A Simulation Framework for Time-Of-Flight Sensors

Maik Keller*, Jens Orthmann* Andreas Kolb* and Valerij Peters†
*Computer Graphics Group, Institute for Vision and Graphics, University of Siegen, Germany
Email: maik.keller@uni-siegen.de
†Center for Sensorsystems (ZESS), University of Siegen, Germany
Email: peters@zess.uni-siegen.de

***Abstract*— Modern *Time-Of-Flight (TOF)* cameras enable the measurement of full-range distance information in real time. The distance information can be calculated by estimating the elapsed time between the emission and the receiving of active light. The simulation of sensors is an essential building block for hardware design and application development. Thus, expensive prototypes can be omitted and down-stream algorithms, e.g. for sensor calibration and sensor data processing, can be tested.**

This paper deals with the simulation of camera-like time-of-flight sensors and focuses particularly on the manipulation of camera parameters and the generation of synthetic sensor data in real-time. The *sensor framework* provides tools for object and camera manipulation as well as animation. All relevant parameters for the camera-like sensor, object appearance and per-pixel sensor information are available to the *TOF-Sensor Simulation Unit (SSU)*, which realizes the specific sensor characteristics.

Moreover, we present a first concept of the simulation of *Photonic Mixing Device (PMD)* sensors. The framework is hardware accelerated to allow for interactive simulator feedback, making use of the programmability of modern *Graphics Processing Units (GPUs)*, i.e. of vertex and fragment programs.

I. Introduction

Various camera-like techniques for measuring distances between objects and a sensor location exist in literature. Prominent examples based on triangulation and/or image correspondences are laser triangulation, structured light and stereo vision [1], [2]. *Time-of-flight (TOF)* sensors, on the other hand, compute the distance to an object by estimating the elapsed time between the emission of light from an active illumination to the arrival at the pixel of a sensor chip. Relatively new camera-like approaches, like the *Photonic Mixing Device (PMD)* [3], [4] and the *Z-Cam* [5], are still developing and only a few prototypes are available.

A simulation program which is able to reproduce the essential sensor characteristics is very helpful in a variety of use cases. Apart from the easy modification of sensor parameters - resolution, focal-length, frame-rate and radial distortion - the main aspect in simulation is the possibility of carrying out experiments under reproducible conditions, especially for dynamic scene setups. Therefore, the simulation results must reflect major sensor characteristics in order to produce results representative of and comparable to real sensor data. With the help of this kind of simulator, the development of the TOF-sensors themselves and of down-stream algorithms, e.g. in the fields of object recognition and image analysis, can be significantly enhanced.

In this paper we present a framework for the simulation of camera-like TOF distance sensors. The TOF-simulator framework provides manipulation techniques for objects and cameras. Using standard graphics rasterization techniques and programmable graphics hardware, all relevant global parameters, e.g. intrinsic camera parameters as well as per-pixel information, are available to the *Sensor Simulation Unit (SSU)*. This unit is the core simulation process, resembling the sensor's hardware characteristics as well as possible sensor-internal data pre-processing. We focus on the real-time simulation of TOF-sensors, thus the SSU is implemented as freely configurable fragment programs on the *Graphics Processing Units (GPUs)*.

The structure of this paper is as follows: in Sec. II, we review related work on TOF-simulation applications. In Sec. III we introduce the simulator's architecture and its generic interface as well as the application's features. Sec. IV describes the API for the SSU. A first concept to integrate PMD-based TOF-sensors is presented in Sec. V. Sec. VII concludes the presented work and comments on further development.

II. Related Work

Today's developers often test their algorithms on both real camera data and on simulated image sequences. Streckel et al. [6] set up a simple simulation procedure which generates synthetic depth images to test enhancements of their *structure from motion*-algorithm using PMD sensors. They use results presented by Kuhnert and Stommel [7] to simulate depth noise for synthetic PMD data. This approach assumes a quadratic relation between sensor-object distance and the standard deviation of the distance information provided by PMD-sensors.

Peters et al. [8] use synthetic test data for the localization of mobile robots. They use a simulator application which is especially developed for the simulation of PMD sensors. The approach they favor is MATLAB-based and not suitable for real-time simulation. The principle of their simulated PMD system is based on the measurement of the time of flight of the transmitted signal, which leads to the distance to the object. The simulated theoretical response of the whole 3D scene, which is a point cloud, is represented by an overlapping of

single point responses. The possibility to position and orientate the illuminator and the sensor separately allows simulating of real bistatic and multibistatic (in case of multi-illuminator systems) configurations.

III. Simulation Framework

The simulation framework is in particular designed for the generation of synthetic sensor data which usually serve as input data for further data processing and algorithm development. Our ideas of facilitating the generation of such synthetic data are based on the reuse of a framework which is able to deal with the simulation of different full-range camera-like TOF-sensors. Therefore, the data-manipulation and sensor-simulation parts of the application are very generic, so that a new sensor can easily be incorporated into the framework.

A sketch of the framework's architecture is shown in Fig. 1. All relevant scene-, object- and camera-parameters (e.g. intrinsic sensor parameters) are fully editable by the user and accessible to the scene manager and to the *Sensor Simulation Unit (SSU)*. Additionally, the framework provides the possibility to add user-defined parameters which are essential for integrating a sensor's characteristic. Next, the scene is processed using standard rasterization techniques and programmable graphics hardware which ends up in data containing "per pixel information", i.e. ideal depth values, material information, object normal and reflectivity for the corresponding object point. All data is written to the GPU memory and thus also available to the SSU for further processing (see Sec. IV).

A. Configuration

The scene with all its objects and definitions is loaded into the application using VRML compatible data [9], [10]. Optical camera parameters like exposure time, focal length and image resolution as well as sensor specific parameters such as the modulation frequency are accessible and editable in the framework's GUI. Additionally, the configuration of object and camera motion during an animation can be specified. For this purpose, the built-in animation system provides an easy way to define object and camera movements for dynamic scenes. A keyframe list is assigned to every object with each keyframe referring to an object's orientation and position of a certain moment in time ("key").

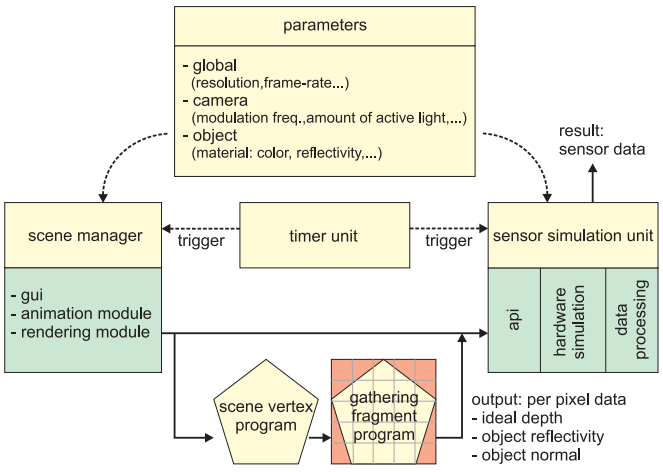

Fig. 1. The architecture of the simulation framework. All parameters are freely configurable by the user and accessible by the scene manager and the SSU. A timer triggers the rendering process (see Fig. 3). The SSU also has full access to the output of the gathering fragment program which calculates several data per pixel.

B. Simulation

The sensor simulation is based on all previously configured parameters. The usage of the GPU's rendering pipeline offers the possibility of data acquisition in real-time. During an animation, each frame can be selected and viewed separately by the user, which simplifies the evaluation process. The values between two keyframes can be interpolated using either linear or cubic interpolation schemes like Catmull-Rom splines. The interpolation of orientations is based upon quaternions in order to safeguard against the ambiguity of interpolating between Euler angles. The result of the simulation is a sequence of synthetic sensor data.

C. Advanced Features

The approach of using standard rasterization techniques is limited to the simulation of *one* "ray" per pixel at *one* moment in time. For more detailed information about the standard rendering pipeline in computer graphics see [11]. In order to compensate for this disadvantage the framework supplies several features.

- The temporal integration provides a super-sampling based on the time-axis.
- The spatial super-sampling allows the sampling of multiple rays per pixel.

Thus, advanced sensor effects like motion blurring and depth of field are also feasible in the framework and can be implemented in the SSU.

IV. Sensor Simulation Unit (SSU)

The Sensor Simulation Unit resembles the characteristics of the sensor's hardware and offers the ability to simulate the sensor-internal data pre-processing. Fig. 2 displays the architecture of the SSU. The real-time emulation of TOF-sensors is achieved by implementing the SSU as freely configurable fragment programs on the GPU. The image data is directly written to the GPU memory with the help of framebuffer objects. This enables the advantages of modern graphics hardware which is fast, flexible and highly configurable [12]. Its parallel stream processors offer a way to outsource a considerable part of the sensor's logic to the graphics card and thus a sensor-pixel can be processed very fast. On the one hand, as already described in Sec. III, the SSU has full access to all relevant parameters (e.g. intrinsic sensor parameters), on the other hand, the per pixel information is also available to the SSU at the end of the rasterization stage.

The implementation of the SSU is useful for more elaborate techniques, for example,

- the temporal and spatial super-sampling (see III-C),

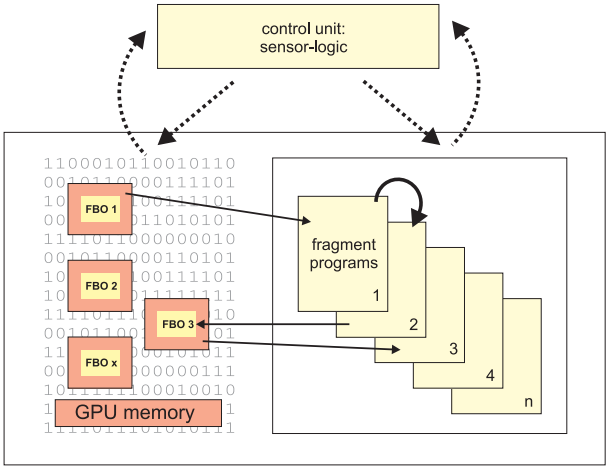

Fig. 2. The SSU architecture. This part of the framework is freely configurable and programmable by the user. The Sensor Simulation Unit is implemented as fragment programs on the GPU which emulate the hardware of the sensor. In combination with the control unit innumerous configurations and procedures are possible.

- the conversion of "per pixel data" to intermediate physical data, as it is produced by the real sensor and controlled by the global sensor parameters,
- and multiple render passes are feasible and may be necessary in the case of more complex networks of fragment programs.

V. Concept: Simulation of PMD-data

In this section we propose the use of the simulation framework for a simulation of TOF-sensors based on the PMD technology. The implementation of the sensor simulation is shown in principle. The fine-tuning and adjustment of the parameters is still in progress and under current development. First, a short introduction to the PMD technology is given followed by the implementation concept.

A. Photo Mixing Detector (PMD)

The PMD camera illuminates the scene with modulated, incoherent NIR-light. Smart pixel sensors [4], [3] gather the reflected light. One pixel samples and correlates the incoming optical signal with the reference signal of the modulated illumination. Thus, the PMD is able to determine the signal's phase shift, and the distance to the according object region can be calculated.

This process can be expressed by the following equations: given a reference signal $g(t)$ and the optical signal $s(t)$ incident to a PMD pixel, the pixel samples the correlation function $c(\tau)$ for a given internal phase delay τ:

$$c(\tau) = s \otimes g = \lim_{T\to\infty} \int_{-T/2}^{T/2} s(t) \cdot g(t+\tau)\, \mathrm{d}t. \qquad (1)$$

In detail, the demodulation of the correlation function is done using several samples of $c(\tau)$ obtained by four sequential PMD raw images $A_i = c(\tau_i)$ which use internal phase delays $\tau_i = i \cdot \frac{\pi}{2}$, $i \in [0, \ldots, 3]$.

Assuming the reference signal as sinusoidal, i.e. $g(t) = \cos(\omega t)$ and not another non-linear signal deformation, the optical response signal is given by $s(t) = k + a\cos(\omega t + \phi)$. Basic trigonometric calculations yields

$$c(\tau) = k + \tfrac{a}{2}\cos(\omega\tau + \phi), \qquad (2)$$

where ω is the modulation frequency, a is the amplitude, k is the correlation function's bias, and ϕ is the phase shift relating to the object distance. Please note that the modulation frequency defines the distance unambiguousness for the distance sensing, e.g. $\omega = 20\,\mathrm{MHz}$ results in an unambiguous distance range of 7.5 m. The final distance d to the object's surface can be calculated based on the four raw images:

$$\phi = \arctan\left(\frac{A_3 - A_1}{A_0 - A_2}\right),\ d = \frac{c}{4\pi\omega}\phi, \qquad (3)$$

where $c \approx 3 \cdot 10^8 \frac{m}{s}$ is the speed of light. For more detailed information about the physical background of the PMD technology see Lange [4].

B. Concept of Implementation

In the following we outline a configuration of the simulation framework which is able to simulate the characteristics of a PMD sensor. The goal is the retrieval of the four phase images (raw images) as well as the final depth image. Therefore, we reverse the data generation process of the PMD-sensor.

1) Theoretical Reflection: According to the phase delay τ, the scene is rendered four times. Please note that during an animation sequence these four images may differ from each other due to camera or object motion. Using the simulator's per-pixel ideal depth information, the corresponding phase shift $\phi = \frac{4\pi\omega}{c}d$ can be reconstructed (see Eq. (3)). The corresponding values of the correlation function, i.e. $A_i = \tau(i \cdot \frac{\pi}{2})$, can be determined, if the concrete form of the function c is known. The final depth value is restored by compositing the four phase images as it is the case for real sensor data.

2) Framework Adaption: As described in Sec. III the framework handles common parameters, e.g. the chip-resolution and the radial distortion, automatically. The user has to specify the object NIR-reflectivity to cope with the PMD-sensor's range of active light wavelengths. Different values can be assigned to different objects during the framework's configuration mode. The gathering fragment program then calculates the reflectivity per PMD-pixel, because the sensor's image processing works on pixel-resolution (see Sec. V-A). The timer unit will trigger the framework to render the scene four times into different framebuffer-objects A_i, $i \in [0, \ldots, 3]$, provided by the memory management unit. The SSU will afterwards be triggered for further data processing.

The relevant framebuffer-objects are now available to the SSU via the framework API. In a first pass, each of the images will be processed separately to compute A_i. Therefore, an SSU fragment program computes the phase images based on the content of the existing framebuffer-objects and writes the result to four new framebuffer-objects. A second pass is implemented as an additional fragment program which does the compositing

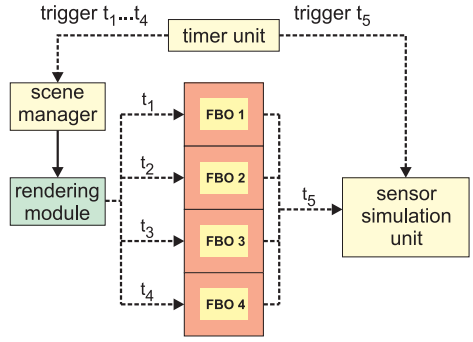

Fig. 3. A simple example of a timer configuration. The timer is able to trigger the scene manager and the SSU unit asynchronously. Here, at first the scene will be rendered four times into different framebuffer objects, finally the SSU is triggered by the timer and starts data processing.

of the phase images A_i. The output will be a depth image according to the equations in Sec. V-A.

VI. Results

Even though we have not fully realized SSU for the PMD-sensor yet, we have set up a simple test sensor scheme in order to verify various base functionalities of the simulator framework. The test SSU simply uses the distance information determined during the rasterization stage in the framework to compute an average distance value.

Here, the basic super-sampling functionalities for both the temporal and the spatial domain are used. As a final result, four subsequent distance images are stored in the color RGBA-components and a final average distance image is generated (see Fig. 4). The result already resembles closely the visual result of a PMD sensor.

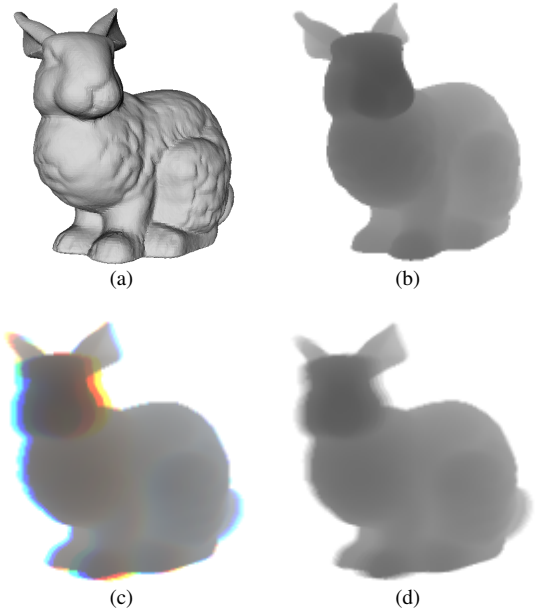

Fig. 4. Simulation results. 4(a) shows the test object. 4(b) displays the ideal depth values as gray-level image. The images in the bottom row are calculated by the SSU for a moving camera. 4(c) represents the depth of four raw images, each coded in one RGBA component. Fig. 4(d) shows the averaged depth images.

VII. Conclusion and Future Work

We have presented a framework to set up simulators for camera-like time-of-flight sensors, i.e. which incorporate an optical lens system and pixel-matrices in order to acquire full-range depth information. The framework's architecture includes fixed subsystems for the user-control of the simulation process, especially for the animation of camera and scene objects and for the sensor parameters. A Sensor Simulation Unit realizes the special TOF-sensor's physical behavior.

Up to now, we have only integrated a simple SSU, which serves as proof of the concept. The concept of the integration on PMD-based distance sensors is at hand and we will integrate this sensor into our framework in the near future. We are very confident to be able to present a fully equipped PMD SSU at the ISSCS workshop in July.

Acknowledgment

This work is partially funded by grant V3DDS001 from the German Federal Ministry of Education and Research (BMB+F).

References

[1] O. Faugeras, *Three-dimensional Computer Vision*. The MIT Press, 1993.
[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Visio*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.
[3] Z. Xu, R. Schwarte, H. Heinol, B. Buxbaum, and T. Ringbeck, "Smart pixel – photonic mixer device (PMD)," in *Proc. Int. Conf. on Mechatron. & Machine Vision*, 1998, pp. 259–264.
[4] R. Lange, "3D time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology," Ph.D. dissertation, University of Siegen, 2000.
[5] 3DV Systems, 2007, http://www.3dvsystems.com.
[6] B. Streckel, B. Bartczak, R. Koch, and A. Kolb, "Supporting structure from motion with a 3D-range-camera," in *Scandinavian Conf. Image Analysis (SCIA)*, 2007.
[7] K.-D. Kuhnert and M. Stommel, "Fusion of stereo-camera and pmd-camera data for real-time suited precise 3D environment reconstruction," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 4780–4785.
[8] V. Peters, F. Hasouneh, S. Knedlik, and O. Loffeld, "Simulation of PMD based self-localization of mobile sensor nodes or robots," in *Symp. on Simulation Technique (ASIM))*, 2006.
[9] Web-3D-Consortium, "VRML specification," http://www.web3d.org/x3d/specifications/vrml/, 1997.
[10] Autodesk, "Maya and 3D Studio Max," http://www.alias.com/, 2006.
[11] T. Akenine-Moller and E. Haines, *Real-Time Rendering*. Natick, MA, USA: A. K. Peters, Ltd., 2002.
[12] J. Owens, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005, ch. Streaming Architectures and Technology Trends, pp. 457–470.