ROS-Based Cognitive Surgical Robotics

Andreas Bihlmaier, Tim Beyl, Philip Nicolai, Mirko Kunze, Julien Mintenbeck, Luzie Schreiter, Thorsten Brennecke, Jessica Hutzl, Jörg Raczkowsky and Heinz Wörn

Abstract The case study at hand describes our ROS-based setup for robot-assisted (minimally-invasive) surgery. The system includes different perception components (Kinects, Time-of-Flight Cameras, Endoscopic Cameras, Marker-based Trackers, Ultrasound), input devices (Force Dimension Haptic Input Devices), robots (KUKA LWRs, Universal Robots UR5, ViKY Endoscope Holder), surgical instruments and augmented reality displays. Apart from bringing together the individual components in a modular and flexible setup, many subsystems have been developed based on combinations of the single components. These subsystems include a bimanual telemanipulator, multiple Kinect people tracking, knowledge-based endoscope guidance and ultrasound tomography. The platform is not a research project in itself, but a basic infrastructure used for various research projects. We want to show how to build a large robotics platform, in fact a complete lab setup, based on ROS. It is flexible and modular enough to do research on different robotics related questions concurrently. The whole setup is running on ROS Indigo and Ubuntu Trusty (14.04). A repository of already open sourced components is available at https://github.com/KITmedical.

Keywords Cognitive robotics · Medical robotics · Minimally-invasive surgery · Modular research platform

1 Introduction

Research into the robot-assisted operating room (OR) of the future necessitates the integration of diverse sensor and actuator systems. Due to the rapidly progressing

A. Bihlmaier (✉) · T. Beyl · P. Nicolai · M. Kunze · J. Mintenbeck · L. Schreiter · T. Brennecke · J. Hutzl · J. Raczkowsky · H. Wörn
Institute for Anthropomatics and Robotics (IAR), Intelligent Process Control and Robotics Lab (IPR), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
e-mail: andreas.bihlmaier@kit.edu

H. Wörn
e-mail: woern@kit.edu

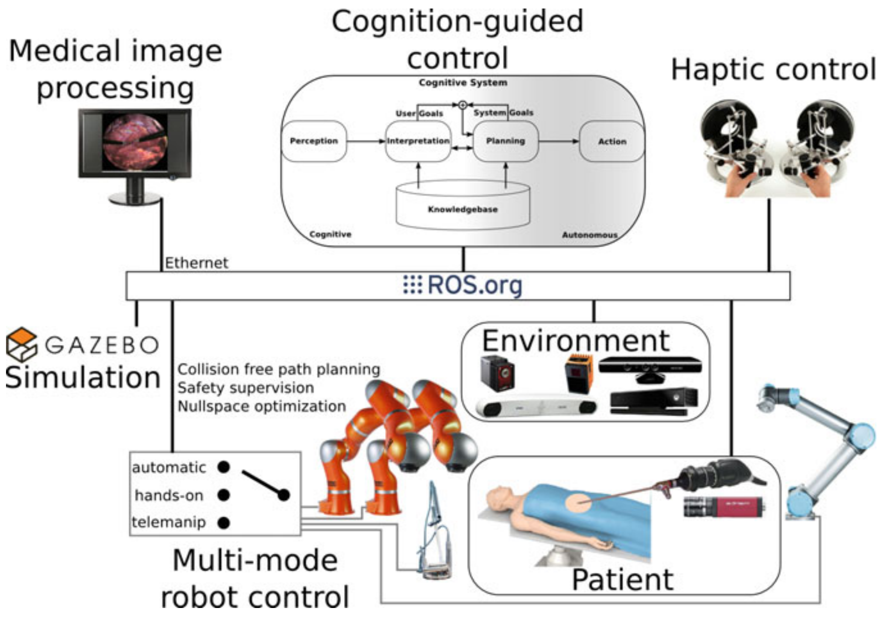

Fig. 1 Overview of our modular ROS-based research platform for robot-assisted minimally-invasive surgery (cf. [4]). The system contains several components to perceive the environment, the patient and the user. Physically actions can be executed by different robots. Interchangeability of components is essential, i.e. higher level algorithms should not have to know which robot, tracking system or camera is used. The whole setup also exists as a virtual model for the robotics simulator Gazebo. Thus algorithms can be evaluated in simulation on a single host without the necessity of accessing the real lab setup. This benefits both researchers and students by reducing the problem of scheduling lab access

state of the art, the volatility of project funding and the diversity of research questions, a modular and flexible platform is essential. Instead of each researcher developing a separate setup for his project, synergies can be taken advantage of if the core platform is used and extended by multiple researchers.

The ROS-based OP:Sense platform [9] tries to accomplish this for robot-assisted minimally-invasive surgery (MIS). Figure 1 shows the currently integrated components. Some components are directly used in the research applications. For these ROS provides a standardized and network transparent interface to use them—concurrently—from any computer within the lab network. This way many lab resources can be shared and do not have to be under exclusive control, e.g. sensors. Other resources require arbitration, e.g. control of robot manipulators. Yet, it is still advantageous that they are not bound to a specific control host. However, the more interesting platform subsystems are those relying on multiple resources and present these resources as more capable components to research applications. For example, it is either possible to control the redundant lightweight robots directly from the application or to interface them through a subsystem. This subsystem uses the environment sensors to optimize their redundant degree of freedom automatically. Examples of research applications include an automated endoscopic camera guidance robot [2], intuitive human robot interaction in the OR based on multiple fused Kinect cameras [1] and probabilistic OR situation recognition [7].

The remainder of the chapter consists of the following topics:

- First, a brief introduction to robot-assisted surgery and current research topics in this field are provided.
- Second, the general lab setup in terms of computing and networking hardware is described.
- Third, an overview of the diverse components that are integrated into the platform. For some components we rely on packages provided by the ROS community. These will be referenced and briefly discussed. The components that were either heavily adapted or created in our lab and released to the community will be treated in greater detail. Our components comprise different perception components (Kinects, Time-of-Flight Cameras, Endoscopic Cameras, Marker-based Trackers, Ultrasound), input devices (Force Dimension Haptic Input Devices), robots (KUKA LWRs, Universal Robots UR5, ViKY Endoscope Holder), surgical instruments and augmented reality displays.
- Fourth, following the single components, this part focuses on subsystem, which provide higher level functionality through the combination of multiple components. The subsystems include a bimanual telemanipulator, multiple Kinect people tracking, knowledge-based endoscope guidance and ultrasound tomography.
- Fifth, we will focus on organizational and software engineering aspects of the overall setup. In our case the robot actually is the whole lab. Therefore, change and configuration management are particular problems to be addressed.

2 Background

Research in the field of surgical robotics has seen a similar shift to research in industrial robotics. Instead of aiming for fully automated systems with a minimum of human involvement, the goal is to provide the surgeon and more general the operating room (OR) staff with cooperating assistance systems. These systems require information about the current situation within the OR, in particular about the tasks and activities of the OR staff. The robots in the OR must also be sensitive to their immediate surroundings instead of simply executing a pre-programmed task. Research questions pertain to improve the robot's capabilities of perception, planning or action. In order to facilitate research that relies on progress in multiple of these dimensions, we designed a ROS-based modular platform for cognitive surgical robotics.

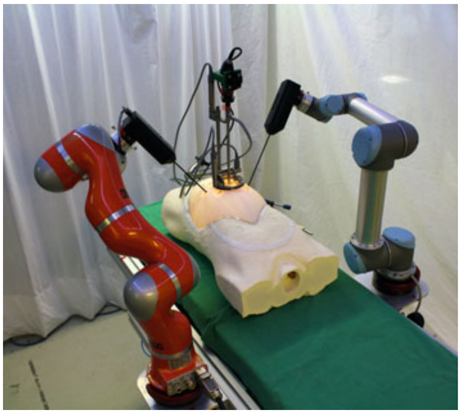

Fig. 2 An example slave side of the telemanipulation scenario featuring three different types of robots: Two manipulating robots (LWR, UR5) with attached articulated tools in the minimally-invasive configuration and the ViKY system holding the endoscopic camera

3 ROS Environment Configuration

Although industrial robot manipulators are part of our lab setup, the actual robotic system is the whole operating room. For that reason, each component and subsystem of the platform will be described on its own in the following chapters. Beforehand, only a few basic points about the complete computing and networking hardware are presented here. The first part of the computational infrastructure are the core machines that provide a ROS abstraction to the components and subsystems. The second part are the individual researchers' client machines, which run the high-level applications. In the core platform a large Gigabit Ethernet switch serves as central networking hub. Some large components contain an additional internal network hierarchy that is connected to the central switch (e.g. Sects. 4.4 and 4.5). Core computing is distributed across about 15 computers. For the most part these consist of commodity desktop machines, some with high-end graphics cards (see Sect. 4.6), some small form-factor PCs and a few single-board computers. All core machines run Ubuntu Linux.

4 Components

4.1 Robots

KUKA LWR IV Each of the KUKA LWR arms (cf. left robot in Fig. 2) has seven degrees of freedom with position and torque sensors in each joint. Both robots are controlled from a single PC. The communication protocol between the robots and the PC—the Fast Research Interface (FRI)—is provided by KUKA and is based on UDP. We control the robots with 1 kHz update rate, so the computer has to respond within 1 ms. If the computer does not meet the cycle deadline, i.e. does not respond in time to the measured joint data with new target parameters, the robot control performs an emergency stop of the robot. Thus, for each robot there are two threads. One thread is running at the highest Linux kernel priority handling the FRI communication. FRI joint positions and torques are stored and converted into Cartesian position and forces/torques. Then one interpolation step towards the current target is performed, either only in joint space or first in Cartesian space and then validated and truncated within joint space. The other thread with normal priority manages the ROS communication. Its update rate is decoupled from the robot and flexible, depending on the task.

We decided to use the publisher/subscriber mechanism since the `actionlib` is not suited for our scenario where we have constantly changing target positions. These are due to the fact that we operate in a highly dynamic and unpredictable environment. The robot is controlled online by a human instead of moving on a predefined trajectory. Messages to the robot will not be forwarded directly, instead they update the target for the interpolation, so velocity and acceleration constraints are handled by the controller and the pose update rate is flexible. For robot control in joint space we use `sensor_msgs/JointState`, both for reading the current robot position and commanding a new target. In the second case, the semantics of the velocity field are adapted to command the maximum joint velocity and the effort field is used to command the maximum joint acceleration. For Cartesian control we use `geometry_msgs/Pose` and specify fixed velocity and acceleration limits in an initialization file. The robot controller PC handles some further low level control strategies like hands-on mode and an optimization for the elbow position (the robot redundancy). In hands-on mode, each joint accelerates into the direction of external torque (the compensation of gravity and dynamic torques is done by the KUKA controller), while heavily decelerated by viscous friction, so the robot comes to halt quickly after being released. To optimize the elbow position, a cost function is created which considers the following aspects: distance of each joint to its limits, distance of each hinge joint to its stretched position (to avoid singularities), distance of the elbow to an externally specifiable target. The last point is used for interacting with surgical personnel, which is further described in Sect. 5.3.

Universal Robots UR5 The UR5 robot is supported out of the box through the universal_robot stack provided by the ROS Industrial community. However, in addition to the actionlib interface, we added the same topic interface as described for the LWR IV above. Furthermore, a Gazebo plugin was developed that exposes the manufacturer proprietary network format (see Sect. 5.6).

Trumpf ViKY A ROS interface for the motorized endoscope holder ViKY was developed that allows servo position control over the network. The interface is the same as for the two other types of robots. But since the ViKY only has three degrees of freedom (DoF) a Cartesian topic is not provided, instead trocar coordinates are exposed. The latter consist of spherical coordinates centered in the remote center of motion, which is defined by the small trocar incision in the patient's abdominal wall.

Path Planning and Collision Avoidance For both lightweight manipulators, to be more precise for any combination of manipulators and tools attached to the OR table, a URDF robot description has been created.[1] Based on the robot's URDF we use MoveIt! for path planning and collision avoidance. Path planning is an optional component in some configurations of the system (cf. Sect. 5.1). However, collision avoidance is always active with respect to all static parts of the environment.

4.2 Endoscope Cameras

Endoscopic cameras are a prerequisite for minimally-invasive surgery, thus the platform provides not only one but two kinds of endoscope cameras. The first camera is a medical device, R. Wolf Endocam Logic HD, which is interfaced through a HD-SDI connection with a PC-internal video capture card, Blackmagic Design DeckLink. A ROS driver, decklink_capture, was developed which provides the images using image_transport. The second camera is an industrial GigE Vision camera, Allied Vision Manta G-201, which is compatible with the community provided prosilica_driver package. Proper use of ROS name remapping and of the information provided in `sensor_msgs/Image` ensures that higher level applications transparently work with both cameras.

4.3 OR Perception System

One major focus of OP:Sense is the perception of the robotic system and its environment, especially people acting around and interacting with the robots. As the environment in the OR is often very crowded, we developed a dense sensor system to avoid occlusions. The sensing system consists of the following components that each publishes its data under a separate topic namespace:

- an optical tracking system (ART): `/art/body1..n`
- a time-of-flight (ToF) 3D camera system (PMD): `/pmd/S3/camera1..n`
- a structured light 3D camera system (Kinect): `/kinect/camera1..n`

Figure 3 gives an overview of the realized network topology for the whole perception system. Figure 4 shows a picture of the real setup as realized in the laboratory. In the following, the components and their implementation as parts of our ROS network are explained in more detail.

[1] To cope with the many possible combinations, we defined the models in a hierarchical manner using the Gazebo SDF format, which we convert to URDF using the sdf2urdf converter provided by our pysdf package.

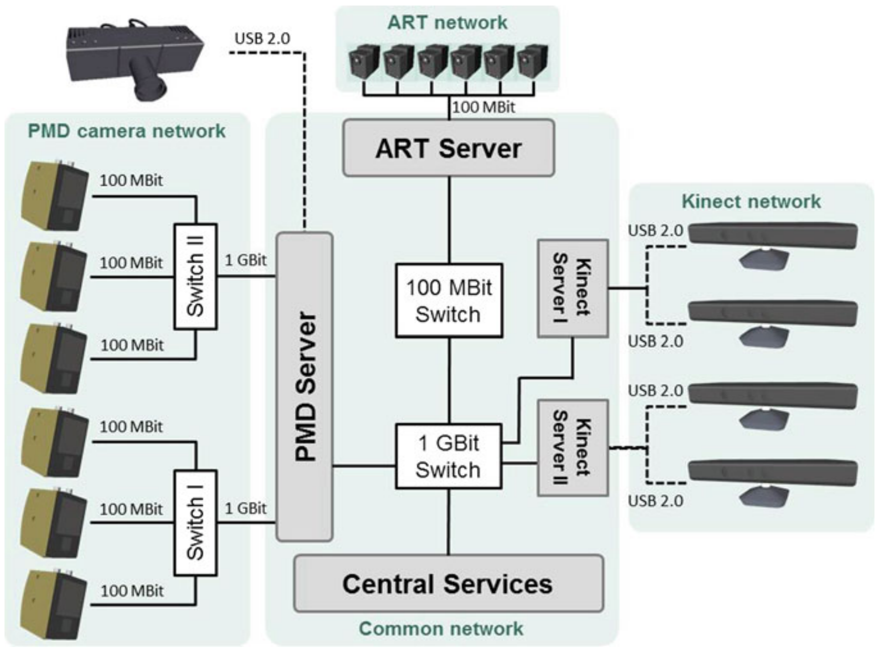

Fig. 3 Network topology of perception subsystems

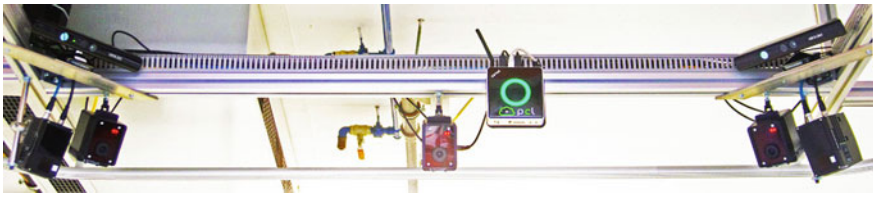

Fig. 4 One side of our sensor rig with different cameras: Kinect (*top left* and *top right*), ToF cameras (PMD S3, outermost), optical tracking system (ART, with two visible *red* LEDs), Kinect server (*center*, with visible *green circle*)

4.4 Marker-Based Optical Tracking

For high-accuracy 6D pose tracking e.g. of medical instruments, the ARTTrack2 system by ART is used. We use a six-camera configuration in order to reliably track rigid bodies in a volume over the OR table in presence of occlusions, e.g. by robots or humans. The tracking system provides a network stream of the tracked data in a simple proprietary network protocol. For usage in the ROS-based OP:Sense environment, a ROS node was implemented that provides the pose of tracked objects both as a ‘raw’ data stream with all values provided in original ART format (`Float64MultiArray`) and as a `PoseStamped` message with correct `frame_id` to be directly accessible in ROS nodes and via the ROS command line tools.

The scientific value of the tracking system is the precise tracking of rigid bodies, e.g. to record and analyse the motions of surgical instruments during an intervention. From a technical point of view, we use it as a day-to-day tool for conveniency tasks, such as quickly acquiring an object’s location, and routine tasks such as registration. The coordinate system defined by the optical tracking system is used as a world coordinate system, where applicable. Registration methods have been developed for the different sensors and actors present in our system, such as cameras (3D and 2D, e.g. endoscopic), robots (end effectors equipped with marker spheres), medical phantoms and systems for augmented reality (projector and LED array).

4.5 Time-of-Flight Cameras

For low latency, low resolution 3D scene perception a multi-ToF-camera system is used. It consists of six pmd[vision] S3 cameras (64×48 px) and one pmd[CamCube] 2.0 (204×204 px). This camera system is for applications where speed matters more than high resolution, e.g. safety critical applications such as human-robot-interaction in a shared workspace and collision avoidance.

As ToF cameras are prone to interference when used in the same space, we implemented a time- and frequency-based multiplexing synchronization. The cameras are controlled via their proprietary API from a dedicated PC over Ethernet segments separate from the ROS network (and USB in case of the CamCube). The data acquisition and processing/publishing have been split into two different threads, thus a slower processing/publishing (e.g. publishing to many subscribers via TCP) does not affect the rate with which the cameras are triggered.

Acquired data is lightly preprocessed (filtering typical ToF artifacts such as jumping pixels) and published onto the ROS network with the according data types, e.g. `sensor_msgs/PointCloud2` for point clouds and using image_transport for the amplitude, depth and (in case of CamCube) greyscale images. As the low-level details of the camera control such as time-multiplexed triggering are hidden from the ROS network, there is also no direct low-level per camera control exposed via ROS (such as offered in the pmd_camcube_3_ros_pkg package). Instead, control is implemented as a service that allows to reconfigure the camera system to different preconfigured modes, e.g. a mode with high frame rate and reduced integration time (resulting in higher noise) or with high integration times and subsequent triggering that offers the best data quality.

4.6 RGB-D Cameras

In order to allow a high resolution supervision of the scene, which is of great importance for human action detection within the operating room, RGB-D cameras are used in OP:Sense. More specifically, we utilize the Microsoft Kinect 360 cameras of which four are attached to a ceiling-mounted camera rig (see Fig. 4). The Kinect camera depth sensing is based on a structured light pattern which is projected onto the scene. This pattern in the infrared spectrum is observed using an infrared camera. The principle is based on stereo vision using an active component. Through the use of an astigmatic lens system, the light dots of which the structured light is composed are observed as ellipses with orientation. Their geometric relation and size is dependent on the position of the surface the light is projected, relative to the camera sensor and can therefore be used to reconstruct the 3D scene information. The output data of the Kinect cameras is a 11 bit depth map with 640×480 pixels resolution. In our system, this depth map is directly used for people tracking (Primesense NITE framework) in addition to the calculation of point clouds. The four Kinect cameras are mounted in a rectangular configuration of approximately 1.80×2.10 m over the OR table. The center of the field of view of each camera is in the center of the rectangle to observe the operating table and it's surrounding with the highest possible resolution that can be achieved with Kinect cameras.

The Microsoft Kinect 360 cameras are integrated into the ROS environment using the OpenNI package which allows for using the Kinect cameras from within a ROS system. To keep the Kinects' footprint in the OR as small as possible, we split the access and processing of Kinect data between different computers. Each two Kinects are connected to a small form-factor PC (Zotac ZBOX nano AD10, featuring two USB host controllers and a 1 GBit Ethernet port). This Kinect server runs the driver component of the OpenNI package and publishes the depth map and unprocessed RGB image on the network. The processing of all connected Kinect cameras is done on a central server located further away from the OR table, where sterility and space usage are not an issue. This approach allows for a flexible number of Kinect cameras which can easily be adapted to local requirements. The current small Kinect servers, which have been integrated in 2011, still use internal fans for cooling, which is problematic in the sterile OR environment. By today, the same system could be easily realized with passive-cooled Kinect servers.

In addition, a new Kinect-based camera system using the Time-of-Flight based Kinect One is currently being integrated. This system uses a four camera configuration like the system described above. However, for the upcoming Kinect One system we use a small computer for each camera running Windows 8.1 that already performs initial data processing such as point cloud calculation and user tracking based on the Microsoft Kinect SDK 2.0.

The processed data is published to ROS using a custom bridge based on win_ros. At time of implementation, win_ros was available only for Visual Studio 2010 whereas Kinect SDK 2.0 requires Visual Studio 2012, so we split the ROS bridge into two components that exchange their data using shared memory. Data is published using the following message types:

- `sensor_msgs/PointCloud2`: Organized, 512 × 424 px RGB point cloud (extended with a "uid" field that encodes the user id if the according point corresponds to a tracked user).
- `sensor_msgs/Image`: 1920 × 1080 px RGB image.
- `geometry_msgs/PoseArray`: Joint poses per tracked human.
- `UInt8MultiArray`: Tracking state for each joint per tracked human (as defined by Kinect SDK 2.0: not tracked, inferred, tracked).

To deal with the expected data volume, a new 10 GBit Ethernet segment is currently added to our ROS network through which the published data will be transferred to a Ubuntu-based workstation for further processing.

4.7 Input Devices

One of the main research topics in the scope of OP:Sense is human robot cooperation. Therefore interfaces that allow for a natural interaction with the system are required. In the telemanipulation scenario the surgeon directly controls the robotic arms using haptic input devices accessible from a master console. The input system is composed of a left hand device, a right hand device and a foot pedal. Monitors are included in the master console providing an endoscopic view together with additional information about the environment as well as the system and intervention state. The haptic input devices are 7 axis devices from Force Dimension. In OP:Sense we use the Sigma.7 device (right hand) and the Omega.7 device (left hand). The devices are composed of a delta kinematic for translational movement in Cartesian space coupled to a serial kinematic for rotational input. Additionally, a gripper is attached at the handle of the input device to actuate medical grippers or coagulators attached to the medical robot. The Omega.7 device (Fig. 5) is capable of displaying translational forces and gripper forces to the user. The Sigma.7 device additionally allows to render torques on the rotational axis.

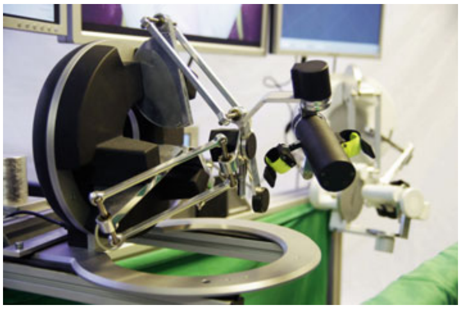
Fig. 5 The Omega.7 haptic input device mounted at the master console

For the integration of these devices, a ROS wrapper based on the available Linux driver of the haptic input devices has been written. This wrapper is a ROS node which is run for each device. It allows to access the pose of the device using a topic of type `geometry_msgs/PoseStamped`. The gripper position can be accessed via a topic of type `std_msgs/Float32`. In order to render forces and torques on the devices, a topic of type `std_msgs/Float32MultiArray` with 3 translational entries, 3 rotational entries and 1 value for the gripper opening is used. The foot pedal component includes two pedals and is used in the telemanipulation scenario as a clutching system (deadman switch) and to change the scaling factor. The pedal component is a medical pedal that has been connected to a USB I/O adapter that can be accessed using an open source driver. Our ROS wrapper publishes all I/O channels as `std_msgs/Float32MultiArray`.

4.8 OpenIGTLink-ROS-Bridge

OpenIGTLink (Open Network Interface for Image Guided Therapy) is a standardized network protocol, used for communication among computers and devices in the operating room. The protocol provides a simple set of messaging formats, e.g. pose data, trajectory data, image data or status messages. Reference implementations are available in C/C++, Matlab and Java. OpenIGTLink is supported, among others, by 3D Slicer, IGSTK, MeVisLab, MITK and Brainlab. In order to connect components using OpenIGTLink with OP:Sense a bridge between OpenIGTLink and ROS was implemented. This bridge provides user-defined duplex communication channels between ROS nodes and OpenIGTLink connections. OpenIGTLink clients and servers are supported. The channels convert the messages into the required target format. Each channel is able to

- subscribe messages from ROS topics, convert them and send the converted messages to OpenIGTLink connections;
- receive messages from OpenIGTLink connections and publish them on ROS topics.

Table 1 shows the currently supported OpenIGTLink message types and the mapping to ROS message types. Due to the complexity of the OpenIGTLink `IMAGE` message, it had to be split into 3 ROS messages.

Table 1 The mapping between OpenIGTLink and Ros messages

OpenIGTLink message type	ROS message type
`POSITION`	`geometry_msgs/PoseStamped`
`TRANSFORM`	`geometry_msgs/TransformStamped`
`IMAGE`	`sensor_msgs/Image` `tf/tfMessage` `geometry_msgs/Vector3Stamped`

4.9 Ultrasound Imaging

The OP:Sense platform provides an ultrasound system for intraoperative imaging and navigation. The functionality of the system allows tracked live ultrasound imaging, acquisition of 3D volumes, ultrasound-guided interventions, intraoperative navigation and preoperative imaging fusion [5]. As hardware components the Fraunhofer DiPhAS ultrasound research platform, a 2D transducer with tracker marker mount, a tracking system and a workstation computer for data processing are used. The transducer can be mounted to a robot. As software components, the Plus toolkit for navigated image-guided interventions and 3D Slicer for visualization and planning are used. The components are connected via OpenIGTLink. Any OpenIGTLink compatible device can be used for tracking system or imaging. Alternatively, any device supported by Plus can be used.

The integration with OP:Sense is realized by the OpenIGTLink-ROS-Bridge component. Plus as the central component receives pose data from the bridge node which acts as an OpenIGTLink server. Pose data can be received from any ROS topic that provides either messages of type `geometry_msgs/PoseStamped` or `geometry_msgs/TransformStamped`. The tracked image stream is provided by Plus via an OpenIGTLink server. The OpenIGTLink-ROS-Bridge connects to the Plus server and makes the stream available in the ROS network.

4.10 Surgical Instruments

OP:Sense can be used with different types of surgical instruments

- Rigid instruments: Instruments that can be used for open surgery and are rigidly attached to the end effector of the robots, such as scalpels.
- Articulated minimally-invasive instruments: Standard minimally-invasive instruments with a motor for opening/closing of the instrument and one for the rotation along the instrument shaft.
- Flexible Surgical Instruments: Research instruments with flexible shafts that allow for better manoeuvrability and dexterity.

Articulated Minimally-Invasive Instruments The instruments are standard laparoscopic instruments with a modified gripping mechanism and a motorized rotation along the instrument shaft. These instruments are used in the minimally-invasive telemanipulation configuration of the OP:Sense system. Faulhaber brushless DC-servomotors with an integrated motion controller and a CAN interface provide motorization. An instrument control node serves as a bidirectional wrapper between ROS messages and appropriate CAN messages. After calibration, the node accepts a gripper opening angle on a `std_msgs/Float64` topic (in percentage of maximum angle) and publishes its current opening angle on another topic of the same type. The same also holds for the rotation angle of the instrument shaft rotation.

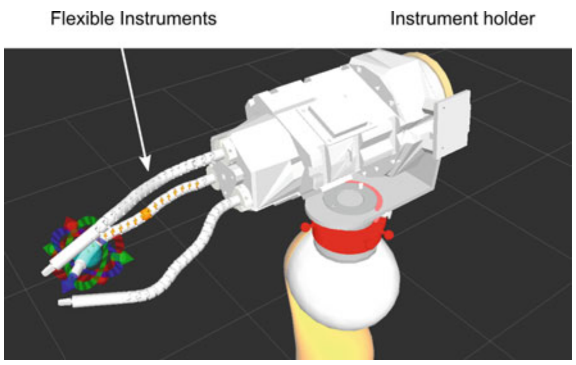

Fig. 6 Instrument holder with flexible instruments

Flexible Surgical Instruments For interventions in the abdominal area, a higher dexterity of the instruments than that achievable by traditional laparoscopic instruments can be benefitial. For this case, an instrument holder with three flexible instruments (see Fig. 6) was designed, that can be attached to the end-effector of a robot [8]. Using this holder, all three instruments can rotate around a common axis and move individually on a translational axis. The flexible part of the instrument, 180 mm in length and with a diameter of 10 mm, is composed of two segments. Each segment with a length of 90 mm consists of a stack of alternating rigid and soft elements, which are equipped with two Degree of Freedom (DoF) that are actuated individually by cables. The rigid elements are stereolithograpically printed, whereas the soft elements are made out of vacuum casted silicone. By pulling the cables, the silicone elements are deformed and the segment bends into the according direction. Additionally, in the center axis of the flexible structure a channel is realized where an optical sensor is integrated, capturing the current shape of the instrument. A gripper with one DoF is fixed at the tip of two instruments and a chip-on-the-tip camera module is attached to the third instrument. From the kinematical point of view, each flexible segment features 54 DoF. These are reduced to two main axis of motion, represented as revolute DoFs around the x- and y-axis per silicon element. The control of one instrument via ROS is realized by a `sensor_msgs/JointState` topic for the instrument holder and each instrument. Due to the negligible velocity of the joints, only the position field contains the x- and y-axis values for both segments. In the same way, there are topics for the grippers.

4.11 Augmented Reality

With respect to the field of human-machine-interaction, OP:Sense integrates modalities for spatial augmented reality, where information is directly provided in the scene without assistant devices such as AR glasses. For projecting information onto the situs, a full HD short-distance projector (Benq TH 682ST) is mounted over the OR

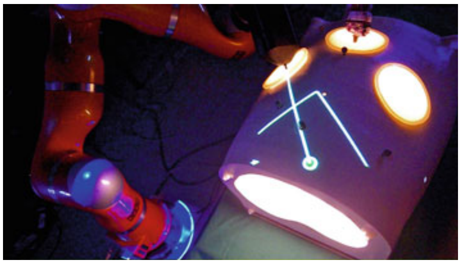
Fig. 7 Exemplary live scenario for spatial augmented reality for minimally-invasive interventions with projected instrument shaft, tip point and view cone of the endoscopic camera

table and is connected to one of the Kinect servers (see Sect. 4.6). Connection via HDMI is possible by the configuration of the vendor-specific graphics card driver in order to have a non-modified 1920 × 1080 pixel output signal.[2]

Rendering of content is performed using openframeworks, which we extended with a ROS interface. For displaying basic geometric information like projecting the trocar points and instrument poses onto a patient, we use the scalable vector graphics (SVG) format. We implemented a SVG preprocessor as part of our augmented reality node that parses the SVG string for a custom tf extension and replaces the tf frame ids with the correct coordinates from the point of view of the projector. If for example an application needs to highlight the robot's end effector (which pose is available on tf), it can simply send an SVG graphics that contains `<circle *tf [cx|cy] RobotToolTip /tf* r=100 fill=blue (...) />`. The augmented reality node will receive this message, continuously evaluate it based on the tf transformations available on the ROS network and project the according image. Figure 7 shows a demonstration scenario with live projection based on current system data available on tf. To keep the network load low, we realized a continuous reactive projection, e.g. onto a target that provides position updates with 100 Hz, by sending only one initial message to the projection node.

5 Subsystems

5.1 Telemanipulation

Telemanipulation is a concept to use robots in a master-slave mode. This means that input devices (masters) are used to control the robots (slaves) of a system. The concept can be used in hazardous environments such as chemistry labs. In medicine, telemanipulation is usually applied to minimally-invasive surgery (MIS) to allow the

[2]In our case, for an AMD graphics card on Ubuntu, the required command is: `aticonfig-set-dispattrib=DFP2,sizeX:1920` and `,sizeY:1080` as well as `,positionX:0` and `,positionY:0`.

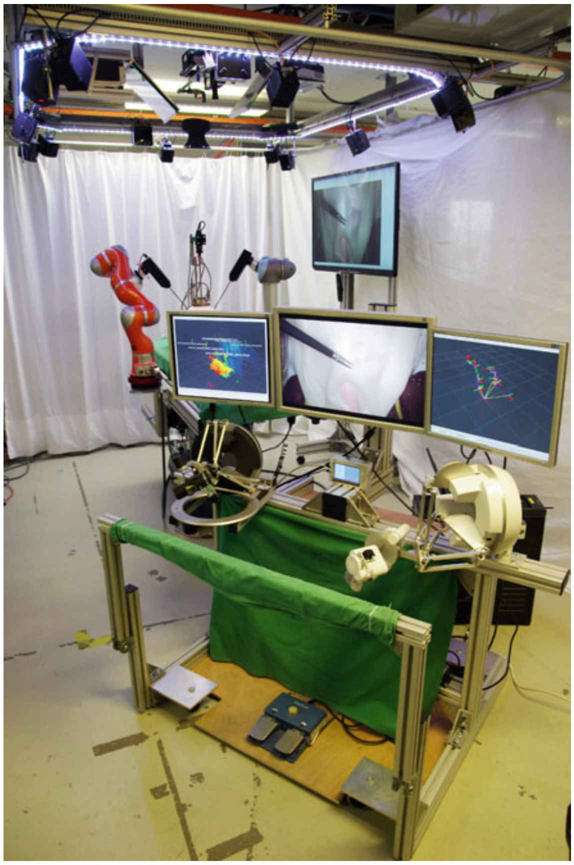
Fig. 8 Overview of a large part of the components (see Sect. 4) in our lab setup in a configuration for telemanipulation (see Sect. 5). *Front* Master console with haptic input devices, foot pedals and screens. *Back* Different robots mounted to the OR table. *Top* Perception and AR system

surgeon to work more similar to open surgery than is the case in traditional MIS. Additional goals are improvement of ergonomics and accuracy, e.g. by providing a scaling factor between human input and instrument motion.

Our system is composed of a master console with haptic input devices and slave robots, mounted to the operating bed, to which the surgical tools are attached. The system is depicted in Figs. 8 and 2. A third robot such as the UR5 or the ViKY is utilized to hold the endoscope providing vision to the situs. The instruments of the robots are introduced to the patient's body through trocars which are the remote center of motion for the telemanipulation task. This remote center of motion creates the fulcrum effect which causes a mirrored motion of the tip of the instruments compared to the surgeon's hand motion. In non-robotic surgery, this effect can only be compensated by the surgeon through practical experience. With a telemanipulated robot, this effect can be compensated in software. Additionally, the remote center of motion restricts the movement of the tooltip to four degrees of freedom (rotation around the tool shaft, three translations of the tooltip), which means that two rotations (hand wrist movement) are lost as long as no additional joints at the tooltip are used.

Finally, a problem in MIS is that the image from the endoscopic image is displayed on a monitor which is in most cases not registered to the surgeon's hand, thereby producing an additional cognitive load. In OP:Sense the complete system is registered with respect to the endoscopic camera whose image is displayed at the master console. Thus, the surgeon's vision of the instruments is in the same coordinate frame than his/her eyes, i.e. the instrument's position and orientation corresponds to the surgeon's hand position and orientation. The configuration of the telemanipulation system, e.g. setting which input device controls which robot, is completely controlled by launch files and can therefore be adapted to various combinations of input device and robot. The telemanipulator directly uses the robot controlling mechanisms of OP:Sense described above, i.e. the input and output of the telemanipulator is a `geometry_msgs/Pose`. This means that for a bimanual setup the telemanipulation system is launched twice with different robot-input device configurations.

The telemanipulation system is registered optically using the optical tracking system and the endoscopic image, working in the base frame of the tracking system. In addition, tracking is used to acquire the current position of all components during the operation. This allows to move the robots and the camera during the procedure while providing a stable coordinate system to the surgeon. The telemanipulation system is composed of the nodes listed below, which are implemented as nodelets to minimize the latency and the (de)serialization overhead:

- Input device node: The haptic device node described in Sect. 4.
- Telemanipulation node: This node receives the input device position, the robot position and in case of articulated instruments also the instrument position. Additionally, the node reads the calibration and periodically published tracking poses (`geometry_msgs/Pose`). The system is triggered through the update rate of the haptic input devices, that thereby serves as clock generator of 1 kHz. Each cycle new positions for the instrument tooltip are computed, which are published as `geometry_msgs/Pose` messages.
- Trocar node: A trocar point can be defined by means of an optically tracked pointing device. The device is pointed at the desired position at which moment a `std_srvs/Empty` service is called. The trocar node reads the current position of the pointing device and uses its position as trocar point. When the service is called for the first time, the trocar algorithm is activated. As soon as the trocar algorithm is running, it continuously calculates the necessary pose of the robot end effector while abiding to the trocar constraint (for each instrument pose, the trocar point has to be along the shaft of the instrument). This results in a robot pose that holds the instrument tip at the desired position, but neglects the desired orientation of the instrument. In case of the use of an articulated instrument, the trocar node also computes the required pose for the instrument. The final output of the trocar node is a topic of type `geometry_msgs/Pose` which is passed to the robot controller and a topic of type `std_msgs/Float64` representing the desired rotation of the rotation axis of the instrument in radian.

  The telemanipulator can also be used in an open surgery mode where no trocar is used. In this case the trocar algorithm is bypassed and the output of the telemanipulation mode is directly used to compute the new robot end effector pose based on the calculated tooltip pose. Articulated instruments are not used in this case as in open mode no trocar point is present and all 3 rotations and translations of the robot can be used.

- Robot controller: The robot controller directly drives the robot to the desired pose computed by the trocar node without using path planning. This is possible as the increments of the path in telemanipulation mode are small enough given an appropriate update rate of the telemanipulator.

As described above, the pedal serves as a dead man switch (which disconnects the robot when not pressed) and for setting the motion scaling. The pedal's current configuration is passed to a pedal node which passes the state of the clutching pedal as a `std_msgs/Bool` to the telemanipulation node. Whenever the clutch is closed, the telemanipulation node performs a registration routine after which any movement of the input devices and the robot is computed relative to the position of the robot and the haptic input device at the time the clutch closed. The second pedal switches between a set of scaling factors. Every time the scaling is changed, the current scaling factor is passed to the telemanipulation node as a topic of type `stdmsgs/Float32` where it is used to scale the motion of the haptic input device with respect to the robots movement.

5.2 Multi-RGBD People Tracking

Another important part in the scope of OP:Sense is the semantic perception of the environment. To allow for sophisticated human robot interaction such as situation detection or the optimization of the robot pose with respect to the human, information about the human pose is needed. For this purpose we utilize the Kinect system where each Kinect camera is registered to a reference one. The approach is based on OpenCV and PCL. For registration, we first detect a checkerboard in both RGB cameras (reference camera and camera to be registered), then we use the depth camera to find the 3D positions of the checkerboard corners in both Kinect frames. The resulting correspondences are then used to estimate the transform between the cameras. This is repeated until all cameras are registered to each other. In a second registration step we use the reference camera and the RANSAC plane detection algorithm to detect the floor plane which is later on used for the computation of people positions.

The aim of the people detection system is to integrate information about the position of the humans, the points representing the humans and the skeletal configuration of the humans. As written in Sect. 4.6, one workstation is dedicated to perform all computations on both the RGB and the depth images from the Kinect. A filter pipeline implemented as nodelets is used to perform a fusion on the heterogeneous data from the Kinect. The components of this pipeline are:

- People detection node: The Primesense NITE algorithm is used to detect and track people in the camera frames. The algorithm runs for every camera and extracts a depth image that only includes the pixels representing a human (other pixels are black). NITE is capable of detecting up to 16 humans at a time so we compute 16 depth images. We then perform an erosion operation on the depth images to remove some of the noise introduced by the use of multiple Kinect cameras. For each camera we then publish 16 topics whereas each topic is used for one of the users depth images.[3] Additionally we add the skeletal configuration of each human that is computed by the NITE algorithm with respect to the camera frame where the human is detected to the tf tree. Finally, we publish topics of type `std_msgs/Float32` to provide the keys of users detected and the keys of users of which tracking information are available for following computations.
- User image processor node: This node computes a 3D pointcloud for each human detected in a Kinect camera using the known transfer function of the Kinect. In the human detection system, this node runs once for every Kinect camera and subscribes to the 16 depth image topics representing the users. After the computation of a point cloud for every user, we perform an additional noise removal step using the statistical outlier removal algorithm from PCL. This produces a low noise point cloud for every human detected in the scene. For every possible detected human a `sensor_msgs/PointCloud2` topic is created which is used to publish the point clouds representing the users. When four Kinect cameras are used, this creates 64 topics that are either empty or contain information associated to a detected human.
- Fusion: This node is not triggered by the camera driver as the Kinect cameras are not running synchronously. Instead, it uses its own computation loop for clock generation. It subscribes to all point clouds representing users and to the array of detected users published by the NITE node. The purpose of this node is to combine the clouds of corresponding users that are detected by multiple cameras. As a measure to determine these correspondences, the centroid of each pointcloud representing a human is computed using the according PCL algorithm and projected to the floor. To allow for a more robust correspondence estimation, we project the centroids of the point clouds to the floor. The Euclidean distance between the centroids is used to concatenate the point clouds of humans which were detected in multiple cameras. The final point clouds are published as `sensor_msgs/PointCloud2`. Additionally we use a custom message holding a 2D array that provides the information about corresponding humans in multiple camera views.
- Distance computation: This node computes features that can be used to infer information about the current state in the OR. Features include distances between humans and different parts of the robots or between human and human. One particular feature is the distance between the human closest to the two robots. It is

[3]Unfortunately, the NITE nodes cannot be run as nodelets.

used as a simple low-dimensional input to the robot cost function in Sect. 4.1. The node subscribes to the combined point clouds of the Fusion node and uses CUDA to perform brute force Euclidean distance computation between the point clouds and CAD models of scene items. An example for such a CAD model is the robot in its current joint configuration.

5.3 Human-Robot-Interaction

In the design of a large modular surgical robotic platform, ease of use and intuitive interaction has to be considered, but also safety for the medical staff, the patient as well as the technical devices. Therefore, we combine probabilistic models and rule based functions in order to interpret the context information in an ongoing surgery. Context information can be modelled by using workflows. These workflows are composed of individual workflow steps presented as Hidden Markov Models (HMM). For each workflow step, a different HMM was trained using previously acquired training data. In Fig. 9 a representation of the target workflow (autonomous switching to the hands-on mode) is given. During the training phase we recorded various features to fit each HMM individually.

In the following example we employ four workflow steps and thus four HMMs (`'s'` = start, `'a'` = touch EE, `'b'` = move robot, `'l'` = release EE). Additionally, we annotate each step by capturing the corresponding keyboard input, including the ROS time to synchronize the recorded features and the keyboard input afterwards. The quality of the online classification strongly depends on the selected feature vector. One representative feature for the workflow is the minimal distance between the human and the robot end effector as calculated by the Multi-RGB-D People Tracking subsystem (cf. Sect. 5.2) and provided as a `std_msgs/Float`. Another feature is the robot's velocity which is included in the `sensor_msgs/JointState` messages provided by its controller (see Sect. 4.1). The last feature "approach" indicates if the human is moving towards or away from the robot derived from minimal distance feature. The implementation was done in Python, for which a Gaussian HMM

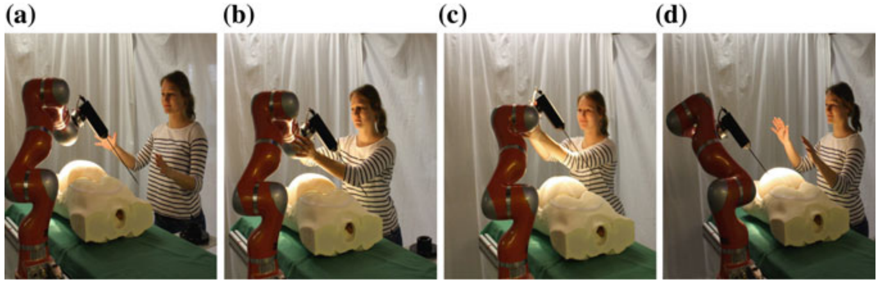

Fig. 9 Intuitive switching of a robot from position mode into Hands-On mode and back, based on the perception subsystem using Hidden Markov Models. **a** Start position 's'. **b** Touch EE 'a'. **c** Move robot 'b'. **d** Release EE 'l'

model is provided within the package scikit. After training of the HMMs, they can be used for realtime online classification in OP:Sense. The feature vector is passed to each HMM and evaluated by using the log likelihood probability implemented in scikit. The HMM with the highest likelihood will be set as the most likely state. The result state is published as a `std_msgs/String` for other subcomponents.

5.4 Endoscope Guidance

The teamwork between the surgeon and the human camera assistant in minimally-invasive surgery poses many challenges and opportunity for improvement. Therefore, we research having a robot instead of a human assistant guiding the endoscope. Motorized endoscope holders, such as ViKY (see Sect. 4.1), are already commercially available. However, they require that the surgeon manually controls every repositioning, thus further increasing his cognitive load. A cognitive endoscope guidance system is currently being researched that provides autonomous endoscope repositioning by means of a knowledge base. Since this assistance system is not developed standalone, but based on our ROS platform, many synergies can be exploited and additional functionality is provided to the surgeon. For further information, we refer to the detailed description in [2].

5.5 Ultrasound Tomography

The ultrasound imaging component can be used to create 3D volumes using the freehand-3D-ultrasound ability. To acquire more detailed volumes with fewer distortions a robotic ultrasound tomography was implemented. Using a robot mounted transducer it is possible to record equidistant and parallel slices which allows a direct volume reconstruction. In addition, the area of interest can be scanned from several directions, which can decrease speckle noise. Due to the force control scheme of the LWR robot, it is possible to acquire scans by following uneven surfaces with the probe such as the human body. As a tracking system, the pose data from the optical tracking system is subscribed. Alternatively it is possible to use the robot position as tracking data. The robot and ultrasound imaging system, using the OpenIGTLink-ROS-Bridge, is controlled from a Python ROS node. As an HMI, a GWT GUI using rosjs was implemented, so any device with a browser can be used for inputs. The results are displayed in 3D Slicer. The reconstructed volume can be published to the ROS network using the OpenIGTLink message type `IMAGE`.

5.6 Simulation

Due to the cost and complexity of the complete platform, only a single instance of it exists in our institute. It is shared by multiple researchers and their students. Naturally this leads to a bottleneck in scheduling time for complex experiments. The modularity and network transparency provided by ROS mitigate the problem to some extent, because sensor components and subsystems can be used in parallel. However, in the common case that multiple researchers require exclusive access to some components, e.g. the OR table mounted robots, another solution is required. Providing an accurate simulation of large parts of the setup, of which multiple instances, running on office desktop computers, can be used independently from each other, helped us a lot. More details about our use of the Gazebo simulator for this purpose and how the simulation can help to perform advanced unit and regression testing (Robot Unit Testing) is published in [3].

5.7 Software Frameworks

Table 2 provides an overview of important libraries and frameworks, besides ROS, that are used in our setup.

6 Organization and Software Engineering

6.1 Registration and Calibration

In OP:Sense, calibration and registration are organized in a separate ROS package. The usual reference for calibration is the optical tracking system. For calibration

Table 2 Major software frameworks used in our surgical robotics platform

Name	Website	Version(s)
3D Slicer	http://www.slicer.org/	4.4
Chai3D	http://www.chai3d.org/	
Gazebo	http://gazebosim.org/	4.0; 5.0
Matlab	http://de.mathworks.com/products/matlab/	
MITK	http://mitk.org/	
MoveIt!	http://moveit.ros.org/	
OpenCV	http://www.opencv.org	2.4; 3.0-beta
Openframeworks	http://openframeworks.cc/	
OpenIGTLink	http://openigtlink.org/	2
Plus	https://www.assembla.com/spaces/plus/wiki	2.2
Point Cloud Library (PCL)	http://www.pointclouds.org	
Scikit-learn	http://scikit-learn.org/stable/	0.15.2

purposes we use a tracking pointer with a rigid tip location which is found using pivotisation. Using the transformation acquired by pivotisation, we can now compute the position of the pointer's tip with reference to the optical tracking system. The pointer is then used to register other devices such as cameras to the optical tracking system using landmarks in the scene that can be touched with the pointer and can at the same time be detected within the camera's image. Using OpenCV we compute the transformation from the camera coordinate frame to the optical tracking system. If the camera body itself is equipped with markers, e.g. in case of the endoscopic camera, we can compute the transformation between the optical frame and the frame of the markers, allowing to precisely calibrate the optics to the tracking system. The calibrated pointer is also used for online calibration tasks such as setting new trocar points for the telemanipulation system. If a new transformation is computed through registration, the translation and the quaternion representing the pose are saved to file. These registration files can be loaded through a pose publishing node that is controlled via a launch file to publish the registration information on the tf tree and/or on a dedicated `geometry_msgs/Pose` topic.

6.2 TF and Pose Topics

OP:Sense uses both the tf tree and a `geometry_msgs/Pose` based mechanisms to send transformations. tf is usually selected if a transformation has to be visualized. Additionally, tf is beneficial if the publishing frequency of a transformation is low or average. For complex transformation chains this can easily deliver the transformation from a frame to any other frame in the tree. Unfortunately, tf is not as performant as publishing transformations on Pose topics and degrades further when transformations are published with high frequency. For fast transformation publishing such as in the telemanipulation use case (1000 Hz) we solely rely on sending transformations on dedicated `geometry_msgs/Pose` topics and perform the frame multiplications in the node. During the development phase, we usually keep the publishing frequencies low and use tf. At the end of a development cycle when the frequency is increased we switch to topics which enables most of the nodes to use both mechanisms. Additional benefits of this method are that transformations can at any time be visualized using RViz and the load on the tf is reduced to a minimum level.

6.3 Windows/Matlab Integration

The research and development process for the realization of math intensive components such as flexible instruments (see Sect. 4.10) or inverse kinematics for robots requires the use of a rapid prototyping environment. In our case Matlab/Simulink is the tool for development of the robotic system kinematics, workspace analyses and control design. However, the problem in our setup is, that Matlab runs on a

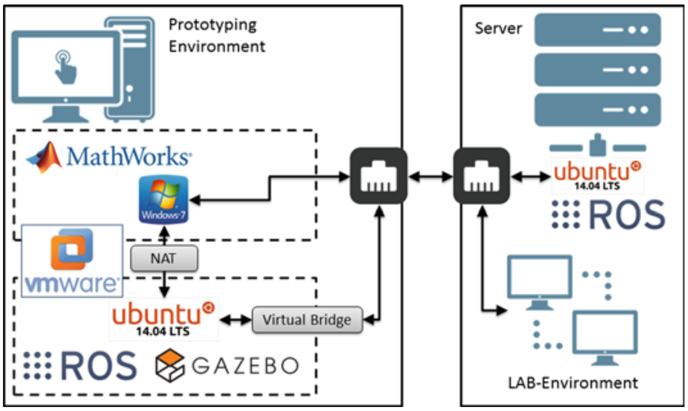

Fig. 10 Infrastructure to connect Gazebo via ROS with Matlab on Windows through a virtual Linux machine

"Windows-Desktop-PC" and the roscore of the lab on a Linux system. In order to control and interact with both systems, the lab setup, where the final hardware runs, as well as the prototyping system and simulation in Matlab on Windows, we decided to use virtualization (see Fig. 10). Inside the Linux guest system, Ubuntu 14.04 is installed including ROS (Indigo) and Gazebo (4.0). The communication from Windows to the lab environment is established by a physical network adapter with a dedicated static IP. Additionally, there is a virtual bridge from the virtual machine also using the physical network adapter of the host system. Now, it is possible to send ROS messages from Windows to the lab's ROS network in the same way as to the virtual one. The configuration inside the virtual machine is limited to starting a local roscore and setting up the environment variables accordingly. Regarding the Matlab part in this setup, the ROS I/O package for Matlab 2013b is used to communicate with ROS. Therefore, a Matlab class-object is implemented to create nodes, topics and the associated publishers and subscribers. For future work, the latest Matlab release 2015a with full ROS support via the Robotic System Toolbox will be used, thereby achieving the integration of ROS, Gazebo, Simulink and real-time hardware.

6.4 Software Repositories and Configuration Management

The balance between independence of each researcher on the one hand and keeping all components and subsystems working together on the other is difficult to get right. After trying out many different ways to organize the software and configuration of our platform, we suggest the following best practices for such a large setup: First, maintain one source code repository for each part that is usable on its own. Second, changes to core components that change an interface visible to the ROS network must be documented and agreed upon beforehand. Third, have one repository with a hierarchical set of launch files to bring up different parts of the core components and subsystems. Fourth, always mind the software engineering principles of "Single Point of Truth" (SPOT) or "Don't Repeat Yourself" (DRY) [6] with regard to code and configuration information. Fifth, maintain a (virtual) blackboard listing all available system functionality and links to documentation on how to use it.

References

1. T. Beyl, P. Nicolai, J. Raczkowsky, H. Worn, M.D. Comparetti, E. De Momi, Multi Kinect People Detection for Intuitive and Safe Human Robot Cooperation in the Operating Room. In: *2013 16th International Conference on Advanced Robotics (ICAR)* (2013), pp. 1–6
2. A. Bihlmaier, H. Wörn, Automated Endoscopic Camera Guidance: A Knowledge-Based System towards Robot Assisted Surgery. In *Proceedings for the Joint Conference of ISR 2014 (45th International Symposium on Robotics) and ROBOTIK 2014 (8th German Conference on Robotics)* (2014), pp. 617–622
3. A. Bihlmaier, H. Wörn, Robot Unit Testing. In *Proceedings of the International Conference on Simulation, Modelling, and Programming for Autonomous Robots (SIMPAR 2014)*, (2014), pp. 255–266
4. A. Bihlmaier, H. Wörn, ROS-based Cognitive Surgical Robotics. In *Workshop Proceedings of 13th International Conference on Intelligent Autonomous Systems (IAS-13)* (2014), pp. 253–255
5. T. Brennecke, N. Jansen, J. Raczkowsky, J. Schipper, H. Woern, An ultrasound-based navigation system for minimally invasive neck surgery. Stud. Health Technol. Inf. **196**, 36–42 (2014)
6. A. Hunt, D. Thomas, *The Pragmatic Programmer: From Journeyman to Master* (Addison-Wesley, Boston, 1999)
7. L. Schreiter, L. Senger, T. Beyl, E. Berghöfer, J. Raczkowsky, H. Wörn, Probabilistische Echtzeit-Situationserkennung im Operationssaal am Beispiel von OP:Sense. In: *13. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V*, (2014), pp. 177–180
8. J. Mintenbeck, C. Ledermann, R. Estana, H. Wörn, EndoSnake–A Single-Arm-Multi-Port MIRS System with Flexible Instruments. In: *13th International Conference on Intelligent Autonomous Systems (IAS-13)* (2014)
9. P. Nicolai, T. Brennecke, M. Kunze, L. Schreiter, T. Beyl, Y. Zhang, J. Mintenbeck, J. Raczkowsky, H. Wörn, The OP: sense surgical robotics platform: first feasibility studies and current research. Int. J. Comput. Assist. Radiol. Surg. **1**(8), 136–137 (2013)

Authors' Biography

Andreas Bihlmaier Dipl.-Inform., obtained his Diploma in computer science from the Karlsruhe Institute of Technology (KIT). He is a Ph.D. candidate working in the Transregional Collaborative Research Centre (TCRC) "Cognition-Guided Surgery" and is leader of the Cognitive Medical Technologies group in the Institute for Anthropomatics and Robotics-Intelligent Process Control and Robotics Lab (IAR-IPR) at the KIT. His research focuses on cognitive surgical robotics for minimally-invasive surgery, such as a knowledge-based endoscope guidance robot.

Tim Beyl Dipl.-Inform. Med., received his Diploma in Medical Informatics at the University of Heidelberg and University of Heilbronn. He is working as a Ph.D. candidate in the medical group at the IAR-IPR, KIT. His research focuses on human robotic interaction for surgical robotics, telemanipulation, knowledge based workflow detection and scene interpretation using 3D cameras.

Philip Nicolai Dipl.-Inform., received his a Diploma in computer science at Universität Karlsruhe (TH). He is working as a Ph.D. candiate in the medical group at the IAR-IPR, KIT. His research focuses on safe applications of robots to medical scenarios, safety in human-robot interaction based on 3D scene supervision and application of augmented reality for intuitive user interaction.

Mirko Kunze M.Sc., received his master's degree in mechatronics at Ilmenau University of Technology. He is working as a Ph.D. candidate in the medical group at the IAR-IPR, KIT. His research focuses on workspace analysis for redundant robots.

Julien Mintenbeck M.Sc., received his master degree in mechatronics at the university of applied science of Karlsruhe. He is working as a Ph.D. candidate in the medical group at the IAR-IPR, KIT. His research focuses on active controllable flexible instruments for minimally invasive surgery and autonoumous microrobots for biotechnological applications.

Luzie Schreiter Dipl.-Inform. Med., received her degree at the University of Heidelberg and University of Heilbronn. She is working as a Ph.D. candidate in the medical group at the IAR-IPR, KIT. Her research focuses on safe and intuitive human-robot interaction as well as workflow identification in a cooperative systems, such as surgical robotics systems for minimally-invasive surgery.

Thorsten Brennecke Dipl.-Inform., received his Diploma in computer science at TU Braunschweig. He is working as a Ph.D. candidate in the medical group at the IAR-IPR, KIT. His research focuses on sonography aided computer assisted surgery.

Jessica Hutzl Dipl.-Ing., received her Diploma at the Karlsruhe Institute of Technology (KIT). She is a Ph.D. candidate working in the TCRC "Cognition-Guided Surgery" and is a member of the Cognitive Medical Technologies group in the IAR-IPR, KIT. Her research focuses on port planning for robot-assisted minimally invasive surgery with redundant robots.

Jörg Raczkowsky Dr.rer.nat, is an Academic Director and Lecturer at the Karlsruhe Institute of Technology. He received his Diploma in Electrical Engineering, and Doctoral Degree in Informatics from the Technical University of Karlsruhe in 1981 an 1989 respectively. From 1981 to 1989 he was a Research Assistant at the Institute for Process Control and Robotics working in the field of autonomous robotics and multi sensor systems. Since 1995 he is heading the Medical Robotics

Group at the Intelligent Process Automation and Robotics Lab at the Institute of Anthropomatics and Robotics. His current research interests are surgical robotics, augmented reality and workflow management in the OR.

Heinz Wörn Prof. Dr.-Ing., is an expert on robotics and automation with 18 years of industrial experience. In 1997 he became professor at the University of Karlsruhe, now the KIT, for "Complex Systems in Automation and Robotics" and also head of the Institute for Process Control and Robotics (IPR). In 2015, he was awarded with the honorary doctorate of Ufa State Aviation Technical University, Russia. Prof. Wörn performs research in the fields of industrial, swarm, service and medical robotics. He was leader of the Collaborative Research Centre 414 "Computer- and Sensor-assisted Surgery" and is currently in the board of directors and principal investigator for robotics in the Transregional Collaborative Research Centre (TCRC) "Cognition-Guided Surgery".