submitted for presentation at the 4th International Conference on Astrodynamics Tools and Techniques
3-6 May 2010, ESAC, Madrid, Spain

GNC SYSTEMS DEVELOPMENT IN CONJUNCTION WITH A RVD HARDWARE-IN-THE-LOOP SIMULATOR

Tristan Tzschichholz and Toralf Boge

German Aerospace Agency (DLR)

ABSTRACT

In the last years several satellite projects started providing on-orbit servicing capabilities. This means that an on-orbit servicing spacecraft approaches to a client satellite, captures it, takes over the attitude and orbit control and possibly performs additional maintenance tasks. One of the critical phases of such a mission is to ensure a safe and reliable rendezvous and docking (RvD) process. Especially this phase has to be analyzed, simulated and verified in detail. For the special case of verification of rendezvous and docking sensors and systems, DLR has built a new hardware-in-the-loop simulator capable of testing and verifying rendezvous and docking subsystems. This simulator is known as the European Proximity Operations Simulator (EPOS). It is part of the German Space Operations Center (GSOC) located near Weßling. After completion of the baseline simulator concept in 2009, the facility is going to be used for the first time for an internal research project concentrating on a CCD-based rendezvous and docking sensor. This leads to results of both the research project and information about the facility in operation. The paper presents first results and provides insight into ongoing work and projects as well as an outlook of future things to come.

Key words: rendezvous; docking; pose estimation; texture segmentation; vision-based sensor; hardware-in-the-loop; simulation.

1. INTRODUCTION

1.1. New Challenges for RvD in Space

Rendezvous and Docking in space is an important system functionality. Today or in future it will be used for different kinds of missions like:

- Manned Spaceflight Missions in Low Earth Orbit (e.g. ISS)
- Interplanetary Missions (e.g. Mars Sample Return)
- Servicing- and Inspection Missions (e.g. SMART-OLEV)

Rendezvous and Docking is state of the art for manned spaceflight missions today. For the other applications new technological requirements can be found for:

- Rendezvous phase
  Typically the target satellites have not been built for rendezvous and docking tasks. Therefore the rendezvous sensors and systems have to cope with completely uncooperative targets.
- Docking phase
  The robotic based mechanisms have to ensure a safe and reliable gripping or docking at a target without any foreseen docking mechanisms.
- Degree of Autonomy
  Primarily for interplanetary missions with long signal propagation delays, the on-board autonomy plays an important role.

The new technological challenges are mainly related to OOS missions because the RvD process has to cope with uncooperative targets. Therefore one of the critical phases of such a mission is to ensure a safe and reliable rendezvous and docking (RvD) process. Especially this phase has to be analyzed, simulated and verified in detail. Classical approaches e.g. numerical simulations deliver only limited results. Therefore simulations, tests and test facilities has to be defined where the entire RvD process including the flight HW of GNC components and systems can be simulated and tested under utmost realistic conditions of the space environment.

1.2. Reference OOS-Missions

Recently, several satellite projects have focused on providing on-orbit servicing (OOS) capabilities in the near future. The scenarios involve an on-orbit servicing spacecraft approaching and docking to a client satellite. The paper is based on the following two reference mission scenarios.

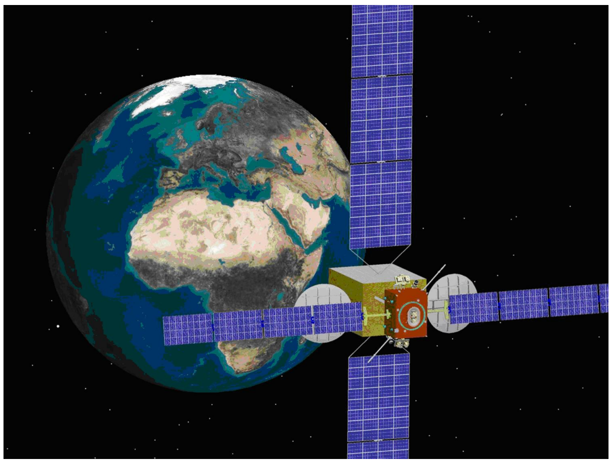


Figure 1. SMART-OLEV docked at a Geostationary satellite

1.2.1. SMART-OLEV

The objective is the orbital lifetime extension for commercial Geo-stationary satellites (**OLEV**- **O**rbital **L**ifetime **E**xtension **V**ehicle). A service satellite shall approach to a client satellite, dock on it and take over the attitude and orbit control of the client. This scenario is shown in Figure 1.

1.2.2. DEOS

DEOS (**DE**utsche **O**rbitale **S**ervicing Mission) is a technology demonstration mission in low Earth orbit where various scenarios in the area of rendezvous and docking as well as re-entry capabilities will be considered. According to the current Phase A study DEOS will consist of both a servicing and a dedicated client spacecraft, which will be launched together into an initial orbit. Primary mission goal is the capturing of a tumbling and non-cooperative client satellite with a manipulator on the servicing spacecraft and the re-entry (de-orbit) of the rigidly coupled configuration within a pre-defined orbit corridor. To achieve the envisaged goal a dedicated set of experiments have to be conducted in which the complexity will be stepwise increased over the mission period. The two spacecrafts are depicted in Figure 2.

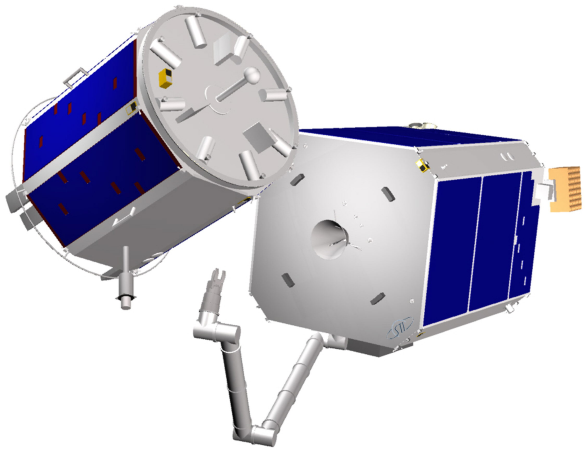


Figure 2. servicer and client satellite of DEOS

2. RVD SIMULATOR EPOS 2.0

2.1. Overview

DLR has more than two decades of experience in the field of simulating RvD maneuvers. The previous EPOS facility was a test bed jointly developed by ESA and DLR for the simulation of spacecraft maneuvers notably over the last few critical meters of the rendezvous phase (without docking simulation). The last intensive utilization was the test and verification of the ATV RvD sensors and systems which are used for the approach to ISS. After dismantling the former EPOS facility, DLR began construction to build up a completely new simulation facility. Test and verification capabilities for complete RvD processes of on-orbit servicing missions will be provided. The new DLR RvD simulation facility comprises a hardware-in-the-loop simulator based on two industrial robots (of which one is mounted on a 25 m rail system) for physical real-time simulations of rendezvous and docking maneuvers. This test bed will allow simulation of the last critical phase (separation ranging from 25 m to 0 m) of the approach process including the contact dynamics simulation of the docking process. Moreover, its main advances are:

- It is a highly accurate test bed. The measurement and positioning performance will be increased by factor 10 compared to the former EPOS facility.
- Dynamical capabilities will allow for high commanding rates and the capability of force and torque measurements.

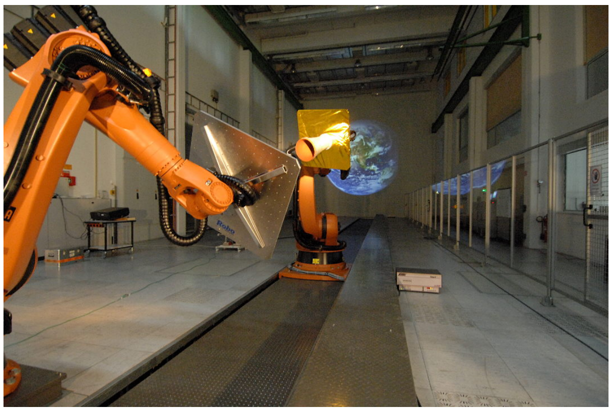


Figure 3. The Facility EPOS 2.0

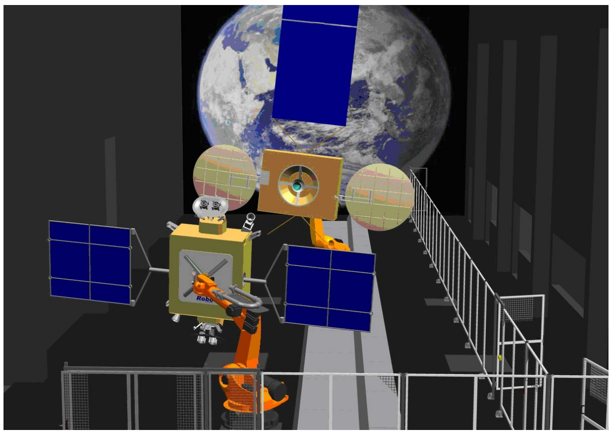
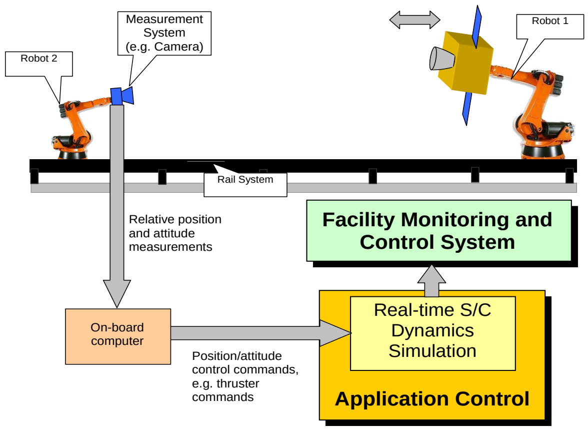


Figure 4. EPOS set up for SMART-OLEV


Measurement System (e.g. Camera)
Robot 2
Robot 1
Rail System
Relative position and attitude measurements
Facility Monitoring and Control System
On-board computer
Real-time S/C Dynamics Simulation
Position/attitude control commands, e.g. thruster commands
Application Control
Figure 5. EPOS HIL Simulation scenario [2]

- The simulation of sunlight illumination conditions as well as the compensation of Earth-gravity force are both part of the assembly to generate an utmost realistic simulation of the real rendezvous and docking process.
- The utilization of standard industrial robotics H/W allows a very high flexibility related to different application scenarios.

The new facility consists of the following components (for details, see [1]):

- A rail system mounted on the floor to move an industrial robot up to a distance of 25 m.
- An industrial robot (robot 2) mounted on the rail system for simulating the 6 degree of freedom of one spacecraft.
- An industrial robot (robot 1) mounted at the end of the rail system for simulating the 6 degree of freedom of the second spacecraft.
- A PC-based monitoring and control system to monitor and control the RvD simulation on the facility. In can be divided into three levels.
  - The local robot control where all axes of the robots are separately controlled
  - The facility monitoring and control system (FMC) where the entire facility is controlled in real time
  - The application control system where the actual RvD-simulation application is running

2.2. A typical RvD Hardware in the Loop Scenario

A typical set up of the EPOS facility for a SMART-OLEV RvD simulation scenario is shown in figure 4. For "hardware in the loop" scenarios such as these, the RvD sensors and the robotic manipulator arm are mounted on one robot and a typical satellite mockup of the client satellite is mounted on the other robot. The RvD sensors can measure the relative position and attitude of the client satellite. On this basis, the on-board computer calculates the necessary thrusters or reaction wheel commands. These are put into the real time simulator. For the next sample, this dynamic simulator computes an update of the state vector (position attitude of the spacecrafts) based on all relevant environmental and control forces and torques. Then, the state vector for the new sample will be commanded to the facility. This scenario is depicted in figure 5.

2.3. Facility Performance

Table 1 provides an overview about the EPOS motion capabilities.

Parameter	**Robot1**	**Robot2**
Position:		
X [m]	-2,5 - +2,5	-2,5 - +24,5
Y [m]	-1,0 - +4,0	-2,5 - +2,5
Z [m]	-0,5 - +1,5	-0,5 - +1,2
Roll [deg]	-300 - +300	-300 - +300
Pitch [deg]	-90 - +90	-90 - +90
Yaw [deg]	-90 - +90	-90 - +90
max. Velocity:		
Translational [m/s]	2	2
Rotational [deg/s]	180	180
Command IF		
Command Frequency [Hz]	250	250

Table 1. EPOS motion capabilities

Because EPOS will be used for RvD sensor verification purposes the facility was extensively calibrated after its installation. With a laser tracker device an overall positioning accuracy of the facility of better than 2 mm (3D 3σ) and an orientation acuracy of 0.2° (3D 3σ) have been verified. In addition it is planned to develop a online measurement system which measures the relative position between both robots and commands corrections to the robots. So the achieved position accuracy will be in sub millimeter range.

3. USING EPOS FOR RVD IMAGE GENERATION

3.1. Overview

As a first part, the EPOS facility has been used to generate imagery of the OLEV mockups for the design of the pose estimation algorithm described later in this paper. To this purpose, a motion command generator (MCG) has been implemented in MATLAB. It is capable of performing linear motions. Using the MCG, an image sequence can be generated to test the pose estimator.

Since the actual positions of the robots are known, the performance of the pose estimation algorithm can be determined. However, in this early state, image generation is not yet working automatically and it is not feasible to produce longer sequences of images, since the positions would have to be calculated manually for each image captured. Automating this capture process is a task for future work. Nevertheless, this section describes the tools used to generate the imagery.

3.2. Motion Command Generator (MCG)

For generation of the trajectories s/w tool called Motion Command Generator (MCG) has been used.

The EPOS MCG is a MATLAB based program consisting of functions, which allows the user to build up and simulate any kind of asynchronous trajectory commands. These functions can generate acceleration phases, a phase with constant velocity and deceleration phases. All calculated accelerations are based on a squared sinus function to avoid any step in the acceleration profile. Output of the functions are a list of robot positions and orientations with the sample time of the EPOS facility. With these functions it is very easy to build up complex command queues for simple user applications (see figure 6). A typical application is the generation of user-defined trajectories for rendezvous sensor tests.

MATLAB was chosen as the host development program because of its mathematical and matrix solving capabilities. In addition MATLAB provides an extended data visualization and plotting capability for analysis of the generated trajectories.

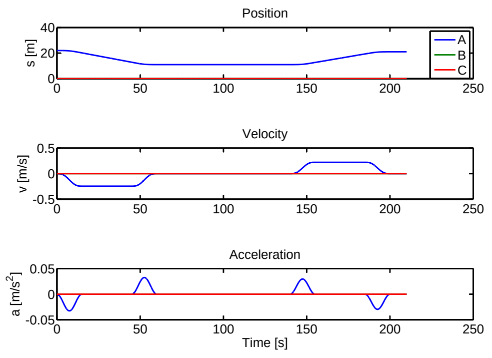


Figure 6. Visualization of MCG Trajectory Sequence

3.3. Camera and capturing images

For capturing images, a Prosilica Gigabit Ethernet Vision camera (GC-655) has been used. It is a monochrome sensor with very high sensitivity delivering up to 90 frames per second at VGA resolution. What is much more important here is, however, the synchronization capabilities. The camera has trigger inputs which can be used to trigger the acquisition. This way, it can be automatically synchronized to the facility. This synchronization is planned for the future and will be implemented soon.

To date, the camera is used in conjunction with a very simple image capture application which allows capturing images with a specific framerate set. Images are compressed using the lossless HUFFYUV codec and stored on-disk in an Audio-/Video Interleave (AVI) file for later processing.

The images obtained are processed offline using MATLAB. This allows automatic code generation in C for later implementation in the HIL scenario. Furthermore, it allows very fast design and optimization of the algorithm.

4. DEVELOPMENT OF A VISION SENSOR-BASED POSE ESTIMATOR

4.1. Introduction

The goals of on-orbit servicing satellites or technology have already been summarized. In order to accomplish that, it is necessary to determine the pose of the target object very accurately. The control system driving the active spacecraft to the docking/ berthing target needs a precise input of the whereabouts of the target spacecraft. This information needs to be obtained by sensors. In this specific case, a vision based sensor is being used to provide this position and orientation estimate (pose) in form of a model-based approach. Since only a single CCD sensor is being used, additional information about the target is necessary to compensate for the lack of information.

4.1.1. Overview

The idea of the algorithm is twofold; first, it is assumed that the target can be seen by the sensor at all times, but is is not known how far the target is located relative to the active spacecraft and, furthermore, the direction vector is not known. In this first phase, the so-called initialization or acquisition phase, the target satellite must be found in the image generated by the sensor, as reliably as possible. Depending on the distance to the target satellite, one can distinguish two tracking modes: The far- range tracking mode and the mid- or near-range tracking mode. The far-range tracking mode provides the control system with a direction vector and the approximate distance to the target. In the mid-range tracking mode, all six degrees of freedom of the target are provided. A hysteresis provides the switch between both modes in cases where the distance to the target is decreased or increased over time.

In this work, texture segmentation is used. In contrast to classical edge extraction methods, texture segmentation algorithms also work very well in cases where the image is noisy and cluttered or the local contrast is very weak.

4.1.2. Related work

In this section, a few related papers shall be reviewed shortly. To start with, determining the relative position of a target object was the goal of [3]. By using morphological filtering, as well as knowledge about the shape, the position of a target satellite was determined. However, the orientation of the object can not be obtained with this approach. It is more suitable for long-range tracking.

A more advanced (but also much more expensive) approach was made in [4]. Here, two satellites with two vision sensors, capable of communicating with each other, track a target satellite. From the two positions, the 3D position of the target satellite can be determined in all cases. This is also feasible for on-orbit servicing tasks, however, in this paper, a single servicer satellite is assumed.

In [5], the tracking process is split up into two parts. A 2D part, which accounts for large 2D displacements, and a 3D refinement part, from which the pose estimate is obtained. The algorithm is tracking edges using local contrast as the main information. It is, in principle, very similar to one of the first monocular 3D trackers (RAPiD) presented in [6].

4.2. A real-time 3D object tracker

In this section, the algorithm which is used to track the pose of the uncooperative target object, is presented in detail. First, this section gives a short overview of the algorithm. As already mentioned, the presented algorithm is a pose tracker, which means it needs an approximate pose a priori, which it refines. The object is then tracked over time by using the tracking result of the previous frame as input.

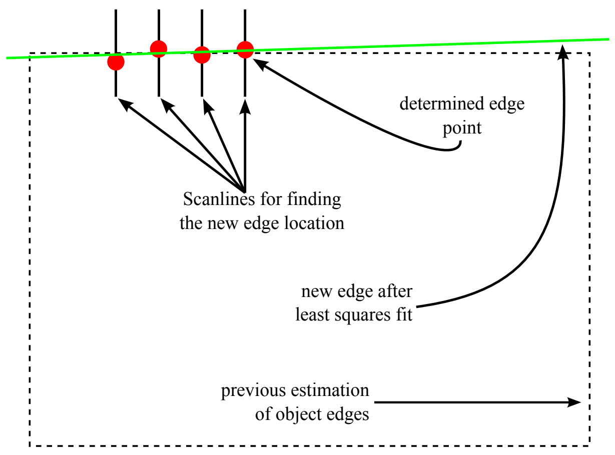

determined edge point
Scanlines for finding the new edge location
new edge after least squares fit
previous estimation of object edges

Figure 7. Illustration of edge tracking using texture segmentation

The tracker has two parts. The first part is the texture segmenter, which does most of the work, tracking the outer edges of the satellite. Once the edges are known, the lines obtained can be intersected to give the corner points. These corner points, in turn, can be assigned 3D direction vectors. Up to this point, all that can be done without knowing anything of the target object in terms of size or dimensions is actually done.

The remaining part (estimating full 6 DOF) requires information about the target object, since the problem cannot be solved otherwise due to underdetermined equation systems (missing information). In this article, the target is assumed to be of rectangular shape and the both side lengths of the target object are known. This is sufficient information to calculate the complete 6 DOF. Figure 7 shows an illustration of how the algorithm works. It is described in detail in the following sections.

4.2.1. Tracking edges using texture segmentation

The edge tracker is an important part of the pose estimator. Assuming the location of an edge is known up to a certain extent, its location is refined for each image received from the sensor. This implementation is based on a paper by Shahrokni, Drummond and Fua [7] (The improved version presented in [8], which uses multiple scanlines in order to retrieve a more stable transition matrix, is not used due to its much higher computational need). To put the idea in words, it is assumed that the edge is a border line of two different textured surfaces, in this case the target objects' surface and the background. These two areas will have very distinct textures in the 2D image. Consequently, it is possible to track the edges using a texture segmenter.

The segmentation works by scanning over the expected edge in perpendicular direction. The resulting lines are called scanlines. Their direction is denoted by the two-element vector $\vec{q}$. Each scanline has a texture change point, where the texture properties change. This point is estimated probabilistically. The details of this process shall now be summarized.

First, the whereabouts of an edge are known in terms of its start and end points, $\vec{a}$ and $\vec{b}$. Next, the difference vector $\vec{d} = \vec{a} - \vec{b}$ is obtained. After that, beginning with $\vec{b}$, the edge is scanned to refine its position. Given a certain stepsize s,

$$\vec{p_i} = \vec{b} + \vec{d} \cdot s \cdot i. \tag{1}$$

For each of these points, a scanline is constructed with length l in such a way that its center is located at $\vec{p_i}$. The length of the scanline needs to be adjusted depending on the distance to the object. If it is far away, the object will appear smaller, decreasing the number of available sampling points and vice versa. Currently, for simplicity and low computational need, this length is determined by

$$l = \begin{cases} \hat{l} + 1 & \hat{l} \bmod 2 = 0 \\ \hat{l} & \text{otherwise,} \end{cases} \tag{2}$$

with

$$\hat{l} = \left\lfloor -z \cdot \frac{41}{4} \right\rfloor, \tag{3}$$

where z is the position of the target coordinate frame reference point in Z direction. As a next step, the direction of the scanline needs to be determined.

$$\vec{q} = \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix}, \tag{4}$$

with $\phi = \operatorname{atan2}(b_2 - a_2, b_1 - a_1) - \pi/2$ being the angle of the scanline, is the direction vector. Then, the scanline is a function of x:

$$\vec{f_i}(x) = \vec{p_i} + \vec{q} \cdot (x - \hat{x}), \tag{5}$$

where $\hat{x}$ is half the number of pixels to scan, i.e. half the length of a scanline. x then counts from 0 to $2\hat{x}$. In this way, pixels are scanned along that line.

The pixel intensities obtained are stored in an array called S. From the paper cited above, it is assumed, that this array is a result of having two different texture-generating processes, T_1 and T_2 and a change point c, which is an index on S. c is what needs to be determined, because this is where the edge is located. This can be done by maximizing the probability of the change point being located at a certain index/ position x, which holds

$$P(c = x) = P(S_1^c | T_1) \cdot P(S_{c+1}^n | T_2), \tag{6}$$

where n is the number of pixels on the scanline. It is now very convenient, that these probability terms can be calculated using the algorithm shown in [7]. Performing the calculation over all c, $P(c = x)$ will have a maximum at the index where the edge is most likely located.

4.2.2. Texture segmentation failure

Texture segmentation is very stable and robust, but it can fail in cases where over- or undersaturated pixels are located in vicinity. A number of 5 or more neighbored pixels having exactly the same intensity is considered to be characteristic for a scanline crossing over- or underexposed image areas (since it has the same effect, no more distinction is necessary). In this case, it is impossible to extract texture information and as a consequence, the result does not provide a good basis for determining the position of the edge.

In a situation such as this, the algorithm can be extended to fall back and use a plain edge detector. More specifically, a Sobel-like kernel

$$\eta = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{7}$$

is convolved with the scanline pixel intensities. To make this more stable, the resulting values are weighted using

$$\psi(x) = e^{-0.1 \left| x - \frac{|S|}{2} \right|}, \tag{8}$$

where x is the index of the pixel on the scanline S, and then normalized. The position of 1 (maximum value) in this array can then be used as an indicator where the edge is probably located. It is not as exact as the texture segmentation method, but provides good results in the mentioned cases of very high contrast changes at the edge location.

4.2.3. Determining the edge

The resulting series of points is then being checked for outliers by the RANSAC algorithm [9]. To summarize: multiple times, two points are selected at random, a line is fit and then the *consensus set* is obtained, which contains all points within a certain error margin ϵ of the line fit. It is critical that this error margin parameter is set to a feasible value. In practice, $\epsilon = 3$ pixels has been determined to give acceptable results. The largest consensus set is then used in the following.

After this step, a clean list of image points is available, which describe a line- the line of the real edge. These edge points can then be line-fitted. The fitting is performed with a least-squares approach. The residual is

$$\xi = \sum_i \left(d - (x_i \cos\theta + y_i \sin\theta)\right)^2 \tag{9}$$

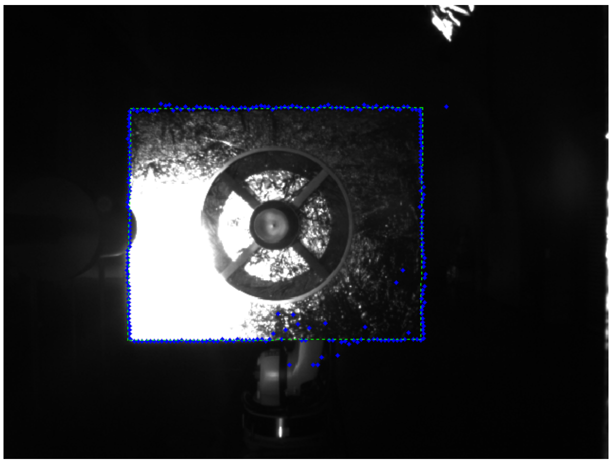


Figure 8. Tracking outer edges of the OLEV mockup, using texture segmentation

which, after some calculation, yields

$$\begin{aligned}\frac{\partial\xi}{\partial\theta} &= 2d\sin\theta\sum_i x_i - 2d\cos\theta\sum_i y_i \\ &\quad -\sin\theta\cdot 2\cos\theta\sum_i x_i^2 \\ &\quad -2\sin^2\theta\sum_i x_i y_i + 2\cos^2\theta\sum_i x_i y_i \\ &\quad +2\cos\theta\sin\theta\sum_i y_i^2 \qquad (10)\\ \frac{\partial\xi}{\partial d} &= 2d - 2\cos\theta\sum_i x_i - 2\sin\theta\sum_i y_i. \qquad (11)\end{aligned}$$

Now $\partial\xi/\partial d$ can be resolved for d and then equation (10) can be solved, using a numeric root finder (since good starting values are available and Newton Raphson method converges locally-quadratic, this method is used).

After that, the orientation (θ) and the distance from the origin (d) of the line have been obtained. After all four lines have been fitted in this way, four intersection points are obtained, representing the corners of the object. Figure 8 shows a prototype of the algorithm implemented in MATLAB tracking the edges of a real-world sized OLEV mockup mounted on the second robot of the EPOS facility. The blue points are the texture process change points, the green lines are result of the line fit. Corner points are retrieved by intersecting the lines. This concludes the edge tracker.

4.2.4. Obtaining the pose

Once the position of the edges and their intersection points are known, one can fit a three-dimensional model of the target (in this case, a simple rectangle) to the data points. The squared distances of the projected corner points and the detected corner points is then minimized in the image space using the least-squares method.

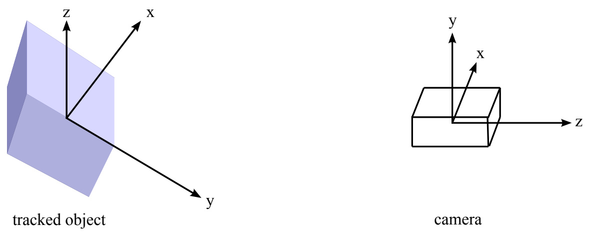


Figure 9. The target being tracked by a single camera. The target is assumed to be located somewhere on the negative Z axis.

In detail, the residual to minimize is

$$\chi = \sum_i \left\langle \vec{p}_i - P\left(\vec{t} + R\vec{s}_i\right)\right\rangle, \qquad (12)$$

where $\vec{p}_i$ is the position of the individual corner point in the image, $\vec{s}_i$ is the relative position of the corner point with respect to the center in three-dimensional space, $P = KH$ is the projection matrix, which, in turn, consists of the orientation of the camera and its calibration matrix

$$K = \begin{pmatrix} f/c & 0 & x_0 & 0 \\ 0 & f/c & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad (13)$$

where f is the focal length of the optics, c is the pixel grid constant on the camera chip and x_0, y_0 are the pixel indices which describe the center of the image. The orientation of the camera is encoded in the matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (14)$$

R is the rotation matrix describing the transformation from the camera coordinate frame to the coordinate frame of the tracked object, together with the translation vector $\vec{t}$. In the optimization, the rotation matrix is reduced to the three Euler angles α, β and γ:

$$\begin{aligned} R &= \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \\ &\quad \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} \times \\ &\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}. \qquad (15)\end{aligned}$$

Then, the residual is minimized for the six variables ($\vec{t}$, α, β, γ). As a result, the pose estimate is obtained. The coordinate frames involved are shown in figure 9.

4.2.5. Results

In the following, the results of a rotational movement are shown, for the sake of completeness. Due to the lack

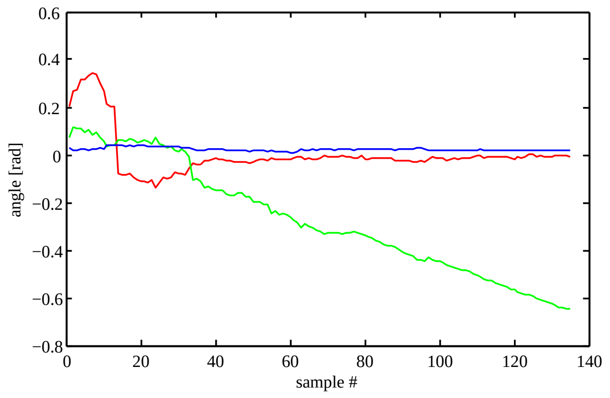


Figure 10. Euler angles during a rotational movement (α: red, β: green, γ: blue)

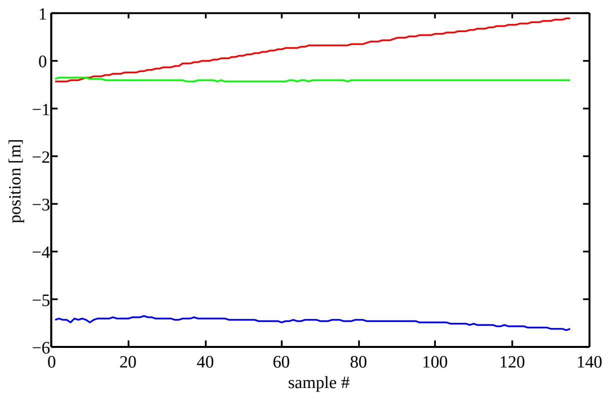


Figure 11. Position of the target reference point during a rotational movement (x: red, y: green, z: blue)

of synchronization, there is no reference measurement. However, the movement can be recognized clearly from figures 10 and 11. This movement was done by commanding one of the robots manually. Near frame #50, the robot stopped for a short time. It can also be seen that once the tracker has locked onto the target, the tracking is stable. It is planned to automate this in order to retrieve statements about accuracy.

5. SUMMARY AND PROSPECT

This paper gave an insight into the challenges concerning RvD processes for on-orbit servicing missions. Based on these, the new EPOS facility at DLR GSOC was presented in detail. It is designed for the simulation of RvD processes such as the ones shown in the beginning. In addition, an image generation process performed on the EPOS facility was described and used for the development of a first prototype of an image processing algorithm.

In the second part, the article summarized the current development stage of such a real-time pose tracker algorithm based on texture segmentation, edge detection and very limited knowledge of the target object. The tracker is (in terms of applications) limited to rendezvous and docking situations and the like. For example, it can not track the target if it is rotating arbitrarily (there is a constraint on the first two Euler angles). Apart from that, there is still a lot of work remaining, for example:

- camera calibration, which allows targets to be tracked in arbitrary distances without having to worry about errors introduced by the optics
- an interface to EPOS, in order to evaluate the performance of the algorithm in terms of accuracy

These tasks will be implemented in order to increase the accuracy of the tracker. The initialization of the tracker is also a problem remaining to be solved, as is the switching hysteresis between long-range and mid-range tracking.

REFERENCES

[1] T. Boge et. al. Hardware-In-The-Loop Simulator für Rendezvous- und Dockingmanöver. In *DGLR Jahrestagung*, 2009.

[2] T. Rupp, T. Boge, R. Kiehling, and F. Sellmaier. Flight dynamics challenges of the german on-orbit servicing mission DEOS. In *21th International Symposium on Space Flight Dynamics*, 2009.

[3] C. Miravet, L. Pascual, E. Krouch, and J. M. del Cura. An image-based sensor system for autonomous rendez-vous with uncooperative satellites. In *7th International ESA Conference on Guidance, Navigation & Control Systems*, 2008.

[4] Tong Chen and Shijie Xu. Double line-of-sight measuring relative navigation for spacecraft autonomous rendezvous. *Acta Astronautica*, 2009.

[5] Éric Marchand and Patrick Bouthemy. A 2D-3D model-based approach to real-time visual tracking. *IVC*, 19:941–955, 2001.

[6] C. Harris. *Tracking with Rigid Objects*. MIT Press, 1992.

[7] Ali Shahrokni, Tom Drummond, and Pascal Fua. Texture boundary detection for real-time tracking. In *European Conference on Computer Vision*, pages 566–577, 2004.

[8] Ali Shahrokni, Tom Drummond, and Pascal Fua. Fast texture-based tracking and delineation using texture entropy. *IEEE International Conference on Computer Vision*, 2:1154–1160, 2005.

[9] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.