New Methods of Render-Supported Sensor Simulation in Modern Real-Time VR-Simulation Systems

Jürgen Rossmann, Nico Hempe, Markus Emde
Institute for Man-Machine Interaction
RWTH Aachen University
GERMANY
{rossmann, hempe, emde}@mmi.rwth-aachen.de

Abstract: In recent years, VR-Simulation Systems have taken up an essential part in engineers’ and researchers’ daily work. For the close-to-reality simulation of entire environments the term ‘Virtual Testbeds’ has been coined. In a Virtual Testbed, complete mission scenarios are simulated as a whole instead of focusing on a variety of details. Sensor simulation is an important aspect in many envisaged simulation scenarios dealing with robotics. In this paper, we focus on a novel sensor simulation approach of optical sensors supported by the render module. By using data and techniques originally developed for 3D rendering, simulation results can be improved whilst simultaneously increasing the performance significantly, which is essential for real-time simulation tasks. Beside the render-relevant data, our sensor simulation approach does not require any additional data to achieve high accurate results. Our approach is integrated into a modern VR-Simulation-System. Thus, different Virtual Testbed scenarios ranging from the simulation of wood harvesting processes in simulated forests up to the virtual prototyping of space exploration robots, are presented to emphasize the synergy between the sensor and render component.

Key–Words: Sensor simulation, VR simulation systems, computer graphics, robotics

1 Introduction

Testing and evaluation are important tasks in an engineers’ and researchers’ daily work. As testing in the real world often is time-consuming, expensive and, as in the fields of space robotics, sometimes even impossible, VR-Simulation Systems are widely used for these tasks. Up to now, various specialized simulation systems have been developed for even more fields of application. On the one hand, these simulation systems are built and optimized for specific areas of interest to deliver adequate results in their respective fields of expertise. On the other hand, many practical applications require the combined simulation of complex processes. Today, this directly leads to problems combining the simulation systems and thus potentially unsatisfactory results arising already from e.g. the incompatibility of considered physical properties of a system: Does a rigid body simulation interact with a terra-mechanics simulation components by exchanging ‘displacement’, or momentum of force or even power. A general answer to such questions has not yet been found.

Due to many different topics and application areas currently covered by research projects our common main development focus is the continuous development of a structure of a ‘world-simulator’, a 3D simulation and Virtual Reality system which is capable of integrating various physical processes. In a ‘Virtual Testbed’, all simulation components are part of a single framework covering all tasks of the whole simulation scenario. Considering the virtual prototyping of a mobile robot, a realistic sensor simulation is essential to optimize the robot’s components as well as the operating software. Especially the realistic simulation of laser scanners is of high importance to avoid errors resulting from incorrect sensor data returned by the scanner. In reality, laser scanners may not work properly in every situation. Certain material parameters or hardware-related factors lead to unpredictable results. Due to the fact that Virtual Testbeds are built to simulate whole simulation scenarios close to reality, even these factors need to be taken into account to deliver results required by researchers or engineers for system development and optimization reasons. In figure 1 certain Virtual Testbed scenarios including sensor simulation are demonstrated. The continuous development of advanced modeling tools and modern capturing techniques as well as the availableness of advanced graphics features delivered by modern graphics hardware has raised the rendering quality of virtual environments to a level close to photorealism. As simulation results mostly depend on the available input data in general, this data can be used by the render engine as well as the simulation. Moreover, even widely
used graphics effects that only affect complex geometry for performance reasons can be helpful for sensor simulations. As a camera is a sensor in general and many other kinds of sensors usually are just a special kind of a camera, it is obvious to use the renderer to simulate these kinds of sensors.

This paper is organized as follows: Section 2 presents related work in the fields of simulation systems as well as sensor simulation frameworks. Section 3 introduces the general concept of render-supported sensor simulation, followed by the implementation details into an existing 3D VR simulation system in section 4. In section 5 current Virtual Testbeds are presented and the advantages of our approach are emphasized. Finally, section 6 will conclude the results and discuss further usecases for future work.

2 Related Work

To date, many different simulation systems including sensor simulation have been developed. Most of them are designed for industrial robot simulation like [1]. Simulation systems, which are not confined to an industrial work cell, are the ROAMS Simulation Environment [2], the 3DROV simulation and verification tool [3] or combinations of adapted and integrated off-the-shelf software tools like Matlab/Simulink and SIMPACK as described in [4]. The drawback of those system designs is the loss of flexibility or the need for special data or model representations. Whereas the aforementioned simulation systems are exclusively designed for the simulation of planetary rovers, our goal is to provide a comprehensive simulation tool for various mobile robots as well as other simulation scenarios. The sensor simulation component is an important part for this task.

Well known examples for existing simulation systems with sensor integration are Microsoft Robotics Developer Studio [5] or the Player/Stage/Gazebo project [6]. These simulation systems have been enhanced over time and represent sophisticated simulation environments. As Player/Stage/Gazebo integrates various simulation tools, configuration is complex and error-prone. In contrast to the render-supported sensor simulation approach presented in this paper, these simulation systems do not use the GPU to accelerate or improve the simulation results. Simulation systems designed to meet the needs of mobile robotics test environments like EyeSim or USARSim [7] are designed for certain robot platforms and are limited in their functionality. Adapting the implementation to new domains is difficult and time consuming. As error modeling is an important task to acquire close-to-reality results for sensor simulations, several studies were performed addressing measurement errors caused by hardware factors as well as certain material properties of scanned surfaces. As this paper mostly focuses on laser scanner simulations, [8] gives a good introduction into this field.

3 Render-Supported Sensor Simulation

In this section, we will introduce the novel approach to use the render component to directly support the sensor simulation. Focussing on optical sensors, the camera represents the basis of this kind of sensor. As most optical sensors are just a special kind of a camera, it is obvious to use the renderer to simulate these kinds of sensors. On modern, programmable graphics hardware it is possible to map the characteristics of an optical sensor to a camera model by using shaders. As modern sensors - like laser scanners for example - deliver data consisting of several millions of measurement values, it is even more important to move the calculations from the CPU to the GPU to fulfill real-time simulation criteria. GPGPU is widely used for hardware-accelerated computation of complex calculations in many simulation components but it does not rely on any render data of the virtual scene. In contrast to GPGPU, our approach takes place as an additional common render pass which mostly utilize data already loaded to the GPU by the common render plugin for scene rendering purposes. Moreover, data used for advanced graphics effects like normal maps, displacement maps or gloss maps are taken into account to support the sensor simulation and to achieve precise results on a per-pixel basis.
3.1 Simulating a Laser Scanner

As the camera is represented as a common pinhole camera in OpenGL, the scene shader needs to be modified to fit the desired optical sensor camera model. As this paper mostly focuses on laser scanner simulation, we will show how a laser scanner can be modeled and simulated using shaders. A common laser scanner casts laser rays and records the light intensity received by the sensor, which was reflected by the obstacles. This is almost the same like in a common camera model except that the camera does not cast any light. This can easily be simulated by adding a light source to the laser scanner’s position which emits light with the laser’s color. Moreover, this approach allows for the simulation of side-effects resulting from exterior light sources illuminating the scanned objects. By modifying the shadings lighting calculations and providing additional physical information to the light source relevant for laser rays, even more precise lighting calculations are possible.

Most laser scanners are able to scan rows with a field of view of 360 degrees, thus, the render pass must be divided into multiple passes with smaller field of views, i.e. four render passes with 90 degrees. In fact, some approaches exist that are able to render 180 degree field of view like used for point light dual paraboloid shadow maps [9], but they suffer from incorrect polygon transformations as the GPU performs a linear interpolation during rasterization. However, even without the mentioned transformation problems, a field of view diversion could be necessary depending on the desired scan resolution. As the results are rendered into textures and common laser scanners are able to scan in steps smaller than 0.002 degrees, texture size limitations have to be taken into account when calculating the field of views of the render passes.

Depending on the available data, different calculations can be performed in the shader to improve the simulation result. Figure 2 shows screenshots of a laser scanner simulation with varying available datasets. The laser rays as well as their reflection vector are visualized, the length of the reflection vectors represents how much light of the laser ray is diverted by the surface and not reflected back to the laser sensor for depth measurement. The upper left part shows a laser scan on a flat, glossy surface. This model only consists of a single polygon with a single material. As expected, most light is reflected, making it hard to detect the correct depth for the laser scanner. By changing the material parameters, the reflection can be influenced in desired ways, but it is limited to global, uniform values for the whole polygon. This scenario represents the input data many common laser scanner simulations available for general scene descriptions rely on. Especially for the simulation of high-accuracy laser scanners delivering a high scanning resolution, this approach delivers inadequate results. On the upper right part of the figure, textures available by the render component are taken into account. In contrast to global material descriptions, textures provide information on a per-pixel basis, ideal for high-resolution scans of even less detailed scene descriptions that consist of less polygons. The image illustrates how the reflection values change depending on the surface’s color. A single color texture as used in common render scenes is able to deliver enough data for error measurements depending on the surface properties as described by [8]. One major outcome of that paper is the fact, that laser scanners deliver inadequate results for black surfaces, illustrated by the short vectors at the wall’s gaps. On the lower left part of figure 2 a normal map is used to alter the reflection vector of the laser rays. Moreover, by taking displacement maps and gloss maps into account the simulation is able to deliver close-to-reality results. To archive close-to-reality results, different
error models appearing for real-world laser scanners need to be taken into account. Most of these error models can easily be added and calculated directly on GPU-side by the shader. A detailed description of the implementation of our approach including the integration of error models is introduced in the following section.

4 Implementation

The sensor simulation components are integrated into a modern 3D VR simulation system currently used for research projects at our institute. It is able to handle widespread simulation tasks under one single simulation framework based on an advanced object-oriented database framework. This database includes all model data needed for the simulation, the complete simulation state is described by nodes, extensions and properties, which delivers an ideal basis for the interaction between simulation and render components and allow for synergy effects. Up to now, many simulation modules - including rigid body dynamics, terramechanics, etc. - have already been implemented and integrated into the modular simulation framework extending its range of application. The sensor simulation functionalities are implemented in two system components. The first component is the sensor simulation plugin that defines the general sensor simulation structure. The second component is the sensor renderer which uses the presented render-supported sensor simulation approach and performs the render tasks.

4.1 The Sensor Component

The newly developed sensor framework allows the parallel integration of real and simulated sensors into the system and provides a smooth transition between simulation and real world setups. The sensor framework design allows for easy setups of Virtual Testbeds using a generic communication concept for the interaction of all components offering a focussed view on every component of the system to analyze and optimize its behavior. In a first step the communication environment has been standardzed to an input-output handler concept that is based on an IO-board metaphor. An abstract base class is provided for all types of data. This provides connectivity to different system plugins like the sensor plugin and the render plugin used to support sensor simulation tasks. Moreover, all data required for the simulation task can be stored in the common simulation system’s database. Figure 3 shows an UML diagram as the conceptual view of the sensor framework, emphasizing the three major levels of inheritance. The first layer represents the aforementioned communication components. Abstract implementations of sensors, error models, data logging components as well as the visualization component inherit from the first layer and constitute the second layer of the sensor framework. Basic inputs and outputs required for all inherited components are placed in this layer. The third layer inherits the abstract realization and efficiently realizes the real, virtual and simulated sensors. Additionally, it specifies error models especially designed to represent an error scenario of a sensor or specialized sensor data visualization for different use cases. Furthermore, the third layer differentiates among three kinds of sensors. They are classified as an implementation of real hardware API, simulated sensors as well as virtual sensors, necessary to integrate algorithmic results and recorded sensor data. APIs provided by real sensors are in use to connect and control real hardware components. Thus, parameters can be altered at run-time, which is important for certain algorithms that require different resolutions of sensor data to calculate correct results. Simulated sensors represent the second class. To carry out necessary test series, simulated sensors can be used in an appropriate Virtual Testbed. Due to the fact that simulated sensors are implemented to deliver or work on ideal data, specific error models need to be applied to grant close-to-reality simulations required for Virtual Testbed scenarios. Virtual sensors represent the third class of sensors in our framework. They are used to induct algorithmic results or recorded data into the network of connected components.

4.1.1 Error Modeling

Comparing the delivered data of real hardware sensors to simulated ones shows large deviations. Thus, besides the ability to simulate ideal sensors it is required to add error models for Virtual Testbeds to
archive realistic simulation results. In reality the output of a sensor is based upon several criteria. Focusing on the simulation of a realistic laser scanner, this includes maximum scanning distances, certain material properties of scanned surfaces as well as typical, hardware-related error characteristics like random noise. The discrepancy between the output of the real hardware components and the simulated ones is flattened by adapting characteristics of real hardware components like biased and statistical error and connecting these components in an adequate way. The simulation system in use provides a graphical user interface to compose a sensor network based on input and output data ports. In addition standard error models for depth noise, dirt or reflection characteristics, context-dependent errors, as caused by certain detected colors for example, have been implemented. These error models are based on measurement results as presented in [8]. Depending on the chosen error models, the scene complexity as well as the sensor scan resolution, the simulation of a laser scanner including all these factors leads to a huge calculation load. As a scan process of a common laser scanner can consists of several million single ray casts, these calculations can hardly be handled in real-time with common CPU-based simulation approaches.

4.2 Using the Rendere Component

A second plugin was implemented that is based on the render-supported sensor simulation approach presented in section 3. This plugin handles the sensor simulation’s render pass using existing render-related data provided by the simulation system’s core render plugin. Moreover, the sensor renderer loads optional material parameters and textures containing additional information - like physical properties that describe the surface material in more detail - to improve the accuracy of the results. They can easily be attached to the simulation system’s scene graph in real-time using it’s GUI. Figure 4 illustrates the dataflow between the sensor plugin and the sensor renderer. The sensor plugin sends a render request to the sensor renderer. This request contains the desired sensor model, the desired output data and the error models that should be applied. The sensor renderer chooses corresponding shader code chunks from a library of sensor and error models and compiles the final scene shader. Depending on the desired calculations - like reflection vectors, surface colors, reflection intensities, etc. - textures are generated and bound as render target. The sensor renderer activates the sensor calculation shader and calls draw routines provided by the 3D data manager of the simulation systems core render plugin. After rendering, the results are read from the textures and send back to the sensor simulation plugin.

5 Applications and Results

In this section results of certain application areas of Virtual Testbeds are presented using the sensor simulation approach introduced in this paper. The first application scenario is a Virtual Testbed used in the ‘Virtual Crater’ research project [10]. The aim of this project is to find proper methods and parameters for physically based simulations of walking robots for extraterrestrial exploration tasks. This shall be achieved by comparing the results of reference experiments implemented both in reality and in Virtual Reality. In contrast to well-tested sensor environments on earth, measurement errors can occur arising from special circumstances observed in space environments. The render-supported sensor simulation approach delivers scanner results that consider certain error models like sensor noise resulting from fluctuation of temperature or measurement errors effected by adverse material characteristics. In contrast to common laser scanner simulations, our approach delivers accurate results due to the usage of additional data like optional textures containing physical material properties on a per-pixel basis. Furthermore, render-specific effects that simulate complex geometry like normal-, displacement- and gloss-mapping are used to further enhance the results. In contrast to common laser scanner simulations, our approach delivers accurate results due to the usage of additional data like optional textures containing physical material properties on a per-pixel basis.
pass without LOD. To be able to compare this result to the sensor render pass, the same field of view was chosen for the laser scanner and the sensor resolution was set to 768x768, which corresponds to 589,824 single measurement points. This is quite more than a modern 3D laser scanner can measure per second (less than 500,000 points per second). The sensor render pass including the data read back from the result texture (3-component float) takes approximately 14ms. The difference can be explained by the workload necessary for the data readback, which costs most of the measured time. The pure sensor render time is comparable to the normal render time. Depending on the applied error models and desired sensor calculations, the pure render time can show little variations, but still archives great performance.

The second presented Virtual Testbed is the 'ProDemo' Testbed [11]. The main objective of this project is the development of new methods for the programming of industrial robots. By using a laser scanner system, spatial information is easily acquired and used for modeling of the work cell. The system approach aims at reducing the effort and the cost of the setup of robotic systems in order to support the cost-effective deployment of automation technology in small and medium enterprises (SME). As the design of a physical test bed is very time-consuming and not very flexible in terms of different work cell configurations, the setup of virtual testbeds allows testing the implemented exploration approaches in a greater variety of industrial surroundings effectively. Moreover, sensor measurement errors resulting from materials inappropriate for laser based sensors like glossy metallic materials can be taken into account to optimize the software.

Finally, the third presented application is the 'Virtual Forest' Testbed. An application in use is the localization and navigation system called VisualGPS [12]. This approach has been implemented for the forest and deals with sensor technologies in the forest. Figure 6 shows screenshots of the Virtual Forest Testbed. GPS based localization and navigation in the forest suffers from low position accuracy and, in worst case scenarios, even signal loss. This leads to wrong position estimations or total lack of knowledge of the current position. As position awareness is of high importance in the forestry sector of central Europe for managing and accounting reasons, a sensor fusion based approach has been implemented determining the position of a vehicle in combination with a landmark map delineated from aerial survey data. The sensor simulation in use combined with the simulated error models like weather influences improved the robustness of the implemented self-localization algorithms. The self-localization module is now in use on a real forest work machine and allows robust and highly accurate self localization.

6 Conclusion and Future Work

In this paper, a novel render-supported approach for optical sensor simulation was presented. It is based on the data provided for rendering purposes and capable of delivering more precise results than common CPU-based sensor simulations. This is archived by

---

1The Virtual Forest is supported by the State of North Rhine Westphalia (NRW), Germany, the forest administration of North Rhine Westphalia and the European Union (Europaischer Fond für regionale Entwicklung - EFRE)
adding render-specific calculations - like normal, displacement or gloss-mapping - typically used to simulate more complex scene geometry and improve its visual appearance for simulation reasons. In contrast to common laser scanner simulations which deliver results on a per-vertex accuracy, our render-supported approach allows for per-pixel accuracy. Thus, good simulation results can also be archived for low-polygon models. Moreover, additional data like textures containing optional physical properties that allow for even more precise calculations can easily be attached through the simulation system.

A major advantage of the presented approach is the fact that it is possible to use the render component as well as well-known rendering techniques to significantly improve sensor simulation components. Thus, the render component is no longer solely being used for rendering purposes. The presented Virtual Testbeds illustrated, that a close-to-reality simulation of optical sensors is essential for many tasks. We showed that our approach delivers high-precision results close-to-reality. Finally, by doing most sensor simulation calculations on the GPU-side with a single additional render pass, fast simulation cycles can be granted, which is a major criteria for on-the-fly development in Virtual Testbeds. The realistic simulation of various camera models including lens refraction, distortion and CCD-dependent properties will be one of the next fields of interest to be tackled. With these new features, the rendered scene can also be used as a basis for image-based simulation modules which handle stills or videos of real-world cameras. This capability becomes very important in the fields of space robotics and mission plannings, where relatively little real image or video data has been captured so far.

Due to the flexible and modular structure of the sensor framework as well as our general render-supported simulation approach, these features can easily be integrated, further improving the accuracy and performance of new as well as existing simulation components.

References:


