ROBOTIC VERIFICATION OF SPACECRAFT RENDEZVOUS IN-LOOP WITH REAL-TIME SATELLITE DYNAMICS SIMULATION

F. Rems, German Aerospace Center (DLR)/German Space Operations Center (GSOC), Münchener Straße 20, 82234 Weßling, Germany

Abstract

The complexity and reliability needs of space rendezvous, especially with uncooperative targets in typical On-Orbit-Servicing scenarios like Active Debris Removal, require sophisticated Hardware-in-the-Loop simulation on ground for development support and verification. This has to include real sensor hardware, the real rendezvous GNC algorithms as well as real target surface materials. Such complex closed-loop rendezvous simulations involve numerous interacting components, which leads to various problems and poses an enormous challenge to the overall system. The European Proximity Operations Simulator (EPOS), located at GSOC, DLR, in Oberpfaffenhofen near Munich, is the perfect tool for coping with these challenges. It is a robotic Hardware-in-the-Loop simulator for the final 20 meters of rendezvous maneuvers. It consists of two industrial robots, one of which is mounted on a linear rail. It can handle true-to-scale satellite mockups and provides realistic illumination conditions via a powerful daylight spotlight. This paper describes the strongly upgraded EPOS control system. Raising EPOS simulation capabilities to a totally new level, the new features include: collision avoidance; real-time robot joint optimization to avoid robot singular configurations and other laboratory constraints; flexible connection handling with remote command sources; a convergence algorithm for seamlessly reaching dynamic starting conditions; flexible mapping of commanded satellite positions given in any arbitrary coordinate system into the EPOS laboratory. The enhancements transform EPOS into a “Plug-and-Play” facility that can easily handle complex Hardware-in-the-Loop rendezvous simulations with many components in a robust and flexible manner. The paper concludes with a closed-loop simulation example of a typical uncooperative approach.

1. INTRODUCTION

Automated rendezvous is a key spaceflight technology.

The Automated Transfer Vehicle (ATV) has demonstrated successfully that automated rendezvous with the ISS, a cooperative target, is state of the art [1]. In the context of rendezvous, a cooperative target is an operational space vehicle. Its precise location is known, its attitude is stabilized, communication is working properly, and it may even have navigation aids on its outer surface like markers or retro-reflectors.

However, automated rendezvous awaits new challenges in the near future. The number of debris objects in space has increased considerably. This debris threatens the future of spaceflight as we know it. The collision if Iridium 33 und Kosmos 2251 has demonstrated impressively the severe consequences [2] [3] [4] [5]. It has proven the need for actively removing the debris from orbit. One option is having a chaser spacecraft rendezvous with some debris object, grasp and deorbit it safely [6]. Another challenge for automated rendezvous lies in the field of maintaining satellites in-orbit, thereby extending their operational lifetimes and increasing return on investment [7] [8]. And finally, automated rendezvous can help to assemble large structures in-orbit, like large aperture telescopes or even spaceships heading for the outer planets [9] [10] [11].

In these cases, automated rendezvous has to handle uncooperative targets. There is most likely no communications and no precise information about the target‘s location or status. Without any attitude stabilization, the target object may be tumbling. Under these circumstances, the chaser spacecraft has to actually look for the target with visual sensors. It uses the sensor data to estimate the target object‘s relative position and attitude. Based on that information, the chaser can approach its target precisely and safely [12] [13].

However, the environmental conditions make this kind of rendezvous a highly complex and difficult scenario. The target‘s surface may be composed of a mixture of diffuse and highly reflective materials, without any markers or retro-reflectors dedicated to support rendezvous navigation. The chaser faces varying and extreme illumination conditions. From total darkness, over sharp shadows on the target, to full sunlight, anything is possible. And the chaser has to execute the rendezvous manoeuvre with the highest reliability possible. A collision with the target has to be prevented at all cost.

Considering the complexity and required reliability of automated rendezvous, on ground simulation is mandatory. However, pure software simulation is not enough. The physical effects of chaser sensor hardware and target surface materials affected by sunlight are far too difficult to simulate in software accurately. The real hardware has to be used as far as possible.

The European Proximity Operations Simulator (EPOS) is a tool to do just that. Its main components are two industrial robots, one mounted on a linear rail. In a typical rendezvous simulation scenario, one robot carries a mockup of the target satellite, possibly true-to-scale with

realistic surface materials. The other robot carries visual navigation sensors, representing the chaser spacecraft. A 12 KW spotlight realizes realistic illumination conditions [14] [15].

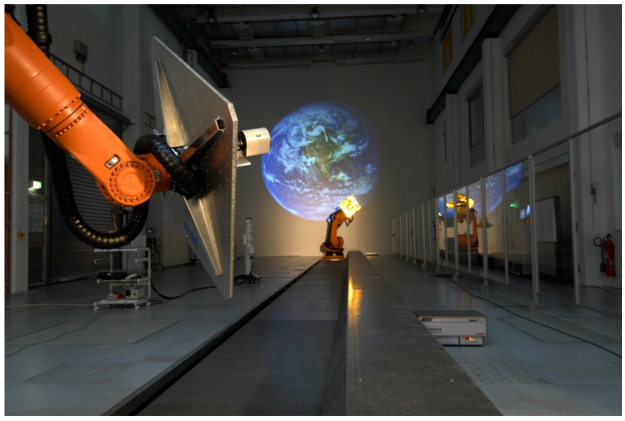

FIGURE 1. Glance into the EPOS laboratory.

Up till now, rendezvous simulations with EPOS are dominated by two kinds of scenarios: open-loop, and simple closed-loop. In the former, trajectories are pre-calculated beforehand. The robots follow these trajectories; ground truth and images are logged and can be processed after the simulation. In the latter, the loop is closed in real-time, incorporating image processing and simple GNC functionalities on a single real-time PC. In both cases, the type of relative movement of chaser and target is simple; typically, the chaser approaches on a straight line a target that is either stabilized or shows minimal motion.

However, complex OOS scenarios demand complex Hardware-in-the-Loop simulations on ground. Such simulations exceed by far those simple scenarios that are state-of-the-art. They are not limited to relatively stabilized target objects. Rather, strongly tumbling motion has to be handled. Approach trajectories don't always follow a straight line, but involve fly-around phases, resulting in an extensive three-dimensional motion around the target. The loop is closed with the real GNC algorithms, involving navigation sensor processing, navigation filter, control algorithms, actuator models and alike. All these functionalities are distributed over multiple components connected via an IP network with typical effects like jitter and packet delay. And what's more, all of these functions and algorithms may even be running on embedded systems with strong similarity to actual flight computers.

In DLR's On-Orbit-Servicing End-to-End project a distributed simulation system is developed for Hardware-in-the-Loop simulations of OOS missions. In principle, the very algorithms that would fly on a real mission can be simulated with the E2E infrastructure in closed-loop. The project involves two hardware-in-the-Loop simulators, one of which is EPOS, GNC-system, robotics grasping payload, a high fidelity satellite simulator, the real GSOC network infrastructure and real space communication protocols (CCSDS, PUS). Simulations are carried out by operators from a multi-mission control room at GSOC which is used for real missions at the same time. From the operators' perspectives, it looks like flying a real OOS mission [16] [17].

In DLR's ScOSA project [18], a rendezvous GNC system will be running on a next generation on-board computer prototype, connected to the EPOS facility, simulating an approach manoeuvre in the loop.

This paper describes a considerable extension to the EPOS core control system. This extension, the External EPOS Interface (ExtEPOS), forms an additional abstraction layer between the EPOS facility and an external commanding source like a satellite simulator, such that complex Hardware-in-the-Loop scenarios can be realized flexibly and safely. It raises EPOS's capabilities and usage spectrum considerably, and pushes the simulator to a totally new level.

In this paper, the approaching spacecraft is called chaser and the spacecraft or object to be approached is denoted target. The combination of a satellite's position and attitude is called its pose. The host connecting to ExtEPOS is called ExtEPOS client.

2. EXTEPOS REQUIREMENTS

2.1. EPOS Core Control System

The European Proximity Operations Simulator (EPOS) is a Hardware-in-the-Loop facility that mimics the final 20 meters of rendezvous manoeuvres between two space objects. It consists of two industrial robots, with six degrees of freedom each. One is mounted on a 25 meter long linear rail, on which it can move in real-time to produce a large range of distances (see FIGURE 2). In a typical scenario, the fixed robot carries a satellite mockup, representing the target satellite. This can be a true-to-scale mockup with realistic surface materials (see FIGURE 3). The robot moving on the linear rail carries sensor equipment, e.g. a CCD camera (see FIGURE 4). It represents the chaser spacecraft [14].

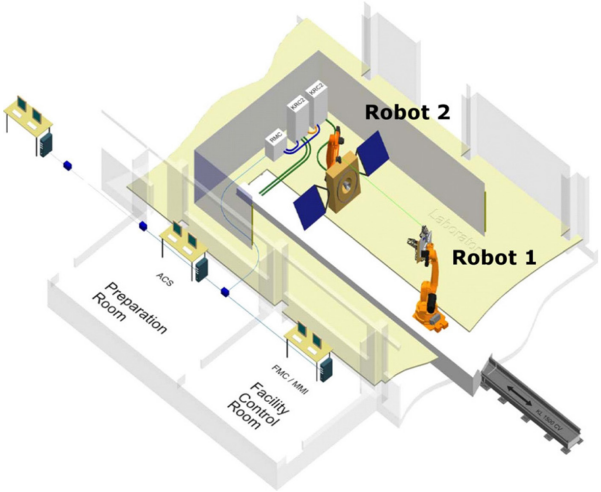

FIGURE 2. EPOS laboratory overview.

For controlling the facility, position and attitude commands must be provided to the facility at 250 Hz in the laboratory frame. Typically, this is done by a Matlab/Simulink model in combination with the Real Time Workshop, running on a

real-time operating system. This computer has to be connected to the core control system via EtherCAT bus, which also implies synchronized timing. In general, the stream of position and attitude commands must be calculated such that the robots can follow the trajectory, considering joint workspace, and that the facility is not damaged by collision. This turns out to be quite a challenge in complex scenarios.

FIGURE 3. True-to-scale target mockup with real MLI and solar arrays mounted on robot 2. Black curtain in background and around robot to mimic dark space background.

2.2. Complex Closed-Loop Simulation on EPOS

Complex simulation scenarios, like the OOS End-to-End Simulation project, involve numerous distributed components like sensors, embedded/desktop PC hardware, and, of course, hardware-in-the-loop simulators like EPOS [16]. The EPOS core control system‘s interface demands an EtherCAT connection. This is a real-time bus implying time synchronization between all connected components. With many components involved, both the integration of EtherCAT hardware in each simulation component as well as the synchronization with the EPOS clock is a strong requirement. And if another hardware-in-the-loop simulator with its own clock is involved, this hard synchronization becomes almost impossible. Therefore, ExtEPOS uses standard Ethernet (IP/UDP) to connect an ExtEPOS client commanding the facility. Ethernet is no traditional real-time bus, however, its performance in a local network is sufficient, and its wide availability clearly outweighs its drawbacks.

As a consequence, ExtEPOS has to handle various effects the network may introduce.

FIGURE 4. Carbon fiber optical breadboard on robot 1. For flexible mounting of sensors and associated equipment. CCD camera calibration pattern in front of the breadboard.

Even if the ExtEPOS client is running on a real-time operating system, its system clock will run slightly slower or faster than that of the machine running ExtEPOS. Hence, there will be some kind of drift. Ignoring other network phenomena (jitter), this leads to two effects, depending on which clock runs faster. There is a regular ExtEPOS time step, during which no data packet is received. Or, there is a regular ExtEPOS time step, during which two packets are received.

The latter case is illustrated in FIGURE 5. The yellow line represents the EPOS control system, i.e. ExtEPOS, executed with a frequency f_{EPOS}. The small vertical lines indicate ExtEPOS processing time steps. Similarly, the blue line illustrates an ExtEPOS client (e.g. orbit/attitude simulator), running with a frequency of $f_{client} > f_{EPOS}$. The ExtEPOS client runs a bit faster than ExtEPOS, and has slightly shorter processing time steps. At each time step, the ExtEPOS client transmits a trajectory command in the form of a UDP packet, numbered in blue in FIGURE 5. The arrows indicate transmission of the packets. ExtEPOS receives the packets and processes them at the following time step. Since the ExtEPOS client runs faster, ExtEPOS may occasionally receive a second packet, before the first can be processed. This is indicated by the red numbers. Only the most recent packet is considered, older packets are discarded since they are outdated.

In reality there is jitter, a statistically distributed duration the packets need from source to destination. As indicated in FIGURE 6, the effect of system clock drift escalates to a whole chaotic sequence of reception of multiple packets (red numbers) and reception of no packets (red question marks). This effectively results in regular intervals of

unnecessary noise in the commanded trajectory.

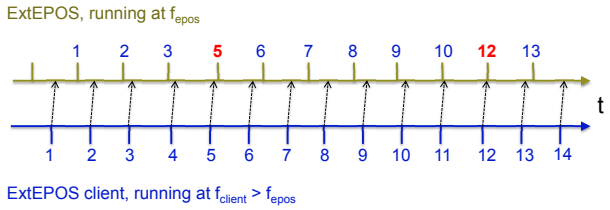

FIGURE 5 Clock drift effect without jitter.

Beside this clock drift effect, UDP does not guarantee that a packet is delivered to its destination at all. The stream of UDP trajectory data packets may include sporadic packet losses. ExtEPOS must be capable of handling lost packets, however unlikely that may be in a Local Area Network (LAN).

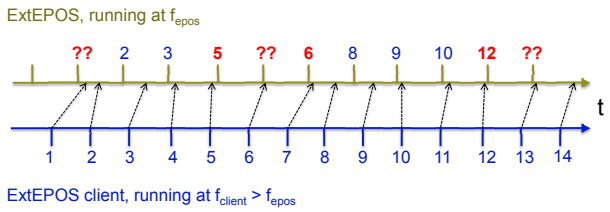

FIGURE 6 Clock drift effect with jitter.

Assuming, that the network related problems are handled appropriately, ExtEPOS has to deal with another issue: Complex simulation scenarios do not take place in some artificial laboratory frame. Rather, these scenarios are simulated in an orbital frame or some similar coordinate system. As an example, this coordinate system could be the Earth Centre Inertial (ECI) frame. There could be a chaser satellite approaching some non-operational target satellite in a Low-Earth-Orbit (LEO). The chaser might execute a fly-around as part of an inspection phase prior to any berthing or docking manoeuvre. The whole approach could take numerous orbits. In ECI, this motion would be quite complicated and vastly three-dimensional. However, the core control system of the EPOS facility expects commands in the laboratory frame. There is a clear gap to be filled by some kind of mapping algorithm. This is no trivial task. There are a variety of motion constraints the robots must not violate. First and foremost, there are robot joint constraints. In practice, two types of constraints have to be considered. Not all, but most of the joints have a limited range of motion. And there are singular configurations, which is essentially a combination of joint angles that must not be reached, lest the robot‘s joints try to rotate with infinite speed and the robot stops. What‘s more, there are constraints in the laboratory, like floor, walls, safety fence etc. The mapping algorithm has to exploit the facility‘s 13 degrees of freedom (6 per robot, one for the linear rail), to realize the six degrees of freedom relative pose between chaser and target, while not violating any constraints. The algorithm has to solve this optimization problem in real-time from closed-loop position and attitude commands, meaning that there is no a-priori information about the next command.

The EPOS core control system expects smooth commanding of the robots. This includes some starting ramp to accelerate the robots from zero velocity. However, an orbit and attitude dynamics simulation does not start at zero velocity. It starts at physically correct starting conditions in orbit. Even if the relative pose between chaser and target does scarcely change at the beginning of a simulation, there will be at least some relative motion. The robots would have to follow this motion from one moment to the next, i.e. in 4ms. This would result in an enormous peak in acceleration. What‘s more, to include a simple acceleration ramp to avoid this problem is not enough. This ramp takes some time, during which the satellites‘ states in the already running software simulation may have changed again. The acceleration ramp has to be updated at each time step to target the new state. This is essentially a control problem, with the special requirement that the target state has to be reached with an accuracy better than the facility‘s. One could, in theory, pause the orbit and attitude dynamics simulation until the very point the EPOS robots have reached the starting conditions. This, however, turns out to lack necessary flexibility and robustness. It introduces strong dependencies and it makes a distributed simulation system comprising more than a single component next to EPOS simply impossible. A better solution is needed.

Safety is an important subject, when talking about complex Hardware-in-the-Loop simulations with EPOS. The robots have a mass of about 1.2 tons each, and, as an example, robot 1 is capable of moving on the linear rail with a velocity of over 1 m/s. If something goes wrong in the simulation, and there occurs a smooth acceleration ramp, these kinds of velocities are possible. However, the EPOS core control system provides neither collision avoidance nor some specifiable speed limit. But for complex simulation scenarios, ExtEPOS has to provide this mandatory functionality.

3. EXTEPOS ARCHITECTURE

3.1. Overview

ExtEPOS provides the functionalities that are required for complex rendezvous simulations. In its current state, it is implemented as a Mathworks Simulink model, running via Real-Time-Workshop on a real-time computer with WindRiver VxWorks operating system. It also includes some embedded Matlab and c++ s-functions.

FIGURE 7 depicts a simplified, schematic view of ExtEPOS, its components and its surrounding interfaces. The Core Control system forms the basis, on which ExtEPOS is build. The Core Control system (see [14]), in essence, combines the two industrial robots and the linear rail into a single laboratory. As a service, this layer provides the possibility to control the robots in the laboratory frame. Connecting to the Core Control system, ExtEPOS provides the service of commanding the facility in any arbitrary coordinate system, catching all operational conditions like dynamic realization of boundary conditions, collision avoidance etc.. Thus ExtEPOS provides a further abstraction, adding enormous flexibility and operational safety. In FIGURE 7, from bottom to top, the abstraction level rises, from a set of robot joints to two satellites in orbit.

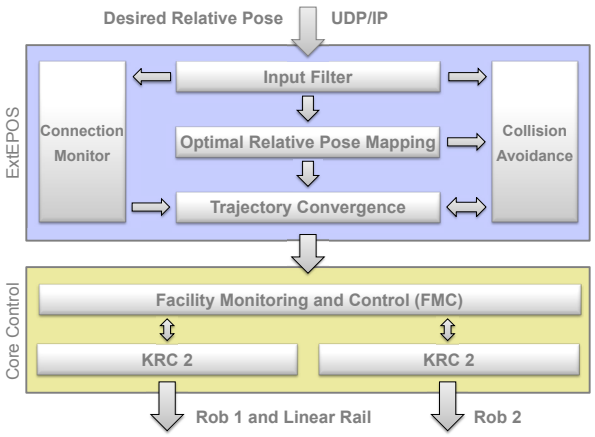

FIGURE 7. ExtEPOS schematic overview.

The individual components of ExtEPOS are:

- Input Filter: Handling of network related effects.
- Optimal Relative Pose Mapping: Coordinate transformation of relative pose in any frame into the laboratory, in such an optimal way, that laboratory constraints are avoided.
- Trajectory Convergence: Elegantly converging to dynamically changing starting conditions in real-time, and constraining maximum robot velocity and angular velocity in the laboratory.
- Connection Monitoring: Monitoring of command and connection status.
- Collision Avoidance: Making the facility gracefully handle trajectories outside the valid range with the possibility of autonomous recovery.

Input filter, Optimal Relative Pose Mapping and Trajectory Convergence are algorithm steps that are executed one after another. Connection Monitoring and Collision Avoidance are orthogonal to the central steps; they need information from them and at the same time control their behaviour. Thus, Connection Monitoring and Collision Avoidance contribute to an underlying control logic that controls ExtEPOS's behaviour (section 3.5.).

In the following, the individual building blocks of ExtEPOS are described in more detail.

3.2. Input Filter

The Input Filter handles network-related effects like clock drift induced jitter and packet loss. These problems are described in detail in section 2.2.

The Input Filter handles clock drift induced jitter by having an internal buffer for one and only one data packet. It stores this packet as soon as two packets are received within one ExtEPOS time step. In each time step, during which packets are received properly, the buffered packet is fed to the Input Filter output, and the newly received packet is pushed into the buffer. The buffer is essentially a FIFO with one element. As soon as a time step occurs during which no packet has been received, the buffered packet is used to fill that gap. With no packet being buffered now, the Input Filter feeds the received packets to the output directly. Only if two packets are received again in one time step, the most recent packet is buffered again. This effectively removes the clock-drift induced jitter effect with the mere cost of a minimum delay. What is left, are the occasional, jitter-free double packet time steps that the proceeding algorithm stages can handle easily.

After removing this effect, there may still be missing packets due to the UDP protocol. The Input Filter solves this problem by cubic extrapolation. This is not a perfect replacement for the original packet. But, for reasonable robot velocities in typical space rendezvous simulations, a few 4ms time steps can easily be bridged by extrapolation.

3.3. Optimal Relative Pose Mapping

ExtEPOS takes position and attitude of two objects in any arbitrary coordinate system. It doesn't even need any information about this coordinate system, as long as both objects' poses are given consistently in the same coordinate frame.

The core idea of simulating complex real-time trajectories with EPOS is the fact that only the relative pose between chaser satellite and target object is important. The EPOS facility provides 13 degrees of freedom. Each robot has 6 axes, i.e. 6 degrees of freedom, and the linear rail counts as 1 degree of freedom. Fortunately, the relative pose of chaser satellite and target object is unambiguously determined by 6 degrees of freedom, 3 for relative position and 3 for relative attitude. Thus, there remain 7 degrees of freedom for handling the laboratory constraints.

Optimal Relative Pose Mapping in ExtEPOS is an iterative process. It consists of two major algorithmic steps: A purely geometric mapping and the optimization of the mapping parameters (see FIGURE 8).

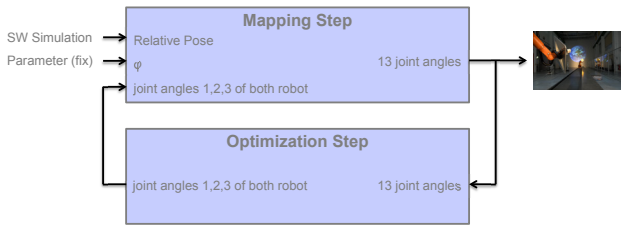

FIGURE 8. Optimal Relative Pose Mapping.

Geometric Mapping: This step expresses the facility's 13 joint angles (i.e. degrees of freedom) as a function of the relative pose between chaser and target satellite, and some mapping parameters. Altogether, relative pose and mapping parameters must constitute 13 degrees of freedom. The mapping parameters encompass a parameter phi and joint angles 1, 2 and 3 of each robot. Parameter phi allows influencing the orientation of the chaser/target combination about their connecting line in the laboratory. It is included in the set of mapping parameters to provide a minimum of manual control of the mapping. E.g. with phi a certain orientation in the laboratory with respect to the sun simulator can be

favoured. The remaining mapping parameters are joint angles 1, 2 and 3 of each robot. This choice simplifies the mapping algorithm. More details are beyond the scope of this paper. To summarize: The mapping step is provided with the relative pose, parameter phi, and joint angles 1, 2 and 3 of each robot. One could visualize that these joint angles are simply passed through to the robots, and the remaining angles are calculated from parameter phi and the relative pose.

Optimization: The optimization step works as feedback for the mapping step. The optimization step takes the 13 joint angles as input and provides the joint angles 1, 2, 3 of each robot to the mapping step. In each 4ms time step, the optimization algorithm tries to change joint angles 1, 2 and 3 of each robot such that all 13 joint angles move away from their limits, towards the minimum of an objective function. For that purpose it is assumed that the relative pose of the satellite mockups is constant, which is a good approximation for every time step of 4ms duration. (Phi is a parameter and is constant anyway.) The objective function to be minimized is a superposition of individual objective functions for each joint angle. Thus, the overall minimum is a compromise between the individual joint angle objective functions. The optimization step uses a gradient-descent-like algorithm.

In ExtEPOS's current state, the mapping algorithm does not consider translational constraints like the laboratory floor or the walls explicitly. However, as it turns out, staying away from joint angle constraints has the robot also stay away from the translational constraints of the laboratory in most cases. Joint angle limits for optimization are chosen more restrictively than possible by the hardware, such that the aforementioned effect is increased and the impact of translational laboratory constraints is reduced.

3.4. Trajectory Convergence

ExtEPOS takes an unconventional approach of realizing initial conditions. There are no actual initial conditions. The ExtEPOS client starts commanding and the EPOS robots start moving. They asymptotically approach the commanded trajectory. At some point, they begin to follow the commanded trajectory precisely, which ExtEPOS confirms to the client. This can be repeated arbitrarily often. As soon as the client stops sending trajectory commands, the EPOS robots stop. Then, when it restarts sending, the EPOS robots again start moving and approach the commanded trajectory asymptotically. This functionality can be compared to a clutch in an automobile. In practice, this means that the ExtEPOS client, i.e. satellite simulator, can be stopped and started arbitrarily with different initial conditions without any action required on part of EPOS.

The only drawback of this approach is that the control loop cannot be closed right from the beginning with real sensors, since the robots need some time to converge to the commanded trajectory. However, in most scenarios this can be handled easily.

The clutch is implemented as a virtual plant, a simple second order system, combined with a forward control term that inverses the plant. The plant incorporates two more elements that distinguish it from textbook second order systems. The second derivative and the first derivative of the value in question (e.g. a position component) are limited to specified values. In that way, the robots' acceleration and velocity in the laboratory is constrained. So, while converging to the commanded trajectory, the robots do not accelerate wildly, but rather in a well-defined way. Specifically, the proper choice for the damping parameter in the second order system guarantees asymptotic behaviour. This design has a drawback. The forward control term incorporates first and second order derivatives, which are very sensitive to noise. As soon as noisy data are provided, the algorithm doesn't converge properly any more. In most cases, the EPOS facility is not commanded with noisy data. It is commanded from a dynamics simulation. The EPOS robots are supposed to move how the satellites would actually move in orbit, and not how sensors would measure the movement. And, as outlined above, the Input Filter reduces network related effects that could induce noise.

This clutch-like trajectory convergence component implicitly fulfils another task. Since it is always involved, even if the robots have converged already, there is always an acceleration and velocity limit in charge. Suppose the robots follow the commanded trajectory precisely, and suddenly an error occurs in the dynamics simulation, which has the robots accelerate smoothly to dangerously high velocities. With the convergence algorithm this is not possible. In such a case, the robots would try to follow the commanded trajectory with the maximum specified acceleration and velocity. But not faster. This protects the facility from damage and wear. It also makes robot motion during real-time closed loop simulations more predictable for the EPOS operator.

3.5. Control Logic

So far, the central components of ExtEPOS, i.e. Input Filter, Optimal Relative Pose Mapping as well as Trajectory Convergence, have been introduced along with their functions during regular operation. However, there are situations where these functions behave differently. This behaviour is steered by a control logic component (not shown in FIGURE 7 to keep a good overview) depending on the system's current state. This state is determined by Connection Monitoring and Collision Avoidance, as indicated by the arrows in FIGURE 7.

3.6. Connection Monitoring

ExtEPOS receives trajectory commands in the form of IP/UDP packets. In contrast to IP/ICP there is no direct way to determine whether the ExtEPOS client is still there. For that purpose, Connection Monitoring applies a timeout to the stream of trajectory commands. As soon as there has not been received a proper data packet for a specified amount of time, the ExtEPOS client is considered non-existent. As a consequence, the Trajectory Convergence component brings the robots to a graceful halt. This is a regular state and ExtEPOS is still active, waiting to receive new trajectory commands.

3.7. Collision Avoidance

The EPOS core control system does not incorporate any collision avoidance functionality. ExtEPOS does. It implements a simple and fast way to prevent that chaser and target collide with the floor, the walls or with one another. A rectangular box is defined for chaser and target. Collision avoidance detects a collision if these boxes touch, or if one of these boxes touches other laboratory constraints. Collision Avoidance in ExtEPOS follows a three zone concept: Green, yellow, and red. These zones exist around chaser and target, red being the innermost, yellow a small layer around the red zone, and green all the remaining space to infinity (see FIGURE 9). They also exist on the inner (virtual) walls of the laboratory workspace, which one can imagine as a three-dimensional rectangle in the laboratory, within which the robots are allowed to move. For illustrative reasons, the principles are subsequently described by a collision between chaser and target.

FIGURE 9. Collision states.

So there are actually two rectangular boxes (red and yellow). To illustrate the behaviour of ExtEPOS in case of a collision, consider e.g. the chaser robot in danger of colliding with the target. From the perspective of the chaser, if its yellow zone touches the target‘s yellow zone, this is a yellow collision state. If red zones touch, this is a red collision state. Otherwise, it’s a green state. There are two possibly different collision states, one for commanded pose and one for actual pose. If the robots have not yet converged to the commanded trajectory, there is a difference between these poses. Depending on both collision states, ExtEPOS either follows the commanded trajectory or halts the robots. As with a non-responding ExtEPOS client, ExtEPOS doesn‘t just stop working in that case. Rather, it stays ready for moving the robots again, provided that the collision state can be recovered. Table 1 illustrates the logic.

Table 1. Collision avoidance logic.

Commanded	Actual	
Red	no impact	Stop!
Yellow	no impact	Stop!
Green	Red	Stop!
Green	Yellow	Go!
Green	Green	Go!

How can it be recovered? Consider when the actual collision state is yellow, but the commanded is green. This can happen when the commanded trajectory temporarily leads throw the yellow or red zone but ends up in the green zone again. That is the very reason for the yellow zone. The robots stop temporarily, actual pose ending in the yellow zone. But the commanded pose is green, so the robots are allowed to move through the yellow zone toward the green zone. In contrast, once in a red state, the robots have to be moved manually out of it. Any other solution would be too dangerous. Such a red actual collision state does rarely occur in practice, since the robots are brought to a halt within the boundaries of the yellow zone. In a sense, the yellow zone is an intermediate zone that allows the facility to recover from a temporal, not realizable (red) commanded trajectory. The zone concept also applies to collisions with walls and the joint limit of the linear rail. It therefore serves an additional purpose. The facility can now take any commanding without the danger of undefined ExtEPOS behaviour in any case. If the relative pose cannot be represented due to a collision, e.g. if the relative distance is too large for EPOS, the robots just do nothing. As soon as the relative pose is realizable again, the robots start to converge to the desired trajectory. During a longer simulation with a far range rendezvous purely in software, EPOS hooks up seamlessly as soon as the relative distance is small enough. This works well save for rare cases, where a red zone has to be crossed to reach the green zone.

3.8. Plug-and-Play

The functions outlined so far work together seamlessly. In that way, ExtEPOS makes the EPOS facility to a plug-and-play Hardware-in-the-Loop simulator that handles safely any trajectory commands in any coordinate system; realizes initial conditions in a simple, flexible and robust way; and makes optimal use of the facilities 13 degrees of freedom.

4. SIMULATION SCENARIO EXAMPLE

In this section, a simple simulation example is shown. A detailed analysis of the numerical results of this simulation scenario is beyond the scope of this paper. Rather, a key view on the trajectory data is picked to illustrate some of

the ExtEPOS concepts and functions.

The satellite mockup, depicted in FIGURE 3, is mounted to the EPOS robot 2 and serves as the target object, while an industrial CCD camera is fixed to the carbon fiber breadboard on robot 1 (the central camera in FIGURE 4). The simulation setup is based on that of the E2E project (see [16]). It uses the GNC system software presented in [17].

The scenario starts at about 25m distance from the target. The chaser is approaching at 5cm/s. At the hold point at 5m distance from the target, the chaser comes to a halt. This could be an inspection point or something similar. Now the chaser changes the approach angle while keeping the distance constant. This is essentially an inspection phase, during which the target's condition is observed. Finally, the chaser proceeds towards the target at 1cm/s until it reaches the mating point, close to the target.

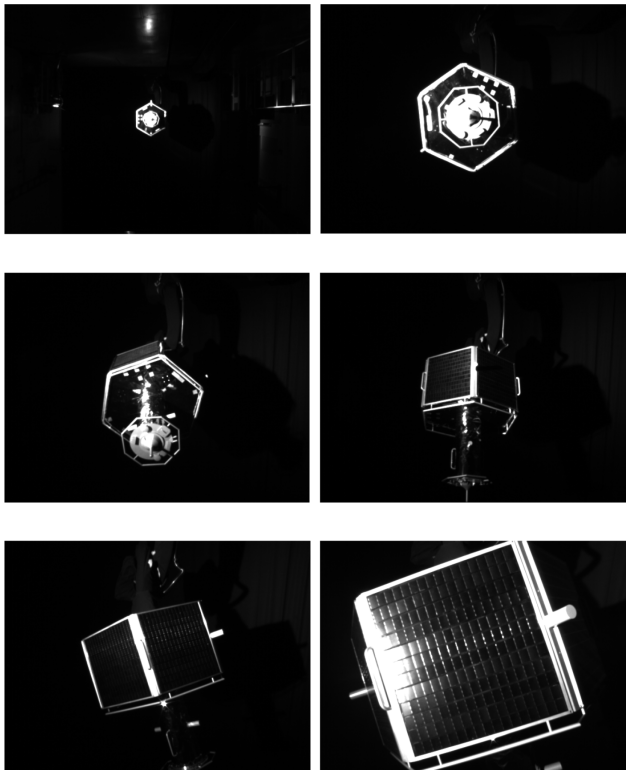

FIGURE 10. Photo series of main GNC sensor (CCD camera) from perspective of the chaser, during approach with fly-around.

FIGURE 10 shows a series of images taken by the CCD camera, which in this scenario is the primary and only optical navigation sensor for the GNC system. The black theatre curtain creates a deep black background. Self-shadowing of the target is clearly discernable. Strong and spotty reflections show especially during the inspection phase. In general, the solar arrays appear very dark. And also the Multi-Layer-Insulation foil, which is so bright under diffuse illumination (compare FIGURE 3), is essentially dark in most images.

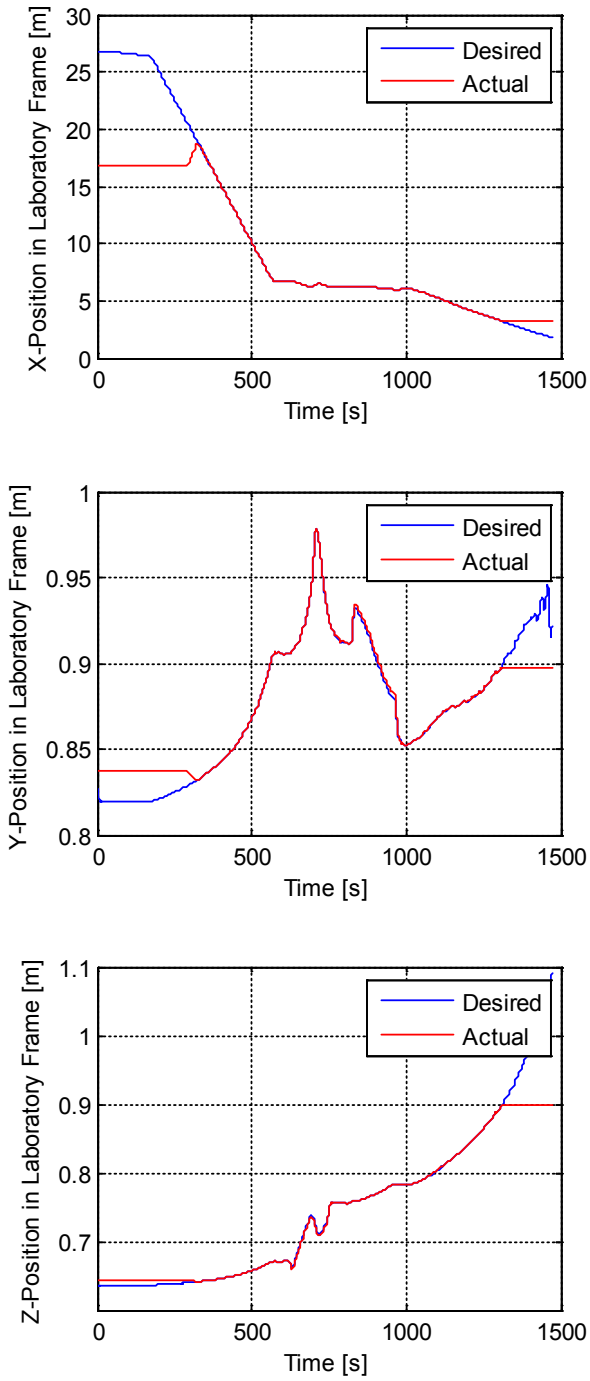

FIGURE 11. Chaser position in the laboratory over time.

FIGURE 11 shows the x, y and z positions of the chaser spacecraft (the robot on the linear rail) in laboratory coordinates over time. The x axis is oriented along the linear rail. The z axis points upwards in the laboratory. The y axis completes the right-hand coordinate frame.

At first, consider the x position. At the beginning of the approach, the chaser's desired position is more than 25 meters. This is something not realizable in the laboratory. The collision avoidance logic has the robots stand still and stand by for valid commands. In the software simulation, the chaser approaches the target; the chaser's desired x

position decreases. At some point, the relative pose is realizable by the facility and the robots start moving. Robot 1 accelerates and moves towards the desired position, which keeps changing. Robot 1 asymptotically converges to the desired trajectory. Robot 1's position decreases further until the hold point at a distance of 5 meters with respect to the target (in FIGURE 11 the absolute positions are given, not the distance). Now follows the fly-around phase. The chaser's position in the laboratory is not constant. This is due to the joint angle optimization algorithm which results in translational movement in the laboratory during the fly-around, while preserving relative pose. Finally, there is the last approach phase to the mating point, slower with 1cm/s. But something goes wrong. The chaser doesn't stop before the target. It just approaches further and further. At that point, collision avoidance takes over again. The robots are brought to a halt. The desired chaser position comes closer and closer to the target which would have ended in a collision.

The approach scenario involves an inspection phase. The target doesn't change its orientation in the inertial frame, but rather the chaser flies around it (see FIGURE 10). At five meter distance, this would require a considerable amount of laboratory volume. However, the Optimal Relative Pose Mapping component of ExtEPOS handles this automatically, by mapping the relative Pose into the laboratory in an optimal way. This means that the target is changing its orientation, while the chaser robot's motion in the laboratory is minimal. This is becomes evident from comparing the components of the chaser's position in the laboratory. While the x position encompasses the whole range of the approach (about 20 m in the laboratory), the motion in y and z direction is minimal (20 to 30 cm).

5. CONCLUSION

Future OOS missions encompass a probably automated approach manoeuvre of a chaser spacecraft towards some target object, e.g. some piece of space debris. This target is likely to be uncooperative. During the highly critical approach phase, the chaser has to handle difficult and partially unforeseeable illumination conditions, while guaranteeing the highest reliability. On-ground simulation and verification of such a system with real hard- and software during the approach is mandatory. EPOS, a robotic rendezvous simulator is the perfect tool for this task. Recently, its capabilities have been enhanced considerably in the form of an additional abstraction layer on-top of the EPOS core control system. It is now fit for simulating complex rendezvous scenarios in closed loop with a satellite dynamics simulator and other hardware-in-the loop components. It makes optimal use of the facility's 13 degrees of freedom simultaneously; it takes trajectory commands in any coordinate frame, mapping the relative pose into the laboratory autonomously; it uses an intelligent collision avoidance and connection monitoring logic, keeping the facility ready for simulation at all times, dealing with all possible input; and it handles starting conditions in a very simple, flexible and robust manner. The enhanced EPOS facility plays an important role in a large, distributed simulation environment in the context of DLR's On-Orbit-Servicing End-to-End Simulation project.

6. ACKNOWLEDGEMENTS

I thank my colleagues Eicke-Alexander Risse and Heike Benninhoff at GSOC for helping me with executing the example simulation and for supporting me in numerous tests of the ExtEPOS software during its development.

7. REFERENCES

[1] G. Personne, A. L. y diaz, and P. Delpy, "Atv gnc synthesis: overall design, operations and main performances," in *6th International ESA Conference on Guidance, Navigation and Control Systems*, Loutraki, Greece, 17-20 October 2005.

[2] C. Bonnal, J.-M. Ruault, and M.-C. Desjean, „Active debris removal: Recent progress and current trends", *Acta Astronaut.*, vol. 85, pp. 51-60, 2013.

[3] S.-I. Nishida, S. Kawamoto, Y. Okawa, F. Terui, and S. Kitamura, "Space debris removal system using a small satellite," *Acta Astronaut.*, vol. 65, no. 1-2, pp. 95–102, 2009.

[4] T. Kelso, "Analysis of the iridium 33-cosmos 2251 collision," in *Advances in the Astronautical Sciences*, vol. 135, 2010, pp. 1099–1112.

[5] C. Pardini and L. Anselmo, "Physical properties and long-term evolution of the debris clouds produced by two catastrophic collisions in earth orbit*," Adv. Space Res.*, vol. 48, no. 3, pp. 557–569, 2011.

[6] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A review of space robotics technologies for on-orbit servicing," *Prog. Aerosp. Sci.*, vol. 68, pp. 1–26, 2014.

[7] A. Graham and J. Kingston, "Assessment of the commercial viability of selected options for on-orbit servicing (oos)," *Acta Astronaut.*, vol. 117, pp. 38–48, 2015.

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

[9] H. MacEwen and C. Lillie, "Infrastructure for large space telescopes," *J. Astron. Telesc. Instrum. Syst.*, vol. 2, no. 4, 2016.

[10] C. Lillie, "On-orbit assembly and servicing for future space observatories," in *Space 2006*, vol. 1, 2006, pp. 522–533.

[11] E. Gralla and O. De Weck, "Strategies for on-orbit assembly of modular spacecraft," *JBIS - Journal of the British Interplanetary Society*, vol. 60, no. 6, pp. 219–227, 2007.

[12] H. Benninghoff, T. Boge, and F. Rems, "Autonomous navigation for on-orbit servicing," *KI-Kuenstliche Intelligenz*, vol. 28, no. 2, pp. 77–83, 2014.

[13] H. Benninghoff, F. Rems, and T. Boge, "Development and Hardware-in-the-Loop Test of a Guidance, Navigation and Control System for On-Orbit Servicing," *Acta Astronaut.*, vol. 102, pp. 67–80, 2014.

[14] DLR Space Operations and Astronaut Training. (2017). EPOS 2.0 RvD Simulator. *Journal of large-scale research facilities*, 3, A107.

[15] O. Ma, A. Flores-Abad, and T. Boge, "Use of industrial robots for hardware-in-the-loop simulation of satellite rendezvous and docking," *Acta Astronaut.*, vol. 81, pp. 335–347, Dec. 2012.

[16] H. Benninghoff, F. Rems, B. Brunner, M. Stelzer, P. Schmidt, R. Krenn, C. Stangl, and M. Gnat, "End-to-end simulation and verification of gnc and robotic systems considering both space and ground

segment," in *10th International ESA Conference on Guidance, Navigation & Control Systems*, Salzburg, Austria, 29 May - 2 June 2017.

[17] F. Rems, E.-A. Risse, and H. Benninghoff "Rendezvous GNC-System for Autonomous Orbital Servicing of Uncooperative Targets". *10th International ESA Conference on Guidance, Navigation & Control*, 29. May - 02. June, Salzburg, Austria, 2017.

[18] T. Peng, K. Höflinger, B. Weps, O. Maibaum, K. Schwenk, D. Lüdtke, A. Gerndt, "A Component-Based Middleware for a Reliable Distributed and Reconfigurable Spacecraft Onboard Computer", *Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on*, 26-29 Sept., Budapest, Hungary.