Realizing the eRobotics Approach in Semantics-Based Multi-Domain VR Simulation Systems

Nico Hempe
Institute for Man-Machine Interaction
RWTH Aachen University
Aachen, Germany
hempe@mmi.rwth-aachen.de

Jürgen Roßmann
Institute for Man-Machine Interaction
RWTH Aachen University
Aachen, Germany
rossmann@mmi.rwth-aachen.de

***Abstract*— Multi-domain VR simulation systems provide a framework to bring together various modules to achieve the desired goals. In particular, complex technical or mechatronical systems like mobile robotic applications mostly neglect rendering capabilities; consequently, commonly applied models are designed with minimal effort resulting in purely functional virtual testing environments. The novel field of eRobotics is currently being developed as an intuitively applicable eService related to VR in advanced robotic and engineering applications. We will introduce novel system structures and techniques that meet the requirements to realize a semantics-based multi-domain VR simulation system, which combines complex simulations and state-of-the-art rendering. Data provided by geo information services is used for the creation of close-to-reality virtual environments with minimal effort; additionally, the close interplay of simulation and rendering modules enables the transition from a passive render component toward an active one, further improving accuracy and performance of non-rendering-related tasks like optical sensor simulations.**

Keywords- VR simulation systems; real-time rendering; semantic world modeling; eRobotics

I. Introduction

Virtual Reality (VR) is a commonly used term and designates the conversion of real world information to computer generated, virtual worlds. Apart from entertainment, VR simulation technology is a well-known field of virtual training and engineering today. An important application area of these systems is Virtual Prototyping (VP), where a digital model of a product in development can be experienced prior to construction. By enhancing a Virtual Prototype with further functionalities, which also include the collaboration of the simulated technical components, Virtual Prototypes can be regarded as Virtual Testbeds [1].

Complex technical or mechatronical systems like mobile robotic applications require a large number of different sub-systems to simulate all desired tasks. Multi-domain VR simulation systems represent the top-of-the-range systems that cover multiple technical and visual aspects not limited to a single task or domain. They provide a framework to bring together various simulation and rendering modules in order to achieve the desired goals; however, the amount of included functionalities, frameworks and libraries is limited in common system structures due to complexity, compatibility and performance reasons. In order to address this issue, simulation and rendering modules are usually separated into independent frameworks, which has an obvious reason; both frameworks rely on specific data types and structures to perform best. In particular, applications that integrate state-of-the-art computer graphics demand for highly specific data structures to provide high-quality visualizations with real-time performance. While rendering-related applications traditionally rely on a scene graph structure, a simulation database may be optimized for other purposes using a completely different structure. These structures may hardly fit into each other without affecting description power, flexibility, manageability or performance in a negative way; as a consequence, an application can become render- or simulation-centric. In particular, simulation-centric applications in non-commercial contexts usually do not provide the flexibility required to integrate advanced rendering techniques and, consequently, only provide purely functional virtual environments limited to simple testing scenarios. In this contribution, we present a novel, semantics-based multi-domain simulation and rendering framework that brings together both areas in a holistic engineering support tool based on the principles of eRobotics. As depicted in Fig. 1, this system combines a powerful simulation and render framework with the help of a central semantic database based on the principles of semantic world modeling.

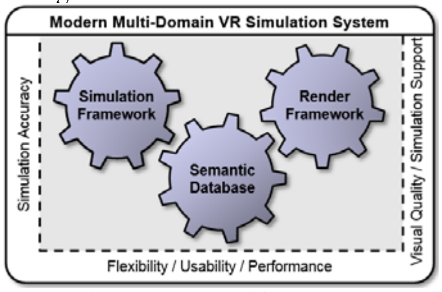
Modern Multi-Domain VR Simulation System
Simulation Framework
Render Framework
Semantic Database
Simulation Accuracy
Visual Quality / Simulation Support
Flexibility / Usability / Performance

Figure 1. Concept of an eRobotics-capable multi-domain VR simulation system that combines complex simulations and attractive virtual worlds.

eRobotics applications aim to allow for accurate virtual testing of newly developed approaches and systems under realistic conditions in order to and make them more robust and to further shift costly and time consuming physical prototypes to the end of the development process. Since simulations get more and more complex and the accuracy requirements rise with advanced technology development, the calculation power of modern graphics hardware must not be neglected for simulation purposes. Apart from the recent trend of general-purpose computing on graphics processing units (GPGPU) [2], synergy effects arising from the combination of simulation and rendering technologies in a holistic eRobotics system can help to face the complexity issue. In particular, the simulation of various sensors is an essential task in robotics-related contexts and part of many current multi-domain VR simulation systems.

In Section 2, related work regarding current mobile robot simulators, the usage of semantic data for graphics applications as well as GPU-based optical sensor simulations will be presented. In Section 3, the eRobotics approach and its goals and challenges are introduced. Section 4 presents the concept and structure of the developed multi-domain VR simulation system, which is capable to meet eRobotic requirements. Section 5 introduced the possibilities for rendering-supported simulations in the fields of optical sensors. Subsequently, Section 6 presents current applications that benefit from the concepts of eRobotics and the presented approaches. Finally, Section 7 concludes this work and gives an outlook to future developments.

II. RELATED WORK

A. *Current Mobile Robotic VR Simulation Systems*

It is still a challenge to develop a comprehensive multi-domain VR simulation system that has the capabilities to serve as a development basis in a wide range of application areas. Regarding the field of mobile robot simulators, well known systems are USARSim [3], Player/Stage/Gazebo [4], ROAMS [5] or WeBots [6], which are applied in a broad range of research and development projects. An in-depth comparison of these systems is given by Staranowicz and Mariottini [7]. Apart from physics simulations and rigid body dynamics, these systems also feature the simulation of various sensors, which are vital for mobile robotics applications. The virtual environments used for testing and evaluation are usually modelled by hand with high effort; hence, available models are usually purely functional and limited to a simple scenario like an indoor environment with several rooms, an alleyway with several houses or a small outdoor environment, as illustrated in Fig. 2. Even if a few published works try to realize more realistic environments [8] [9], these systems are also limited by the provided technical possibilities regarding the integration of state-of-the-are rendering techniques.

Figure 2. Screenshots of current robot simulators. Left: USARSim, middle: Gazebo, right: Webots. Images taken from the developers' websites.

B. *Semantic World Modeling*

Semantic world models provide a powerful basis to combine different data structures in a single system database. They are stored in schema-aware graph databases, which contain elementary facts and rules and allow a set of nodes with dynamic attributes to be arbitrary linked to other nodes through edges [10]. Semantic world models can describe the environment much more detailed and are not limited to spatial relations or geometry such as predefined data structures like scene graphs. They can contain almost any kind of data and give complex information about the surrounding. Graph databases can also be realized following the object-orientated paradigm in order to make them practical for software systems. These databases are called Graph-Oriented Object Databases (GOOD) [11].

C. *Applying Semantic Data to Graphics Applications*

Real-time graphics applications demand for highly-specialized data structures like scene graphs to efficiently integrate state-of-the-art rendering techniques and to apply optimization approaches like culling and batching in order to grant real-time performance. The general concept of semantic graph databases does not fulfill these requirements; hence, Mendez et al. suggest to separate semantics from rendering in the scene graph by using semantic tags as attributes [12]. Recently, Tobler picked up this idea and suggested a more complete solution by fully separating semantics from rendering [13]. He introduced a split scene graph architecture, which contains a separate semantic and rendering scene graph as illustrated in Fig. 3.

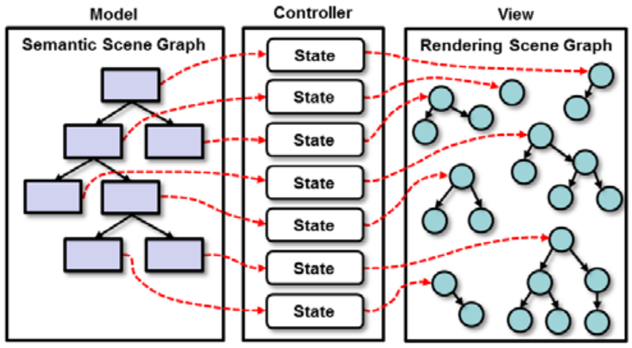

Figure 3. Illustration of the split scene graph approach to separate semantic from rendering data.

This concept follows the Model-View-Controller (MVC) design pattern, which allows for the usage of a single model under different environments by separating the model from the system view using a controller that coordinates the translation [14]. It can be used to separate the user interface from the computation parts. The semantic scene graph represents the model as created by the user, the rendering scene graph represents the view necessary for the system environment and the rule objects represent the controller that dynamically translate the model view into the system view.

D. *Rendering-Supported Optical Sensor Simulations*

Different classes of optical sensors are an important part in robotics-related domains, which can benefit from rendering support. While computer graphics applications typically mimic a pin-hole camera to generate ideal imagery, real world cameras do not meet that quality because of technical and physical limitations. A popular example for accurate camera simulations is the Image Systems Evaluation Toolbox (ISET) [15] implemented in MATLAB, which considers specific sensor- and optics-related effects as well as processor and display properties. Recently, Kucis and Zemcik [16] demonstrated, how these effects can be simulated on the GPU with post-processing shaders.

Regarding light detection and ranging (LiDAR) scanners, many currently available simulations are based on CPU-based ray-object intersection tests or simple depth render passes for performance reasons. A more accurate simulation was presented by Kukko and Hyyppä [17], which also features specific error models.

Time-of-Flight (ToF) cameras are based on a hybrid technical approach and combine the advantages of digital camera systems and LiDAR scanners to provide the distance to objects in a 3D scene based on the traveling time of a short emitted infrared light pulse. Many current ToF sensor simulations are based on MATLAB and are not real-time capable [18]. A recent survey of the field of GPU-supported ToF simulations is given by Kolb et al. [19]. Exemplary implementations that apply the GPU for ToF simulations have been introduced by Keller and Kolb [20].

III. THE EROBOTICS APPROACH

The research field of eRobotics has been established recently in order to cope with the inherent complexity of advanced robotics and mechatronics development [21]. It also helps to ease the development and to significantly cut down the cost. The aim of eRobotics is to provide a holistic but comprehensive software tool for the development of complex technical systems to be able to efficiently represent the inherent capabilities of a robotic system to an application developer and to the user. The applied render framework plays a vital role that has mostly been neglected in the past, in particular for expert systems; however, no modern VR simulation systems can afford to dismiss state-of-the-art computer graphics. Even in small projects, the vital role of quickly available, visually pleasing demonstrations of project ideas and results in attractive virtual worlds as a sales argument for possible follow-up projects must not be neglected. This includes all common VR domains ranging from education over 3D simulations and virtual training up to multi-domain simulations as illustrated in Fig. 4.

A key feature of eRobotics systems is the central semantic database, which enables semantic world modeling. Semantic databases can contain almost any kind of data and give complex information about the surrounding. This information can be used for various applications and offer a huge potential for the render component. If interpreted correctly, semantic information can be applied to greatly improve the visual quality of virtual scenes, which were not built regarding the integration of state-of-the-art rendering techniques.

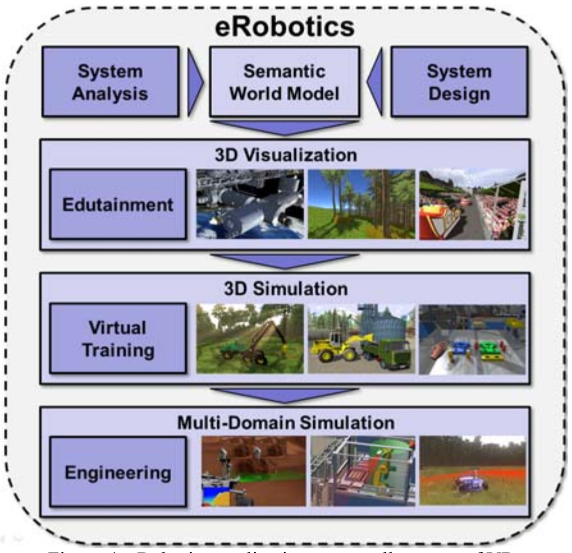

Figure 4. eRobotics applications cover all aspects of VR.

A. *Challenges and Development Goals*

One reason for the gap between simulation complexity and rendering quality in multi-domain VR simulation systems is the fact that real-time computer graphics is one of the fastest evolving areas in computer science. In no other area new techniques and approaches are presented more quickly due to the enormous progress of processing power and features provided by current graphics hardware. Established data structures can undergo fundamental changes with the ongoing evolution and new technical features may not efficiently fit into the current implementation; as a consequence, a system based on a too restrictive data structure can become outdated quickly. It is still a challenge to develop a holistic, comprehensive and sustainable multi-domain VR simulation system, which keeps easily manageable, extendible and understandable when exceeding a specific range of features. The combination of a highly-flexible system architecture with a central, semantics-based database provides the possibilities to face the arising challenges. While common render frameworks demand for a spatial, render-centric data structure in order to perform best, semantic databases are not optimized for rendering purposes; consequently, new challenges for the render framework arise, which has be able to

- deliver realistically looking virtual worlds out of semantic data sets not built for rendering purposes.
- extract and optimize geometry out of semantic datasets on-the-fly to grant real-time performance.
- actively support the simulation process in specific tasks in order to improve accuracy and performance.

The broad range of domains and applications of eRobotics systems create some challenges that have to be solved in order to realize a practical and sustainable system, which is capable of integrating a state-of-the-art render framework. The system has to

- be built in a modular manner to flexibly adapt to new domains with minimal effort.
- allow for intercommunication of all included components without creating dependencies.
- provide a database that is capable of storing a broad range of data structures and types.
- enable the usage of modern GPU features and rendering techniques to deliver highly accurate real-time results suitable for engineering tasks.

IV. AN EROBOTICS-CAPABLE MULTI-DOMAIN VR SIMULATION SYSTEM

A. *General System Architecture*

In order to benefit from a holistic system concept and to be able to explore the provided technical possibilities, the close interplay of all modules involved is a vital requirement, regardless if they are simulation- or rendering-related. Fig. 5 depicts the general system structure, which enables the realization of such a system.

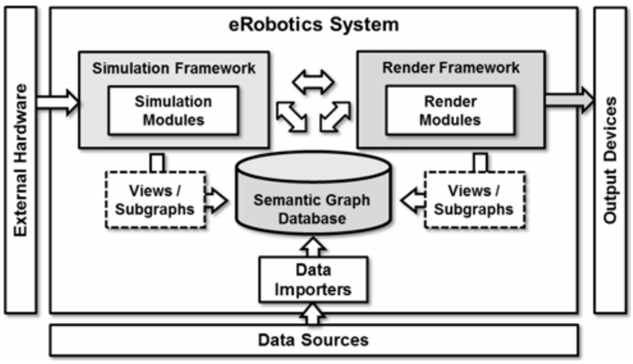

Figure 5. General structure of the system, which allows for the integration and intercommunication of simulation- and rendering-related modules.

The key idea is to base simulation and rendering modules on a central, active and extendible object oriented graph database. Each module can define its own view to the database and integrate subgraphs to meet its specific data structure requirements. To properly react to data manipulations and to enable the intercommunication of all modules involved, an active database concept is vital. Apart from the data itself, active databases further implement algorithms to modify and supervise the contained data as illustrated in Fig. 6; in addition, the active database integrates an intelligent messaging system that informs the system components of specific events like creation, data manipulations or deletion, necessary to apply the MVC pattern for the creation of rendering-optimized scene graphs out of the semantic datasets. To minimize dependencies between the modules and to realize a decoupled observer pattern, the signals and slots pattern as introduced by the Qt framework was chosen for the realization [22].

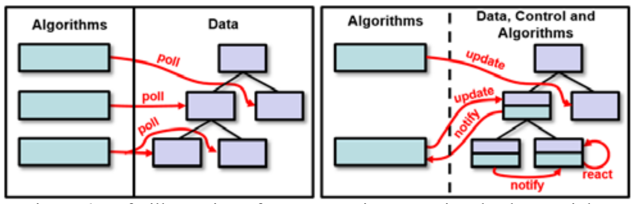

Figure 6. Left: illustration of a system using a passive database. Right: active database that directly integrates data monitoring and functionalities.

B. *Micro-Kernel-Based System Architecture*

In accordance with the flexibility requirements to realize a sustainable and manageable system, a modular architecture is vital, in particular for the render component that has to quickly adapt to novel technical trends. The micro-kernel pattern is a popular approach to address the complexity problem. Originally developed for operating systems and embedded systems [23], it can also be applied to software systems that have to adapt to changing conditions. The main idea is to separate a minimal functional core from extended functionality and project-specific parts. As illustrated in Fig. 7, the micro-kernel defines the very basic structure of the framework and also serves as a socket for plugging in extensions and coordinating their collaboration.

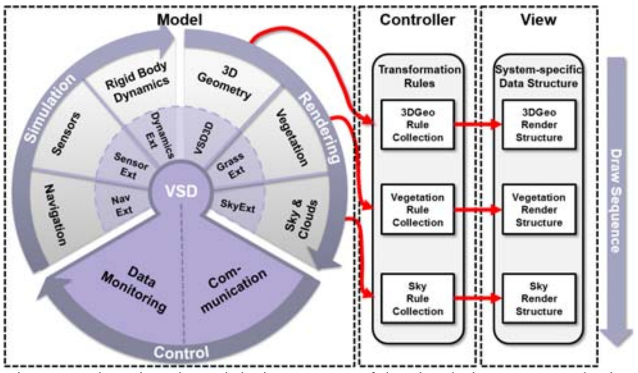

Figure 7. The micro-kernel design pattern of the simulation system. Plugins can extend functionalities with new algorithms and self-defined elements.

For the VEROSIM Simulation System, which is co-developed at our institute in cooperation with our industrial partners, this micro-kernel is called the Versatile Simulation Database (VSD) [24]. Fully implemented in C++, the VSD further provides the central building blocks for data management, meta information, communication, persistence and user interaction. Plugins integrate new functionality and extend the database by defining new semantic elements. Regarding the realization of the MVC pattern, the 3D geometry plugin defines a set of semantic nodes (VSD3D), which can be used to model the scene in a “natural” fashion. The plugins further define a set of rules that transform these semantic nodes into a render-optimized data structures automatically by using the data monitoring possibilities provided by the VSD. The used models can completely be decoupled from system-specific requirements, which enables ease of modeling and the import of a wide range of available datasets. As illustrated by the vegetation and sky render plugins in Fig. 7, additional effects can be integrated similarly and implement their own domain-specific data structures and transformation rules for efficient rendering. The corresponding transformation rules can easily be modified to adapt to other systems with little effort; in contrast to the split scene graph approach, the plugins in our approach keep track of all related semantic nodes and organize them globally, enabling efficient implementations regarding buffer usage, state-sorting and batching.

C. Enhancing Virtual Environments by Using Semantic Datasets

The presented system structure provides the flexibility to import data from various data sources and allow their interpretation for rendering-related tasks as illustrated in Fig. 8. For illustration reasons, the central database is separated into a semantic and a geometry-related part. While simulation plugins can access and modify all data in order to visualize and represent their results and, moreover, all plugins are able to directly communicate through the integrated signal and slots pattern, the render plugins mainly work with the imported data in three different ways.

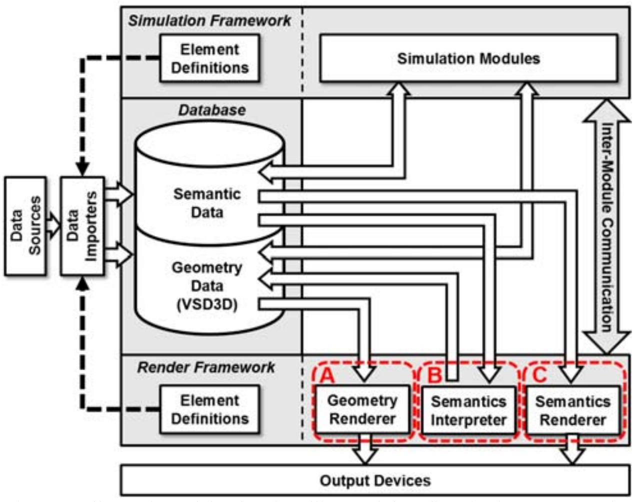

Figure 8. Illustration of the data handling and data flow in the systems. The three common ways how the render framework can work with semantic data are marked with A, B and C.

The first way (A) is straightforward: geometry information can directly be handled and rendered by the 3D geometry render plugin. The VSD3D geometry elements form the traditional rendering-related part of the database as used by many other render frameworks, which consists of geometry, transformations, textures and materials. Data received from external sources usually contain information that can directly be converted into these elements. The importer extracts this information and writes it into the geometry part of the central database. In the second case (B), semantic data may not contain information that can be rendered directly, but information that can be processed by interpreter plugins, which convert them to VSD3D elements; subsequently, these elements are added to the geometry database and handled by the 3D geometry renderer, similar to the first case. Finally, semantic elements can be interpreted and converted into specific, rendering-optimized data structures by semantic render plugins that meet the domain-specific requirements (C). The created data structure is not written to the geometry database but organized and handled by the corresponding plugins in domain-specific views or subgraphs.

The three introduced ways of data handling are sufficient to automatically generate and render highly realistic virtual environments from data imported from available data sources like Geographic Information Services (GIS) instead of modeling them manually with high effort. GIS servers are widely available to the public nowadays, but usually contain only minimal geometric information like the terrain geometry or surface textures, which can be rendered directly; however, they provide a lot of semantic information that describe the environment more detailed, such as Digital Elevation Models (DEM), satellite imagery, land coverage, infrastructure or even weather conditions. Multiple semantics-based render modules have been developed and integrated that work with this data and integrate effects like realistic ground vegetation or a dynamic sky with clouds and weather effects, which rely on specific render data structures to work satisfactory [25] [26]. These scene descriptions have the potential to go way beyond common, hand-modelled digital environments regarding realism, accuracy, scale and rendering quality. An example is given in Fig. 9, which illustrates the benefits of semantic data interpretation from data received from GIS servers for the rendering of vivid, life-like virtual outdoor environments.

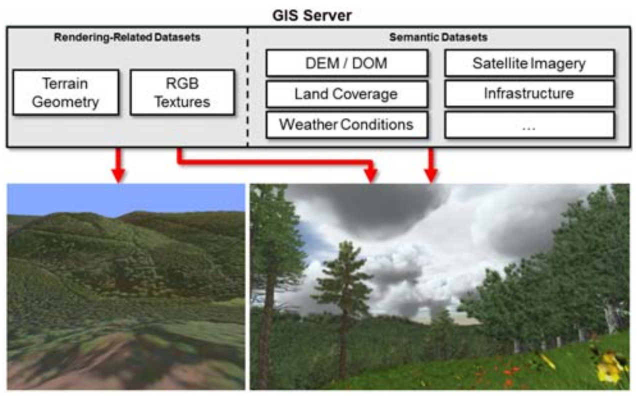

Figure 9. Comparison of virtual environments rendered out of datasets commonly provided by GIS servers with and without semantically enhanced rendering.

D. Supporting the Simulation with Render Modules

eRobotics applications are not only aimed at improving the visual quality in engineering domains, they also aim to allow for accurate virtual testing of newly developed approaches and systems under realistic conditions. Optical sensor simulations are an essential field in robotic applications and accurate virtual testbeds rely on close-to-reality sensor data streams. The presented system concept further allows for rendering-supported simulations, which will be addressed in the following section.

V. RENDERING-SUPPORTED OPTICAL SENSOR SIMULATIONS

The presented simulation system features a sensor framework that allows to conveniently develop applications with sensor support [27]. As illustrated in Fig. 10, an application can request a sensor data stream through a sensor abstraction layer, which triggers a simulated or a real world sensor to provide the acquired data to the application for further processing. In an ideal case, the application performs similarly with both data streams. The general simulation process consists of a simulation plugin, which can apply rendering techniques and data provided by the render framework to support accurate real-time sensor simulations.

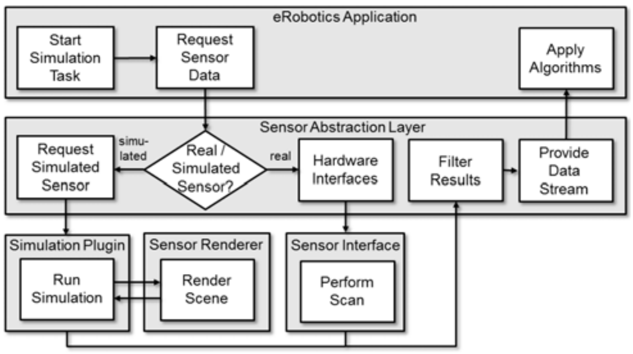

Figure 10. A hardware abstraction layer is used to decouple application development from the sensors, regardless if real world or simulated ones.

A. *Digital Camera Simulation*

The usage of the render component for the simulation of digital camera systems is the most straightforward use case, since realistically rendered virtual environments are capable of reaching an image quality close to photorealism. Even if it is a common practice to directly use rendered images for computer vision or image processing approaches, they strongly differ from images acquired by real world cameras due to physical and technical limitations. These effects need to be considered to improve the robustness of developed approaches and to properly simulate real world conditions. The structure of the camera simulation is depicted in Fig. 11. Before the render pass is triggered, the camera simulation plugin integrates a range of shaders into a flexibly configurable camera effects rendering pipeline, which is applied as post-process. Currently, the simulation supports a range of optical features and sensor features, which can be observed in imagery acquired from real world cameras.

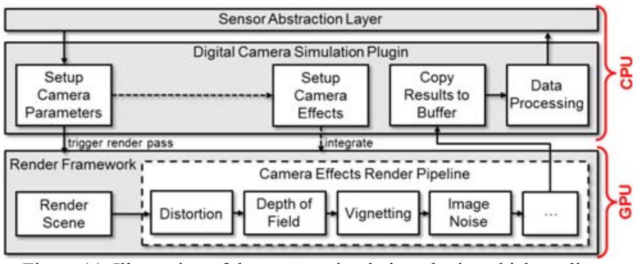

Figure 11. Illustration of the camera simulation plugin, which applies a range of post-processing rendering effects to simulate camera features.

The simulation includes optical features like geometric distortion, chromatic aberration, depth-of-field and vignetting as well as sensor features like image noise and light bleeding, which can be simulated by using high dynamic range (HDR) rendering techniques. After the pipeline execution, the image is optionally filtered and finally provided to the application. As illustrated in Fig. 12, a broad range of digital camera systems with matching effects can efficiently be simulated by combining all these effects.

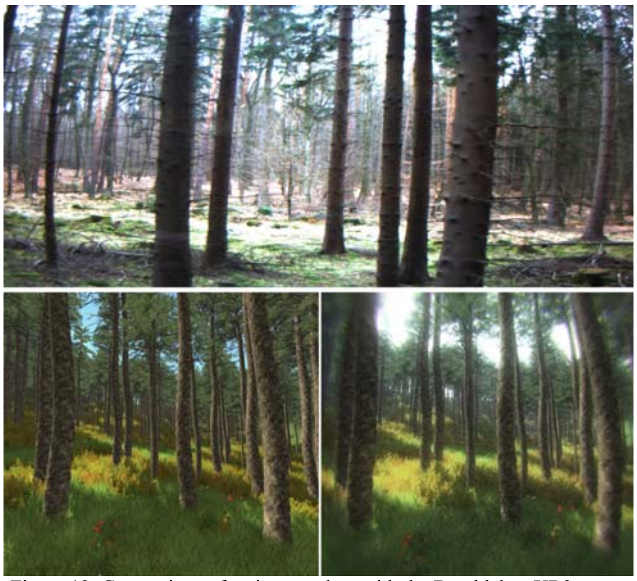

Figure 12. Comparison of an image taken with the Bumblebee XB3 stereo camera (top) and the corresponding rendered image without (bottom left) and with (bottom right) the simulation of typical camera effects.

B. *LiDAR Simulation*

Apart from camera simulations, the benefits of rendering support for other kinds of optical sensors are obvious: Measurements of optical sensors depend on various consistencies regarding reflections on the surfaces. The renderer usually has this information, which describes the surface in more detail in order to apply high-quality rendering techniques, which rely on textures, gloss maps, normal maps or displacement maps. These resources can also be used to further improve the simulation accuracy of LiDAR sensors. Similar to the camera simulation, the LiDAR simulation plugin integrates a post-processing chain to properly simulate the depth estimation of the real world sensor. While simple GPU-based LiDAR simulations only use the depth information of the render pass, thus, returning ideal simulation results, several aspects need to be considered for results comparable with those acquired from real world sensors. An important property that is given for all scanners is the minimum remission value (commonly between 2% and 10%), which defines the amount of emitted energy that need to be reflected back to the sensor to be able to correctly measure the distance; in addition, the noise level increases with decreasing remission. Fig. 13 demonstrates the benefits of a rendering-supported LiDAR simulation, which uses rendering-related data and techniques to simulate the surface reflectance more accurately compared to simple CPU-driven ray intersection approaches. Depending on the calculated remission, a matching amount of statistical noise is added to the simulated depth measurements.

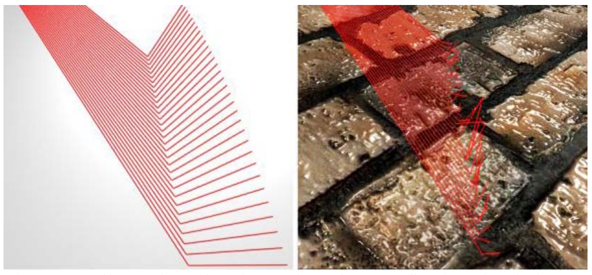
Figure 13. Illustration of a LiDAR laser beam reflection simulation on surfaces without (left) and with (right) the usage of render data and techniques to properly estimate the remission value.

C. *Time-of-Flight (ToF) Camera Simulation*

ToF cameras rely on a hybrid technical approach and combine the advantages of digital cameras and LiDAR scanners. They capture the depth information of a whole 3D scene at once with up to 160Hz instead of relying on point-by-point laser beam scanning; however, these sensors also suffer from limitations similar to those occurring at digital cameras. Fig. 14 shows images that were captured with the MESA SR4000, which illustrate typical problems.

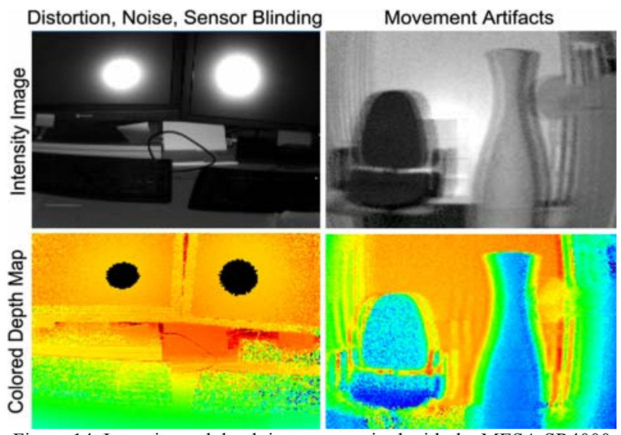

Figure 14. Intensity and depth images acquired with the MESA SR4000 ToF sensor, which demonstrate typical measurement errors.

The effects of distortion, noise and sensor blinding are comparable to those observed at digital camera and LiDAR systems. In the resulting point cloud, noise typically leads to trembling results (known as “wiggling”), while depth measurements in a pixel that observes a region with inhomogeneous depth can lead to flying pixels at the objects’ edges. While simple ToF sensor simulations sample the scene’s depth information and directly generate an ideal point cloud, we developed a GPU-based simulation pipeline that is capable of simulating the whole processing chain of ToF sensor systems with matching error models as shown in Fig. 15. To reduce wiggling effects, the data stream in real world applications is usually filtered by the scanner itself, the driver or a specific software before it is passed to the actual application; hence, the simulation also provides an optional filter pass.

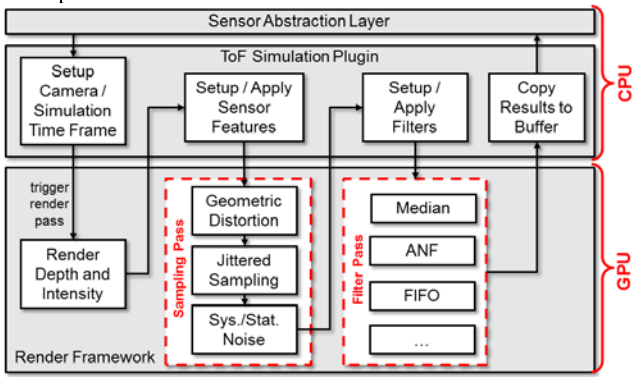

Figure 15. Structure of the ToF camera simulation plugin.

The simulation steps are illustrated in Fig. 16. First, geometric distortion for the acquired intensity image is simulated similarly to the digital camera simulation. Second, bilinear and jittered sampling between the pixel regions is used to properly simulate flying pixels and to produce more irregular results, which better match the wiggling behavior of real world sensors. Finally, noise is added; in contrast to normal distributed statistical noise, the statistical noise of ToF images tend to look like a nearly periodic, sine-like function [19], which is also considered in the simulation. It becomes clear that ideal results as shown in the top-middle image of Fig. 16 strongly differ from the results achieved under consideration of all technical aspects of a real world sensor as shown in the top-right image.

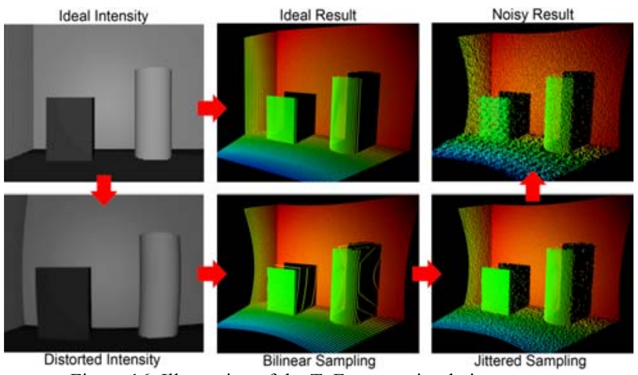

Figure 16. Illustration of the ToF sensor simulation process.

Finally, sensor blinding and motion artefacts are considered. Sensor blinding is imitated by returning a measurement error for regions with too high intensity values as observed on highly reflective surfaces. For the simulation of motion artefacts, the scene’s intensity is rendered four times using HDR techniques with different exposures to imitate the depth sampling process of the real world sensor. In each pass, the camera and the objects are moved with a quarter time frame; as a result, the typical artefacts can be observed at the objects edges in the calculated depth values. The achieved results are shown in Fig. 17, which are comparable to the real world results shown in Fig. 14.

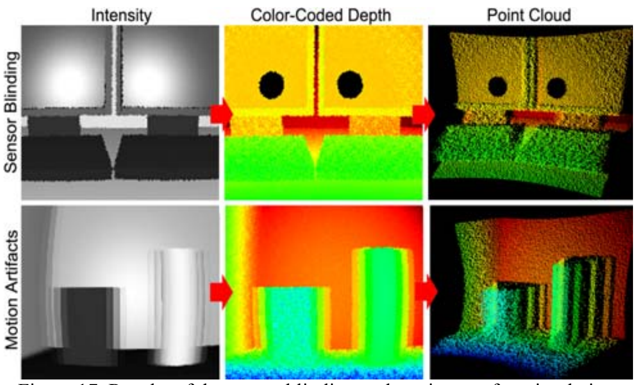

Figure 17. Results of the sensor blinding and motion artefact simulation.

VI. Applications

In this section, selected applications that benefit from the usage of realistic virtual testing environments as well as rendering-based simulation techniques are presented. Fig. 18 shows the currently developed wood harvester simulator, which imports relevant data directly from GIS servers. The simulator applies the presented approaches to generate a realistic virtual environment with minimal effort instead of relying on manually created models. On a standard consumer PC equipped with an Intel Core i7-3930K and a NVIDIA GeForce 780 GTX, the simulation runs well above 60 fps. The machine operator can not only be trained in a realistically looking virtual environment, the system further creates a virtual mapping of the real world area he or she will later work in.

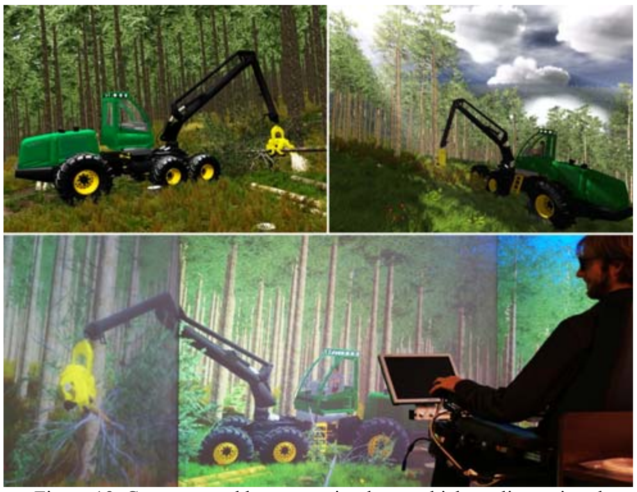
Figure 18. Current wood harvester simulator, which applies a virtual mapping of the real world environment to increase training effectiveness.

All these aspects not only increase the simulation realism and training effectiveness, they are also highly attractive and joyful to use. The virtual environments achievable with the presented multi-domain VR simulation system not only allow for edutainment and virtual training applications, they are also well suited to be used as close-to-reality virtual testing grounds for mobile robots in advanced virtual testbeds; in contrast to the limited testing environments and the mostly functional graphical presentations of common multi-domain VR simulation systems, the presented simulation system not only allows for attractive presentations and visualizations, it further profits from the concepts of rendering-supported simulations to provide highly accurate sensor data streams. Fig. 19 shows the simulation of the SeekurJr mobile robot equipped with a stereo camera, a ToF camera and a LiDAR scanner to support navigation and localization techniques.

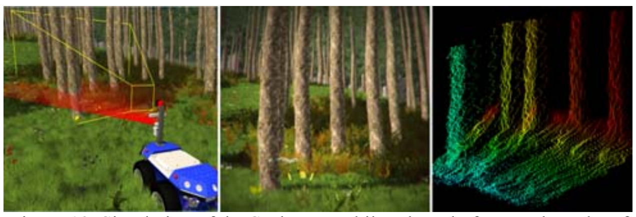
Figure 19. Simulation of the SeekurJr mobile robot platform and results of the simulated stereo camera (middle) and a ToF camera (right).

The SeekurJr is a skid steer, all-weather robot platform intended to be used as a research platform for the development and testing of mobile robotics related approaches in outdoor environments. Since real world experiments and physical setups can be complicated and time consuming to realize, the simulation setup provides a cost-efficient and convenient alternative for development support and testing in early project phases. Novel ideas as well as early implementations and approaches can be tested in the virtual testbed right from the beginning of the project. When a physical setup is realized in later project phases, it already features detailed tested and more robust algorithms, thereby shortening the physical testing period. One of the projects related to the SeekurJr mobile robot platform was the development of a landmark-based localization framework, which is able to estimate the current position of mobile systems in a forested environment [28]. Since it is not possible to directly test every feature or variation of the developed algorithms in a real world environment at reasonable costs and time efforts, physical testbeds are commonly used to test the algorithms under laboratory conditions at a smaller scale.

Fig. 20 illustrates typical testing setups for the development of computer vision algorithms based on a simulated stereo camera and the corresponding stereo matching results. The middle part of the figure shows a simple virtual testing environment, which is typically used corresponding testing scenarios. Even if the simulated stereo camera already features the introduced optical and sensor-based error models as observed by real world cameras, the quality of the stereo matching results seems to be too close to ideal results and differ strongly from the results achieved with a physical testbed as shown in the left part of the figure. In a worst case scenario, these differences can cause a malfunction of the developed algorithms under real world conditions, even if they worked satisfactorily and reliably in the virtual testbed.

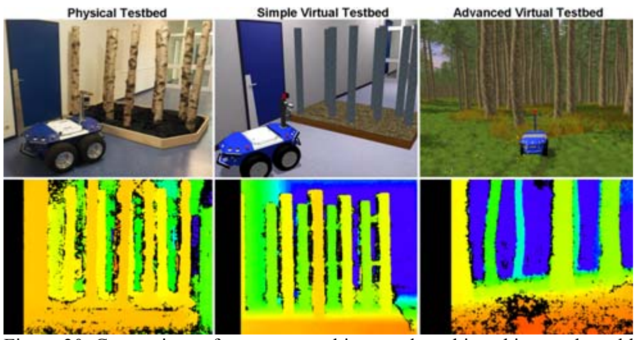

Figure 20. Comparison of a stereo matching results achieved in a real world testbed, a simple virtual testbed and a virtual testbed that provides realistic virtual testing environments.

The right part of Fig. 20 shows results achieved with the close-to-reality virtual testbed using the fully dynamic virtual forest model; in contrast to simple virtual scenarios, the algorithms can directly be tested in highly configurable, large-scale environments, which also incorporate dynamic, volumetric vegetation, realistic lighting conditions with dynamic shadows and different sun positions as well as dynamic weather conditions. On the test system, the shown virtual testbed with a simulated stereo and ToF camera and a 2D LiDAR scanner runs at approx. 45 fps. The results are comparable with those reached in real world tests [29].

The last project presented in this section should demonstrate the aspects of knowledge transfer and the possibilities of early, visually pleasing presentation of novel project ideas and first results with low effort. The basic idea of this project was to mount LiDAR scanners on a wood harvester in order to apply the computer vision and localization approaches developed for the SeekurJr navigation framework. This navigation system should support the machine operator by guiding him to selected trees in the stand marked for felling to improve the wood harvesting effectiveness. Due to the usage of the realistic virtual forest models, the simulated LiDAR scanners and the computer vision algorithms developed for the SeekurJr testbed, a fully functional simulation of such a harvester was created within a short period of time and with low efforts. This virtual testbed is shown in Fig. 21, which was used to vividly demonstrate the possibilities and advantages of the project ideas in an attractive virtual environment. During the project, the simulation models and approaches were refined for efficient testing of the newly developed algorithms before they were applied to the real harvester as illustrated in the right part of the figure. Because of the high running costs of the real harvester and the little time it was available for testing reasons, it becomes clear that the project strongly benefited from the usage of the developed virtual testbed.

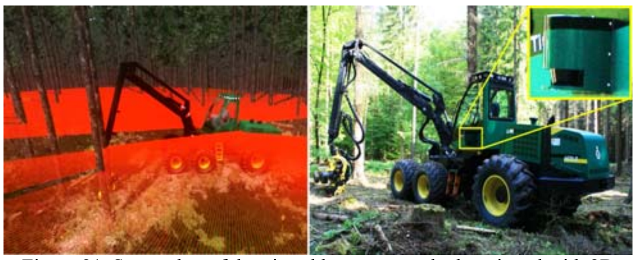
Figure 21. Screenshot of the virtual harvester testbed equipped with 2D LiDAR scanners for forest navigation and harvesting support (left) and the real world setup that was realized after detailed testing and optimization in the virtual testbed (right).

VII. Conclusion and Future Work

In this contribution, we presented the novel eRobotics approach, which aims at providing a comprehensive software environment to realize complex robotic simulations with realistic virtual testing environments. We showed, that semantic world models provide a huge potential to realize such systems. The presented, flexible system structures in combination with the active, object-oriented graph database enable rendering components to efficiently work on semantic datasets. The system is capable to provide realistic virtual testing environments that can be generated from readily available data sources like GIS servers with little effort, suitable to be used as attractive edutainment and virtual training applications. Apart from the visual enhancements, the interconnected system components enable active rendering components, which directly support optical sensor simulations in order to achieve close-to-reality results, hardly possible with common approaches or models. Our future work will concentrate on the extension of the addressed domains as well as the integration of new semantics-based algorithms to further enhance the visual quality of current and future eRobotic applications. Finally, further domains like flow simulations and data visualizations are currently evaluated that can benefit from the concepts of rendering-supported simulations. Up to now, more than 30 different eRobotic applications have been realized with the presented multi-domain VR simulation system [30]. Fig. 22 shows a selection of these applications, which benefit from the presented approaches and range from industrial over terrestrial up to space robotics applications.

Acknowledgment

The work presented in this paper was done as part of the Virtual Forest project. The Virtual Forest project is co-financed by the European Union and the federal state of North Rhine-Westphalia, European Regional Development Fund (ERDF). Europe - Investing in our future.

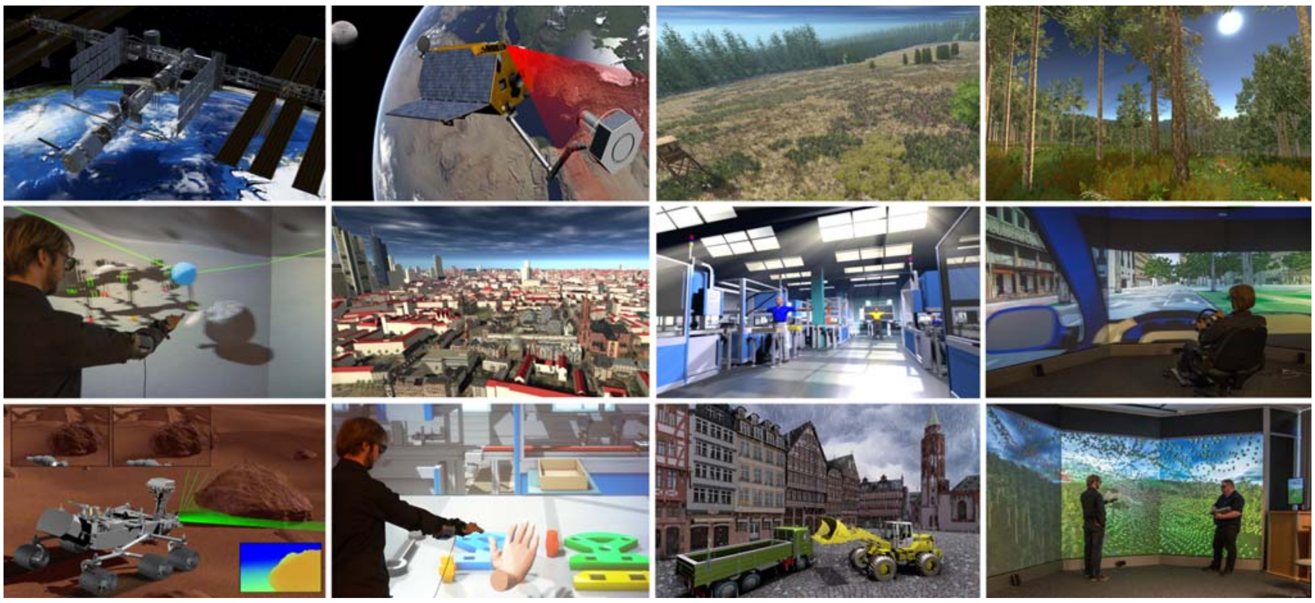

Figure 22. More than 30 eRobotics applications in different domains have been realized until today.

REFERENCES

[1] J. Roßmann, and B. Sommer, "The Virtual Testbed: Latest Virtual Reality Technologies for Space Robotic Applications," Proc. of 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2008), pp. 1–8, 2008.

[2] J. Sanders, and E. Kandrot, "CUDA by Example: An Introduction to General-Purpose GPU Programming," Addison-Wesley Longman, Amsterdam, 1st edition, July 2010.

[3] B. Balaguer, S. Balakirsky, S. Carpin, M. Lewis, and C. Scrapper, "USARSim: a validated simulator for research in robotics and automation," Workshop on Robot Simulators: Available Software, Scientific Applications, and Future Trends at IEEE/RSJ, 2008.

[4] N. Keonig, and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," Proc. of IEEERSJ International Conference on Intelligent Robots and Systems IROS, vol. 3, pp. 2149-2154, 2004.

[5] A. Jain, J. Huineau, C. Lim, W. Lincoln, M. Pomarantz, G. Sohl, and R. Steele, "Roams: Planetary surface rover simulation environment," Proc. of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2003), pp. 19-23, 2003.

[6] O. Michel, "Cyberbotics Ltd. Webots TM : Professional Mobile Robot Simulation," Int. Journal of Advanced Robotic Systems, pp. 39-42, 2004.

[7] A. Staranowicz, and G.L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," Proc. of the 4th International Conference on Pervasive Technologies Related to Assistive Environments PETRA 11, vol. 1, pp. 39-42, 2011.

[8] J. Alemany, and E. Cervera, "Design of High Quality, Efficient Simulation Environments for USARSim," Technical Report ICC 2011-10-1, University Jaume I, Castello, Spain, 2011.

[9] R. Rathnam, M. Pfingsthorn, and A. Birk, "Incorporating large scale SSRR scenarios into the high fidelity simulator USARSim," IEEE International Workshop on Safety, Security and Rescue Robotics SSRR 2009, Denver, USA, pp. 1-6, 2009.

[10] J.R. Abrial, "Data semantics," Proc. of the IFIP Workshop Conference on Data Base Management, pp. 1-60, Cargése, Frace, 1974.

[11] M. Gyssens, J. Paredaens, J. Van Den Bussche, and D. Van Gucht, "A graph-oriented object database model" IEEE Transactions on Knowledge and Data Engineering, vol. 6(4), pp. 572-586, 1994.

[12] E. Mendez, G. Schall, S. Havemann, D. Fellner, D. Schmalstieg, and S. Junghanns, "Generating Semantic 3D Models of Underground Infrastructure," IEEE Computer Graphics and Applications, vol. 28(3), pp. 48-57, 2008.

[13] R.F. Tobler, "Separating semantics from rendering: a scene graph based architecture for graphics applications," Visual Computer, vol. 27(6-8), pp. 687-695, 2011.

[14] E. Gamma, R. Helm, R. Johnson, and J.M. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley Professional Computing Series, vol. 206, 1994.

[15] J.E. Farrell, F. Xiao, P.B. Catrysse, and B.A. Wandell, "A simulation tool for evaluating digital camera image quality," Proc. of the SPIE, vol. 5294, pp. 124-131, 2004.

[16] M. Kucis, and P, Zemcik, "Simulation of Camera Features," Proc. the 16th Central European Seminar on Computer Graphics, pp. 117-123, 2012.

[17] A. Kukko, and J. Hyyppä, "Laser Scanner Simulator for System Analysis and Algorithm Development : a Case With Forest Measurements," ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, pp. 234-240, 2007.

[18] V. Peters, and O. Loffeld, "A bistatic simulation approach for a high-resolution 3D PMD (Photonic Mixer Device)-camera," International Journal of Intelligent Systems Technologies and Applications, vol. 5(3-4), pp. 414-424, 2008.

[19] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-Flight Cameras in Computer Graphics," Computer Graphics Forum, vol. 29(1), pp. 141-159, 2010.

[20] M. Keller, and A. Kolb, "Real-time simulation of time-of-flight sensors," Simulation Modelling Practice and Theory, vol. 17(5), pp. 967–978, 2009.

[21] J. Rossmann, "eRobotics: The symbiosis of Advanced Robotics and Virtual Reality Technologies," Proc. of the 32nd ASME Computers and Information in Engineering Conference CIE 2012, vol. 2, pp. 1395-1403, Chicago, USA, 2012.

[22] J. Blanchette, and M. Summerfield, "C++ GUI Programming with Qt 4," 2nd edition, Prentice Hall, 2008.

[23] D.B. Golub et al., “Microkernel Operating System Architecture and Mach,” Proc. of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures, pp. 11-30, 1994.

[24] J. Rossmann, M. Schluse, C. Schlette, and R. Waspe, “A New Approach to 3D Simulation Technology as Enabling Technology for eRobotics,” 1st International Simulation Tools Conference & EXPO 2013 (SIMEX'2013), pp. 39-46, Brussels, Belguim, 2013.

[25] N. Hempe, and J. Rossmann, “Efficient Real-Time Generation and Rendering of Interactive Grass and Shrubs for Large Sceneries,” Proc. of the 13th IASTED International Conference on Computer Graphics and Imaging (CGIM 2012), pp. 240-247, Crete, Greece, 2012.

[26] N. Hempe and J. Rossmann, “Geometric Interpretation and Optimization of Large Semantic Data Sets in Real-Time VR Applications,” Proc. of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2012), pp. 1403-1412, Chicago, USA, 2012.

[27] M. Emde, J. Roßmann, B. Sondermann, and N. Hempe, ”Advanced Sensor Simulation in Virtual Testbeds: A Cost-Efficient Way to Develop and Verify Space Applications,” Proc. of AIAA Space 2011 American Institute of Aeronautics and Astronautics (AIAA) Confrence and Exposition, pp. 1-11, 2011.

[28] J. Rossmann, P. Krahwinkler, and A. Bücken, ”Mapping and navigation of mobile robots in natural environments,” German Workshop on Robotics – Advances in Robotics Research - Theory, Implementation, Application, pp. 43-52, 2009.

[29] N. Hempe, J. Rossmann, and B. Sondermann, “Generation and Rendering of Interactive Ground Vegetation for Real-Time Testing and Validation of Computer Vision Algorithms,” Electronic Letters on Computer Vision and Image Analysis (ELCVIA), vol. 12(2), pp. 40-53, 2013.

[30] J. Rossmann, “Advanced and Efficient Space Robotics Development: The eRobotics Approach,” Proc. of the 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2013), pp. 1-8, Noordwijk, The Netherlands, 2013.