Methods

Michael Fennel*, Antonio Zea, and Uwe D. Hanebeck

Intuitive and Immersive Teleoperation of Robot Manipulators for Remote Decontamination

Intuitive und immersive Teleoperation von Robotermanipulatoren für die Dekontamination aus der Ferne

Received April 20, 2022

Abstract: Worker safety is one of the most important aspects of decontamination tasks in hazardous environments. This motivates the development of (semi-) autonomous robotic systems that can be teleoperated from a safe distance using simple commands such as 'move the manipulator over there and grab'. In this paper, we introduce a new control station concept called Digital Twin Control System aimed at robots with manipulator arms. It consists of three components: (1) A unified communication interface that abstracts the remote robot's functionalities into easy-to-use interaction modes, (2) an immersive visualization and assistant system to operate the interface, and (3) a haptic rendering system that can simulate arbitrary robot arms. We demonstrate how the proposed system can be used by untrained operators to pick up contaminated objects remotely in a test scenario.

Keywords: Teleoperation, Haptics, VR, AR

Zusammenfassung: Bei Dekontaminationsaufgaben ist der Schutz der Arbeiter in gefährlichen Umgebungen von zentraler Bedeutung. Dies motiviert die Entwicklung von (semi-)autonomen Robotersystemen, die sich aus sicherer Entfernung mit einfachen Befehlen wie „bewege den Manipulator nach dort und greife“ teleoperieren lassen. Diese Arbeit stellt ein neues Leitstandkonzept für Roboter mit Manipulatorarmen im Sinne eines „Digital Twin Control System“ vor. Es besteht aus drei Komponenten: (1) eine einheitliche Kommunikationsschnittstelle, die die gesamte Funktionalität des Roboters mit einfach zu bedienenden Interaktionsmoden abstrahiert, (2) ein immersives Assistenz- und Visualisierungssystem, um die Schnittstelle zu bedienen, und (3) ein haptisches Renderingsystem, das beliebige Roboterarme simulieren kann. Ein Testszenario zeigt, wie ein Operator kontaminierte Objekte ohne Vorabtraining aus der Ferne vom Boden aufheben kann.

Schlagwörter: Teleoperation, Haptik, VR, AR

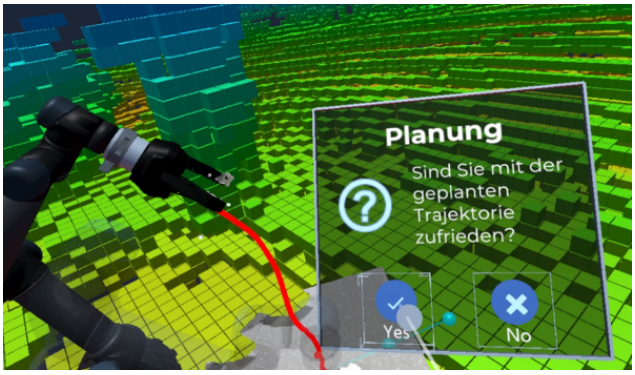

Fig. 1: Example of a remote robot being teleoperated in VR, with an assistant asking whether the trajectory (red) should be executed: *"Planning: Are you satisfied with the planned trajectory?"*.

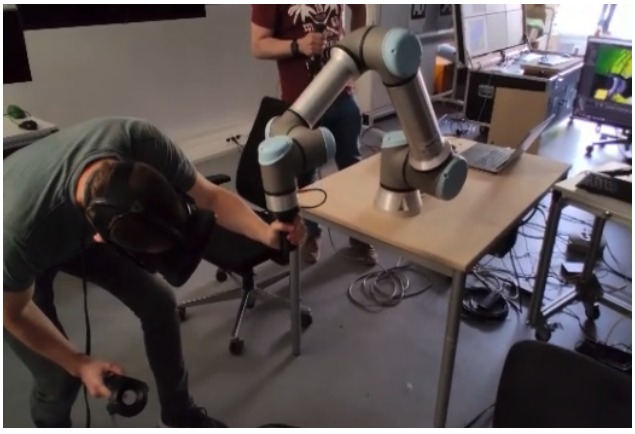

Fig. 2: Physical setup of the control station. A teleoperator with a VR headset operates a haptic robot (UR16e) that is programmed with haptic rendering to behave like a remote UR5.

***Corresponding author: Michael Fennel,** Intelligent Sensor-Actuator-Systems Laboratory, Karlsruhe Institute of Technology, Germany, e-mail: michael.fennel@kit.edu
Antonio Zea, Uwe D. Hanebeck, Intelligent Sensor-Actuator-Systems Laboratory, Karlsruhe Institute of Technology, Germany, e-mail: antonio.zea@kit.edu, uwe.hanebeck@kit.edu

1 Introduction

Contamination with hazardous substances can take several forms. A pollutant can leak into the soil and groundwater due to improper disposal in a landfill, a part or tool can spread toxic substances after operation in a chemical factory, or the walls in a decommissioned nuclear plant retain radioactive material. These events can have substantial negative effects on the environment and threaten the health of people in the surrounding regions. This shows the importance of decontamination tasks, and highlights the challenge of keeping workers safe during these (usually lengthy and physically demanding) operations. As an alternative to a regime of safety measures and protection suits, where the possibility of exposure is still present, new approaches have been introduced in recent years. These focus on keeping humans away from danger by deploying robotic systems instead. Work in this context includes tracked vehicles [1], single- and two-arm manipulators [2, 3], and climbing robots [4, 5]. However, most of these systems have no autonomy and require low-level teleoperation, which can often be unreliable, tedious, error-prone, and requires extensive training.

The ROBDEKON project (Robotic Systems for Decontamination in Hazardous Environments) [6] focuses on research into robotic systems for decontamination with capabilities ranging between shared and full autonomy. To achieve this goal, the consortium employs a variety of robots including transport vehicles, autonomous excavators [7], humanoid robots [8], four-wheeled mobile platforms, and climbing robots, depending on the scenario. In practice, this presents a dilemma: The robots are not (yet) smart enough to operate autonomously all the time, but as each of these robots has complex hardware and individual functionalities, it is not feasible to expect a teleoperator to know how to teleoperate all of them directly. Instead, in ROBDEKON, we aim to simplify the interactions between the control station and the robot by introducing an abstraction layer based on the Robot Operating System (ROS), called the ROBDEKON interface. The idea is to classify robots based on their functionality, such as whether they can move, or grab an object, or dig a hole in the ground, and so on. Thus, instead of setting individual torques or joint positions, a control station can query which higher-level functions a robot can execute, and then send these more simple and more abstract commands instead, if desired. This permits decoupling of the control station from the robot hardware and, in turn, facilitates the development of more abstract teleoperation modalities such as 'dig over there' or 'grab that object'. This also facilitates the introduction of a 'generic' ROBDEKON control station usable by any teleoperator for any decontamination scenario with any robot. Here, an operator will rely most of the time on higher-level functions that lean on the robot's own autonomous functions, and only switch to low-level actions if this is strictly necessary.

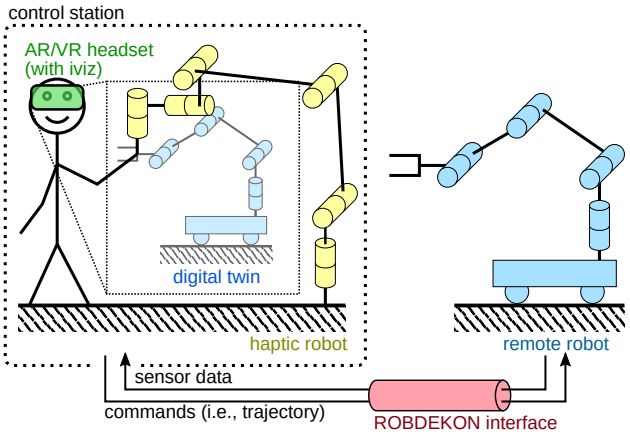

Fig. 3: Sketch of the Digital Twin Control System. The digital twin is a visual and haptic duplicate of the remote robot that is 'felt' through the haptic robot and visualized in VR or AR.

This concept of a robot-independent control station also allows for new teleoperation concepts based on technologies such as Augmented Reality (AR) and Virtual Reality (VR). This is particularly useful given that the robot needs to operate in 3D, especially when handling manipulator trajectories, which can be difficult to visualize on flat desktop monitors. However, compared to working with a replica of the manipulator, AR and VR cannot provide the haptic 'feeling' of operating a real kinematic chain, such as inertia, impulses, or force feedback, which portray useful information about joint limits, self-collisions, or whether a grabbing operation succeeded. This can be addressed by using a generic manipulator with six Cartesian degrees of freedom, such as an affordable Universal Robots UR16e, which can be programmed with a haptic rendering algorithm to mimic the mentioned properties for arbitrary robots. As an example, the user sees a simulation (a 'digital twin') of a UR5 in VR in Fig. 1 while holding a UR16e (the 'haptic robot') programmed to behave like a (scaled) UR5 in the real world as seen in Fig. 2. The advantages of this approach are that the digital twin may be operated offline without live communication with the remote robot and that only the robot's URDF specification is required, which is commonly provided as part of the drivers. Note that we are only interested in imitating the feeling of the end effector at the user's hand,

and the remaining links may differ arbitrarily. We denote this modality of a control station, which uses AR/VR and haptic rendering to teleoperate arbitrary robots based on a unified communication interface, as a *Digital Twin Control System*. The concept is sketched in Fig. 3.

In this work, we will present an example implementation of this idea. The target scenario is as follows: A four-wheeled rover with a manipulator arm is tasked with picking small contaminated items from the ground in a remote location. The teleoperator has no line of sight to the scene, and therefore depends on the rover's sensor data (cameras and LIDARs) to navigate the environment. The tasks of the teleoperator are as follows: (1) move the rover next to a contaminated object, (2) prepare a trajectory for the arm to grab the object, (3) send it to the robot for execution, and (4) validate that the operation succeeded. In the case of failure, repeat (2) and (3) until done.

This paper is structured as follows. In Section 2, we first introduce the methodology, which describes the ROBDEKON interface, the visualization and interaction systems, and the haptic rendering algorithm. Then, in Section 3, we show how all these concepts were integrated into the Digital Twin Control System. In Section 4, we describe how the system works in a practical scenario. Finally, we present to conclusion in Section 5.

2 Methodology

This section provides an overview of the 'building blocks' of the proposed teleoperation system: The ROBDEKON interface for communication, the software to visualize the robot and the remote environment, and a brief description of how the haptic rendering works.

2.1 ROBDEKON Interface

The heterogeneity of robot systems, control stations, and safety requirements in the decontamination scenarios is a positive feature of the project and a challenge that needs to be overcome. As illustrated in Fig. 4, the robots involved in ROBDEKON range from transport vehicles (e.g., tractors) over construction machines (e.g., excavators) to robots with complex manipulators (e.g., humanoids and 6 DOF manipulators). This complexity also extends to the software stack, as the robots originate from different labs, which results in a variety of robot skills that implement different levels of autonomy. On the operator side, several control station concepts exist, ranging from classical

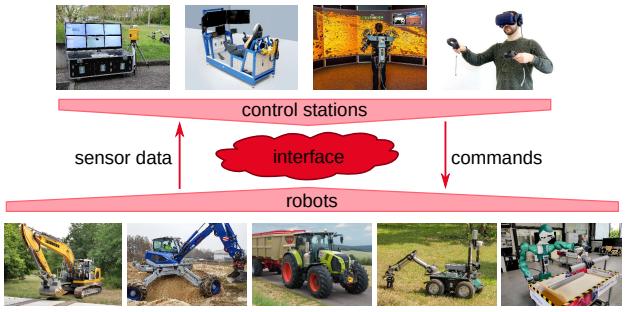

Fig. 4: Examples of control stations and robots that appear in the context of ROBDEKON. The goal is to communicate sensor data, commands, and other information over a unified interface.

replicas of the driver's cabin to modern human machine interfaces using AR and VR techniques. In this context, a critical consideration during remote operation is to ensure that functional safety aspects (e.g., connection latency, emergency stop) are taken into account and that the communication is extensible regarding new features and robots. Addressing these issues requires a unified interface that allows for reliable control and monitoring of a robot under these requirements. Unfortunately, state-of-the-art solutions were found insufficient. In [9], a communication library independent of the utilized robotic middleware is proposed. However, the problem of unifying the access to similar functions across different robots is not solved. The same holds for the solution presented in [10], where ROS is utilized as a communication layer and the robot abstraction is part of the robot-specific graphical user interface. A larger degree of unification is provided by the ROS-based modularity concept from [11]. However, this concept requires physical modules to be specified beforehand, which creates unnecessary dependencies for the control station software.

To meet the stated requirements, a new common robot communication interface was developed in ROBDEKON, whose main features are outlined in the following. Since all robots are still operated in research and development contexts, ROS was chosen as the base middleware due to its simplicity and widespread adoption. On top of ROS, a standardized partitioning scheme that consists of *functional groups*, *autonomy levels*, and *modes* was introduced in order to ensure interoperability among the different systems. These abstractions can be accessed through standard ROS topics, services, and actions.

Functional Groups: The division into functional groups is based on the fact that most atomic actions a robot performs only require to move specific parts and not the whole robot at once. Thus, we define a functional group as a physical part of the robot that enables a set of

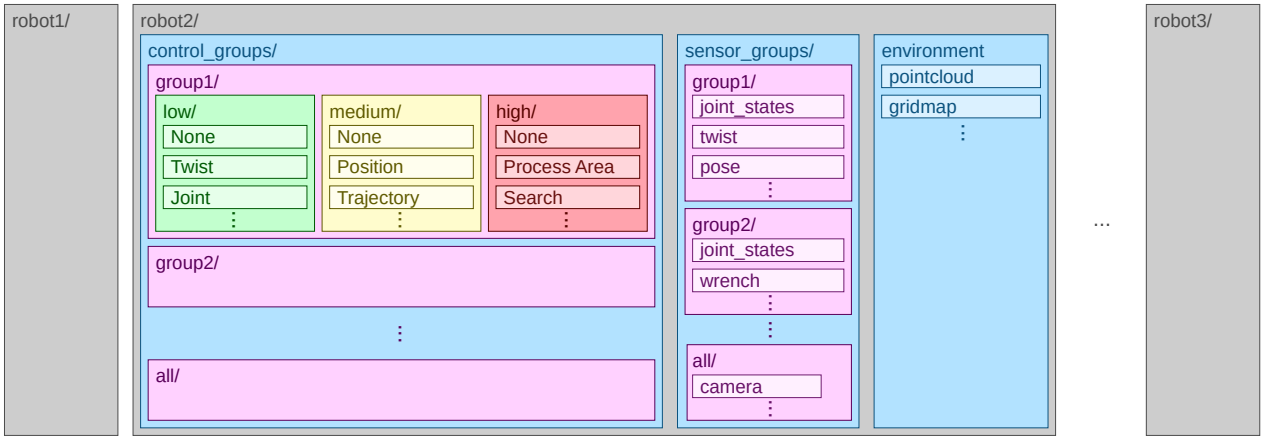

Fig. 5: Representation of the proposed partitioning concept with functional groups, autonomy levels, and modes in ROS namespaces.

interactions with the environment. In the case of an excavator, possible functional groups are 'platform' and 'arm'. For humanoids, a split into left and right arm, respectively grippers, and platform is possible. It is up to the robot developer to create a meaningful set of functional groups, and if a given robot feature requires actions from two or more groups at the same time, these groups can be combined into a robot-specific *functional meta group*.

Autonomy Levels: For each functional group, the different actions a group can perform can be categorized based on their level of autonomy:

- *stop:* The functional group is halted and inactive.
- *low:* The robot is controlled manually, for example by setting torques or joint positions on the low-level controllers.
- *medium:* Basic skills, which require more involved calculations and some autonomy on the robot, are executed. No complex perception, planning, or reasoning is involved.
- *high:* Complex tasks expressed in an abstract task description are executed fully autonomously.

Modes: Finally, the different autonomy levels of each functional group are split into different modes, where each mode represents a specific skill that can be implemented by the robot. For example, the *twist* mode of level *low* sets a velocity, the *position* mode of level *medium* navigates directly to the target coordinates, while *dig_hole* of level *high* digs a hole autonomously. Note that, while the interface defines the list of all possible modes, it is up to the developer to define what they mean exactly in the context of the control group, and which ones the robot supports at all.

For the implementation of this concept, we employ ROS namespaces as depicted in Fig. 5. Each robot has its own sub-namespace, which holds all robot-specific interfaces, in the root namespace. All control-specific topics, services, and actions are collected in the *control_groups* namespace, which in turn contains all functional groups and meta groups as sub-namespaces. Within these, a sub-namespace for each autonomy level is instantiated. Switching between different autonomy levels and modes is performed using services in the respective namespaces. Related sensor and state data is also provided to the control station in a structured way. For this purpose, the corresponding topics are published in sub-namespaces of the *sensor_groups* namespace according to the functional group to which they belong. In addition, environment data that cannot be assigned to a specific functional group, such as generic maps, can be made available in the *environment* namespace.

A major requirement that arises from the desired interoperability between various robots and control stations is that the control station software is robot-agnostic. On the one hand, this is achieved through the usage of URDF files with robot-specific name prefixes to transfer information about geometry, visuals, and topology. These specifications are published as parameters in the ROS master. On the other hand, discovery mechanisms are implemented in the form of ROS services. This allows a control station to enter a possibly unknown scenario and find out which robots are active, which features (functional groups, autonomy levels, and modes) they support, and other important information. Regarding safety, the interface is equipped with an emergency stop mechanism and a watchdog monitoring setpoints, which potentially lead to an infinite movement. Additionally, the aforementioned switching services ensure that the robot is always in a definite and known state and that potentially interfering commands are not possible. On a higher level, the whole

robot can be marked as 'in use' by a control station for mutual exclusive access. Moreover, a partially machine-interpretable logging facility is included in the interface as an alternative to the rather unregulated rosconsole.

Based on these concepts, a specification was created and a reference library for the robot-side was implemented in C++. In this way, the effort of integrating new robots with the ROBDEKON interface is limited to creating a URDF file and writing and registering custom callbacks.

2.2 Visualization and XR Interactions

Dealing with dynamic ROS data requires a flexible visualization platform that can handle large amounts of data even in devices with fewer CPU resources. There are multiple applications that deal with this challenge, ranging from the traditional desktop-based *Rviz* to the more web-oriented *Foxglove*. However, for this project, we aim for a teleoperation solution that can be deployed anywhere at any time — for example on a tablet when exploring a landfill on site, on a smartphone next to an excavator in a contaminated environment, or in a head-mounted display (HMD) while a robot maps a remote decommissioned power plant. This can be achieved easily by using new technologies in the field of extended reality (XR), an umbrella term that covers both VR and AR. For example, when teleoperating a rover handling hazardous waste, a real-time reconstruction of the distribution of contaminants in the remote location can be displayed in 3D around the user, providing an optimal overview that can be intuitively navigated by walking around in it.

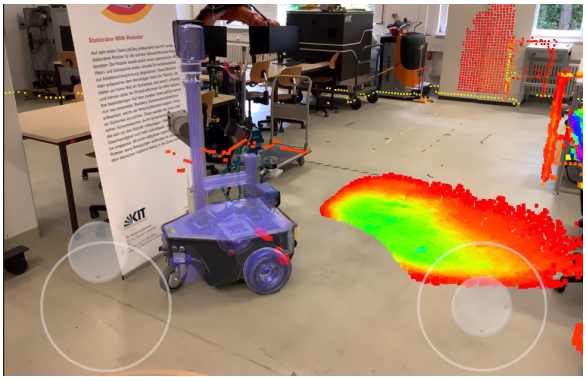

Fig. 6: Example teleoperation of a Gammabot (blue) from KIT-IPR, a ROBDEKON partner, using an iPad in AR. The captured point cloud is overlaid in real time.

Unfortunately, state-of-the-art visualization apps focus almost exclusively on desktop operating systems and browsers. To address this, we introduced iviz [12], a new ROS visualization app designed from scratch to support native AR and VR in mobile devices. It is based on the Unity engine and written in C#, which allows us to target a wide variety of desktop and mobile platforms, with focus on AR frameworks such as Google's ARCore and Apple's ARKit. The full software packet is provided as an open source project under the MIT license.

Fig. 7: Live teleoperation of an excavator from Fraunhofer IOSB, a ROBDEKON partner, with an iPad. The excavator's joints can be controlled individually by spinning the markers in blue.

While visualization is a cornerstone of remote teleoperation, user interaction is of equal importance. For traditional 2D devices such as tablets and monitors, ROS provides a package for interaction based on 'interactive' markers, i.e., customizable geometric figures such as cubes, cylinders, lines, or meshes that can be configured to respond to interactions such as mouse clicks, taps, and drags. This, in turn, facilitates the programming of simple user interfaces. However, these modalities were designed for a third-person view on a 2D screen and are not suitable for the range of 3D manipulations required by AR and VR teleoperation. This motivated the development of widgets that are specific to the modes and autonomy levels that appear in the ROBDEKON interface. For example, a simple way to handle twist messages (velocities) is shown in Fig. 6, where a robot is moved by dragging a pair of virtual joysticks. In case the target area of a digging operation needs to be declared, this can be achieved by pointing to a position on the floor with the hand, followed by adjustments through dragging markers over the ground. Similarly, a robot can be commanded to move joints with the help of markers that are dragged to the target position by the user as seen in Fig. 7. Another important interaction mechanism are dialog boxes, which allows the system to ask questions to the user and inform them of what is going on. Fig. 1 shows an example of a dialog

box asking the user whether the programmed trajectory is correct. Other dialogs include text notifications, alerts with multiple options, and menus.

2.3 Haptic Rendering of Manipulators

Thanks to the use of AR/VR displays as the core of the control station and the resulting depth perception, it is easy to visualize data sent from the robot in a native and hence intuitive way for the operator. Unfortunately, this cannot be transferred directly to the input of the robot commands, especially for manipulator trajectories. Graphical user interfaces have the disadvantage that it is difficult for the user to provide rotational and translational movements intuitively, especially in 3D space with six degrees of freedom. This can be improved, for example, by using hand tracking data to set trajectory poses. However, human hand poses have very few constraints, often making the trajectories incompatible with the kinematic properties of the remote robot, for example the shovel of an excavator. To mitigate this issue, some mapping from full 3D poses to constrained poses could be performed, but this may result in discontinuous correspondences, which in turn can produce dangerous behavior. In practice, kinematic singularities will further complicate the situation. For these reasons, the haptic rendering algorithm in [13] was developed. As a result, a haptic twin of an arbitrary serial manipulator, consisting of prismatic and revolute joints, with known forward kinematics can be created.

Principle: The basic idea of this approach, which is sketched as part of Fig. 3, is to render a down-scaled haptic twin (light-blue mobile robot) regarding the forward kinematics

$$\underline{x} = \underline{f}\left(\underline{q}\right) \tag{1}$$

of the end effector pose $\underline{x}$ and the joint angle limits $\underline{q}_{\min} \leq \underline{q} \leq \underline{q}_{\max}$ of the remote manipulator. In this way, the digital twin is composed of a visual twin from AR/VR and a haptic twin. The physical realization of the latter is achieved using a kinesthetic haptic interface (yellow robot in Fig. 3) that is called *haptic robot* throughout this paper. The proposed rendering algorithm is based on the well-known dynamics equation

$$\underline{\tau} = \underline{\tau}_{\mathrm{dri}} + \underline{\tau}_{\mathrm{con}} - \underline{\tau}_{\mathrm{dis}} = \mathbf{H}\left(\underline{q}\right)\ddot{\underline{q}} + \underline{c}\left(\underline{q}, \dot{\underline{q}}\right) , \tag{2}$$

where $\mathbf{H}$ represents the joint space inertia matrix and $\underline{c}$ the joint torques occurring due to centrifugal and Coriolis forces. The total effective torque $\underline{\tau}$ is the sum of the driving torque $\underline{\tau}_{\mathrm{dri}}$, the constraint torque $\underline{\tau}_{\mathrm{con}}$, and the dissipative torque $\underline{\tau}_{\mathrm{dis}}$. Gravity-induced torques are omitted in the model as they would put a permanent load onto the operator. To make the haptic twin compliant towards the operator's intention, the user-exerted forces and torques at the end effector of the haptic robot are transformed into driving torques of the haptic twin using its end effector Jacobian matrix. As long as no joint limits are active, the resulting motion can then be calculated by solving and integrating (2).

To obtain the unknown variables $\mathbf{H}$ and $\underline{c}$, the composite-rigid-body algorithm and the recursive Newton-Euler algorithm are utilized, respectively. The inertia distribution of the real robot could be used for this purpose, but the necessary information is often not available in practice for heavy machinery or it would yield uncomfortable high inertia that cannot be displayed safely under human operation. For this reason, an artificial inertia distribution is used that concentrates most of the inertia at the end effector of the rendered robot. In addition, all other links are assigned with a small fraction of the total inertia to avoid ill-conditioned and hence non-invertible matrices.

In practice, passive mechanisms always approach a static state due to energy dissipation caused by friction. To include this effect in the haptic rendering and to increase safety, a viscous joint friction and an isotropically acting Cartesian viscous friction are included in the model through $\underline{\tau}_{\mathrm{dis}}$. During operation, the first kind of friction mainly ensures good behavior close to the kinematic singularities of the rendered robot. The second kind of friction enables that the damping felt by the operator, which has Cartesian perception, that does not depend too much on the link lengths and the rendered configuration.

Joint Limits: So far, the behavior of a serial manipulator with infinite joint limits can be emulated. In practice however, this could still lead to motions which cannot be tracked by the remote robot. To incorporate this into the rendering algorithm, the joint limits $\underline{q}_{\min}$ and $\underline{q}_{\max}$ are interpreted as potential contacts between rigid bodies following the theory by Featherstone [14]. According to this theory, the movement of a joint at a limit is described with the help of a suitable constraint torque in the corresponding entry of $\underline{\tau}_{\mathrm{con}}$. This constraint torque is calculated in [13] using a system of linear equalities and inequalities, which describe the motion at the contact. If the contact (i.e., the limit) remains active, the equations require that the constraint torque magnitude is non-zero to maintain the contact, whereas zero constraint torque and a joint acceleration away from the limit are required in the case of a breaking contact.

For physically credible behavior, the preservation of the mechanical impulse must be enforced wherever possible. To achieve this, a generalized and initially unknown

impulse in joint space coordinates is introduced for each joint, which is at a limit. This impulse causes a specific change of the joint velocities when applied. The impulse must ensure that the contact either remains intact (i.e., new velocity is zero), or that the contact will break without an additional impulse (i.e., new velocity causes a movement away from the contact). Similar to the constraint torque above, this can be mathematically formulated as a set of equalities and inequalities as shown in [13].

The obtained generalized momentum is then used to correct the joint velocities. After that, the constraint torque is calculated before the actual integration step is performed in each rendering iteration. By combining the resulting joint configuration $\underline{q}$ with the forward kinematics (1), the resulting end effector pose can be obtained. This information is then fed into the haptic robot as the setpoint of a Cartesian pose controller. Fig. 8 summarizes the whole haptic rendering pipeline. To be robot-agnostic, the implementation allows passing the kinematic model of the remote robot as a URDF file.

3 The Digital Twin Control System

This chapter describes how the building blocks from Section 2 fit together to form the Digital Twin Control System.

3.1 Control Station

The control station can be seen in Fig. 2. It consists of a space of at least $3\,\mathrm{m} \times 3\,\mathrm{m}$ with a table in the center. The haptic robot (in the real world) and the digital twin (in the virtual world) are located on top of the table. The system can be used either as an AR or VR platform. For AR, a Microsoft Hololens 2 is provided, and the user interacts with the environment with hand tracking. For VR, the hardware setup is a HTC Vive tethered to a PC, and for interaction, the VR controllers are used. In both modalities, an AR spectator mode with iPad tablets is available for additional users interested in the scene. While a chair is provided for the user's convenience, the entire space is navigable and the user can stand up and move around if desired.

In the following section, we will discuss some aspects of the implementation:

1. the initialization mechanism, which is in charge of registering the physical world to the virtual world,
2. the visualization system, which reconstructs the remote environment from sensor data,
3. and the XR user interface (UI) system, consisting of a set of dialog boxes, menu boxes, and widgets to control the ROBDEKON interface.

3.1.1 Initialization

Before the digital twin can be teleoperated, it is first necessary to calibrate the coordinate system of the virtual world so that it matches the real one. This process needs to be kept simple, as the origin is on top of the table and it is quite common for it to move slightly between sessions. For the VR system, we need to take into account that the user cannot see the real environment, and thus, the table itself also needs to be reproduced virtually. This is achieved with a simple assistant that asks the user to mark the four table corners with the VR controllers. In AR, the table becomes less prominent as the user can see their real surroundings. A simple solution is to use a QR marker with known pose and size on the table, which can be detected by the AR device. The world pose can then be further adjusted by the user with the assistant software.

3.1.2 Visualizing the Environment

Once the coordinate systems of the real and virtual worlds match, the haptic robot and the digital twin are loaded from their URDF specifications. The digital twin is overlaid on top of the haptic robot, which is displayed semi-transparently to avoid unintended collisions with the operator. Their origins are fixed relative to each other. Thus, if the digital twin is translated, it visually remains in place in front of the user, while its surroundings move in the opposite direction. This ensures that the user is always sitting right next to both robots.

However, in order for the user to recognize this motion, the robot's environment needs to be visible, which is not a simple task in a live teleoperation where the only knowledge of the remote robot's surroundings stems from sparse sensor data. For static elements such as walls or terrains, two types of messages are used: voxel maps, which describe the environment as 3D cubes (for example Fig. 1, seen in green and cyan), and 2D occupancy grids, which can be extruded vertically to visualize obstacles. For highly dynamic elements, live point clouds (as in Fig. 6) can be used together with traditional ROS markers such as lines, cubes, and spheres.

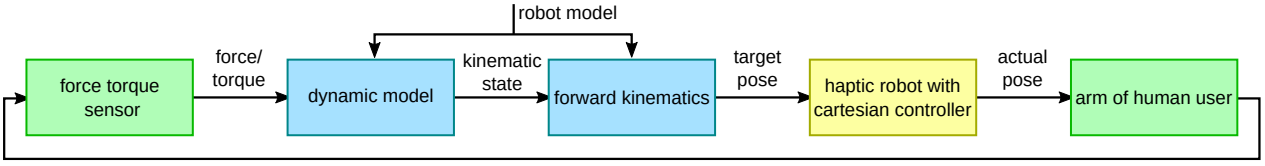

Fig. 8: Overview of the haptic rendering pipeline. The colors yellow, green, and blue indicate that the part belongs to the haptic robot, the controlled robot, and the human operator, respectively.

3.1.3 XR Assistant for the ROBDEKON Interface

The final component of the control station is the user assistant app, which glues the ROBDEKON interface and the iviz interaction mechanisms together. It determines which visual components need to be shown for which robots depending on the scenario, and translates the user responses into commands for the interface. Conceptually, the app is implemented as a script that can be quickly updated, adjusted, and extended depending on the needs of each deployment. While it can be integrated into the visualizer, it is usually run as a separate app on a desktop PC. This is useful when running iviz on a tablet or a standalone HMD, as packaging and uploading new versions of the app can be extremely time consuming. Furthermore, keeping the assistant external means that the script can be implemented in a programming language that requires less expertise to work with, such as Python or Javascript.

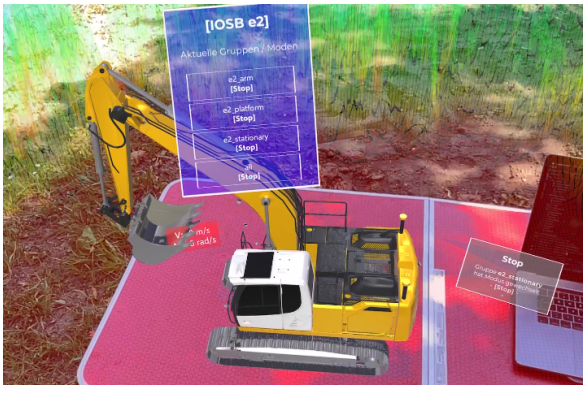

Fig. 9: Example of controlling the ROBDEKON interface from an iPad in AR. The dialog box in blue shows a clickable menu with functional groups and their current autonomy levels. The caption below the title says: *"Current groups/modes"*.

A simple implementation can be seen in Fig. 9. Here, a menu displays all available functional groups, obtained using the discovery mechanism. When an entry is clicked, a new menu pops up with the available autonomy levels and modes. If a mode is selected in response to this, the corresponding widgets are automatically created. The provided widgets include:

- A disc that acts like a spring and can be dragged to set the robot velocity for the *low/twist* mode,
- a draggable point whose position determines the trajectory to be sent for the *medium/trajectory* mode
- a set of discs that can be rotated (see Fig. 7) to adjust the joint positions for the *low/joints* mode,
- and a resizable and draggable rectangle to set the area to dig for the *high/dig_hole* mode.

A floating button is positioned above the robot (shown in blue above the excavator in Fig. 7) that allows the user to return to the selection menu. More complex interaction sequences can also be created, for example to guide a user through detailed scripts and decision trees. This is useful for the more complex workflows of Section 3.2, where a user is asked to visually validate the result of the robot's actions in order to determine the next steps.

3.2 Operating the Digital Twin Control System

The XR interaction modalities provided by the control station are complemented with the haptic rendering method from Section 2.3 as sketched in Fig. 3, resulting in the new modality that is called Digital Twin Control System. In the following, we will present an application of this system in a simple example scenario, where a remote manipulator needs to be moved to a given position.

The procedure, which is also depicted in Fig. 10, is as follows. First, a full snapshot of the robot's environment is fetched. This includes maps, location, and kinematic state, but most importantly a sufficiently accurate 3D-representation of the environment that we want to manipulate, e.g., a dense, colored point cloud. In our scenario, this information is transmitted via the ROBDEKON interface and rendered in iviz. After that, the haptic robot is positioned so that its end effector pose matches the end effector of the remote robot as perceived by the operator

in their AR/VR environment. A digital twin is created in the same configuration, and the operator will then be instructed to grasp the end effector and guide it along a desired trajectory in order to fulfill the given task. This process can be paused by the operator at any time or restarted if the entered trajectory is unsatisfactory. The digital twin simulation runs offline, and thus, except for sensor data, no communication between the robot and the control station is necessary until the operator chooses to execute the selected trajectory. In the interface, the trajectory is sent to the robot for execution using the *trajectory* mode within the autonomy level *medium* of the manipulator's functional group.

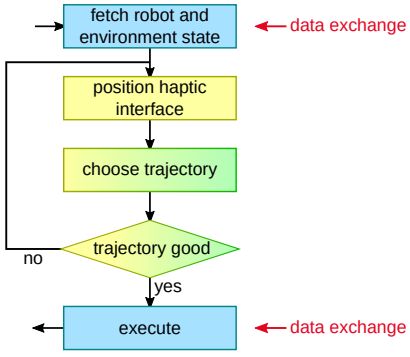

Fig. 10: Flow chart of the Digital Twin Control example. Blue boxes are implemented by the remote robot, yellow boxes by the control station, and green denotes inputs from the operator.

The advantage of this procedure is that it provides an immersive 3D experience for the operator with fine and accurate control of the robot's movements in six degrees of freedom. The proposed method is completely robot-agnostic thanks to leveraging the ROBDEKON interface, with the only requirement that the robot manipulator supports the *trajectory*-mode. Since data from the remote robot is only needed at the beginning (for the setup) and at the end of the procedure (for validation), no permanent or high-bandwidth connection is required. Furthermore, stability issues as they are typical for classical teleoperation setups [15] are completely avoided, because there is no permanent feedback loop involved between the remote robot and the control station. Furthermore, the user input is least abstracted, decreasing the risk of unintentional or unsafe user commands.

Despite all the advantages, some negative implications for safety must be considered as well. First, the operator is not able to react quickly to changes of the environment. While this is usually not a problem in static and access-restricted decontamination scenarios, it could become a problem in disaster scenarios. In this case, another layer of functional safety has to be provided locally on the robot or a more direct mode of live teleoperation must be used. Second, the user could plan a trajectory that leads to damage of the robot or its environment. However, this holds for any kind of teleoperation. Lastly, the interaction with the haptic robot at the operator side introduces a new safety risk due to possible collisions and pinch points, especially when VR content covers the haptic robot. Although this risk is acceptable for the current stage of research, further suitable safety features must be added before for a broad deployment.

Fig. 11: The modified Husky robot from FZI used during the test deployment is shown grabbing a golf ball representing a contaminant. The box on top is considered as the safe containment.

4 Evaluation

In order to validate the proposed concepts, a prototype of the Digital Twin Control was deployed in a real-life test scenario in cooperation with the FZI Forschungszentrum Informatik, a ROBDEKON partner. For this purpose, a modified Husky robot from Clearpath Robotics with an attached Universal Robots UR5 manipulator, as depicted in Fig. 11, was utilized. In addition to the basic robot telemetry, which includes the kinematic state, the Husky transmitted a voxel map of the environment, camera images, and most importantly, a colored point cloud from an Intel RealSense depth camera rigidly attached to the three-finger gripper. The control station, depicted in Fig. 2, consisted of a tethered HTC Vive running an instance of iviz and a table-mounted Universal Robot UR16e. The UR16e acted as the haptic robot and, for this reason, was equipped with a handle and an integrated dead-man switch at its end effector. For the experiments, the remote robot and the control station were not in sight of each other. The communication with the robot was realized over a standard Wi-Fi network using the ROBDEKON interface from Section 2.1.

4.1 Retrieval of Contaminants

In the scenario, contaminated items spread over an unknown area needed to be retrieved by picking them up and storing them in a safe compartment. For this purpose, the disc markers shown in Fig. 7 were used to move the platform close to the object to be collected using the *platform/low/twist*-mode. After that, the pose of the depth camera, which is coupled to the remote robot's end effector, was adjusted by the operator with the *arm/low/twist*-mode to maximize the visual quality of the object of interest if necessary. By this point, the control station already had a full voxel map of the robot's static environment in all conducted experiments. Then, the Digital Twin Control subroutine presented in Section 3.2 was launched with the goal of selecting a trajectory for grasping the desired object. In the studied examples, a trajectory to grasp a metal can was selected by the operator as depicted in Fig. 12 and Fig. 13. The resulting trajectory was then sent to the robot for execution using mode *arm/medium/trajectory*. The grasp action was triggered with the appropriate command in mode *arm/medium/gripper*. For the example, Fig. 14 shows the resulting movement of the remote robot. To check whether the grasp was successful, the end effector was lifted using mode *arm/low/twist* afterwards, with the operator verifying visually that the can did not fall. Finally, the contaminated object was deposited into the rover's container with mode *arm/high/unload*.

4.2 User Study

In order to evaluate the performance of the proposed Digital Twin Control System, a user study with 8 male subjects aged between 23 and 32 years was conducted. In a pre-experimental survey, all participants stated that they had little or no experience with VR technologies. 87.5 % claimed to have little or no familiarity with teleoperation methods, while the remainder reported average familiarity. Furthermore, all participants had experience playing video games, ranging from little (25.0 %) over medium (62.5 %) to very high (12.5 %) experience.

At the beginning of the experiments, all participants were introduced verbally to the task and the technologies being used. Furthermore, the Husky robot and the object to be grasped were shown to the participants once in real life. After being equipped with the VR system, each participant performed a short test trial, where the objective of selecting trajectories for grasping objects and all relevant parts of the system were explained briefly. This way, the users could acclimatize and get familiar with the system. However, this trajectory was not sent to the Husky robot for execution. With this training level, the participants were asked to retrieve an object placed randomly in front of the Husky robot in three subsequent trials.

Of the resulting total of 24 trials, 22 (91.6 %) led to a successful grasp. On average, the subjects needed 11.2 s to choose a suitable trajectory with a standard deviation of 3.5 s. The minimal and maximal time were 5.4 s and 21.4 s, respectively. We also asked the users to rate their experience regarding mental demand, intuitiveness, and motion sickness. None of them reported signs of motion sickness. 75.0 % of the users stated the system was very intuitive, and the remainder stated that it was moderately intuitive. The mental demand was rated small by 62.5 %, medium by 25.0 %, and high by 12.5 % of the users.

5 Conclusions

In this paper, we presented a novel teleoperation concept for decontamination scenarios within the ROBDEKON project. This approach, called the Digital Twin Control System, consists of three components. First, a unified communication interface was developed that can fully teleoperate a variety of robots using abstracted functionalities as building blocks. Second, the cross-platform visualization tool iviz was extended with an assistant system that can operate the entire ROBDEKON interface from within an AR or VR environment. To improve immersion even further, we deployed a haptic rendering algorithm to extend the control station with meaningful haptic feedback without requiring a low-latency connection to the robot.

The evaluation in Section 4 proved that the designed components work well together. The VR environment, together with the assistant application providing informative dialog boxes at every step, was well received by the testers as shown by the low mental demand. Moreover, the haptic rendering system facilitates intuitive and accurate control over the remote robot, as evidenced by the high success rate and low completion times, even with users that had little to no teleoperation experience. Still, there is room for improvement in future work. An important extension of the ROBDEKON interface is planned in order to transmit higher level information from the robot to the VR operator, for example mesh reconstructions and bounding volumes. This information is more understandable than raw point clouds or voxel maps, and more robust against calibration issues and low measurement quality. This can also enhance the haptic feedback by providing cues of how the gripper is touching an object.

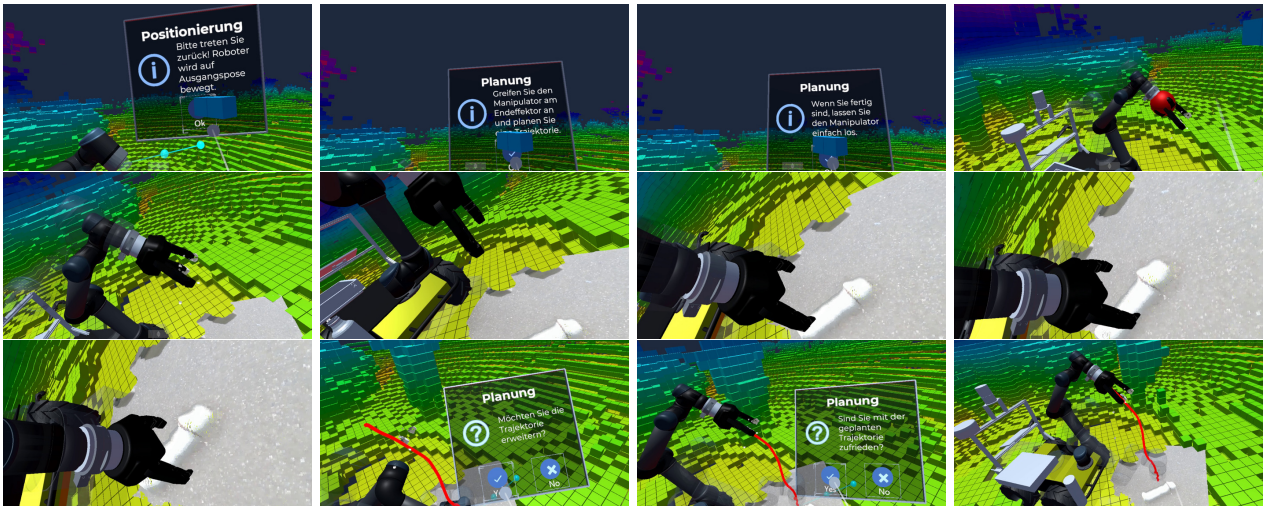

Fig. 12: View of the user in VR during the trajectory selection process. The frames are $2.5\,\mathrm{s}$ apart. At the beginning, the dialog boxes inform the user about what to expect. At the end, the user confirms the trajectory. The titles and captions in order are: *"Positioning: Please step back! The robot will be moved to initial position."*, *"Planning: Grasp the manipulator at the end effector and plan a trajectory."*, *"Planning: If you are done, simply release the manipulator."*, *"Planning: Do you want to extent the trajectory?"*, *"Planning: Are you satisfied with the planned trajectory?"*.

Fig. 13: Interaction of the user with the haptic robot during the trajectory selection process. The frames are approximately synchronized with those shown in Fig. 12.

Fig. 14: The resulting trajectory that corresponds to the selected trajectory from Fig. 12 and Fig. 13. Note that the playback speed was reduced for safety reasons.

Funding: This work was supported by the ROBDEKON project (grant No. 13N14675) of the German Federal Ministry of Education and Research.

References

[1] Arne Rönnau et al. "Robust 3D scan segmentation for teleoperation tasks in areas contaminated by radiation." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 2419–2424.

[2] Allahyar Montazeri and Josef Ekotuyo. "Development of dynamic model of a 7DOF hydraulically actuated teleoperated robot for decommissioning applications." In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 1209–1214.

[3] Pierre Besset and C James Taylor. "Inverse kinematics for a redundant robotic manipulator used for nuclear decommissioning." In: *2014 UKACC International Conference on Control (CONTROL)*. IEEE. 2014, pp. 56–61.

[4] Michael Mende et al. "Environment modeling and path planning for a semi-autonomous manipulator system for decontamination and release measurement." In: *2014 World Automation Congress (WAC)*. IEEE. 2014, pp. 54–59.

[5] Woosub Lee, Masahiro Hirai, and Shigeo Hirose. "Gunryu III: reconfigurable magnetic wall-climbing robot for decommissioning of nuclear reactor." In: *Advanced Robotics* 27.14 (2013), pp. 1099–1111.

[6] Janko Petereit et al. "ROBDEKON: Robotic systems for decontamination in hazardous environments." In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 249–255.

[7] Thomas Emter et al. "Algorithm toolbox for autonomous mobile robotic systems." In: *ATZoffhighway worldwide* 10.3 (2017), pp. 48–53.

[8] Tamim Asfour et al. "Armar-6: A collaborative humanoid robot for industrial environments." In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 447–454.

[9] Leon Cedric Danter et al. "Lightweight and Framework-Independent Communication Library to Support Cross-Plattform Robotic Applications and High-Latency Connections." In: *15th International Symposium on Systems, Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*. 2020.

[10] Stefan Fabian and Oskar von Stryk. "Open-Source Tools for Efficient ROS and ROS2-based 2D Human-Robot Interface Development." In: *2021 European Conference on Mobile Robots (ECMR)*. 2021, pp. 1–6.

[11] Ivana Kruijff-Korbayová et al. "German Rescue Robotics Center (DRZ): A Holistic Approach for Robotic Systems Assisting in Emergency Response." In: *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2021, pp. 138–145.

[12] Antonio Zea and Uwe D. Hanebeck. "iviz: A ROS Visualization App for Mobile Devices." In: *Software Impacts* 8 (2021).

[13] Michael Fennel et al. "Haptic Rendering of Arbitrary Serial Manipulators for Robot Programming." In: *IEEE Control Systems Letters* 6 (2021), pp. 716–721.

[14] Roy Featherstone. *Robot dynamics algorithms*. Boston, Dordrecht, Lancester: Kluwer Academic Publishers, 1987.

[15] D. A. Lawrence. "Stability and transparency in bilateral teleoperation." In: *IEEE Transactions on Robotics and Automation* 9.5 (1993), pp. 624–637.

Author Information

Michael Fennel
Intelligent Sensor-Actuator-Systems Laboratory,
Karlsruhe Institute of Technology,
Germany
michael.fennel@kit.edu

Michael Fennel is a researcher and PhD student at the Intelligent Sensor-Actuator-Systems Laboratory of the Karlsruhe Institute of Technology. His research interests include robotics, autonomous systems, haptic systems, and human machine interaction.

Antonio Zea
Intelligent Sensor-Actuator-Systems Laboratory,
Karlsruhe Institute of Technology,
Germany
antonio.zea@kit.edu

Dr.-Ing. Antonio Zea is a researcher at the Intelligent Sensor-Actuator-Systems Laboratory of the Karlsruhe Institute of Technology. His research interests include robot teleoperation with VR/AR and human machine interaction.

Uwe D. Hanebeck
Intelligent Sensor-Actuator-Systems Laboratory,
Karlsruhe Institute of Technology,
Germany
uwe.hanebeck@kit.edu

Prof. Dr.-Ing. Uwe D. Hanebeck is a chaired professor of Computer Science at the Karlsruhe Institute of Technology in Germany and director of the Intelligent Sensor-Actuator-Systems Laboratory. His research interests include information fusion, nonlinear state estimation, and stochastic modeling.