Link to the published version of the paper:
https://doi.org/10.1109/AERO63441.2025.11068600

System Architecture and Design Considerations for the Humanoid Robot Rollin' Justin in Context of the Surface Avatar Mission

Adrian S. Bauer, Anne Köpken, Nesrine Batti, Jörg Butterfaß, Tristan Ehlert, Werner Friedl, Thomas Gumpert, Florian S. Lay, Xiaozhou Luo, Ajithkumar N. Manaparampil, Luisa Mayershofer, Antonin Raffin, Florian Schmidt, Daniel Seidel
German Aerospace Center (DLR), 82234 Weßling, Germany
adrian.bauer@dlr.de, {*firstname.lastname*}@dlr.de

Emiel den Exter, Rute Luz
European Space Agency (ESA)
2201AZ Noordwijk, Netherlands
emiel.den.exter@ext.esa.int, rute.luz@ext.esa.int

Annika Schmidt
German Aerospace Center (DLR), 82234 Weßling, Germany
Technical University of Munich, 85748 Garching, Germany
an.schmidt@tum.de

Peter Schmaus
German Aerospace Center (DLR)
82234 Weßling, Germany
peter.schmaus@dlr.de

Daniel Leidner
German Aerospace Center (DLR), 82234 Weßling, Germany
University Bremen, 28359 Bremen, Germany
daniel.leidner@dlr.de

Thomas Krüger
European Space Agency (ESA)
2201AZ Noordwijk, Netherlands
thomas.krueger@esa.int

Neal Y. Lii
German Aerospace Center (DLR)
82234 Weßling, Germany
neal.lii@dlr.de

***Abstract*— With continuous advancements in robotics, both in hardware and in software, the feasibility to deploy robotic assistants as co-workers for astronauts in real mission scenarios is coming in sight. In the context of the Surface Avatar International Space Station (ISS) telerobotic technology demonstration mission, we study the requirements in terms of user interface (UI), robotic capabilities, and communication to enable efficient usage of robots as astronaut's co-workers. During the experiments, astronauts onboard the ISS command a team of heterogeneous robots at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany, to perform experimental tasks in a Mars analog environment. In order to complete the tasks successfully, the astronauts have to select between different robot command modalities, namely teleoperation and supervised autonomy. While previous publications have mostly focused on the UI, the interfaces between robots and the UI, and the overall mission concept, this work sheds light on the robotic back-end and provides a description of our reference implementation. Utilizing the humanoid robot Rollin' Justin as our prime use case, we describe the modules that enable the robotic capabilities that are offered to the astronauts as well as their implementations. As a core aspect of Surface Avatar is the ability to select from different command modes, i.e. supervised autonomy and direct teleoperation, we put special focus on the high-level modules that enable supervised autonomy, such as knowledge representation, belief state representation, and reasoning, as well as the teleoperation interfaces. The paper also describes the integration of the aforementioned modules into the overall system. In this work we share the decisions and iteration processes that lead up to our current design, the motivation behind the decisions, the limitations they imply on the system, and the lessons learned during the process. This work particular examines these modules in the context of the Surface Avatar experiment session and describes, in particular, the improvements that have been achieved in comparison to previous versions. While the system continues to evolve to support new features for upcoming experiment sessions, our description covers the state of the robot during the first ISS experiment sessions and the two following prime sessions of Surface Avatar.**

979-8-3503-5597-0/25/$31.00 ©2025 IEEE

TABLE OF CONTENTS

1. INTRODUCTION

Driven by recent developments in robotics, particularly with humanoid robots' vast potential in mobility and dexterous object handling, coupled with human curiosity and interest in space, the deployment of robotic assistants as co-workers for astronauts is on the horizon. As robots can be designed to operate in hazardous environments, accomplish tasks infeasible for humans, and operate without breaks, they are perfect companions for astronauts, supporting them in dirty, dull, and dangerous jobs. As almost any task in space can be considered dangerous, there is great potential for robots in space.

Extensive previous work including METERON SUPVIS Justin [1] and Analog-1 [2], have shown a multitude of possibilities to command a robot as co-worker and as immersive physical avatars on-site. We also ponder a future of commanding teams of robots with different command modalities and command abstractions [1]. It is with this inspiration that we commenced the Surface Avatar *International Space Station* (ISS) telerobotic technology demonstration mission [3]. Surface Avatar, led by *German Aerospace*

Center (DLR) with partner *European Space Agency* (ESA), studies the factors that enable astronauts to make efficient use of robotic teams through different styles of command, ranging from direct teleoperation to Supervised Autonomy as co-workers. In this context we put special focus on the design of an intuitive user interface (UI), understanding the requirements of communication between astronaut and robot, and the robot capabilities that enable efficient interaction.

While in the METERON experiments astronauts on the ISS commanded one robot at a time, Surface Avatar extends this approach to commanding a heterogeneous team of robots located at DLR in Oberpfaffenhofen, Germany. So far, three sessions have been carried out with a team of up to four different robots with different form factors and functionalities. The tasks that the astronauts were requested to solve, ranged from maintenance tasks such as checking on assets and interacting with them to solve malfunctions, to scientific tasks like distributing seismometers in the environment or selecting, picking, and analyzing rock samples.

To solve these tasks, the astronauts are supported by a user interface on board the ISS consisting of a graphical user interface (GUI), a joystick, and a Force Dimension sigma.7[1] haptic input device. They can issue task-level commands to the robots via the GUI, which the robots execute autonomously, or control the robots more directly, or immersively via an open-loop joystick or the sigma.7 with multi-Degree-of-Freedom (DoF) force-reflection.

While the UI for Surface Avatar has previously been described in multiple publications [4], [5] this work sheds light on the system architecture of DLR's humanoid robot Rollin' Justin [6] that enables the integration of complex robot into the Surface Avatar operations. Core aspects we must realize include the ability to enable the communication layer, as well as the interfaces for switching between and linking the different teleoperation modalities with the joystick and sigma.7 devices.
The contribution of this paper is to give an overview of the software system on Rollin' Justin that was used throughout six ISS-to-ground teleoperation experiments in the Surface Avatar Mission.

In the following we link to related work in section 2, followed by a description of Surface Avatar, including the requirements and constraints it implies on the robot in section 3. Next, we describe the Rollin' Justin system in section 4 before addressing the reasoning framework that enables efficient remote operation in section 5. Finally we discuss the limitations and the lessons learned of the architecture in section 6 before concluding the paper in section 7.

2. Related Work

Aside from the current Surface Avatar mission, there have been previous studies on space-to-ground or ground-to-space teleoperation of robots. ESA's Haptics and Interact experiments demonstrated the feasibility to command, from the ISS, with force-reflection with a single-rotational DoF joystick [7], [8], [9]. Haptics-1 primarily collected data about psycho-motor performance in operating the 1 DoF force-reflection joystick in microgravity. During Haptics-2, a haptic joystick on ground was coupled with the joystick on orbit, allowing for a teleoperated handshake between orbit and ground.

Telerobotics has played an important role in space, with the Canadarms both being used in the construction and servicing of the ISS for more than 20 years [10], [11]. To advance human-robot collaboration, numerous studies have been carried out to develop the technology necessary for orbit-surface robot command. The vision is to enable a future with holistic, Scalable Autonomy based command of robots in cislunar space and beyond [12]. In ROKVISS and KONTUR-1 [13] a 2 DoF robot mounted on the ISS was teleoperated from ground. In the follow-up experiment KONTUR-2 [14], [15], a 2 DoF force feedback joystick was upmassed and installed on-board the ISS to teleoperate a 2 DoF robot on ground as well as the humanoid SpaceJustin in a series of experiments. They used a direct point-to-point communication via S-Band, leading to low latency but putting an upper limit on the experiment time at 8-10 minutes. The system architecture of the KONTUR-2 mission is described in [16].

As discussed in section 1, different command abstraction levels have been demonstrated in ISS-to-Earth experiments. Using Supervised Autonomy, astronauts were able to command and manage a robot as an in-situ co-worker at the task level in METERON SUPVIS Justin to perform various inspection, maintenance, and assembly tasks [17], [18]. As the operator on the ISS and the robot on ground were connected via the Tracking and Data Relay Satellite System (TDRSS), there were virtually no limits on the experiment duration at the cost of a communication delay between operator and robot of $\approx$ 800ms. On the other hand, the possibility to intervene by taking over the robot directly in unknown situations and environments remains necessary, especially in exploration missions. The Analog-1 mission combines task-level commands and teleoperation. They demonstrated the possibility to stably command a robot dexterously with high-DoF input and force reflection, under high time-delay conditions [19].

Furthermore the *US National Aeronautics and Space Administration* (NASA) successfully landed the rovers Sojourner [20], Spirit [21], Opportunity [22], Curiosity [23], and Perseverance [24] on Mars. Furthermore, the team of Perseverance and the Ingenuity robotic helicopter [25] marked the first time of robot collaboration on the Martian surface. Due to the long round trip time between Earth and Mars, the mode of teleoperating these rovers differs fundamentally from our approaches, which deal with time delays in the range of below a second instead of minutes to hours. The systems engineering process of Curiosity is described in [26], systems engineering the parameters of Perseverance is described in [27].

NASA also deployed the Astrobee [28] on the ISS, an intravehicular free-flying system that can be commanded from ground through a GUI.

3. The Surface Avatar Mission

The Surface Avatar mission [3] investigates how to efficiently command a team of robots located on earth from aboard the ISS. The insights gained during this project will support developments for future missions where robots on the surface of extraterrestrial celestial bodies are remotely commanded by the crew of a nearby spacecraft.

Teleoperation from orbit faces multiple challenges, one of them being the time delay introduced by the communication

[1] `https://www.forcedimension.com/products/sigma`, last accessed Oct. 4th 2024

Figure 1. The robots in the Surface Avatar experimental Area, namely (1) Bert, (2) Interact, (3) Rollin' Justin and (4) TINA arm mounted to a lander.

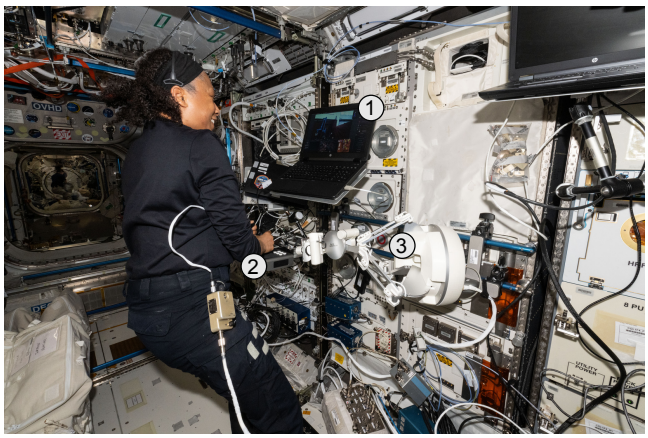


Figure 2. NASA astronaut Jeanette Epps on board the ISS in front of the Surface Avatar Experiment setup. The setup consists of (1) a computer running the Surface Avatar GUI, (2) a joystick, and (3) a sigma.7 haptic input device. image credit: ESA/NASA

infrastructure. In our setup, the operator and the robot exchange their data through the Multi Purpose Computer and Communication (MPCC) link [29], which connects the ISS to ground over the TDRSS. While this allows near-continuous coverage, it also introduces a round trip time delay of ∼800ms. An additional challenge is the limited bandwidth that is available for communication between operator and robot. During our experiments, the bandwidth for both, up and down-link, is 4 Mbit/s each.

One of the key features of Surface Avatar is that the astronaut commands a team of heterogeneous robots through a consistent, robot-agnostic User Interface (UI). This not only means that the user would not need a steep learning curve for new robotic assets that are introduced to the team, but also enhances the modularity to scale up the robotic team. Fig. 1 shows the heterogeneous robotic team, which consist of four robots: Rollin' Justin (3), Interact (2) [2], Bert (1) [30], and a robotic TINA arm [31] mounted to a lander (4). Rollin' Justin is a humanoid robot with two four-fingered hands and is good at dexterous manipulation tasks. The Interact rover consists of an AMBOT platform with rubber wheels and two KUKA light weight robot arms, one with a camera mounted at its end-effector and one with a Robotiq gripper. It is well suited for traversing long distances in rough terrains. Bert is a small quadruped robot that is able to traverse narrow passages. Finally, the lander mounted arm is used for sample stowage handling or providing scientific instruments from within the lander.

The Surface Avatar project consists of multiple experiment sessions, with time between each session to integrate feedback and lessons learned from previous sessions. In addition to this the size of the robot team grew between experiments. While the first experiments in 2022 started with a single robot (Rollin' Justin), the experiments in 2023 and 2024 each included three of the four robots.

The *Robot Command Terminal* (RCT) plays a central role in the Surface Avatar environment. It allows the operator to choose between two operation modalities, which are: Teleoperation and task-level autonomy. Fig. 2 shows an astronaut using the RCT during a Surface Avatar experiment session. The RCT consists of a graphical user interface (GUI) on a computer (1), a 3 DoF open-loop joystick (2), and 7-DoF Force Dimension sigma.7 force-reflection input device (sigma.7). The joystick and the sigma.7 are used to operate the robot in teleoperation while the GUI is used to command the robot via task-level autonomy.

The goal of the GUI is to give the astronaut insights into

Figure 3. Screenshot of the GUI used during the Surface Avatar experiments. The parts of the GUI are: (1) view of the robot's camera, (2) a map view, (3) a messenger panel, (4) an avatar view of the robot, (5) the available task-level commands, (6) teleoperation paneel, and (7) overlays marking the believed position of objects known to the robot.

the robot's belief state of the environment and allows them to issue task-level commands to the robot. Fig. 3 shows a screenshot of the GUI, which consists of six main elements: The *Main Camera View* (1) shows the view of the robot's camera augmented with blue overlays (7) of objects known to the robot. The overlays help to understand the robot's believe state. Upon selecting an object, the *command view* (5) on the right hand side is populated with the actions the robot can perform with the selected object. Two other views that give insights into the robot's believe state are the *Map View* (2), which shows the robot's believe state of the world from a birds eye view and the *Robot Avatar View* (4), which shows the robot's current configuration in a 3D viewer. The operators are always free to raise questions in a conversation with ground control via the *Messenger* (3), and receive experiment instructions.

If the operator decide to command the robot manually they

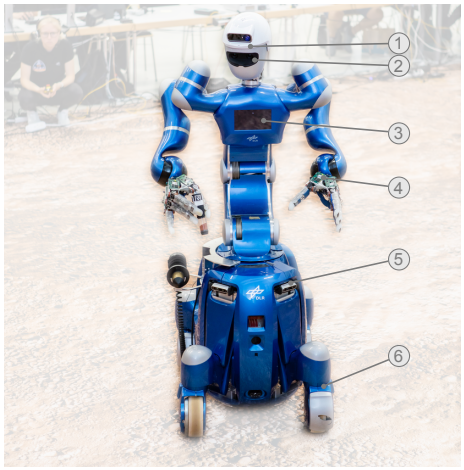


Figure 4. Image of Rollin' Justin showing (1) Azure Kinect camera, (2) the stereo camera setup, (3) the touchscreen, (4) DLR Hand-II, (5) realsense cameras in the base, and (6) the wheels.

can select the a teleoperation mode in the *Teleoperation Panel* (6) on the lower right. After selecting a teleoperation mode, the operator is able to move the robot manually by using the joystick or the sigma.7 device. While the joystick is used to teleoperate the robot's head or base, the sigma.7 is used to teleoperate the robot's arm. As the operator engages in teleoperation of the robot's arm, the robot's manipulator follows the motions that the operator commands to the sigma.7. The sigma.7 combines three translational and three rotational DoFs, allowing the user to move the manipulator in all six spatial directions. An additional 7th DoF in the form of a gripper is used to open and close the robot's manipulator. During teleoperation of the robot arm, external forces that get applied to the end-effector are rendered on the sigma.7 to increase situation awareness of the operator. The force feedback teleoperation with a round trip time delay of about 800ms is implemented with Time Domain Passivity Control (TDPC)[19].

4. System Description

Rollin' Justin [6] is a humanoid robot that consists of a wheeled base [32] and a humanoid upper body [33] as shown in Fig. 4. In the following we will describe various aspects of the robot that are related to hardware and communication, including the *Hardware Abstraction Layer (HAL)*.

Hardware

Rollin' Justins upper body consists of a torso and two arms, that are built from the DLR *Light Weight Robot's (LWRs)* joints [34], a head, and two of DLR's Hand II [35] hands (4 in Fig. 4). This adds up to 54 joints in the upper body as shown in Tab. 1 of which 43 are actuated and actively controlled in our experiments. The joints that are not actuated contain the fourth torso joint as it is an underactuated mirror joint, one joint per finger that is coupled with another finger joint, and one joint in each hand that is used to reconfigure the hand, a feature that we do not make use of in the Surface Avatar experiments.

The mobile base of the robot contains the battery and the PCs that are used to control Rollin' Justin. It also contains four legs (6 in Fig. 4), that each possess a lockable damper mechanism, a mechanism to extend the legs to increase stability of the platform, and a wheel. The wheels can be steered and driven individually which gives the robot base a holonomic behavior.

To sense its interactions with the environment, Rollin' Justin has torque sensors in the three actuated torso joints, in all arm joints, and in the 12 finger joints. It also possesses multiple cameras to sense the environment, namely four Realsense d435i in the base (5 in Fig. 4), an Azure Kinect in the head (1 in Fig. 4), and a stereo camera setup consisting of two Alvium 1800 C-319[2] cameras in the head (2 in Fig. 4). Regarding sensors, Rollin' Justin also contains an *Inertial Measurement Unit (IMU)* in the base. Furthermore, Rollin' Justin has a touch screen mounted on its torso (3 in Fig. 4) which allows to display information to and receive input from users. For communication with its surroundings, Rollin' Justin is also equipped with a speaker in its head.

Rollin' Justin was designed to be a self-sufficient unit which means that all compute necessary to operate the robot is on the robot itself. Thus, Rollin' Justin contains two real-time PCs, one application PC, and three NVIDIA® Jetsons®. Except for one NVIDIA Jetson which is mounted at the head of Rollin' Justin, all computing resources are located in the base. The application PC runs on an Intel® Core™ i7-7820EQ with 32GB RAM and a NVIDIA® GeForce® GT 1030. Real-time PC 1 runs on a Intel® Core™2 Quad CPU Q9000 with 4GB RAM and real-time PC 2 runs on an Intel® Pentium®4 with 1GB RAM.

Communication and HAL

The computer on Rollin' Justin are connected via Ethernet and Fig. 5 shows the resulting network architecture. As the Jetsons stream high-bandwidth video data that is only required by the application PC, they operate in a separated sub-net in order not to block the main network that connects the real-time PCs and the application PC. Additionally, a WiFi bridge is used to connect to Rollin' Justin from an outside terminal PC to observe telemetry and send commands. It is also used to connect Rollin' Justin to the experiment network, which also includes the RCT on ISS. The two resulting subnetworks are interconnected exclusively via the application PC, which functions as a relay point, allowing for the forwarding of information between the two networks.

For inter-process communication we use the DLR developed middleware *Links and Nodes (LN)* [36] which supports asynchronous communication via *topics* and synchronous communication via *services*. As LN supports communication via network, it connects the processes over all computers on Rollin' Justin. Furthermore, LN blends with the DLR developed driver layer, *robotkernel* [36], which we use as HAL. The robotkernel handles communication with the hardware on the robot, e.g. the robot joints, the steerings of the base, and the wheels.

LN also comes with a configurable UI, the LN manager, that can be used to start and stop processes among different hosts, investigate their status, and check their output. It is also possible to define dependencies between processes in the ln manager which substantially eases starting of different setups

[2] `https://www.alliedvision.com/en/products/alvium-configurator/alvium-1800-c/319/`, last accessed Oct. 4th 2024

Table 1. Displaying the number of joints per body part of Rollin' Justin and the number of joints that are actively controlled in our experiments.

Body Part	Torso	Left Arm	Right Arm	Head	Left Hand	Right Hand	Total
Number of Joints	4	7	7	2	17	17	**54**
Joints actively controlled in Surface Avatar	3	7	7	2	12	12	**43**

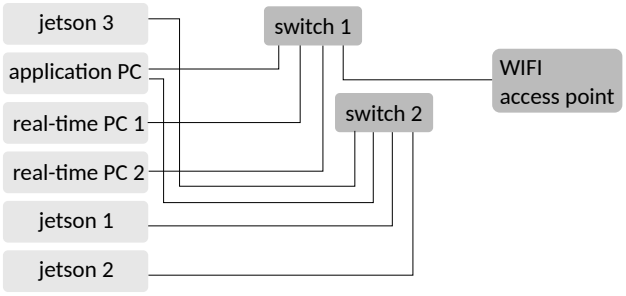


Figure 5. Network setup of the computers on Rollin' Justin. The two switches also represent two distinct sub-nets. Switch 1 connects the real-time PCs, the application PC, and a WIFI acces point allowing external connections. Switch 2 connects the application PC with the jetsons in a separate network.

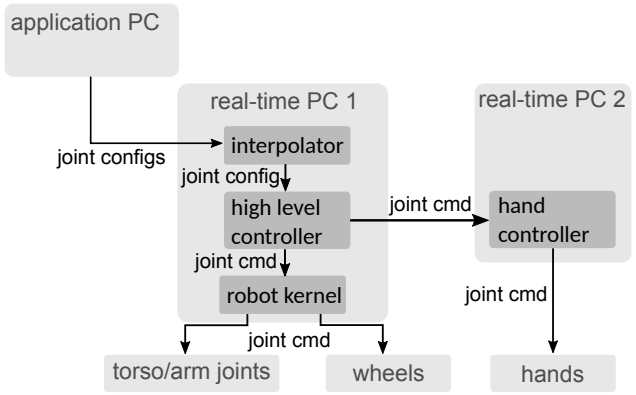


Figure 6. The path of creating joint commands for Rollin' Justins actuators from a desired joint configuration.

that contain many processes while adhering to certain requirements of the startup order. Furthermore, the LN manager allows to inspect data that is shared between processes in the form of topics. Overall, the LN manager plays a central role in our setup on Rollin' Justin as it serves as a central hub for all the information the team in the lab needs to monitor the status of the robot. The automation of starting many processes following predefined orders in combination with a restart check-list reduced reboot time of our robot to complete operability to approximately 4 minutes.

Given a program on the application PC commands a movement to the robot, it starts by sending the desired joint configurations to an interpolator on the first real-time PC. In our setup, a quadratic-spline-via interpolator running at 1kHz generates an interpolated trajectory and commands it to the high level robot controller running on the same real-time PC. Based on the selected control strategy, the highlevel controller generates joint commands for every joint of the robot, split up into commands for the hands and commands for all other joints. The joint commands, except for the hands, are then sent to the robotkernel which distributes them via SERCOS[3] bus to the robot joints. If the commanded trajectory includes movements of the wheels of the base, their actuation commands are distributed by the robotkernel via EtherCAT[4] to the wheels. The commands for the hands, in contrast, are sent from the high level controller to a hand controller running on the second real-time host. The reason for using a dedicated real-time host for the hands is to comply with the special hardware requirements for communication with the hands. This process is depicted in Fig. 6.

As we employ the *Data Distribution Service (DDS)* [37] for communication between robotic assets in the experiment and for communication with the astronaut on the ISS, we added a LN-to-DDS bridge which serves as an adapter between the DDS side of communication and the LN side. The LN-to-DDS bridge allows forwarding topics and services from DDS to LN or vice versa.

5. Reasoning Framework for Efficient Remote Operation

In order to be efficient co-workers for the astronauts, our robots provide command modes on multiple levels of autonomy, i.e. task-level Supervised Autonomy and teleoperation [3]. In the following we will distinguish between these command modes and the software modules that enable them. This manifests in our setup through the fact that only one of the command modes can be active at any time. When the robot is commanded in Supervised Autonomy, it does not accept teleoperation commands until the commanded action is finished. On the other hand, when the operator commands the robot via teleoperation, they can issue a task-level command at any given time, but it will lead to the teleoperation being disabled. This switching behavior is realized in a top-level module, that listens to all commands from the RCT and, thus, serves as the sole entry point to the robot.

In general our system follows service-oriented software architecture. As the requirements for a humanoid system are complex and vary strongly between software components, a monolithic software structure would not only be undesired but also impossible. Our system stretches over multiple hosts, some of them running real-time operating systems, others do not. This also reflects in the programming languages used to write different modules. Controllers are mostly implemented as simulink models, other real-time critical modules such as the HAL are written in C/C++, and reasoning modules are mostly written in python. This is partially due to different

[3] `https://www.sercos.org/`, last accessed Oct. 4th. 2024
[4] `https://www.ethercat.org`, last accessed Oct. 4th 2024

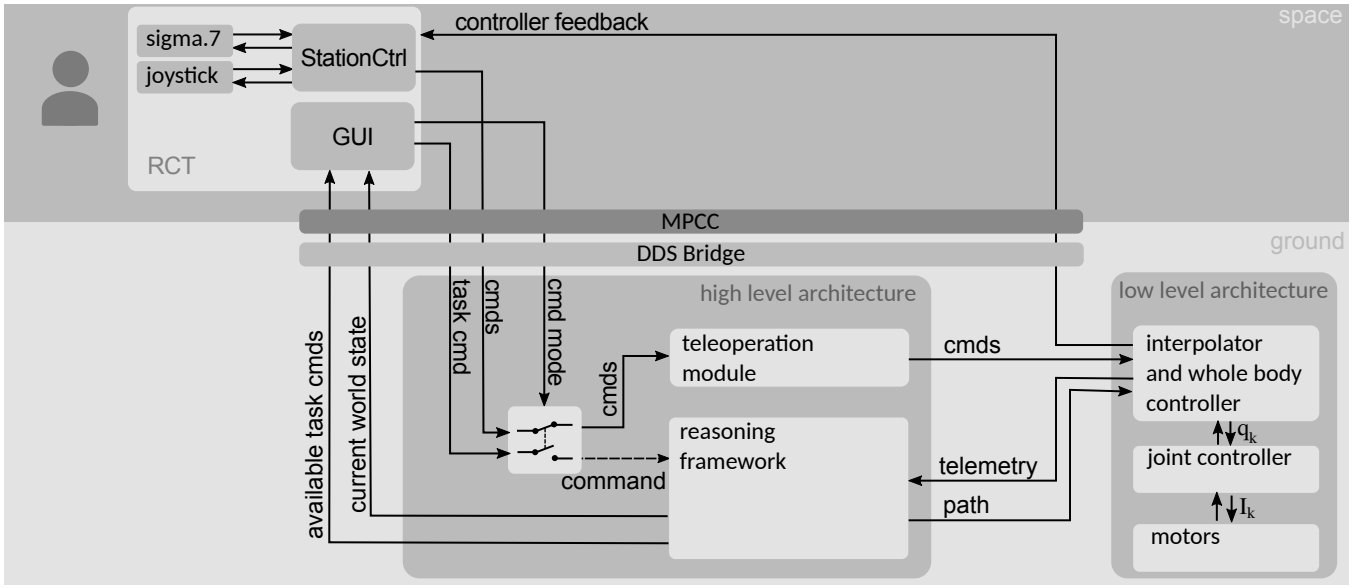


Figure 7. Overview depicting the relevant modules for commanding Rollin' Justin from Space.

requirements concerning the efficiency of programs, on the other hand an effort like building and maintaining a humanoid robot spans multiple disciplines and communities where different preferences over programming languages are prevalent. This lead us to follow a service-oriented architecture. In this architecture we also use an orchestrator that serves as an entry point for external commands and orchestrates the modules of the robot in such a way that they produce the desired outcome.

In the following we describe the module that enable task-level autonomy and teleoperation according to Fig. 7 and Fig. 8, followed by some additional information on the orchestrator. Fig. 7 depicts the overall setup with the space domain on top and the ground domain below, both connected via MPCC. The robot publishes the current world state and the available task-level commands to the RCT. The operator at the RCT selects to either send a task-level command (task cmd) to the robot or to command it via sigma.7 or joystick (cmd). The command mode (cmd mode) represents which of the two command modalities is selected. Depending on the command mode, either the teleoperation module or the reasoning framework are activated. If the robot arm is teleoperated, the robot also publishes controller feedback to the RCT.

Task-Level Autonomy

In order to allow for task-level autonomy, robots must announce their *capabilities* to the GUI. Capabilities represented in the form of actions that the robots can execute and render as task-level commands in the GUI. As the robot needs to be able to execute the actions directly, they must be fully defined. While there is no restriction on how to generate the actions on the robot side, we generate them in an approach based on hybrid planning with *Action Templates* [38], [39].

As the number of possible actions easily grows to an overwhelming number and we aim to support the astronauts in the selection of meaningful actions, we have a filtering pipeline for the actions in place [40]. The core question our system aims to answer is: "Given the current state of the environment of the robot, what are reasonable actions for an orbiting crew to choose?". To answer this question, we use a two step approach where the first step is to find out which actions are *possible* given the current world state and the second step filters the possible actions based on certain rules defined by the Mission Control team [40]. In the following we will describe the software modules that we employ to generate and filter the actions on the robot before announcing them to the astronauts. Fig. 8 shows an overview of these modules.

To extract the possible actions given an environment state, we start with the concept of *affordances*[41] which relates to the possibilities an object offers for interaction. Typical examples for affordances are a chair offering the affordance `sit_on` or a mug offering the affordances `fill_with` or `drink_from`. In our framework we represent the affordances in terms of action templates that are bound to objects or object classes.

Action Templates are defined in [38] and consist of two parts: the *symbolic header* and the *geometric body*. The symbolic header contains a symbolic description of the action in *Planning Domain Definition Language (PDDL)* [42] in terms of preconditions, parameters, and effects of the action. The geometric body on the other hand consists of primitive, robot agnostic operations that can be executed by the robot. Given a goal state in PDDL representation, the robot employs a method of *integrated Task and Motion Planning (integrated TAMP)* coined the *Hybrid Planner* [39] to generate a solution to reach the goal. Using the PDDL representations of the actions, the robot generates a task plan using the *Fast Downward* planner [43] which it then refines on the geometric level through a motion plan that is generated via openRAVE [44]. If the hybrid planner encounters a problem during generating the motion plan, it employs backtracking to find a suitable solution. This can mean that the planner either selects an alternative parameter on the geometric level (e.g. an alternative grasp) or that it triggers replanning on the symbolic level to find a different plan.

As mentioned above, Action Templates are bound to objects or object categories. We store static information about

objects, such as Action Templates, in the *Object Database (ODB)* [38]. As the ODB is designed to be extensible it can hold any sort of static information about objects but it usually contains the geometry in terms of a mesh, information about PDDL predicates the object supports, and Action Templates. It also supports inheritance of object properties as objects can derive from "virtual objects" which are interpreted as object classes. This allows to define an Action Template for a base class and reusing it for all deriving objects.

In addition to the static object knowledge, the robot also needs to be able to store runtime modifiable information about its environment. In our system this is achieved through the use of the *World Representation (WSR)* [38]. The WSR holds instances of the objects defined in the ODB alongside arbitrary runtime information about the objects, for example their position relative to an absolute coordinate frame and their PDDL states. Based on the information of the WSR, the hybrid planner creates plans to reach a specified goal.

The approach described above is well suited when the robot is commanded via desired goal states. For example the robot could be tasked to reach the PDDL goal `on table apple` and would find a sequence of actions that result in the expected goal. However, as commanding goal states is less intuitive than commanding actions, the GUI is designed to command actions. In order to compile a list of possible actions to announce to the GUI, we employ a module called *Mission Control* [40]. First of all, the Mission Control compiles a list of possible actions from the Action Templates related to the objects in the current world state. As the action descriptions in PDDL are very generic and, for runtime reasons, there are no sanity checks at this level, this can result in unreasonable actions. For example, by default every object can be picked based on the generic Action Template `pick` for the base object category, which can result in actions like picking objects in the environment that are too large or too heavy for the robot to pick. Thus, in a second step, these actions are filtered by comparing them to an allowlist and storing the result as available actions. In a third step, the Mission Control triggers the Symbolic Planner to determine the length of the shortest action sequence for every action in the available actions, that reaches its preconditions. In the last step, the Mission Control applies some user defined filters on the remaining actions. Typical filters are for example whether the list of the action sequence to reach the effects of an action is longer than a predefined threshold or whether all objects related to the action are within a maximum distance around the robot. Both of the filters reduce the number of actions announced to the GUI and improve relevance of these actions.

An overview of the modules that are involved in this process and how they interact is depicted in Fig. 8.

Aside from announcing possible actions to the GUI, the robot must also publish the position of the objects it knows of, to the GUI such that they can be displayed with overlays. As this information is also stored in the WSR, we added a functionality to publish it to the GUI. The WSR is, however, not solely used in the Surface Avatar context and is also used by multiple teams at DLR which do not require this functionality, thus, we integrated a plugin infrastructure into the WSR. Through the use of a plugin infrastructure, we can make sure not to deviate from the software used by other teams and streamline development while still being able to adapt the behavior of the software to our needs.

Another improvement we made to the original WSR is the

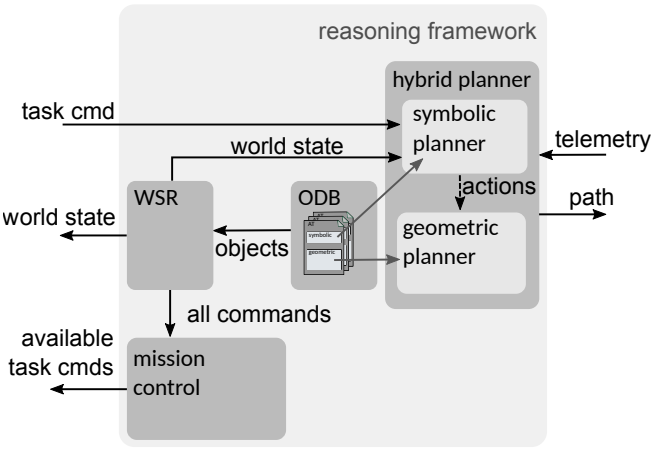


Figure 8. The reasoning framework employed on Rollin' Justin in the Surface Avatar mission.

ability to represent object frames not only in terms of a shared absolute origin, but with respect to other object's origins. This allows to create parent-child relations between objects where the child object moves accordingly to the motion of the parent object. Thus, when the frame of the parent object is updated, for example through a perception update, the frame of the child object is automatically updated as well. For perceiving the position of the robot in its environment the robot makes use of a *Simultaneous Localization and Mapping (SLAM)* approach [45]. Objects and their positions are perceived by the robot via AprilTag fiducial markers [46].

When adding more agents to the environment, it becomes necessary to include information also from other agents into the world representation. This concerns the positions of other agents as well as information about environmental assets. To allow for this, we added external sources to the world representation which can be used, for example, to update the pose of another agent based on its localization information.

Whenever an action is commanded from the operator to the robot, the robot employs the Hybrid Planner to create a valid motion plan for executing the action. During the process of compiling the list of actions to be announced to the GUI, we employ solely symbolic planning for determining the action sequence length. This is to cope with high number of action being checked. In comparison, symbolic planning is significantly faster than motion planning. Thus it is possible that actions that are presented to the operator can not be executed. We accept this behavior as it reduces the time needed for updating the list of actions which needs to be done after every action execution of the robot due to changes in the environment.

Teleoperation

Based on the approach of *Scalable Autonomy* as described in [3], the operator is free to chose on which level of autonomy to control the robot. In addition to task-level autonomy, the operator is able to command the robot via direct input through the sigma.7 device or the joystick in our setup. According to the classification of [47], this command mode falls under *Shared Control* as the robot follows the continuous input from the operator but augments it with its knowledge about the environment. In the Surface Avatar framework every robot can offer different Shared Control Capabilities.

On Rollin' Justin we implemented a mode to drive the base of the robot via the joystick, move the head via the joystick, and move the right arm via the sigma.7 device.

Driving the robot base and moving the robot head via the joystick are implemented in model mediated teleoperation. This means that the robot does not directly follow the inputs from the user at the remote side but instead follows a model that is published at the remote side. The RCT laptop, therefore, creates a virtual model of the robot, which is directly controlled via the operator input. The state of this model is then streamed to and followed by the robot. For driving the robot, the model of the robot becomes visible on the GUI and directly follows the operator input. The operator can, thus, command the position of the virtual model on the GUI to the desired position of the robot without having to deal with the delay between the robot and the command side. Similarly, when the operator commands the robot to move its head, they command a desired orientation of the head which is then locally followed by the robot. When the operators stop commanding the virtual model, as it reached the desired configuration, the robot still finishes to move to the desired configuration.

Another aspect in which the robot supports the operator during teleoperated driving of the base is collision avoidance. Due to the communication time of $\approx 800ms$ between robot and remote side in our experiments, combined with the restricted situation awareness because of the narrow field of view of the robot, the risk of collisions with environment assets of the robot becomes imminent. To cope with that risk and avoid collisions between Rollin' Justin and the environment, Rollin' Justin runs a collision avoidance algorithm that makes use of the four cameras in the base. Collisions are avoided by, firstly, creating a combined representation of the environment of the robot in an octree [48]. Next, we project the octree onto a two dimensional occupancy grid and compute the minimal distance from the center of the robot to the next obstacle in every direction. Given the minimum distance, we scale the velocities according to [49].

For teleoperating the arm of the robot we employ the *Time Domain Passivity Approach (TDPA)* [50], [19], which enables passivity based teleoperation with force feedback under time delays.

In the context of teleoperating the arm, it is also possible for the operator to open and close the hand of the robot via the gripper of the sigma.7 device. In our initial implementation, the position commanded from the gripper was directly mapped to the 12 dimensional position of the fingers. Therefore we defined an *open* grasp $\mathbf{g_o}$ and a *closed* grasp $\mathbf{g_{cl}}$ and calculated the commanded grasp $\mathbf{g_{cmd}}$ using the commanded gripper pose $cmd \in [0, 1]$ as:

$$\mathbf{g_{cmd}} = (\mathbf{g_{cl}} - \mathbf{g_o}) \cdot cmd \tag{1}$$

Using the approach from eq. (1) showed to have multiple drawbacks. First of all, when enabling teleoperation the approach lead to the robot's hand jumping directly to the position currently commanded by the operator. This is problematic as it is hard for the operator to estimate how far exactly to open the gripper to mirror the current hand configuration of the robot and because it requires the operator to focus on the hand when enabling teleoperation. Another problem with this approach is that the robot acts, i.e. opens or closes its hand, without a direct *motion* command from the operator but simply because of the position of the gripper when teleoperation is enabled.

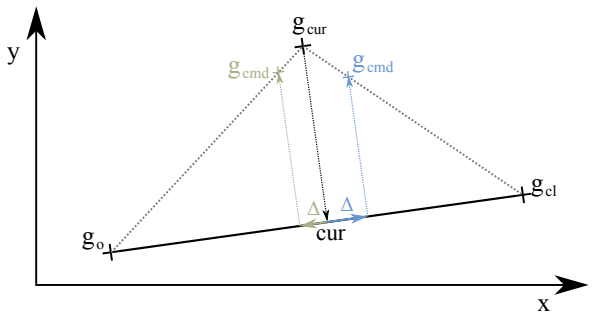


Figure 9. 2 dimensional representation of the approach to update the commanded grasp given a positive (in blue) or a negative Δ(in green). Intuitively the current grasp is projected on the line connecting the open and closed grasp definition, the Δ is applied, and the resulting point is projected back onto the line connecting either the open or close grasp with the current grasp.

To cope with the initial jump of the hand we implemented a *carrier* approach that keeps the hand static until the commanded grasp passes through the current grasp. Given the current grasp $\mathbf{g_{cur}}$ we compute a scalar cur that maps it to the same input space as the scalar commanded gripper pose according to eq. (2).

$$cur = \frac{(\mathbf{g_{cur}} - \mathbf{g_o}) \cdot (\mathbf{g_{cl}} - \mathbf{g_o})}{\|\mathbf{g_{cl}} - \mathbf{g_o}\|^2} (\mathbf{g_{cl}} - \mathbf{g_o}) \tag{2}$$

Using the current grasp, the hand starts to follow the commanded gripper pose when a change of the sign of $\Delta = cmd - cur$ is detected for the first time.

In the endeavor of extending this approach to a variety of mission protocols, the need for supporting multiple grasps on the robot emerged. As the operators were offered the possibility to select a grasp among multiple predefined grasp sets, the robot must support seamless transitioning between the grasp sets. A grasp set consists of two points in a 12 dimensional space that represent the open hand configuration and the closed hand configuration. While moving the gripper at the sigma.7, the robot hand moves along the line connecting both hand configurations in the 12 dimensional space. Intuitively, the approach is to project the current hand configuration onto the line connecting open and closed hand configuration, move on the line according to the commanded gripper position and project the new position back onto the line connecting either close or open configuration with the current configuration, depending on the commanded directions. Fig. 9 visualizes how a new grasp is computed in two dimensions and eq. (3) displays the general equation where the grasps $\mathbf{g}$ can be of arbitrary but consistent dimensionality.

$$\mathbf{g_{cmd}} = \begin{cases} \mathbf{g_{cur}} & \Delta = 0 \\ \mathbf{g_{cl}} + \frac{\Delta}{1-cur} (\mathbf{g_{cur}} - \mathbf{g_{cl}}) & \Delta > 0 \\ \mathbf{g_o} + \frac{cmd}{cur} (\mathbf{g_{cur}} - \mathbf{g_o}) & \Delta < 0 \end{cases} \tag{3}$$

Another source of information that can be used to ease control of the hand during teleoperation is whether the hand currently grasps an object. When switching from task-level autonomy to Shared Control, this knowledge is available and exploited

by our setup. In order to reduce the risk of dropping an object while it being grasped we initially lock the hand when switching to arm teleoperation while an object is grasped. The operator then has to manually unlock the hand before being able to open it. Unlocking the hand is implemented as a robot-centric action available via the task command panel and the state of the hand, locked versus unlocked, is stored in the WSR.

Gluing Things Together

Shared Control and task-level autonomy, as described above, both have unique requirements. The software modules dealing with Shared Control have high requirements to speed, as they deal with data that they receive at a rate of 166.6Hz. The module implementing the TDPC [19] for teleoperating the right arm of the robot even needs to run on a real-time host in order to guarantee stability. Other teleoperation modes like head or base teleoperation do not come with as strong requirements, as they run on the application PC, but still need to be able to handle the incoming data.

The requirements for the task-level autonomy modules, in contrast, mostly concern maintainability and flexibility. They run on the application PC as they do not have any real-time requirements. As they often require maintenance to add new features or improve their behavior, these modules are written in python.

The orchestration of all these different modules is performed in *verbose*, a DLR developed tool. Verbose allows to run python scripts, connect them to state-flows, and edit them at runtime. It also supports to run code in thread-based parallelism and share global variables. We use verbose as the single entry point for commands to the robot, for orchestration of modules, and for rapid prototyping of new functionalities. The approach for seamless switching between grasps, as described in Fig. 9, for example, was initially developed and tested in verbose before it was implemented in the controller.

6. DISCUSSION

Across all six ISS-to-ground sessions, comprising four preliminary and two primary sessions, the astronauts successfully completed all assigned tasks within our experiments. While a comprehensive analysis of the telecommanding concept will be presented in a forthcoming publication, these results preliminarily indicate that the robotic backend architecture effectively supported the requisite features for seamless UI-based command of Rollin' Justin from the ISS.

The system described above is still work in progress as throughout the experiments related to Surface Avatar, new features are continuously being implemented. Thus, the first and most important lesson we learned is to maximize modularity. Given the complexity of software needed to run a humanoid robot, it is of utmost importance to be able to easily swap out parts without the risk to trigger side-effects on other, seemingly unrelated, parts of the software. We increase modularity by separating functionalities into processes that use standardized interfaces in terms of Links and Nodes Message Definitions [36]. For the same reason we implemented a plugin infrastructure in the WSR as it separates out a functionality in a small, easily understandable and maintainable, package. The gold standard we are following in modularizing our software is to achieve loose coupling and high cohesion.

In order to deal with the increased complexity due to the growing number of modules through modularization and their interdependencies, we make use of DLR's Continuous Integration Software SYstem (CISSY) which combines a build host, an artifact server, and a package manager. For proper research rigor, and even more so for critical space systems, tractability of development history and documentation is paramount. We therefore implemented code reviews that check, among other code quality factors, for adequate documentation.

As our robot consists of multiple interconnected hosts that run software that interfaces with real hardware, it is challenging to create perfect copy of the system to use it for developing and testing. It prove, nevertheless, extremely important to have isolated setups that allow at least for testing subsets of the robots capability, as the availability of the robotic hardware becomes a bottleneck when a team of developers is working on it in parallel.

When employing a service based architecture as presented here for Rollin' Justin, the orchestrator and the glue code play a vital role in proper execution. However, its main function of connecting existing components is frequently less visible than other modules. In layman's term, it is a something that is not noticed until it is broken, which could then be difficult, or impossible, to recover from. As such, insufficient attention and resources dedicated to the glue code and orchestrator can later hinder the easy integration of new modules or could even result in failure of a mission.

Having designed the robot as a stand-alone unit is of great value in environment where wireless communication is challenging. This is particularly noticeable during the experiment runs when all robotic assets are enabled and communicate via Wi-Fi, reducing available bandwidth substantially. Additionally, the robots sometimes block another robots communication as they are moved in between the other robot and the Wi-Fi access point that connects it to the central network. In these situations it proves to be advantageous to require as little communication bandwidth as possible between the robot and the central experimental network.

So far the we use the described software stack as a proof of concept implementation on an Earth-based robot. While we take care to design the software architecture and components in a way that does not impede potential deployment in a space mission, the software is not yet space qualified. The computing power required to run the presented software depends highly on the implementation of all of its components. However, by running all of the software locally onboard the robot on the hardware described in section 4 and avoiding cloud-based solutions, we pave the way to eventually deploy this software on a flight-ready robot.

The system described in this paper was, and continues to be, well suited for the experiments in the Surface Avatar Mission. However, it is largely based on model-based knowledge in terms of objects, the world representation, and action templates. With current trends such as learning and especially foundation models, we will be tasked to redesign our system to leverage the full potential of these new technologies.

7. CONCLUSION

The system described in this paper has successfully been used in multiple astronaut trainings, two ISS-to-ground prime ses-

sions, and four preliminary ISS-to-ground sessions. Throughout these sessions, it prove successful in enabling astronauts onboard the ISS to command Rollin' Justin on multiple levels of autonomy. During that time the system underwent multiple transformations and the features of the system kept growing and still continue to do so.

We will continuously work on improving our software design further to be able to offer ever more features for the upcoming third Surface Avatar prime session in the first half of 2025 and the Axiom-4 session in the second half of 2025. Our next steps in Surface Avatar for these sessions are mixed-type command robot collaboration and supporting setups with possibly multiple RCTs at different locations. We also aim for the execution of longer task-sequences which requires additional efforts in environment perception and modeling, as well as detection of and reaction to unexpected events such as failures.

Another upcoming effort will be to make use of the experience gained with our current system in designing an architecture that is not only tailored for one robot but instead supports a broader variety of robots.

ACKNOWLEDGMENTS

Realizing the Surface Avatar experiments would not have been possible without the support of the German Space Operations Center (GSOC), the Columbus Control Centre (ColCC), and the European Astronaut Training Centre (EAC). We thank them for the support during experiment preparation, testing, and astronaut training.

REFERENCES

[1] N. Y. Lii, C. Riecke, D. Leidner, S. Schätzle, P. Schmaus, B. Weber, T. Krueger, M. Stelzer, A. Wedler, and G. Grunwald, "The Robot as an Avatar or Co-worker? An Investigation of the Different Teleoperation Modalities through the KONTUR-2 and METERON SUPVIS Justin Space Telerobotic Missions," in *Proc. Int. Astronaut. Congr. IAC*, Bremen, Germany, Oct. 2018.

[2] T. Krueger, E. Ferreira, A. Gherghescu, L. Hann, E. den Exter, F. P. van der Hulst, L. Gerdes, A. Pereira, H. Singh, and M. Panzirsch, "Designing and testing a robotic avatar for space-to-ground teleoperation: The developers' insights," in *71st Int. Astronaut. Congr. IAC 2020*. International Astronautical Federation, 2020.

[3] N. Y. Lii, P. Schmaus, D. Leidner, T. Krueger, J. Grenouilleau, A. Pereira, A. Giuliano, A. S. Bauer, A. Köpken, F. S. Lay, M. Sewtz, N. Bechtel, S. Bustamante Gomez, M. Denninger, W. Friedl, J. Butterfass, E. Ferreira, A. Gherghescu, T. Chupin, E. den Exter, L. Gerdes, M. Panzirsch, H. Singh, R. Balachandrand, T. Hulin, T. Gumpert, A. Schmidt, D. Seidel, M. Hermann, M. Maier, R. Burger, F. Schmidt, B. Weber, R. Bayer, B. Pleintinger, R. Holderried, P. H. Pavelski, A. Wedler, S. von Dombrowski, H. Maurer, M. Görner, T. Wüsthoff, S. Bertone, T. Müller, G. Söllner, C. Ehrhardt, L. Brunetti, L. Holl, M. Bévan, R. Muehlbauer, G. Visentin, and A. Albu-Schäffer, "Introduction to Surface Avatar: The First Heterogeneous Robotic Team to be Commanded with Scalable Autonomy from the ISS," in *Proc. Int. Astronaut. Congr. IAC*, vol. IAC-22. Paris, France: International Astronautical Federation, IAF, Sep. 2022.

[4] P. Schmaus, A. S. Bauer, N. Bechtel, M. Denninger, A. Köpken, F. S. Lay, F. Schmidt, M. Sewtz, T. Krüger, D. Leidner, A. Pereira, and N. Y. Lii, "Extending the Knowledge Driven Approach for Scalable Autonomy Teleoperation of a Robotic Avatar," in *2023 IEEE Aerosp. Conf. AERO 2023*. Big Sky, MT, USA: IEEE, May 2023.

[5] P. Schmaus, N. Batti, A. S. Bauer, J. Beck, T. Chupin, E. den Exter, N. Grabner, A. Köpken, F. S. Lay, M. Sewtz, D. Leidner, T. Krüger, and N. Y. Lii, "Toward Multi User Knowledge Driven Teleoperation of a Robotic Team with Scalable Autonomy," in *2023 IEEE Int. Conf. Syst. Man Cybern. SMC*. Honolulu, Oahu, HI, USA: IEEE, Jan. 2024.

[6] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schaffer, and G. Hirzinger, "Rollin' Justin - Mobile platform with variable base," in *Proc. 2009 IEEE Int. Conf. Robot. Autom. ICRA*. Kobe, Japan: IEEE, May 2009, pp. 1597–1598.

[7] A. Schiele, "Towards the interact space experiment: Controlling an outdoor robot on earth's surface from space," in *Proc 13th Symp. Adv. Space Technol. Robot. Autom. ASTRA*, 2015.

[8] A. Schiele, M. Aiple, T. Krueger, F. van der Hulst, S. Kimmer, J. Smisek, and E. den Exter, "Haptics-1: Preliminary Results from the First Stiffness JND Identification Experiment in Space," in *Haptics Percept. Devices Control Appl.*, F. Bello, H. Kajimoto, and Y. Visell, Eds. Cham: Springer International Publishing, 2016, pp. 13–22.

[9] A. Schiele, T. Krüger, S. Kimmer, M. Aiple, J. Rebelo, J. Smisek, E. den Exter, E. Mattheson, A. Hernandez, and F. van der Hulst, "Haptics-2 — A system for bilateral control experiments from space to ground via geosynchronous satellites," in *2016 IEEE Int. Conf. Syst. Man Cybern. SMC*, Oct. 2016, pp. 892–897.

[10] L. Oshinowo, R. Mukherji, C. Lyn, and A. Ogilvie, "On the Application of Robotics to On-Orbit Spacecraft Servicing-The Next Generation Canadarm Project," in *Proc 11th Intl Symp. Artif. Intell. Robot. Autom. Space ISAIRAS*, 2012, pp. 3–7.

[11] M. Hiltz, C. Rice, K. Boyle, and R. Allison, "CANADARM: 20 Years of Mission Success Through Adaptation," in *International Symposium on Artificial Intelligence, Robotics and Automation*, Montreal, Jun. 2001.

[12] N. Y. Lii, T. Krüger, P. Schmaus, D. Leidner, S. Paternostro, A. S. Bauer, N. Batti, A. Köpken, F. S. Lay, A. N. Manaprampil, L. Mayerhofer, R. Luz, and E. den Exter et. Al., "Everything is awesome if you are part of a (robotic) team: Preliminary insights from the first ISS-to-surface multi-robot collaboration with scalable autonomy teleoperation," in *Proc. 75rd Int. Astronaut. Congr. IAC*. International Astronautical Federation, Oct. 2024.

[13] G. Hirzinger, K. Landzettel, D. Reintsema, C. Preusche, A. Albu-Schaeffer, B. Rebele, and M. Turk, "ROKVISS – ROBOTICS COMPONENT VERIFICATION ON ISS," in *Proc 8th Int. Symp. Artifical Intell. Robot. Autom. Space - ISAIRAS*, Munich, Germany, Sep. 2005.

[14] C. Riecke, J. Artigas, R. Balachandran, R. Bayer,

A. Beyer, B. Brunner, J. Buchner, T. Gumpert, R. Gruber, F. Hacker, K. Landzettel, G. Plank, S. Schätzle, H.-J. Sedlmayr, N. Seitz, B.-M. Steinmetz, M. Stelzer, J. Vogel, B. Weber, B. Willberg, and A. O. Albu-Schäffer, "KONTUR-2 MISSION: THE DLR FORCE FEEDBACK JOYSTICK FOR SPACE TELEMANIPULATION FROM THE ISS," in *The International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2016)*, Beijing, China, Dec. 2016.

[15] J. Artigas, R. Balachandran, C. Riecke, M. Stelzer, B. Weber, J. H. Ryu, and A. Albu-Schaeffer, "KONTUR-2: Force-feedback teleoperation from the international space station," in *Proc 2016 IEEE Int Conf Robot. Autom. ICRA*, Stockholm, Sweden, May 2016, pp. 1166–1173.

[16] M. Stelzer, B.-M. Steinmetz, P. Birkenkampf, J. Vogel, B. Brunner, and S. Kühne, "Software architecture and design of the Kontur-2 mission," in *2017 IEEE Aerosp. Conf.*, Mar. 2017, pp. 1–17.

[17] P. Schmaus, D. Leidner, T. Krüger, A. Schiele, B. Pleintinger, R. Bayer, and N. Y. Lii, "Preliminary Insights From the METERON SUPVIS Justin Space-Robotics Experiment," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3836–3843, Oct. 2018.

[18] P. Schmaus, D. Leidner, T. Krüger, R. Bayer, B. Pleintinger, A. Schiele, and N. Y. Lii, "Knowledge Driven Orbit-to-Ground Teleoperation of a Robot Coworker," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 143–150, Jan. 2020.

[19] M. Panzirsch, A. Pereira, H. Singh, B. Weber, E. Ferreira, A. Gherghescu, L. Hann, E. den Exter, F. van der Hulst, L. Gerdes, L. Cencetti, K. Wormnes, J. Grenouilleau, W. Carey, R. Balachandran, T. Hulin, C. Ott, D. Leidner, A. Albu-Schäffer, N. Y. Lii, and T. Krüger, "Exploring planet geology through force-feedback telemanipulation from orbit," *Sci. Robot.*, vol. 7, no. 65, p. eabl6307, Apr. 2022.

[20] B. Wilcox and T. Nguyen, "Sojourner on Mars and Lessons Learned for Future Planetary Rovers," in *International Conference On Environmental Systems*, Jul. 1998, p. 981695.

[21] R. E. Arvidson, S. W. Squyres, R. C. Anderson, J. F. Bell III, D. Blaney, J. Brückner, N. A. Cabrol, W. M. Calvin, M. H. Carr, P. R. Christensen, B. C. Clark, L. Crumpler, D. J. Des Marais, P. A. de Souza Jr., C. d'Uston, T. Economou, J. Farmer, W. H. Farrand, W. Folkner, M. Golombek, S. Gorevan, J. A. Grant, R. Greeley, J. Grotzinger, E. Guinness, B. C. Hahn, L. Haskin, K. E. Herkenhoff, J. A. Hurowitz, S. Hviid, J. R. Johnson, G. Klingelhöfer, A. H. Knoll, G. Landis, C. Leff, M. Lemmon, R. Li, M. B. Madsen, M. C. Malin, S. M. McLennan, H. Y. McSween, D. W. Ming, J. Moersch, R. V. Morris, T. Parker, J. W. Rice Jr., L. Richter, R. Rieder, D. S. Rodionov, C. Schröder, M. Sims, M. Smith, P. Smith, L. A. Soderblom, R. Sullivan, S. D. Thompson, N. J. Tosca, A. Wang, H. Wänke, J. Ward, T. Wdowiak, M. Wolff, and A. Yen, "Overview of the Spirit Mars Exploration Rover Mission to Gusev Crater: Landing site to Backstay Rock in the Columbia Hills," *J. Geophys. Res. Planets*, vol. 111, no. E2, 2006.

[22] S. W. Squyres, R. E. Arvidson, D. Bollen, J. F. Bell III, J. Brückner, N. A. Cabrol, W. M. Calvin, M. H. Carr, P. R. Christensen, B. C. Clark, L. Crumpler, D. J. Des Marais, C. d'Uston, T. Economou, J. Farmer, W. H. Farrand, W. Folkner, R. Gellert, T. D. Glotch, M. Golombek, S. Gorevan, J. A. Grant, R. Greeley, J. Grotzinger, K. E. Herkenhoff, S. Hviid, J. R. Johnson, G. Klingelhöfer, A. H. Knoll, G. Landis, M. Lemmon, R. Li, M. B. Madsen, M. C. Malin, S. M. McLennan, H. Y. McSween, D. W. Ming, J. Moersch, R. V. Morris, T. Parker, J. W. Rice Jr., L. Richter, R. Rieder, C. Schröder, M. Sims, M. Smith, P. Smith, L. A. Soderblom, R. Sullivan, N. J. Tosca, H. Wänke, T. Wdowiak, M. Wolff, and A. Yen, "Overview of the Opportunity Mars Exploration Rover Mission to Meridiani Planum: Eagle Crater to Purgatory Ripple," *J. Geophys. Res. Planets*, vol. 111, no. E12, 2006.

[23] A. R. Vasavada, "Mission Overview and Scientific Contributions from the Mars Science Laboratory Curiosity Rover After Eight Years of Surface Operations," *Space Sci Rev*, vol. 218, no. 3, p. 14, Apr. 2022.

[24] J. Balaram, M. Aung, and M. P. Golombek, "The Ingenuity Helicopter on the Perseverance Rover," *Space Sci Rev*, vol. 217, no. 4, p. 56, Jun. 2021.

[25] M. Golombek, N. Williams, H. Grip, T. Tzanetos, J. Balaram, J. Maki, R. Deen, F. Ayoub, M. Mischna, C. Brooks, E. Romashkova, M. Deahn, J. Tarnas, T. del Sesto, L. Crumpler, and R. Sullivan, "The Mars Helicopter, Ingenuity: Operations and Initial Results," *44th COSPAR Sci. Assem.*, vol. 44, p. 362, Jul. 2022.

[26] R. Welch, D. Limonadi, and R. Manning, "Systems engineering the Curiosity Rover: A retrospective," in *2013 8th Int. Conf. Syst. Syst. Eng.*, Jun. 2013, pp. 70–75.

[27] R. S. Siegfriedt, E. Bohannon, A. Girerd, I. Trettel, and B. Roth, "Making or Breaking a Rover- Systems Engineering Parameters On-Board the Mars 2020 Perseverance Rover," in *2022 IEEE Aerosp. Conf. AERO*, Mar. 2022, pp. 1–14.

[28] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, and E. Smith, "Astrobee: A New Platform for Free-Flying Robotics on the International Space Station," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, Beijing, Jun. 2016.

[29] T. Müller, D. Burdulis, and C. Corsten, "MPCC and Ku-IPS, New Ways to Control the Next Generation of Columbus Payloads-Ground Segment Aspects," in *Proc. Int. Astronaut. Congr. IAC*, Adelaide, Australia, Sep. 2017, pp. 1–11.

[30] D. Lakatos, K. Ploeger, F. Loeffl, D. Seidel, F. Schmidt, T. Gumpert, F. John, T. Bertram, and A. Albu-Schaffer, "Dynamic Locomotion Gaits of a Compliantly Actuated Quadruped With SLIP-Like Articulated Legs Embodied in the Mechanical Design," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3908–3915, Oct. 2018.

[31] M. Maier, M. Chalon, M. Pfanne, R. Bayer, M. M. Mascarenhas, H.-J. Sedlmayr, and A. L. Shu, "TINA: Small torque controlled robotic arm for exploration and small satellites," in *Proc. Int. Astronaut. Congr. IAC*, Washington D.C., 2019.

[32] M. Fuchs, P. R. Giordano, C. Borst, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, and K. Kunze, "Justin's mobile platform: A workspace extension for two-handed manipulation," in *Proc. ICRA*, 2009.

[33] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner,

H. Hirschmuller, S. Kielhofer *et al.*, "A humanoid two-arm system for dexterous manipulation," in *Proc 6th IEEE-RAS Int Conf Humanoid Robots 2006*, Genova, Italy, Dec. 2006, pp. 276–283.

[34] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot III-are we reaching the technological limits now?" in *Proc. 2002 IEEE Int. Conf. Robot. Autom. Cat No02CH37292*, vol. 2, May 2002, pp. 1710–1716 vol.2.

[35] J. Butterfass, M. Grebenstein, H. Liu, and G. Hirzinger, "DLR-Hand II: Next generation of a dextrous robot hand," in *Proc. 2001 ICRA IEEE Int. Conf. Robot. Autom. Cat No01CH37164*, vol. 1, May 2001, pp. 109–114 vol.1.

[36] F. Schmidt and R. Burger, "How we deal with software complexity in robotics:'Links and nodes' and the 'robotkernel'," in *14th IEEE-RAS Int. Conf. Humanoid Robots Humanoids*, 2014.

[37] G. Pardo-Castellote, "OMG Data-Distribution Service: Architectural overview," in *23rd Int. Conf. Distrib. Comput. Syst. Workshop 2003 Proc.*, May 2003, pp. 200–206.

[38] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Proc. 2012 IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2012, pp. 429–435.

[39] D. S. Leidner, *Cognitive Reasoning for Compliant Robot Manipulation*, 1st ed., ser. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2019, vol. 127.

[40] D. Leidner, P. Birkenkampf, and N. Y. Lii, "Context-aware Mission Control for Astronaut-Robot Collaboration," in *Proc 14th Symp Adv. Space Technol. Robot. Autom. ASTRA*. Leiden, The Netherlands: European Space Agency (ESA), Jun. 2017.

[41] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, J. B. Robert E Shaw, Ed. Hillsdale, N.J. : Lawrence Erlbaum Associates, 1977, pp. pp.67–82.

[42] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL - The Planning Domain Definition Language," Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165, 1998.

[43] M. Helmert, "The Fast Downward Planning System," *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, Jul. 2006.

[44] R. Diankov, "Automated Construction of Robotic Manipulation Programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, 2010.

[45] M. Sewtz, X. Luo, J. Landgraf, T. Bodenmüller, and R. Triebel, "Robust Approaches for Localization on Multi-camera Systems in Dynamic Environments," in *2021 7th Int. Conf. Autom. Robot. Appl. ICARA*, Feb. 2021, pp. 211–215.

[46] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc 2011 IEEE Int Conf Robot. Autom. ICRA*, Shanghai, China, May 2011, pp. 3400–3407.

[47] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and Beyond for Assistive Humanoid Robots," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 175–226, Nov. 2013.

[48] D. J. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer," Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, Techinal Report IPL-TR-80-111, 1980.

[49] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *2010 Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 3109–3116.

[50] J.-H. Ryu, D.-S. Kwon, and B. Hannaford, "Stable Teleoperation With Time-Domain Passivity Control," *IEEE Trans. Robot. Automat.*, vol. 20, no. 2, pp. 365–373, Apr. 2004.

BIOGRAPHY

Adrian S. Bauer *received a Bachelor in Mechanical Engineering in 2012, a Bachelor in Cognitive Sciences in 2015, and a master in Robotics, Cognition, Intelligence from TU Munich in 2018. Currently he is pursuing a PhD in robotics at the German Aerospace Center. His interest is in enabling robotics to generate meaningful symbolic plans in presence of epistemic uncertainty.*

Anne Köpken *received a Bachelor in Electrical Engineering from TU Munich in 2019, and a Master in Robotics, Cognition, Intelligence from TU Munich in 2021. She spent one semester at the JSK Laboratory at the University of Tokyo in 2019/20. Currently she is pursuing a PhD in robotics at the German Aerospace Center in Oberpfaffenhofen near Munich. She is interested in enabling robots to cope with unexpected situations and finding ways to prevent and recover from failures.*

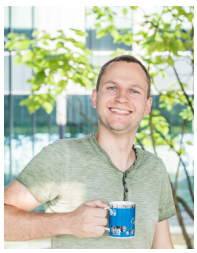
Nesrine Batti *received her "Diplôme d'Ingénieur" from the National Institute of Applied Science and Technology, University of Carthage, Tunisia in 2022. Since then she is pursuing a PhD in robotics at the German Aerospace Center in Oberpfaffenhofen, Germany. She investigates how robots can learn from past errors and transfer this acquired experience to other robots to enhance the collective performance.*

Jörg Butterfaß *received his diploma degree from the Technical University of Darmstadt in 1993 and his doctoral degree in 1999 respectively. He joined the German Aerospace Center (DLR) Institute of Robotics and Mechatronics in 1993 where he was involved in the design of various robotic hands. A further working field is the design of robotic sensors for space applications.*

Tristan Ehlert *holds a B.Sc. in General Engineering Science from the TUHH and an M.Sc. in Robotics and Mechatronics from the TU Munich. Since 2023 he works at the German Aerospace Center (DLR), where he investigates and develops elastic robots for efficient locomotion.*

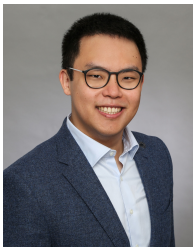
Emiel den Exter *works as Industrial Design and Human Factors Engineer at the ESA Human Robot Interaction Lab and ATG-Europe VirtualLab. He is specialised in developing novel user interfaces for Robotics and XR Engineering solutions within the Space sector.*

Werner Friedl *received his Dipl.-Ing.(FH) in Mechatronic at the University of Applied Sciences in Munich and started at DLR in 2004. In 2006 he developed the torso of DLR's Humanoid Justin. In the DLR Hand-Arm- project he developed the forearm of the AWIWI hand and AWIWI II. Since 2015 he is responsible for the mechanical hand development at DLR. His main research focus includes variable stiffness actuation, tendon driven hands and grasping.*



Thomas Gumpert *received his B.Eng. in Mechatronics 2012 and in 2018 his M.Sc. in Applied Research on Mechatronic Systems from the University of Applied Sciences Augsburg. He joined the German Aerospace Center (DLR) Institute of Robotics and Mechatronics in 2008 with focus on electrical drives. Since 2015 he is heading the Drive Technology Lab. As part of different teams he was involved in the development of the lightweight robot SARA and space qualified robotic arm CAESAR as well as the space qualified force feedback joystick Kontur-2. Currently he is responsible for the robotic hardware of the humanoid robots Rollin and Agile Justin and he coordinates the development of DLR's quadruped robot bert with his main focus on actuator and sensor electronics.*

Florian S. Lay *received his B.Sc. degree in Engineering Science in 2018, and his M.Sc. in "Robotics, Cognition, Intelligence" in 2020 both from the Technical University of Munich. Since 2020 he is pursuing a PhD in robotics at the German Aerospace Center (DLR). His interests range from symbol grounding and emergence for task and motion planning to multi-robot world representations.*

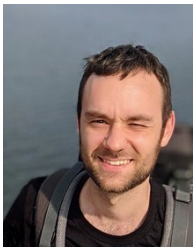


Xiaozhou Luo *received his B.Sc. in Mechanical Engineering in 2019, and his M.Sc. in Aerospace Engineering in 2023, both at the Technical University of Munich. Following his graduation, he started as a research fellow at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). His research focuses on SLAM and the creation of semantic 3D maps for navigation purposes in dynamic indoor environments.*

Rute Luz *received her BSc and MSc degrees in aerospace engineering from Instituto Superior Técnico, Lisbon, Portugal, where she is currently pursuing her PhD. Since 2018 she is a researcher at the Institute for Systems and Robotics, and at the Interactive Technologies Institute. In 2024, she also integrated the team at the Human Robot Interaction lab at ESA. Her research interests involve human-robot interaction, haptic interfaces, and robotics and enabling more effective teleoperation.*

Ajithkumar N. Manaparampil *received his B.E. in Mechanical Engineering from the University of Pune, India and his M.Sc. in "Robotic Systems Engineering" from RWTH Aachen University. He is a Research Associate at the Center for Robotics and Mechatronics of the German Aerospace Center (DLR) since November 2023. His research interests include teleoperation with a focus on assistance in manipulation tasks.*

Luisa Mayershofer *received a B.Eng. in Electrical Engineering from the University of Applied Sciences in Augsburg and a M.Sc. in "Robotics, Cognition, Intelligence" from the Technical University of Munich. Since November 2023, she is pursuing a PhD at the Center for Robotics and Mechatronics of the German Aerospace Center (DLR). Her research interests lie in Human-Robot-Interaction, with a focus on teleoperation and task model estimation as well as multimodal user interfaces.*



Antonin Raffin *is a research engineer in reinforcement learning (RL) at the German Aerospace Center (DLR). He is the lead developer of Stable-Baselines3, an open source library that implements deep RL algorithms. Raffin's work focuses on improving the reproducibility of RL and its applications in robotics. In particular, he aims to learn directly on real robots.*

Annika Schmidt *holds a B.S. degree in Biomimikry and received her M.S. in Mechanical Engineering from the Technical University of Delft. Since then she has been working jointly at the Technical University of Munich and the German Aerospace Center to obtain her Ph.D. in bioinspired control mechanisms for robots. In this, she combines Neuroscience elements with human user studies and control theory for soft robots.*

Florian Schmidt *received his Master of Science degree from the Munich University of Applied Sciences in 2007 (computer graphics and digital image processing). Since then he is with the German Aerospace Centers Institute of Robotics and Mechatronics. In his master thesis he developed a planning system to solve the rubik's cube with the humanoid robot Justin. His research interests include task and motion planning, real-time capable software architectures and novel robot programming interfaces.*

Daniel Seidel *received his M.Sc. degree in "Intelligent Systems" from Bielefeld University, Germany, in 2014. Afterwards he joined the Chair for Sensor Based Robotic Systems and Intelligent Assistance Systems of Prof. Albu-Schäffer at the Technical University of Munich. Futhermore, he is an ongoing researcher at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) where he is involved in the development of the DLR quadruped robot Bert. His main interests lie in motion generation and planning for elastically actuated legged robots and machine learning on hardware systems.*

Peter Schmaus *received his M.Sc. Degree in "Robotics, Cognition, Intelligence" from Technical University of Munich, Germany, in 2013. He joined the German Aerospace Center (DLR) Institute of Robotics and Mechatronics in 2011 where he was involved in the ISS-to-ground telerobotics projects Kontur-2, METERON SUPVIS Justin, and became Co-Investigator of the Surface Avatar experiment suite. His main interests lie in Shared Autonomy and effective Human-Robot Interaction.*

Daniel Leidner *Daniel Leidner received his diploma degree in communications engineering in 2010, and his M.Sc. degree in information technology in 2011 with distinction from the Mannheim University of Applied Sciences, Mannheim, Germany. In 2017 he received the Ph.D. degree in artificial intelligence from the University of Bremen, Bremen, Germany. His dissertation was honored with the Georges Giralt PhD Award as well as the Helmholtz Doctoral Prize. He is a Professor at the Institute of Artificial Intelligence at the University of Bremen and leads the department of Autonomy and Teleoperation at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. As Co-Investigator of the Surface Avatar experiments, he investigates artificial intelligence in the context of astronaut-robot collaboration.*

Thomas Krüger *is head of the Human Robot Interaction Lab at ESA. Involved with ESA's haptic experiments since the beginning, Thomas headed the technology development for the ANALOG-1 experiments and is Co-Investigator on Surface Avatar. He received his PhD in Lab Automation from the University of Rostock and worked on wind turbines before joining ESA.*

Neal Y. Lii *is the domain head of Space Robotic Assistance, and the co-founding head of the Modular Dexterous (Modex) Robotics Laboratory at the German Aerospace Center (DLR). Neal received his BS, MS, and PhD degrees from Purdue University, Stanford University, and University of Cambridge, respectively. He has served as the principal investigator of the ISS-to-Earth telerobotic experiments, Surface Avatar, and METERON SUPVIS Justin. Neal is primarily interested in the use of telerobotics in both space and terrestrial applications.*