Enabling Communication between Heterogeneous Robots and Human Operators in Collaborative Missions

Marco Sewtz[1,2], Florian Samuel Lay[1], Xiaozhou Luo[1], Thibaud Chupin[2], and Neal Y. Lii[1]

[1]German Aerospace Center, Institut of Robotics and Mechatronics, Muenchener Str. 20, 82234 Weßling, Germany
{*Firstname.Lastname*}@dlr.de
[2]European Space Angency, Human Robot Interaction Lab, Keplerlaan 1, 2201AZ Noordwijk, Netherlands
{*Firstname.Lastname*}@ext.esa.int

Abstract*—Enabling the return of human presence to the lunar surface constitutes a central objective for various space agencies. While previous lunar missions were of limited duration, the landscape is poised for significant transformation in the coming years, characterized by extended surface operations and the establishment of a permanent base near the lunar south pole. As emphasized in the National Aeronautics and Space Administration (NASA)'s technology roadmap and echoed by the European Space Agency (ESA) in its Terrae Novae 2030+ roadmap, the pivotal role of robotics is underscored for attaining a sustainable lunar base. The Surface Avatar mission, led by the German Aerospace Center (DLR) and partnered by ESA, represents a pioneering effort aimed at investigating the practical application of scalable autonomy through multi-modal tele-operation and task-oriented command protocols. This approach empowers astronauts with the capability to oversee and direct a diverse fleet of robots, each with unique functions and capabilities. Aboard the International Space Station (ISS), crew members are currently entrusted with the command of a diverse ensemble of ground-based robots, including the wheeled humanoid known as Rollin' Justin, the versatile rover Interact, the articulated arm of a lander mockup, and a small four-legged system named BERT. Nonetheless, the coexistence of multiple disparate robotic systems within the same network presents a considerable challenge in achieving sustainable development. Adapting to each system's specific requirements with every update or altering the communication infrastructure to accommodate new combinations of robots is not conducive to long-term operational efficiency. This work delves into a comprehensive and modular approach designed to mitigate these challenges by minimizing the prerequisite knowledge required for each system, offering an out-of-the-box solution for situational awareness during ongoing missions, and streamlining the integration of additional systems into the mission environment through shared components of the communication infrastructure. This streamlined integration process necessitates the development of robotic-specific wrappers around individual subsystems, ensuring compatibility and interoperability across the entire robotic ensemble.

TABLE OF CONTENTS

979-8-3503-0462-6/24/$31.00 ©2024 IEEE

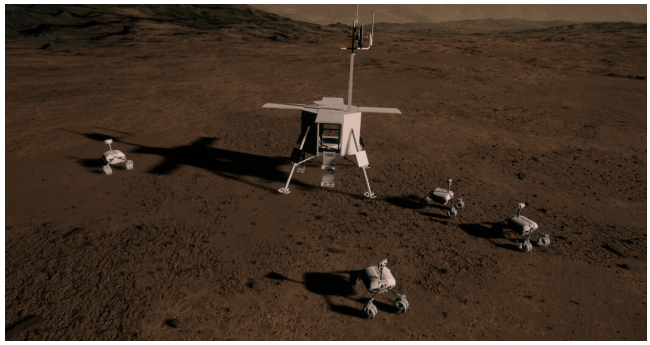


Figure 1: Multi-robot scenario on the Martian surface. A lander and a heterogeneous team of robotic systems with different capabilities are performing actions.

1. INTRODUCTION

Achieving the objectives of advancing human presence on the Moon and Mars presents one of the space community's most significant challenges. To address these challenges effectively, integrating robotic assistance is imperative, particularly in tasks related to habitat and infrastructure construction, as well as scientific endeavors. However, it is evident that a one-size-fits-all approach will not suffice. Therefore, the deployment of specialized systems becomes essential to establish a reliable and sustainable human presence on extraterrestrial surfaces.

The joint mission known as Surface Avatar, a collaborative effort between the German Aerospace Center (DLR) and the European Space Agency (ESA), extends the capabilities of astronauts aboard the International Space Station (ISS) by enabling them to command a diverse fleet of robots [1]. By employing a comprehensive strategy that combines scalable autonomy through multi-modal tele-operation and task-oriented commanding, this mission introduces innovative avenues for tele-robotics.

However, managing multiple distinct robots operating within the same mission poses a formidable software engineering challenge. In this work, we address the issue of coordinating a heterogeneous set of robots, each developed by different

 DOI: 10.1109/AERO58975.2024.10521167

teams, to ensure seamless interoperability. Our proposed approach for communication and data exchange within the mission encompasses the following key features:

- Implementation of a centralized concept that fosters awareness of the current robot configuration participating in the network.
- Adoption of a modular approach that maximizes code reusability and minimizes platform-specific implementations.
- Establishment of a system capable of broadcasting a robot's capabilities and progressively enabling new features based on the current software stack.

2. Related Work

Previous research in the field of multi-robot communication has primarily focused on achieving reliable message transmission from one robot to another [2]. Only recently, the research community has started to address the critical challenge of organizing and managing message flows within a group [3].

Early work by Tanner et al. [4], inspired by ornithological studies conducted by Helbig et al. [5], explored the behavioral dynamics of birds. This research delved into phenomena such as escape panic, in which groups trapped in large enclosed spaces seek escape routes and attempt to apply avian algorithms to particle systems.

Building upon this foundation, Meshabi et al. [6], [7] argued that effective data transmission across a growing network of participants requires control over the network itself. Carpin et al. [8] and Parker et al. [9] subsequently advanced the field by implementing distributed navigation and sensing capabilities, incorporating diverse sensors on each robot. However, comprehensive knowledge of the entire network was still a prerequisite for executing operations. Concurrently, Berkey et al. [10], drawing on the foundational work of Cao et al. [11] and Arkin [12], proposed a communication architecture enabling the broadcast of sensor capabilities across both centralized and decentralized networks.

In recent years, further research has begun to specialize in specific domains within the realm of robotics. Queralta et al. [13], for instance, emphasized the challenges posed by heterogeneous robotic teams in multi-robot competitions like ICARUS [14] and the DARIUS project [15]. These researchers have underscored the necessity for interoperability and have advocated for the development of a common framework for message communication [16].

However, the question of how to achieve interoperability in the context of a growing or evolving fleet remains unresolved and is still an ongoing research field. In this work, we propose a concept that tries to fulfill the guarantee of interoperability and further assist a growing and unknown number of network participants to effectively communicate and exchange system capabilities.

3. Surface Avatar

Starting with Kontur [17], [18], it successfully demonstrated reliable and dexterous control of a robotic arm from the ISS. The ability of controlling and steering a rover on the ground using a haptic feedback device mounted in the Columbus module was achieved by Analog 1 [19], [20], which effectively demonstrated tele-operation capabilities from space.

Figure 2: The humanoid and wheeled robotic system Rollin' Justin developed by DLR.

Taking it a step further, METERON SUPVIS Justin [21] presented the astronauts with the possibility of executing task-oriented commands on the system and supervising the operation of complex actions by the robot [22]. Following the acquired knowledge within the previous missions, Surface Avatar is targeted to explore the possibilities of commanding a heterogeneous fleet of systems. With this, it is achieved by combining the building blocks of tele-operation and task-oriented commanding while allowing an operator to select the best solution for the given situation.

While the first experiments only involve the humanoid, wheeled platform Rollin' Justin developed by DLR, the mission schedule foresees new robotic assets constantly added to the overall setup. As illustrated in Figure 2, Rollin' Justin is equipped with two robotic arms, enabling its capability to perform complex maintenance and manipulation tasks, including connecting specific objects to scientific instruments. In conjunction with sensors integrated into the joints of the system, applied torques are translated into force feedback that helps the crew perceive the remote environment surrounding the robot. Additionally, several cameras mounted to the head and the base platform provide multiple viewing angles that the operator can use to enable visual localization and navigation, obstacle avoidance, and reasoning about the world.

The second robot participating within the scope of Surface Avatar is the rover-like platform Interact. As illustrated in Figure 3, it is equipped with a robotic arm attached to the front face, thus possessing comparable capabilities to the Rollin' Justin mentioned above. In addition, an extra camera on a second arm is mounted on top of the system, enabling a nearly 360-degree view as well as a close-up of the workspace of the front-facing manipulator. Its design enables the inspection and manipulation of objects at lower heights, particularly on the ground.

Furthermore, a mock-up of a future landing system, to which a robotic arm [23] is attached, serves as a base station for the overall mission setup. Figure 4 depicts the lander arm assembly within the Surface Avatar setup. In its initial configuration, the lander acts as a distribution node for objects stowed inside the payload bay, such as instruments for scientific investigations. The robotic manipulator supports the transfer process to other approaching robots. Eventually, it can retrieve containers for sample-return missions and safely place them inside the lander. Moreover, the system features cameras that provide the crew with additional views.

At last, a quadruped system called BERT will be added in

Figure 3: The rover-like system Interact developed by ESA.

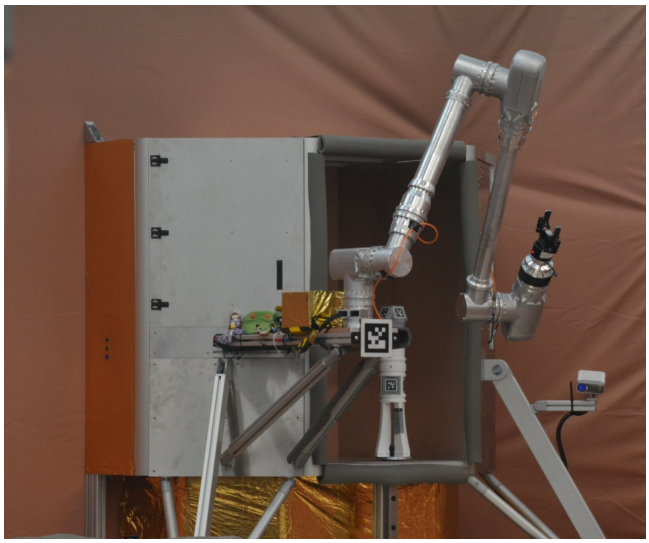


Figure 4: The lander mock-up and the attached arm.

the future to Surface Avatar. Due to its small size, BERT is capable of exploring cavities, and its unique locomotion system enables efficient movements across rough terrain. However, limited computational resources and a constrained power budget restrict the autonomous capabilities of this system, yet the system's ultimate characteristics still need to be finalized. BERT is shown in Figure 5.

Overall, the displayed platforms represent a broad spectrum of robotic capabilities and mechanical systems. A humanoid system with omni-directional drive, systems with limited degrees of freedom in their locomotion, as well as pure static systems, are included. Within the heterogeneous fleet, the workspaces of the manipulators differ, and grasping capabilities depend on the currently mounted end effector. Ultimately, the robots' initial operational periods range from almost 15 years in the past for the oldest system to a point in the future for a system presently under development. This results in different styles and concepts in the respective software stacks. Furthermore, each robot is developed by different teams, and not all solutions for specific problems are equal.

All of this underscores the need for a unified yet general way

Figure 5: The small, quadruped robot BERT currently in development at DLR.

to communicate between the systems, exchange data, and notify a human operator about the capabilities and properties of each system. Additionally, the method should be able to identify the current set of clients, their configuration, and capabilities, offering a way to easily address specific participants directly or indirectly by their capabilities. This will enable the development of submodules on each system independent of the final scenario, robot count, and configuration. Lastly, due to the nature of a research platform, it should consider constantly changing modules and interfaces.

4. Communication Backend

The presented approach describes a concept for communication that is independent of the actual framework or middleware used for transmitting data across system boundaries. However, we expect certain requirements to hold true for the used implementation:

- The framework must be able to address specific clients within the system using a unique identifier. The identifier must be persistent over time and customizable by the user.
- New clients must be able to connect to the network at any point and do not to be known or configured prior to the first connection. The framework must be able to reconfigure itself dynamically when a new client is added or clients no longer respond.
- It is expected that new channels can be opened between clients and data can be exchanged using requests. A request is a pair of two messages: the first sent from the requester to the receiver, the second sent from the receiver back to the requester after successful processing. Both can, but might not, contain a custom data payload.

A central communication node is integrated and is intended to be available in all scenarios. It is referred to as the central unit (CU), and it is the only network participant that has to be known by all robots. The main purpose of the CU is to manage the active communication between the participants in the network. Additionally, every robot is equipped with a client gateway that takes care of the network-related logic and acts as the translation layer between the robotic-specific implementation of specific subsystems of the robot and the common communication structure of the Surface Avatar network. Once the robot is registered in the network, it can send requests to specific clients or receive requests from others. The overall architecture is illustrated in Figure 6.

In the following, we will outline the core functionality that

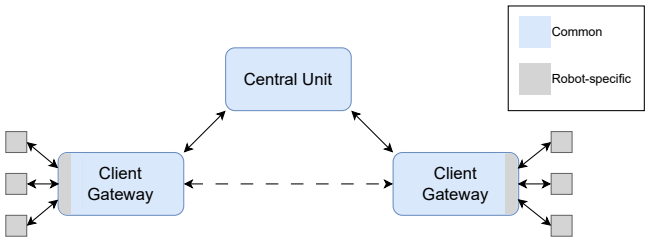


Figure 6: Schematic illustration of the centralized communication within the Surface Avatar network. Before any process can send or receive data, the client has to register itself at the central unit. Afterwards, requests can be sent using the common infrastructure in the network. Each gateway serves as the translation layer between the robot-specific implementations and handles network-related logic, such as the registration.

will be available in any configuration.

Register

Before any client can use the Surface Avatar network, it has to register itself at the CU, which involves transmitting its unique identifier to the administrative unit. If this is acknowledged positively, the client can start sending requests. In case of an error, for example, if the unique identifier is already registered, the registration is not considered successful, and participation in the network is not possible.

Get all connected

This is a request to the CU of the network, triggering a respond with all currently registered robotic assets. As a result, the client can use this list to obtain more detailed information about any specific client.

Get all loaded modules

In contrast to the previous functions, this request will be sent to a specific client in the network. It will respond with a list of all currently available modules on the associated system, as well as a string representing the version number of each module. The role of modules will be discussed in Section 5.

Unregister

In case a robot wants to leave the Surface Avatar network, it should properly unregister itself. The CU will then release the unique identifier and notify all participants that the client has unregistered. Furthermore, any ongoing request to the disconnecting client will be interrupted and rendered unsuccessful momentarily without waiting for a timeout.

As a higher-level instance, the CU will continuously check the connection to any registered client by sending a ping to each. If this ping is not acknowledged, it is assumed that the client in an error state and is expected to disconnect from the network inproperly. Thus, any ongoing request will be blocked until a timeout is reached to automatically cancel the operation.

A typical sequence of requests and network exchange is shown in Figure 7. First, only one system is registered in the network. Therefore, the CU will only return the name of the requesting client, in this case, Robot A. As soon as a second asset connects to the network, a request for getting all connected assets will return a list of multiple robots. This list can now be used for retrieving information about a specific

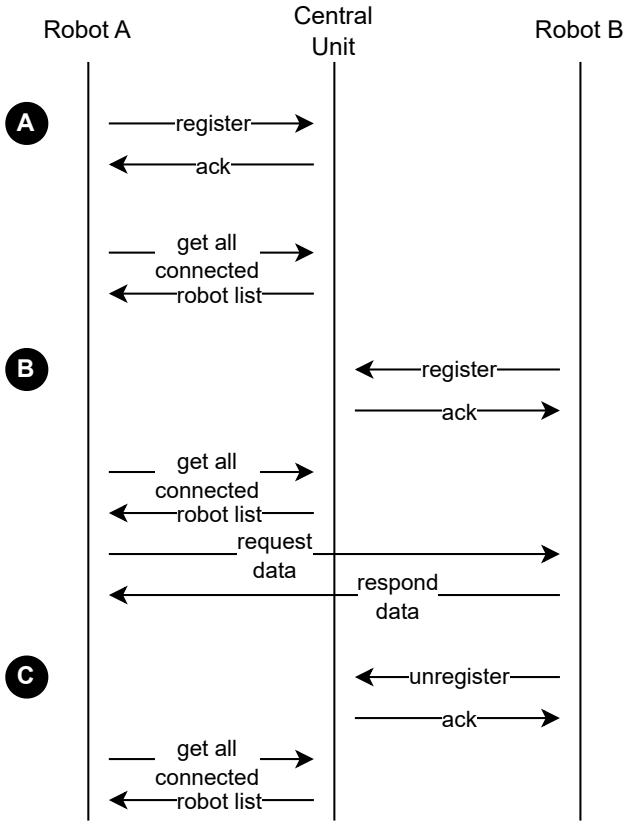


Figure 7: Example of message sequences. A) Only one robot is successfully connected to the network. The request for getting all connected assets will only return the requester himself. B) Two robots are connected to the Surface Avatar network. A request for getting all connected assets will return Robot A and Robot B. After this, the systems can start to exchange data. C) After properly disconnecting, only Robot A will be returned for the request to get all connected assets.

system and getting a list of all currently loaded modules. At the end, the system unregisters itself to leave the network.

5. MODULES

As mentioned in Section 4, the clients are organized into modules. While the core functionality is mandatory for any implementation, all other modules are optional. In general, each module summarizes logical functionalities and data providers that represent basic subsystems on a robot. For example, the battery module shall provide functions that describe if a system can be powered by a battery, how many separate units are supported, how many are connected, the status of each, and if they are currently loaded or not. A localization module would provide information about whether the system can currently localize itself, its position and orientation in 2D or 3D space, the uncertainty of the current estimate, and additional information if available.

Each module has to implement a function that returns the name of the module and a string representing the version of the implementation. The latter should be a valid semantic versioning (SemVer) representation as this allows easy tracking of compatibility between different versions. A three-number SemVer string is known as Major, Minor, and Patch. While

a change in the patch version only fixes small bugs that do not alter the behavior of the module, a change in the minor version adds new functionality. However, compatibility is preserved for any older version. A change in the major version notifies application programming interface (API)-breaking changes that are non-compatible.

The module-specific implementations describe the way two clients can exchange information in the Surface Avatar network. To achieve a generic approach, modules can provide properties and functions.

Properties

A property is a static function that will always return the same value. It shall describe how the robot can implement the current module. In case of the battery module, the overall module might offer properties such as *has_battery* and *has_battery_capacity*. This indicates if the system has the possibility to be battery-powered and whether the system can measure the current capacity.

Functions

A function is a dynamic method whose execution can return a changing value, alter the internal state of the robot, or anything else. It represents the actual functionality of the module. Using the example of the battery module, if the properties *has_battery* and *has_battery_capacity* are set to be true, the function *get_battery_capacity* returns a valid value and can be used to get insight into the current battery status. Otherwise, it should not be used as the behavior is undefined.

The above-mentioned implementations describe the network-wide functionality and exchange protocols and can be reused on any system. However, each robot will need a specific adapter to communicate with the local processes. While each module can be a wrapper for a single process, it is not limited to this. For instance, in the case of the localization module example, the value of a *get_accumulated_drift* function might trigger visual odometry (VO) and the wheel odometry (WO) systems and compare the trajectory estimates of both to get a measurement of the total drift and identify slip.

Lifecycle

A module includes a state machine that mirrors the lifecycle of a module and is under the control of the associated gateway client, as depicted in Figure 8. Upon startup, each module undergoes configuration by loading and parsing a YAML file. Once all modules have completed their configuration, the state machine transitions to the "loaded" state. Depending on the implementation, an automatic transition to the "running" state may occur, or it may be initiated by an external trigger. Subsequently, the module reaches the end of its lifecycle, prompting the gateway client to initiate the unloading procedure triggered by the "unregister" action. This final transition ensures the robot remains in a stable and safe condition, with no data loss, before the module is ultimately destroyed.

6. CAPABILITIES

One of the major driving factors for introducing this approach is the communication of capabilities of specific robot implementations. Design specifications of different subsystems are constantly changing, and with a growing number of robots in the mission, updates have to be pushed to several platforms at once to maintain a stable overall system. This ultimately leads to a significant overhead to maintain a stable state of the whole multi-robot organization.

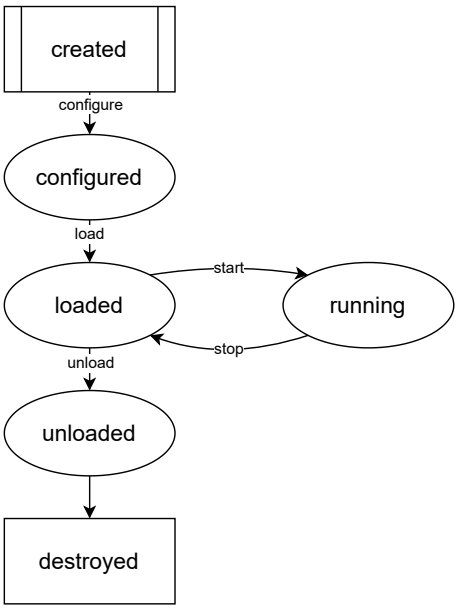


Figure 8: The state machine represents the life cycle of a module, starting with the configuration. Then, the module enters a configured state. Afterward, it transitions to the loaded state when the gateway client finishes configuring all modules. The module can toggle between the loaded and running states indefinitely. Eventually, the module is unloaded and can be destroyed.

Exploiting the communication backend implementation in Section 4, the software can easily check for compatibility between systems and prevent sending requests to incompatible robots. This is achieved by the ability to compile a list of connected robots and query for the loaded modules and their versions. Further, the properties of the individual modules described in Section 5 help to enable new functionalities only if they are supported by the current software stack and the general capabilities of the robot.

To illustrate the application of the method above, imagine a scenario in which a robot shall investigate a damaged surface on a habitation module. For this task, the robot has to approach the habitat closely and visually inspect the damage. As a minimum requirement, any robot in question for this task must implement a perception module that signals the existence of a camera with a suitable viewing-frustum. Further, it has to equipped with a locomotion system that enables an operator to remotely move the robot. Now, the astronaut is able to remotely position the system to manually inspect the habitat. Preferably, the system also implements a navigation module. At this point, the system can move autonomously to the final destination, reducing tedious work for the crew. In case the module further has the property *has_obstacle_avoidance*, the safety margin for moving the system closely to habitats can be reduced, as the system can prevent crashes due to localization errors.

7. EXTENDED EXAMPLE

In the following, we want to further elaborate on the functionality of the presented concept in more detail using two longer examples to showcase its potential and possibilities.

Mapping Service

The first example is depicted in Figure 9. It involves three distinct robots: the humanoid robot Rollin' Justin, the rover Interact, and the legged system BERT, each developed by different teams. With this, all three robots possess the ability to traverse terrain and are equipped with visual sensors.

Concurrently, an independent team is working on a mapping service that amalgamates visual data to construct a precise map of the environment. For this purpose, the service necessitates access to the video data, calibration information for each camera, global coordinates, and an assessment of measurement uncertainties.

At first, version *0.1.0* of the perception module provides the required camera data and information, while the localization module in the same version supplies the position details. Initially, only one system can communicate with the mapping service.

To integrate the second robot, Interact, into the mapping service, the perception module is the only component that needs to be incorporated. This enables the mapping service to access all essential information without necessitating any further robot-specific integration.

However, due to limited computational resources on the small BERT platform, obtaining global position data, including uncertainty measurements, is not feasible. To address the challenge of incorporating the third system, an update of the service can be implemented. The onboard computer of BERT can detect artificial landmarks known as fiducials. Additionally, for short distances, the trajectory can be estimated using leg odometry and a linear uncertainty model. Using the data of both, the service is able to extend the map.

The revised mapping service implementation now requires camera data and information, along with either position information or fiducial and odometry data. Importantly, this update to the localization module remains compatible with previous versions, obviating the need for adjustments in the other robots' implementations.

Capabilities for Task Execution

Secondly, broadcasting of capabilities can also serve to assist human operators in selecting an appropriate robotic system for a specific task.

One recurring task within the protocols of Surface Avatar involves Rollin' Justin to conenct a plug to a socket in front of him. However, the successful execution of this task can be hampered by many factors including inaccuracies in the localization of the system, the socket, or the robot's hand. In such cases, the robot initiates an abort of the action and requests assistance from the crew. The crew can then employ tele-operation to manually connect the socket, aided by a 2D video stream from the robot and force-feedback mechanisms. Nonetheless, this task remains complex due to imperfections in perceiving the actual environment.

To enhance the operator's perspective, an additional system can be requested to provide an alternative side view, offering depth perception for inserting the connector into the socket. By querying all systems equipped with locomotion, navigation, and perception capabilities, the crew can access a list of suitable robots for this specific task. This approach expedites the task completion process and reduces the astronaut's need for extensive knowledge about the mission setup, the robots, and their capabilities.

8. Conclusion

This work introduces a communication framework designed for incorporating a diverse fleet of robots. The approach facilitates automatic discovery of all participating robots. Notably, the proposed system offers a significant advantage by broadcasting the capabilities of each individual robot. This empowers each client to request a list of all robots or a specific subset that matches a predefined set of requirements for communication and data exchange.

Furthermore, the modularization of the entire interface simplifies the development process. By encapsulating the network architecture and robot infrastructure, the framework minimizes the overhead required for implementing network logic on each system, preserving the original software stack.

In the end, we have successfully demonstrated the functionality of our concept in a small-scale simulated environment and intend to integrate the framework into upcoming space-to-ground experiments involving all participating robotic systems.

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 (Col-CC), and the European Astronaut Training Centre (EAC). We thank them for the support during experiment preparation, testing, and astronaut training.

This work is partly supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the project "SMiLE2gether" (LABAY102).

References

[1] Lii, N. Y. et al., "Introduction to Surface Avatar: the First Heterogeneous Robotic Team to be Commanded with Scalable Autonomy from the ISS," in *Proceedings of the 73rd International Astronautical Congress (IAC)*, 2022.

[2] R. Doriya et al., "A brief survey and analysis of multi-robot communication and coordination," in *International conference on computing, communication & automation*. IEEE, 2015, pp. 1014–1021.

[3] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*. IEEE, 2012, pp. 1–5.

[4] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic control*, vol. 52, no. 5, pp. 863–868, 2007.

[5] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, 2000.

[6] M. Mesbahi, "State-dependent graphs," in *42nd IEEE International Conference on Decision and Control*

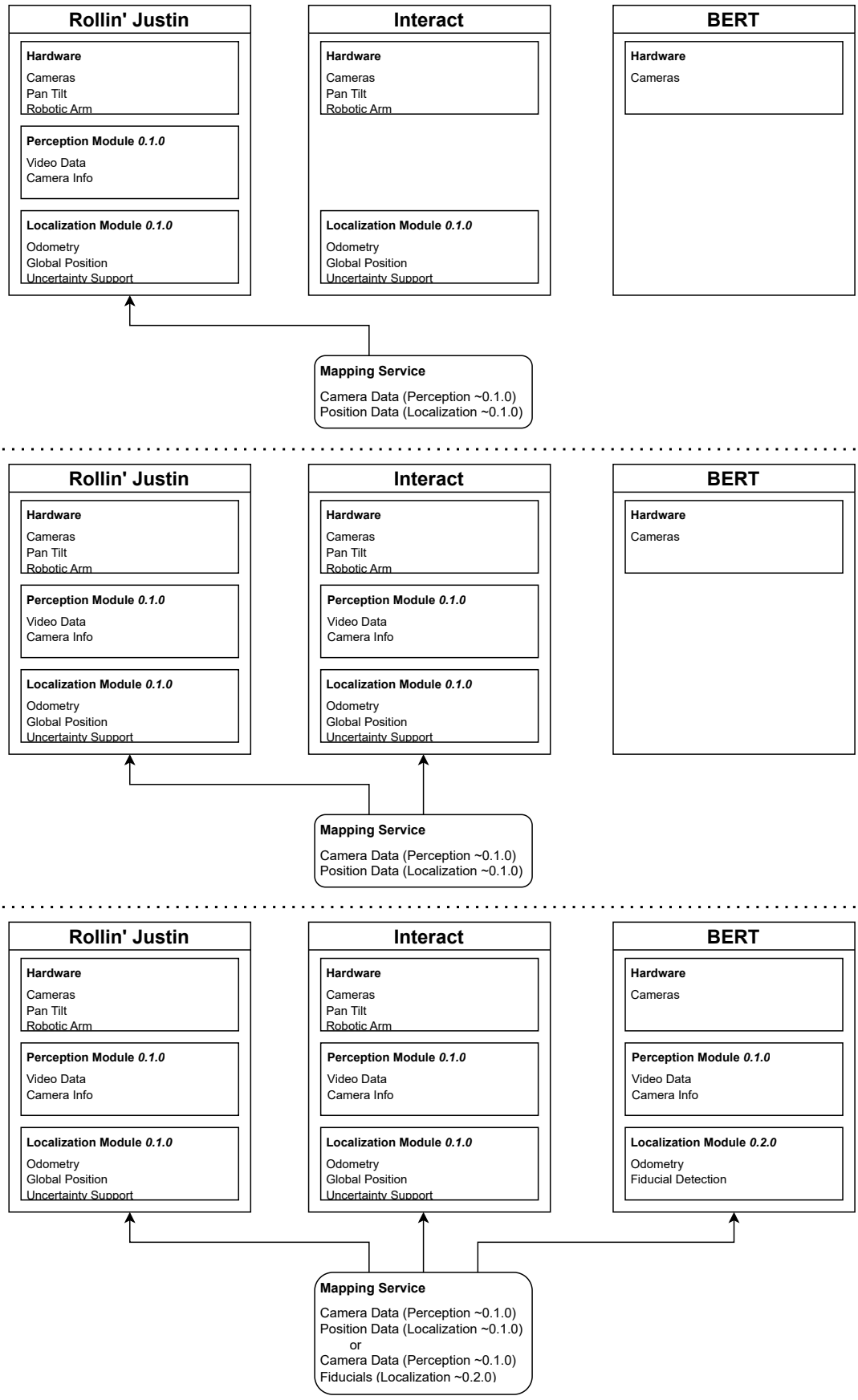


Figure 9: Exemplary steps of an ongoing research of an external mapping service. Out-of-the-box, only one system can provide all necessary functionalities to be incorporated into the service. However, small adjustments add the second system to the service network. The last robot needs an additional approach due to its computational limitations. This can be achieved without compromising interoperability with the previous systems.

(IEEE Cat. No. 03CH37475), vol. 3. IEEE, 2003, pp. 3058–3063.

[7] ——, "On state-dependent dynamic graphs and their controllability properties," *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 387–392, 2005.

[8] S. Carpin and L. E. Parker, "Cooperative leader following in a distributed multi-robot system," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2994–3001.

[9] L. E. Parker et al., "Tightly-coupled navigation assistance in heterogeneous multi-robot teams," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 1016–1022.

[10] G. A. Bekey and A. Robots, "From biological inspiration to implementation and control," 2005.

[11] Y. U. Cao, A. B. Kahng, and A. S. Fukunaga, "Cooperative mobile robotics: Antecedents and directions," *Robot colonies*, pp. 7–27, 1997.

[12] R. C. Arkin, *Behavior-based robotics*. MIT press, 1998.

[13] J.P. Queralta et al., "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *Ieee Access*, vol. 8, pp. 191 617–191 643, 2020.

[14] D. Serrano et al., "Icarus and darius approaches towards interoperability," in *IARP RISE Workshop, At Lisbon, Portugal. Proceedings of the NATO STO Lecture Series SCI-271*, vol. 1, 2015.

[15] G. D. Cubber et al., "Introduction to the use of robotic tools for search and rescue," *Search and Rescue Robotics*.

[16] P. Chrobocinski et al., "Darius project: Deployable sar integrated chain with unmanned systems," in *2012 International Conference on Telecommunications and Multimedia (TEMU)*. IEEE, 2012, pp. 220–226.

[17] C. Riecke et al., "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.

[18] J. Artigas et al., "Kontur-2: Force-feedback teleoperation from the international space station," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[19] T. Krueger et al., "Designing and testing a robotic avatar for space-to-ground teleoperation: the developers' insights," in *Proceedings of the 71st International Astronautical Congress (IAC)*, 2020.

[20] W. Carey et al., "Meteron analog-1: A touch remote," in *Proceedings of the 73rd International Astronautical Congress (IAC)*, 2022.

[21] Lii, N. Y. et al., "Toward scalable intuitive teleoperation of robots for space deployment with the meteron supvis justin experiment," in *Symposium on Advanced Space Technologies for Robotics and Automation*, 2017.

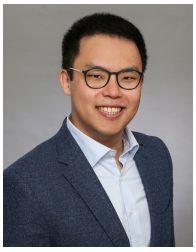
[22] Schmaus et al., "Extending the knowledge driven approach for scalable autonomy teleoperation of a robotic avatar," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–10.

[23] Maier et al., "Tina: small torque controlled robotic arm for exploration and small satellites," in *Proceedings of the International Astronautical Congress, IAC*, 2019.

BIOGRAPHY

Marco Sewtz *is the lead for software and interfaces of the new lunar exploration rover at the German Aerospace Center (DLR). He received his B.Eng. at the University of Applied Sciences of Munich and his M.Sc. at the Technical University of Munich. His interests focuses on SLAM and multi-modal perception of the environment. Before his current role, he worked as an electrical designer for high-performance processing modules for space hardware at Airbus Defence and Space.*

Florian S. Lay *received his B.Sc. 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.*

Thibaud Chupin *received his M.Sc. in Automation & Controls Engineering at the Politecnico di Milano. He works at the ESA Human Robot Interaction Lab in Noordwijk, Netherlands. He maintains and develops the robot control stack for the Interact rover. Before his current role, he worked as a Robotics Engineer designing a Lunar rover for future exploration missions.*

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 B.Sc., M.Sc., 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 tele-robotic experiments Surface Avatar and METERON SUPVIS Justin. Neal is primarily interested in the*

use of tele-robotics in both space and terrestrial applications.