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

Simulating realistic sensor data is of underlying importance to assist the algorithms and control systems evaluation, avoiding the risks and costs of real-life experiments in field robotics. Particularly in underwater robotics domain, this is so due the complexity to reproduce reasonable model of sensors, hydrodynamic forces acting on the robot and environments where the operations take place. This paper introduces an innovative underwater imaging sonar simulator by vertex and fragment processing on GPU (Graphics Processing Unit). The virtual scenario is composed by the integration between Gazebo simulator and the Rock (Robot Construction Kit) framework. The OpenSceneGraph 3D frame is processed during shader rendering in order to compute a 3-channel matrix with depth and intensity buffers and angular distortion values, which is subsequently fused to build the sonar image. Additional characteristics of imaging sonars such as speckle noise, material properties and object's surface irregularities are also addressed and introduced in sonar frame. To export and display simulation resources, this approach was written in C++ with OpenCV support as Rock packages. The method is evaluated by simulating two kind of imaging sonar devices in different scenarios with huge fidelity and high frame rate.

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

1. 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. It requires the availability of an applicable simulation platform for rapid prototyping and reproducible virtual environments and sensors 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. Underwater robots usually demand expensive hardware and their target domain can be difficult to access depending on the application. 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.

Due the AUV acts below the photic zone, with high turbidity and hugh light scattering, the image acquisition by optical devices is limited by short ranges and visibility conditions. Knowing these limitations, the high-frequency sonars systems have been used on navigation and perception applications. Acoustic waves are significantly less affected by water attenuation, facilitating operation at greater ranges even as low to zero visibility conditions with a fast refresh rate. Thus, sonar devices address the main shortcomings of optical sensors though at the expense of providing, in general, noisy data of lower resolution and more difficult interpretation.

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.

This paper presents a computationally efficient sonar simulator which manipulates the rendering pipeline to compute a sonar image by two kind of imaging sonar devices.

2. Background

2.1. Sonar Image Model

Sonars are echo-ranging devices that use acoustic energy to locate and survey objects in a desired area underwater. The sensor's transducer emit pulses of sound wave (or ping) until they

Email addresses: `romulo.cerqueira@ufba.br` (Rômulo Cerqueira)

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

hit with any object or be completely absorbed. When the acoustic signal collides with a surface, part of this energy is reflected, while other 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. 1. The horizontal and vertical beamwidths are represented by the azimuth ψ and elevation θ angles respectively, where each sampling along the beam is named *bin*. Since the speed of sound underwater is known or can be measured, the time delay between the emitted pulses and their echoes reveals how far the objects are and how fast they are moving. 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 arc at a fixed range, as seen in Fig. 1. Therefore, the 3D information is lost in the projection into a 2D image [3].

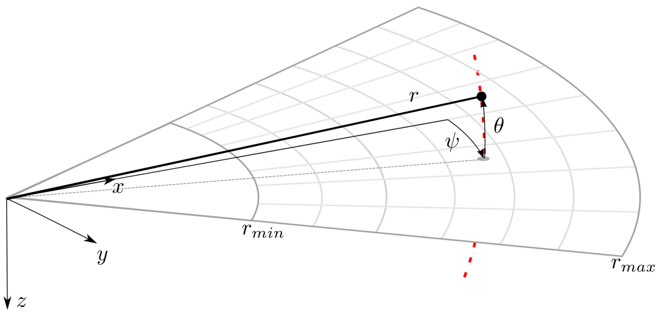


Figure 1: Imaging sonar geometry [3]. By the projection process, all 3D points belong the same elevation arc (represented as dashed red line) will be represented to the same image point in the 2D plane. So the range r and the azimuth angle ψ are measured, however the elevation angle θ is lost.

2.2. *Sonar Characteristics*

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

(a) Shadowing: This effect is caused by objects blocking the sound waves transmission and causing regions behind them without acoustic feedback. These regions are defined by a black spot in the image occluding part of the scene;

(b) Non-uniform resolution: The amount of pixels used to represent an intensity record grow as its range increases. This fact causes image distortions and object flatness;

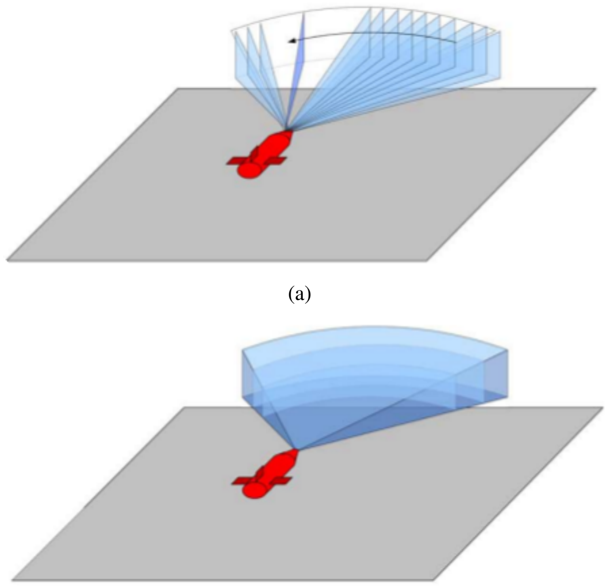
(c) Changes in viewpoint: Imaging the same scene from different viewpoints can cause occlusions, shadows movements and significant alterations of observable objects [4]. For instance, when an outstanding object is insonified, its shadow gets shortened as the sonar becomes closer;

(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 [5].

2.3. *Underwater 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. 2(a)), 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. In contrast, the acquisition of a scan image involves a relatively long time and introduces distortions by vehicle movement. This sonar device is useful for obstacle avoidance [6] and navigation [7] applications.

For the FLS, as seen in Fig. 2(b), with n beams being read simultaneously, the whole forward view is scanned and 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 [8], mosaicing [4], target tracking [9] and 3D reconstruction [3] approaches.



(a)

(b)

Figure 2: Different underwater sonar readings: Mechanically Scanning Imaging Sonar (a) and Forward-Looking Sonar (b).

3. Closely-related Works

Recent works have been proposed models based on ray tracing and tube tracing techniques to simulate sonar data with very accurate results but at a high computational cost. An application of optical ray tracing to the simulation of underwater side-scan sonar imagery was formulated by Bell [10]. The 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. Waite [11] used of frequency-domain signal processing to generate synthetic aperture sonar frames. In this method, the acoustic image was created by expressing the Fourier transform of the acoustic pulse used to insonifying the scene.

For FLS simulations, Saç *et al* [12] described the sonar model by computing the ray tracing in frequency domain. When a ray hits an object in 3D space, three parameters are calculated to process the acoustic data: the euclidean distance from the sonar axis, the intensity of returned signal by Lambert Illumination model and the surface normal. The reverberation and shadow phenomena are also addressed. In DeMarco *et al* [13], the rays are used in Gazebo and ROS [3](Robot Operating System) integration to simulate the acoustic pulse and produce a 3D point cloud of covered area. Since the material reflectivity was statically defined, it resulted in the same intensity values for all points on a single object. Gu *et al* [14] modeled a FLS device where the ultrasound beams were formed by a set of rays. However, the acoustic image is significantly limited by its representation by only two colors: white, when the ray strike an object, and black for shadow areas. This approach was evolved by Kwak et al [15] by adding a sound pressure attenuation to produce the gray-scale sonar frame, but the other physical characteristics related to sound transmission are disregarded.

The proposed approach herein entails several novelties. As opposed to the related works, the depth and normal values are directly manipulated during the scene formation, which generate sonar frames with a low computational cost and allow the usage by real-time applications. Also, this method is able to reproduce any type of underwater sonar images, as seen in evaluation tests with two kind of sonar devices.

In addition to our previous work [16], the normal data can also be defined by bump mapping technique and material's reflectivity. Moreover, the speckle noise is modeled as a non-uniform Gaussian distribution and added to final sonar image.

4. GPU-based Sonar Simulation

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. 3 and is detailed in the following subsections. The sonar simulation is written in C++ with OpenCV [4] support as Rock packages.

4.1. Underwater Environment

The Rock-Gazebo integration [2] provides the underwater scenario and allows real-time Hardware-in-the-Loop simulations, where Gazebo handles the physical engines and the Rock's visualization tools are responsible by the scene rendering. The graphical data in Rock are based on OpenSceneGraph [5] library, an open source C/C++ 3D graphics toolkit built on OpenGL. The osgOcean [6] library is used to simulate the ocean's visual effects, and the ocean buoyancy is defined by the Gazebo plugin as described in Watanabe et al [2].

All scene's aspects, such as world model, robot parts (including sensors and joints) and others objects presented in the environment are defined by SDF files, which uses the SDFormat [7], a XML format used to describe simulated models and environments for Gazebo. Also, the vehicle and sensor robot description must contain a geometry file. Visual geometries used by the rendering engine are provided in COLLADA format and the collision geometries in STL data.

Each component described in the SDF file becomes a Rock component, which is based on the Orocos RTT (Real Time Toolkit) [8] 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 [16].

4.2. Shader Rendering

Modern graphics hardware presents programmable tasks embedded in GPU. Based on parallel computing, this approach can speed up 3D graphics processing and reduce the computational effort of Central Processing Unit (CPU).

The rendering pipeline can be customized by defining programs on GPU called shaders. A shader is written in OpenGL Shading Language (GLSL) [9], a high-level language with a C-based syntax which enables more direct control of graphics pipeline avoiding the usage of low-level or hardware-specific languages. Shaders can describe the characteristics of either a vertex or a fragment (a single pixel). Vertex shaders are responsible by transform the vertex position into a screen position by the rasterizer, generating texture coordinates for texturing, and lighting the vertex to determine its color. The rasterization results in a set of pixels to be processed by fragment shaders, which manipulate their locations, depth and alpha values and interpolated parameters from the previous stages, such as colors and textures [17].

In this work, the underwater scene is sampled by a virtual camera, whose optical axis is aligned with the intended viewing direction of the imaging sonar, even as the range and opening angle. By programming the fragment and vertex shaders, the sonar data is computed as:

(a) *Depth* is the camera focal length and is calculated by the euclidean distance to object's surface point;

(a) *Intensity* presents the echo reflection energy based on an object's surface normal;

(a) *Angular distortion* is the angle formed from the camera center column to the camera boundary column, for both directions.

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

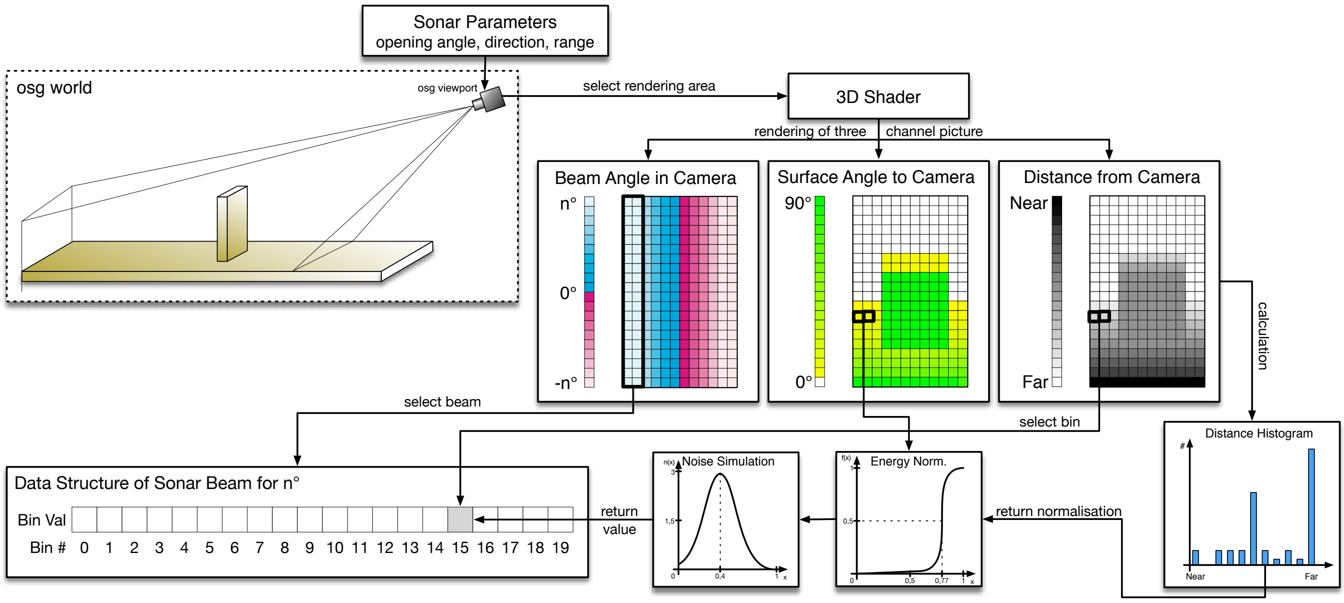


Figure 3: A graphical representation of the individual steps to get from the OpenSceneGraph scene to a sonar beam data structure.

These data are normalized in [0,1] interval, where means no energy and maximum echo energy for intensity data respectively. For depth data, the minimum value portrays a close object while the maximum value represents a far one, limited by the sonar maximum range. Angle distortion value is zero in image center column which increases for both borders to present the half value of horizontal field-of-view.

Most real-world surfaces present irregularities and different reflectances. For more realistic sensing, the normal data can also be defined by bump mapping and material properties. Bump mapping is a perturbation rendering technique to simulate 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 computed to convert the normal values to world space. The different scenes representation is seen in Fig. 4.

Moreover, the reflectance allows to describe properly the intensity back from observable objects in shader processing according their material properties (e.g. aluminium has more reflectance than wood and plastic). When an object has its reflectivity defined, the reflectance value R is passed to fragment shader and must be positive. As seen in Fig. 5, when the normal values are directly proportional to the reflectance value R.

At the end, the shader process gives a 3-channel matrix data of intensity, depth and angular distortion stored in each channel.

4.3. Synthetic Sonar Data

The 3D shader matrix is processed in order to build the corresponding acoustic representation. Since the angular distortion is radially spaced over the horizontal field of view, where all pixels in the same column have the same angle value, the first step is to split the image in number of beam parts. Each column is correlated with its respective beam, according to sonar bearings, as seen in Fig. 3.

Each beam subimage is converted into bin intensities using the depth and intensity channels. In a real imaging sonar, the echo measured back is sampled over time and the bin number is proportional to sensor's range. In other words, the initial bins represent the closest distances, while the latest bins are the furthest ones. Therefore, a distance histogram is evaluated to group the subimage pixels with their respective bins, according to depth channel. This information is used to calculate the accumulated intensity of each bin.

Due to acoustic beam spreading and absorption in the water, the final bins have less echo strength than the first ones, because the energy is lost two-way in the environment. In order to solve it, the sonar devices use a energy normalization based on time-varying gain for range dependence compensation which spread losses in the bins [18]. 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 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 sonar image resolution needs to be big enough to fill all bins informations. In this case, the number of bins involved is in direct proportion to the sonar image resolution.

4.4. Noise Model

Imaging sonar systems are perturbed by a multiplicative noise known as speckle. It is caused by coherent processing of

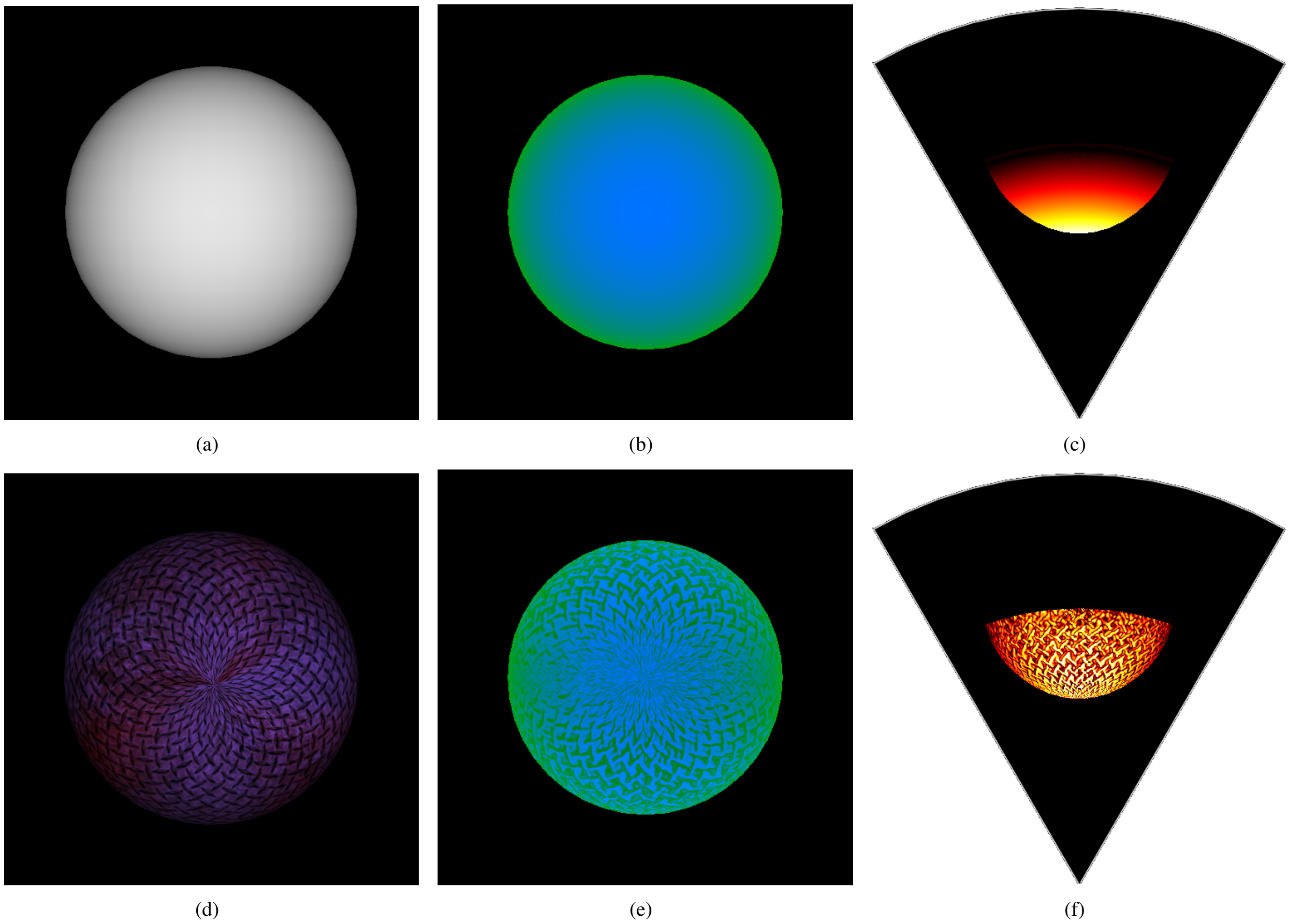


Figure 4: Shader rendering with bump mapping processing example: sphere without texture (a) and with texture (d); their respective shader image representation in (b) and (e), where the blue is the normal channel and green is the depth one; and the final acoustic image in (c) and (f). By bump mapping technique, the texture changes the normal directions and the sonar image are more realistic in comparison to real objects appearances.

backscattered signals from multiple distributed targets, that degrades image quality and the visual evaluation. Speckle noise results in constructive and destructive interferences which are shown as bright and dark dots in the image. The noisy image has been expressed as [19]:

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

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 [20].

A Gaussian distribution is built following a non-uniform distribution, skewed towards low values, as seen in Fig. 3, 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

5.1. Experiment Settings

For the evaluation of the proposed simulator, the experiments were conducted by using a 3D model of FlatFish AUV equipped with two MSIS and one FLS sensors on different scenarios, as seen in Fig. 6.

The MSIS sensors are located in AUV's top and back and they are configured as follows: opening angle of 3° by 35°, 500 bins in the single beam, a full 360° sector scan reading and a motor step of 1.8°. By other hand, the FLS takes place in AUV's bottom with the following set: field of view of 120° by 20°, 256 beams simultaneously, 1000 bins per each beam and angle tilt between the sonar and AUV of 20°.

5.2. Experimental Evaluation

The virtual FLS from FlatFish AUV was used to insonify scenes in three scenarios. The first one is composed by the docking station in parallel with a pipeline on the seabed, as seen

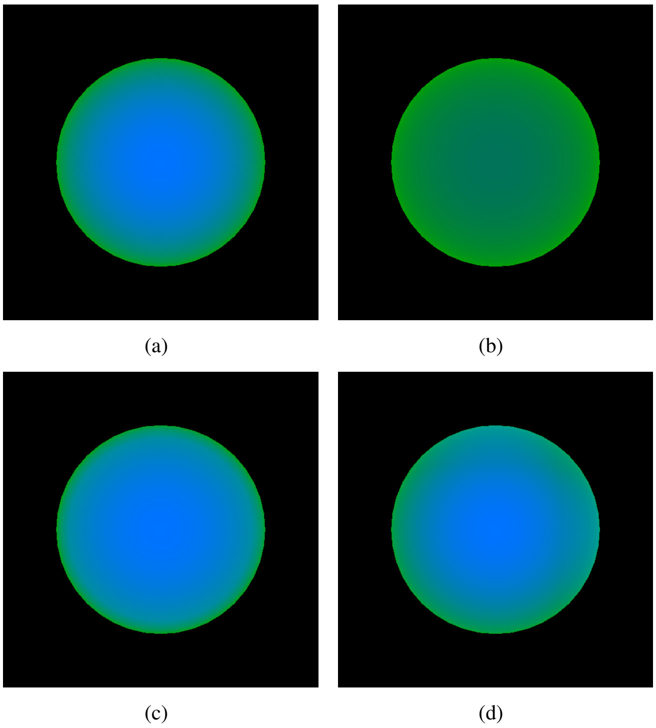


Figure 5: Examples of different reflectance values R on shader image representation, where blue is the normal channel and green is the depth channel: raw image (a); R = 0.35 (b); R = 1.40 (c); and R = 2.12 (d).

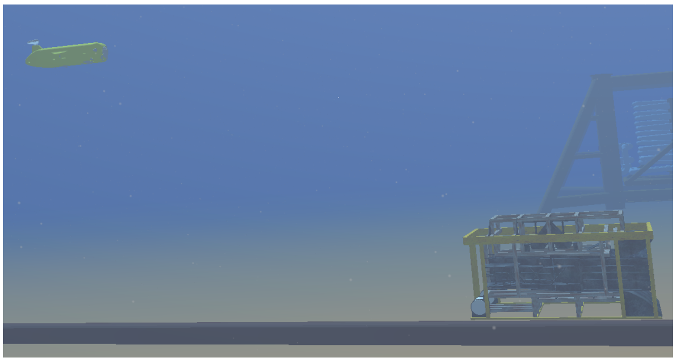


Figure 6: FlatFish AUV in ROCK-Gazebo underwater scene.

in Fig. 7(a). The target's surface is well-defined in the simulated acoustic frame, as seen in Fig. 7(b), even as the shadows and speckle noise. Given the docking station is metal-made, the texture and reflectivity were set, resulting in a higher intensity shape in comparison with the other targets.

The second scenario presents the FlatFish in front of a manifold model on a non-uniform seabed, as seen in Fig. 7(c). The target model was insonified to generate the sonar frame from the underwater scene. The frontal face of the target and the shadow behind it, as well the portion of the seabed and the degraded data by noise, are clearly visible in the FLS image.

The third scenario contains a SSIV (SubSea Isolation Valve) structure connected with a pipeline in the bottom, presented in Fig. 7(e). The targets' shapes are well-defined, such as their shadows.

Due the sensor configuration and the robot position, the initial bins usually present a blind region in the three simulated scenes, caused by absence of objects at lower ranges, similar with real images. Also, the brightness of seafloor decreases when it makes farthest from sonar due the normal orientation of surface.

The MSIS sensor was also simulated in three different experiments. The FlatFish robot in a big textured tank composed the first scene, as seen in Fig. 8(a). Even as the first scenario of FLS experiment, the reflectivity and texture were set to the target. The rotation of frontal sonar head position, by a complete 360° scanning, produced the acoustic frame of tank walls, seen in Fig. 8(b).

The second experiment involves the vehicle's movement during the data acquisition process. The scene contains a grid around the AUV, as seen in Fig. 8(c), and the frontal MSIS is used. This trial induces a distortion in the final acoustic frame, because the relative sensor's position with respect to surrounding object changes while the sonar image is being built, as seen in Fig. 8(d). In this case, the robot rotates 20° left during the scanning.

The last scenario presents the AUV over oil and gas structures on the sea bottom, as seen in Fig. 8(e). Using the back MSIS, with a vertical orientation, the scene was scanned in order to produce the acoustic visualization. As seen in Fig. 8(f), the objects' surfaces present clear definition in the small scanning section of the seafloor.

5.3. Computation time

The performance evaluation for this approach was determined as part of suitable analysis for real-time applications. The experiments were performed on a personal computer with Ubuntu 16.04 64 bits, Intel Core i7 3540M processor running at 3 GHz with 16GB DDR3 RAM memory and NVIDIA GF108GLM video card.

The elapsed time of each sonar data is stored to compute the mean and standard deviation metrics, after 500 iterations, as presented in Tables 1 and 2. After changing the device parameters, such as number of bins, number of beams and field of view, the proposed approach generated the sonar frames with a high frame rate, for both sonar types. Given the Tritech Gemini 720i, a real forward-looking sonar sensor with a field of view of 120° by 20° and 256 beams presents a maximum update rate of 15 frames per second, the results grant the usage of the sonar simulator for real-time applications. Also, the MSIS data built by the simulator is able to complete a 360° scan sufficiently time short in comparison with a real sonar as Tritech Micron DST.

Moreover, since the number of bins is directly proportional to sonar image resolution, as explained in Section 4.3, this is also correlated with the computation time. When the number of bins increases, the simulator will have a bigger scene frame to compute and generate the sonar data.

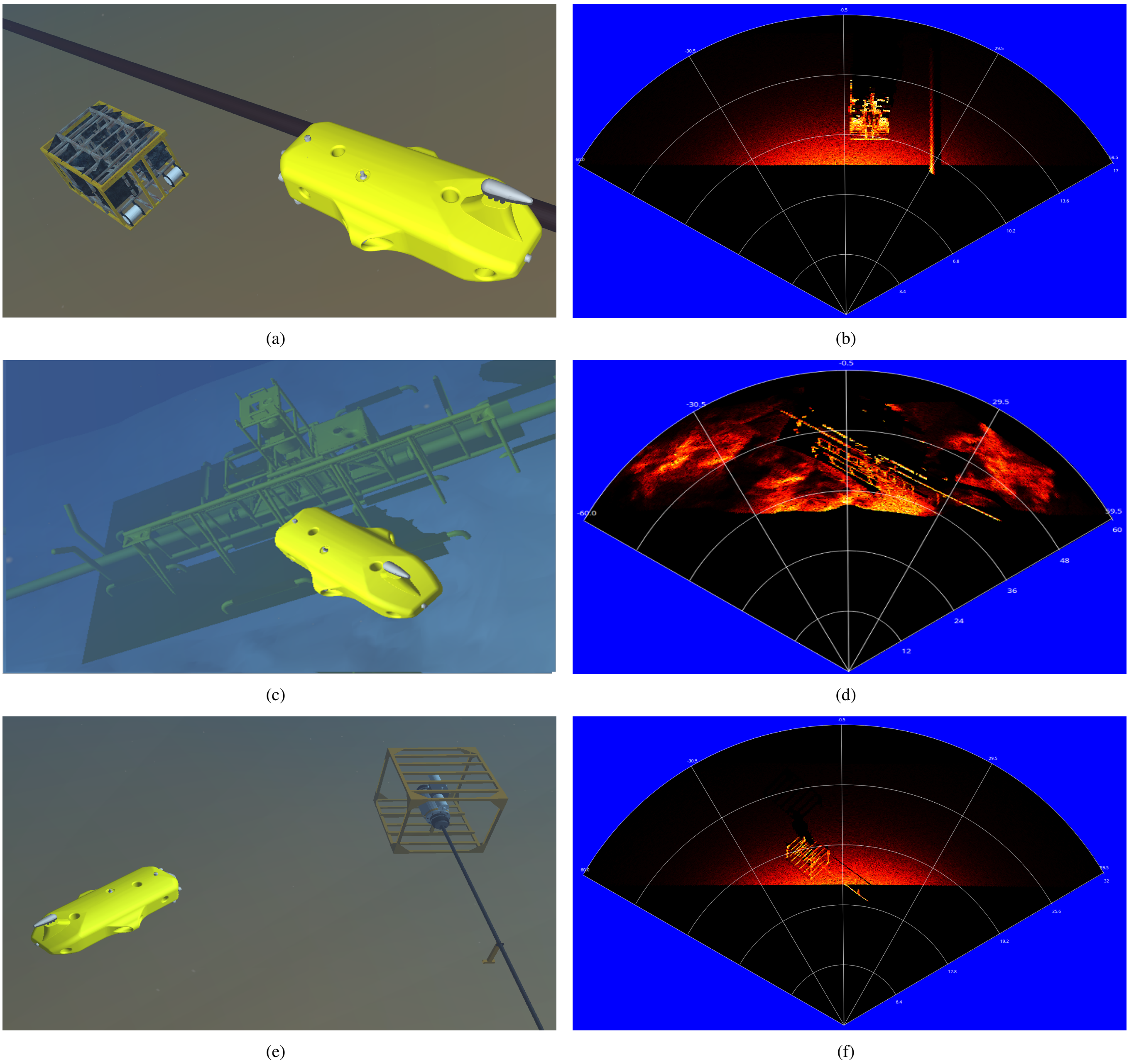


Figure 7: Forward-looking sonar simulation experiments: Figs. (a), (c) and (e) presents the trials underwater scenarios and Figs. (b), (d) and (f) are the following acoustic representations, respectively.

6. Conclusion and Outlook

We presented a GPU-based approach for imaging sonar simulation. By the evaluation results on different scenarios, the targets were well-defined on simulated sonar frames. The same model was able to reproduce the sensoring of two kind of sonar devices (FLS and MSIS). Moreover, the real sonar image singularities, such as speckle noise, surface irregularities, shadows, material properties and shapes are also addressed and represented on the synthetic acoustic images.

In addition, the processing time was calculated with different sonar parameters (field of view, number of bins and number of beams). The vertex and fragment processing during the underwater scene rendering accelerates the sonar image building and the mean and standard deviation metrics certified the performance is much closely to real imaging sonars. Therefore, the results granted the usage of this imaging sonar simulator by real-time applications, such as target tracking, obstacle avoidance and localization and mapping algorithms.

Next steps will focus on qualitative and computation-efficiency evaluations with other imaging sonar simulators.

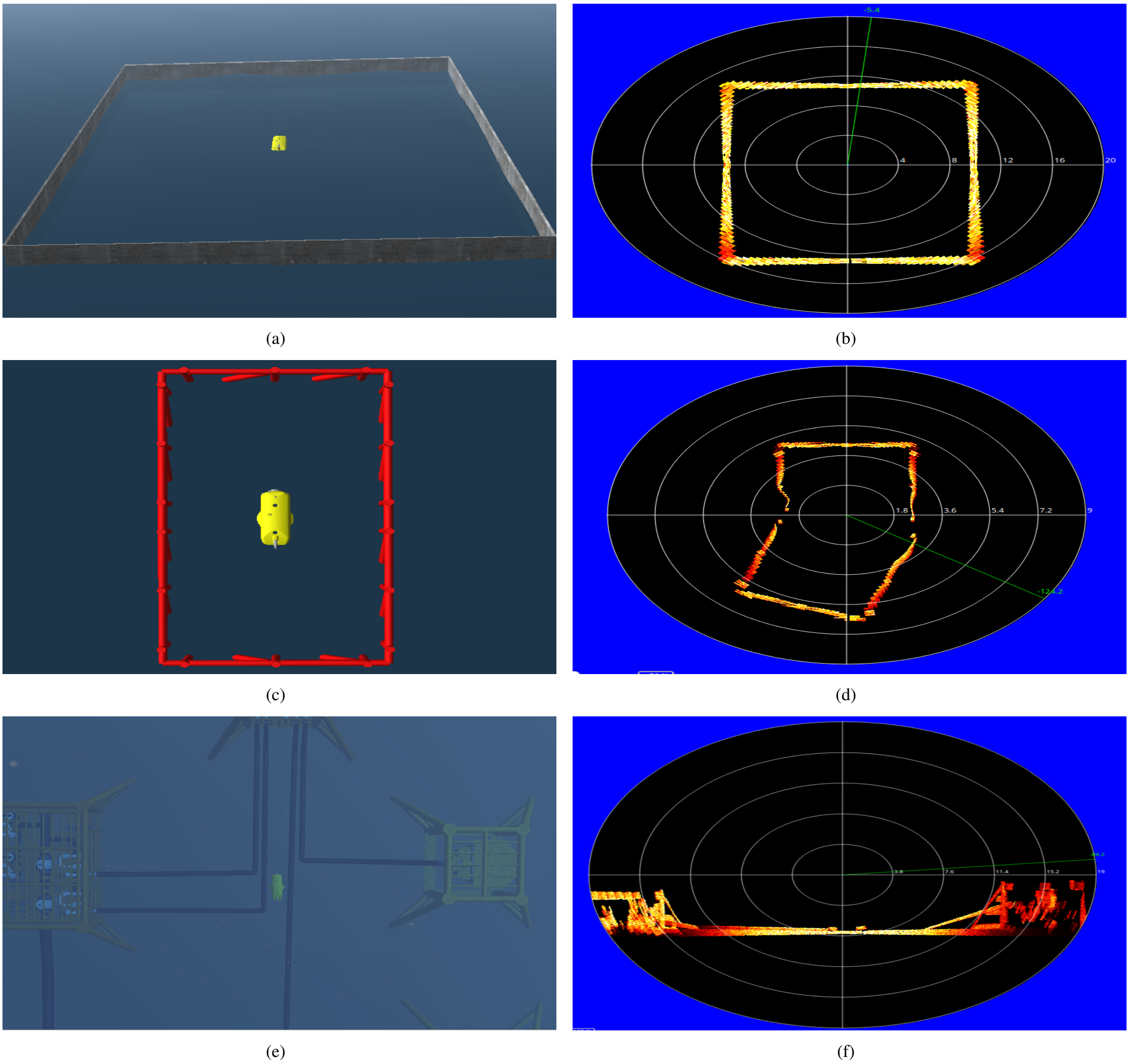


Figure 8: Mechanical Scanning Imaging Sonar trials: the underwater scenes represented in Figs. (a), (c) and (e) and their following simulated frames in Figs. (b), (d) and (f).

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] Albiez J, Joyeux S, Gaudig C, Hilljegerdes J, Kroffke S, Schoo C, et al. FlatFish - a compact auv for subsea resident inspection tasks. In: Proceedings of the MTS/IEEE OCEANS 2015. Washington DC, USA; 2015, p. 1–8.

[2] 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.

[3] Huang TA, Kaess M. Towards acoustic structure from motion for imaging sonar. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2015, p. 758–65.

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

Table 1: Processing time to generate FLS frames with different parameters.

Number of Beams	Number of Bins	Field of View	Mean (*sec*)	Standard Deviation (*sec*)
128	500	120° x 20°	0.0546834	0.00373812
128	1000	120° x 20°	0.0722763	0.00894485
256	500	120° x 20°	0.19877	0.0170872
256	1000	120° x 20°	0.218282	0.0119873
128	500	90° x 15°	0.0774186	0.0118534
128	1000	90° x 15°	0.0945958	0.0102294
256	500	90° x 15°	0.260864	0.0184956
256	1000	90° x 15°	0.26867	0.0166807

Table 2: Processing time to generate MSIS samples with different parameters.

Number of Bins	Field of View	Mean (*sec*)	Standard Deviation (*sec*)
500	3° x 35°	0.00881959	0.000709754
1000	3° x 35°	0.0345122	0.0015794
500	2° x 20°	0.0103457	0.000665683
1000	2° x 20°	0.0417138	0.00368668

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

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

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

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

[9] 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.

[10] 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.

[11] Wait AD. Sonar for Practising Engineers. Wiley; 2002.

[12] Saç H, Leblebicioğlu K, Bozdaği Akar G. 2d high-frequency forward-looking sonar simulator based on continuous surfaces approach. Turkish Journal of Electrical Engineering and Computer Sciences 2015;(23):2289–303.

[13] 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.

[14] Gu JH, Joe HG, Yu SC. Development of image sonar simulator for underwater object recognition. In: 2013 OCEANS - San Diego. 2013, p. 1–6.

[15] Kwak S, Ji Y, Yamashita A, Asama H. Development of acoustic camera-imaging simulator based on novel model. In: IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC). 2015, p. 1719–24.

[16] 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 (SIBGRAPH'16). São José dos Campos, Brazil; 2016, p. 1–5.

[17] Fernando R, Kilgard MJ. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2003.

[18] Urick R. Principles of Underwater Sound. Peninsula Publishing; 2013.

[19] Lee J. Digital image enhancement and noise filtering by use of local statistics. IEEE Trans Pattern Anal Mach Intell 1980;:165–8.

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