Imaging Sensor Emulation and Dynamics Simulation for PIL/HIL

J.Eggert[(1)], S.Weikert[(1)], I. Kossev[(1)]

[(1)]Astos Solutions GmbH
Meitnerstraße 8
70563 Stuttgart, Germany
Email:
Jochen.Eggert@astos.de
Sven.Weikert@astos.de
Ivan.Kossev@astos.de

ABSTRACT

Latest developments of computer hardware and computer software allow the utilization of new approaches in software verification facilities (SVF) and AIV simulators. This paper describes a highly configurable dynamics and environment simulator running on a dSPACE real-time platform and a LIDAR and camera simulator. Both are designed to support SVF and to improve its efficiency. The dynamics and environment simulator is based on the ASTOS software which provides rapid configuration of any space flight scenario. Space environment models represent the ECSS standard for Earth observation. The dynamics is capable to model rigid body dynamics, multi-body dynamics of e.g. manipulator arms and flexible dynamics based on beam approximations or NASTRAN files using the DCAP software. The camera and LIDAR simulator provides in real-time raw data information of space scenarios such as rendezvous and docking or planetary landing. A highly realistic representation guarantees for the quality of the raw data, which is intended as input for feature tracking and state estimation algorithms. The LIDAR simulator is successfully used in the German orbital servicing mission DEOS within SVF. Its major advantage is the low cost approach to verify vision-based navigation concepts without expensive test facilities such as EPOS. This paper will present the interfaces to the SVF the performance and lessons learned of both units.

ASTOS FOR REAL-TIME APPLICATIONS

The ASTOS software, originally developed to optimise spacecraft trajectories evolved during the past years to a universal analysis tool for space missions. Newly developed interface made it a suitable component for hardware and processor in the loop simulations. These new interfaces are described in this paper.
From its heritage ASTOS is designed as a closed self-standing tool that models all relevant components of the spacecraft and the environment and performs the state integration. For this purpose ASTOS provides a huge library of equipment and environmental models of variant complexity.
ASTOS standalone is designed as an open-loop simulation software. Some simple control algorithms are part of ASTOS but complex control or guidance algorithms are not provided. The closed-loop simulation capability of ASTOS is realized with an interface to MATLAB/Simulink.
In this new linked mode, developed for coupled GNC and mission analysis, Simulink becomes the main tool that controls the simulation process. On-board algorithms as well as custom sensor and actuators are then typically modelled in Simulink, while at least the environment as well as the equations of motion is provided by ASTOS (further models can be used if no specific demands require custom models; see Fig. 1).
As an alternative to built-in ASTOS models and newly developed Simulink models, ASTOS comes with a MATLAB toolbox that comprises standard models for typical sensor models (e.g. Lidar, star tracker) and actuator models (e.g. thrusters, gyros, magnetorquers), control algorithms (e.g. spin control, target pointing, quaternion feedback), utility blocks (e.g. coordinate transformations) as well as blocks to realize a telecommand/telemetry console and also a simulation console (via that it is possible e.g. to simulate system errors). Furthermore all analysis and plotting functions of ASTOS can be still accessed.
ASTOS is linked to the Simulink model by a special ASTOS model block that is part of the ASTOS toolbox for MATLAB/Simulink. Via this block the user can select desired output functions and specifies Simulink-driven controls (see Fig. 3). The detailed scenario is defined by the classical ASTOS GUI (see Fig. 2) in the same way as it is done for standalone scenarios. Therefore it is also quite simple to migrate a standalone scenario into a coupled Simulink scenario.

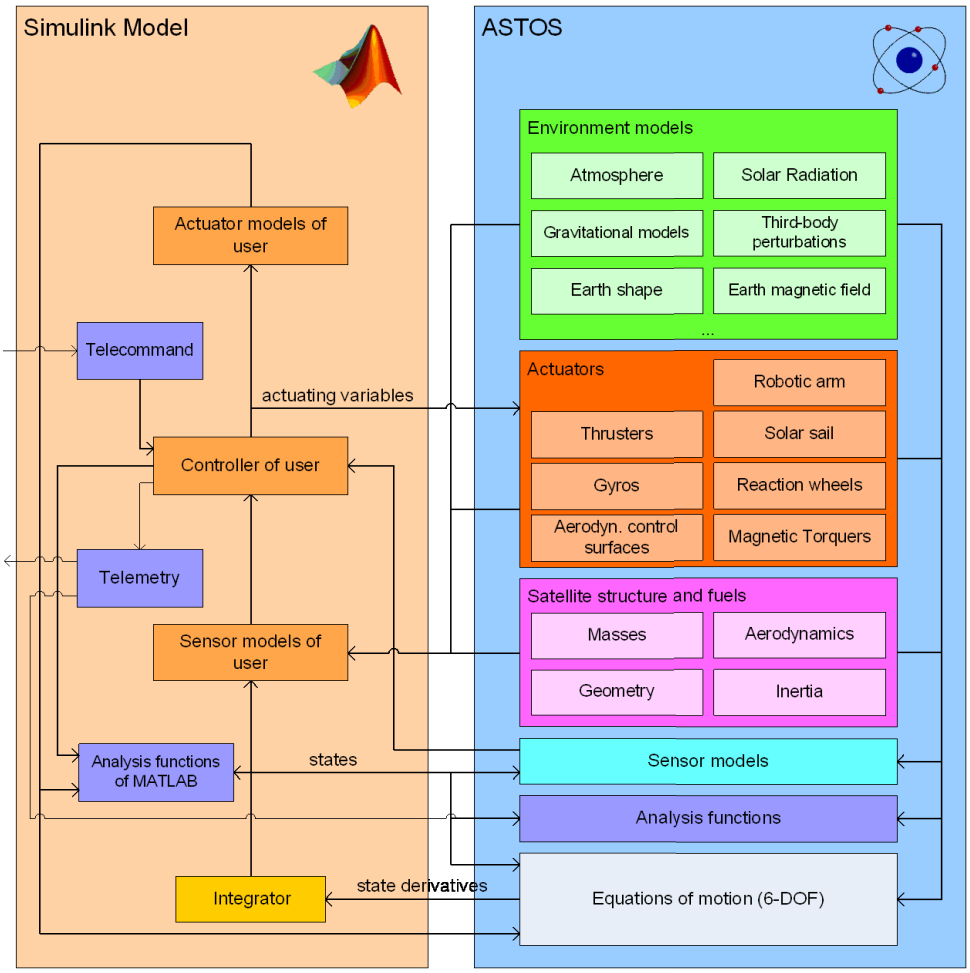

Fig. 1 Architecture of coupled GNC analysis

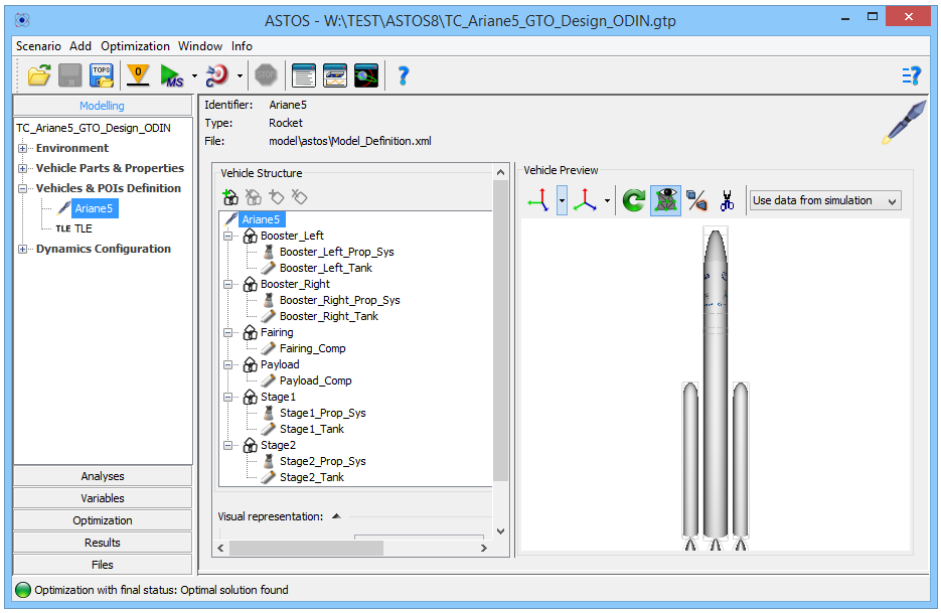

Fig. 2 ASTOS graphical user interface showing the vehicle configuration

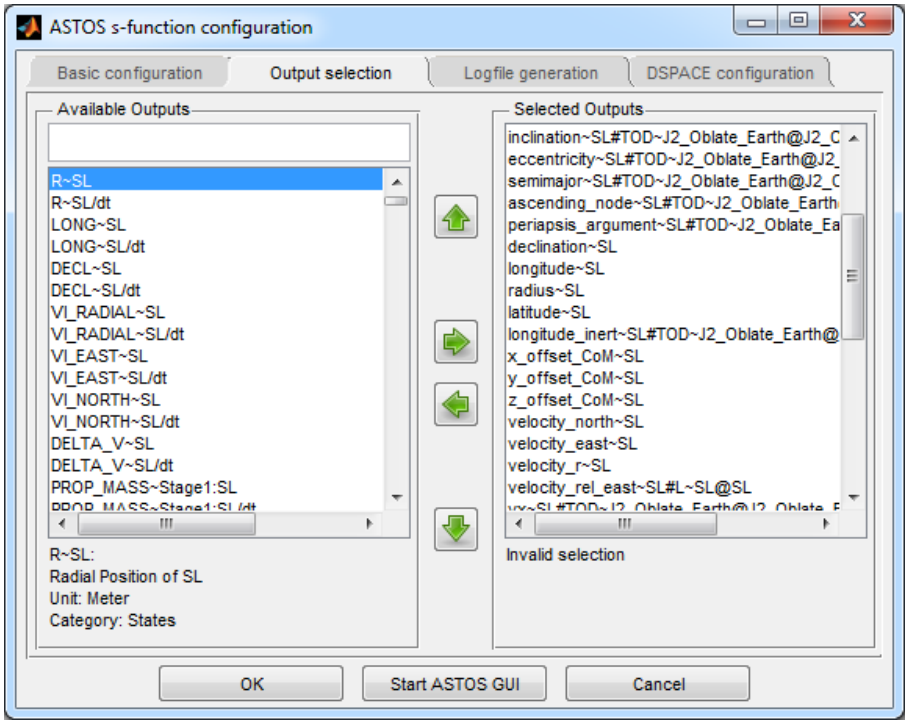

Fig. 3 Selection of output functions in the ASTOS Simulink block

A tool-chain developed by dSpace and that is also based on the MATLAB/Simulink environment allows to convert Simulink models and to run them on a dSpace hardware platform with real-time operating system and a multitude of available interface cards. Via these interfaces real sensors and actuators can be included into the simulation system (HIL).

Still this real-time system needs an environment and dynamics simulator, which can be again implemented as a link to the ASTOS software. The dSpace DS1006 multi-core processor board [1] provides several real-time cores plus one additional Linux core. This core is used as bridge between the real-time algorithms running on the DS1006 board and the ASTOS dynamics and environment simulator that still runs on a classical PC system (this PC system must be fast enough to answer the real-time requests from the DS1006). This is realized via a TCP/IP interface implemented on the DS1006 Linux core. During the conversion process and the upload of the Simulink model to the DS1006, the ASTOS Simulink block is automatically converted into a shared-memory interface between the real-time and the Linux process.

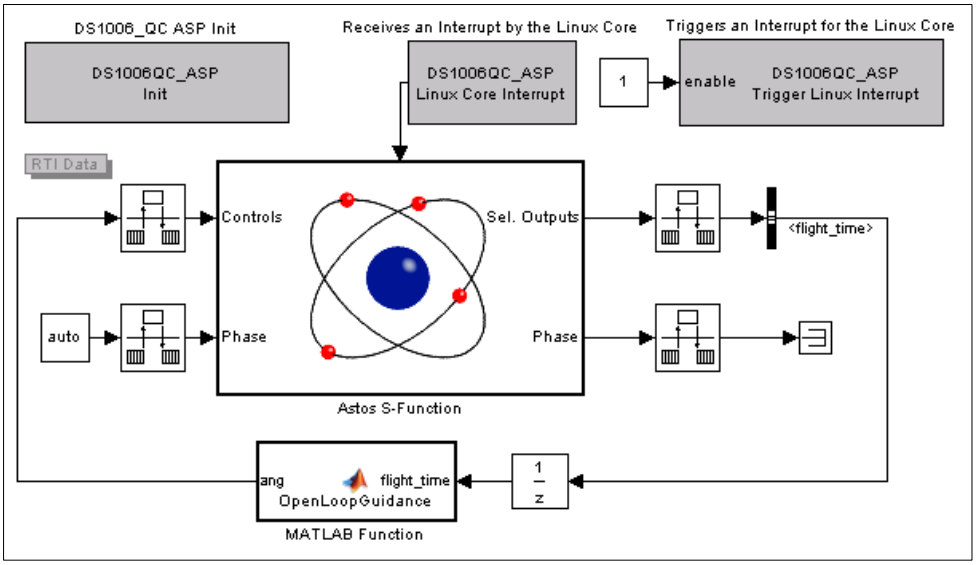

Fig. 4 (Simple) Simulink model including the ASTOS interface block, prepared for automatic export to the DS1006 board

Modelling of flexible dynamics

ASTOS itself is a tool that models all vehicles as rigid bodies. However, there are possibilities to consider multi-body dynamics and even flexible structures.
First of all ASTOS provides a model for robotic arms. This model requires the relative rotational accelerations at each joint of the arm and calculates the overall motion of the multi-body system. The accelerations can be provided by a robotic arm controller modelled in Simulink or via user-defined look-up tables. Of course it is possible to use this model also for other moving parts like large antenna dishes or solar arrays. For latter it is desirable to model also their flexibility. As mentioned before this is not possible with ASTOS directly but via an interface to the ESA-funded DCAP multi-body dynamics simulation software [2]. This is done in a way that the user does not need to interact with the DCAP user interface but the whole DCAP configuration is exported by ASTOS. The results produced by DCAP can provide:

- Time-tagged node positions and accelerations
- A linearized state-space model of the flexible dynamics

As an alternative to the shape import from ASTOS, DCAP can load also NASTRAN files to model the flexible dynamics.

CAMERA AND LIDAR SIMULATOR

The camera and LIDAR simulator is based on VESTA. VESTA is an open-source graphical 3D engine for animation of space scenarios, visualization of mission analysis results and managerial presentations, developed by Astos Solutions GmbH. VESTA itself already provides a lot of features required for photo-realistic image generation based on 3D computer graphics. It is fully implemented in C++ and OpenGL.
An integral part for photo-realistic image generation is the computation of shadows. The camera and LIDAR simulator supports shadows cast between discrete objects as well as self-shadowing, e.g. a shadow cast by a satellite dish on the surface of the associated satellite. Additionally, eclipse shadows are supported, which is quite essential in space scenarios.
The atmosphere visualization is based on physical properties including air density, altitude and temperature. Also physical effects like Rayleigh or Mie scattering [4] are considered in the computation which results in a realistic appearance of atmospheres. Reflections are also simulated by the simulator with certain restrictions. Due to the real-time aspect, only mirror like single reflections are considered. Additionally, no reflections of an object onto itself is simulated.

Sensor Models

The camera and LIDAR simulator supports two camera models and one LIDAR model. Each camera model is able to produce coloured images as well as monochromatic images. Currently, only images with 8-bits per colour channel are generated. The LIDAR model is simulating a scanning LIDAR which uses a single light beam and a configurable scanning pattern.

Camera Models
The first camera model is the so called *Simple Camera*. Basically, this camera model is the standard model used by OpenGL. It can be compared to a pinhole camera. The user defined parameters for this type of camera are mainly the horizontal and vertical resolution, the vertical field of view, the aspect ratio and the frame rate.
A more complex camera model is the so called *Physical Camera*. This camera model is based on the thin lens equation (1) according to [1]. An image generated with this camera model, including defective pixels, is shown in Fig. 5.

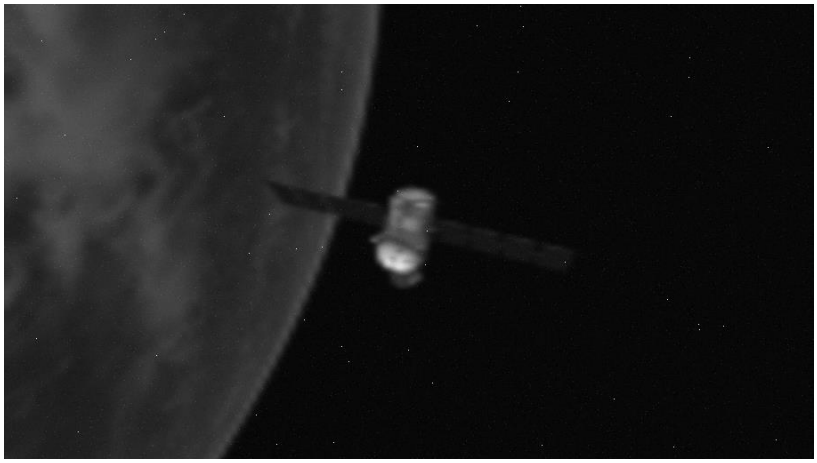

Fig. 5 An image generated by the *Physical Camera* model. The satellite is completely out of focus.

$$1/f = 1/distance_{object} + 1/distance_{image} \quad (1)$$

One major feature of this camera model is the ability to simulate depth of field. Depth of field is quite essential for photo-realistic images, because every real camera provides sharp edges only at a certain distance.
Depth of field is not the only effect simulated by this camera model. Amplifier noise, fixed pattern noise as well as defective pixels are also available and can be enabled if required. Amplifier noise adds Gaussian distributed noise to each pixel of the generated image. Fixed pattern noise simply adds a constant noise, which is defined by a noisy image, to the final image.
Defective pixels are defined in groups. Each group is defined by the number of pixels affected by this group divided by the total number of pixels and a RGB colour. The pixel locations for each group are computed randomly at the beginning of the simulation and are kept constant until the end. Each pixel of a group is set to the constant colour defined for that group. With this approach, it is possible to simulate pixel defects like dark or bright dot defects as well as stuck pixels.

LIDAR Model
The LIDAR model implemented in the camera and LIDAR simulator is based on a laser beam which scans the whole scene using a certain scanning pattern. The scanning pattern is based on a combination of a triangular function and a sinusoidal function. The exact pattern is defined as

$$\begin{aligned} \mathrm{x} &= x_0 + A_{sin} * \sin(2 * \pi * f_{sin} * t) \\ y &= y_0 + A_{tri} * \mathrm{sign} * (2 * index_{sample}/samples_{total} - 1) \end{aligned} \quad (2)$$

A_{sin} and A_{tri} are the amplitudes of the two functions. f_{sin} is the frequency of the corresponding sinusoidal function and t is the time elapsed since the beginning of the simulation. x_0 and $y_0$are the user defined center positions which need not the be located at the field of view center of the LIDAR sensor. The triangular function is not directly based on the time t. It uses the index of the current sample which is then related to the elapsed time. $sign$ switches from 1 to -1 or vice versa, every time the triangular function reaches its maximum or minimum.
The LIDAR sensor also provides the possibility to add noise to the generated data. The noise can be applied independently to the sample point locations as well as to the distances computed by the LIDAR model. For each type, the noise added to the data is computed from a fixed bias and a random noise value.

Customized Wavelengths

Normally, the camera and LIDAR simulator evaluates the visible spectrum, which is sampled at three wavelengths, the red, green and blue colour channel. These three colour channels can be interpreted as light emitted at wavelength of approximately 700 nm, 540 nm and 450 nm respectively. In general, the three colour channels are not fixed to these wavelengths. Each colour channel can be interpreted as any wavelength located somewhere in the full spectrum. For

example, the three colour channels can be interpreted as the wavelength, 780 nm, 1100 nm and 1400 nm, which are located in the near infrared spectrum (IR-A).
The big advantage of the camera and LIDAR simulator is that the information about the considered wavelengths is mainly located in the material definition stored inside the 3D model. By adjusting the material properties, it is possible to simulate light emitted at wavelengths outside the visual spectrum.
Because the sensitivity of camera sensors depends on the wavelength, each camera model provides the possibility to specify a luminosity function (3) which converts the three colour channels (R, G, B) into a single luminance value Y.

$$\mathrm{Y} = c_R * R + c_G * G + c_B * B, \qquad c_R + c_G + c_B = 1 \tag{3}$$

c_R, c_G and c_B are defining the amount of luminance which is added by the corresponding colour channel (wavelength). This function is only used for monochromatic image generation which is quite common for cameras capturing light outside the visible spectrum, e.g. thermal infrared.

Interfaces

The camera and LIDAR simulator provides a set of different input and output channels. Some of these channels can be used for input and output. The different channels can be mixed if required, but not all combinations are available for all sensors. For example, the LIDAR sensor only provides a TCP/IP output channel. Nevertheless, the input channel for the LIDAR sensor can be freely selected from the available input channels
During the simulation, the camera and LIDAR simulator requires updated positions and attitudes for each defined object. This data is either defined by a specialized text file, containing time samples for each object position and attitude (given as quaternion) or it is dynamically computed by an external component, which provides this data using a TCP/IP connection. Additionally, some sensor parameters can be changed during simulation in case the TCP/IP input channel is used, e.g. the exposure factor of a camera or the amplitudes of the LIDAR scanning pattern.
For the image data generated by the camera sensors, three output channels are available. Image data can be sent over a TCP/IP connection or a Camera Link® connection. Alternatively, image data can be written as PNG images. For the LIDAR sensor, only the TCP/IP output channel is available.

Simulator Integration

The camera and LIDAR simulator can be integrated into a wide range of simulation environments, due to the large number of input/output channel combinations. Two scenarios are shown in Fig. 6. The easiest way of integrating the simulator is to use it as an offline image generation tool. The input of the object states can be provided by a single text file, containing precomputed trajectory samples. Object states are interpolated linearly between these time samples. The output of the camera and LIDAR simulator is a set of PNG images which are afterwards used by the simulation environment without any direct invocation of the simulator itself.
On the other side, the camera and LIDAR simulator can be fully integrated into a closed loop simulation. This scenario is shown on the right side of Fig. 6. The object states for each frame are computed by the ASTOS library which is running on a standalone PC and are transmitted to the simulator using a TCP/IP connection. The camera and LIDAR simulator than generates an image based on these object states and transmits it to a real-time Simulink model executed on the dSPACE platform. The Simulink model is computing new controls based on visual navigation algorithms operating on the received images. These controls are finally sent back to the ASTOS model which is than computing the new object states for the next frame. This scenario also emphasizes the flexibility gained by not requiring the same computer to provide the input data as well as to receive the generated images.

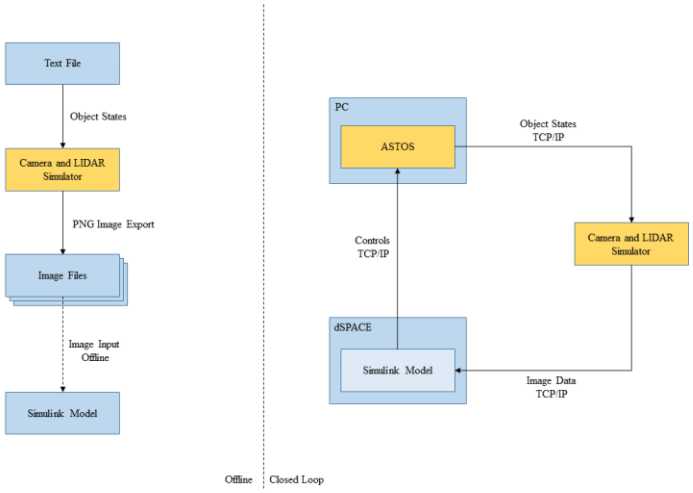

Fig. 6 Two different use cases for the integration of the camera and LIDAR simulator

REFERENCES

[1] "Quad Power – Performance Boost for HIL simulation with new DS1006 Processor Board", *dSPACE Magazine 1/2010*, dSPACE GmbH, Paderborn, 2010

[2] G. Baldesi, T. Voirin, A. Martinez Barrio, M. Claeys, "Methodologies to Perform GNC Design and Analyses for Complex Dynamic Systems Using Multibody Software", *Advances in Aerospace Guidance, Navigation and Control*, Springer, pp.431-450, 2011

[3] E. Hecht, *Optics*, Addison-Wesley, 4th ed., August, 2001.

[4] J. Strutt, "On the electromagnetic theory of light", *Philosophical Magazine*, series 5, vol. 12, pp. 81-101, 1881.