PHYSICALLY BASED SONAR SIMULATION AND IMAGE GENERATION FOR SIDE-SCAN SONAR

Philipp Woock[a], Jürgen Beyerer[a]

[a]Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB,

Philipp Woock, Fraunhofer IOSB, Fraunhoferstrasse 1, 76131 Karlsruhe,
philipp.woock@iosb.fraunhofer.de

Abstract: *The paper describes a method to generate side-scan sonar images from any three-dimensional geometry and vehicle path. These generated images can be used for evaluation of reconstruction algorithms or for training purposes (be it human or for deep neural networks).*
To achieve this, we use the open-source modeling tool Blender in conjunction with the Mitsuba renderer and a custom sonar signal processing stage.
The basic workflow is to model any geometry and vehicle path in Blender. The geometry information is then exported to a scene description format, which the Mitsuba renderer is able to read. From there, we let Mitsuba perform raytracing on the scene and obtain various scene maps: per-pixel distances to the surface elements in the scene and per-pixel information about the normal vector of the surface element.
Given that information, a physically based sonar simulation is done by modeling the travel of individual sound packets to the surface elements. This simulation can be done in a multitude of ways: For chirped sonars or continuous-frequency sonars and it can be done coherently or incoherently. An available sonar beam pattern is also taken into account in the simulation to correctly model the narrow beam of a side-scan sonar. Due to the generic modelling, the simulation is not restricted to a single device and realistic measurements can be simulated as the generated images exhibit the typical speckle noise characteristic which is a problem in many applications.

Keywords: *Side-scan, sonar, simulation, geometry, speckle, trajectory, FM, CW*

1. MOTIVATION

Obtaining ground truth in sonar is difficult for many reasons, the main issues being the huge efforts to create and place exactly known sonar targets with large dimensions and to have exact repeatability in terms of vehicle trajectory. Therefore, a simulation environment is necessary to provide means to evaluate different configurations and settings. Furthermore, a simulation tool allows to train humans or algorithms with the possibility to generate lots of training images from a scene. The need for a flexible side-scan sonar simulation as a forward model became evident for the task of reconstruction [1] [2].

There has been quite some work in sonar simulation: Riordan et al. [3] show a sonar simulation in slices of a dynamically refined triangular mesh. More real-time oriented is the simulation of Pailhas [4], while the simulation of Groen [5] is more targeted to SAS applications. A very sophisticated simulation using tube paths instead of simple rays was shown by Gueriot et al. [6]. For our application a raytracing solution seemed to be the best tradeoff between complexity and realism, which is why the simulation ideas are partly based on the work of Bell et al. [7] [8].

This paper details a method how any geometry in Blender can be used as a basis for sonar simulations.

2. WORKFLOW

2.1. Outline

The process to create the simulated sonar images is a multi-step process and is outlined as follows. Details to each step are given in the following sections:

1. Create 3D ground truth scene geometry in Blender
2. Set up camera (sonar) parameters and vehicle trajectory as camera path
3. Render the animation
4. Obtain resulting scene description files (XML) and model (serialized) files
5. Batch process scene files to apply sonar rendering settings to all files of animation
6. Set up sonar simulator settings
7. Process sonar files
8. Stack resulting side-scan sonar lines to obtain the classic side-scan data view

Fig. 1: Blender UI with scene of a tilted shipwreck of a patrol boat on the seafloor.

2.2. Preparations

We use the Mitsuba raytracing renderer to extract all the necessary geometry information from the scene. Therefore, the Mitsuba Blender Plugin[1] is a prerequisite for the subsequent steps. It is responsible for the creation of the scene files, which are later used for the sonar signal creation.

2.3. Scene geometry

The scene setup basically consists of any geometric elements that are allowed to be imported into Blender. The scene in Fig. 1 and Fig. 2 has mainly two parts: a seafloor using Blender's perlin noise surface generator and a freely available 3D ship model[2]. The ship model consists of approximately 41000 vertices and 80000 triangular surfaces. The seafloor consists of about 1,000,000 vertices and 2,000,000 edges and triangles.

In the shown scene in Fig. 1 the ship length is 56 meters, the width is 10.5 meters while the seafloor is 203 meters long and rather flat (height variation 0.1 meters between lowest and highest point).

2.4. Camera and trajectory

The simulation uses the blender concept of a camera to capture all the items the sonar "sees" in the scene. The camera setup defines the field of view the sonar can maximally ensonify, whereas the exact application of a sonar beam pattern happens later in the process. It is advisable to use not too large viewing angles as the sonar contributions from near to the outer edges are diminishing for a sharply focused side-scan beam, while it greatly increases processing time. The main lobe and the most important side lobes should be inside the field of view (see Fig. 2).

The camera path in Blender describes the vehicle trajectory and any motion that can be applied to a camera in Blender can be used for the simulation.

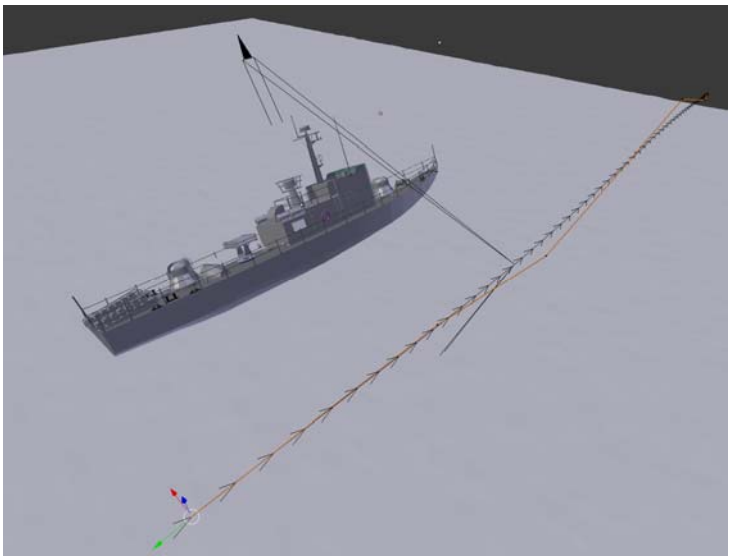

Fig. 2: Detail view of modelled path with the narrow viewing "wedge" the sonar ensonifies. In this case the vehicle trajectory is modelled as a spline curve.

[1] https://blenderartists.org/forum/showthread.php?270335-Addon-Mitsuba-Blender-Plugin-(26-Nov-15)
[2] Find the model e.g. via search on Yobi3D: https://www.yobi3d.com/q/3d-models/pboat?page=1

The rendering step creates scene files for the Mitsuba renderer and executes them. The visual results obtained are only a by-product, the serialized geometry files and XML scene files are the important part for the sonar signal generation.

2.5. Prepare for sonar calculation

The created scene files are batch-processed in order to change the renderer from parameters that are only useful for generation of *visually* appealing files to parameters useful in sonar processing. The XML file entries that are changed in this step are:

Firstly, the sample count per pixel is reduced to one. In contrast to optical imaging where anti-aliasing techniques are used to make visually nice-looking smooth edges, we need geometry information from exactly one single object that is hit by a ray. It is detrimental to have averaged distance (and surface normal) information on the edge pixels, because that leads to "ghost" sonar echoes from non-existant objects.

Secondly, for sonar processing we want to use Mitsuba's multichannel integrator instead of a "visual" integrator to obtain distance and normal maps of the scene, i.e., this results in a "sonar camera" that returns geometric information like distance to surface and surface normal for each pixel.

3. SONAR SIMULATION

The sonar simulation itself is subdivided into three parts that are detailed in the following sections in this chapter.

3.1. Beam pattern

We model the side-scan sonar as an ideal rectangular membrane, i.e., the angular directivity $\Gamma_p(\varphi_a, \varphi_e)$ is given as:

$$\Gamma_p(\varphi_a,\varphi_e) := \frac{\sin(\frac{kl_x}{2}\cos\varphi_a)}{\frac{kl_x}{2}\cos\varphi_a}\frac{\sin(\frac{kl_y}{2}\cos\varphi_e)}{\frac{kl_y}{2}\cos\varphi_e} = \mathrm{si}\left(\frac{kl_x}{2}\cos\varphi_a\right)\mathrm{si}\left(\frac{kl_y}{2}\cos\varphi_e\right) \quad \text{with} \quad \mathrm{si}(x) := \frac{\sin(x)}{x}$$

This is calculated at each pixel position of our "sonar camera", and use that to weight the incoming signal from that pixel.

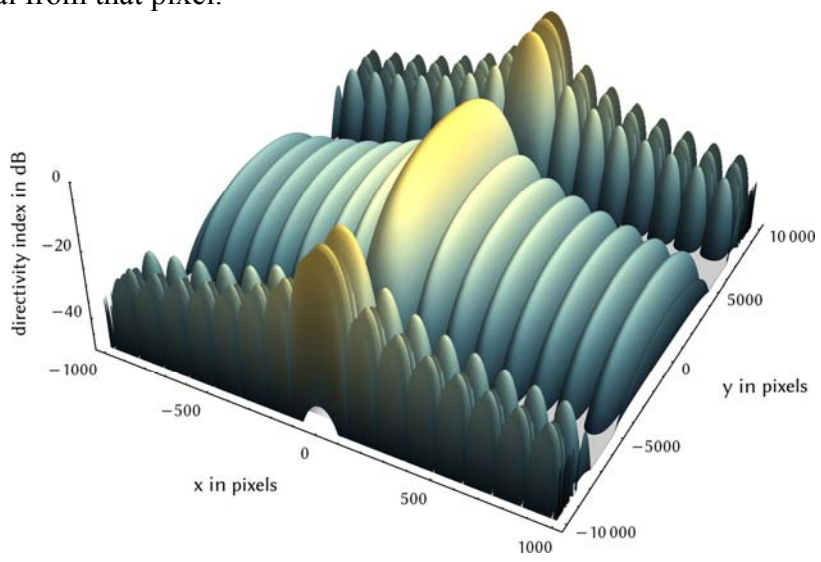

Fig. 3: Directivity pattern of rectangular membrane mapped to image pixel coordinates

The simulation default parameters model a side-scan sonar with opening angles of 1° (azimuth) by 50° (elevation) at 400 kHz, which should be fairly representative of a typical mid-to-high frequency side-scan sonar. This equals transducer membrane sizes of about 190 mm by 3.8 mm. The resulting pattern is shown in Fig. 3.

3.2. Source signal generation

It is assumed that the generated pulses do not suffer from compression effects on the conversion from electrical power to mechanical power. That is, for the simulation of constant frequency (continuous wave, CW) we use sinusoidal waves. All the waves are windowed by a flat-top Blackman-Harris window that is a good compromise between maximizing transmitted energy and side lobe suppression.

We do not exploit the fact that the signal is band-limited and could do with sub-nyquist sampling as it is done, e.g. in [7]. The reason is that we also want to allow the simulation of chirped sonars (frequency modulated, FM). While they typically do not use the full bandwidth, we do not want to restrict the simulation abilities. This should however be considered as an area for potential future speedup.

We use the following waveform $x(t)$ for linearly chirped FM sonars with base frequency f_0 and rising κ, depending on the bandwidth B_s:

$$f(t) = f_0 + \kappa t$$

$$\kappa = \frac{B_s}{\tau}$$

$$x(t) = \sin(2\pi \int_0^t (f_0 + \kappa s) ds\ ) = \sin(2\pi (f_0 t + \frac{\kappa}{2} t^2))$$

Additionally, we offer an exponential chirp where the instantaneous frequency is given as $f(t) = f_0 \kappa^t$:

$$x(t) = \sin(2\pi f_0 \int_0^t (\kappa^s) ds\ ) = \sin(2\pi f_0 \frac{\kappa^t - 1}{\ln \kappa})$$

In order to see the influence of the speckle noise, the simulation can be done in the actual carrier frequency or in the baseband.

For the purpose of creating sonar images for a shape estimation method [1] [2] the simulation can also output speckle-free images by using only windowed rectangular pulses.

3.3. Time domain signal processing

Signal processing is performed in the time domain by combining all the contributions from each ray. The simulation offers to do incoherent processing or coherent processing for the CW pulses by using the analytic signal. The signal is then lowpass-FIR-filtered. The FM simulation performs pulse compression for the chirp signals.

One can also choose whether the simulation should also employ a perfect time-varying gain or not.

4. RESULTS

4.1. Test scene

A simplistic anchor mine is represented by three balls and a cube. The sonar passes by the mine and its main lobe is directed roughly to the mooring of the mine. This test scene is depicted in Fig. 4.

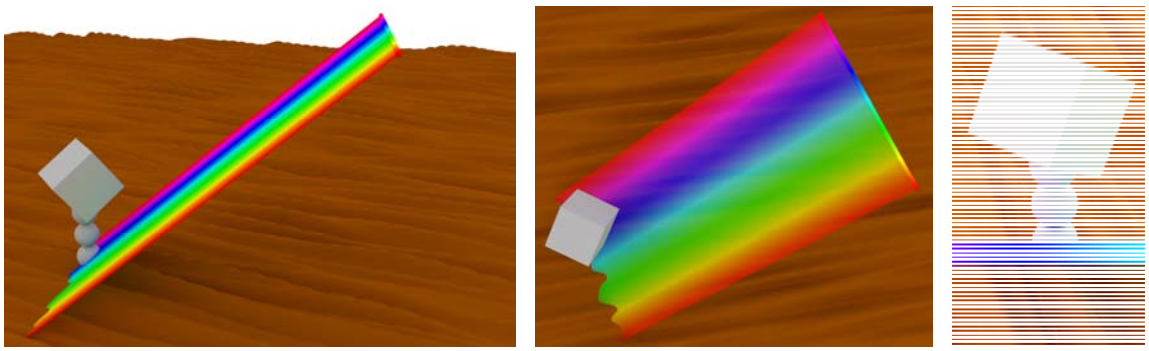

Fig. 4: Scenario for comparison of wave simulation modes in oblique view (left) and top view (middle). View perpendicular to the vehicle motion path (right), i.e., the sonar "look" direction. The closely spaced colored spheres with rods depict the vehicle position and the sonar main lobe orientation. The coloring changes over time for better visualization.

The resulting sonar data using different simulation modes is detailed in the next section.

4.2. Simulation modes

The simulation results for the different available modes in the simulation framework are shown in Fig. 5 – Fig. 7. Aside from the speckle noise seen in the carrier frequency simulations, the linear FM baseband image shows the smearing effect introduced by the pulse length separately. The SNR is also higher on the FM image than on the CW image which is to be expected due to pulse compression.

A simulated image of a sunken shipwreck in a slightly rippled seafloor is shown in Fig. 8 and Fig. 9.

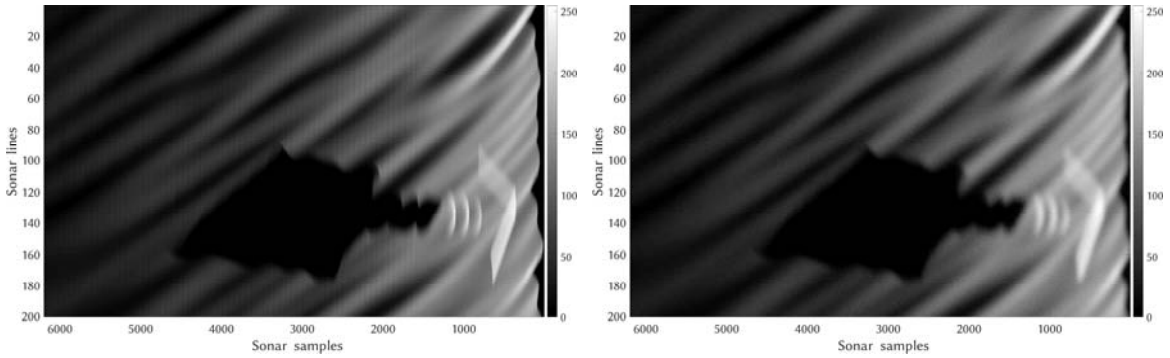

Fig. 5: Sonar baseband images. Left: No-speckle-mode (basically showing a perfect sonar image). Right: linear FM in baseband

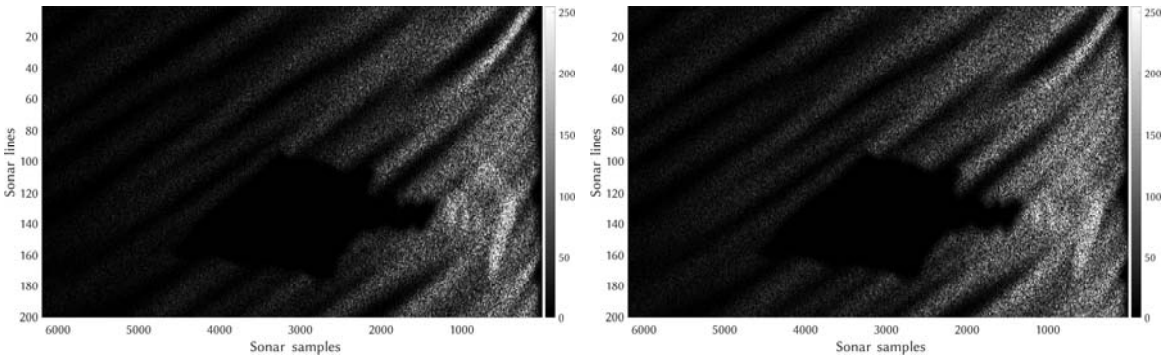

Fig. 6: Sonar images in carrier. Left: linear FM (4x contrast enhanced), CW with incoherent processing (4x contrast enhanced)

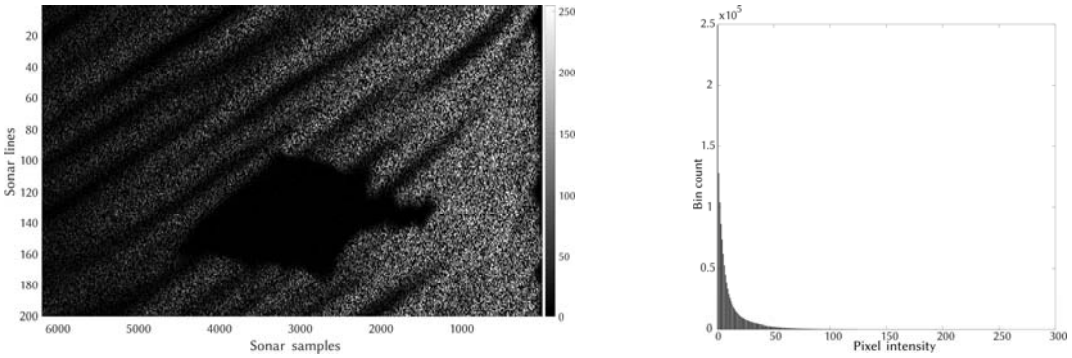

Fig. 7: Left: CW with coherent processing (3x contrast enhanced). Right: pixel brightness distribution for incoherent CW image.

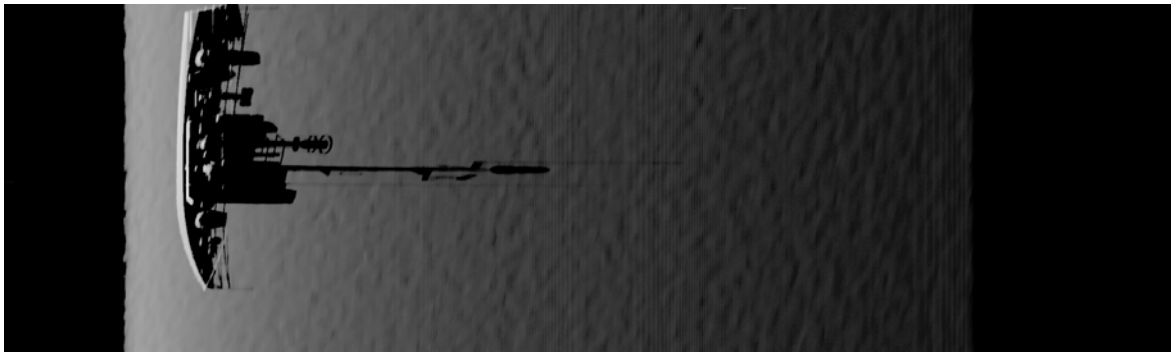

Fig. 8: Shipwreck from Fig. 1 and Fig. 2 in an FM baseband simulation using a straight vehicle trajectory.

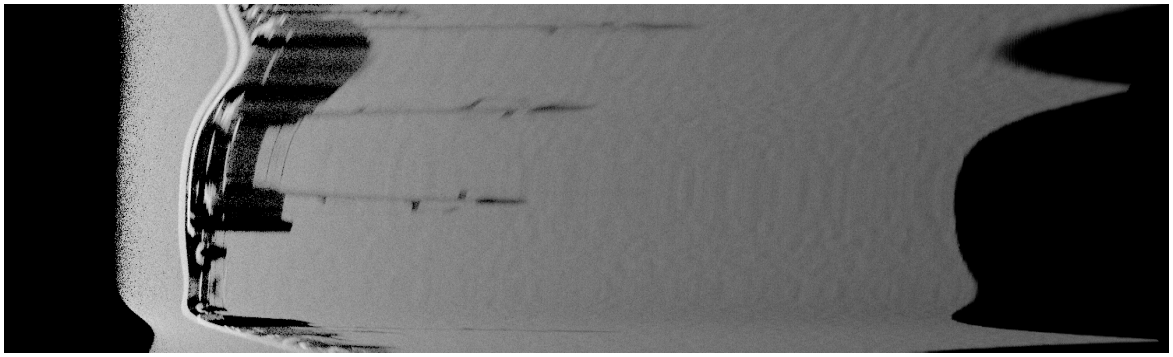

Fig. 9: Shipwreck from Fig. 1 and Fig. 2 in an incoherent CW simulation using a non-straight (slightly curved) trajectory causing image distortions

5. OUTLOOK

The presented framework could also be used for simulation of other scanning sonars like, e.g., mechanically scanning imaging sonars.

REFERENCES

[1] P. Woock and J. Beyerer, "Seafloor shape estimation by raytraced kernels," Taipei, Taiwan, 2014.

[2] P. Woock and J. Beyerer, "Seafloor heightmap generation using 1D Kernel Reconstructions," 2015.

[3] J. Riordan, E. Omerdic and D. Toal, "Implementation and application of a real-time sidescan sonar simulator," in *Oceans 2005 - Europe*, 2005.

[4] Y. Pailhas, Y. Petillot, C. Capus and K. Brown, "Real-time sidescan simulator and applications," in *OCEANS 2009*, 2009.

[5] J. Groen, "Adaptive motion compensation in sonar array processing," 2006.

[6] D. Guériot and C. Sintes, Sonar Data Simulation, INTECH Open Access Publisher, 2011.

[7] J. M. Bell, "A Model for the Simulation of Sidescan Sonar," Heriot-Watt University, Department of Computing and Electrical Engineering, Edinburgh, 1995.

[8] J. M. Bell and L. M. Linnett, "Simulation and analysis of synthetic sidescan sonar images," *IEE Proceedings -Radar, Sonar and Navigation,* vol. 144, pp. 219-226, 08 1997.