VR-OOS: The DLR's Virtual Reality Simulator for Telerobotic On-Orbit Servicing With Haptic Feedback

Mikel Sagardia, Katharina Hertkorn,
Thomas Hulin, Simon Schätzle
German Aerospace Center (DLR)
Muenchener Str. 20,
82234 Wessling, Germany
+49 8153 28-1039
Mikel.Sagardia@dlr.de

Robin Wolff, Johannes Hummel,
Janki Dodiya, Andreas Gerndt
German Aerospace Center (DLR)
Lilienthalplatz 7
38108 Braunschweig, Germany
+49 531 295-2970
Robin.Wolff@dlr.de

VR-OOS: The DLR's Virtual Reality Simulator for Telerobotic On-Orbit Servicing With Haptic Feedback

Mikel Sagardia, Katharina Hertkorn, Thomas Hulin, Simon Schätzle
German Aerospace Center (DLR)
Muenchener Str. 20,
82234 Wessling, Germany
+49 8153 28-1039
Mikel.Sagardia@dlr.de

Robin Wolff, Johannes Hummel, Janki Dodiya, Andreas Gerndt
German Aerospace Center (DLR)
Lilienthalplatz 7
38108 Braunschweig, Germany
+49 531 295-2970
Robin.Wolff@dlr.de

Abstract— **The growth of space debris is becoming a severe issue that urgently requires mitigation measures based on maintenance, repair, and de-orbiting technologies. Such on-orbit servicing (OOS) missions, however, are delicate and expensive. Virtual Reality (VR) enables the simulation and training in a flexible and safe environment, and hence has the potential to drastically reduce costs and time, while increasing the success rate of future OOS missions. This paper presents a highly immersive VR system with which satellite maintenance procedures can be simulated interactively using visual and haptic feedback. The system can be used for verification and training purposes for human and robot systems interacting in space. Our framework combines unique realistic virtual reality simulation engines with advanced immersive interaction devices. The DLR bimanual haptic device HUG is used as the main user interface. The HUG is equipped with two light-weight robot arms and is able to provide realistic haptic feedback on both human arms. Additional devices provide vibrotactile and electrotactile feedback at the elbow and the fingertips. A particularity of the realtime simulation is the fusion of the Bullet physics engine with our haptic rendering algorithm, which is an enhanced version of the Voxmap-Pointshell Algorithm. Our haptic rendering engine supports multiple objects in the scene and is able to compute collisions for each of them within 1 msec, enabling realistic virtual manipulation tasks even for stiff collision configurations. The visualization engine ViSTA is used during the simulation to achieve photo-realistic effects, increasing the immersion. In order to provide a realistic experience at interactive frame rates, we developed a distributed system architecture, where the load of computing the physics simulation, haptic feedback and visualization of a complex scene is transferred to dedicated machines. The implementations are presented in detail and the performance of the overall system is validated. Additionally, a preliminary user study in which the virtual system is compared to a physical test bed shows the suitability of the VR-OOS framework.**

TABLE OF CONTENTS

978-1-4799-5380-6/15/$31.00

1. INTRODUCTION

Space systems like satellites are designed to operate for a predefined mission period. Once on orbit, the system hardware cannot be modified anymore. In opposition to other technical systems like airplanes, automobiles, trains, etc., failed sub-systems cannot be repaired after launch. For this reason, verification and validation over the development lifecycle of a spacecraft is extremely important. Moreover, only space certified sub-systems are allowed in order to ensure a maximum quality and robustness. Nevertheless, it cannot be excluded that sub-systems fail. Therefore, all essential sub-systems are redundant to provide a backup in case of malfunctions. If those fallback solutions do not work as well, the whole mission could be in jeopardy.

In the past, spacecraft servicing missions were carried out just for very prestigious or expensive space systems which worked unsatisfactorily. The most famous cases are the Hubble space telescope (HST) servicing missions. NASA space shuttles brought astronauts to the telescope on orbit to set the HST in operation after the replacement of an incorrectly shaped mirror and to expand the mission by installing technically advanced instruments and devices [1]. Repairing, maintenance, and extensions are the main incentives for servicing missions [2]. But past missions were extremely expensive. And an extravehicular activity (EVA) of an astronaut is very perilous. Reducing costs and risks are the main driving factors for research on robotic on-orbit servicing (OOS) approaches.

Space systems are often unique copies, and suddenly appearing malfunctions may merely be analyzed approximately on Earth by the responsible engineers. Similarly, unexpected complications may occur for standard maintenance activities like device replacements or refueling operations. Therefore, purely autonomous robotic servicing missions do not seem to be reasonable. Instead, the most promising approach is telerobotics. In this case, the service engineer controls a service robot remotely in order to perform the maintenance tasks. According to the robot camera images and the on-site findings, the engineer can adapt the foreseeable tasks.

However, satellite systems in use nowadays are not designed for service operations. It is also still an open question how service satellites including robotic arms have to be specified so that those systems can flexibly be used for several classes of spacecraft systems. To elaborate a matching couple of maintainable satellites and service satellites, a physical mock-up could be built up in research laboratories. However, especially when variants of solutions have to be investigated, the physical approach can become very expensive and inflexible. Here, virtual workspaces come into play. Such

(a) (b) (c)

Figure 1. (a) The DLR's humanoid SpaceJustin designed for researching on on-orbit servicing (OOS) telerobotic maintenance; (b) The DLR's bimanual haptic interface HUG for telepresence and virtual reality applications; (c) The VR-OOS simulator on an immersive projection-based display for simulating and training telerobotic OOS tasks.

environments cannot replace physical tests completely but can help to adapt current blueprints much faster in order to iterate to feasible solutions. Those can then be constructed as physical prototypes for final evaluations.

The goal of the project depicted in this paper is to specify and evaluate appropriate interfaces between engineers and immersive virtual environments (IVE). It is crucial that the virtual scene appears as realistic as possible. Otherwise, hypotheses cannot be proved adequately and findings could be wrong. The closeness to reality requirement does not only address the visualization. Indeed, in most cases, just a plausible visualization of the virtual scenario may already be sufficient. Much more important but also considerably more challenging seems to be the interaction interface (see Fig. 1). Virtual objects for assembly simulation have to behave physically correctly. Therefore, collision detection has to be fast and accurate. An efficient physics engine has to manage multibody motion reliably. And finally, a sufficient force feedback is essential to emulate a realistic service environment to the engineer. In this line, it turns out that haptic force feedback is superior to pure visual feedback in terms of movement precision, mental workload, and spatial orientation [3].

In this paper, we demonstrate our approaches to implement immersive virtual workspaces for interactive satellite design and remote servicing training. After a survey about related work, the fundamental system architecture of the virtual reality simulator is described in Section 3. The different interaction devices incorporated and techniques developed are explained in detail in Section 4. To demonstrate the effectiveness of our approaches, a use case scenario has been developed and is described in Section 5. After depicting the experimental results in Section 6, we conclude the main points of the current status of our project and point out possible next research steps.

2. Related Work

The potential of virtual reality (VR) technology for training and simulation of on-orbit servicing tasks has long been recognized. For instance, a virtual environment was used for the training of EVA servicing activities for the Hubble space telescope (HST) servicing mission in 1993 [4]. The system included graphical representations of the HST, the space shuttle cargo bay, as well as maintenance hardware. It did not simulate the dynamic behavior of the objects and did not include force feedback. Force feedback was integrated in a similar VR simulation for astronaut training in preparation for EVA on space shuttle missions [5]. This system also included the dynamic simulation of the objects in zero-gravity environments as well as the shuttle's remote manipulator system, and allowed payloads being manipulated. A VR simulator for astronaut training using force feedback was presented in [6]. An exoskeleton in addition to a data glove was used as input and force feedback device. The software framework included collision detection, force computation and a dynamics module. The system focused on the task of grasping a box using a handle, extracting it from a slot and inserting it into another slot. A more recent VR simulator for astronaut training used a data glove in combination with a CyberGrasp for force feedback, a head-mounted display and motion tracking, as well as physics simulation [7]. The system was used to train astronauts for space walks, including climbing along a handrail, and load retrieval.

Compared to these systems, our VR-OOS approach is unique, as it sensibly combines novel sophisticated software and hardware components for visualization, interaction, collision detection, and tactile and force feedback. The technologies provide a high level of immersion to the operator through multi-modal feedback and can be used to verify designs, train astronauts, or teach robotic systems to interact appropriately in space environments. VR-OOS also allows for several use case scenarios with multiple moving objects that are able to interact with each other. An exemplary future mission which may benefit from the VR-OOS system is the German orbital servicing mission DEOS [8], [9]. The goal of this mission is to demonstrate advanced maintenance tasks on a satellite by using a torque-controlled robot [10]. Such a robot is equipped with torque-sensors that precisely measure the interaction forces between the robot and the satellite [11]. This advantageous property allows for haptically teleoperating the robot in space, which means that a human operator commands the movements of the robots while perceiving and controlling the interaction force that occurs in space.

This mission exploits DLR's funded experience in torque-controlled robots of more than two decades, since the first light-weight robot was built in 1991. This robot technology has shown its suitability for space during the ROKVISS mission launched in December 2004, in which a robot arm was teleoperated from ground including force-feedback [12]. Moreover, DLR could already successfully prove that their haptic telepresent control concepts work in a real-world scenario in which communication delays of more than half a second occur [13]. The concept also works for complex humanoid service robots such as DLR's SpaceJustin (see

Fig. 1a), which may be used in a similar form for OOS tasks.

3. THE INTERACTIVE VIRTUAL REALITY ENVIRONMENT

Our virtual reality (VR) simulator is designed to enable real-time digital mockup interaction for the simulation and training of typical on-orbit servicing tasks. It targets immersive VR environments, e.g., using large projection-based 3D displays or head mounted displays (HMD) with head-tracked stereo vision, as well as various interaction devices for the user to interact with the virtual world. Fig. 2 shows a block diagram of the simplified system architecture. It consists of four core components. The visualization component provides photo-realistic real-time rendering of the scene from a user's viewpoint, including the satellite's components, tools and the space environment. It reads user tracking data x_{tracking}, such as head tracking or tracking of body parts, and passes it on to the framework as user pose x_{user}. Similarly, the collision and force computation component reads the pose of the user's hand x_{hd} and calculates the respective force F_{hd} for providing haptic feedback to the user when collisions occur during interactions with virtual objects. This force is also used as F_{user} to move manipulated objects in the movement simulation. The movement simulation computes the dynamic and kinematic behavior of the virtual objects based on user input, interactions with other virtual objects, and physical constraints. All three components above base their simulations on predefined models that describe object properties, such as geometry, material and mass, as well as constraints, such as the maximum angle a joint can rotate. Additionally, there is a state machine and logic control component that steers the conditional behavior of virtual objects during an assembly task, such as switching on lamps or starting the motor on an electric screwdriver. It monitors the data flow between the other components and changes object states accordingly. Interaction devices include not only haptic devices, but also displays, tracking systems, and other interfaces, as well as various interaction techniques that allow an efficient use of them.

A VR simulator has strict requirements for the accurate simulation of the dynamic environment at interactive rates. Typical use case scenarios can be complex and demanding in computation. In order to fulfill the real-time requirement, our framework uses a distributed architecture. This way, force computation, movement simulation, and visualization can run on dedicated machines. In addition, our framework abstracts each component to be stand-alone with interfaces for communication, data management, and simulation processes. Internally, each component manages its own representation of the simulation with its optimized algorithms and data structures. Changes of input parameters and simulation results are synchronized across the distributed components by a network communication layer and a common message format.

The advantage of such a generic abstraction is that it provides modularity and improves the flexibility of the system. It allows an easier implementation of alternative components or inclusion of existing simulation systems and supports an uncomplicated exchange of them. A disadvantage of using a distributed system architecture is that the required network communication for synchronizing the distributed simulations induces delay and other negative effects due to network characteristics, such as jitter and packet loss. Our framework addresses this with a number of mechanisms for reducing latency and maintaining consistency across the distributed simulations. These include advanced message queuing, separation between reliable, and non-reliable messages and low-overhead communication protocols, as detailed in [14]. We report the end-to-end latency for transmitting synchronization messages across the distributed simulation system in Section 6.

State Machine and Logic Control

The logic component manages the states and events that are not covered by the movement simulation. This includes monitoring of states, for example, the *on*/*off* state of a virtual switch, and triggering behavioral actions based on predefined conditions, such as turning on a lamp when the state of a virtual switch is *on*. It also includes the dynamic adjustment of parameters for force computation or movement simulation, such as stiffness or degrees of freedom in mechanical constraints. This makes the simulation of complex mechanics, like the snapping mechanism of cable connectors, much simpler.

Additionally, the logic component controls the data stream between the distributed simulation components, checks the validity and fulfillment of constraints, and can adjust data if necessary before passing it on. For example, when the force computation delivers a force that is above the maximum allowed value, then the logic component can decide to cap the force before passing it to the haptic device.

Visualization

Apart from training the correct sequence of sub-tasks, a goal of the simulation environment is to allow the user to get an awareness of the arrangement and appearance of parts and tools. Hence, a realistic and high-quality visualization of the satellite components and the space environment are important factors for the success of a training simulation. This not only includes the photo-realistic rendering of detailed virtual objects with correct shading and high-resolution textures, but also the correct representation of the environmental effects that exist in orbit, such as bright sunlight, hard shadows, and a moving earth in the background.

Being a training and analysis tool, another research aspect of the visualization component is to augment the photo-realistic visualization of the virtual scene with the information-based, non-photo-realistic visualization of scientific data. Examples are the display of collisions between the robot and satellite parts, or the visualization of possible motion paths to avoid collisions. Additionally, hints on the order of servicing sub-tasks or other instructions could be overlaid on top of satellite parts.

Utilizing the modular system architecture, we concurrently developed three independent implementations of the visualization component, each with its own specific advantages. Each of the visualization implementations organizes the virtual objects in a scene graph and continuously synchronizes it with state updates from the movement simulation. One implementation is based on InstantReality [15], a VR development framework based on X3D that integrates well with the existing robot simulation environment at DLR. Another is based on the ViSTA VR-toolkit [16], a C++ framework for the development of VR applications, used to implement advanced rendering techniques. A screenshot is shown in Fig. 3. The visualization of realistic material effects of satellite components has been realized through a number of shader programs using the OpenGL shading language (GLSL). The

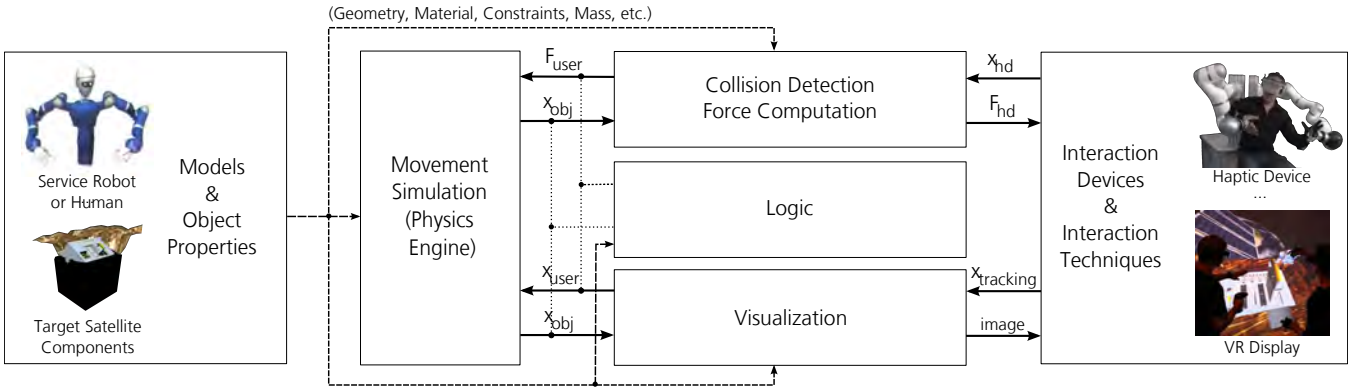


Figure 2. Simplified system architecture with core components and data flow. Data is passed between the interaction devices and core components. Additionally, the logic component monitors the data flow and allows one to change object states for steering conditional object behavior. The simulations in the core components are based on predefined models and object properties.

satellite's multi-layer foil (MLI), for example, uses an environment mapping shader for creating the metallic reflection effect. Other parts use their own specific shaders to simulate the visual effects of their particular surface materials.

The space environment draws stars at their known positions extracted from ESA's Tycho-1 star catalog with approximately 1 million entries. The atmospheric scattering effect of Earth in the background is based on a clear sky atmospheric model and calculates the scattering integral along a ray through the scene. Our algorithm uses precomputed scattering tables in order to achieve high frame rates [17].

All the above mentioned rendering techniques approximate the effects of light and reduce accuracy in favor of short computation times. We are currently developing a parallel and distributed real-time ray-tracing system based on NVIDIA OptiX[2]. Ray-tracing is an image generation method that creates physically more accurate images by tracing the path of light through the scene and considering some of the physical properties of object materials. Our ray tracing method allows high quality rendering of light effects, such as reflection, refraction, and shadows; however, it is significantly more computationally intensive. It splits an image into smaller parts and computes these in parallel on a cluster of 12 graphics processing units (GPU). With this prototype, we achieve a performance of up to 15 frames per second in high-definition resolution (1920x1080 pixels).

Collision Computation: Data Structures

In order to perform virtual manipulation tasks with haptic feedback, it is essential to employ a haptic rendering algorithm. These algorithms detect the overlap between colliding objects and compute a collision force that can be displayed to the user. Several approaches have been presented in recent years, but the field of collision feedback is still an ongoing research topic. Some well known methods work with simplified convex hull representations [18] or build hierarchical models [19] in order to achieve the challenging update rates demanded by haptic applications. A thorough survey about different haptic rendering methods is given in [20]. The haptic rendering algorithm used in our framework is an improved reimplementation of the Voxmap-Pointshell (VPS) Algorithm, initially presented by McNeely et al. [21], [22] and improved by Barbič and James [23]. This algorithm computes collision forces and torques of complex geometries with an update rate of 1 kHz. To achieve such a high computation frequency, two types of data structures are precomputed for each colliding object-pair: voxelmaps and pointshells (see Fig. 4).

[2]NVIDIA OptiX - http://www.nvidia.com/object/optix.html

Figure 3. Real-time visualization of a satellite and space environment realized through OpenGL shader language.

Voxelmaps are 3D grids in which each voxel stores a discrete distance value $v \in \mathbb{Z}$ to the surface. Voxels on the surface layer have $v = 0$, voxels in the k^{th} inner layer $v = k$, and voxels in the k^{th} outer layer have $v = -k$. Additionally, the scalar voxelmap function $V(\mathbf{P})$ yields the signed distance value of a point $\mathbf{P}$ in the voxelmap (see Fig. 5(a)). Pointshells are sets of points uniformly distributed on the surface of the object; each point $\mathbf{P}_i$ has an inwards pointing normal vector $\mathbf{n_i}$ (see Fig. 5(b) and Eq. (1)).

Point-sphere hierarchies are built down-top using the points of the plain pointshells. As shown in Fig. 4(c), the K closest points are clustered together and the point in the cluster closest to its center of mass is selected to be the cluster parent point. When all points of the leaf level are clustered, the clustering process for the next level is started using the parent points. This process is repeated until only one point describes the whole object. Next, for each cluster of each level, a minimally bounding sphere [24] which recursively contains all cluster children points is computed, as shown in Fig. 4(d). Barbič and James [23] scatter points top-down and build a similar structure, targeting simulations with deformable

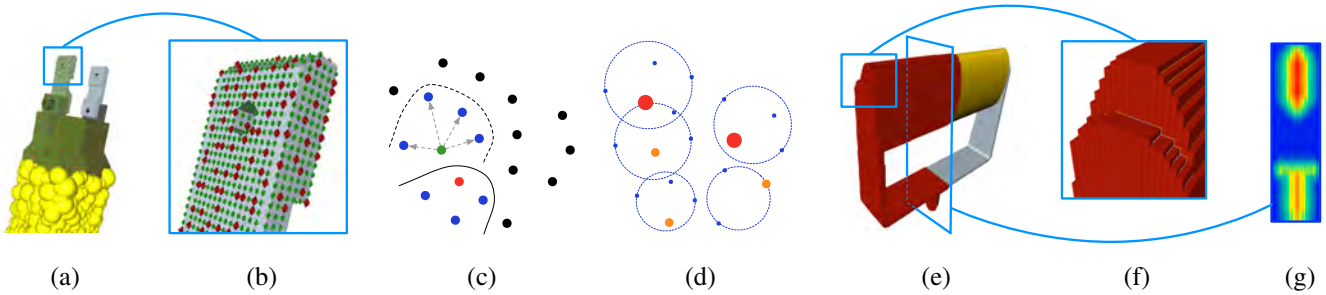


Figure 4. Point sampled and voxelized representations of a virtual gripper and a handle. (a) Point-sphere tree of the gripper: one sphere level in yellow and two successive point levels; (b) Close up of the two last point levels: the red set contains $K = 4$ times more points; (c) Clustering of the point-sphere hierarchy: K closest points are ordered into clusters ($K = 4$ in this case); (d) Minimally bounding spheres that contain all the children points of a cluster; (e) Half-voxelized handle; (f) Close up of the voxelized handle. (g) Transversal section of the voxelized handle: distance (turquoise-blue) and penetration (yellow-red) values are computed with the $V(\mathbf{P})$ signed distance function in Eq. (1).

objects. In contrast, our approach uses minimally bounding spheres in the clusters, and the algorithm is optimized for fast and accurate collision detection between rigid bodies. In this work, we used the fast and accurate voxelmap and pointshell generator presented by [25].

Collision Computation: Distances and Penalty Forces

In this section, we explain the computation of distance or penetration values and the calculation of penalty force values based on the data structures described in the previous paragraphs. In order to understand which points are selected for this computation see the next section.

The total collision force and torque ($\mathbf{f}_{\text{tot}}$, $\mathbf{t}_{\text{tot}}$) for each colliding object-pair is computed as the sum of all collision forces and torques ($\mathbf{f}_i$, $\mathbf{t}_i$) generated by colliding points $\mathbf{P}_i$. Points are colliding if their voxel value $v \geq 0$. Then, their normal vectors $\mathbf{n}_i$ are weighted by their penetration in the voxelmap $V(\mathbf{P}_i)$ resulting in the collision force $\mathbf{f}_i$. Torques $\mathbf{t}_i$ generated by colliding points are the cross product between forces $\mathbf{f}_i$ and point coordinates $\mathbf{P}_i$, all magnitudes expressed in the pointshell frame that has its origin in the center of mass. The computation of the single forces is summarized in Fig. 5 and Eq. (1):

$$\begin{aligned} \mathbf{f}_i = V(\mathbf{P}_i)\mathbf{n}_i = (\underbrace{v(\mathbf{P}_i)s}_{\text{global}} + \underbrace{\mathbf{n_i e}_i}_{\text{local}})\mathbf{n}_i, \\ \mathbf{t}_i = \mathbf{P}_i \times \mathbf{f}_i, \\ \mathbf{f}_{\text{tot}} = \sum_i \mathbf{f}_i,\ \mathbf{t}_{\text{tot}} = \sum_i \mathbf{t}_i \ \ \{\forall i \mid V(\mathbf{P}_i) \geq \mathbf{0}\}. \end{aligned} \tag{1}$$

The signed distance function $V(\mathbf{P}_i)$ of the voxelmap is split into two components: global and local distances (or penetrations). The global distance or penetration $v(\mathbf{P}_i)s$ is the value v of the voxel layer in which the point is multiplied by the size of the voxel s; thus, it indicates the approximative penetration of the point in the voxelmap. The local distance or penetration $\mathbf{n}_i\mathbf{e}_i$ is the projection of the vector between the pointshell point and the voxel center $\mathbf{e_i}$ on the normal vector of the point; hence, it denotes the depth of the point within the voxel.

Collision Computation: Hierarchy Traverse

The computation time of the algorithm depends on the number of points of the pointshell that has to be checked for collision, whereas the voxelmap resolution (i.e., the voxel size) affects only the quality of the force magnitudes [26]. Ideally, higher pointshell and voxelmap resolutions should be used only in likely colliding areas, which is realized using the point-sphere hierarchies. Note that increasing the resolution augments the required memory space quadratically in the case of the pointshell and cubically in the case of the voxelmap.

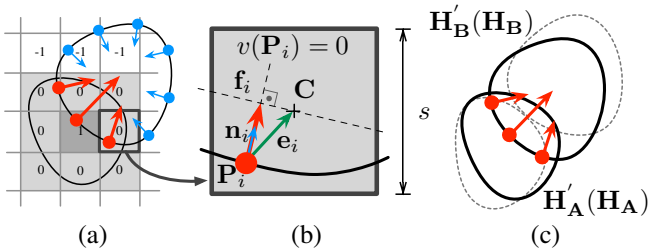


Figure 5. (a) Voxelized and point-sampled objects in collision. Each voxel has its voxel layer value v related to its penetration in the voxelmap, and each point $\mathbf{P}_i$ its inwards pointing normal vector $\mathbf{n}_i$. (b) The single point force $\mathbf{f}_i$ can be computed scaling the normal vector using its penetration with the penalty-based force computation approach. The cross products of forces and points yield torques. (c) Representation of the contact manifold $\{\mathbf{P}_i,\mathbf{n}_i,V(\mathbf{P}_i)\}$. The physics engine can then compute the motion that separates the objects from collision ($\mathbf{H}' \leftarrow \mathbf{H}$).

Fig. 6 summarizes the hierarchy traverse which is called once every cycle. At the beginning of each cycle, the uppermost cluster, which has the sphere that encloses all points in the pointshell, is pushed to a FIFO-queue. Then, the clusters of the queue are iteratively popped in breadth-first manner. In case the popped cluster sphere is colliding, the parent point of the cluster is checked for collision and the children clusters are pushed to the queue. If a leaf cluster is processed, all points in the cluster are checked, not only the parent point. Whenever a point is colliding, its single collision force is added to the sum yielding the total force, as explained in the previous subsection. A detailed explanation of the collision computation with hierarchy traverse can be found in [27].

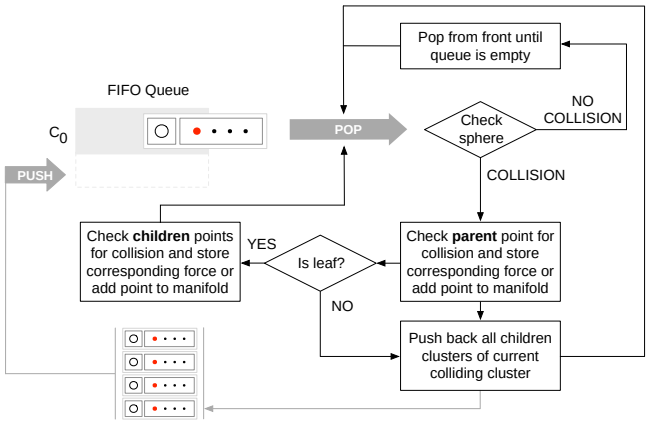


Figure 6. Breadth-first hierarchical traverse of the point-sphere tree. The FIFO queue is initialized with the root cluster c_1 that contains all points. Each cluster is defined with a minimally bounding sphere (circle), a parent point (red), K cluster points, and the addresses to parent and children clusters.

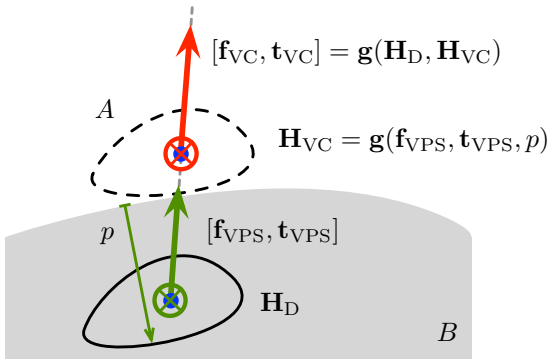


Figure 7. Virtual coupling used to enhance the user's collision perception. Given that the VPS algorithm is a penalty based method, there must occur interpenetration in order to compute a collision force. Forces and torques from the VPS are used to separate the interpenetrating objects (moving them to the dashed configuration) while displaying hard contacts on the surface.

Collision Computation: Force Scaling and Virtual Coupling

Forces and torques generated according to Eq. (1) need to be scaled due to the fact that their stiffness strongly depends on the number of points that collide, and therefore, on the pointshell resolution that is used. In order to overcome these effects, the empirical correction factor λ defined in Eq. (2) is used to scale the resulting collision forces $[\mathbf{f}_{\mathrm{VPS}}, \mathbf{t}_{\mathrm{VPS}}] = \lambda[\mathbf{f}_{\mathrm{tot}}, \mathbf{t}_{\mathrm{tot}}]$. The correction factor depends on the total number of points in the pointshell N and the number of colliding points n. With the chosen formula, we increase the effect of contacts consisting of few colliding points:

$$\lambda = \begin{cases} \left(\frac{\ln(\rho N + e)}{\ln(n+e)}\right)^{\alpha} \frac{1}{\rho N} & \text{if } n < \rho N \\ \frac{1}{n} & \text{if } n \geq \rho N, \end{cases} \tag{2}$$

The parameter ρ denotes the percentage of points that are colliding; it is empirically set to be 1%, whereas $\alpha = 2$ and $\ln(e) = 1$.

Additionally, since the VPS Algorithm is a penalty based method that yields volumetric forces [28], further force post-processing is required to handle the so-called *tunneling-effect*[3] with a higher fidelity. In order to tackle this problem, the virtual coupling method explained in this section was developed. Our implementation resembles a simplified heuristics of the constraint-based haptic rendering algorithm presented by Ortega et al. [29], commonly known as the *God Object Method*. The goal of our implementation is to compute a virtual coupling pose $\mathbf{H}_{\mathrm{VC}}$ of the object that remains on the collision surface, although we penetrate the surface with the haptic device ($\mathbf{H}_{\mathrm{D}}$). In order to achieve that, we first compute a step transformation vector $\Delta\mathbf{H}$ out of the resulting penetration ($p > 0$) and the collision forces and torques computed by the VPS algorithm ($\mathbf{f}_{\mathrm{VPS}}$, $\mathbf{t}_{\mathrm{VPS}}$). This step transformation denotes the minimal motion of the object to resolve the penetration:

$$\Delta\mathbf{H} = \eta p[\Delta\mathbf{x},\ \Delta\phi] \in \mathbb{R}^6, \tag{3}$$

[3]The *tunneling effect* appears among penalty-based algorithms when collision forces between objects that are just one polygon thick are computed. In these cases, the penalty value – the penetration, in the case of the VPS Algorithm – yields very small contact forces allowing interpenetration.

where

$$p = \max_i \left(V(\mathbf{P_i})\right) > 0,$$

$$\Delta\phi = \frac{\mathbf{t}_{\mathrm{VPS}}}{\|\mathbf{t}_{\mathrm{VPS}}\|}\left(\sigma\frac{\|\mathbf{f}_{\mathrm{VPS}}\|}{\|\mathbf{t}_{\mathrm{VPS}}\|} + \frac{\|\mathbf{t}_{\mathrm{VPS}}\|}{\|\mathbf{f}_{\mathrm{VPS}}\|}\right)^{-1}, \tag{4}$$

$$\Delta\mathbf{x} = \frac{\mathbf{f}_{\mathrm{VPS}}}{\|\mathbf{f}_{\mathrm{VPS}}\|}\sigma\frac{\|\mathbf{f}_{\mathrm{VPS}}\|}{\|\mathbf{t}_{\mathrm{VPS}}\|}\|\Delta\phi\|.$$

The inertia and mass ratio is set to be constant with an empirically determined value of $\sigma = 0.035\,\mathrm{m}^2$. The filter constant is $\eta = 0.2$. This step transformation $\Delta\mathbf{H}$ and the previous virtual coupling pose $\mathbf{H}_{\mathrm{VC}}$ are subtracted to the haptic device pose $\mathbf{H}_{\mathrm{D}}$. The projection of the resultant pose on the VPS forces and torques yields the current virtual coupling pose $\mathbf{H}_{\mathrm{VC}}$ shown in Fig. 7. In order to compute the corresponding coupling forces and torques, first $\mathbf{H}_{\mathrm{VC}}$ is transformed into object coordinates, and the translation and rotation vector δ is computed:

$$\delta = [\mathbf{x}_\delta, \phi_\delta] \in \mathbb{R}^6 \leftarrow \mathbf{H}_{\mathrm{D}}^{-1}\mathbf{H}_{\mathrm{VC}}. \tag{5}$$

This vector δ contains the necessary components to compute the virtual coupling forces:

$$[\mathbf{f}_{\mathrm{VC}}, \mathbf{t}_{\mathrm{VC}}] = [k_{\mathrm{f}}\mathbf{x}_\delta, k_{\mathrm{t}}\phi_\delta]. \tag{6}$$

The stiffness constants k_{f} and k_{t} are selected according to the maximum stiffness that can be displayed by the haptic device to obtain a realistic impression of collisions between rigid objects while keeping it stable.

Although we are aware of the limitations of these heuristics, this procedure turned out to be very satisfactory in practice. Colliding objects are appropriately visualized on the surface, avoiding the interpenetration configurations characteristic of penalty-based haptic rendering algorithms. Moreover, hard contacts can be generated even between thin or even non-watertight objects.

Collision Computation: Multibody Framework

Our system allows for collision computation between several objects. Fig. 8 shows the most important elements and

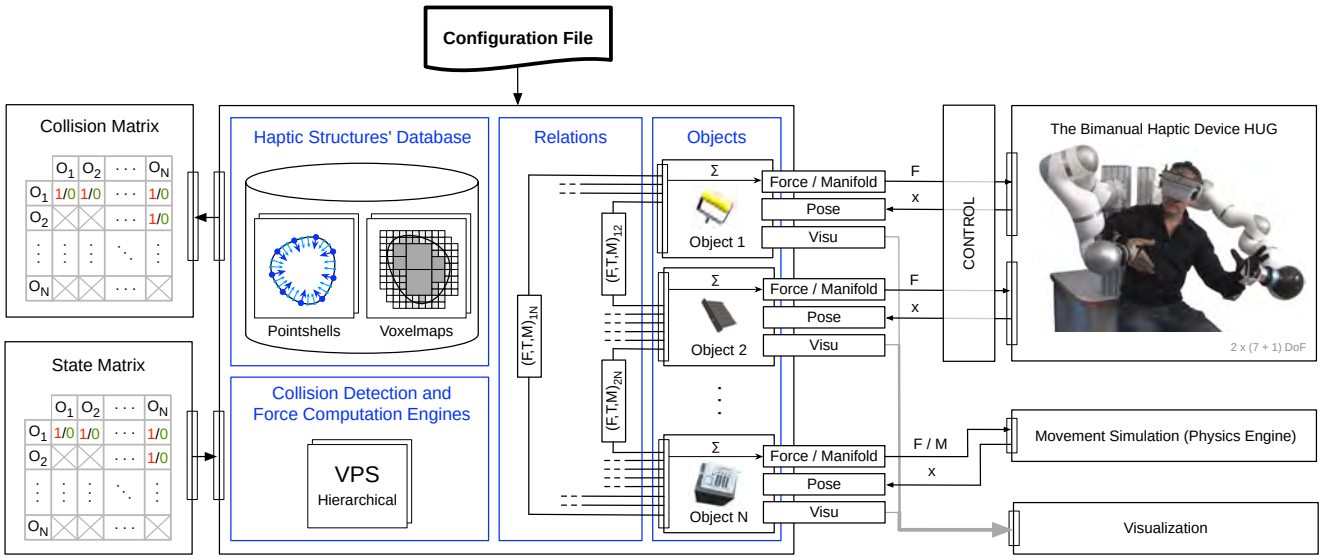


Figure 8. Overview of the multiple object environment framework for collision detection. Configuration steps and data stream are shown, as well as the most important connections within the environment. The scenarios are easily scalable and can be described in a XML-like configuration file.

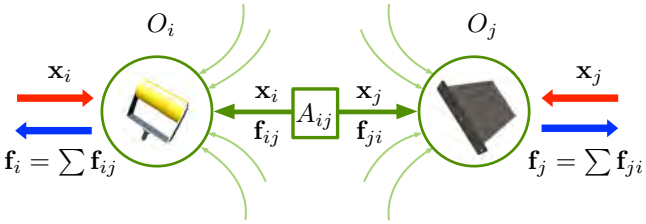


Figure 9. Two objects (handle O_i and module O_j) linked through a collision detection call or algorithm (A_{ij}). The user sends the pose of each object. Each algorithm computes the collision between its corresponding objects reading the last received pose ($\mathbf{x}_i$, $\mathbf{x}_j$). Each object receives the collision forces, torques, and manifolds of its corresponding algorithms, and sends a sum of all forces ($\mathbf{f}_i$, $\mathbf{f}_j$) outwards to the user.

interfaces of the framework. In order to build a scenario, the user describes in a simple configuration file the objects present during the collision detection process. This description contains basically the file-paths of the objects and their initial poses in the virtual reality, as well as additional parameters required by the VPS algorithm. This simple but powerful object-oriented script enables describing rich multi-object scenarios in a very comfortable and scalable way.

Once the configuration file is parsed, the library

1. generates a data-base with all haptic data structures (voxelmaps and pointshells) used in the simulation,
2. determines which object pairs can collide with each other, and thus, should be able to be checked for collision, and
3. starts and keeps the collision detection engine threads (VPS) between objects that likely could overlap.

The whole object and algorithm pool is organized similarly to a graph where the nodes are the objects and the edges correspond to the collision detection algorithms. The implementation is parallelized.

As shown in Fig. 8, the user can set the pose of each object; the contact forces or manifolds can be read independently. Therefore, each object can be connected to a visualization engine, a haptic device, or a movement simulator. This structure allows for collaborative frameworks. Additionally, the user has access to collision and state matrices. On the one hand, the collision matrix provides the information whether object O_i is colliding with O_j. On the other hand, the state matrix can be used to activate or de-activate the collision detection between objects O_i and O_j.

Each object is aware of all the objects with which it can collide through the linking algorithms (see Fig. 9). Every time a new collision force is sent to the object force buffer, all object forces will be summed and sent outwards to the user if the eldest object force is being updated in the force buffer.

Collision detection and force computation should be carried out in a dedicated computer, given that it is an expensive and critical task. The complexity of the problem can increase quadratically with the number of objects in the scenario, thus, additional methods must be employed in order to relax it [30]. Two approaches have been implemented in our framework: (*i*) *object grouping* and (*ii*) the use of *spatio-temporal coherence*.

Object Grouping— By defining object groups in the configuration file we create object families in which collision detection between its members is not performed, since it is not necessary. For instance, the collision detection between the switches and the satellite can be saved, or the collision detection between two consecutive links of a mechanism like a gripper body and its jaws.

Spatio-Temporal Coherence—Collision detection has a high spatio-temporal coherence, that is, the state in two consecutive cycles is very similar. This idea is implemented in practice by observing the distance values ($p < 0$) of the

objects: the distance is divided by a maximum possible arm movement velocity (1 m/s in our scenarios) yielding the coherence time in which the collision query can be dispensed, since we know objects cannot overlap during that time period.

Movement Simulation

Once the contact data of the objects has been computed, this can be sent either to the haptic device so that the user feels the contacts during manipulation, or to the movement simulation module (see Fig. 8 right). This section covers the last case. The connection to a haptic device is introduced in Section 4.

We use the physics engine Bullet[4] for computing the movement dynamics of the objects that are not coupled to the user via the human machine interface. Many important works were published in the past years regarding dynamics simulation [31], and several physics libraries are publicly available, each one with its advantages and disadvantages. Our choice for Bullet is because this engine is open source, multi-platform, supported by an active community of developers, and it ranks better performance results compared to other libraries [32].

Bullet has its own collision detection engines, but to ensure high update rates it usually works with simplified geometries. Therefore, we bypass the collision detection of Bullet — both narrowphase and broadphase — and directly use the rigid body dynamics module which solves the motion equations — i.e., Newton-Euler — of the objects subject to their constraints. Note that each object of the scene has its own representation as rigid body in Bullet.

We implemented two approaches for providing contact data to the Bullet library. With the first method, forces and torques computed by VPS are scaled and applied to the rigid body representations in Bullet. With the second method [27], we pass the contact manifold $\{(\mathbf{P}, \mathbf{n}(\mathbf{P}), V(\mathbf{P}) > \mathbf{0})_i\}$ (see Fig. 5) to the physics library. Then, Bullet computes the corresponding constraint-based forces and integrates them obtaining the object movement.

The main difference between both approaches is the physical meaning of the forces. The second method seems certainly a more coherent approach. However, the number of contact points supported by Bullet is limited to a fixed number. On the other side, the forces generated by our multibody collision detection framework contain a weighted sum of all contact points and are easy to integrate into Bullet. Due to these reasons, the first method was used during the experiments reported in this work. Future work will address this issue by improving the contact manifold method.

4. Interaction Devices and Techniques

One of the key characteristics of sophisticated virtual environments is a real-time multi-modal interaction between the human operator and the virtual world. To create an immersive experience for the operator, such a virtual environment not only involves vision but also other sensory modalities such as the haptic modality. Our goal is to provide a multi-modal virtual environment that delivers a lifelike simulation in order to familiarize astronauts and robot operators to the specific challenges that arise in telerobotic on-orbit servicing tasks.

Next to telerobotics, an application area of VR-OOS is astronaut training. The effectiveness of training environments is established on their ability to deliver a successful transfer of training. In the context of virtual environments, the effectiveness is determined by the amount of correct transfer of skills and knowledge from the virtual environment to the real world. When the environments and the situations are similar, the relevant information is accessed and transferred — the greater the similarity, the greater the transfer [33]. In order to enable a great transfer of training, we seek to provide an environment which includes realistic simulation of dynamic and kinematic behavior of satellite components, as well as provision of a natural and intuitive simulation of physical interaction between human and the environment, especially the realistic grasping of objects. Hence, selecting an appropriate interaction device and interaction technique is of vital importance. VR-OOS is designed as a research platform to investigate and advance novel interaction techniques. The system not only includes support for immersive virtual environment and standard interaction devices such as motion tracking of head, hand and fingers, or commercial haptic feedback devices, but also novel custom built devices, which are outlined in the following sections.

Figure 10. The bimanual haptic device HUG of the DLR provides the human operator with force feedback from the virtual world. Magnetic clutches couple the human hands safely to the robots, while data gloves determine the finger positions.

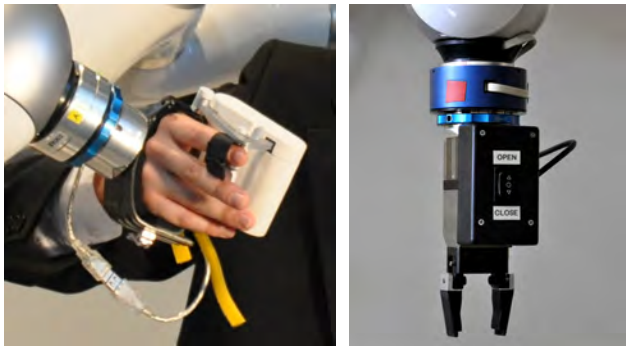
The DLR Bimanual Haptic Device HUG

The most relevant hardware component of the VR-OOS framework is the DLR Bimanual Haptic Device HUG [34] shown in Fig. 10. This highly sophisticated haptic device is composed of two Light Weight Robot arms, which are able to provide a human operator with realistic haptic force feedback on both hands. The robots have seven degrees of freedom each and are commanded at an update rate of 1 kHz, while the internal joint and motor current controllers run at even faster rates of 3 kHz and 40 kHz, respectively. To ensure safe operation, HUG is equipped with a multi-layer safety architecture comprising collision avoidance, redundant sensors, a dead-man loop, and magnetic clutches for the hand interfaces.

Compared to commercial haptic devices, HUG has several unique characteristics that bring valuable advantages to the VR-OOS scenario:

- The workspace of the human arm is largely covered by the robot arms [35]. Hence, the operator is able to intuitively perform tasks that require large arm movements without being restricted by the haptic interaction device.

[4] http://bulletphysics.org/wordpress/



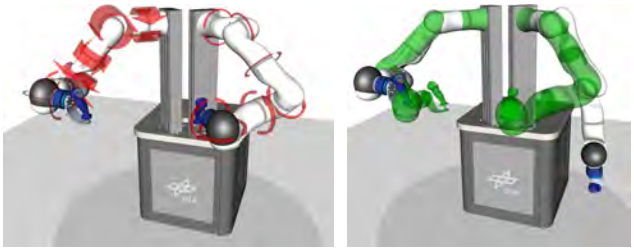
(a) (b)

Figure 11. Two alternative hand interfaces: (a) The active hand feedback device enables the operator to control the grasping force applied on virtual objects; (b) The robotic gripper with a manual switch was used for the pilot study (see Section 6).

- A broad set of various hand interfaces may be used due to the maximum payload of 7 kg per arm. The hand interfaces currently used for grasping and manipulation tasks in the virtual world are passive datagloves (Fig. 10) and active force feedback devices attached as end-effector to the wrists of HUG (left photo of Fig. 11). However, in order to fairly assess the authenticity of the VR-OOS simulation, a robotic gripper (right photo of Fig. 11) was used for the pilot study of Section 6.
- Impedance and admittance controlled operation is possible due to torque and position sensors integrated in each robot joint. Thus, the robots not only can be used to provide the operator with force feedback, but also guide the human hand along predefined trajectories or demonstrate movements. This feature is especially useful for training purposes.
- An additional force-torque sensor at the wrist of each robot arm precisely measures the interaction forces between the human and the device. These forces are used by an impedance feedforward controller [36] to scale down the apparent inertia to 30% of its original size.

To simplify and speed up the programming of such a complex robotic system, a separate interactive robot viewer [37] intuitively augments robot parameters over a virtual representation of the haptic device HUG (see Fig. 12). It features round arrows to indicate the torque difference between the measured and commanded torques of the revolute robot joints, annotated labels (also called *billboards*) to show the numerical parameter values, and a transparent phantom of the robot to reveal the target position of the motion interpolator. These properties not only turn the robot viewer into a powerful tool for supervising the state of the robotic system, but also enable easily simulating and testing new control modules offline, i.e., without the robotic hardware.

With all its characteristic features, HUG enables the operator to intuitively manipulate objects in the virtual scene, interact with the virtual environment, and realistically feel 6-DoF contacts while performing VR-OOS tasks. However, the quality of haptic feedback does not only depend on the haptic device and the haptic algorithm used, but also on the parameters controlling the interaction between the virtual and the real world. To this end, an analytical stability analysis has been conducted [38] and recently complemented by an optimal control approach [39].



(a) (b)

Figure 12. The robot viewer visualizes the current position and system parameters of HUG. (a) The red round arrows show torque differences between the measured and commanded torques of the revolute robot joints. (b) The green transparent robot phantom shows the target position of the motion interpolator.

This approach allows to optimally set the parameters of virtual walls and collisions between virtual objects. It is physically motivated and maximizes the energy dissipation of the transient response of a virtual collision. The method is particularly advantageous for the challenging case of stiff impacts, e.g., if simulating collisions between two stiff metal parts. Thus, it contributes to further improve the quality of haptic feedback provided by HUG.

Electro-Tactile Feedback

Haptic information plays a crucial role in diverse everyday interactions, especially during perception, grasping, and manipulation of objects. While the haptic device HUG provides robust force-feedback, which is important especially for teleoperation tasks, such kind of devices are however floor-mounted and can be bulky for some scenarios. This may restrict the movements of users during training or analysis tasks, for instance, while performing in spacious CAVE-like[5] immersive virtual environments, where users may prefer to move around to explore and understand the environment. In order to cater to such needs, we developed promising alternative mobile interaction devices, which even may be used in parallel to HUG, to increase the level of interaction and haptic feedback.

Fig. 13 shows the first prototype of our lightweight electro-tactile device designed to simulate tactile sensations such as touch, pressure, movement, and slip, that may occur during grasping and manipulation tasks at the finger tip. Electro-tactile stimulation is provoked by a current passed through the skin, which excites directly the afferent nerves and causes tactile sensation. Four different types of nerve endings, namely Meissner's Corpuscles, Merkel's Disks, Pacinian Corpuscles, and Ruffini Endings, can be stimulated in order to simulate many different types of contact and sensation. Therefore, electro-tactile feedback can evoke a range of sensations including tingling, itching, vibration, touch, pressure, pinch, or pain.

Our prototype generates short electric pulses ($<100\,\mu$s, $0-200$ V) and applies them to the skin on the user's finger tips using small electrodes ($2.54 \times 1.28\,\text{mm}^2$). Each finger

[5] A Cave Automatic Virtual Environment (CAVE) is an immersive virtual reality environment where different walls, floor and roof of a cube-like room can be projected with images.

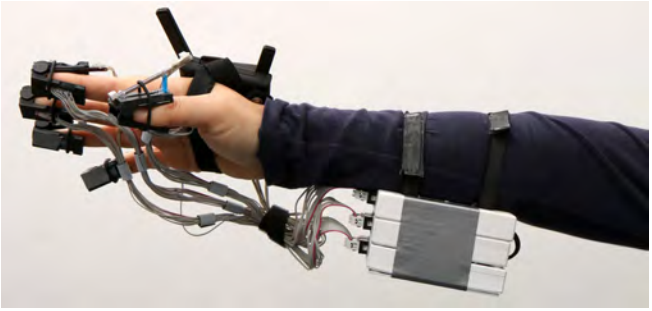


Figure 13. The first prototype of the electro-tactile feedback device mounted onto a user's arm.

Figure 14. The VibroTac provides tactile feedback on the arm using six vibration motors equally distributed. The variation of the rotation frequency of the motors enables a higher resolution than the six segments.

is equipped with eight electrodes to activate different types of nerve fibers at the same time and to enable the device to provide spatial information on the finger tip. Additionally, the thimbles of our prototype are equipped with infrared LEDs which can be synchronized with optical tracking systems.

Vibro-Tactile Feedback

A further mobile feedback device that may be used in immersive virtual environments is the VibroTac (see Fig. 14). This tactile feedback bracelet consists of six vibrotactile actuators which can apply vibrotactile stimuli to the human arm [40]. Each actuator can be activated and controlled independently resulting in a variety of vibrotactile stimulation patterns that can be generated and displayed to the user. It turned out that this vibration feedback is perfectly suited to display direction and collision information to the wearer of the device [41]. In our application, we use the VibroTac to display contacts and collisions between the human arm and virtual objects while the arm is optically tracked.

Pseudo-Haptic Feedback

In case haptic feedback devices are not available or not applicable for the preferred immersive virtual environment due to their heaviness or complexity, pseudo-haptic feedback can be an alternative. In order to improve grasp performance where no haptic feedback can be provided to the user, we proposed three simple pseudo-haptic methods to apply forces onto virtual objects [42]. Our first method uses an indirect interaction technique exploiting a standard interaction device of our VR system, an ART[6] Flystick2. The Flystick2 is a tracked 6-DoF target that has six buttons and an analog thumb joystick along the x and y-axis. We linearly mapped the analog value of the joystick's y-axis (up-down) to opening and closing a virtual hand. Pushing the joystick completely up would fully open the hand, and pulling it completely down

[6]http://www.ar-tracking.com

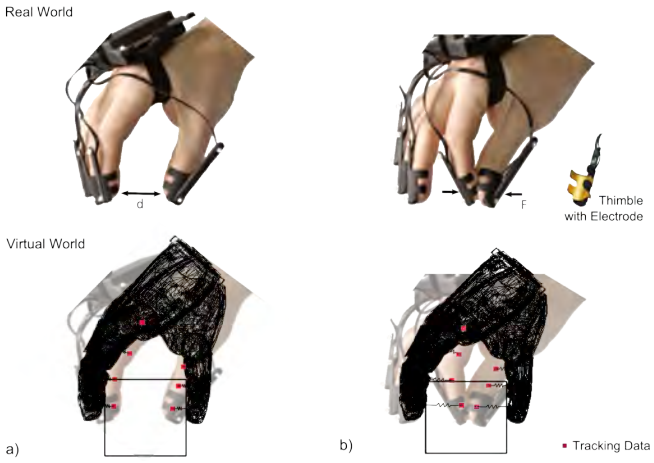


Figure 15. Representing heaviness of grasped virtual objects through: (a) the distance between tracked fingers and thumb; (b) the strength of pinching fingers and thumb.

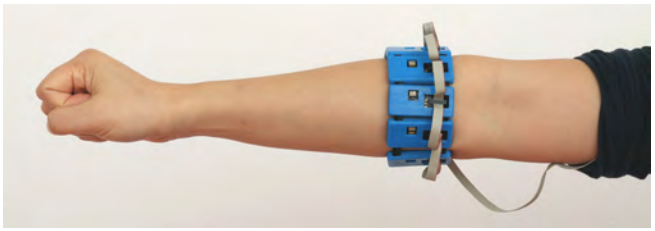


Figure 16. The first prototype of our myoelectric device mounted onto a user's arm.

would close the hand and apply the maximum force. In the joystick's neutral position, the virtual hand is closed, so that no gripping force is applied. To apply a force to the virtual fingers, the user has to pull down the joystick while the force increases linearly with the analog joystick value.

Our second method is a direct interaction technique and is based on the penetration caused by tracked finger positions when grasping a virtual object. This method uses a combination of the kinesthetic and proprioceptive feedback from the fingers (distance between fingers and the thumb) mapped to the grip force in combination with the simulation of appropriate visual feedback of the object behavior (e.g., object being lifted / not lifted). A standard finger tracking device from ART is used in this method and illustrated in Fig. 15a.

Our third method is also a direct interaction technique and exploits self-haptic feedback — the tactile feedback created by the user's fingers when pinching the fingertips and the thumb for grasping a virtual object. The strength of the pinch is mapped directly to the strength of the grasping force in combination with the simulation of the appropriate visual feedback of the object behavior (e.g., object being lifted / not lifted / almost lifted). For grasping light objects, the user pinches very gently. Grasping heavy objects requires the user to pinch stronger. We implemented this method using a finger-tracking device that has been extended with a pinch intensity sensor from ART. It has electrodes attached to the thimbles and measures the electrical impedance between the thumb and any of the four fingers through skin contact. The method is illustrated in Fig. 15b.

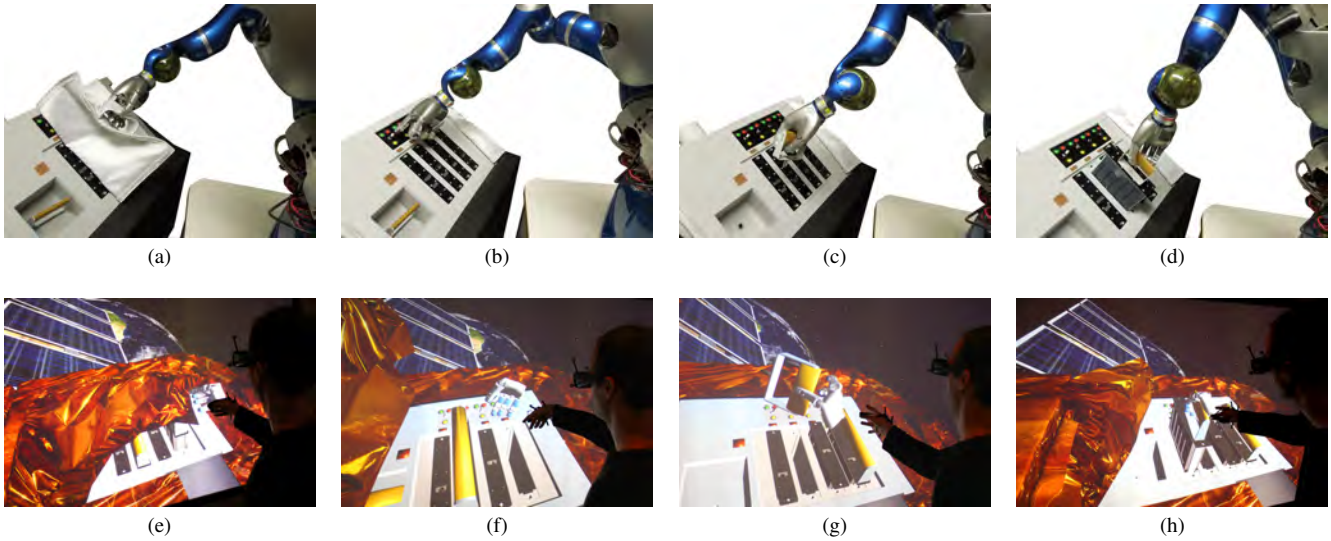


Figure 17. Use Case Scenario. The columns show the several tasks of the use case scenario, where a broken electronic module must be replaced. (a) and (e): remove the MLI cover of the satellite; (b) and (f): turn off the switch of the broken module; (c) and (g): grab the handle tool and insert it into the hole of the module; (d) and (h): take off the broken module. The rows show two setups that enable the user to interact with the use case scenario. (a) to (d): the teleoperated Space Justin; (e) to (h): a virtual reality simulation using a power wall, finger tracking, and pseudo-haptic feedback.

Myoelectric Input

Optical, magnetic, inertia, and ultrasonic tracking systems cannot measure forces on user's fingers after closing the hand for grasping. To solve this issue, various techniques have been developed, such as resistive sensors, as also used above for measuring the pinch force of the pseudo-haptic feedback, or piezo-elements, conductivity measurement between fingers, force-feedback or myoelectric devices. However, readymade consumer products, like the MYO from Thalmic[7], do not offer low-level access to hardware and only provide gesture recognition. Most other available interaction devices are desktop based, heavy, difficult to apply to the user, or invasive, and therefore, not applicable for everyday's research in immersive virtual environments. Hence, we developed our own myoelectric prototype device as shown in Fig. 16. It measures the myoelectric signals of the muscles at eight locations distributed evenly over the upper forearm.

All of our previously introduced feedback and input devices, namely electro-, vibrotactile, pseudo haptic feedback, and myoelectric input, can be used in parallel to the HUG, and hence further enrich the level of immersion for the human operator.

5. Use Case Scenario

Our use case scenario is designed to represent a variety of manipulation tasks and is based on a physical satellite mockup that is used to demonstrate the manipulation capabilities of DLR's telepresence system (Fig. 1a and 1b; upper row of Fig. 17). The virtual mockup allows the fulfillment of these tasks in our simulation (see Fig. 18). We define a general procedure called "replacement of a malfunctioning satellite module" which englobes most of the tasks that can be carried out (see Fig. 17). In this procedure, first, the MLI cover wrapped around the mockup is lifted. Then, the switch related to the broken module is turned off to disconnect the module from its power supply. This new state is displayed by a color change of the lamp located above the module from green to red. To unplug the module, a handle is needed which is grasped with a two-finger gripper. The handle is inserted in the corresponding hook on the module allowing the removal from its cavity. A new module can be inserted in the cavity and the power supply is switched on.

These tasks are realized using various combinations of the components described in Sections 3 and 4, emphasizing the modularity of our approach. The first row of Fig. 17 shows the interaction with the physical mockup performed by SpaceJustin, a humanoid upper body with $17 + 30$ DoF (7 for each arm, 3 for the torso and head and 15 for each robotic hand) shown in Fig. 1b, which is teleoperated using the bimanual haptic device HUG (see Section 4). Simulating the humanoid robot in the virtual environment is one of the goals of our simulation.

Another realization of the satellite maintenance use case is shown in the second row of Fig. 17. A powerwall displays a realistic visualization of the scenario simulating surface materials, stars, and the atmosphere. Light-weight input devices like the myolelectric input device or the finger tracking allow interaction with the virtual environment without disturbing the visual immersion into the scene. The user controls a virtual five-finger robotic hand and can interact with deformable objects like the MLI cover wrapped around the satellite.

A further setup uses the HUG to ensure the high immersion of the user into the virtual environment by combining haptic and visual feedback. The operator wears a head-mounted display and his/her head movements are tracked allowing to intuitively change the viewpoint in the virtual scene. The VibroTac gives feedback on the lower arm of the user if his/her elbow collides with virtual objects. The operator controls the virtual counterpart of a two-finger gripper that is also used in the pilot study described in the next section.

[7] http://www.thalmic.com

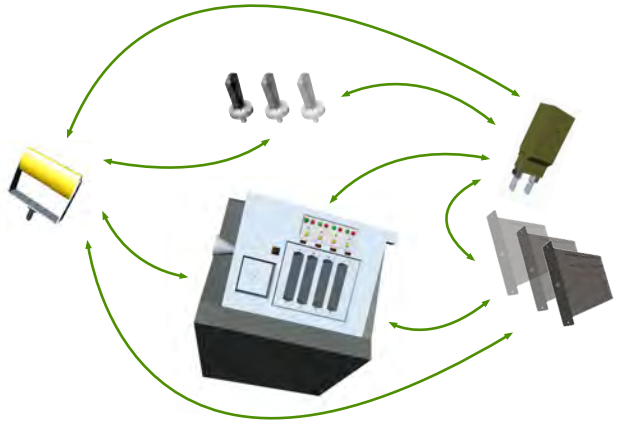


Figure 18. The interaction possibilities in the use case scenario. The virtual representation of a satellite mockup designed to test various manipulation scenarios, i.e., turn on/off switches and (un-)plug modules using a tool. The collision detection module and the physics engine have to cope with the interactions between all the elements in the scene, displayed with green arrows.

This unique setup will allow for a fair comparison between real teleoperated and virtual tasks in future work.

6. Experimental Results

This section reports some key performance values that are more thoroughly discussed in [14] and presents the results of a pilot user study that was carried out for evaluating the system usability.

Performance

We measured the latency for transmitting messages to synchronize the simulation components across the distributed simulation components and processing a simulation cycle. The time between moving the hand using the HUG and delivering the resulting force when touching a moving virtual object was on average $t_1 = 2.8$ ms, ($t_1 = \sum t_{x_{hd}} + t_{F_{\text{user}}} + t_{x_{obj}} + t_{F_{hd}}$, for a description of symbols refer to the data flow in the simplified system architecture in Fig. 2). The time between moving the hand and the visualization of the resulting movement of a touched or pushed object was on average $t_2 = 7.1$ ms, ($t_2 = \sum t_{x_{hd}} + t_{F_{\text{user}}} + t_{x_{obj}} + t_{\text{image}}$, note that t_{image} does not include the time waiting for the vertical sync refresh on the graphics card). A more detailed discussion of the latency measurements of the distributed system can be found in [14].

Pilot User Study

We conducted a pilot study to evaluate the virtual reality simulation and compare it to the real world in order to detect the most important evaluation parameters for a later more thorough study. We present the results we obtained to give an idea on the system usability.

The participants were instructed to perform four manipulation **tasks** with the satellite mockup in three different **conditions**. The four manipulation **tasks** are shown in Fig. 19; they consisted in (a) colliding against a specified square on the satellite mockup, (b) turning off two switches on the satellite, (c) grabbing the handle tool, and (d) inserting the handle tool in a module of the satellite.

The tested **conditions** consisted of:

1. R: Tasks performed in the *reality* (R) on a physical mockup of a satellite with the bare hand.
2. RH: Tasks performed in the *reality* (R) on a physical mockup of a satellite but guiding a robot arm of the HUG (H) in gravity compensation mode (i.e., the robot arm followed the movements of the human without resistance).
3. VR: Tasks performed in the *virtual reality* (VR) on the virtual mockup of the satellite (1:1 replica of the physical) using the robot arm of the HUG.

In the conditions RH and VR, shown in Fig. 20, the participants had to move the gripper of Fig. 11 attached to the end effector. This setup was used to ensure that virtual and physical scenarios were as similar as possible.

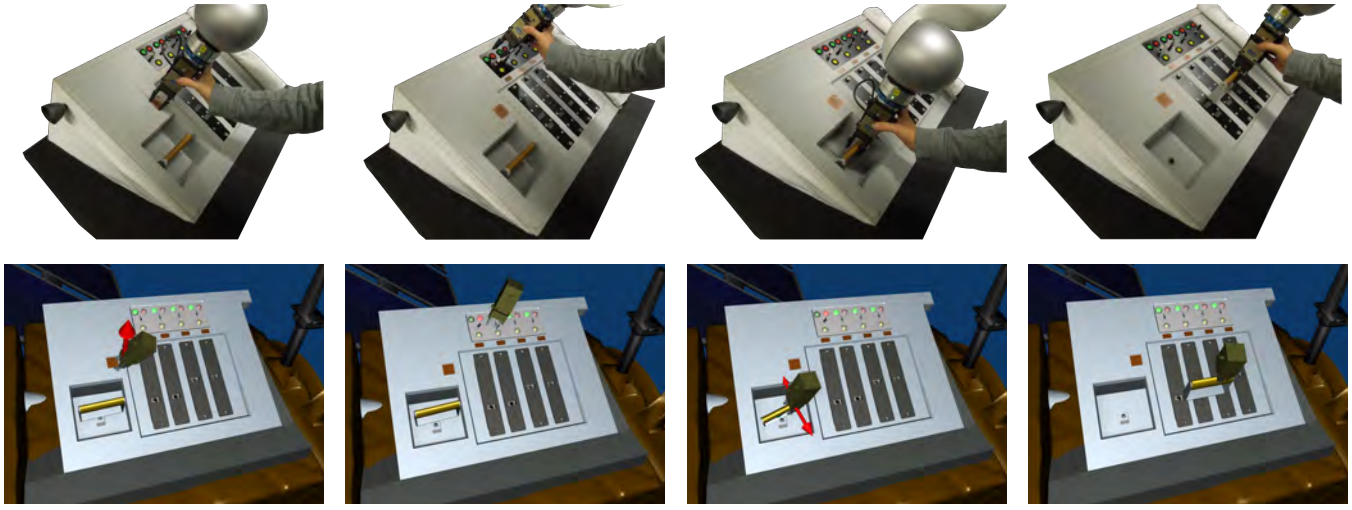
Five students of the DLR (23.4 years on average, 1 female) without experience with the system and moderate experience with computer games participated in the study. The tested conditions were permuted in order to decrease learning effects. In all conditions, participants started their movement from a predefined initial position situated approximately centered with respect to the satellite and about 0.3 m above it. They were asked to carry out the tasks twice, and as quick and precise as possible without stopping between tasks. Each experimental trial finished when the handle was correctly inserted into the module.

We evaluated the performance with objective and subjective measures. Overall, the system proved its usability, since all participants could successfully fulfill all tasks. The most relevant objective measure was the time to complete the task, as shown in Fig. 21, whereas Table 1 collects the questionnaire and the obtained answers related to the subjective measures. The objective data was recorded during the experiments and the users were asked to answer the corresponding part of the questionnaire after completing each condition.

The objective results show an increase in time to complete (TTC) the task comparing R (avg. 5.5 s, std. dev. 1.1 s) and RH (avg. 20.7 s, std. dev. 4.3 s). This clearly is the influence of using a robot arm and a gripper to fulfill the tasks, as all participants were completely new to the system. Performing the tasks in VR took in average 29.8 s (std. dev. 7.0 s).

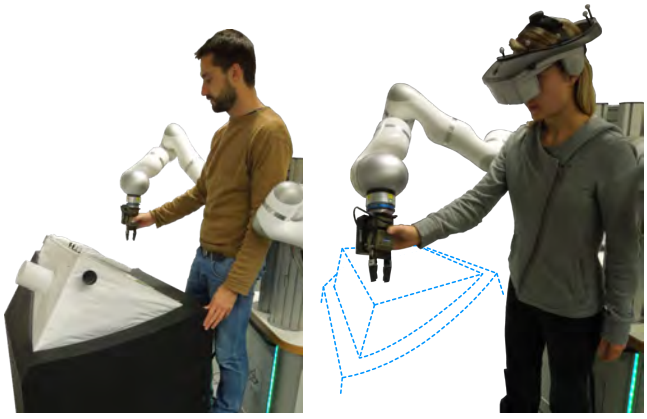
The subjective results show a moderately high presence (item 1.1 from Table 1) and realism (items 5.1 to 5.4) evaluation of our system. It is worth pointing out that the standard deviations indicate the need of a bigger sample to formulate generally valid statements; as mentioned at the beginning of this section, we pursued a pilot study in order to detect tendencies and relevant parameters.

In this line, deviation values below 15% are highlighted in green in Table 1, whereas values above 25% are shown in orange. We conducted a short interview with the participants that originated high deviation values in order to find out the reasons for their choice and found out relevant information to consider in future evaluations. For instance, the high deviation in item 4.3 is due to a participant who was particularly sensitive to having the system around him constraining the otherwise natural movements; in the other conditions this awareness decreased, since he was interacting with the HUG as a tool.



(a) (b) (c) (d)

Figure 19. Participants had to perform four tasks in the pilot study: (a) Collide with the satellite box on a given spot starting from a fixed position situated approximately centered with respect to the satellite and about 0.3 m above it; (b) Turn off two switches; (c) Grab the handle tool; (d) Plug the handle into the hole of a module. Two of the three conditions are displayed here: RH (first row) and VR (second row).



(a) (b)

Figure 20. Setup for the pilot user study. The participants move a gripper attached to one robot arm of the HUG: (a) They perform the manipulation tasks in the real world (RH condition) and (b) also in the virtual reality (VR condition).

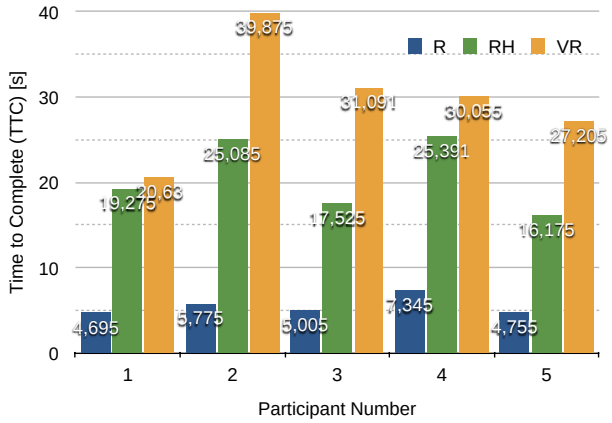


Figure 21. Average time to complete (TTC, [s]) the tasks for each participant and condition; R: tasks performed in the reality with the bare hand; RH: tasks performed in the reality using the gripper attached to one robotic arms of the HUG; VR: tasks performed in the virtual reality with visual and haptic feedback using the HUG.

The increase in TTC commented previously corresponds to an increase in physical and mental complexity in the RH and VR conditions, as it is revealed by the average values of questions 2.1, 2.2, 3.1, and 3.2, compared to the values of items 4.1 and 4.2. Along this line, the evaluation of natural interaction (items 2 to 4.4) decreases as devices and virtual interactions are introduced during the tasks, reaching an overall loss of 30% (items 2.4 and 4.4) according to our preliminary evaluation. As far as the effect of HUG is concerned, we observed a rather small effect of the HUG in the analyzed tasks (items 6.1 to 6.4). Besides of the presented questionnaires, at the end of the experiments the participants were asked to answer the Simulator Sickness Questionnaire [43]; overall, no significant effects were detected.

Although participants moved slower, the pilot study showed the successful performance of all given tasks using our virtual reality simulation with haptic feedback.

7. Conclusions

In order to reduce costs and risks, telerobotic on-orbit servicing (OOS) becomes more and more important for maintenance and life extension of future spacecraft missions. However, the space system and the service robot have to fit together so that servicing operations are possible at all. Furthermore, the operator has to be trained on the tasks to be performed. To find optimal solutions, all interfaces between operator and robot, as well as between robot and spacecraft, have to be designed carefully. This requires continuous design iterations over the development lifecycle. One of the

Table 1. Subjective Results of the experiments displayed in Fig. 19. Average and standard deviation values are given in points and percentage values; Deviations below 15% are in green, whereas deviations above 25% are in orange. All punctuations start at value 1 and increase unit-wise until their specified maximum value.

	Average		Standard Deviation	
	Points	%	Points	%
1. Overall Presence Evaluation in the VR				
1.1. How present did you feel in the VR? (*very present:* 100)	62	62	31.14	31.14
2. VR: Tasks in the virtual reality with HUG				
2.1. Mental complexity (*very complex:* 20)	8.8	41.05	5.02	26.42
2.2. Physical complexity (*very complex:* 20)	5.6	24.21	3.44	18.08
2.3. Constriction level (*very high:* 7)	3.6	43.33	1.14	19.00
2.4. Natural interaction (*very natural:* 7)	4.6	60	0.55	9.13
3. RH: Tasks in the physical reality with HUG				
3.1. Mental complexity (*very complex:* 20)	4	15.79	2.74	14.41
3.2. Physical complexity (*very complex:* 20)	5.4	23.16	2.70	14.22
3.3. Constriction level (*very high:* 7)	4	50	1.22	20.41
3.4. Natural interaction (*very natural:* 7)	5.4	73.33	0.89	14.91
4. R: Tasks in the physical reality without HUG				
4.1. Mental complexity (*very complex:* 20)	2.4	7.37	1.67	8.81
4.2. Physical complexity (*very complex:* 20)	2.6	8.42	2.07	10.91
4.3. Constriction level (*very high:* 7)	3	33.33	2.35	39.09
4.4. Natural interaction (*very natural:* 7)	6.4	90	0.55	9.13
5. How realistic was VR compared to RH in each exercise?				
5.1. Collide with specified spot (*very realistic:* 7)	4	50	1.87	31.18
5.2. Turn off switches (*very realistic:* 7)	3.8	46.67	0.84	13.94
5.3. Grab handle (*very realistic:* 7)	4.8	63.33	1.79	29.81
5.4. Insert handle into module hole (*very realistic*: 7)	4.8	63.33	1.30	21.73
6. How similar was RH compared to R?				
6.1. Collide with specified spot (*very similar:* 7)	6	83.33	0.71	11.79
6.2. Turn off switches (*very similar:* 7)	4.8	63.33	1.64	27.39
6.3. Grab handle (*very similar:* 7)	4.4	56.67	1.52	25.28
6.4. Insert handle into module hole (*very similar:* 7)	5	66.66	1.58	26.35

most feasible approaches is to carry out design development and training with Virtual Reality (VR) simulators.

In this paper, we presented and evaluated the DLR's VR framework for simulating OOS tasks. We could show that the depicted architecture supports two main user interfaces efficiently: teleoperation with the DLR's bimanual haptic device HUG and pure virtual environments with tactile, pseudo-haptic, and haptic approaches. The HUG can efficiently be used whenever realistic haptic feedback on both human hands is required and large workspaces have to be covered. This is especially the case in maintenance and repair tasks where applying the correct amount of force to the objects involved is crucial for task success. Assessed in a virtual environment, the HUG can directly be evaluated for usage with real service robots like DLR's SpaceJustin (see Fig. 1). On the other hand, virtual reality environments are more suitable for varying test cases, as the configuration can quickly be changed with low effort and subsequent costs. A usual disadvantage of pure virtual reality environments, however, is the lack of accurate haptic feedback. A combination of standard pointing devices like the Flystick with myoelectric input sensors has shown an improvement of interaction which alleviates the drawback considerably. Additional feedback devices like the shown vibro- and electrotactile devices attached to elbows and fingertips are also very promising. Nevertheless, more intensive research is required in this domain.

Besides the user interfaces, we presented a highly efficient combination of algorithms for collision detection (our en-

hanced version of the VPS algorithm) and physical simulation (Bullet physics engine). This allows the realization of a complex use case in a manipulation scenario where a broken electronic module must be replaced on a satellite. The scenario involves removing a MLI cover, turning on/off switches, using tools like handles, and extracting modules. In other words, multibody simulation could be performed, with and without movement constraints or articulation, while keeping the 1 ms time constraint and enabling realistic virtual manipulation tasks even for stiff collision configurations.

The framework was evaluated in terms of latency measurements that proved to be sufficiently small to ensure a realistic interaction with the simulation. Additionally, a pilot study was conducted which clearly shows that our approach is able to simulate maintenance tasks for OOS scenarios quite realistically. Although all participants could fulfill the required tasks, the deviation in subjective results was quite high due to the small number of participants. Hence, the next step will be a thorough evaluation of the overall setup with a higher number of users.

To further push the boundaries of our developed framework, we currently exploit additional use case scenarios like tethering, which is frequently needed for satellite maintenance. We also plan to combine our framework with a terrain renderer which generates very realistic scenarios that include a large amount of environmental data in order to simulate manipulation on planetary surface. Finally, we are also interested in evaluating the system as a training environment. Due to the distributed architecture, our system has potential to be used as a collaborative setup where multiple users (e.g., student and teacher) can interact with each other and the simulated environment.

REFERENCES

[1] J. N. Tatarewicz, "The hubble space telescope servicing mission," in *From Engineering Science to Big Science: The NACA and NASA Collier Trophy Research Project Winners*, P. E. Mack, Ed. University Press of the Pacific, 2006, ch. 16, pp. 365 – 396.

[2] A. Ellery, J. Kreisel, and B. Sommer, "The case for robotic on-orbit servicing of spacecraft: spacecraft reliability is a myth," *Acta Astronautica*, vol. 63, pp. 632–648, Jun. 2008.

[3] B. Weber, M. Sagardia, T. Hulin, and C. Preusche, "Visual, vibrotactile, and force feedback of collisions in virtual environments: effects on performance, mental workload and spatial orientation," ser. Lecture Notes in Computer Science, R. Shumaker, Ed. Springer Berlin Heidelberg, 2013, vol. 8021, pp. 241–250.

[4] R. B. Loftin, P. J. Kenney, R. Benedetti, C. Culbert, M. Engelberg, R. Jones, P. Lucas, M. Menninger, J. Muratore, L. Nguyen, T. Saito, R. T. Savely, and M. Voss, "Virtual environments in training: NASA's hubble space telescope mission," in *Interservice/Industry Training Systems & Education Conf.* Springer-Verlag, Nov. 1994.

[5] D. Homan and C. Gott, "An integrated EVA/RMS virtual reality simulation, including force feedback for astronaut training," in *AIAA Flight Simulation Technologies Conf.*, 1996, pp. 216–223.

[6] M. Romano, M. Bergamasco, L. Bessone, R. Henderson, and F. Rossitto, "Advanced methods for astronaut training: Computer based training and virtual reality technology," *Aerotecnica Missili e Spazio*, vol. 79, pp. 54–58, 2000.

[7] Y. Liu, S. Chen, G. Jiang, X. Zhu, M. An, K. Chen, B. Zhou, and Y. Xu, "VR simulation system for EVA astronaut training," in *AIAA Conf. and Exposition*, Sep. 2010.

[8] D. Reintsema, B. Sommer, T. Wolf, J. Theater, A. Radthke, J. Sommer, W. Naumann, and P. Rank, "DEOS – the in-flight technology demonstration of German's robotics approach to dispose malfunctioned satellites." in *Proc. of the 11th ESA workshop on advanced space technologies for robotics and automation (ASTRA 2011)*. Noordwijk, The Netherlands: ESTEC, 2011.

[9] T. Boge, H. Benninghoff, M. Zebenay, and F. Rems, "Using robots for advanced rendezvous and docking simulation," in *Proc. Simulation and EGSE facilites for Space Programmes (SESP)*, Noordwijk, The Netherlands, Sep. 2012.

[10] M. Sagardia, K. Hertkorn, T. Hulin, R. Wolff, J. Hummel, J. Dodiya, and A. Gerndt, "An interactive virtual reality system for on-orbit servicing," in *Proc. IEEE Virtual Reality (VR)*, Mar. 2013, (Video).

[11] P. Rank, Q. Mühlbauer, W. Naumann, and K. Landzettel, "The DEOS automation and robotics payload," in *Proceedings of the symposium on advanced space technologies for robotics and automation (ASTRA 2011)*. Noordwijk, The Netherlands: ESTEC, 2011.

[12] C. Preusche, D. Reintsema, K. Landzettel, and G. Hirzinger, "Robotics component verification on ISS ROKVISS – preliminary results for telepresence." in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2006, pp. 4595–4601.

[13] E. Stoll, U. Walter, J. Artigas, C. Preusche, P. Kremer, G. Hirzinger, J. Letschnik, and H. Pongrac, "Ground verification of the feasibility of telepresent on-orbit servicing," *Journal of Field Robotics*, vol. 26, no. 3, pp. 287–307, Mar. 2009.

[14] R. Wolff, C. Preusche, and A. Gerndt, "A modular architecture for an interactive real-time simulation and training environment for satellite on-orbit servicing," *Journal of Simulation*, vol. 8, no. 1, pp. 50–63, 2014.

[15] J. Behr, P. Dähne, Y. Jung, and S. Webel, "Beyond the web browser - X3D and immersive VR," Charlotte, North Carolina, USA, Mar. 2007.

[16] T. van Reimersdahl, T. Kuhlen, A. Gerndt, J. Henrichs, and C. Bischof, "ViSTA: a multimodal, platform-independent vr-toolkit based on WTK, VTK, and MPI," in *Proc. Int. Immersive Projection Technology Workshop (IPT)*, Ames, Iowa, 2000.

[17] P. Collienne, R. Wolff, A. Gerndt, and T. Kuhlen, "Physically based rendering of the martian atmosphere," in *Virtuelle und Erweiterte Realität : 10. Workshop der GI-Fachgruppe VR/AR*, 2013, pp. 97–108. [Online]. Available: http://elib.dlr.de/86477/

[18] E. G. Gilbert, D. W. Johnson, and S. S. Keehrthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, 1988.

[19] S. Gottschalk, M. C. Lin, and D. Manocha, "Obb-tree: A hierarchical structure for rapid interference detection," in *Proc. of ACM SIGGRAPH*, 1996.

[20] M. Lin and M. Otaduy, Eds., *Haptic Rendering: Foundations, Algorithms, and Applications*. A K Peters, Ltd., 2008.

[21] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proc. of ACM SIGGRAPH*, 1999.

[22] ——, "Voxel-based 6-dof haptic rendering improvements," *Haptics-e: The Electronic Journal of Haptics Research*, vol. 3, 2006.

[23] J. Barbič and D. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *IEEE Trans. Haptics*, vol. 1, no. 1, pp. 39 –52, 2008.

[24] K. Fischer and B. Gärtner, "The smallest enclosing ball of balls: Combinatorial structure and algorithms," in *Proceedings of the nineteenth annual symposium on Computational geometry - Annual Symposium on Computational Geometry*, 2003.

[25] M. Sagardia, T. Hulin, C. Preusche, and G. Hirzinger, "Improvements of the voxmap-pointshell algorithm - fast generation of haptic data-structures," in *53. IWK - TU Ilmenau*, Sep. 2008.

[26] R. Weller, M. Sagardia, D. Mainzer, T. Hulin, G. Zachmann, and C. Preusche, "A benchmarking suite for 6-dof real time collision response algorithms," in *ACM Virtual Reality and Software Technology*, Nov. 2010.

[27] M. Sagardia, T. Stouraitis, and J. L. e Silva, "A new fast and robust collision detection and force computation algorithm applied to the physics engine bullet: Method, integration, and evaluation," in *EuroVR*, 2014.

[28] J. Barbič, "Real-time reduced large-deformation models and distributed contact for computer graphics and haptics," Ph.D. dissertation, Carnegie Mellon University, 2007.

[29] M. Ortega, S. Redon, and S. Coquillart, "A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties," in *IEEE Trans. Visualization and Computer Graphics*, 2007.

[30] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *Proc. of ACM Interactive 3D Graphics Conference*, 1995.

[31] D. Baraff, "Dynamic simulation of non-penetrating rigid bodies," Ph.D. dissertation, Cornell University, 1992.

[32] J. Hummel, R. Wolff, T. Stein, A. Gerndt, and T. Kuhlen, "An evaluation of open source physics engines for use in virtual reality assembly simulations," in *Int. Symp. on Visual Computing*, Jul. 2012, pp. 346–357.

[33] C. J. Hamblin, "Transfer of training from virtual reality environments," Ph.D. dissertation, Wichita State University, Dept. of Psychology, College of Liberal Arts and Sciences, May 2005.

[34] T. Hulin, K. Hertkorn, P. Kremer, S. Schätzle, J. Artigas, M. Sagardia, F. Zacharias, and C. Preusche, "The DLR bimanual haptic device with optimized workspace (video)," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 3441–3442.

[35] F. Zacharias, I. S. Howard, T. Hulin, and G. Hirzinger, "Workspace comparisons of setup configurations for human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 3117–3122.

[36] J. J. Gil, A. Rubio, and J. Savall, "Decreasing the apparent inertia of an impedance haptic device by using force feedforward," *IEEE Trans. on Control Systems Technology*, vol. 17, no. 4, pp. 833–838, Jul. 2009.

[37] T. Hulin, K. Hertkorn, and C. Preusche, "Interactive features for robot viewers," in *Proc. Int. Conf. on Intelligent Robotics and Applications*, Oct. 2012, pp. 181–193.

[38] T. Hulin, A. Albu-Schäffer, and G. Hirzinger, "Passivity and stability boundaries for haptic systems with time delay," *IEEE Trans. on Control Systems Technology*, vol. 22, no. 4, pp. 1297–1309, Jul. 2014.

[39] T. Hulin, R. González Camarero, and A. Albu-Schäffer, "Optimal control for haptic rendering: Fast energy dissipation and minimum overshoot," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 4505–4511.

[40] S. Schätzle, T. Ende, T. Wuesthoff, and C. Preusche, "VibroTac: an ergonomic and versatile usable vibrotactile feedback device," in *Proc. IEEE Int. Symp. on Robots and Human Interactive Communications (RO-MAN)*, Viareggio, Italy, Sep. 2010, pp. 705–710.

[41] B. Weber, S. Schätzle, T. Hulin, C. Preusche, and B. Deml, "Evaluation of a vibrotactile feedback device for spatial guidance," in *Proc. IEEE World Haptics Conference*, Istanbul, Turkey, Jun. 2011, pp. 349–354.

[42] J. Hummel, R. Wolff, A. Gerndt, and T. Kuhlen, "Comparing three interaction methods for manipulating thin deformable virtual objects," in *Proc. IEEE Virtual Reality (VR)*, Mar. 2012, pp. 139–140.

[43] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," *The International Journal of Aviation Psychology*, vol. 3, no. 3, pp. 203–220, 1993.

BIOGRAPHY

Mikel Sagardia *studied Mechanical Engineering at Tecnun (University of Navarra, Spain) and at the Technical University of Munich, and graduated in 2008. He works since 2008 at the DLR - Institute of Robotics and Mechatronics as a researcher in the field of virtual reality. His research interests are in the fields of collision detection, haptic rendering, and physical simulation.*

Katharina Hertkorn *graduated in 2009 at the University of Stuttgart where she studied Technical Cybernetics. She works now at the DLR - Institute of Robotics and Mechatronics as a researcher in the field of telepresence and shared autonomy. She is particularly interested in the fields of control, grasping with multi-fingered hands, haptics, and shared autonomy.*

Thomas Hulin *received his Dipl.-Ing. degree from the Technical University of Munich in 2003. In that year, he also joined the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. His research interests include haptic devices, control and algorithms, physical human-robot interaction, tele- and space robotics, robot visualization and skill transfer. He was involved in the development of the vibrotactile device VibroTac, for which he received the DLR Innovation Award 2012.*

Simon Schätzle *received his Dipl.-Ing. degree in mechatronics from the Technical University of Munich in 2007. Since then he is applied to DLR Institute of Robotics and Mechatronics as research scientist. His research field is in the mechatronic design and human factors of haptic human system interfaces focusing on vibrotactile feedback. He was involved in the development of the vibrotactile device VibroTac, for which he received the DLR Innovation Award 2012.*

Robin Wolff *leads the Virtual Reality team at Simulation and Software Technology at the German Aerospace Center (DLR), where he works since 2010. He received a PhD in Immersive Collaborative Virtual Environments at the University of Salford, UK, in 2007 and was working there as technical director and researcher in several projects afterwards. In 2009, he was a visiting researcher at the Virtual Reality Laboratory at the Technical University of Tampere, Finland. He received his MSc in Scientific and Parallel Computation at the University of Reading, UK, in 2004 and his Dipl.-Ing. in Computer Engineering at the University of Applied Sciences in Berlin, Germany, in 2000.*

Janki Dodiya *is a Research Scientist at Simulation and Software Technology at German Aerospace Center (DLR). She received her PhD in Virtual Environments at the University of Reading, UK, in 2011. She received her MSc degree in Network Centred Computing at the University of Reading, UK, in 2005 and her BSc in Computer Application at the Sardar Patel University, India, in 2002. Her current research activities include human computer interaction techniques in immersive virtual environments.*

Johannes Hummel *studied Computer Science at the Technical University of Munich, and graduated in 2009. After being a freelancer developing software and user interfaces for simulation data management, he works now at the DLR - Institute for Simulation- and Software Technologies as researcher in the field of virtual reality. His research interests are electro-tactile and pseudo-haptic feedback and myoelectric input devices for the improvement of grasping in immersive virtual environments.*

Andreas Gerndt *is the head of the department "Software for Space Systems and Interactive Visualization" at the German Aerospace Center (DLR). He received his degree in computer science from Technical University, Darmstadt, Germany in 1993. In the position of a research scientist, he also worked at the Fraunhofer Institute for Computer Graphics (IGD) in Germany. Thereafter, he was a software engineer for different companies with focus on Software Engineering and Computer Graphics. In 1999 he continued his studies in Virtual Reality and Scientific Visualization at RWTH Aachen University, Germany, where he received his doctoral degree in computer science. After two years of interdisciplinary research activities as a postdoctoral fellow at the University of Louisiana, Lafayette, USA, he returned to Germany in 2008.*