Elsevier Editorial System(tm) for Computers & Graphics

Manuscript Draft

Manuscript Number: CAG-D-17-00069R1

Title: A novel GPU-based sonar simulator for real-time applications

Article Type: VSI: SIBGRAPI 2017

Keywords: Simulated sensor data; sonar imaging; GPU-based processing; Robot Construction Kit; Underwater robotics.

Corresponding Author: Professor Luciano Rebouças de Oliveira, PhD

Corresponding Author's Institution: Federal University of Bahia

First Author: Rômulo Cerqueira, M.Sc.

Order of Authors: Rômulo Cerqueira, M.Sc.; Tiago Trocoli, M.Sc.; Gustavo Neves, BSc; Sylvain Joyeux, PhD; Jan Albiez, PhD; Luciano Rebouças de Oliveira, PhD

Abstract: Mainly when applied in the underwater environment, sonar simulation requires great computational effort due to the complexity of acoustic physics. For that, simulation of sonar operation allows evaluating algorithms and control systems without going to the real underwater environment; that reduces the costs and risks of in-field experiments. This paper tackles with the problem of real-time underwater imaging sonar simulation by using the OpenGL shading language chain on GPU. This proposed system is able to simulate two main types of acoustic devices: mechanical scanning imaging sonars and forward-looking sonars. The underwater scenario simulation is performed based on three frameworks: (i) OpenSceneGraph reproduces the ocean visual effects, (ii) Gazebo deals with physical forces, and (iii) the Robot Construction Kit controls the sonar in underwater environments. Our system exploits the rasterization pipeline in order to simulate the sonar devices, which are rendered by three parameters: the pulse distance, the echo intensity and sonar field-of-view, being all calculated over objects shapes in the 3D rendered scene. Sonar-intrinsic operational parameters, speckle noise and object material properties are also considered as part of the acoustic image. Our evaluation demonstrated that the proposed method is able to operate close or faster than the real-world devices, as well as generating realistic sonar image quality in different virtual underwater scenarios.

Dear Editor,

Please find our manuscript "A novel GPU-based sonar simulator for real-time applications", which we would like to submit for publication as an original article in your journal Computers & Graphics. Our article describes a novel simulator of two types of sonars. The goal is to simulate real-time applications. This work has not been submitted elsewhere.

We are looking forward to hearing from you.

Sincerely,

The corresponding author

A novel GPU-based sonar simulator for real-time applications

Abstract

Mainly when applied in the underwater environment, sonar simulation requires great computational effort due to the complexity of acoustic physics. For that, simulation of sonar operation allows evaluating algorithms and control systems without going to the real underwater environment; that reduces the costs and risks of in-field experiments. This paper tackles with the problem of real-time underwater imaging sonar simulation by using the OpenGL shading language chain on GPU. This proposed system is able to simulate two main types of acoustic devices: mechanical scanning imaging sonars and forward-looking sonars. The underwater scenario simulation is performed based on three frameworks: (i) OpenSceneGraph reproduces the ocean visual effects, (ii) Gazebo deals with physical forces, and (iii) the Robot Construction Kit controls the sonar in underwater environments. Our system exploits the rasterization pipeline in order to simulate the sonar devices, which are rendered by three parameters: the pulse distance, the echo intensity and sonar field-of-view, being all calculated over objects shapes in the 3D rendered scene. Sonar-intrinsic operational parameters, speckle noise and object material properties are also considered as part of the acoustic image. Our evaluation demonstrated that the proposed method is able to operate close or faster than the real-world devices, as well as generating realistic sonar image quality in different virtual underwater scenarios.

Key words: Simulated sensor data, Sonar imaging, GPU-based processing, Robot Construction Kit (Rock), Underwater robotics.

1. Introduction

Simulation is an useful tool for designing and programming autonomous underwater vehicles (AUVs). That allows evaluating the vehicle behavior, without dealing with physical hardware or decision-making algorithms and control systems in real-time trials, as well as costly and time-consuming field experiments. AUVs usually demand expensive hardware and perform long-term data gathering operations, taking place in restrictive sites. When AUVs are not supported by an umbilical cable, and the underwater communication carries on by unreliable acoustic links, the vehicle should be able to make completely autonomous decisions, even with low-to-zero external assistance. While the analysis and interpretation of sensor data can be performed in a post-processing step, a real-time simulation is strongly necessary for testing and evaluation of vehicle's motion response, avoiding involved risks on real-world rides.

AUVs usually act below the photic zone, with high turbidity and huge light scattering. This makes the quality of image acquisition by optical devices limited by a short range, and artificially illuminated and clear visibility conditions. To tackle with that limitations, high-frequency sonars have been used primarily on AUVs' navigation and perception systems. Acoustic waves emitted by sonars are significantly less affected by water attenuation, aiding operation at greater ranges even as low-to-zero visibility conditions, with a fast refresh rate. Although sonar devices usually solve the main shortcomings of optical sensors in underwater conditions, they provide noisy data of lower resolution and more difficult interpretation.

By considering sonar benefits and singularities along with the need to evaluate AUVs, recent works proposed ray tracing-[1, 2, 3, 4, 5, 6] and tube tracing-based [7] techniques to simulate acoustic data with very accurate results, although presenting a high computational cost. Bell [1] proposed a simulator based on optical ray tracing for underwater side-scan sonar imagery; images are generated by acoustic signals represented by rays, which are repeatedly processed, forming a 2D-array. Coiras and Groen [2] used frequency-domain signal processing to produce synthetic aperture sonar frames; in that method, the acoustic image is created by computing the Fourier transform of the acoustic pulse used to insonify the scene. For forward-looking sonar simulations, Saç *et al.* [3] described a 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 considered in the scene rendering. DeMarco *et al.* [4] used Gazebo and Robot Operating System (ROS) [8] integration to simulate acoustic sound pulses by ray tracing technique, also producing a 3D point cloud of the coverage area; the reflected intensity takes into account the object reflectivity, and the amount of Gaussian and salt-and-pepper noises applied in the sonar image is empirically defined. Gu *et al* [5] modeled a forward-looking sonar device, where the ultrasound beams are formed by a set of rays; the acoustic image is significantly limited by a representation using only two colors: white, when the ray strikes an object, and black for shadow areas. Kwak *et al.* [6] improved the previous approach by adding a sound pressure attenuation to produce the gray-scale sonar frame, while the other physical characteristics related to sound transmission are disregarded. Guériot and Sintes [7] introduce a volume-based approach of energy interacting with the scene, and collected by the receiving sonar; the sound propagation is

defined by series of acoustic tubes, being always orthogonal to the current sonar view, where the reverberation and objects surface irregularities are also addressed.

1.1. Contributions

This paper introduces a novel imaging sonar simulator that presents some contributions when compared to the existing approaches. Instead of simulating the sound pulse paths and the effects of their hits with the virtual objects, as presented by ray tracing and tube tracing-based methods [1, 2, 3, 4, 5, 6, 7], we take advantage of precomputed geometric data during the rasterization pipeline to compute the acoustic frame. In addition, all raster data are handled on GPU, accelerating then the simulation process with the guarantee of real-time response, in contrast to the methods found in [1, 2, 3, 4]. Although the systems found in [1, 2, 3, 4, 5, 6, 7] focused on the simulation of specific sonar device, our simulator is able to reproduce two kinds of sonar devices: mechanical scanning imaging sonar (MSIS) and forward-looking sonar (FLS). The intensity measured back from the insonified objects depends on surface normal directions and reflectivity, producing more realistic simulated frames than binary representation, this latter found in [5, 6]. The speckle noise is modeled as a non-uniform Gaussian distribution and applied to our final sonar image, which approaches to real-world sonar operation, differently from [3, 4, 5, 6, 7]. On the other hand, we did not exploit the additive noise as it was considered in [3, 4]. Finally, it is noteworthy that our proposed system simulates physical phenomena since they are constrained to real-time (e.g. decision-making algorithms and control system tuning). Aware of this real-time constraint, the high computational cost phenomena such as reverberation is not included at this point, differently from [3, 7].

The main goal here is to build quality and low time-consuming acoustic frames, according to underwater sonar image formation and operation modes (see Section 2). The pulse distance, the echo intensity and the sonar field-of-view parameters are extracted from the underwater scene during the rasterization pipeline, and subsequently fused to generate the simulated sonar data, as described in Section 3. Qualitative and time evaluation results for the two different sonar devices are presented in Section 4, allowing the use of the proposed simulator by real-time applications. Conclusions and future work are drawn in Section 5.

2. Imaging sonar operation

Sonars are echo-ranging devices that use acoustic energy to locate and survey objects in a desired area. The sonar transducer emits pulses of sound waves (or ping) until they hit any object or be completely absorbed. When the acoustic signal collides with a surface, part of this energy is reflected, while other is refracted. The sonar data is built by plotting the echo measured back versus time of acoustic signal. The transducer reading in a given direction forms a *beam*. A single beam transmitted from a sonar is illustrated 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 as *bin*. The sonar coverage area is defined by R_{min} and R_{max}. Since the speed of sound underwater is known, or can be measured, the time delay between the emitted pulses and the respective echoes (named as *time of flight*) reveals how far the objects are (distance r), as well as how fast they are moving. The backscattered acoustic power in each bin determines the echo intensity value.

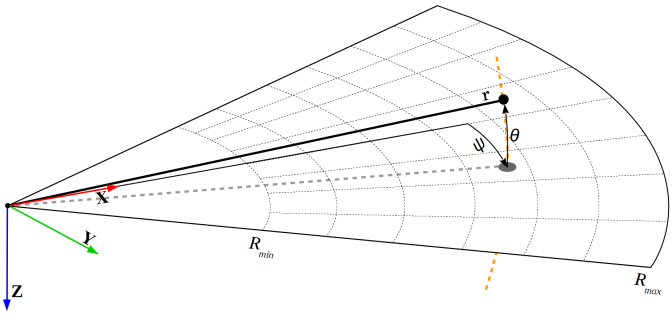


Figure 1: Imaging sonar geometry. By a projection process, all 3D points belonging to the same elevation arc (represented as dashd orange line) will be represented to the same image point in the 2D plane. Range r and azimuth angle ψ are measured, and elevation angle θ is lost. Sonar coverage area is defined by R_{min} and R_{max}.

With different azimuth directions, the array of transducer readings forms the final sonar image. Since all incoming signals converge to the same point, the reflected echoes could have been originated anywhere along the corresponding elevation arc at a fixed range, as depicted in Fig. 1. In the acoustic representation, the 3D information is lost in the projection into a 2D image.

2.1. Sonar characteristics

Although sonar devices overcome main limitations of optical sensors, they present more difficult data interpretation due to:

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 resulting sonar image, occluding part of the scene;

b) **Non-uniform resolution**: The amount of pixels used to represent an echo intensity record in the Cartesian coordinate system grows as its range increases. This situation causes image distortions and object flatness;

c) **Changes in viewpoint**: Imaging the same scene from different viewpoints can cause occlusions, shadows movements and significant changes of observable objects [9]. For instance, when an outstanding object is insonified, its shadow is shorter, as the sonar becomes closer;

d) **Low signal-to-noise ratio (SNR)**: sonars suffer from low SNR mainly due the very-long-range scanning, and the presence of speckle noise introduced by acoustic wave interferences [10];

e) **Reverberation**: This phenomenon is caused when multiple acoustic waves, returning from the same object, are detected over the same ping, producing duplicated objects.

(a) (b)

Figure 2: Different underwater sonar readings: (a) From a mechanical scanning imaging sonar and (b) from a forward-looking sonar.

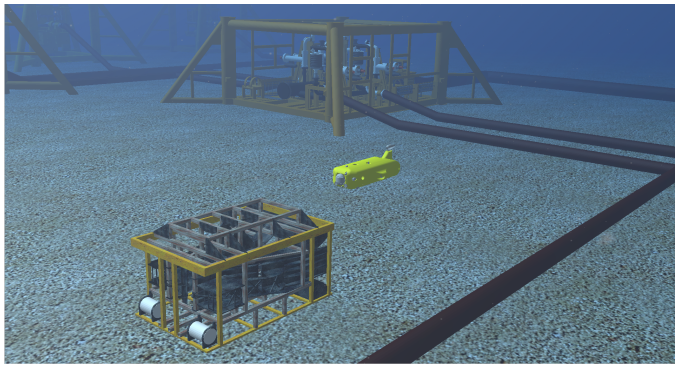


Figure 3: The AUV in Rock-Gazebo underwater scene.

2.2. *Types of underwater sonar devices*

The most common types of underwater acoustic sonars are MSIS and FLS. In the former, the sonar image is built for each pulse, with one beam per reading (see Fig. 2(a)); the resulting sonar images in MSIS are usually depicted 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. The acquisition of a scanning image involves a relatively long time, introducing distortions caused by the vehicle movements. This sonar device is generally applied in obstacle avoidance [11] and navigation [12] applications. As illustrated in Fig. 2(b), the whole forward view of an FLS is scanned and the current data is overwritten by the next scanning in a high frame rate, with all beams being read simultaneously; this is similar to a streaming video imagery for real-time applications; this imaging sonar is commonly used for navigation [13], mosaicing [9], target tracking [14] and 3D reconstruction [15].

3. GPU-based sonar simulation

The goal of our work is to simulate two types of underwater sonar with low computational cost. The complete pipeline of the proposed simulator (from the virtual scene to the simulated acoustic data) is detailed in the following sections. The sonar simulator is written in C++ with OpenCV [16] support as Rock packages.

3.1. *Rendering underwater scene*

In Rock-Gazebo framework [17], Gazebo handles with physical forces, while Rock's visualization tools are responsible by the scene rendering. The graphical data in Rock are based on OpenSceneGraph framework, an open source C/C++ 3D graphics toolkit built on OpenGL. The osgOcean library is used to simulate the ocean visual effects. In our case, Rock-Gazebo integration provides the underwater scenario, allowing also real-time hardware-in-the-loop simulation with a virtual AUV.

All scene aspects, such as world model, robot parts (including sensors and joints) and other virtual objects are defined by simulation description files (SDF), which use the SDFormat [18], an XML format used to describe simulated models and environments for Gazebo. Visual and collision geometries of vehicle and sensor robot are also described in specific file formats. Each component described in the SDF file becomes a Rock component, which is based on the Orocos real-time toolkit (RTT) [19], providing I/O ports, properties and operations as communication layers. When the models are loaded, Rock-Gazebo allows interaction between real world or simulated system components with the simulated models. A resulting scene sample of this integration is illustrated in Fig. 3.

3.2. *Sonar rendering*

A rendering pipeline can be customized by defining GPU shaders. A shader is written in OpenGL Shading Language (GLSL) [20], a high-level language with a C-based syntax, which enables more direct control of graphics pipeline, avoiding the use 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 transforming the vertex position into a screen position by the rasterizer, generating texture coordinates for texturing, and lighting the vertex to determine each color. The rasterization results, in a set of pixels to be processed by fragment shaders, manipulate pixel location, depth and alpha values, and interpolated parameters from the previous stages, such as colors and textures.

In our work, the underwater scenes are sampled by a virtual camera (frame-by-frame), whose optical axis is aligned with the **opening angle**, the intended **viewing direction** and the coverage **range** of the simulated sonar device (see Fig. 4(i)). To simulate the sonar imaging by using virtual camera frames, three

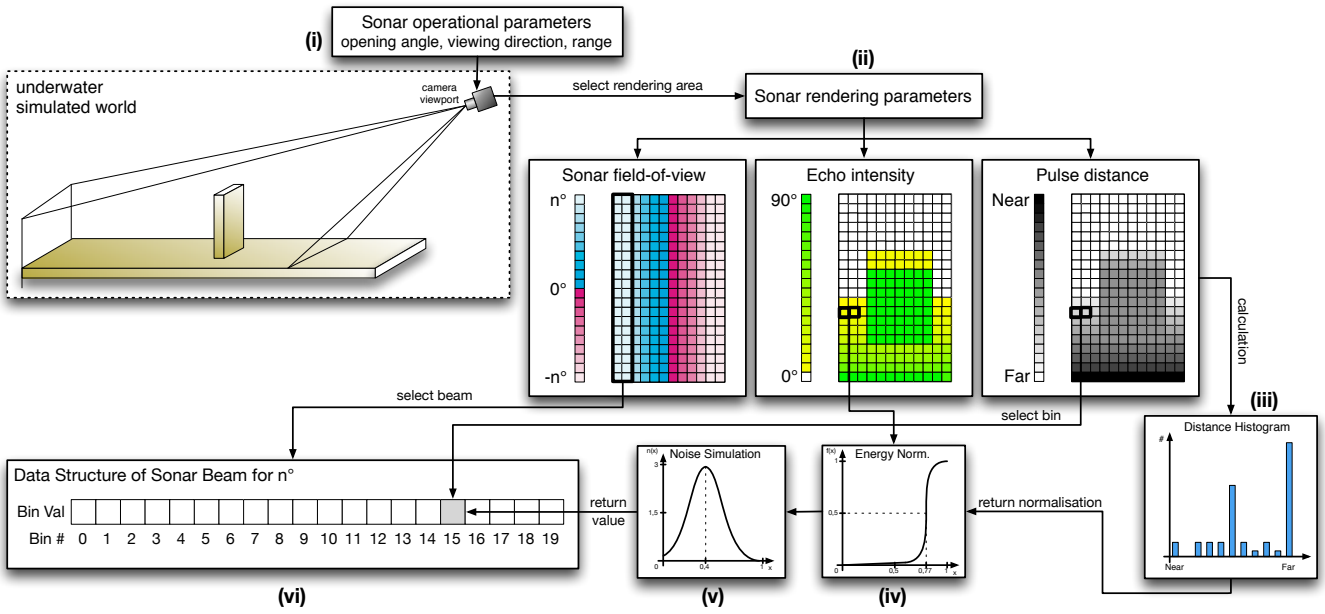


Figure 4: A graphical overview of the imaging sonar simulation process: (i) a virtual camera, specialized as the sonar device, samples the underwater scene; (ii) three 2D parameters are calculated by shader rendering on GPU: sonar field-of-view, echo intensity and pulse distance; the shader information is split into beam parts, according to the angle values, and the bin distance and echo intensity are defined by: (iii) distance histogram and (iv) energy normalization, respectively; (v) the speckle noise is applied to the final sonar data; (vi) and the simulated acoustic data is presented as Rock's data type.

parameters are computed in fragment and vertex shaders, during the rendering pipeline. This way, we are able to use the pre-computed geometric information during the image rasterization process on GPU. The three parameters to render the sonar device using a virtual camera are illustrated in Fig. 4(ii), and are described as follows:

- **Pulse distance** simulates the time of flight of the acoustic pulse, being calculated by the Euclidean distance between the camera center and the object surface;
- **Echo intensity** represents the reflection of the acoustic echo, calculated from the object surface normal regarding the camera;
- **Sonar field-of-view** is represented by the camera field-of-view in the horizontal direction.

By default, the shader encodes the raster data in 8-bit color channels for red, green, blue and alpha (RGBA). In our simulator, RGB channels are used to store the echo intensity, pulse distance and sonar field-of-view parameters to render the sonar from a virtual camera. The echo intensity parameter follows a real sonar common representation as 8-bit values. The pulse distance is replaced by the native GLSL 32-bit depth buffer to avoid precision limitation during the calculation of the distance histogram (see Fig. 4(iii)). As the field-of-view angle varies from the image center to both side directions, the sonar field-of-view is represented by 8-bit values without loss of precision. All of these three parameters are normalized into the interval [0,1]. For the echo intensity parameter, zero means no energy, while one means maximum echo energy. For pulse distance, the minimum value denotes a close object, while the maximum value represents a far one, limited by the sonar maximum range. Every sonar device has a maximum field-of-view; to represent this parameter in the rendering pipeline, the zero angle is in the center of the image, increasing until it reaches the half value of the maximum field-of-view of the simulated sonar device, for both sided borders; for example, if a sonar device has 120° of field-of-view, the zero angle is in the center of the virtual camera, spanning 60° to the right and 60° to the left.

In real-world sensing, surfaces usually present irregularities and different reflectance values. To render these surfaces in a virtual scene, the echo intensity values can also be defined by normal maps (see Fig. 5) and material property information (see Fig. 6). Normal mapping is a perturbation rendering technique to simulate wrinkles on the object surface by passing textures, modifying the normal directions on shaders. This approach consumes less computational resources for the same level of detail, compared with the displacement mapping technique, because the geometry remains unchanged. Since normal maps are built in tangent space, interpolating the normal vertex and the texture, tangent, bi-tangent and normal (TBN) matrices are computed to convert the normal values into the world space. The visual differences of applying normal mapping in the actual scenes are illustrated in Figs. 5(a) and 5(c); in the shader representation, in Figs. 5(e) and 5(b); and the final sonar image, in Figs. 5(d) and 5(f). The reflectance allows properly describing the intensity received back from observable objects in shader processing, according to the material properties (for instance, aluminum has more reflectivity than wood and plastic). When an object has the reflectivity property defined, the reflectance value ρ is passed to the fragment shader and processed on GPU. So, the final pixel intensity represents the product of

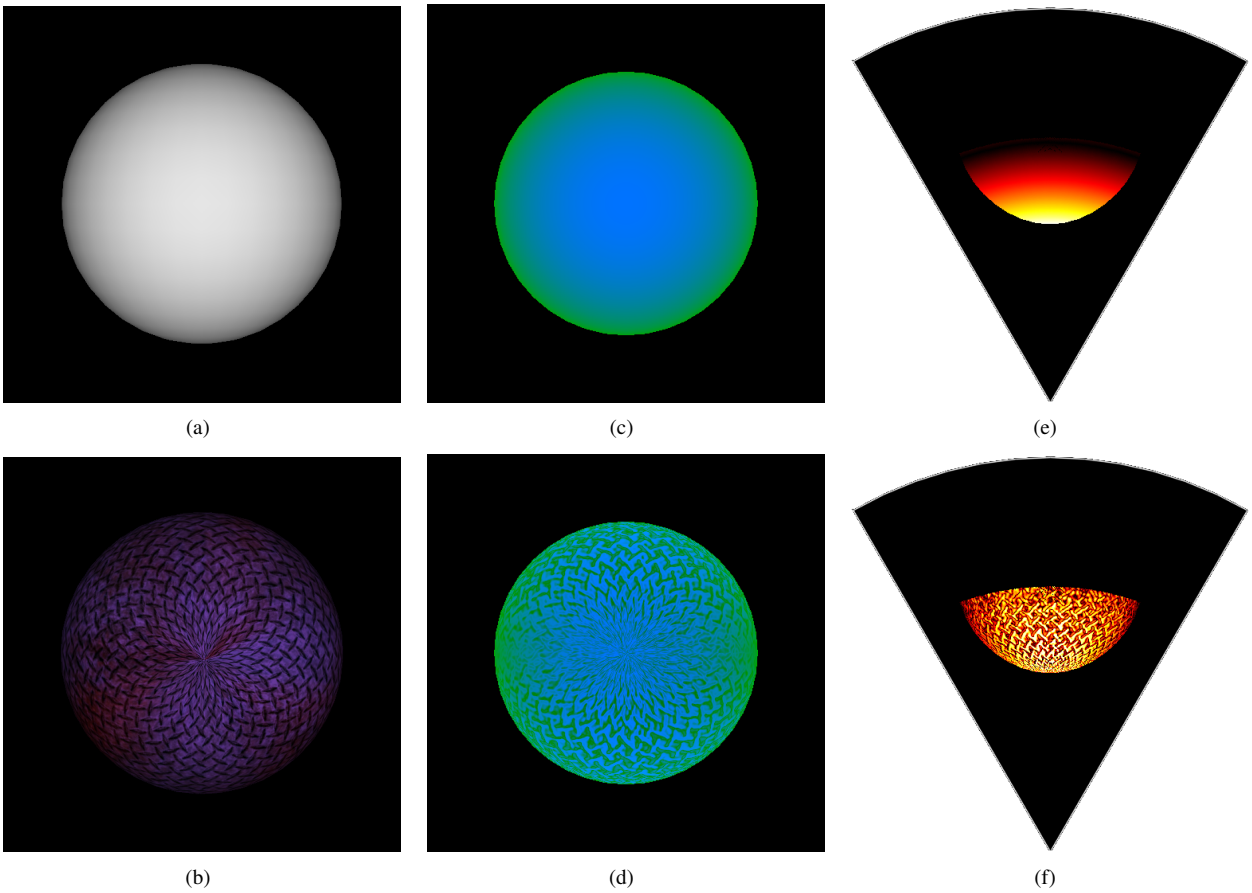


Figure 5: Example of shader rendering with normal mapping: A sphere without (a) and with texture (b); respective shader image representations of the spheres in (c) and (d), where the blue area represents the echo intensity parameter, while the green area represents the pulse distance parameter. The final acoustic images are depicted in (e) and (f). By using normal mapping technique, the textures changes the normal directions, and the sonar image details the appearance of object surface, like in real world sensing.

surface normal angle by the reflectance value ρ. The reflectance affects the shader representation, as depicted in Figs. 6(a), 6(b), 6(c) and 6(d)), with a final sonar image shown in Figs. 6(e), 6(f), 6(g) and 6(h).

3.3. Simulating operation of the sonar device

The sonar rendering parameters are used to compute the corresponding acoustic representation. Since the sonar field-of-view is radially spaced over the horizontal field-of-view of the virtual camera (where all pixels in the same column have the same angle), the first step is to split the image into a number of beams (beamed sub-images). Each column of the sonar field-of-view parameter is related with a respective beam vector, according to sonar bearings, as illustrated in Fig. 4(vi). In turn, one beam represents one or more columns. Each beamed sub-image is converted into bin intensities using the pulse distance and the echo intensity parameters. In a real imaging sonar, the echo measured back is sampled over time, and the bin number is proportional to the sensor range. In other words, the initial bins represent the closest distances, while the latest bins represent the farthest ones. Therefore a distance histogram (see Fig. 4(iii)) is computed in order to group the sub-image pixels with the respective bins, according to the pulse distance parameter and number of bins, and calculate the accumulated intensity in each bin.

Due to the acoustic beam spreading and absorption in the water, the final bins have less echo strength than the first ones. This is so, because the energy is twice lost in the environment. To tackle with that issue, sonar devices use an energy normalization based on time-varying gain for range dependence compensation, which spreads losses in the bins. In our simulation approach, the accumulated intensity, I_{bin}, in each bin (see Fig. 4(iv)) is normalized as

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

where x is the pixel location, N is the distance histogram value (number of pixels) of that bin, $S(i_x)$ is a sigmoid function, and i_x is the echo intensity value of the pixel x. $\times$ defines an element-wise multiplication.

Finally, the sonar image resolution must be big enough to contain all information of the bins. For that, the number of bins involved is directly proportional to the sonar image resolution.

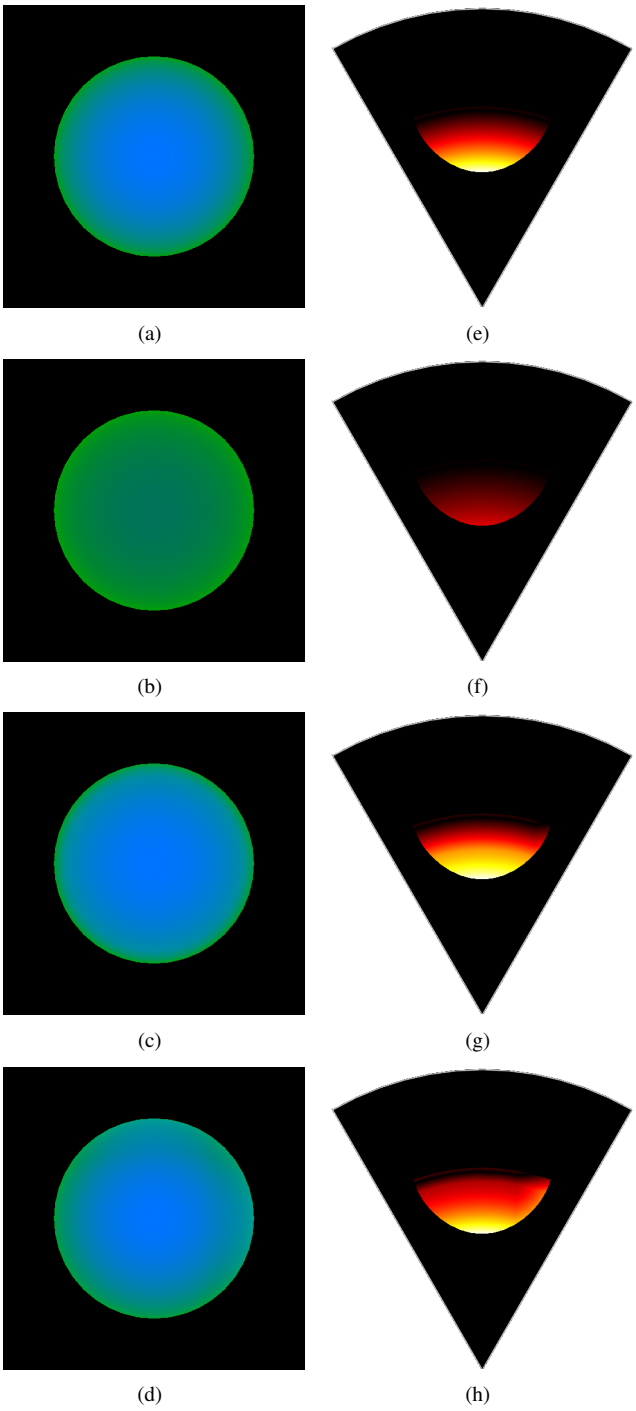


Figure 6: Examples of different reflectance values, ρ, applied in shader image representation of the same target, where blue is the echo intensity parameter and green is the pulse distance parameter: (a) raw image; (b) $\rho = 0.35$; (c) $\rho = 1.40$; and (d) $\rho = 2.12$. The following acoustic images are presented in (e), (f), (g) and (h).

3.3.1. Noise model

Imaging sonar systems are disturbed by a multiplicative noise known as speckle, which is caused by coherent processing of backscattered signals from multiple distributed targets. This effect degrades image quality and 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, following [21]:

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

where t is the time instant, $y(t)$ is the noised image, $x(t)$ is the free-noise image, $n(t)$ is the speckle noise matrix, and $\times$ defines an element-wise multiplication.

This type of noise is well-modeled as a Gaussian distribution. The physical explanation is provided by the central limit theorem, which states that the sum of many independent and identically distributed random variables tends to behave as a Gaussian random variable [22]. A Gaussian distribution is defined by following a non-uniform distribution, skewed towards low values, and applied as speckle noise in the simulated sonar image (see Fig. 4(v)). This noise simulation is repeated for each virtual acoustic frame.

3.3.2. Integrating sonar device with Rock

After the imaging sonar simulation process, from the virtual underwater scene to the representation of the degraded acoustic sonar data by noise, the resulting sonar data is encapsulated as Rock's sonar data type (see Fig. 4(vi)). This data type is provided as I/O port of a Rock's component, allowing the interaction with other simulated models and applications.

Table 1: Sonar device configurations used on experimental evaluation.

Device	# of beams	# of bins	Field of view	Down tilt	Motor Step
FLS	256	1000	120° x 20°	20°	-
MSIS	1	500	3° x 35°	0°	1.8°

4. Simulation results and experimental analysis

To evaluate our simulator, experiments were conducted by using a 3D model of an AUV equipped with an MSIS and an FLS. Different scenarios were casted and studied, considering the sonar device configurations summarized in Table 1. In the experimental analysis, as the scene frames are being captured by the sonars, the resulting acoustic images are sequentially presented, on-the-fly (see Figs. 7 and 8).

4.1. Experimental evaluation

The virtual FLS from AUV was used to insonify the scenes from three distinct scenarios. A docking station, in parallel with a pipeline on the seabed, composes **the first scenario** (see Fig. 7(a)); the target surface is well-defined in the simulated acoustic frame (see Fig. 7(d)), as well as the shadows and speckle noise; given that the docking station is metal-made, the texture and reflectivity were set such that a higher intensity shape was resulted in comparison with the other targets. **The second scenario** presents the vehicle in front of a manifold model in a non-uniform seabed (see Fig. 7(b)); the target model was insonified to generate the sonar frame from the underwater scene (see Fig.

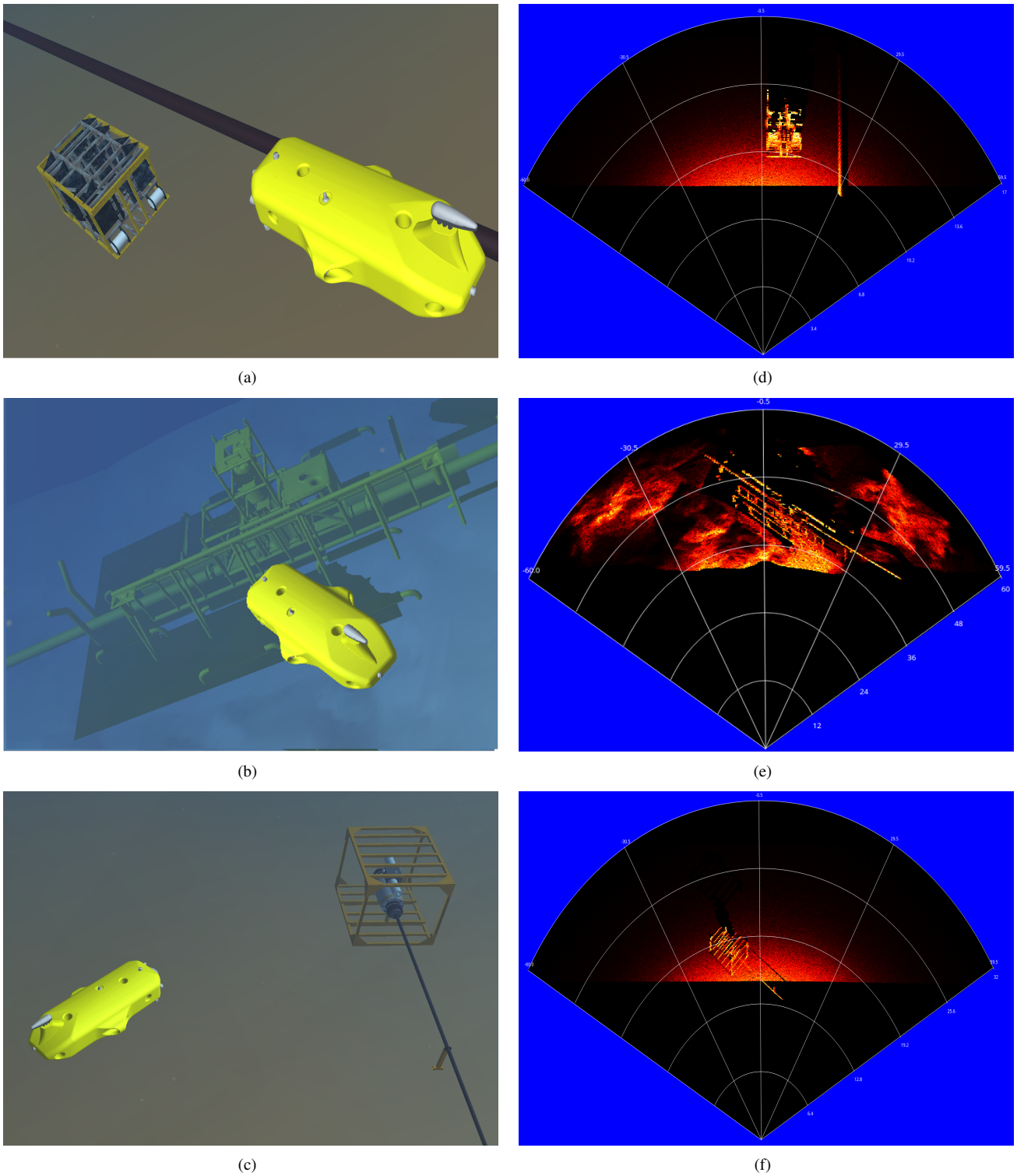
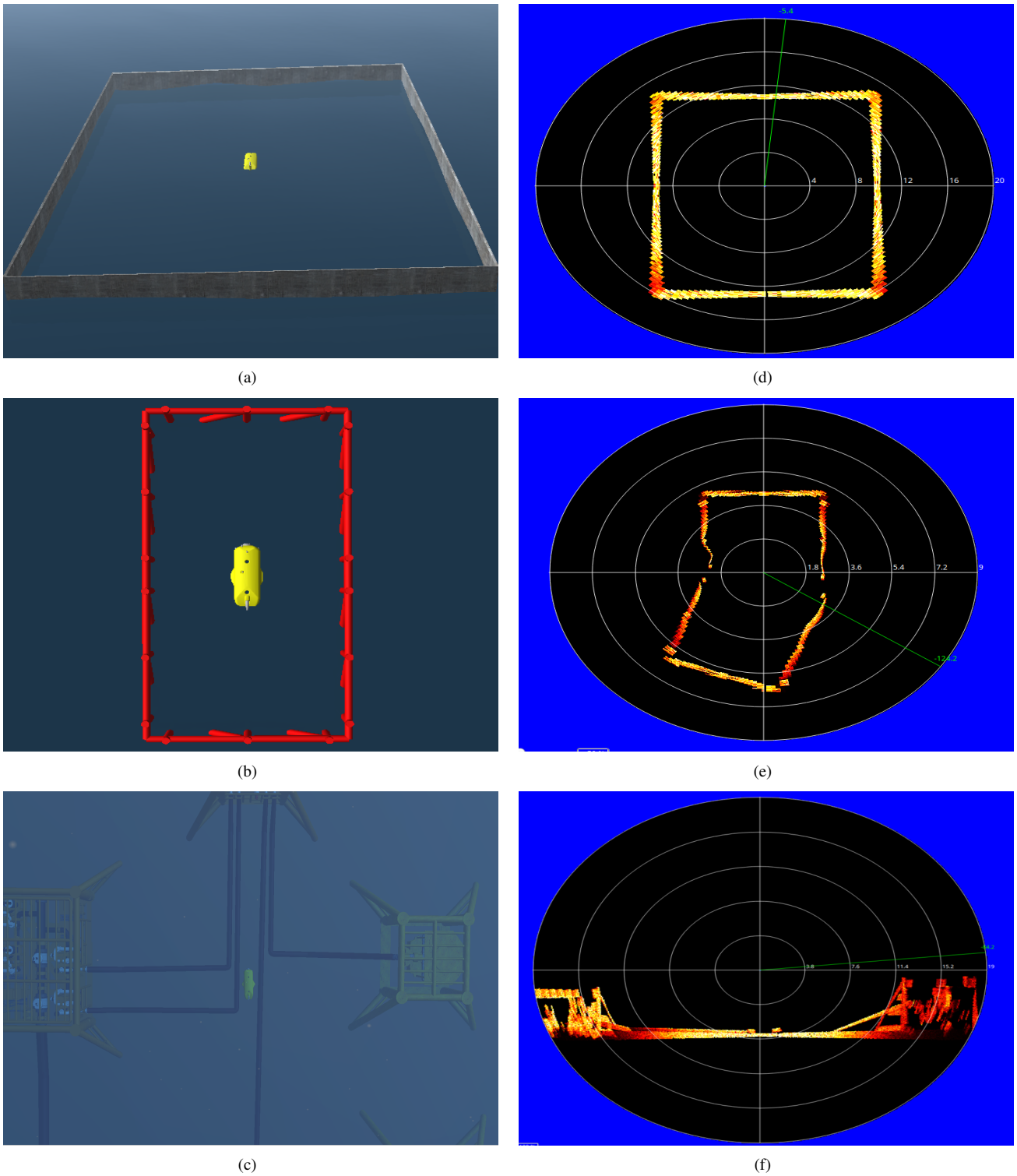


Figure 7: Forward-looking sonar simulation experiments: (a), (b) and (c) present the virtual underwater trials, while (d), (e) and (f) are the correspondent acoustic representations of each scenario, respectively.

7(e)); the frontal face of the target, as well the portion of the seabed and the degraded data by noise, are clearly visible in the FLS image; also, a long acoustic shadow is formed behind the manifold, occluding part of the scene. **The third scenario** contains a sub-sea isolation valve (SSIV) structure, connected to a pipeline in the bottom (see Fig. 7(c)); the simulated acoustic image, depicted in Fig. 7(f), also present shadows, material properties and speckle noise effects. Due to sensor configuration and robot position, the initial bins usually present a blind region in the three simulated scenes, caused by absence of ob-



(a) (d)

(b) (e)

(c) (f)

Figure 8: Experiments using mechanical scanning imaging sonar in three different scenarios (a), (b) and (c), and the respective processed simulated frames in horizontal orientation in (d) and (e), and vertical orientation in (f).

jects at lower ranges, similar to real sonar images. It is noteworthy that the brightness of sea-floor decreases as it is farther from sonar, because of the normal orientation of the surface.

The MSIS was also simulated in three different experiments. The robot in a big textured tank composes **the first scene** (see Fig. 8(a)); similar to the first scenario of FLS experiment, the reflectivity and texture were set to the target; the rotation of the sonar head position, by a complete 360° scanning, produced the acoustic frame of tank walls (see Fig. 8(d)). **The second scene** involves the vehicle's movement during the data acqui-

Table 2: Processing time to generate forward-looking sonar samples with different parameters.

# of samples	# of beams	# of bins	Field-of-view	Average time (ms)	Std dev (ms)	Frame rate (fps)
500	128	500	120° x 20°	54.7	3.7	18.3
500	128	1000	120° x 20°	72.3	8.9	13.8
500	256	500	120° x 20°	198.7	17.1	5.0
500	256	1000	120° x 20°	218.2	11.9	4.6
500	128	500	90° x 15°	77.4	11.8	12.9
500	128	1000	90° x 15°	94.6	10.2	10.6
500	256	500	90° x 15°	260.8	18.5	3.8
500	256	1000	90° x 15°	268.7	16.7	3.7

Table 3: Processing time to generate mechanical scanning imaging sonar samples with different parameters.

# of samples	# of bins	Field-of-view	Average time (ms)	Std dev (ms)	Frame rate (fps)
500	500	3° x 35°	8.8	0.7	113.4
500	1000	3° x 35°	34.5	1.6	29.0
500	500	2° x 20°	10.3	0.6	96.7
500	1000	2° x 20°	41.7	3.7	24.0

sition process; the scene contains a grid around the AUV (see Fig. 8(b)), captured by a front MSIS mounted horizontally; this trial induces a distortion in the final acoustic frame, because the relative sensor position with respect to the surrounding object changes, as the sonar image is being built (see Fig. 8(e)); in this case, the robot rotates 20° left during the scanning. **The last scene** presents the AUV over oil and gas structures on the sea bottom (see Fig. 8(c)); using an MSIS located in the back of the AUV with a vertical orientation, the scene was scanned to produce the acoustic visualization; as illustrated in Fig. 8(f), object surfaces present clear definition in the slice scanning of the sea-floor.

All the experimental scenarios was defined in order to provid enough variability of specific phenomena usually found in real sonar images, such as acoustic shadows, noise interference, surface irregularities and properties, distortion during the acquisition process and changes of acoustic intensities. However, the speckle noise application is restricted to regions with acoustic intensity, as shown in Figs. 7(f) and 8(d). This fact is due to our sonar model be multiplicative (defined in Eq. 2). In real sonar images, the noise also granulates the shadows and blind regions. The sonar simulator can be improved by inserting an additive noise to our model. The impact of incorporating additive noise on the image is more severe than that of multiplicative, and we decided to collect more data before including a specific additive noise in our simulator, at this moment. A second feature missing in our simulated acoustic images are the ghost effects caused by reverberation. This lacking part can be addressed by implementation of a multi-path propagation model [23], where the signal propagates along several different paths, resulting in fading and reverberation effects. Simulating the multi-path reflection is computationally costly, thus we need more time to modeling and including the reverberation phenomenon, to consider the real-time constraints.

4.2. Computational time

Performance evaluation of the simulator was assessed by considering the suitability to run real-time applications. The experiments were performed on a Intel Core i7 3540M processor, running at 3 GHz with 16GB DDR3 RAM memory and NVIDIA NVS 5200M video card, with Ubuntu 16.04 64 bits operating system. The elapsed time of each sonar data is stored to compute the average time, standard deviation and frame rate metrics, after 500 iterations. The results found is summarized in Tables 2 and 3. After changing the sonar rendering parameters, such as number of bins, number of beams and field-of-view, the proposed approach generated the sonar samples with a high frame rate, for both sonar types, in comparison to real-world sonars. For instance, 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; so, the obtained results allow the use of the sonar simulator for real-time applications. Also, the MSIS data used in the simulator is able to complete a 360° scan sufficiently fast in comparison with a real sonar as Tritech Micron DST. For the FLS device, these rates are superior to the rates lists by DeMarco *et al* [4] (330ms) and Saç *et al* [3] (2.5min). For MSIS type, to the best of our knowledge, there is no previous work for comparison.

According to previous results, since the number of bins is directly proportional to sonar image resolution, we can conclude that the number of bins used affects the computational time; when the number of bins increases, the simulator will have a bigger scene frame to compute and to generate the sonar data.

5. Conclusion and future work

A GPU-based simulator for imaging sonar simulation was presented here. The system is able to reproduce the operation mode of two different types of sonar devices (FLS and MSIS) in real-time. The real sonar image singularities, such as multiplicative noise, surface properties and acoustic shadows are addressed, and represented in the simulated frames. Specially for the shadows, the acoustic representation can present information as useful as the insonified object. Considering the qualitative results, the sonar simulator can be used by feature detection algorithms, based on corners, lines and shapes. Also, the computational time to build one sonar frame was calculated using different device settings. The vertex and fragment processing during the underwater scene rendering accelerates the rendered sonar image, providing an average time close or better than real-world imaging devices. These results allow the use of this imaging sonar simulator by real-time applications, such as obstacle detection and avoidance, and object tracking. We are working now on a way to add the reverberation effect to perform a more realistic sensing, without significantly affecting the computational time. We are also working on how to include an additive noise in the simulation of the acoustic images. Future works will focus on qualitative and time consuming comparison with other sonar simulators and with real acoustic images.

References

[1] Bell JM. Application of optical ray tracing techniques to the simulation of sonar images. Optical Engineering 1997;36(6):1806–13.

[2] Coiras E, Groen J. Simulation and 3d reconstruction of side-looking sonar images. In: Silva S, editor. Advances in Sonar Technology; chap. 1. In-Tech; 2009, p. 1–15.

[3] Saç H, Leblebicioğlu K, Bozdağı Akar G. 2d high-frequency forward-looking sonar simulator based on continuous surfaces approach. Turkish Journal of Electrical Engineering and Computer Sciences 2015;23(1):2289–303.

[4] DeMarco K, West M, Howard A. A computationally-efficient 2d imaging sonar model for underwater robotics simulations in Gazebo. In: MTS/IEEE OCEANS Conference. 2015, p. 1–8.

[5] Gu J, Joe H, Yu SC. Development of image sonar simulator for underwater object recognition. In: MTS/IEEE OCEANS Conference. 2013, p. 1–6.

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

[7] Guériot D, Sintes C. Forward looking sonar data simulation through tube tracing. In: MTS/IEEE OCEANS Conference. 2010, p. 1–6.

[8] Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, et al. ROS: an open-source robot operating system. In: Workshop on Open Source Software, held at IEEE International Conference on Robotics and Automation (ICRA). 2009, p. 1–6.

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

[10] Abbot J, Thurstone F. Acoustic speckle: theory and experimental analysis. Ultrasonic Imaging 1979;1(4):303–24.

[11] Ganesan V, Chitre M, Brekke E. Robust underwater obstacle detection and collision avoidance. Autonomous Robots 2015;40(7):1–21.

[12] Ribas D, Ridao P, Neira J. Underwater SLAM for structured environments using an imaging sonar. Springer-Verlag Berlin Heidelberg; 2010.

[13] Fallon MF, Folkesson J, McClelland H, Leonard JJ. Relocating underwater features autonomously using sonar-based SLAM. Journal of Ocean Engineering 2013;38(3):500–13.

[14] Liu L, Xu W, Bian H. A LBF-associated contour tracking method for underwater targets tracking. In: MTS/IEEE OCEANS Conference. 2016, p. 1–5.

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

[16] Bradski G. The opencv library. Doctor Dobbs Journal 2000;25(11):120–6.

[17] 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: IEEE Latin American Robotics Symposium (LARS). 2015, p. 132–8.

[18] SDF. `http://sdformat.org/`; 2017. Accessed: 2017-04-23.

[19] Soetens P, Bruyninckx H. Realtime hybrid task-based control for robots and machine tools. In: IEEE International Conference on Robotics and Automation (ICRA). 2005, p. 260–5.

[20] Rost RJ, Licea-Kane B, Ginsburg D, Kessenich JM, Lichtenbelt B, Malan H, et al. OpenGL shading language. 3rd ed.; Addison-Wesley Professional; 2009.

[21] Lee J. Digital image enhancement and noise filtering by use of local statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence 1980;2(2):165–8.

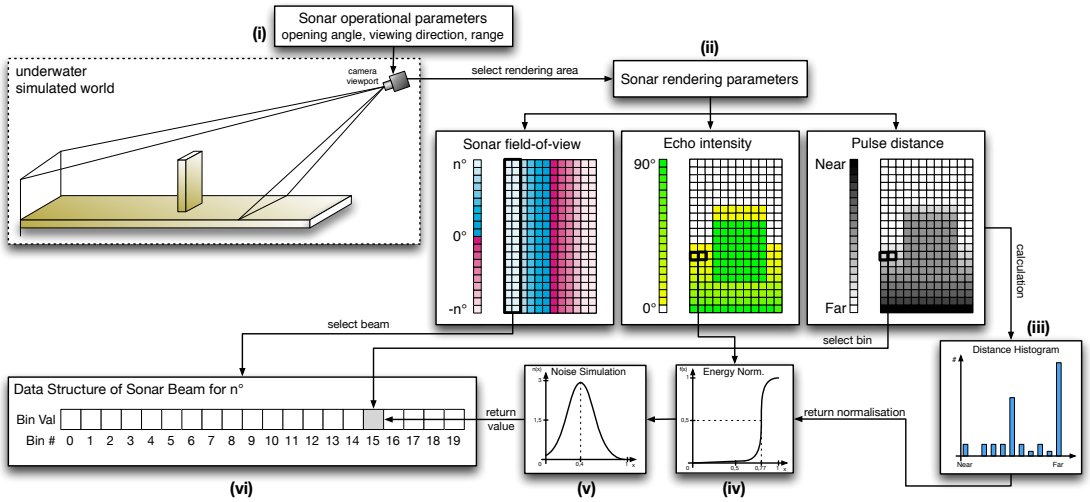
[22] Papoulis A, Pillai S. Probability, random variables and stochastic processes. McGraw Hill; 2002.

[23] Huang J. Simulation and modeling of underwater acoustic communication channels with wide band attenuation and ambient noise. Ph.D. thesis; Carleton University; 2015.

*Graphical Abstract (for review)

Abstract

Mainly when applied in the underwater environment, sonar simulation requires great computational effort due to the complexity of acoustic physics. For that, simulation of sonar operation allows to evaluate algorithms and control systems without going to the real underwater environment; that reduces the costs and risks of in-field experiments. This paper tackles with the problem of real-time underwater imaging sonar simulation by using the OpenGL shading language chain on GPU. This proposed system is able to simulate two main types of acoustic devices: mechanical scanning imaging sonars and forward-looking sonars. The underwater scenario simulation is performed based on three frameworks: (i) OpenSceneGraph reproduces the ocean visual effects, (ii) Gazebo deals with physical forces, and (iii) the Robot Construction Kit controls the sonar in underwater environments. Our system exploits the rasterization pipeline in order to simulate the sonar devices, which are rendered by three parameters: the pulse distance, the echo intensity and sonar field-of-view, being all calculated over objects shapes in the 3D rendered scene. Sonar-intrinsic operational parameters, speckle noise and object material properties are also considered as part of the acoustic image. Our evaluation demonstrated that the proposed method is able to operate close or faster than the real-world devices, as well as generating realistic sonar image quality in different virtual underwater scenarios.



Highlights

- A novel simulator for two types of sonars used in underwater applications
- For real-time applications rather than existing approaches
- Scenarios produced more realistically than other simulators

Response to the Reviewer's Comments

June 4, 2017

We thank the reviewers for their insightful and useful feedback. We have revised our paper in light of their comments. Please, find below our replies and comments about each of the issues raised, and how we modified the paper to address them. Beyond the suggested reviews, we also made changes in some parts of the paper to adhere to the suggestions, turning the paper more coherent and cohesive. To make clear where we modified, all changes in the text are in blue in this revised version.

>> REVIEWER #1 <<

> The proposed work simulates underwater imaging sonar based on OpenGL shading
> language (GLSL) chain, and is able to simulate two main types of sonar sensors:
> mechanical scanning imaging sonars (MSIS) and forward-looking sonars (FLS).
>
> The paper is well written with just small issues like "there is any previous
> work for comparison", "foward-looking sonars" or "the sonar simulator can be
> by feature" that needs to be easily solved.

All corrected. Not only the cited typos, but also other mistakes of the same kind in the revised version. Indeed, we carefully revised the whole text.

> The contribution of the work is interesting for the community, mainly if the
> simulator will become freely available. It can increase the potential of the
> work in terms of impact.

Certainly, on the acceptance of the paper, we will make the simulator publicly available.

> About the work, Figure 3 is confused and needs to be improved. The authors
> argue that the beam are composed by the intensity, depth and angular
> distortion matrix, however it is not clean in the figure. The names makes
> the things hard, please "uniform the names", like "Beam Angle in Camera"
> = "Angular Distortion"? "Surface Angle to Camera" = "Intensity"? "Distance
> from Camera" = "Depth" ?

Done. We improved the terms in the cited figure (now Fig. 4 in the revised paper) to fit to the text. We also completely reformulated the text referring to Fig. 4 to clearly explain in a higher level of understanding.

> In Fig. 6, it is not clear the effect of the parameter "p". How does the
> normal (blue channel) is adopted in the calculus of the acoustic intensity?
> It is more readable to show the final simulation instead of the normal
> image vs. depth image (green channel), as in Fig. 5.

To find the final pixel intensity, we multiply the surface normal angle by the material reflectance ρ. This operation takes place in fragment shader. We rewrote the sentences in section 3.2 to make the idea clearer. Also, Figure 6 was modified to take this consideration.

> About the simulation resolution, the number of bins are dependent of the
> sonar's frequency and the adopted range. The authors know it, as shown in line
> 266, however it could be interesting to define the number of bins in terms of
> these two parameters to make the simulation "more realistic" in terms of
> the real sensor. It could be interesting for the community.

The simulator itself is as freely parameterizable as possible to allow for a wide range of imaging sonars to be simulated. The definition of the number of bins, ranges etc. is done for each individual simulated sonar in the corresponding component, which injects the sonar data into the full simulation. For instance, the simulated FLS and MSIS devices, presented in Section 4, have the exact same parameters and operation modes as the real sonars (Tritech Gemini 720i and Tritech Micron DST). From the point of view of the robot control, the system can be considered very realistic.

> Another interesting thing is related to the noise simulation. It does not appear
> realistic due to the lack of "noise" in the black areas. The application of
> noise in this area can be interesting.

This issue is explained in subsection 4.1. As we described in subsection 3.3.1, the speckle noise modeled in this paper is a multiplicative one, following a non-uniform distribution. The resulting sonar data is composed by an element-wise multiplication between the raw data and the speckle noise. The insertion of additive noise is already addressed as future work to solve this missing part.

> The main limitation of the work is the lacks of reverberation simulation in the
> work. It limits the applicability of the simulation to open waters with a small
> number of objects. Thus, it appears to be the "main future works", however, the
> authors do not mention it.

We have chosen to extend the underwater acoustic phenomena in the simulator, obeying the use in real-time applications. Processing the reverberation in a sonar simulator is computationally costly, then its addition must consider the time consuming again. We took this problem into account and addressed it as future work in the Section 5, as suggested.

> It is hard to validate a simulator, however, I believe the works lake in
> terms of comparison with real data at least in terms of SNR or other metrics.
> It could be interesting to simulate a "real scenario" and compare the results
> obtained by the simulator.

Just now, we had the opportunity to record data with a real imaging sonar of an object which we can fully simulate (see Figs. 7(c) and 7(f) in the revised paper and Fig. 1, in this document). Therefore we plan to use this data next months to analyze the quality of the simulation and publish this later, when the results are satisfied.

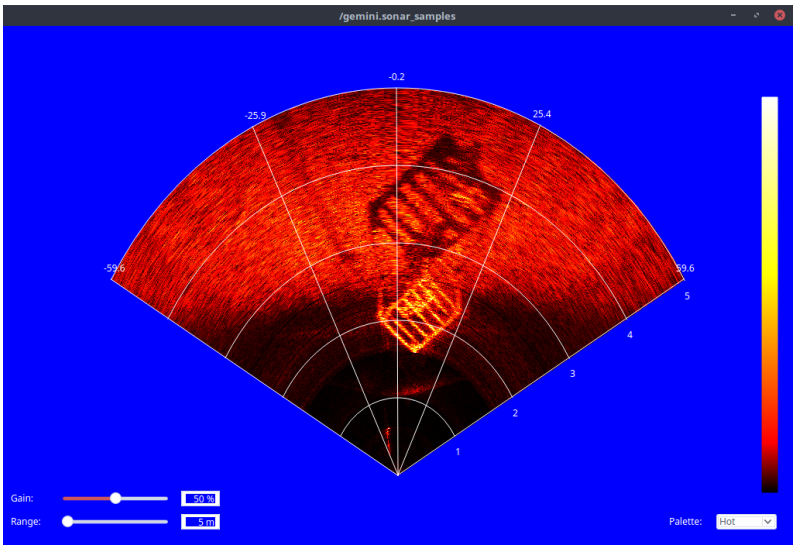


Figure 1: Image of a Tritech Gemini 720i Imaging Sonar of a mock-up of a sub-sea structure.

>> REVIEWER #2 <<

> The Euclidean distance from camera center, the surface normal angles, and the
> angular distortion are recorded as color channels. In this case, there is a
> limitation of 256 values. Does this brings precision problems? Using CUDA or
> OpenCL, couldn't this be addressed in a more elegant way?

The 8-bit channel representation only causes precision problems with the pulse distance parameter. In this case, we replaced the original information by the native GLSL depth buffer, which has 32-bit floating-point values. In fact, this information was not addressed in the first version of the paper. We clarified this point in the revised version.

Indeed, CUDA and OpenCL are both good alternatives to implement the proposed approach. However, we have chosen to use GLSL for three main reasons: 1) native from OpenGL, avoiding to install additional packages; 2) hardware and backward compatibility; 3) use of precomputed geometric information during the rasterization process.

> I would like to see more details on the implementations of the models. I am not
> sure if it is possible to reproduce the solution with the presented text.

We added as much details as possible in this revised version. Conditioned to the acceptance of the paper, the simulator will be available under an open source license and may be downloaded from Github.

> This sentence seems to be from a decade ago paper...: Modern graphics hardware
> presents programmable tasks...

This sentence was removed in our revised manuscript.

> It is unclear to me, but each beam is for a complete column?

Each beam is composed by one or more columns, according to sonar bearings. We took this point into consideration, rewriting the text to make it clearer.

>> REVIEWER #3 <<

> The present work proposes a method for the simulation of sonar sensors.
> Differently from previous methods, the proposed technique takes advantage
> of the GPU to achieve realtime performance and is able to reproduce the
> operation of FLS and MSIS sonars sensors.
>
> Text
> ====
>
> The text contains some confusing passages that demand review. For instance:
>
> --- lines 78-82
> "The intensity measured back from the in-
> sonified objects depends on the accumulated energy based on
> surface normal directions, producing more realistic simulated
> scenes, instead of statically defined by the user, as in [5], or in
> a binary representation as found in [6, 7]."
>
> ---> "accumulated energy" refers to what exactly? Energy accumulated where?
> ---> "instead of statically defined by the user" What does it mean?

We rewrote this paragraph to clarify these issues.

> --- lines 172-174
> "The Rock-Gazebo integration [17] provides the underwa-
> ter scenario, and allows real-time hardware-in-the-loop simula-
> tions."

> ---> What is "Rock-Gazebo integration"? The authors explain it later. However,
> the order of the sentences could be switched to make text a bit clearer.

This sentence was rewritten in the revised version to explain the necessity of the integration of the sonar simulation into a framework, and then the connection to Rock-Gazebo was made. That hopefully makes this paragraph less confusing.

> The Introduction section contains some text that should be in the Related
> Work Section.

We decided to include the related work continuously inside the introduction without a dedicated subsection. We believe that it makes the text more uniform and fluid in the introduction.

> Technically, bump maps are different from normal maps. Bump maps are defined as
> height maps that describes the bumps on parametric surfaces. Normal maps, on the
> other hand, are textures that contain perturbed normal vectors. There are both
> tangential and object space normal maps. It seems that the proposed technique
> uses normal maps. I suggest a review on that.

In fact, the proposed approach uses normal maps to simulate geometric surface irregularities by passing RGB textures and modifying the normal vectors. We revised the manuscript to correct this.

> Some image labels are too small.

The paper have followed the LaTeX template provided by the journal. It is not possible to modify this point.

> No need to put "the {first|second|third|last} {scenario|scene}" in bold on
> lines 322, 327, 334, 343, 347-348, 354-355.

The rationale in putting the sentences in bold was to highlight the beginning of the explanations about each scene/scenario. This helps the reader to follow our description.

> Contributions and limitations
> ============================
>
> In the abstract the authors state the importance of simulating sonar sensors
> and then introduce the proposed method. However, nothing is said
> about the difficulties involved in such a simulation and about the general
> limitations of current methods. Along the text, it seems that the use of the GPU
> for sonar sensor simulation is a contribution, but this is not mentioned in the
> abstract.

We added the use of GPU in the abstract. In fact, we completely revised the abstract to fit to the suggested changes.

> --- lines 69-73
> "This paper introduces a novel imaging sonar simulator, that
> can overcome the main limitations of the existing approaches.
> As opposed to [1, 2, 3, 4, 5, 6, 7], where the proposed models
> simulate a specific sonar type, our model is able to reproduce
> two kind of sonar devices (...)"
>
> ---> These sentences may lead the reader to wrongly interpret the existing
> techniques, that simulate only one type of sensor, as being limited although
> they could even do it very accurately. Actually, it can be (and probably will
> be) the case that the authors of the existing techniques were not concerned
> about simulating several sonar sensor types. I think that this statements should
> be reviewed and rewritten in order to correctly account for that.

We have rewritten the sentences and the paragraph to avoid such confusion and make them clearer.

> ---> The authors claim that the proposed technique overcomes the "main
> limitations" of existing techniques. However, the proposed method itself
> introduces limitations that were not present in some of the existing techniques
> (for instance, the proposed method does not handle reverberation, an important
> phenomenon in the context of sonar sensor simulation). Thus, I think that the
> discourse about contributions versus limitations should be reviewed in order
> to be fair.

We addressed your consideration in the text and review the subsection 1.1 to avoid such confusion in the revised paper.

> The method
> ==========
>
> The explanation about sonar sensors, and how they work, seems to be Ok. However,
> I've missed a formalization of the problem being simulated. Without it, it
> becomes difficult to understand how good the proposed simulation method is
> (which were the simplifying assumptions adopted?). Also, the description of
> the method implementation is quite confusing. I think that a grad student
> could not implement it without too much guessing.

We tried to incorporate all your suggestions and hope to clear out our description of the method.

```
> --- lines 74-78
> "Also, the underwater scene is processed during the pipeline rendering on
> graphics processing unit (GPU), accelerating the simulation process,
> guaranteeing real-time simulation, in contrast to the methods found in
> [1, 2, 4, 5]."
>
> ---> It is not clear why the GPU/rasterization were chosen and how the GPU
> helps in overcoming eventual problems/limitations of the existing techniques.
> Why GPU-based realtime ray tracing was not chosen, for instance?
```

The rasterization was chosen because the geometric data from virtual scene are computed on optimized pipeline GPU embedded. With that, we handled with these precomputed data to get the pulse distance, echo intensity and sonar field-of-view parameters needed to build the acoustic frame (these terms were renamed from distance/depth, angular distortion and intensity in the first version of the paper). In addition, we have reviewed the subsection 1.1 to include this information.

```
> The method takes advantage of the rasterization pipeline in order to build
> the distance, normal and angle maps. This is not stated anywhere, but that
> rendering method is similar to the "deferred shading" (which solves the
> visibility from the camera viewpoint at the same time that stores several
> properties of the rasterized fragments into an auxiliar buffer). This would
> be one advantage related to the use of GPU in this context.
>
> From Section 2 (lines 104-107) it seems that bins are samples placed along the
> beam. However, I could not spot how these samples are obtained from the
> description in Section 3.3. Actually the text explains it, but I've found it too
> confusing. In the beginning I had the impression that the distance, angle and
> normal maps were 2D. Later on, I started thinking that they should be 3D
> (what does "3D shader matrix" mean in line 257?). However, if they are really
> 3D maps, when is each slice of this 3D map (matrix) obtained? I've got really
> confused at this point.
```

The "3D shader matrix" is a 3-channel matrix which stores the sonar rendering parameters of pulse distance, echo intensity and sonar field-of-view from the captured OSG scene. We have rewritten this term on paper to make it clear.

```
> I've found the noise model too arbitrary. How good is this approximation?
```

The speckle noise n is a bell curve, following a Gaussian non-uniform distribution:

$$n(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2},$$

given mean μ and standard deviation σ. In the proposed approach, we defined $\mu = 0.4$ and $\sigma = 0.15$, and the modeled curve is presented in Fig. 4(v) in the paper.

```
> Results
> =======
>
> --- lines 86-88
> "The main goal here is to build quality and low time-con-
> suming acoustic frames, according to underwater sonar image
```

> formation and operation modes (see Section 2)."
>
> ---> The Results Section does not compare the obtained results with data
> obtained with real sensors (validation). The proposed method is not even
> compared to existing techniques (except for speed in some cases). Thus, despite
> the fact that the proposed method can very efficiently simulate two distinct
> types of sonar sensors, it is not possible to assert how good the results are
> (quality).

With regards to other simulation systems, we have the problem that the freely available simulators either don't fit our targeted niche of having a real-time simulation for robots or are bound to specific sonar they simulate. This is the one of the reasons we actually started our development. Real sonar data validation is incredible hard, since there are a lot of parameters to consider. However we will conduct a qualitative comparison using data we were able to record very recently (see Fig. 1 in this document).

> --- lines 370-373
> "In real sonar images, the noise also granulates the shadows and blind regions.
> The sonar simulator can be improved by inserting an additive noise to our
> model."
>
> ---> This seems to be a feature that could be easily included. Why it was not
> added to the system?

Additive noise has a far more sever impact on the image and therefore on any recognition algorithm running on that image. So it is very important not to divert too far from the real world, which is why we e.g. haven't just added a Gaussian additive noise. There is an ongoing work to integrate it, but we are not finished. We added a short sentence in the text explaining this situation.

> Here I make an observation regarding the proposed technique. As was
> mentioned in the paper, several existing techniques make use of the ray tracing
> in order to simulate sonar sensors because reflection is extremely
> important in the context of sound propagation simulation and it
> can be easily simulated with ray tracing. Thus, how do the authors
> plan to efficiently expand their rasterization-based system in order to handle
> reflections? Wouldn't it be easier and more efficient with GPU-accelerated ray
> tracing?

Good point. We have decided to extend the physical phenomena in the simulator obeying the use of real-time conditions. Aware of that, our approach is optimized by taking precomputed geometric data during the rasterization pipeline on GPU to compute the acoustic frame, instead of simulating sound pulse propagation with ray-tracing approach. Computing the multi-path reflection in sonar simulation is computationally costly, thus we need to consider the time consuming again by modeling and including phenomena such as reverberation. This will be made in the future.

> --- lines 373-375
> "This lacking part can be addressed by implementation of a multi-path propaga-
> tion model."
>
> ---> What does it mean "multi-path"? References?

Multipath means that a signal propagates between transmitter and receiver along several different paths, resulting in fading and reverberation effects. We included the reference [22] in the paper, explaining it better.

```
> Decision
> ========
>
> The proposed work is very interesting but seems to be just in early stages
> of development. Some decisions were not very well justified (why GPUs? why
> rasterization?). Some explanations are confusing (the simulation of the sonars).
> Certainly, with some additional work on these points, the paper will be ready for
> submissions. However, in the current state, and from what was already discussed,
> I've decided for its rejection.
```

Many thanks for your feedback. We have addressed your comments and the other reviewers' comments in the revised manuscript, and we hope to have your expectations satisfied this time.

```
>> REVIEWER #4 <<

> The submitted manuscript proposes a sonar simulator for real-time
> applications. Several physical aspects are considered in a computational high
> performance implementation. The final product is impressive.
>
> With respect to the document, it is very well-written and organized, theoretically
> well supported, the presentation is clear and the experiments are sufficient
> to prove the system effectiveness. Everything is well justified and I do not
> have any comment or question about the work. Congratulations.
>
> Under these conditions, I believe the paper is ready to be published.
```

We thank for your feedback. We have addressed the other reviewers' comments in the revised version, and we hope your expectations continue to be satisfied.