SIMULATION OF ON-ORIBIT-SERVICING TASKS USING A MOBILE MANIPULATOR AND A PHYSICS SIMULATION

Steffen W. Ruehl, Andreas Konle, Arne Roennau, and Ruediger Dillmann

FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany

ABSTRACT

We present a testbed for robotic manipulation of modular satellites in micro gravity. Therefore, we combine a real-time physics simulation with a mobile manipulator. The system is able to execute the motion of a simulated object in a micro gravity environment in the high gravity environment of a lab on earth. Using a commercially available simulation software, we are able to account for aspects such as obstacles and friction, which are not modeled in existing zero gravity motion emulation systems. An experimental evaluation is presented.

Key words: On-orbit-satellite-servicing, modular satellites, robotics, manipulation, simulation, micro gravity.

1. INTRODUCTION

Modular satellites [9] may contribute to the efficiency of satellite operation by reducing the amount of payload which has to be launched into space. Instead of replacing satellites with broken components, only the module containing the affected component is replaced. Anyhow, if those tasks were carried out by human astronauts, the high costs of launching humans into space would cancel out the before mentioned benefits. Thus, in order to exploit the advantages, robotic manipulation has to be employed.

A visualization of an servicer is displayed in Fig. 1. To develop modular satellites and suitable robotic servicers, intensive testing is required. Again, high costs of space missions forces us to conduct those tests on earth. Conditions close to the micro gravity in space must be provided. A common mean of the state of the art are simulation environments. Pure software systems simulate the motion of multi body systems with respect to acting forces and torque. Gravity, for example, may be set simply as a parameter. Anyhow, those systems are limited to simulating the aspects described by their models. In real world environments, different aspects influence the outcome of robotic manipulation activities; influences of which system designers may not be aware and which should be discovered during the experiments. On the other hand, robotic systems for the emulation of zero gravity exists and are used for testing satellite rendezvous operations.

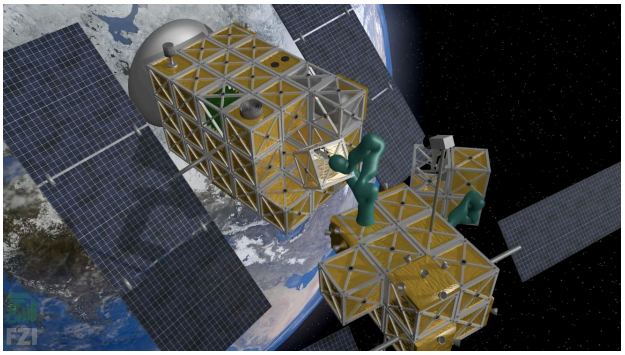

Figure 1. Visualization of a robotic on-orbit-servicing sezanrio.

In this paper, we propose to combine such a robotic simulator with a real time physics engine. Thus, we are able to exploit the ability of existing software simulation systems to simulate a large variety of physical aspects under zero gravity in the large scale. We are also able to easily add new objects or simulations aspects as they are provided by external parties. In the small scale we can use the real hardware with the effects caused in the real world, which are not covered by the simulation but are known to be critical for successful robotic manipulation.

In order to implement such a system, a robust sensor interpretation of the force torque sensor is developed. It is able to account for errors caused by gravity and the fact, that the object is attached to a manipulator. We further address the problem of mapping the generated Cartesian trajectories back to executable, continuous trajectories in the robot's joint space.

The paper is structured as followed: In Sec. 2 we will describe the state of the art in the field of real-time physics simulation and simulation systems for motion simulation under zero gravity on the ground. Our approach is described in Sec. 3. An experimental evaluation of the implemented system is presented in Sec. 4. Finally, Sec. 5 gives conclusions and an outlook.

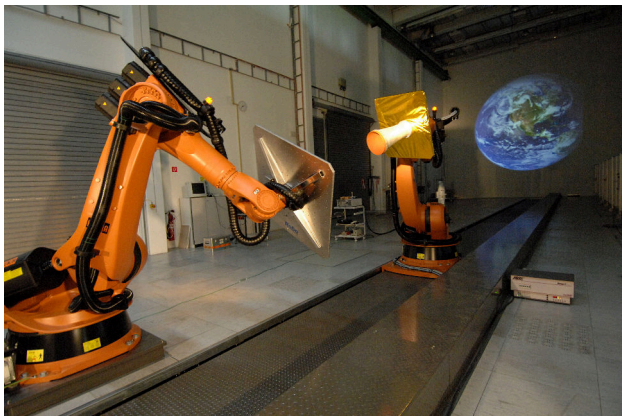

Figure 2. DLR's EPOS simulation system while simulating a orbital rendezvous. Two six axis industrial manipulators provide six dimensional motion. An additional axis extends the workspace to simulate the approach. Source: [8] (DLR CC-BY 3.0).

2. STATE OF THE ART

Real-time simulation systems for mechanical physics simulation originate from the gaming domain and are wide spread today [3]. Also, the application as an efficient real time testbed for robotic algorithm is know for a long time. E. g. the gazebo [5] system has been developed for more than 10 years. Today it is integrated in the Open Source "Robot Operating System" ROS[1] which has become a standard tool in the robotic community. Commercial solutions are available as well, like the Verosim[2][7] software. Although those systems are quick and efficient, simulation is always limited to the simulated aspects. In case of mechanical simulation, object models are often simplified or mechanical properties of surfaces may be unknown or estimated. Effects such as thermal expansion are usually neglected.

Another approach, which is used in the evaluation of space technology to overcome the problem of gravity in lab experiments on earth is the calculated movement of objects by a robotic manipulator. Such a system is DLR's European Proximity Operations Simulator (EPOS) [4, 8], which simulates satellite rendezvous operations as shown in Fig. 2. A Similar system, which is able to emulate the motion of free floating objects and orbital motion is used by the Canadian Space Agency[1]. Those systems use specific motion models, which enable them to precisely calculate the resulting trajectories. Force torque sensors enable them to react to external forces, exerted on the object. As drawback, those systems are specialized on the implemented task an cannot easily be extended with new functionality such as collision checking with virtual objects.

[1] http://www.ros.org/
[2] http://www.verosim-solutions.com/

Figure 3. The robot demonstrator holding a satellite module.

3. SIMULATION SYSTEM

The proposed system creates a loop from the sensor readings of the forces and moments acting on the Tool Center Point (TCP) or the attached objects. Those readings are interpreted in a preprocessing step and passed to the simulation software. There the forces are applied to the manipulated object and resulting object trajectories are generated. Those are mapped to joint trajectories for the manipulator. After the interpolation, those trajectories are executed by the robot. The execution in the real world closes the loop.

3.1. Hardware setup

The hardware of the developed system consists of two Kuka Light Weight Robots (LWR) [2] on a Kuka Omnirob mobile platform (see Fig. 3). The robot is depicted in Fig. 3. The Kuka LWR has seven degrees of freedom (DOF) and an integrated impedance control. This can also be parametrized to generate a gravitation compensation for the arm and attached objects. Anyhow, it is intended for easy robot control. It's accuracy is not sufficient for zero g simulations.

The LWR has torque sensors in its joints, which are used to estimate the forces acting on its TCP. It uses a dynamic model of the arm in order to compensate for forces caused by gravity and inertia. This model is completed by the dynamic model of the object, so causes by the same effects on the objects are compensated as well.

The control interface of the robot system allows us to command the robot configurations and reading its sensor values in real-time. An embedded PC with an Intel Dual-Core i7 -620M and 8 GB of RAM is integrated in the robot and used for the control. The physics simulation is executed on an external workstation.

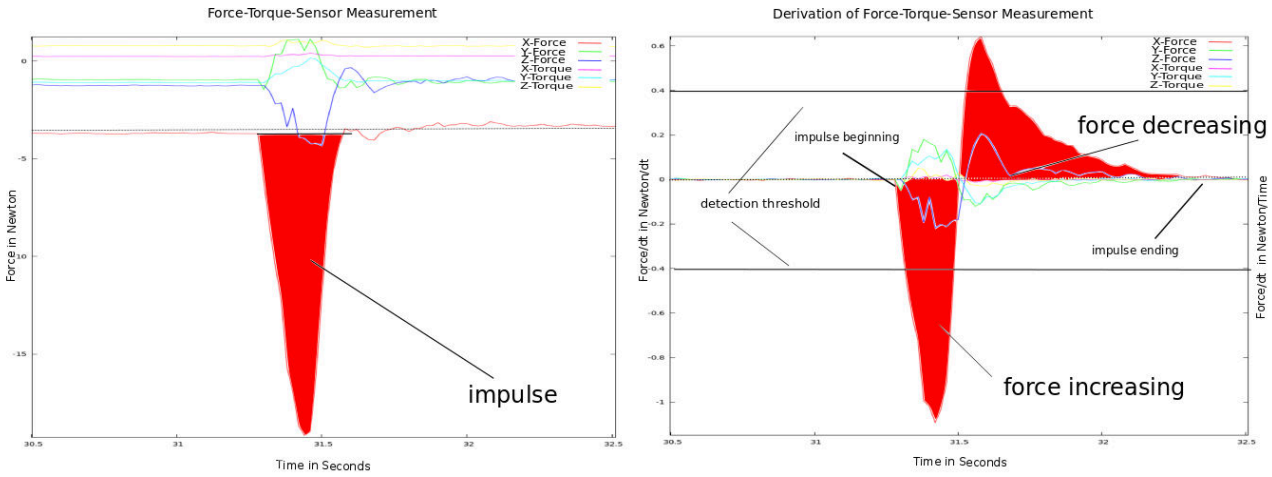

Figure 4. Plot of the measurements of a single impulse in x direction (red). On the left hand side, the input value is shown. The executed impulse is visualized in red. The impulse can be recognized as well as some left noise and a small offset between the null line and the measured values during the absence of an impulse. It is also visible, that a force in the opposing direction is measured after the impulse (not marked in red). On the right hand side, the derivation of the smoothed measurement is plotted in red.

3.2. Force measurements

The force measurements are subject to sensing errors. Since the forces at the TCP are not measured directly at the TCP, but estimated from torques at the joints of the robot, they are also subject to model errors. While the integrated sensors provide sufficient accuracy for many robot control tasks on ground, the integration of forces in the simulation environment will add up systematic errors, leading to undesired motion. Further errors are introduced by a bouncing effect of the robot and the object. To overcome those errors, we use a pre-processing step and on top of that an interpretation based on an impulse model of exerted forces.

To overcome the sensor error, we use the following error model:

$$w = w_{\text{real}} + e_{\text{noise}} + e_{\text{const}} \tag{1}$$

w is the measured wrench vector at the TCP. Two error vectors are added to the real real acting wrench w_{real} :e_{noise} is a component wise normally distributed noise vector. e_{const} is a constant offset.

The noise can be reduced using a low pass based on exponentially smoothing:

$$w_t^* = \alpha \cdot w_{t-1}^* + (1-\alpha) \cdot w_t \tag{2}$$

where the current wrench w_t^* is calculated by the mean of the current measurement w_t and the previous filtered wrench w_{t-1}^*, weighted by the factor α.

Of more concern is the error e_{const} which, when passed into the simulation, causes a drift of the velocity of the simulation. This error is caused by errors in the dynamic model of the object and the manipulator. This error is constant in the static case, but changes when the configuration of the manipulator changes. In our scenario, we are not interested in high speed motions, we can assume it to be constant, leading to:

$$e'_{\text{const}}(t) \approx 0$$

Thus, we can eliminate e_{const} by deriving w_t^*

$$(w_t^* + e_{\text{const}})'(t) = (w_t^*)'(t) + e'_{\text{const}}(t) = w_t^* = f(t) \tag{3}$$

which loses the absolute information about the values, but is free of the considered errors and is used for the interpretation.

A plot of w_t^* is displayed in Fig. 4. An impulse leads to a peak in the measurement which is followed by a peak in the opposing direction. Since we are looking at free floating objects, an impulse will cause the object to move away from the impulse, thus it will loose the contact and no further forces can act on the object. Thus, we can assume that forces always act in the form of impulses. This is used for filtering by detecting those impulses and only passing them to the simulation.

An impulse is detected, when the derivation of w_t^* reaches a threshold. It ends, when $|w_t^*|$ reaches a minimum. The following counter impulse is ignored, until w_t^* falls under a threshold.

The value of an impulse within the detection can now be calculated by integrating

$$\vec{I}[t_s, t_e] = \sum_{i=t_s}^{t_e} (w_t^*), \text{ thus } \sum_{i=t_s}^{t_e} \Delta t * \sum_{j=t_s}^{t} (w_t^*)' \tag{4}$$

In this equation, t_s is the index of the first measurement of the impulse, t_e the index of the last measurements. Further optimizations are applied to account for real time effects.

The resulting detected impulses are passed to the simulation.

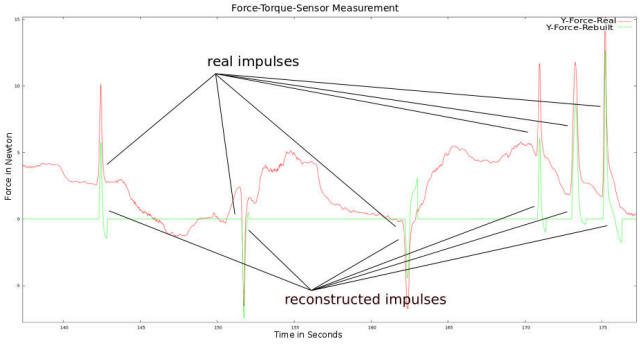

Figure 5. Calculated impulses. The measured force is plotted in red, the reconstructed impulses are visualized in green color.

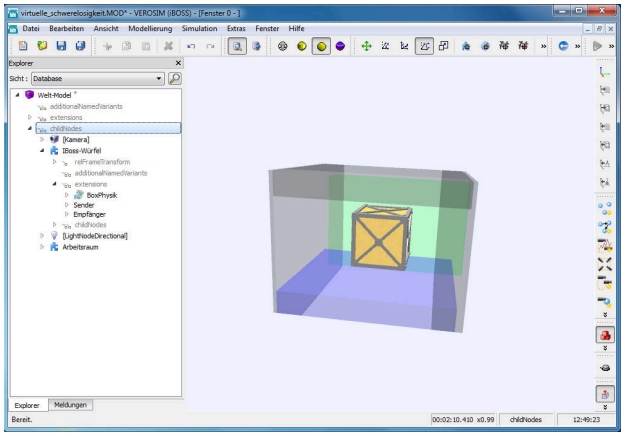

Figure 6. Screenshot of the virtual test-bed Verosim. A visualization of a satellite module is displayed. Its motion is limited by a set of boxes, which form a bounding box.

3.3. Real-time physics simulation

We use the commercially available real-time physics simulation Verosim as simulation environment. In order simulate the physics of a scene, object models are required which cover the following aspects: The geometry is required to detect collisions. The geometric model may be extended with textures for visualization. Friction coefficients of the object's surface is used. Mass and inertia of an object is used to simulate the effect of an impulse on it, as well as its stiffness and damping parameters. The evaluation scenario of this work in Verosim is displayed in Fig. 6.

Only the objects which are in virtual zero gravity and virtual objects are modeled in the simulation. Real objects, such as the operator or manipulator, which exert the impulse on the floating objects are not simulated.

Depending on the constraints of the scenario, impulses can be applied to the object in different ways. In real-time systems, in in each time step where an impulse is detected, this impulse is applied to the object. Alternatively, the impulse is accumulated and applied to the object as one impulse. This enables the processing described in the previous section.

The use of simulation enables us not only to consider the current state, but also to adapt applied impulses in the past. Thus, we can use the single impulse approach in real-time. This leads to short term errors, since impulses do not cause an effect exactly when they emerge. Anyhow, within 0.05 s, this error is corrected and the system behaves as if the impulse was given at the time it occurred.

We also use this feature to create a prediction of the motion, assuming, that no further impulses occur. The predicted trajectory for a short time in the future is passed to the trajectory generation.

3.4. Trajectory generation

The trajectory of the simulated object has to be mapped from Cartesian space to the joint space of the motion executing manipulator. The resulting trajectory has to be smooth in order to provide a realistic execution. Thus, configuration changes have to be avoided. Configuration changes are caused by ambiguities of the inverse kinematics (IK), joint limits and singularities.

Using th IK, a single pose of the object can be mapped to a manipulator position. Since the LWR has seven DOFs, it is over actuated. For each pose P_i, there exist α_i solutions. In theory, for a over actuated system, α_i may be infinite. The IK uses sampling to map this to a finite set of possible solutions $C_i = (c_{i,0}, c_{i,1}, \cdots, c_{i,\alpha_i})$. Using the prediction of the object's pose form the physics simulation, we can define the matrix of possible configurations in the predicted future:

$$P = \begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_n \end{pmatrix} = \begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,\alpha_0} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,\alpha_1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,0} & c_{n,1} & \cdots & c_{n,\alpha_n} \end{pmatrix} \quad (5)$$

where n is the number of predictions. Finding a smooth trajectory is now reduced to the problem of finding a sequence $S = (s_0, \cdots, s_n)$ so that the trajectory $T = (c_{0,s_0}, c_{1,s_1}, \cdots, c_{n,s_n})$ contains no configuration changes.

To find such a sequence, we will minimize a distance function between subsequent configurations. For a pair of two configurations $c_1 and c_2$, we define

$$d(c_1, c_2) = ||\vec{c}_1 - \vec{c}_2||_\infty = || \begin{pmatrix} \Theta_{1,0} - \Theta_{2,0} \\ \Theta_{1,1} - \Theta_{2,1} \\ \vdots \\ \Theta_{1,6} - \Theta_{2,6} \end{pmatrix} ||_\infty \quad (6)$$

where $\Theta_{a,b}$ describes the the value of joint b in c_a. Using the infinity norm yields to the maximum distance be-

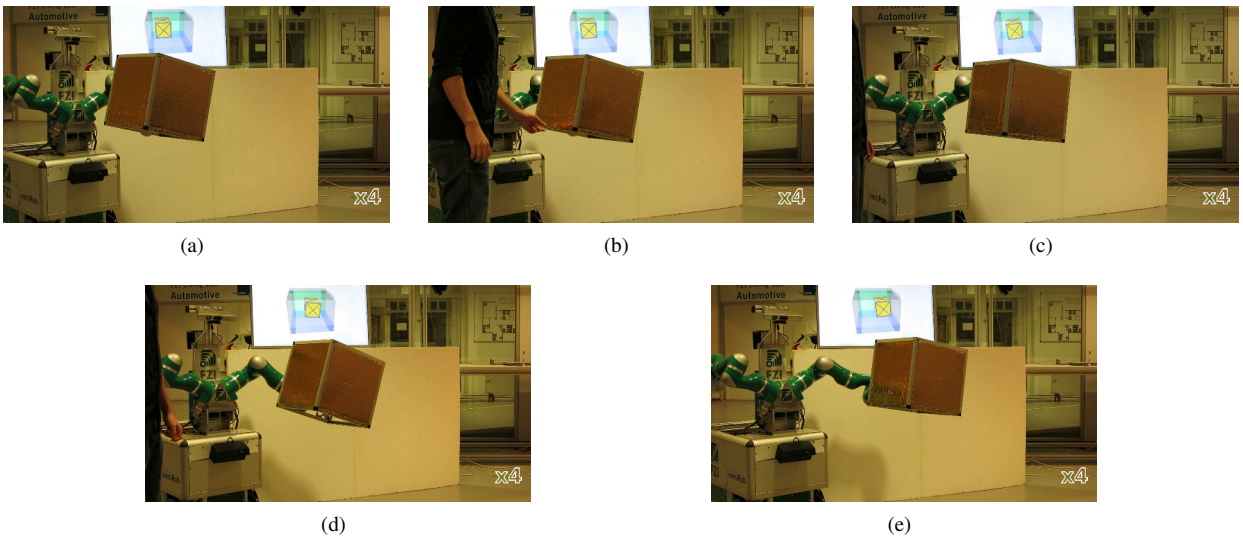

Figure 7. Experimental evaluation of the system. A model of a satellite cube is attached to the manipulator. A visualization of the simulation can be seen in the background. An initial impulse is given to the object in simulation and causes a motion (a). The operator pushes the cube (b), causing an adaption of the motion (c), (d). Finally, the simulated cube hits the wall of the surrounding box and changes the direction of the motion again (e).

tween two joint values. Finding a solution S_o with minimal configuration changes now is reduced to minimizing the squared sum of ratings of each subsequent pair of configurations:

$$S_o = \arg\min_{s} \sum_{i=1}^{n} d(c_{i-1,s_{i-1}}, c_{i,s_i})^2 \qquad (7)$$

This problem can be solved by dynamic programming. Since the time steps are equal sized, the solution also minimizes the maximal speed of the joints.

To map the time discretization of the object trajectory to the robot controller, a fine interpolation is required. The trajectory changes during the motion of the robot. The robot dynamics constrain the maximal velocities and acceleration. Thus, the interpolation has to map the kinematically defined trajectory to a dynamically valid. We use the Reflexxes Library [6], which solves this problem. The resulting configurations are executed by the manipulator in real-time.

4. EXPERIMENTS

First experiments to evaluate the system and its components have been conducted on the hardware platform described in Sec. 3.1.

Experimental results of the impulse detection are shown in Fig. 5. It is visible, that the filtered measurements are still subject to some noise. The detection of impulses are necessary. The reconstruction is able to move the baseline of the reconstructed signal to the zero line, thus the constant error is eliminated. The impulse detection is also able to cut off errors caused by the changing configuration of the system.

Currently, only the manipulator is used, the mobile platform is not integrated. Virtual obstacle have been introduced to limit the simulated space to the workspace of the robot (see Fig. 6).

In a first experiment, we examine the trajectory generation. The workspace is limited as described above, no external forces are applied. The physics simulation moves the cube bouncing between the borders of the workspace, creating linear and angular motions. The trajectory generation generates trajectories, which mimic those motions smoothly. The kinematic manipulator limits the angular location of the cube, since the fixture has to point towards the robot.

In a second experiment, external forces are applied by an operator (Fig. 7). Qualitative, they produce effects as desired. Anyhow, a delay between impulse and reaction of the system was detected. This delay is about 300 ms. When an impulse is applied, the system thus does not exactly reflect the free desired floating behavior.

5. OUTLOOK

Two main aspects will be improved in the system. First, the mobile platform will be used to enlarge the workspace of the simulation. This will allow to simulate more complex tasks and also allow orientations of the object, which are currently kinematically infeasible. Anyhow, this will require synchronous control of the manipulator and the platform, which may influence the latency of the system.

Beyond that, the response time of the system has to be reduced. The main share (250 ms) of the response time is caused by the robotic control interface. Simulation and IK are about 15 ms. Different modes to command the manipulator are available and will be integrated in the future.

Further experiments with a second manipulator will be carried out, where the robot will interact with the floating object, e. g. grasp it. These experiments will illustrate the benefits of the combined system.

REFERENCES

1. Farhad Aghili. A robotic testbed for zero-g emulation of spacecraft. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3654–3661, August 2005.

2. Rainer Bischoff and Johannes Kurth. The Kuka-DLR Lightweight Robot Arm - a New Reference Platform for Robotics Research and Manufacturing. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 741–748, Munich, Germany, 2010.

3. Adrian Boeing and Thomas 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*, pages 281–288, 2007.

4. T Boge, T Wimmer, O Ma, and M Zebenay. EPOS-A Robotics-Based Hardware-in-the-Loop Simulator for Simulating Satellite RvD Operations. In *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 267–274, 2010.

5. N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator, 2004.

6. Torsten Kroger. Opening the door to new sensor-based robot applications - The reflexxes motion libraries. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6–9, 2011.

7. Juergen Rossmann, Thomas Josef Jung, and Malte Rast. Developing virtual testbeds for mobile robotic applications in the woods and on the moon. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 4952–4957, 2010.

8. F Sellmaier, T Boge, J Spurmann, S Gully, T Rupp, and F Huber. On-Orbit Servicing Missions: Challenges and Solutions for Spacecraft Operations. In *SpaceOps 2010 Conference*, number April, 2010.

9. J Weise, K Brieß, A Adomeit, H.-G. Reimerdes, M Göller, and R Dillmann. An Intelligent Building Blocks Concept for on-Orbit-Satellite Servcing. In *i-SAIRAS: International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2012.