, and a mechatronic device, e.g. a robot. Hence, three general interface classes can be identified for the integration of a robot into an IGT system: control (user), visualization, and sensor interface [2]. The fact that a robot follows a separate workflow during the surgical procedure different to the surgeon's workflow, leads to the integration of a state machine on the robot control into the concept of the (state) control interface (see Fig. 1). From the robot's point of view the current control mode describes the robot state during its workflow, e.g. if the robot is moved manually and freely in impedance control mode by the surgeon or if it is moving autonomously position controlled to a specific pose. The current surgical procedure is of negligible interest for the robot, e.g. if it is positioning a needle or a drilling tool. By the integration of the state machine into the interface concept the surgeon is able to control the robotic workflow separately and directly during the intervention. Therefore, an intuitive state control aligned to the surgical workflow is important, because the surgeon shouldn't have to deal with control modes and similar on the other hand.

Especially for image guidance or visual servoing applications, such as optically navigated drilling in the skull base or pedicle screw placement in the spine, the availability of the tool position and orientation in real time with low latency is a crucial aspect for the reliability and performance of the system. Hence, the tracking system is seen as a sensor which is used to improve the performance of the robotic system.

The current work focuses on the control and visualization interface. Accordingly, the sensor interface is not described in detail. The need for a visualization interface is motivated by the different demands for state control and navigation purposes. Reliability and safety are more important for the state control, whereas real time ability and update rates of more than 30 Hz are essential for visualization. Furthermore, a bilateral interface for alive observation is advantageous for state control. Usually, a message converter or further interface technology, e.g. Fast Research Interface (FRI) [3] or Modular Robot Control Library [4], is needed for the communication with the robot control of a proprietary robotic system (see Fig. 2). This leads to an additional communication layer and, hence, to a higher latency and an additional

source of errors. To overcome this problem, a direct integration of an open interface into the robot control is introduced (see Fig. 2).

State of the art

Developments like the extension of the DICOM standard for surgery by the working group 24 [5] or the Integrating the Healthcare Enterprise (IHE) (<http://www.ihe.net>) emphasize the growing importance of the integration of mechatronic systems into the operating room (OR) in general and especially in surgery. Due to these tendencies, various research groups dealt with this issue.

Bzostek et al. [4] developed a network system distribution consisting of an OR network, which connects several workstations via Ethernet Local Area Network (LAN). The devices tracker, robot, and computation machines each possess a server and every application, for example a visualisation, has a client for each device it needs to communicate with. Hence, the communication concept follows a bus network topology. To allow for a use of different robotic systems, a modular software is introduced, which serves as a middleware between the open Application Programming Interface (API) and the proprietary API.

Xia et al. [6] present a concept for mechanical assistance in skull base drilling comprising the NeuroMate (Renishaw plc, UK) robot, a 3D Slicer Workstation, a robot control and application workstation, and a StealthStation (Medtronic, Minnesota, USA). The application workstation acts as the central point for the communication and provides interfaces to other elements of the overall system.

An open protocol for IGT is introduced in [7] and four integration examples are described. In these examples, the protocol is used for data transfer between 3D Slicer, tracking systems (e.g. VectorVision), robots, and imaging devices, which illustrates the quantity of possible fields of application for this protocol. In addition to this, the variety of combinations with different other interfaces and open source programs, e.g. the framework for image processing MeVisLab [9], indicates the acceptance in the field of robot assisted surgery.

Unmet needs

The issue of the existing interfaces for the integration of robots into IGT systems is that other researchers and developers can hardly reuse those existing interfaces due to proprietary license, poor portability, and/or lack of well-defined specification. Although there are several industrial standards for network communication between medical devices as discussed above, none of them are suitable for the integration of robots into IGT systems.

Approach

The aim of this project is to facilitate the integration of robots into IGT systems by providing a simple and well-defined open interface based on an open interface concept [2]. The challenge here is that the protocol must be capable to perform real time communications for controlling hardware state, and sharing control parameters (Table 1). To address those challenges with simple and well-defined open interface, we developed a new interface specialized for the integration of a robot into an IGT system by using the OpenIGTLink

network communication protocol [9]. The OpenIGTLink protocol is well-defined, simple, open, and highly portable, and thus an ideal basis for our communication interface. In this study, we define a new high-level application-specific communication scheme on top of the OpenIGTLink-based communication; this new scheme allows for exchange of information that is required for safe operation of the robot, including state machine management. Our interface is tested and validated by integrating the KUKA Light Weight Robot (LWR) (KUKA Laboratories GmbH, Augsburg) into the medical technology laboratory of the Institute of Mechatronic Systems. The open-source package 3D Slicer (<http://www.slicer.org>) was used as state control and visualization software.

Materials and methods

In this study, a setup consisting of a LWR as well as a control and visualization workstation was used to validate the concept and to show its advantages. For visualization and state control the free open source software package 3D Slicer was used. To achieve a high rate control of the robot the “KUKA Sunrise. Connectivity SmartServo” package was utilized, which meets the demands of typical IGT interventions (see Table 1), e.g. visual servoing. These software packages as well as the used robot are described in more detail in the following section.

3D Slicer

3D Slicer [8] is a free open-source software package built on top of Visualization Toolkit (VTK) and Insight Segmentation and Registration Toolkit (ITK) offering a wide range of image visualization and processing algorithms for medical image computing research. It can visualize volumetric 3D images in 2D and 3D representations using volume reslicing and surface rendering. Additionally, a variety of filters, segmentation, and registration algorithms can be applied to images. 3D Slicer also offers two mechanisms for developers to customize or extend its functionality. “Slicer Extension”, a plugin mechanism to run a custom program code within 3D Slicer, allows adding new features in the software. “OpenIGTLink Interface” (OpenIGTLinkIF), a network interface based on the OpenIGTLink protocol [7], enables data import from external software or hardware.

OpenIGTLink

The OpenIGTLink protocol [7] is a network communication protocol specifically designed for hardware and software components used in image-guided surgery, such as planning/navigation software [9,10], imaging scanners [11], and robotic devices [12]. The protocol is used in the application layer of the Transmission Control Protocol/Internet Protocol (TCP/IP) model. It defines simple messaging formats for data types commonly used in IGT including linear transforms, images, texts, and device statuses. Because of its simplicity and small footprint, the protocol is suitable for data streaming for position tracking and real-time imaging. Currently, Java- based (igtlink4j¹) and free open-source C++ [7] libraries are available to the research community. The OpenIGTLink protocol is used in several popular

¹http://aimlab.wpi.edu/research/projects/OpenIGTLink_Interface.

open-source software packages including 3D Slicer, IGSTK [13], and PLUS [14], as well as in commercial surgical navigation packages.

SmartServo

Recent robot control systems lack the ability of extensibility by the application engineer. Thus, in the past a variety of interfaces has been created. The reasons are various and range from the demands of production guarantees (24/7 availability) to software architectures, which make it difficult to modify the robot's behavior outside of the manufacturers laboratories. Especially, for the use of robots in the field of IGT it is essential to modify their behavior to meet the demands for IGT.

In the new “KUKA Sunrise.Connectivity” package, those shortcomings were overcome, since the well-known Java Technology Framework (<http://www.java.com/>) as basis is engaged. Thereby, the application engineer can easily extend the features of the robot's application by utilizing modern IT technologies (e.g. Java Libraries or JNI (Java Native Interface) for exploitation of legacy C++-code). This fits perfectly to directly support OpenIGTLink within the robot control application without any further interface technology, e.g. XIRP [15], FRI [3], or similar.

To meet the demands of high rate robot control, the visual servoing application, an additional motion package “KUKA Sunrise. Connectivity SmartServo” (SmartServo) was created. This motion type enables the (Java) robot application to modify the robot's path in a high rate (up to 2–3ms), but does not necessarily meet hard real time restrictions itself. The timing of the command sequence can be chosen in wide range, e.g. in the setting within this paper to 10–20 ms. To implement visual servoing applications and similar, the robot's interpolator within the real time system is capable to cope with varying update rates, and computes a trajectory, within real time requirements. The resulting trajectory is of the differentiability class C^2 , thus, the first two derivatives of the robot's position, the acceleration and the velocity, are continuous and the jerk is bounded.

Software implementation

Within the robot control, a state machine was implemented in Java using the SmartServo package to set the new destination of the robot and the control parameters, e.g. stiffness and damping values for impedance control. For the communication with the 3D Slicer workstation, a visualization interface class `LWRVisualisationIF` and a control interface class `LWRStateControlIF` based on the Java OpenIGTLink implementation (`igtlink4j`) were written. A loadable 3D Slicer module `LightWeightRobotIGT` was developed to operate the exemplary state machine. These components are described in the following section.

A state machine (see Fig. 3) was developed providing states typically needed in IGT interventions, using the design pattern of Vlassides et al. [18]. These states are *MoveTo*, *Path*, *VirtualFixtures* and *Free*. Additionally, an *Idle* state is defined as well, in which the robot is holding its position with maximum stiffness. In the *MoveTo* state the robot moves

²http://aimlab.wpi.edu/research/projects/OpenIGTLink_Interface.

with maximum stiffness on the shortest path in Cartesian space to the pose received in the command string (see Table 2 for command strings and parameters). This state is applicable for autonomous movements, e.g. for needle or drill positioning. In *Path* mode the operator can move the robot manually from a starting pose towards the destination received from the state control on a direct path. This is used for prepositioning of the robot within the operating field. The *VirtualFixtures* state, in which a plane or cone can be defined as an active constraint [17] in the workspace, is suitable for workspace restrictions for hands on control. In the *Free* mode, the robot can be moved freely with active gravitation compensation and can be used e.g. for registration purposes. The allowed transitions are illustrated in Fig. 3. Here a direct transition from the impedance control based state *Path* to the position control based state *MoveTo* is not authorized, to avoid abrupt changes in the stiffness values. Furthermore, a transition to the states for prepositioning and targeting are only allowed, if the transformation from image to robot base coordinate system is known to the state machine.

Whenever a new command string is received via the state control interface, the state machine checks command string and parameters for changes. Requirements for potential transitions, e.g. whether the transform from image to robot base coordinate system is known or not, are tested if necessary as well. Afterwards, the new state is initialized/reinitialized according to the received command string and parameter sets. If this procedure was successful, the acknowledge string is mirrored back to the state control containing the UID of the corresponding command string. In case of error in the state machine the robot state is set to *Idle* and an OpenIGTLink status message is sent to the state control.

State machine interface

The *LWRStateControlIF* class is a child class of the *Thread3* class, which provides the possibility for multi-threading, and hence, allow for simultaneous execution of multiple threads. The *igtlink4j* message classes *OpenIGTLinkMessage*, *StatusMessage*, *StringMessage*, and *TransformMessage* are used for bidirectional communication between the state control (e.g. 3D Slicer) and the state machine running on the robot control. In the initialization phase of the interface, a listening server is bound to a port of the KUKA Sunrise control and waits for an incoming request of a client, e.g. the state control software. After a connection is established, the initialization is completed. In the cyclic phase, OpenIGTLink messages are received from and sent to 3D Slicer and the data is provided to the state machine. The used command strings need to follow the pattern “CommandName;parameter1;...;parameterN;”, e.g. for the *Free* mode the command string is “Free;” (see Fig. 3 and Table 2 for a list of the command names and parameters). The OpenIGTLink string provides the encoding type as well as the string length which enables safe string operations. The list of commands and their parameters for the exemplary state machine are shown in Table 2. The message received is sent back to the state control, e.g. the *LightWeightRobotIGT* module, to provide features like an “is alive check”. The device name field of the OpenIGTLink header is used to differentiate between acknowledgement strings “ACK” and commands strings “CMD”. Furthermore, the device name of the OpenIGTLink header contains a unified identification number (UID). Therefore, the device name field in the OpenIGTLink header is set to “CMD_UID” or “ACK_UID” for the

acknowledgement string sent back to the state control. The UID is used to identify missed packages and the timeliness of the data received. If an error occurs an OpenIGTLink status message is sent including the current status of the LWR.

Visualization interface

The visualization interface class *LWRVisualisationIF* is also based on the Thread class and the initialization is analogue to the state control interface. In contrast to the state control this is a unidirectional interface, thus, only data can be sent from the robot control to the visualization. Besides, only transform messages can be sent to the visualization software. To allow for the visualization of the end-effector pose as well as the configuration of the robot, there are two different modes: one for the visualization of the end-effector and another for the visualization of the entire robot configuration (see Fig. 4). In the first mode the homogeneous transformation matrix ${}^B T_{EE}$ containing the end-effector pose with respect to robot base or image base frame is sent in each cycle, whereas in the second mode in every cycle the pose of each joint ${}^{n-1} T_n$ is transmitted to the visualization software. Furthermore, the user can choose between end-effector pose in image space or in robot space in both modes.

State control module

A loadable 3D Slicer Module named *LightWeightRobotIGT* was built as exemplary operating software for the developed interfaces. An OpenIGTLinkIF connector node is created for communication with the state machine via state control interface. Currently, this module offers the possibility to send command strings to operate the exemplary state machine, which are following the patterns described in the previous section (see Table 2). Push buttons are used to change the command strings, whereas the parameters can be set using combo boxes and line edits. In addition to the commands used to change the state of the state machine, commands to activate/deactivate the visualization interface, and to shut down the state machine are supported. Furthermore, the sent UID is increased in each cycle. For visualization purposes a 3D model of the robotic system can be loaded and the separate segments of the robot are linked to the corresponding transformation matrices ${}^{n-1} T_n$ automatically. The received end-effector pose can be saved into a fiducial list and can be used for registration purposes.

Experiments and results

In the test setup, 3D Slicer serves as an integrated graphical user interface for the operator to monitor and guide the surgical procedure. Therefore, the loadable 3D Slicer module *LightWeightRobotIGT* is used, which is available via GitHub.⁴ To visualize the end-effector pose while it is moving, 3D Slicer receives the coordinates of the current end-effector pose using OpenIGTLinkIF Connector nodes. Simultaneously, a 3D model of the end-effector or the complete robot configuration as well as a 3D model of the anatomy created from pre-operative 3D images are displayed (see Fig. 4). Consequently, the operator can monitor the process intuitively. On the robot control a state machine is implemented in Java using the SmartServo package to set the new destination and the control parameters, e.g. stiffness and damping values for impedance control. The state machine, the two OpenIGTLink interfaces,

and communication with the LWR can all be written in Java, which reduces the integration complexity. Additionally, there is no need for middleware or message converter, which allows for a direct and less complex interface.

Communication structure

The KUKA Sunrise robot control and the Slicer workstation (Latitude E6520, DELL Inc., Round Rock, TX, USA) are connected via a peer-to-peer Ethernet TCP-based network using the OpenIGTLink protocol. In the main task of the state machine on the robot control the current destination and, if indicated, the stiffness and damping values are changed in a 10 ms cycle utilizing the SmartServo package. Two separated periodic threads provide the communication with the state control and visualization (see Fig. 5) using the visualization and state control interface class, respectively each cycle rate can be chosen separately. In 3D Slicer, two OpenIGTLinkIF connector nodes are used to handle the communication with the robot control. To analyze the performance of the setup the cycle rate was measured and statistically analyzed.

Results

To analyze the reliability and the performance of the described interfaces experiments have been conducted to validate the ability of this concept for the given context. With the described test setup using the visualization interface class and 3D Slicer for the visualization the current pose of the tool or of all seven joints of the LWR can be sent with update rates greater than 60 Hz. The cycle rate of the state machine thread was not influenced by the cycle time and performance of the two interface classes. The latency of the visualization of the robot was analyzed by recording the movement in 3D Slicer and the movement of the LWR with one camera (HCX909, Panasonic, Osaka, Japan). Hence, the result includes the delay induced by the visualization of 3D models in 3D Slicer. To reduce this effect only the end-effector position was displayed. The video was captured with 50 Hz and a resolution of $1,920 \times 1,080$ pixels and the visualization interface thread was operated with a cycle time of 15ms. The LWR was moved utilizing the *MoveTo* State fourteen times and the beginning and end time of the movement of the virtual and the real LWR was determined ($n=28$). The median value of the latency was 60ms (3 frames) and the minimum and maximum value were 60 ± 20 ms (one frame). Thereby, the visualization interface meets the demands of the given purpose [2]. Furthermore, communication with the state control was tested with an update rate of 20ms whereas the state machine was running with a cycle time of 10ms and the visualization interface with 20 ms. During a test run of 272.8 s 13,640 command strings were successfully sent and 13, 640 acknowledgement strings were received with a mean cycle time of 20.38s. No data loss was detected and no package was sent multiple times.

Conclusion

In this work, an OpenIGTLink based visualization interface and state machine interface, which are following the integration concept in [1], were presented. Therefore, an exemplary java based state machine was implemented on the KUKA Sunrise robot control and a 3D Slicer state control module as a user interface. Just one programming context is needed, because java could be used for both OpenIGTLink interface classes and the state machine.

Thus, the integration cost is expected to be reduced by the decreased complexity of the integration process and the OpenIGTLink interface can be directly integrated into the robot control software. The latter helps to reduce error sources and communication overload. The state machine based control concept allows for an easy implementation of the robotic surgical workflow and, furthermore, can be easily expanded. With the extended 3D Slicer version the state machine can be operated and the visualization of the current robot pose is possible exceeding update rates of 60 Hz. Eventually, these results proof the applicability of the integration concept.

Outlook

Future work will focus on the standardized sensor interface to complete the implementation of the interface concept. Furthermore, an extension of the current 3D Slicer module is planned, which allows for the interactive definition of destination points for targeting and prepositioning.

Acknowledgments

The authors wish to thank Gregory Fischer and Nirav Patel from the WPI for the provision of the java implementation of OpenIGTLink igtlink4j as well as the Organizers of the NAMIC 2013 Summer Project Week. This work is supported in part by the National Institute of Health (R01CA111288, P01CA067165, P41RR019703, P41EB015898, R01CA124377, R01CA138586, R42CA137886, and U54EB005 149) and is funded by KUKA Laboratories GmbH (Augsburg, Germany).

References

1. Lemke HU, Vannier MW. The operating room and the need for an IT infrastructure and standards. *Int J Comput Assist Radiol Surg.* 2006; 1:117–121.
2. Tauscher S, Neff T, Ortmaier T. Interface concept for the integration of a robot into an image-guided therapy system. *Proceedings of the 27th International Congress on Computer Assisted Radiology and Surgery (CARS 2013).* 2013
3. Schreiber, G.; Stemmer, A.; Bischoff, R. The fast research interface for the KUKA lightweight robot; *IEEE Conference on Robotics and Automation (ICRA);* 2010.
4. Bzostek, A.; Kumar, R.; Hata, N.; Schorr, O.; Kikinis, R.; Taylor, RH. Medical image computing and computer-assisted intervention-MICCAI 2000. Berlin: Springer; 2000. Distributed modular computer-integrated surgical robotic systems: implementation using modular software and networked systems; p. 969-978.
5. Treichel T, Gessat M, Prietzel T, Burgert O. DICOM for implantations-overview and application. *J Digit Imaging.* 2012; 25(3):352–358. [PubMed: 21858592]
6. Xia T, Baird C, Jallo G, Hayes K, Nakajima N, Hata N, Kazanzides P. An integrated system for planning, navigation and robotic assistance for skull base surgery. *Int J Med Robot Comput Assist Surg.* 2008; 4(4):321–330.
7. Tokuda J, Fischer GS, Papademetris X, Yaniv Z, Ibanez L, Cheng P, Hata N. OpenIGTLink: an open network protocol for image-guided therapy environment. *Int J Med Robot Comput Assist Surg.* 2009; 5(4):423–434.
8. Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin JC, Pujol S, et al. 3D Slicer as an image computing platform for the quantitative imaging network. *Magn Reson Imaging.* 2012; 30(9): 41–1323.
9. Egger J, Tokuda J, Chauvin L, Freisleben B, Nimsky C, Kapur T, Wells W. Integration of the OpenIGTLink Network Protocol for image-guided therapy with the medical platform MeVisLab. *Int J Med Robot Comput Assist Surg.* 2012; 8:282–290.

10. Elhawary H, Liu H, Patel P, Norton I, Rigolo L, Papademetris X, et al. Intraoperative real-time querying of white matter tracts during frameless stereotactic neuronavigation. *Neurosurgery*. 2011; 68(2):16–506. [PubMed: 21304330]
11. Tokuda, J.; Pan, L.; Lorenz, CH.; Jolesz, FA.; Tempany, CM.; Hata, N. 26th International Congress and Exhibition. Pisa, Italy: 2012 Jun 27–30. Smartphone interface for interactive MRI, Computer Assisted Radiology and Surgery (CARS 2012).
12. Tokuda J, Fischer GS, DiMaio SP, Gobbi DG, Csoma C, Mewes PW, Fichtinger G, Tempany CM, Hata N. Integrated navigation and control software system for MRI-guided robotic prostate interventions. *Comput Med Imaging Graph*. 2010; 34(1):3–8. [PubMed: 19699057]
13. Enquobahrie A, Cheng P, Gary K, et al. The image-guided surgery toolkit IGSTK: an open source C++ software toolkit. *J Digit Imag*. 2007; 20(Suppl 1):21–33. 28.
14. Bartha L, Lasso A, Pinter C, Ungi T, Keri Z, Fichtinger G. Open-source surface mesh-based ultrasound-guided spinal intervention simulator. *Int J Comput Assist Radiol Surg*. 2013; 8(6): 1043–1051. [PubMed: 23729333]
15. Tavares, DM.; de Paula Caurin, GA. Proposal for a device proxy using XIRP. *Industry Applications (INDUSCON)*; 2010 9th IEEE/IAS International Conference on; 2010. p. 1-6.
16. Mönnich, H.; Wörn, H.; Stein, D. OP sense-A robotic research platform for telemanipulated and automatic computer assisted surgery. *Advanced Motion Control (AMC)*; 12th International Workshop on IEEE; 2012. p. 1-6.
17. Jakopc M, Rodriguez y Baena F, Harris SJ, Gomes P, Cobb J, Davies BL. The hands-on orthopaedic robot. *Robotics and Automation. IEEE Trans*. 2003; 19(5):902–911.
18. Vlassides J, Helm R, Johnson R, Gamma E. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, Reading. 1995

Fig. 1.

Overview of the interface concept for the integration of a robot into an IGT system

Fig. 2.
Communication concept with an OpenIGTLink message converter (left) and without the additional layer with an OpenIGTLink Interface integrated into the robot control

Fig. 3.

Finite state machine implemented as an example and the list of internal/external (the received command string) transition conditions

Fig. 4.
Visualization of the current robot configuration, end-effector position and the 3D Model from a sawbone specimen in 3D Slicer using the incoming data from the robot control and the experimental setup at the Institute of Mechatronic Systems

Fig. 5.

Block diagram showing the inter task communication and the interfaces to the control and visualization software, e.g. 3D slicer

Table1

List of interface classes and the requirements

Interface	Requirements	Data
State control	Reliability is-alive-check rate: up to 50Hz bidirectional	Commands parameter status
Visualisation	Real-time ability rate: exceeding 30Hz unidirectional	Transforms
Robot	Real-time ability rate: up to 500Hz bidirectional	Destination control parameter

Table 2

List of the command strings and their parameters for operating the exemplary state machine (img = image space, rob = robot base coordinate system)

Command string	Parameter (command)
IDLE	–
Free	–
VirtualFixtures	Coordinate system = {img, rob}; Geometry = {cone, plane}; Position vector = $x;y;z$; Normal vector = $n_x;n_y;n_z$; Cone angle = α ;
Path	Coordinate system = {img, rob}; Destination = $x;y;z;A;B;C$;
MoveTo	Coordinate system = {img, rob}; Destination = $x;y;z;A;B;C$;