2011 15th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications

# A Modular Architecture for an Interactive Real-Time Simulation and Training Environment for Satellite On-Orbit Servicing

Robin Wolff*, Carsten Preusche§ and Andreas Gerndt*
*Simulation and Software Technology
German Aerospace Center (DLR), Braunschweig, Germany
robin.wolff@dlr.de, andreas.gerndt@dlr.de
§Institute of Robotics and Mechatronics
German Aerospace Center (DLR), Oberpfaffenhofen-Wessling, Germany
carsten.preusche@dlr.de

***Abstract* - This paper outlines the development of a real-time interactive application for the analysis, training and programming of on-orbit servicing tasks within a virtual reality environment. The main challenges put on the system are the real-time simulation of the realistic dynamic and kinematic behavior of satellite components and additionally integrate interaction through a bimanual haptic interface, as well as enable tele-operation of a robot. We give an overview of the application, describe the real-time challenges and outline our approach and proposed system structure.**

*Keywords* - Real-time Applications; Virtual Reality; Physics Simulation; Haptics; Tele-robotics;

## 1. Introduction

On-orbit servicing (OOS) is an interdisciplinary field with increasing importance for the space industry. Although engineers take extensive care when designing and launching satellites, failures do happen. A survey by Ellery et al. [2] showed that most failures occur during orbit injection or in the first year of operation, for example when folding out the solar panel. As space systems are costly, any failure is damaging, not only for commercial operators, but also for research missions. Today, almost no satellite is launched without insurance to cover for failure. However, not only failure, but also a limited lifetime and an increasing amount of space debris are motivations for on-orbit servicing. With a growing number of satellites orbiting Earth, there is a need for innovative methods to repair failures, to refuel a satellite to extend its lifetime, or to remove disabled and adrift satellites and other unused parts from orbit in a controlled and cost-effective way.

One of the most famous example of on-orbit servicing is probably the series of NASA Hubble Space Telescope repair missions starting in 1993, where astronauts fixed and replaced parts in several EVAs (extra vehicle activity). Manned missions like this, however, are both expensive and put a high risk to the astronauts working outside the spacecraft. Robotic servicing constitutes an attractive alternative. The Canada arm has demonstrated successful operation [5]. Ongoing projects and concepts for robotic OOS studies include the Deutsche Orbitale Servicing Mission (DEOS) [23] with the goal to demonstrate the capturing of an uncooperative spacecraft; and the Orbital Life Extension Vehicle (OLEV) [24], a commercial project with the aim to dock on communication satellites and take over the attitude and orbit control system (AOCS) extending the satellite's lifetime. Some service robots work in automatic operation modes, either pre-programmed or (semi)-autonomous. In complex maintenance tasks, however, or in situations where the cause of the failure is unknown and an investigation is required, the actions cannot be pre-programmed. In such cases, it is necessary to operate the service robot manually via a tele-operation interface. The German Aerospace Center (DLR) already started in 1993 to study ways and the effects of operating a robot in space from a ground station in the Robot Technology Experiment (ROTEX) [10], and later in the Robotic Components Verification on the ISS (ROKVISS) project [15]. In ROKVISS, for the first time a force-feedback joystick has been integrated

Controlling a robot in complex scenarios is not trivial and usually requires a robot expert or a well trained person. Analyzing and repairing a failure in a satellite, however, can only be done by a satellite expert. Hence, for planning and performing appropriate repair tasks both satellite and robot experts have to work closely hand in hand. DLR developed a human-scale bimanual haptic interface [12], enabling natural robot control with the user's arms. This interface can be used to operate a humanoid servicing robot [19] in a way where the user is able to control the robot's head and hands with his/her own head and hands, while seeing the same view as the robot sees through a head-mounted display (HMD), displaying the live video of the stereo-camera built into the robot's head. The combination of the bimanual haptic interface and the humanoid servicing robot creates a tele-presence interface, with which a satellite expert can

1550-6525/11 $26.00 © 2011 IEEE
DOI 10.1109/DS-RT.2011.23

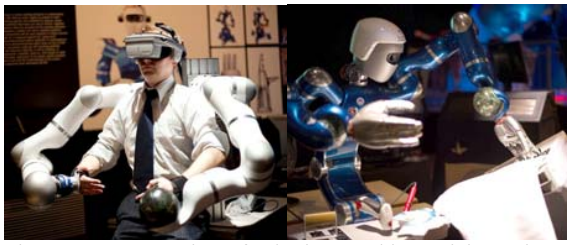

Fig. 1. Demonstration of robotic on-orbit servicing using a tele-presence interface (left) to control a humanoid robot and manipulate a physical satellite mockup (right).

control a robot and intuitively perform servicing tasks without the (direct) help of a robot expert. The setup was used at the International Aerospace Exhibition - ILA 2010 in Berlin to demonstrate tele-operation of a service robot in a number of servicing tasks using a physical test satellite mock-up, see Fig. 1.

Training, programming and testing the tele-operation of robots for OOS tasks may be done in physically based test beds, such as in [1]. This has the advantage that specific hardware and control software to be used during the mission can be integrated directly into the test environment. Certain components or environmental effects, however, have to be simulated, which can make the setup complex. Moreover, testing in a physical environment induces the risk of damage to equipment or even could harm humans. Additionally, making changes to the mockup is often time consuming and connected with costs. Virtual reality (VR) offers a cost-effective alternative for simulation within a flexible and safe environment, and has been used for simulation and training in many application areas, such as surgery.

This paper presents VR-OOS, a virtual reality environment for on-orbit servicing. Our goal is to provide a multi-modal virtual environment that can be used as a platform for the analysis, training and programming of on-orbit servicing tasks, as well as to help to develop and examine new designs of serviceable satellites and servicing robots. The environment will be used to train astronauts in manned servicing missions, as well as to program, and eventually to remotely operate, service robots in space in unmanned missions. The challenges are to provide the accurate real-time simulation of the dynamics of the satellite components under real conditions, combined with photorealistic high-resolution rendering of detailed virtual objects and environmental effects within an immersive, haptics-enabled virtual environment. The paper presents work in progress and our approach to implement the underlying system of the interactive real-time simulation environment using a distributed architecture.

The paper is organized as follows: the next section outlines the selected servicing scenarios that will be used to evaluate the system. Then, an overview of the system components is given. Section 4 outlines the real-time challenges put on the system and Section 5 describes the proposed system architecture. Section 6 discusses preliminary results and Section 7 closes the paper with a conclusion.

## 2. On-Orbit Servicing Scenarios

The repair and maintenance of satellites usually consists of a number of pre-defined sub-tasks that must be executed in a certain order. The goal of the proposed simulation environment is to train the procedure and correct sequence of actions within various on-orbit servicing tasks. In order to support the training of a wide range of possible servicing scenarios, the system must provide a set of basic tasks that often occur and can be combined to various servicing scenarios. We selected a number of tasks that would occur in most servicing scenarios based on common EVAs. These will be used as benchmark for the future evaluation of our system.

**Remove MLI.** As satellites are usually covered by a multi-layer insulation (MLI) foil, the first action would be to open the MLI in order to reach the satellite components underneath. In most cases, the foil will have to be cut using a knife, scissors or other tools and then pulled open. Some satellites have their MLI foil attached by Velcro® fasteners, which can be opened and closed.

**Loosen and tightening screws.** Many parts, such as modules and covers, are fixed with screws. Thus, a second task is to loosen screws, and after the main work tighten them again. This is usually done using a cordless screwdriver.

**Replace a module.** A common task will be to replace electronic parts by exchanging a module. This will be done using a handle that is inserted into a module like a bayonet catch. After rotating the inserted handle by 90 degrees, the module can be pulled out. Inserting a module is done in the same way.

**Flick a switch.** Before removing a module, a task will be to switch off the electronics on the module, and after inserting a new module to switch it on. This is done via a normal lever switch.

**Take measurements.** Finally, a common task is to take measurements at electronic parts. This task is represented using a digital voltmeter and touching specific measurement contacts with the measuring tip.

A milestone of the project is to demonstrate the execution of these tasks within the virtual environment and compare it to interacting with a physical mockup. For this evaluation, the virtual mockup, as seen in Fig. 2 resembles the physical mockup, seen in the foreground in the right image in Fig. 1, in size and arrangement of objects.

The scenarios described above comprise the minimum set

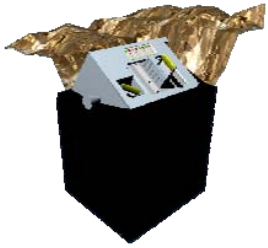

Fig. 2. Virtual satellite mockup.

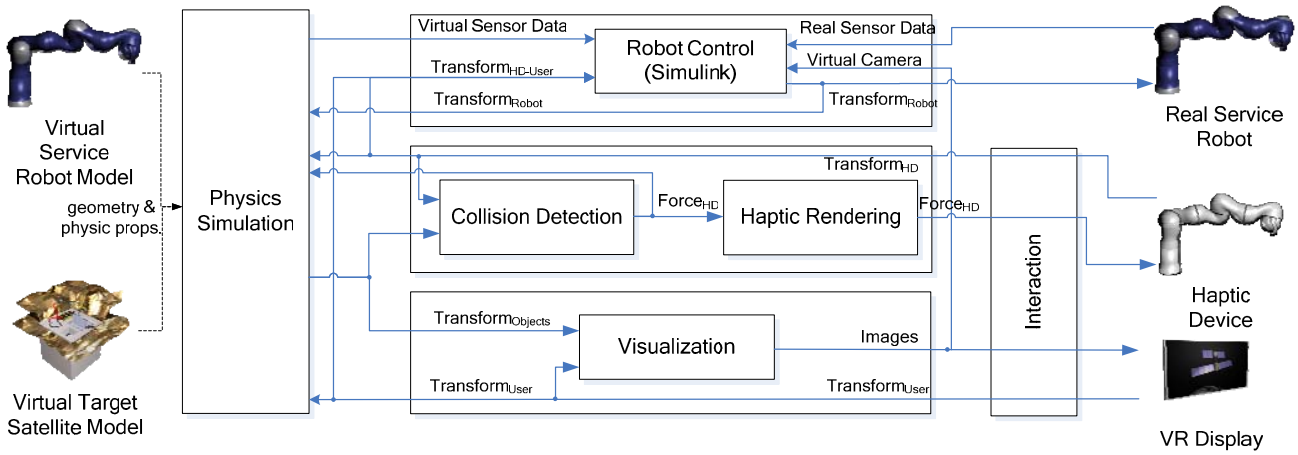


Fig. 3. Block diagram of the system components and the information flow between them.

of tasks that are planned for the first prototype. The final version, will have to allow the simulation of arbitrary tasks, possibly involving the handling of much more complex mechanisms, such as handling bayonet cable connectors, or opening covers with lock-spring mechanisms.

## 3. System Overview

Fig. 3 shows a block diagram of the system components and the flow of information between them. Fundamental components are the physics simulation, visualization, as well as interaction interfaces, which form the base of the virtual environment. The components haptic control and the robot control interface provide additional features for force-feedback and tele-operation, respectively. The following sections describe the components in more detail.

### 3.1. Physics Simulation

A key element of the application is the real-time simulation of the kinematic and dynamic behavior of the satellite components when manipulated by the user. In the first stage of the project, we assume that the servicing robot has already docked to the target satellite, so that the complex flight dynamics of the flying body dynamics can be neglected. For the moment, we only concentrate on the simulation of rigid bodies that make up most of the virtual satellite and robot components. We extend existing models with methods and parameters to match our service scenarios.

Apart from the multi body dynamics of the rigid components of the satellite, an aspect is to simulate the behavior of a foil for the scenario when opening the MLI. This involves modeling the dynamics of bending, cutting or tearing the MLI material.

### 3.2. Haptic Control

The haptic control component generates the necessary data for providing haptic feedback to the user via a haptic device. This includes the detection of collisions between the haptic interaction point (HIP) and any objects within the virtual scene, as well as the computation of the resulting force and torque affecting the HIP. The haptic device delivers the transformation of the end-effector held in the hand of the user, which is used for collision detection. In case the user grasps a virtual object, the complexity of resolving the chain of forces between the interacting objects (virtual scene, grasped object, HIP) can be reduced by attaching the grasped objects with a constant offset to the HIP and apply the resulting transformation based on the HIP's transformation. The resulting force and torque is used to update manipulated objects in the physics simulation.

The force feedback device we use is a human-scale bimanual haptic interface [12], similar to the one depicted in Fig. 4. It is also based on light weight robot (LWR) [11] arms mounted horizontally on a vertical column. It offers a working area similar to human arms and matches that of the humanoid robot "Justin", described above. For software development and experiments, we occasionally use a Phantom Omni from SensAble, which provides only translational force-feedback and no torque.

### 3.3. Robot Control

As mentioned above, we intend to simulate the humanoid service robot "Justin" [19], shown in Fig. 5. It is based on two LWR arms. These are designed similar to the human

Fig. 4. Bimanual haptic interface [12].

arm in its size and working area, aiming at an own weight to payload ratio of at least 1:1. A robot arm has 7 revolute joints, each with motor, gear unit, power supply, force and torque sensors built in. It reaches 0.9 m when fully stretched, weights 13.5 kg, while it can lift up to 15 kg. The robot hands are made human-like too with 4 joints each and house 15 motors. The arms and hands are mounted to a torso, also based on LWR technology, covering a workspace similar to that of a human. The head accommodates a stereo camera, a laser-stripe sensor and an inertial measurement unit.

The controller of the robot dynamics is currently implemented in a number of Simulink® modules. For the simulation, these are accessed via a UDP communication interface. The robot control reads transformations of either the haptic device or that of the tracked user, in case interaction is performed without force-feedback. As output, it delivers the transformation of the robot parts in form of a rotation angle of each robot joint, where the joints are organized in a hierarchical tree and placed at relative positions in the virtual environment. Alternatively, it can deliver the absolute location and orientation of joints when these are organized in a flat hierarchy or a list. This data is used by the physics simulation to update the rigid bodies resembling the robot parts.

Data from the robot's force and torque sensors, as well as data from the other sensors are fed into the control unit. In the case where the robot is not connected for tele-operation but is fully simulated, these sensors have to be simulated in the form of virtual sensors. In this case, the inertial measurement unit and the laser-stripe sensor are emulated within the physics simulation, while the images of the simulated stereo camera are generated within the visualization component.

### 3.4. Visualization

Besides, 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 appearance and arrangement of parts and tools. Hence, the realistic and high-quality visualization of the satellite components and the environment are important factors for the success of a training simulation. This does not only include the photorealistic 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 and hard shadows.

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

Fig. 5. Humanoid service robot "Justin" [19].

### 3.5. Interaction

In the proposed simulation and training system, the user interacts through a VR display, and optionally the haptic device. In order to support the user's feeling of presence in training, the chosen display would preferably be an immersive display, such as a HMD, large screen Powerwall, or a CAVE-like surround projection-based display system. Both, the haptic device and the VR display, track the position and orientation of the user's hands for two-hand interaction. Finger tracking is provided either by using a CyberGlove® when interacting through the bimanual haptic device, or using an optical finger tracking system. Additionally, the user's head is tracked. This is used to render the view in the correct perspective based on the user's current viewpoint. Apart from the manipulation of virtual objects, interaction includes the navigation within the virtual environment, as well as the control and configuration of the running simulation.

Interaction within the environment should be intuitive and easy to adapt for non-expert users. In order to train realistic motion of actions, the interaction interface must be as transparent to the user as possible. If the interaction interface is too complex, for example, it could disturb and hinder the training process, or even train wrong actions. In order to aid the development of effective interaction methods, the proposed system will act as a research platform for studying novel interaction techniques and metaphors.

## 4. Real-Time Challenges

One of the main challenges of the interactive simulation environment is the modeling and simulation of the dynamic behavior of the satellite components and their interactions with the user or the simulated robot in real-time. The computed dynamics and their kinematic effects must reflect the real conditions and behavior of servicing a satellite in orbit. The 3D models used in the simulation environment will be taken from CAD tools from the original construction designs. These are often highly detailed, and not necessarily optimized for use in a VR environment. Moreover, some mechanical mechanisms of satellite components can be quite complex, involving many individual parts. The latching mechanisms of a bayonet cable connector may involve

threaded caps, rings, springs, several pins and holes, for example, where all the linear and angular constraints and friction have to be resolved every simulation cycle. Computing the dynamics of several satellite components or even the complete satellite in real-time is a high target.

The requirement to visually display the results of the simulation in real-time is another challenge. The detailed complex geometric models of the satellite and robot have to be rendered in high resolution and photo-realistic quality at rates of at least 30 Hz. Moreover, for immersive visualization, high responsiveness to viewpoint changes is crucial for the feeling of presence, and if too low can result in motion sickness [18].

A further challenge is the integration of haptic feedback. The haptic rendering of stiff surfaces typically requires sampling rates in the order of 1 kHz to achieve sufficient stability [16]. This puts high requirements on a fast collision detection and force computation, as the geometry of the 3D models in the scene is expected to be complex.

Finally, being an interactive simulation, any responses to actions made by the user within the virtual environment should be displayed with minimum delay. Acceptable end-to-end delay in object-focused tasks is sought to be in the area of 35 ms [7]. Informal experiments have shown that latency for calculating collision responses should be below 25-30 ms, otherwise responses may feel unrealistic [17].

## 5. System Architecture

There were two key aspects involved in designing the system's software architecture. Firstly, the system had to fulfill the demanding computational needs for enabling the real-time simulation of a complex dynamic environment. Secondly, the system had to be flexible and extensible to be able to explore and experiment with different approaches for optimizations of algorithms and state-of-the-art high-performance computing hard- and software, such as supercomputers, PC-clusters, and multi-core and GPGPU programming. Additionally, the system had to provide an interface to add future subsystems. Therefore, the system is implemented in a distributed architecture, where the physics simulation, the haptic rendering and the visualization processes are running as separate modules on dedicated machines, as illustrated in Fig. 6.

In order to aid easy development of alternative module implementations and extensions, the functional structure of a module is made generic, shown in Fig. 7. Each module consists of a communication and a simulation process. Both are running in a separate thread and exchange data via the producer-consumer pattern [8]. The simulation process is the heart of a module. Specific modules implement their individual functionality here. The communication process is responsible for propagating updates between the modules. The current system consists of physics simulation, haptic control, visualization and a manager module. Other modules, such as the robot control module for tele-operation and a module to enable collaboration between remote sites across the Internet within a shared simulation environment, are

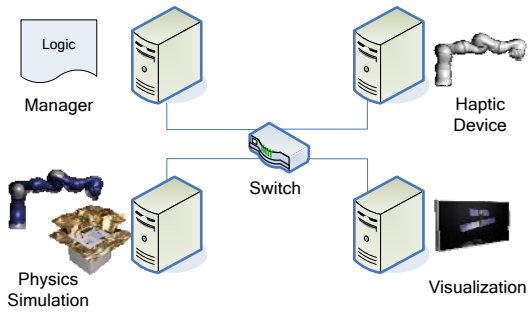


Fig. 6. Hardware setup of the simulation environment (without tele-operation module).

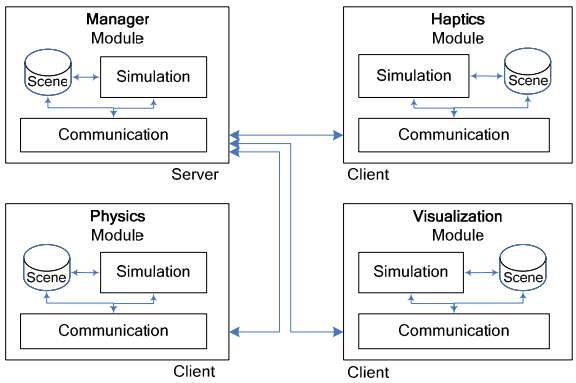


Fig. 7. Logical structure of the modularized system architecture (without tele-operation module).

added in the near future. The manager module is a central component of the system that acts as server, while all other modules are clients that connect to the master and can be added and removed on demand, even at runtime.

Each module manages its own internal representation of the scene. A scene consists of a hierarchy of objects, also called nodes, each with a given state. Common state parameters include at minimum a unique identification string and a transformation matrix to describe the location of the object within the scene. Other information, such as mass, friction, or shading effects, that is specific to a particular module implementation is added to the internal node's state. For example, a physics module would internally represent the scene in a physics world via rigid bodies and constraints, while the visualization module would represent the scene via geometry nodes and shading materials in a scene graph.

All modules implement the same functional structure. Within each processing cycle, a module

- first reads state updates received from other modules;
- interprets the messages and updates the internal scene representation;
- steps the simulation or processes object behavior;
- gathers any state changes and communicates these and any other necessary status messages to the other modules.

### 5.1. Haptics Module

The haptics module is responsible for controlling the haptic device, as well as for the necessary collision detection and force computation to provide force feedback. Discrete collision detection and force computation is implemented using an extension of the Voxmap-Point-Shell™ (VPS) algorithm [21]. The algorithm utilizes two data structures, voxel map and point shell, to represent the solid parts of static objects by volume-based pixels (voxels) and the surface of moving objects by a net of contact points each with a normal pointing inwards, see Fig. 8, left image. This allows to traverse the point shells efficiently to test for intersections with voxels and thus detect collisions. The penetration depth is obtained through the voxel value (Fig. 8, middle) The penetration and the point normal yield a collision force (Fig. 8, right), which is then summed together to compute force and torque. With this algorithm we are able to compute collision responses within the boundary of 1 ms and meet the required 1 kHz update-rate in highly complex scenes.

The haptic module receives the transformations of moving objects and updates the nodes in the internal representation, before starting the collision detection and force calculation in the simulation process. The transformation of the haptic interaction point and the calculated forces and torques of grasped objects are propagated to other modules by the network process.

### 5.2. Physics Module

The simulation of rigid body physics is currently implemented using the open source real-time physics engine Bullet [3], as it provides better overall results compared to other available real-time physics simulation systems [2]. Bullet offers discrete and continuous collision detection and rigid body dynamics including various constraint solvers and generic constraints with support for constraint limits and motors. Soft body dynamics is supported via cloth, rope or deformable object structures.

For accelerated collision detection, simple objects are approximated through basic collision shapes, such as box, sphere, or cylinder, which allow for optimized collision detection. More complex objects are either decomposed into a group of a number of base collision shapes, or their triangle mesh is used directly. In the latter, the physics engine utilizes the efficient Gilbert-Johnson-Keerthi (GJK) algorithm to perform convex collision detection. In future versions, we plan to exploit the collision detection of the VPS algorithm, as used in the haptics module, offering high performance and high accuracy of contacts (depending on configured voxel size), even with very complex geometries.

The relationship between bodies that are part of a mechanical system is defined via physical constraints. In Bullet, constraints are described through joints, such as hinge, slider or generic 6DoF joints. These can be combined with a motor adding linear or angular force to simulate springs. The mechanism of a lever switch, for example, is approximated using two rigid bodies, the base and the handle, connected via a hinge joint. An angular motor is applied to keep the handle to one side. If the handle is moved over its toggle threshold, the motor force is inverted so that the handle snaps to the other side. Other mechanisms are approximated in a similar manner. The use of simplified rigid body dynamics and constraint management does increase simulation speed on one side. However, on the other side, the simulation accuracy will be compromised. Thus, a balance has to be found between simulation speed and fidelity. A solution may be the multi-rate simulation of sub-systems with different performance requirements, as proposed in [13].

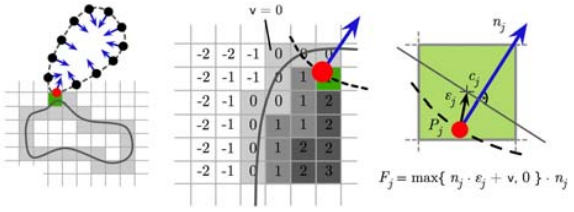

Fig. 8. Principle of the VPS™ algorithm.

The modular design of the system architecture allows us to experiment with alternative implementations of the physics simulation. In the near future, we plan to investigate the suitability of other real-time physics engines, such as PhysX™ and ODE, as well as interface to simulation systems found in the engineering area, such as Simulink® and Modelica®.

### 5.3. Visualization Module

Visualization and interaction via VR displays is provided by the visualization module. It is currently implemented using the VR toolkit ViSTA [20], which offers support for a wide range of VR interfaces and scalable to multi-display technology. The scene is organized in a scenegraph (via OpenSG) that is continuously synchronized with state updates from the physics module and rendered during the simulation process of the module. High-quality rendering is managed through the use of advanced rendering techniques utilizing shaders and high resolution textures.

### 5.4. Manager Module

The manager module hosts the central logic of the system. While the physics module handles the dynamics and kinematics of the individual parts in the simulation, the manager handles the semantics. This includes, for example, monitoring the on/off state of a switch, but also the management of dynamic constraints. For example, in the case of the lever switch, as mentioned above, this would be the change of the angular motor to its inverse if the handle crosses the toggle point. This semantic could of course be implemented within the physics module directly. However, as we wanted a platform to experiment with different implementations of the physics and haptic control, we wanted to remove most of the semantics from the modules in order to simplify their development.

As the physics engine is expected to implement measures for increasing stability, such as through spring and dampers, the manager is responsible for the recognition and management of inter-part geometric constraints between colliding objects. It monitors the result of the physics engine for allowable rigid body motion and intervenes if geometric constraints or semantic states were detected.

Additionally, the manager module provides an interface to the user to control the whole simulation system, such as starting, stopping and resetting the simulation, as well as to record simulation and training sessions for analysis and evaluation.

## 5.5. Scene Description

Besides of importing geometric models from CAD tools, the user needs to specify physics and haptics properties of the virtual objects, as well as their location within the scene and relationships to other objects. Ideally, all this information would go into one central description of the scene. In order to provide a flexible and accessible platform, we need a mechanism for the user to be able to make changes to the scene, load other 3D models and adjust parameters easily, as well as to exchange data with other sites. Hence, we were looking for a suitable file format.

A common exchange format used by many product data management systems in the mechanics and engineering field is STEP [25]. It is also used in the aerospace industry. For example, STEP is used as format for product data exchange between ESA and NASA. However, being a format to cover the complete product life cycle, it is heavy weight, spanning several parts and sub-specifications. Additionally, STEP is rarely supported by common VR toolkits.

COLLADA is a XML-based exchange format managed by the Khronos Group [14]. Since introduced in 2004, it is widely supported in the animation and creative media field and increasingly (used/popular) in engineering and mechanical simulation applications. These days, COLLADA can be exported from many common CAD and 3D modeling tools. The XML schema defines elements to describe 3D asset data, such as mesh geometries, materials, shading effects, animations, and physics properties [4]. We use specification 1.4, as the newer specification 1.5 is not yet fully adopted by most exporters of CAD or 3D modeling tools.

Our COLLADA loader reads a document and stores all data in an internal structured data representation. Each module uses this to obtain the necessary information required to setup their internal representation of the scene. Additionally required information that cannot be described following the COLLADA specification, such as parameters for the haptic properties of nodes, is added via extension nodes using the `<extra>` element.

## 5.6. Communication Layer

A disadvantage of using a distributed system architecture is that the necessary network communication induces delay and other negative effects due to network characteristics, such as jitter and packet loss. As mentioned above, our system adopts a client/server architecture. This means that all state updates are mediated across the manager, rather than sent directly to the target module. The reason for this was the easier management of semantics across modules. However, a consequence is that the delay between interactions and their response is double that of a peer-to-peer architecture. Although, our simulation environment (excluding tele-operation) is generally situated within a local area network (LAN), where round-trip-times of less than 1 ms are typical today, mechanisms to minimize delay should be in place. Our system provides the following features and mechanisms to minimize delay caused by communication:

**Decoupled from simulation.** In order to free the simulation process from message transfer, the communication is implemented in its own threaded network handler. As mentioned above, both processes exchange data via the producer-consumer pattern. The simulation process pushes outgoing messages in a queue and reads pending incoming messages from another queue. The communication process uses these queues to process all pending messages in parallel to the simulation cycle.

**Managed queues.** The majority of messages sent in our system describe discrete updates of absolute force and/or location, such as from rigid bodies. With a continuous stream of such updates, it is preferable to communicate only the latest update, rather than to ensure that every single update is processed. Thus, the queues offer a sorting mechanism to keep only the most recent update of an object.

**Vital and non-vital message types.** As a most-recent sorting method cannot be applied to all types of messages, the system divides them into vital and non-vital messages. Vital messages include system commands or changes to constraint parameters, for example, while non-vital messages all continuous motion and force updates. These are then sent using reliable and unreliable protocols, respectively, via dedicated network channels.

**Optional direct messages.** The default method for distributing a message is to broadcast it via the manager module. In the case where no checks on semantics are necessary, a module is allowed to send a message directly to another module, without passing the manager.

**Loose synchronization.** In order to achieve highest performance of the local simulations in the modules, our architecture provides a loose synchronization mechanism, enabling the simulations to asynchronously run at their full rate without waiting for synchronization. This, however, means that update messages are generated and consumed at different rates. For example, the haptics module will generate updates every millisecond, while the physics module may be able to read them only every 16 ms. If modules would communicate the updates at their local processing rate, this may cause messages queuing up in the network buffers and processing queues. To overcome this problem, the system incorporates an update distribution rate control, where update messages are sent depending on the average simulation processing time of the receiving target

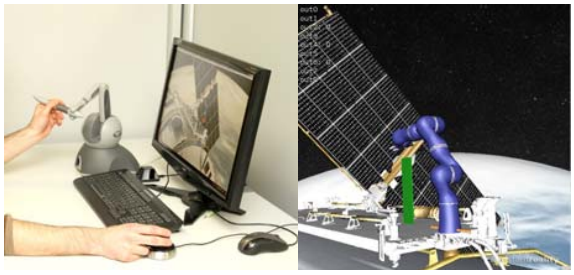

Fig. 9. Evaluating the proposed system architecture with an early prototype using a desktop haptic device (left image). The user moves the endpoint of the LWR attached to a satellite to grab the green object floating in space (right image).

module and the estimated network delay. This allows a module to adjust its rate of sending state changes to match the average simulation rate at the target module.

**Data reduction.** A common measure to reduce traffic for communicating state updates of continuous motion, such as tracking data, is to employ spatial-temporal filtering [22]. Spatial filtering sets a threshold on differences in translation and rotation data before issuing a new update message. This, however, is not useful when subtle movements are important. With temporal filtering, an update is only sent after minimum time between distribution cycles has passed. Our system implements a combined spatial-temporal filtering mechanism on a per object basis. In order to reduce the overhead induced by the communication layer for processing single incoming and outgoing network packets, as well as to increase synchronization, several updates of the same simulation frame can be bundled into one message, as long the resulting data fits into a single packet. Further optimizations applied in tele-presence systems, such as perception-based data reduction of transmitted haptic data based on just noticeable differences [9], may be investigated in future versions.

## 6. Preliminary Results

Work is currently in progress for implementing the system architecture described above. So far, we have implemented the mechanisms of three scenarios: flick a switch, loosening and tightening screws, and removing a module using a bayonet handle. First tests in a desktop setting interacting with a Phantom Omni® haptic device have been conducted to evaluate the proposed system architecture, Fig. 9.

In an informal experiment, we measured the end-to-end delay for displaying haptic interaction. This is the time from generating a message in the haptics module signaling a state change in the position of the HIP until receiving a response message at the visualization module. This includes the delay induced by the manager module passing the message on to the physics module, which is running a simulation cycle and passes a new update message with resulting positions of the rigid bodies to the manager, that forwards the message to the visualization module. Fig. 10 shows the measurement result. We find an average end-to-end delay of 28 ms. Thus, our system performs within the bound of the required 35 ms.

Fig. 11 shows the number of update messages for mediating the response to changes of the HIP through the system. We measured an average throughput of 61 update messages per second. This corresponds to the described loose synchronization measure, which keeps the haptics module from sending update messages faster than the physics simulation rate. This was set to 60 Hz during the measurements.

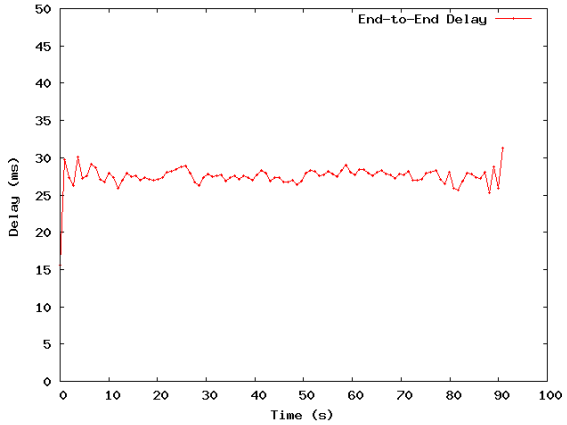


Fig. 10. End-to-end delay.

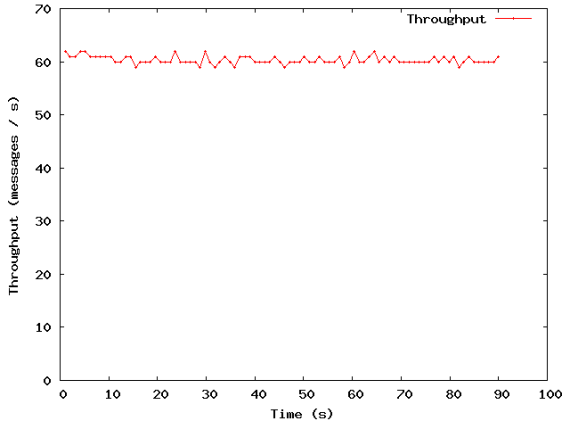


Fig. 11. Throughput.

## 7. Conclusion

This paper gave an overview of a real-time interactive simulation and training environment used as a platform for the analysis, training and programming of on-orbit servicing tasks, as well as our work in progress developing the underlying infrastructure. The aim is to allow performing manual and robotic assembly and disassembly tasks within a haptic-enabled immersive virtual environment. The main challenges are the real-time simulation and rendering of the correct dynamic behavior and the realistic appearance of satellite components and their complex mechanisms resembling the real conditions in space, as well as to integrate interaction through a haptic device. The paper proposes a distributed system architecture in order to utilize dedicated computing resources to fulfill the real-time

constraints. The architecture divides the physics simulation, visualization and haptic rendering into separate modules that run in parallel on dedicated machines. A central manager mediates the communication of state updates, while managing the global semantics of object behaviors. Preliminary results have shown that our system is able to provide an end-to-end latency across the modules of 28ms for sending updates from moving the haptic device to displaying the visual response.

The modular design of the proposed architecture facilitates the research and development of improvements and extensions with focus on individual aspects of the simulation environment. Future work will investigate alternative real-time physics engines, develop optimizations to the haptic rendering and explore two-handed interaction techniques.

## References


[1] J. Artigas, P. Kremer, C. Preusche, G. Hirzinger, "Testbed for telepresent on-orbit satellite servicing", *In Proceedings of the Human-Centered Robotic Systems Conference (HCRS)*, Munich, Germany, 2006.

[2] A. Boening, T. Bräunl, "Evaluation of real-time physics simulation systems", *In Proceedings of the 5th international conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE)*, pp. 281-288, Perth, Australia, 2007.

[3] http://bulletphysics.org, 2011.

[4] E. Coumans, K. Victor, "COLLADA physics", *In Proceedings of the 12th International Conference on 3D Web Technology*, pp. 104-105, 2007.

[5] N.J. Currie, B. Peackock, "International Space Station Robotic Systems Operations - A Human Factors Perspective", *In Proceedings of Human Factors and Ergonomics Society Annual Meeting*, Human Factors and Ergonomics Society, pp. 26-30(5), 2002.

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

[7] S. R. Ellis, M. J. Young, B. D. Adelstein, & S. M. Ehrlich, "Discrimination of Changes of Latency During Voluntary Hand Movement of Virtual Objects", *In Proceedings of the Human Factors and Ergonomics Society*, Houston, Texas, 1999.

[8] E. Gamma, R. Helm, R. Johnson, J.M. Vlissides, *"Design Patterns: Elements of Reusable Object-Oriented Software"*, Addison-Wesley Professional, 1994.

[9] P. Hinterseer, S. Hirche, S. Chaudhuri, E. Steinbach, M. Buss, "Perception-Based Data Reduction and Transmission of Haptic Data in Telepresence and Teleaction Systems", *In IEEE Trans. on Signal Processing*, Vol. 56, Issue 2, pp.: 588-597, 2008.

[10] G. Hirzinger, B. Brunner, K. Dietrich, J. Heindl, "ROTEX - the first remotely controlled robot in space," *in IEEE Intl. Conf. on Robotics and Automation*, San Diego, CA, USA, 1994.

[11] G. Hirzinger, N. Sporer, A. Albu-Schäffer, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot III - are we reaching the technological limits now?", *In IEEE Int. Conf. on Robotics and Automation (ICRA2002)*, pp. 1710–1716, Washington, DC, USA, 2002.

[12] T. Hulin, M. Sagardia, J. Artigas, S. Schaetzle, P. Kremer, C. Preusche, "Human-Scale Bimanual Haptic Interface", *In Proc of the 5th International Conference on Enactive Interface*, Pisa, Italy, 2008.

[13] M. Karkee, B.L. Steward, A.G. Kelkar, Z.T. Kemp II., "Modeling and real-time simulation architectures for virtual prototyping of off-road vehicles", Virtual Reality, Springer-Verlag London, UK, pp. 83-96, 2011.

[14] http://www.khronos.org/collada, 2011.

[15] K. Landzettel, A. Albu-Schäffer, B. Brunner, A. Beyer, R.Gruber, E. Krämer, C. Preusche, D. Reintsema, J. Schott, B.-M. Steinmetz, H.-J. Sedlmayr, G. Hirzinger, "ROKVISS Verification of Advanced Light Weight Robotic Joints and Tele-Presence Concepts for Future Space Missions", *In Proceedings of the 9th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA)*, ESTEC, Noordwijk, Netherlands, 2006.

[16] W.R. Mark, S.C. Randolph, M. Finch, J.M. Van Verth, and R.M. Taylor, II. Adding force feedback to graphics systems: Issues and solutions. *In ACM SIGGRAPH Computer Graphics and Interactive Techniques*, pp. 447-452, New Orleans, Louisiana, 1996.

[17] J. Marsh, M. Glencross, S. Pettifer, R. Hubbold, "A Network Architecture Supporting Consistent Rich Behavior in Collaborative Interactive Applications", *In IEEE Trans. on Visualization and Computer Graphics*, Vol. 12, No. 3, pp. 405-416, 2006.

[18] M. Meehan, S. Razzaque, M. Whitton, and J. Brooks, F.P., "Effect of latency on presence in stressful virtual environments", *In Proceedings of IEEE Virtual Reality*, pp. 141-148, 2003.

[19] C. Ott, O. Eiberger, W. Friedl, B. Bäuml, U. Hillenbrand, Ch. Borst, A. Albu-Schäffer, B. Brunner, H. Hirschmüller, S. Kielhöfer, R. Konietschke, M. Suppa, T. Wimböck, F. Zacharias, and G. Hirzinger, "A Humanoid Two-Arm System for Dexterous Manipulation", *In Proceedings of IEEE-RAS International Conference on Humanoid Robots*, pp. 276-283, Genova, Italy, 2006.

[20] T. van Reimersdahl, T. Kuhlen, A. Gerndt, J. Henrichs, C. Bischof, "ViSTA: a multimodal, platform-independent VR-Toolkit based on WTK, VTK, and MPI", *In Proceedings of the 4th International Immersive Projection Technology Workshop (IPT)*, Ames, Iowa, 2000.

[21] M. Renz, C. Preusche, M. Potke, H. Kriegel, G. Hirzinger, "Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm", *In Proceedings of the Eurohaptics*, pp. 149-154, Birmingham, UK, 2001.

[22] D. Roberts, D. Marshall, S. McLoone, D. Delaney, R. Aspin, "Exploring the use of local inconsistency measures as thresholds for dead reckoning update packet generation", *In IEEE Distributed Simulation and Real-Time Applications*, pp. 95-202, Montreal, Canada, 2005.

[23] T. Rupp, T. Boge, R. Kiehling, F. Sellmaier, "Flight Dynamics Challenges of the German On-Orbit Servicing Mission DEOS", *21st International Symposium on Space Flight Dynamics*, Toulouse, France, 2009.

[24] F. Sellmaier, T. Boge, J. Spurmann, S. Gully, T. Rupp, F. Huber, "On-Orbit Servicing Missions: Challenges and Solutions for Spacecraft Operations", *SpaceOps 2010 Conference*, American Institute of Aeronautics and Astronautics, Huntsville, Alabama, USA, 2010.

[25] STEP, Standard for the Exchange of Product model data, ISO 10303.