SOFTWARE-IN-THE-LOOP SIMULATION OF A PLANETARY ROVER

Matthias Hellerer[1], Martin J. Schuster[2], Roy Lichtenheldt[3]

[1]*German Aerospace Center (DLR), Institute of System Dynamics and Control, Münchner Str. 20, 82234 Weßling, Germany, E-mail: Matthias.Hellerer@DLR.de*

[2]*German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Münchner Str. 20, 82234 Weßling, Germany, E-mail: Martin.Schuster@DLR.de*

[3]*German Aerospace Center (DLR), Institute of System Dynamics and Control, Münchner Str. 20, 82234 Weßling, Germany, E-mail: Roy.Lichtenheldt@DLR.de*

ABSTRACT

The development of autonomous navigation algorithms for planetary rovers often hinges on access to rover hardware. Yet this access is usually very limited. In order to facilitate the continued development of these algorithms even when the hardware is temporarily unavailable, simulations are used. To minimize any additional work, these simulations must tightly integrate with the rover's software infrastructure. They are then called Software-in-the-Loop simulators.

In preparation for the 2015 DLR SpaceBot Camp, a simulation of the DLR LRU rover became necessary to ensure a timely progress of the navigation algorithms development. This paper presents the Software-in-the-loop simulator of the LRU, including details on the implementation and application.

1 INTRODUCTION

The prospects of extraterrestrial body exploration depend strongly on the capabilities of planetary rovers. The most restrictive factor for the coverage of large areas is that currently all movements of a rover are remotely controlled from earth with long round-trip communication delays (e.g. 8 to 48min to Mars [1]). To facilitate the investigation of larger areas, rovers have to act autonomously, with minimal human supervision.

The software development of autonomy systems for planetary rovers is a substantial and complicated task. The software is typically very specialized for a specific hardware. However, in practice the access to the rover for testing the autonomy system is very limited, recommended modifications are hard to realize and the test of risky operations is often skipped.

Concerning these limiting conditions, simulations and virtual prototypes provide a much needed instrument for the development and testing of new autonomy systems. They are often available earlier than real hardware and in unlimited numbers. Further virtual prototypes can be easily modified and tested without exposing the hardware to danger. If a simulation is designed in such a way that the control algorithms may be tested with the simulation instead of the real hardware without any changes to the control algorithm, the approach is called a Software-in-the-Loop (SiL) simulation [2]. This principle is depicted in Figure 1. The rover control algorithms generate commands, these commands are either send to the Simulation or the rover. Based on the commands the rover reacts and thereby changes its environment or its perception thereof, which in the form of sensor data is fed back to the control algorithms. For the rover control software it is irrelevant whether it communicates with a real or a simulated rover.

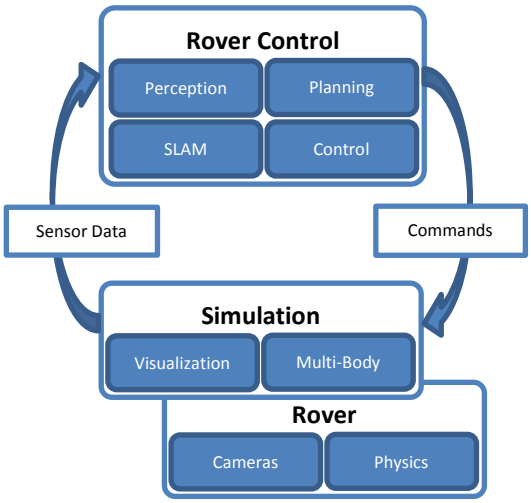

Figure 1: Software-in-the-loop simulation The bottom part can switch between the hardware and the simulation without changes to the controller

This paper presents such a SiL simulation of a planetary rover. To do so it first introduces a generic framework for rover simulations and the simulated rover. It then presents in detail the SiL simulator and

its components before giving a brief recount of the performance at a rover challenge, where this simulator was used.

2 THE DLR ROVER SIMULATION TOOLKIT

The DLR Rover Simulation Toolkit is a generic framework for the simulation of planetary rovers, developed at DLR and written in the modelling language Modelica. The purpose of the framework is to support the development of new rover designs and components and in the future to directly assist Phase A-B studies. It is therefore optimized for quickly building and adapting rover simulations and to thoroughly analyze and compare them. This may be done on the level of single components, larger assemblies and especially the whole rover context. To facilitate this, the framework includes a large number of predefined components, ranging from small building blocks such as PID controlled servo motors, to entire rovers.

3 THE LIGHTWEIGHT ROVER UNIT (LRU)

During the preparation for the 2015 DLR SpaceBot Camp [3], a German robotics challenge for autonomous rovers, the need for a SiL simulation of the Lightweight Rover Unit (LRU) arose. The LRU, developed at the DLR Robotics and Mechatronic Center, is a terrestrial prototype of an agile, small-size lunar exploration rover for technology demonstration and research on autonomy concepts. It is equipped with stereo cameras as well as a robotic arm for mobile manipulation and an agile locomotion system based on two bogies [4][5][6]. Figure 2 shows a simulated model of the LRU.

The LRU uses three cameras for the perception of its environment, mounted to a pan-tilt unit on a mast, as can be seen in Figure 2. The two outer cameras form a stereo camera pair, recording synchronized grayscale images, while the center camera records color images. Dense depth data is recovered through stereo reconstruction via Semi Global Matching (SGM) [7] on a FPGA at a frame rate of 14Hz. It is used for self-localization as well as obstacle avoidance, environment modelling and path planning. Fast, local self-localization is achieved through a fusion of visual stereo odometry estimates together with inertial measurement data [8]. For global self-localization as well as the generation of a 3D environment model, the local estimate as well as the depth images are fed into a SLAM (Simultaneous Locating and Mapping) algorithm to generate a map of the robots surroundings for path planning [9][10].

4 THE SOFTWARE-IN-THE-LOOP SIMULATOR

A simulation can only ever represent a defined portion of a system, depending on its application. As discussed this simulator will be used by the autonomy subsystem and is therefore primarily focused on the camera-based vision and the locomotion related subsystems. The individual components are broken down as follows:

4.1 Multi-Body Simulation Of The Rover

The Multi-Body model is based on the CAD model used to build the robot. It includes all relevant kinematic components, as well as masses and inertias.

Figure 2: Multi-Body model of the Lightweight Rover Unit (LRU)

As depicted in the 3D simulation model of the rover in Figure 2, the rover has a central body. Attached on top of the body is a mast, with a pan-tilt-unit, holding the rovers three cameras. The locomotion system is based on two bogies, one in the front and one on the back, which are mounted to an actuated rotational spring-damper. Directly attached at the rear bogy is a robotic arm, and at the end of the bogies are the steering units to which wheels with hub motors are mounted [11]. Excluding the robotic arm, which is of little importance in the given context, this sums up to twelve degrees of freedom for the rover plus the wheel ground contact, discussed later.

Besides the rover, the simulation also entails a model of the rover's surroundings. The model used here is a mesh generated from a laser scan of the 2013 SpaceBot Cup arena [11][12].

With help of the DLR Visualization Library [14] and its tight integration with the Modelica multi-body library, the kinematic model was easily augmented with a 3D Visualization, as shown in Figures 2, 4 and 6. This forms the basis for the virtual camera system discussed later.

4.2 Wheel-Ground Contact

The simulation of wheel-ground interaction is a focus of our research and we can therefore refer to a large catalogue of different algorithms. A comprehensive list of available algorithms, including detailed comparisons can be found in [15].

For the given purpose, computational efficiency is the most important factor and may be achieved at expense of accuracy. We assume that all pathways are either easily traversable or not traversable at all. For this reason a single point of contact model with spring-damper characteristics was selected. In this model each wheel has a line in the direction of the Z-Axis (assuming gravity acts in the direction on -Z) and searches for the first intersection with a 3D object along this line. If a contact is found, a virtual spring-damper element is introduced to simulate the compliance of the soft soil and the wheel. Further a non-linear damper used to approximate the friction. To keep this part of the simulation simple, we selected fixed values for the spring and damper constants, based on a more intricate Bekker-Wong formulation [15]. For each wheel this correlates to six degrees of freedom, so for all four wheels and including the multi-body system discussed earlier, the total number of degrees of freedom in this model is 36.

4.3 Drive-Train

This component simulates the drive train, from the wheel flange, via the gears to the motor and the motor driver, specifically the integration of the motor driver in the rover controller.

To meet the high requirements on the locomotion subsystem, the LRU uses custom designed permanent magnet synchronous motors. To drive these, it uses servo controllers, designed for industrial automation and with a field bus interface [11]. These include a motor driver which handles the low level motor control while presenting a simplified interface to the field bus. The user selects between one of three modes: torque, speed, position, and sets an associated value. The rest is handled by the controller. Therefore, to build a transparent interface, the simulation has to include both, the low level controller and the field bus interface used to communicate with the rover controller.

Simulating the entire field bus would be very complex. To avoid this, the field bus interface is abstracted in the rover controller. This interface is replaced by a custom interface which directly communicates with the simulation, thereby considerably simplifying this part of the simulation while maintaining the interface to the rest of the controller.

The low level controller structure is depicted in Figure 3. It is always based on a torque, directly applied to a flange. Depending on the mode of operation, this torque is either set to a value demanded via the field bus interface, or a demanded speed or position is set using PID controllers.

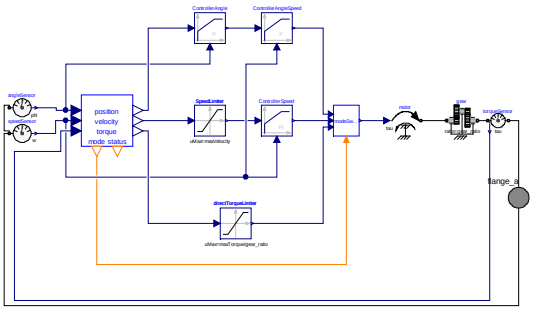

Figure 3: Low level motor controller model

4.4 Virtual Cameras

On the LRU, the two stereo cameras are triggered at exactly the same time. The camera images are then fed into SensorNet. SensorNet is a custom DLR middleware based on shared memory, specifically designed to handle sensor data with large bandwidth requirements, such as image data with very low latency. To keep the latency down and to avoid image distortions, no compression algorithm is used. The cameras have a resolution of 1292x964 with 8bit depth for greyscale or per color channel and operate at 15Hz. This results in a bandwidth of about 18MB/s for each stereo camera and 53MB/s for the color camera, which brings the total to nearly 90MB/s. Handling and processing this amount of data in a real-time environment requires specialized software solutions.

For the simulation the idea was to utilize the existing 3D simulation visualization to generate images of the rover's surroundings and use those instead of real camera images. Therefore the visualization software was adapted to render images in memory only. The DLR Visualization Library is based on OpenSceneGraph and OpenGL which does not support greyscale image formats [16]. Additionally OpenGL allows the user to request different color image formats when reading rendered image data, yet it internally will convert the image data from its internal format to the requested format every time an image is read. Practical experiments have shown that this internal conversion is very slow. As discussed earlier, the required data throughput is very high. Therefore the data conversion had to be thoroughly optimized, using Streaming SIMD Extensions (SSE), a special extension of the CPU

instruction set, designed for handling large amounts of data [17].

The integration with SensorNet is again implemented fully transparent. Images previously acquired by cameras are now generated by the simulation. This allows us to test the entire image processing pipeline, yet for practical tests of the SLAM and path-planning algorithms, a new problem arose. As described before, the LRU uses an FPGA for the 3D reconstruction. This is necessary to run the complex SGM algorithm at the desired rate of 14Hz.

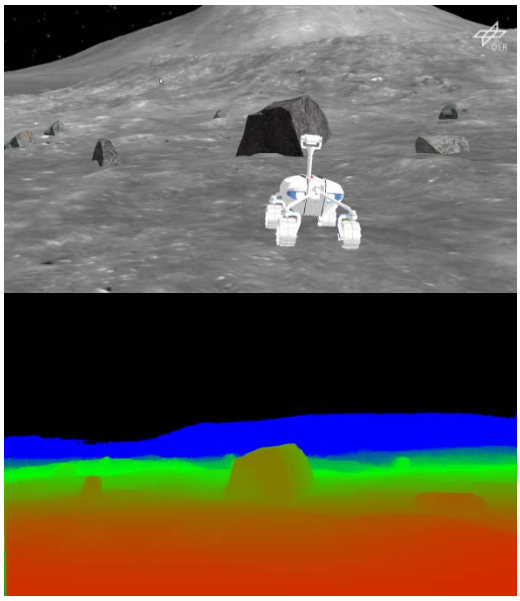

Figure 4: The rover in a simulated environment (here on the moon) and the perceived depth image with colors denoting the distance from the camera

FPGAs are very costly and therefore only available in a limited number. In order to provide depth images to all developers at a decent speed when no FPGA is available, we have to extract the depth information directly from the simulation. The FPGA program reads the image data from one SensorNet stream and writes the resulting depth image to another one. To emulate this behavior, the visualization reads the OpenGL depth buffer [16] and writes the depth data into an SensorNet stream, in the same way the FPGA program does, to once again guaranty a transparent interface. An example of such a virtual depth image is depicted in Figure 4. The upper part shows the simulated rover in a lunar environment and the bottom image depicts the corresponding depth image, with colors denoting the distance of each pixel from the camera. Clearly visible is the color gradient of the receding ground and the large boulder in front of the rover.

5 THE 2015 DLR SPACEBOT CAMP

Figure 5: The Lightweight rover unit (LRU) at the 2015 DLR SpaceBot Camp final

As mentioned before, the presented Software-in-the-Loop simulation was built to support the developers of autonomous systems, in preparation of the 2015 DLR SpaceBot Camp.

The DLR SpaceBotCamp is a German national exhibition for planetary rovers, where ten teams show their capabilities in a challenge to fulfill a given mission scenario. The scenario is loosely based on a real mission scenario, where the rover is placed in an unknown environment, with very limited communication capabilities. Hidden in this environment are two objects which the rover has to autonomously find, pick up and place at a base station [3] [11].

The SiL framework in operation is depicted in Figure 6. Visible in the upper left hand corner is the control program, used to start and monitor all components. To the right of that is Simulation Visualization, showing the rover in the SpaceBot Cup training environment. The bottom part shows the rovers perception system. On the left, one of the stereo camera images, overlaid with a 3D mesh, based on the depth perception. The bottom center shows the 3D reconstruction of the environment. Clearly visible is the rover in the map and a few areas marked red, to indicate unnavigable terrain. The right side finally shows a 2D representation of the reconstructed map.

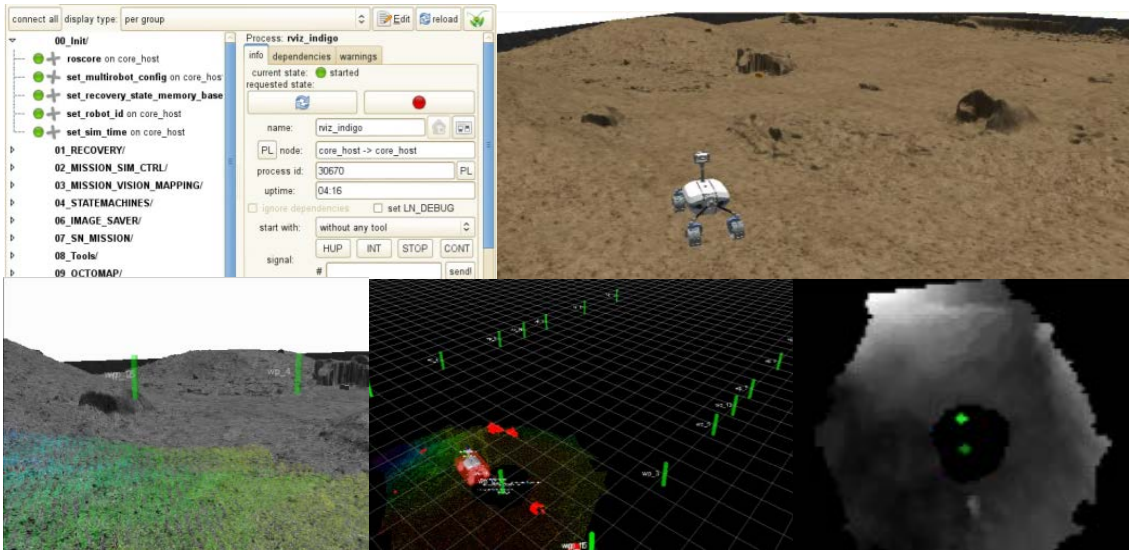

Figure 6: The Software-in-the-loop simulation, visualization and rover controller in action

This way the SiL simulator was used in preparation of the SpaceBot Camp. At the event the team performed very impressively. The LRU was the only rover able to perform all tasks within the original specification in a single run and did so in half the given time [18]. The rover during its run is depicted in Figure 5.

6 CONCLUSION

This article presented a Software-in-the-Loop simulator for a planetary rover, including details on its structure, implementation, and practical application. While originally build as simple extension of an existing rover simulation as part of the Rover Simulation Toolkit, it also contributed significantly to the improvement of the toolkit.

An important lesson learned is that the communication between the developers and the users is absolutely key to success, even more so than for most software developments. The algorithms used for space systems are highly specialized for a specific non-standard hardware. They often use very specific aspects or features and even use data otherwise considered a byproduct. Determining which aspects have to be modeled at which level of detail, must be considered constantly and in close cooperation with the user.

The presented Software-in-the-Loop simulator was used in preparation of the 2015 DLR SpaceBot Camp. Here it made the developers of higher level control algorithms independent of hardware access. Due to constant hardware modification, access to the rover is rare and even if it is available it can only be used by a single person. Therefore the hardware needs to be disassembled regularly and might be unavailable for many days. With the simulator, the control algorithm developers can work more efficiently and without interruption.

Even though the simulator has already been used very successfully, it is still under development. The biggest challenge at the time of writing is the performance. On modern hardware the simulator runs at about 80% real-time. This is sufficient for most purposes but to really test certain time critical functions, this needs to be improved to 100%.

With the SpaceBot Camp completed, the simulator is continuously developed further for new applications and improved with new functionality. By the end of 2017, the ROBEX project will presents its progress in the form an analog mission, where the LRU rover will have to autonomously set up an active seismic network in an environment similar to a planetary surface [4][19]. The presented Software-in-the-Loop simulator will be used und improved further in preparation for this event.

Acknowledgement

This work was supported by the Helmholtz Association, project alliance ROBEX, under contract number HA-304.

References

[1] T. Ormston, “Time delay between Mars and Earth,” *Mars Express*, 2012. [Online]. Available: http://blogs.esa.int/mex/2012/08/05/time-delay-between-mars-and-earth/. [Accessed: 13-Jan-2016].

[2] “Software-in-the-Loop Modeling and Simulation.” [Online]. Available: http://www.acm-sigsim-mskr.org/MSAreas/InTheLoop/softwareInTheLoop.htm. [Accessed: 04-Apr-2015].

[3] “DLR - Space Administration - DLR SpaceBot Camp,” 2016. [Online]. Available: http://www.dlr.de/rd/en/desktopdefault.aspx/tabid-8101/13875_read-35268/. [Accessed: 13-Jan-2016].

[4] A. Wedler, M. Hellerer, B. Rebele, H. Gmeiner, B. Vodermayer, T. Bellmann, S. Barthelmes, C. Lange, L. Witte, N. Schmitz, M. Knapmeyer, A. Czeluschke, L. Thomsen, C. Waldmann, M. Wilde, and Y. Takei, “Robex – Components and Methods for the Planetary Exploration Demonstration Mission,” in *13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2015, no. 1.

[5] B. Schäfer, J. Albiez, M. Hellerer, M. Knapmeyer, G. Meinecke, O. Pfannkuche, L. Thomsen, T. Wilde, T. Wimböck, and T. van Zoest, “Robotic Developments for Extreme Environments – Deep Sea and Earth’s Moon,” in *i-SAIRAS Proceedings*, 2014.

[6] A. Wedler, A. Maier, J. Reill, C. Brand, H. Hirschmüller, M. Schuster, M. Suppa, A. Beyer, N. Y. Lii, M. Maier, H.-J. Sedlmayr, and R. Haarmann, “Pan/Tilt-Unit as a perception module for extra-terrestrial vehicle and landing systems,” in *ASTRA*, 2013.

[7] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[8] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa, “Stereo vision based indoor/outdoor navigation for flying robots,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3955–3962.

[9] M. J. Schuster, C. Brand, H. Hirschmueller, M. Suppa, and M. Beetz, “Multi-Robot 6D Graph SLAM Connecting Decoupled Local Reference Filters,” *Iros 2015*, pp. 5093–5100, 2015.

[10] C. Brand, M. J. Schuster, H. Hirschmuller, and M. Suppa, “Submap matching for stereo-vision based indoor/outdoor SLAM,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5670–5677.

[11] M. J. Schuster, C. Brand, S. Brunner, P. Lehner, J. Reill, S. Riedel, T. Bodenmüller, K. Bussmann, S. Büttner, A. Dömel, W. Friedl, I. Grixa, M. Hellerer, H. Hirschmüller, M. Kassecker, Z.-C. Marton, C. Nissler, F. Ruess, M. Suppa, and A. Wedler, “The LRU Rover for Autonomous Planetary Exploration and its Success in the SpaceBotCamp Challenge,” in *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2016.

[12] D. Holz and S. Behnke, “Registration of Non-Uniform Density 3D Point Clouds using Approximate Surface Reconstructione,” in *Proceedings of the International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*, 2014.

[13] D. Holz and S. Behnke, “Registration of Non-Uniform Density 3D Point Clouds using Approximate Surface Reconstruction,” 2014. [Online]. Available: http://www.ais.uni-bonn.de/mav_registration/. [Accessed: 04-Apr-2016].

[14] M. Hellerer, T. Bellmann, and F. Schlegel, “The DLR Visualization Library - Recent development and applications,” in *Proceedings of the 10th International Modelica Conference*, 2014, pp. 899–911.

[15] R. Lichtenheldt and M. Hellerer, “Heterogeneous, multi-tier wheel ground contact simulation for planetary exploration,” in *ECCOMAS Thematic Conference on Multibody Dynamics*, 2015.

[16] “glReadPixels - OpenGL 4 Reference Pages.” [Online]. Available: https://www.opengl.org/sdk/docs/man/html/glReadPixels.xhtml. [Accessed: 04-Apr-2016].

[17] A. Schaub, M. Hellerer, and T. Bodenmüller, “Simulation of Artificial Intelligence Agents using Modelica and the DLR Visualization Library,” in *9th International Modelica Conference*, 2012.

[18] H. Hirschmüller and A. Wedler, “Success of the RMExplores! Team at SpaceBot Camp 2015,” 2015. [Online]. Available: http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-3755/17612_read-44875/. [Accessed: 13-Jan-2016].

[19] A. Czeluschke, M. Knapmeyer, F. Sohl, and N. Schmitz, “Die ROBEX-ASN Studie – Geophysik auf dem Mond,” in *DGG-BDG Seminar und Workshop “Oberflächennahe Erkundung,”* 2013.