A Novel GPU-based Sonar Simulation for Real-Time Applications

Rômulo Cerqueira[a,b], Tiago Trocoli[a], Gustavo Neves[a], Luciano Oliveira[b], Sylvain Joyeux[a], Jan Albiez[a,c]

[a] *Brazilian Institute of Robotics, SENAI CIMATEC, Salvador, Bahia, Brazil*
[b] *Intelligent Vision Research Lab, Federal University of Bahia, Salvador, Bahia, Brazil*
[c] *Robotics Innovation Center, DFKI GmbH, Bremen, Germany*

Abstract

XXXXXXXXXXXXXXXXXXXXXX

Key words: Synthetic Sensor Data, Sonar Imaging, GPU-based processing, Robot Construction Kit (Rock), Underwater Robotics.

1. Introduction

2. Sonar Background

Sonars are echo-ranging devices that use acoustic energy to locate and survey objects in a desired area underwater. The sonar's transducer emits an acoustic signal (or ping) until they hit with any object or be completely absorbed. When the ultrasonic wave collides with a surface, part of this energy is reflected and other one is refracted. Then the sonar data is built by plotting the echo measured back versus time of acoustic signal.

A single beam transmitted from a sonar is seen in Fig. XX. The horizontal and vertical beamwidths are represented by the azimuth θ_B and elevation ϕ_B angles respectively, where each sonar record along the beam is named bin. Since the speed of sound underwater is known or can be measured, the time axis effectively corresponds to distance from the device. The backscattered acoustic power in each bin determines the intensity value.

The array of transducer readings, with different azimuth directions, forms the final sonar image. Since all incoming signals converge on the same point, the reflected echoes could have originated anywhere along the corresponding elevation width. Therefore, the 3D information is lost in the projection into a 2D image [1].

Although the sonar devices address the main shortcomings of optical sensors, they present more difficult data interpretation, such as:

(a) Shadowing: It is caused by objects blocking the ultrasonic waves transmission and causing regions behind them without acoustic feedback;
(b) Nonuniform resolution: The amount of pixels used to represent a record intensity grow as its range increases.
(c) Changes in viewpoint: Imaging the same scene from different viewpoints can cause occlusions, shadows movements and significant alterations of observable objects. For instance, when an outstanding object is insonified, its shadow gets shortened as the sonar becomes closer to it;
(d) Low SNR (Signal-to-Noise Ratio): The sonar suffers from low SNR mainly due the very-long-range scanning and the presence of speckle noise introduced caused by acoustic wave interferences [2].

2.1. Imaging Sonar Devices

The most common types of acoustic sonars are MSIS (Mechanical Scanning Imaging Sonar) and FLS (Forward-Looking Sonar). In the first one (Fig. XX), 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° sector reading (or the desired sector defined by left and right limit angles), the accumulated sonar data is overwritten. Since the vehicle is moving during the data acquisition process, a undistortion method need to be applied. This sonar device is useful for obstacle avoidance [3] and navigation [4] applications.

For the FLS, with n beams being read simultaneously, the current data is overwritten by the next one with a high framerate, similar to a streaming video imagery for real-time applications. This imaging sonar is commonly used for navigation [5], mosaicing [6] and target tracking [7] approaches.

2.2. Underwater Sonar Devices

3. Closely-related Works

4. GPU-based Sonar Simulation Modeling

The goal of this work is to simulate any kind of underwater sonar by vertex and fragment processing, with a low computational-time cost. The complete pipeline of this implementation, from the virtual scene to the synthetic acoustic image, is seen in Fig. XX and detailed in the following subsections. The sonar simulation is written in C++ with OpenCV [1] support as ROCK [2] packages.

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

4.1. Underwater Environment

The Rock-Gazebo integration [8] provides the underwater scenario and allows real-time Hardware-in-the-Loop simulations, where Gazebo handles the physical engines and the Rock's graphics tools are responsible by the scene visualization. The graphical data in Rock are based on OpenSceneGraph [3] library, a open source C/C++ 3D graphics toolkit built on OpenGL. The osgOcean [4] library is used to simulate the ocean's visual effects, and the ocean buoyancy is defined by the Gazebo plugin as described in [8].

All scene's aspects, such as terrain, robot parts (including sensors and joints) and others objects presented in the environment, are defined by SDF files, which uses the SDFormat [5], a 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 to allow other system components to interact with the simulated models [9].

4.2. Shader Rendering

Shaders run on the graphic card and allow to take full control of the rendering pipeline process. The OpenGL Shading Language (GLSL) [7] is a high-level language with a C-based syntax which handles the graphics pipeline on the GPU (Graphics Processing Unit). In order to simulate the sonar sensoring, the fragment and vertex shaders are designed as a camera in an orthographic projection, with a desired position, orientation and field of view. Then the 2D sonar data is computed as:

- *Depth* is the camera focal length and is calculated by the euclidean distance to object's surface point;
- *Intensity* presents the echo reflection energy based on an object's surface normal.

Most real-world surfaces present irregularities and different reflectances. For more realistic sensing, the normal data can also defined by bump mapping and material properties. Bump mapping is a normal-perturbation rendering technique to simulate bumps and wrinkles on the object's surface by passing textures and modifying the normal directions. It is much faster and consumes less resources for the same level of detail compared to displacement mapping, because the geometry remains unchanged. Since bump maps are built in tangent space, interpolating the normal vertex and the texture, a TBN(tangent, bitangent and normal) matrix is used to convert the normal values to world space.

[3] http://www.openscenegraph.org/
[4] http://wiki.ros.org/osgOcean
[5] http://sdformat.org
[6] http://www.orocos.org/rtt
[7] https://www.opengl.org/documentation/glsl/

4.3. Synthetic Sonar Data

The 3-channel matrix is processed in order to simulate the virtual sonar data. Initially the matrix is splitted in beam parts. The angular distortion is radially spaced over the horizontal field of view, so each column is correlated with its respective beam, according to sonar bearings, as seen in Fig. **??**.

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

$$I_{bin} = \sum_{x=1}^{n} \frac{1}{n} \times S(i_x), \tag{1}$$

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, $S(i_x)$ is the sigmoid function and i_x is the intensity value of the pixel x.

Finally, the shader image resolution needs to be big enough to fill all bins. If the number of bins is greater than 750, the depth histogram will contain some blank spaces that will reflect 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.

4.4. Noise Model

Imaging sonar systems are perturbed by a multiplicative noise known as speckle. It is caused by coherent processing of backscattered signals from multiple distributed targets, that degrades image quality and the visual evaluation. The noisy image has been expressed as [**?**]:

$$y(t) = x(t) \times n(t), \tag{2}$$

where t is the time instant, $y(t)$ is the noised image, $x(t)$ is the free-noise image and $n(t)$ is the speckle noise matrix. This kind of noise is well-modeled as a Gaussian distribution. The physical explanation is provided by the Central Limit of Theorem, which states that the sum of many independent and identically distributed random variables tends to behave a Gaussian random variable [10]. For a more realistic sensing, a Gaussian distribution is built and applied as speckle noise in the simulated sonar image. After that, the simulation sonar data process is done.

4.5. ROCK's Sonar Structure

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

5. Results and Discussion

6. Conclusion and Outlook

References

[1] Hurtós N, Palomeras N, Carrera M, Bechlioulis C, Karras GC, Heshmati-alamari S, et al. Sonar-based chain following using an autonomous underwater vehicle. In: Proceedings of the IEEE/RSJ IROS 2014. 2014, p. 1978–83.

[2] Abbott JG, Thurstone FL. Acoustic speckle: Theory and experimental analysis. Ultrasonic Imaging 1973;1(4):303–24.

[3] Ganesan V, Chitre M, Brekke E. Robust underwater obstacle detection and collision avoidance. Autonomous Robots 2015;:1–21.

[4] Ribas D, Ridao P, Neira J. Underwater SLAM for Structured Environments using an Imaging Sonar. Springer-Verlag Berlin Heidelberg; 2010.

[5] Fallon MF, Folkesson J, McClelland H, Leonard JJ. Relocating underwater features autonomously using sonar-based slam. Journal of Ocean Engineering 2013;:500–13.

[6] Hurtós N. Forwad-looking sonar mosaicing for underwater environments. Ph.D. thesis; Universitat de Girona; 2014.

[7] Liu L, Xu W, Bian H. A lbf-associated contour tracking method for underwater targets tracking. In: Proceeding to OCEANS 2016 MTS/IEEE Monterey. 2016, p. 1–5.

[8] Watanabe T, Neves G, Cerqueira R, Trocoli T, Reis M, Joyeux S, et al. The rock-gazebo integration and a real-time auv simulation. In: Proceedings of 12th Latin American Robotics Symposium (LARS'15). Uberlândia, Brazil; 2015, p. 132–8.

[9] Cerqueira R, Trocoli T, Neves G, Oliveira L, Joyeux S, Albiez J. Custom shader and 3d rendering for computationally efficient sonar simulation. In: Proceedings of 28th Conference on Graphics, Patterns and Images (SIGRAPH'16). São José dos Campos, Brazil; 2016, p. 1–5.

[10] Papoulis A, Pillai S. Probability, random variables and stochastic processes. McGraw Hill; 2002.

[11] Bell JM, Linnett LM. Simulation and analysis of synthetic sides-can sonar images. In: Proceedings of the IEEE Radar, Sonar and Navigation. 1997, p. 219–26.

[12] DeMarco K, West M, Howard A. A computationally-efficient 2d imaging sonar model for underwater robotics simulations in gazebo. In: Proceedings of the MTS/IEEE OCEANS 2015. Washington DC, USA; 2015, p. 1–8.