Custom Shader and 3D Rendering for computationally efficient Sonar Simulation

Rômulo Cerqueira*†, Tiago Trocoli*, Gustavo Neves*, Luciano Oliveira†, Sylvain Joyeux* and Jan Albiez*‡
*Brazilian Institute of Robotics, SENAI CIMATEC, Salvador, Bahia, Brazil, Email: romulo.cerqueira@fieb.org.br
†Intelligent Vision Research Lab, Federal University of Bahia, Salvador, Bahia, Brazil
‡Robotics Innovation Center, DFKI GmbH, Bremen, Germany

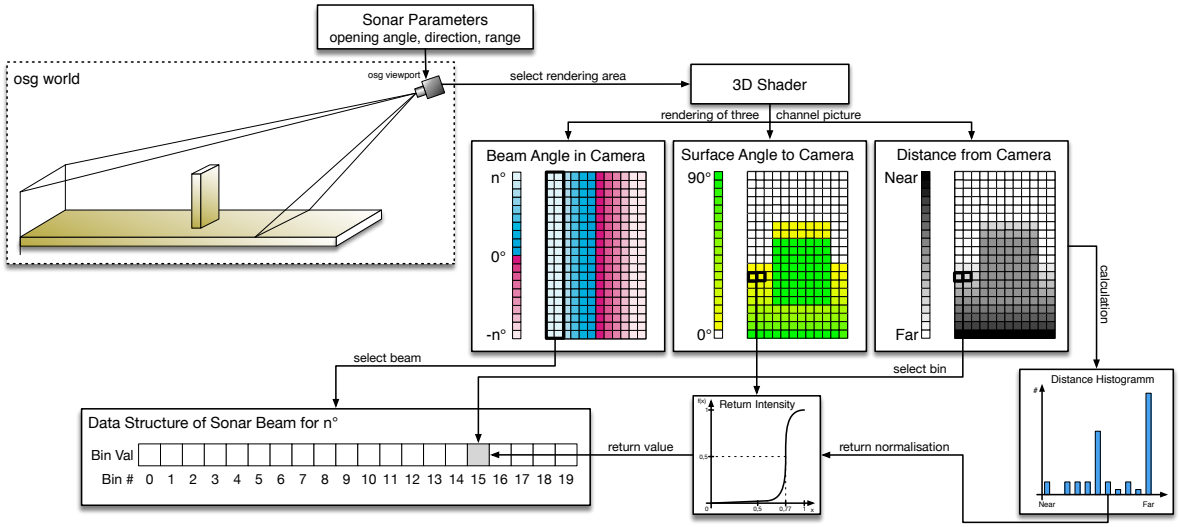

Fig. 1. A graphical representation of the individual steps to get from the OpenSceneGraph scene to a sonar beam data structure.

***Abstract*—This paper introduces a novel method for simulating underwater sonar sensors by vertex and fragment processing. The virtual scenario used is composed of the integration between the Gazebo simulator and the Robot Construction Kit (ROCK) framework. A 3-channel matrix with depth and intensity buffers and angular distortion values is extracted from OpenSceneGraph 3D scene frames by shader rendering, and subsequently fused and processed to generate the synthetic sonar data. To export and display simulation resources, this approach was written in C++ as ROCK packages. The method is evaluated on two use cases: the virtual acoustic images from a mechanical scanning sonar and forward-looking sonar simulations.**

***Keywords*-Synthetic Sensor Data; Sonar Imaging; Robot Construction Kit (ROCK); Underwater Robotics.**

I. Introduction

When designing and programming autonomous robotic systems, simulation plays an important role. This applies to physically correct simulations (which are needed to design the hardware but take longer to calculate), as well as to simulations which are not completely physically correct but run in real-time. The latter kind of simulation is important when it comes to developing and testing the control system of autonomous robots, especially the higher level parts. The key element here is that the simulation has to be *good enough* to test the decision making algorithms in the control system.

When dealing with autonomous underwater vehicles (AUVs) a real-time simulation plays a key role. Since an AUV can only scarcely communicate back via mostly unreliable acoustic communication, the robot has to be able to make decisions completely autonomously. While the part dealing with the analysis and interpretation of sensor data can be thoroughly tested on recorded data, for the test and verification of the vehicle's *reaction* to this data, a simulation is needed to reduce the risk of vehicle damage or even vehicle loss in the real world.

In the FlatFish project [1] was developed an interface to integrate the Gazebo real-time simulator [1] into the software framework ROCK [2] as presented in [2]. With this integration it is able to simulate basic underwater physics and underwater camera systems. The missing part, needed by most underwater robots, was the sonar system.

In this paper we present our current sonar simulation approach which uses a custom shader in a 3D rendering pipeline

[1]http://gazebosim.org
[2]http://rock-robotics.org/

to compute a sonar image with low computational cost. The model representation is presented in Figure 1.

A. Related work

Several models have been used to simulate sonar data. [3] applied frequency-domain signal processing to generate synthetic aperture sonar image. In this method, the acoustic image was created by expressing the Fourier transform of the received signal in terms of the transmitted signal. Simplifications in the frequency domain model resulted in a basic illumination model.

An application of optical ray tracing to the simulation of underwater side-scan sonar imagery was presented in [4]. The sonar images were generated by the use of acoustic signals represented by rays. The process of projecting rays is repeated for a 2D-array, representing all angles the sonar can emit signal. Then a 3D point cloud is constructed from the ray detection point with high computational costs.

The basic methodology of 2D forward-looking sonar simulation, using optical ray tracing combined with processing in the frequency domain, was proposed in [5]. The average simulation time of 2.5 minutes for one simulated image prevents its evaluation in real time.

In [6], a 2D forward-looking sonar was proposed using acoustic tubes instead of rays. This implementation added noise to the point cloud generated by rays before converting it into a sonar image, but the material reflectance was statically defined. It resulted in same intensity values for all points on a single object.

We are not aware of any previous work which customizes the 3D rendering pipeline to generate underwater sonar images – the present work therefore represents an important innovation in sonar simulation. Another contribution is that the method proposed herein is able to reproduce any type of underwater sonar images, as seen in evaluation tests with two kind of simulated sonars.

II. Sonar Background

Sonars use sound propagation in water to detect and identify submerged objects and their boundaries. An acoustic signal (or ping) is emitted by the sonar into an area to be observed. Then, the sonar listens for echoes that have been produced by the acoustic signal bouncing back from objects in the area [7].

A single beam emitted from a sonar transducer is shown in Figure 2. The azimuth θ_B and elevation ϕ_B widths show the horizontal and vertical beamwidths of the emitted sound pulse, respectively. The sonar data is formed by plotting the intensity received over time of the acoustic signal. Each record is also named as *bin*. So, every beam has a number of bins. Since the speed of sound underwater can be measured, the time-of-flight effectively corresponds to sonar range.

The propagation of acoustic waves in each beam can be modeled by the acoustic version of the wave equation [8]. Finally, the array of transducer readings forms the sonar image. Since all incoming signals converge on the same point, the reflected echo could have originated anywhere along the corresponding elevation width. Therefore, the 3D information is lost in the projection into a 2D image [9].

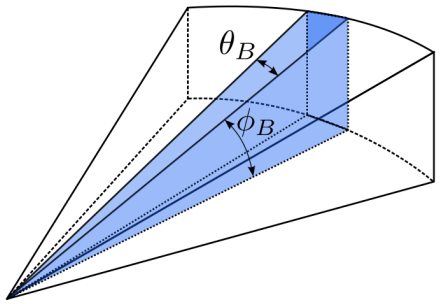

Fig. 2. Single sonar beam's geometry[6]

III. Development

The goal of this work was to simulate any kind of underwater sonar with low computation-time cost. The complete pipeline of this implementation, from the virtual scene to the synthetic acoustic image, is seen in Figure 1 and detailed in the following subsections.

A. Underwater Scene

The underwater scene was achieved by the ROCK-Gazebo simulator [2], where Gazebo is used to simulate the kinematics and the ROCK graphics tools are responsible for the visualization. ROCK's graphical engines are based on the OpenSceneGraph [3] library, which is a C/C++ 3D graphics toolkit based on OpenGL. The osgOcean [4] library is used to simulate the ocean's visual effects, and the ocean buoyancy is simulated by the Gazebo plugin described in [2].

The underwater scene's components, such as robot parts, sensors and joints, and the objects in the environment, are defined by means of SDF (Simulation Description Format) files, which uses the SDFormat [5], an XML format used to describe simulated models and environments.

Each component described in the SDF file becomes a ROCK component, which is based on the Orocos RTT (Real Time Toolkit) [6] and provides ports, properties and operations as its communication layer. When the models are loaded, ROCK-Gazebo creates ports that allow other system components to interact with the simulated models.

The underwater scene is illustrated in Figure 3.

B. Shader Rendering

Modern graphics hardware offers a way to customize tasks embedded in Graphical Processing Units (GPU). Based on parallel processing, this approach can speed up 3D graphics processing and reduce the computational effort of the Central Processing Unit (CPU).

[3]http://www.openscenegraph.org/
[4]http://wiki.ros.org/osgOcean
[5]http://sdformat.org
[6]http://www.orocos.org/rtt

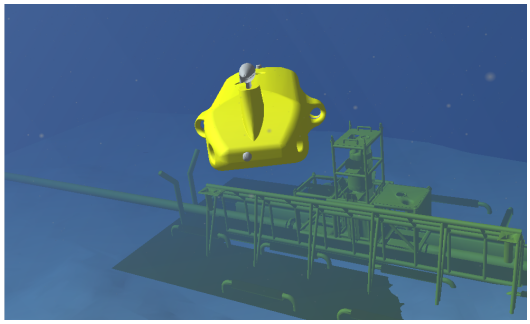

Fig. 3. The FlatFish robot in the ROCK-Gazebo underwater scenario.

The OpenGL Shading Language (GLSL [7]) is a high level programming language similar to C/C++, which allows to handle the rendering pipeline executed on the GPU. In this work, the rendering pipeline was specialized to simulate the sonar sensor as a camera of the 3D rendering process, with same 3D position, orientation and field of view horizontal and vertical (FOV-X, FOV-Y). With this approach, it is possible to compute 3D sonar data in a cost efficient parallel process:

- **Intensity** simulates the echo reflection energy based on an object's surface normal;
- **Depth** is the euclidean 3D distance between the camera focal point and the object's surface point;
- **Angle distortion** is the angle formed from the camera center column to the camera boundary column, for both directions;

These data are normalized between 0.0 and 1.0, where means, respectively, no echo energy and maximum echo energy for intensity data. For depth data, the minimum value portrays a close object while the maximum value represents a far object, limited by sonar max range. Angle distortion has 0.0 value in center column, and 1.0 value in both border columns which presents FOV-X half value. At the end, the shader process gives a 3-channel matrix data of intensity, depth and angle distortion stored in each channel.

C. Synthetic Sonar Data

The 3-channel matrix is processed in order to simulate the virtual sonar data. The initial step is to split the matrix in beam parts using the angular distortion presented in the shader matrix. In this case, all pixels in a column have the same angle value. Since the angular distortion is equally spaced over the FOV-X degree sector, each column is correlated with its respective beam, according to sonar bearings, as seen in Figure 1.

Each beam sub-image (with its respective columns) is converted into bin intensities using the depth and intensity values from shader process. In a real sonar, the bin number is proportional to the real distance from the sensor. In other words, the initial bins represent the closest distances, while the latest bins are the furthest ones. Therefore, a depth histogram is evaluated to associate each pixel with its respective bin, according to the depth channel. This information is used to calculate the intensity of each bin.

Due to acoustic attenuation in the water, the final bins have less echo strength than the first ones, because energy is lost in the environment. In order to correct for this, the sonar uses an energy normalization that applies a time varying gain to spreading losses in the bins. In this simulation approach, the accumulated intensity in each bin is normalized as seen in Equation 1:

$$I_{bin} = \sum_{x=1}^{n} \frac{1}{n} * sig(i(x)) \quad (1)$$

where I_{bin} is the accumulated intensity in the bin after the energy normalization, x is the pixel in the shader matrix, n is the depth histogram value (number of pixels) of that bin, $sig(.)$ is the sigmoid function and $i(.)$ is the intensity value of the pixel.

Since the shader returns a normalized final data in 8-bits color space (1 / 256 = 0.00390625), if the number of bins are greater than 256, the depth histogram will contain some blank spaces that will be reflected in the final sonar image as "black holes". To avoid this problem, it is necessary to distribute the sonar intensity data by applying a simple linear interpolation. After this, the simulation sonar data process is done.

For mechanical scanning sonars, with one beam per reading, the sonar image is built for each pulse. These images are usually shown on a display pulse by pulse, and the head position reader is rotated according to motor step angle. After a full 360 degree sector reading (or the desired sector defined by left and right limit angles), the accumulated sonar data is overwritten. For forward-looking sonars, with n beams being read simultaneously, the current data is overwritten by the next one, similar to a camera image.

D. ROCK's Sonar Structure

To export and display the sonar image, the simulated data is encapsulated as ROCK's sonar datatype and provided as an output port of ROCK's component.

IV. Results and Discussion

In order to evaluate the proposed method, the synthetic images generated by the underwater sonar simulators are presented here. The virtual scenario consisted of the FlatFish AUV, a manifold located in the seabed (Figure 3) and a grid around the robot (Figure 6).

The first experiment applied a forward-looking sonar with the following configuration: field of view of $120°$ by $20°$; 256 beams simultaneously; 500 bins per each beam; range set at $50m$; and angle tilt between the sonar and AUV at $20°$. The manifold model was ensonified to generate the acoustic image and its respective shader image from the OpenSceneGraph scene is presented in Figure 4. The frontal face of the manifold and its shadow, as a portion of the seabed, are clearly visible in the final sonar image, as seen in Figure 5.

A mechanical scanning sonar on top of the robot was simulated in the second experiment. It was configured as

[7] http://www.opengl.org/documentation/glsl/

Fig. 4. The shader image acquired by FlatFish's forward-looking sonar sensor.

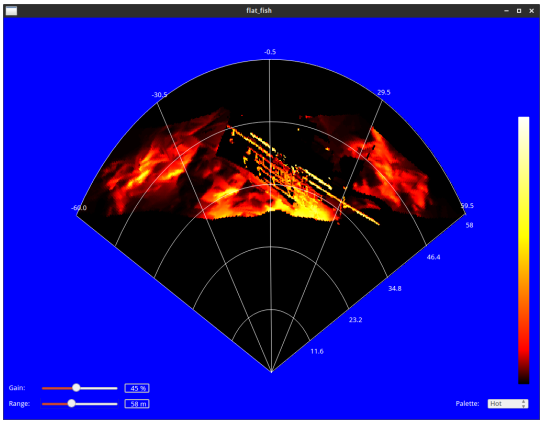

Fig. 5. The simulated forward-looking sonar image.

follows: field of view of 3° by 35°; 500 bins in the single beam; 360° sector scan reading; and a motor step angle of 1.8°. The rotation of the sonar head position produced the synthetic sonar image of the grid surrounding the robot seen in Figure 7.

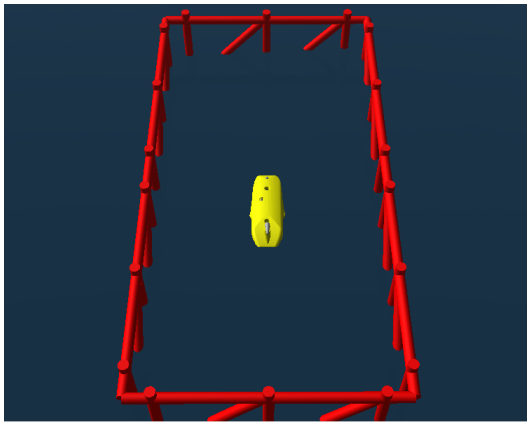

Fig. 6. The underwater scenario used in the mechanical scanning sonar simulation.

Finally, the computation-time was evaluated. For 150 sampling frames, the proposed method produced one multibeam sonar data every 121.44 milliseconds and one singlebeam sonar data every 8.5 milliseconds, much faster than the rates listed by the authors in [6] (1 second) and [5] (2.5 minutes).

In both experiments, the approach was able to simulate qualitative acoustic images in real time for different kinds of underwater sonars successfully.

V. Conclusion & Outlook

In this paper we presented a method using the shader engine of modern graphic cards to simulate different kinds

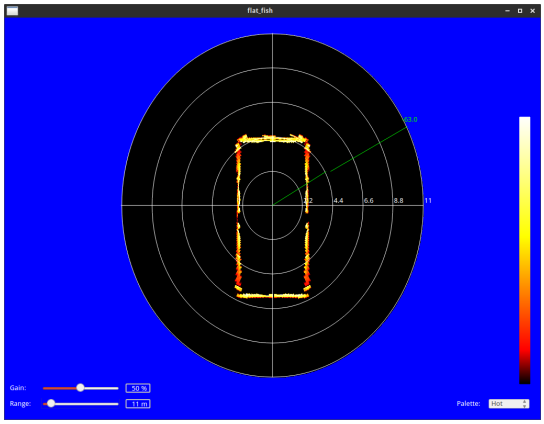

Fig. 7. The simulated mechanical scanning sonar image.

of sonars in a time-efficient way. The system is already used with success in our underwater projects as an extension of the Gazebo simulator.

Future work will focus mainly on adding different kinds of noise to make the images more realistic, add a simple refraction model and extend the 3D world by material properties to allow for different sonar reflections. Furthermore, we will perform a comprehensive comparison with other sonar simulators.

Acknowledgment

The authors would like to thank Shell Brazil and ANP for financing the work and SENAI CIMATEC and DFKI RIC for the great institutional support.

References

[1] J. Albiez, S. Joyeux, C. Gaudig, J. Hilljegerdes, S. Kroffke, C. Schoo, S. Arnold, G. Mimoso, R. Saback, J. Neto, D. Cesar, G. Neves, T. Watanabe, P. Merz Paranhos, M. Reis, and F. Kirchner, "FlatFish - a compact auv for subsea resident inspection tasks," in *Proceedings of the MTS/IEEE OCEANS 2015*, Washington DC, USA, Oct 2015, pp. 1–8.

[2] T. Watanabe, G. Neves, R. Cerqueira, T. Trocoli, M. Reis, S. Joyeux, and J. Albiez, "The rock-gazebo integration and a real-time auv simulation," in *Proceedings of 12th Latin American Robotics Symposium (LARS'15)*, Uberlandia, Brazil, Oct 2015, pp. 132–138.

[3] A. D. Wait, *Sonar for Practising Engineers*. Wiley, May 2002.

[4] J. M. Bell and L. M. Linnett, "Simulation and analysis of synthetic sides- can sonar images," in *Proceedings of the IEEE Radar, Sonar and Navigation*, 1997, pp. 219–226.

[5] H. Saç, K. Leblebicioğlu, and G. Bozdağı Akar, "2d high-frequency forward-looking sonar simulator based on continuous surfaces approach," *Turkish Journal of Electrical Engineering and Computer Sciences*, no. 23, pp. 2289–2303, 2015.

[6] K. DeMarco, M. West, and A. Howard, "A computationally-efficient 2d imaging sonar model for underwater robotics simulations in gazebo," in *Proceedings of the MTS/IEEE OCEANS 2015*, Washington DC, USA, Oct 2015, pp. 1–8.

[7] E. Coiras and J. Groen, "Simulation and 3d reconstruction of side-looking sonar images," in *Advances in Sonar Technology*, In-Tech, Ed., 2009, ch. 1, pp. 1–14.

[8] D. S. Drumheller, *Introduction to Wave Propagation in Nonlinear Fluids and Solids*. Cambridge University Press, 1998.

[9] N. Hurtos, "Forwad-looking sonar mosaicing for underwater environments," Ph.D. dissertation, Universitat de Girona, 2014.