

Manuscript Number:

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:** Sonar simulation requires great computational effort due to the complexity of acoustic physics related to the underwater environment. This fact turns the challenge of reproducing sensor data into a non-trivial task. On the other hand, simulation of sonar operation data 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 proposes a novel real-time underwater imaging sonar simulator, which uses the 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 virtual underwater simulation was conceived based on three frameworks: (i) OpenSceneGraph (OSG) reproduces the ocean visual effects, (ii) Gazebo deals with physics effects, and (iii) the Robot Construction Kit (Rock) controls the sonar in underwater environments. The proposed sonar simulation returns a matrix comprised of the echo intensity, the distance to the target object and the angle distortion, being all calculated over object shapes and material properties in the 3D rendered scene. Sonar-intrinsic speckle noise and object material properties are also considered as part of the sonar image. Our evaluation demonstrated that the proposed method is able to operate with high frame rate, as well as 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

Sonar simulation requires great computational effort due to the complexity of acoustic physics related to the underwater environment. This fact turns the challenge of reproducing sensor data into a non-trivial task. On the other hand, simulation of sonar operation data 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 proposes a novel real-time underwater imaging sonar simulator, which uses the 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 virtual underwater simulation was conceived based on three frameworks: (i) OpenSceneGraph (OSG) reproduces the ocean visual effects, (ii) Gazebo deals with physics effects, and (iii) the Robot Construction Kit (Rock) controls the sonar in underwater environments. The proposed sonar simulation returns a matrix comprised of the echo intensity, the distance to the target object and the angle distortion, being all calculated over object shapes and material properties in the 3D rendered scene. Sonar-intrinsic speckle noise and object material properties are also considered as part of the sonar image. Our evaluation demonstrated that the proposed method is able to operate with high frame rate, as well as 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

1     Simulation is an useful tool for designing and programming  
2 autonomous robot systems. That allows evaluating robot behav-  
3 ior, without dealing with physical hardware or decision-making  
4 algorithms and control systems in real-time trials, as well as  
5 costly and time-consuming field experiments.

6     When working with autonomous underwater vehicles (AUVs),  
7 simulation of facilities are specially relevant. AUVs usually de-  
8 mand expensive hardware and perform long-term data gather-  
9 ing operations, taking place in restrictive sites. As AUV does  
10 not need umbilical cable, and the underwater communication  
11 carries on by unreliable acoustic links, the robot should be able  
12 to make completely autonomous decisions, even with low-to-  
13 zero external assistance. While the analysis and interpretation  
14 of sensor data can be performed in a post-processing step, a  
15 real-time simulation is strongly needed for testing and evalua-  
16 tion of vehicle's motion responses, avoiding involved risks on  
17 real world rides.

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

30     By considering sonar benefits and singularities along with

32 the need to evaluate AUVs, recent works proposed ray tracing-  
33 and tube tracing-based techniques to simulate acoustic data with  
34 very accurate results, although having a high computational  
35 cost [1, 2, 3, 4, 5, 6, 7]. Bell [1] proposed a simulator based  
36 on optical ray tracing for underwater side-scan sonar imagery;  
37 images were generated by acoustic signals represented by rays,  
38 which are repeatedly processed, forming a 2D-array. Coiras and  
39 Groen [2] used frequency-domain signal processing to produce  
40 synthetic aperture sonar frames; in that method, the acoustic  
41 image was created by computing the Fourier transform of the  
42 acoustic pulse used to insonify the scene. For forward-looking  
43 sonar simulations, Guériot and Sintes [3] introduce a volume-  
44 based approach of energy interacting with the scene, and col-  
45 lected by the receiving sonar; the sound propagation is defined  
46 by series of acoustic tubes, being always orthogonal to the cur-  
47 rent sonar view, where the reverberation and objects' surface ir-  
48 regularities are also addressed. Saç *et al.* [4] described a sonar  
49 model by computing the ray tracing in frequency domain; when  
50 a ray hits an object in 3D space, three parameters are calcu-  
51 lated to process the acoustic data: the Euclidean distance from  
52 the sonar axis, the intensity of returned signal by Lambert Illu-  
53 mination model and the surface normal; the reverberation and  
54 shadow phenomena are also considered in the scene rendering.  
55 DeMarco *et al.* [5] used Gazebo and Robot Operating System  
56 (ROS) [8] integration to simulate the acoustic sound pulses by  
57 a ray tracing technique, and to produce a 3D point cloud of the  
58 covered area; since the material reflectivity was statically de-  
59 fined, the final sonar image presented the same intensity values  
60 for all points on a single object. Gu *et al.* [6] modeled a forward-  
61 looking device where the ultrasound beams were formed by a  
62 set of rays; the acoustic image is significantly limited by its rep-  
63 resentation using only two colors: white, when the ray strikes

64 an object, and black for shadow areas. Kwak *et al.* [7] evolved  
 65 the previous approach by adding a sound pressure attenuation  
 66 to produce the gray-scale sonar frame, while the other physical  
 67 characteristics related to sound transmission are disregarded.

### 68 1.1. Contributions

69 This paper introduces a novel imaging sonar simulator, that  
 70 can overcome the main limitations of the existing approaches.  
 71 As opposed to [1, 2, 3, 4, 5, 6, 7], where the proposed models  
 72 simulate a specific sonar type, our model is able to reproduce  
 73 two kind of sonar devices: mechanical scanning imaging sonar  
 74 (MSIS) and forward-looking sonar (FLS). Also, the underwa-  
 75 ter scene is processed during the pipeline rendering on graph-  
 76 ics processing unit (GPU), accelerating the simulation process,  
 77 guaranteeing real-time simulation, in contrast to the methods  
 78 found in [1, 2, 4, 5]. The intensity measured back from the in-  
 79 sonified objects depends on the accumulated energy based on  
 80 surface normal directions, producing more realistic simulated  
 81 scenes, instead of statically defined by the user, as in [5], or in  
 82 a binary representation as found in [6, 7]. The speckle noise  
 83 is modeled as a non-uniform Gaussian distribution and added  
 84 to the final sonar image, performing a more realistic sensing,  
 85 differently to [3, 4, 6, 7].

86 The main goal here is to build quality and low time-con-  
 87 suming acoustic frames, according to underwater sonar image  
 88 formation and operation modes (see Section 2). The shader ma-  
 89 trix with depth and normal buffers, and angular distortion values  
 90 are extracted from underwater scene, and subsequently fused to  
 91 generate the simulated sonar data, as described in Section 3.  
 92 Qualitative and time evaluation results for two different sonar  
 93 devices are presented in Section 4, and allow for the use of the  
 94 proposed simulator in real-time applications.

## 95 2. Imaging Sonar operation

96 Sonars are echo-ranging devices that use acoustic energy to  
 97 locate and survey objects in a desired area. The sonar trans-  
 98 ducer emits pulses of sound waves (or ping) until they hit with  
 99 any object or be completely absorbed. When the acoustic sig-  
 100 nal collides with a surface, part of this energy is reflected, while  
 101 other is refracted. The sonar data is built by plotting the echo  
 102 measured back versus time of acoustic signal. The transducer  
 103 reading in a given direction forms a *beam*. A single beam trans-  
 104 mitted from a sonar is illustrated in Fig. 1. The horizontal and  
 105 vertical beamwidths are represented by the azimuth  $\psi$  and el-  
 106 evation  $\theta$  angles, respectively, where each sampling along the  
 107 beam is named as *bin*. The sonar covered area is defined by  
 108  $R_{min}$  and  $R_{max}$ . Since the speed of sound underwater is known,  
 109 or can be measured, the time delay between the emitted pulses  
 110 and their echoes reveal how far the objects are (distance  $r$ ), as  
 111 well as how fast they are moving. The backscattered acoustic  
 112 power in each bin determines the intensity value.

113 With different azimuth directions, the array of transducer  
 114 readings forms the final sonar image. Since all incoming sig-  
 115 nals converge to the same point, the reflected echoes could have  
 116 been originated anywhere along the corresponding elevation arc

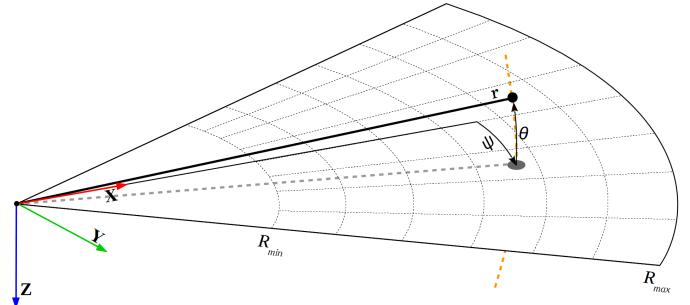


Figure 1: Imaging sonar geometry. By the projection process, all 3D points belonging to the same elevation arc (represented as dashed red line) will be represented to the same image point in the 2D plane. This way, range  $r$  and azimuth angle  $\psi$  are measured, and elevation angle  $\theta$  is lost. The sonar covered area is defined by  $R_{min}$  and  $R_{max}$ .

117 at a fixed range, as depicted in Fig. 1. In the acoustic represen-  
 118 119 tation, the 3D information is lost in the projection into a 2D

### 120 2.1. Sonar characteristics

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

- 124 (a) **Shadowing:** This effect is caused by objects blocking the  
 125 sound waves transmission and causing regions behind, with-  
 126 out acoustic feedback. These regions are defined by a  
 127 black spot in the image occluding part of the scene;
- 128 (b) **Non-uniform resolution:** The amount of pixels used to  
 129 represent an intensity record in the cartesian coordinate  
 130 system grows as its range increases. This fact causes im-  
 131 age distortions and object flatness;
- 132 (c) **Changes in viewpoint:** Imaging the same scene from dif-  
 133 ferent viewpoints can cause occlusions, shadows move-  
 134 ments and significant alterations of observable objects [9].  
 135 For instance, when an outstanding object is insonified, its  
 136 shadow is shorter, as the sonar becomes closer;
- 137 (d) **Low signal-to-noise ratio (SNR):** The sonar suffers from  
 138 low SNR mainly due the very-long-range scanning, and  
 139 the presence of speckle noise introduced caused by acous-  
 140 tic wave interferences [10];
- 141 (e) **Reverberation:** This phenomena is caused when multiple  
 142 acoustic waves returning from the same object are detected  
 143 over the same ping, producing duplicated objects.

### 144 2.2. Types of underwater sonar devices

145 The most common types of underwater acoustic sonars are  
 146 MSIS and FLS. In the former, the sonar image is built for each  
 147 pulse, with one beam per reading (see Fig. 2(a)); these sonar  
 148 images are usually shown on a display pulse by pulse, and the  
 149 head position reader is rotated according to motor step angle.  
 150 After a full  $360^\circ$  sector reading (or the desired sector defined  
 151 by left and right limit angles), the accumulated sonar data is  
 152 overwritten. The acquisition of a scanning image involves a rel-  
 153 atively long time, introducing distortions caused by the vehicle

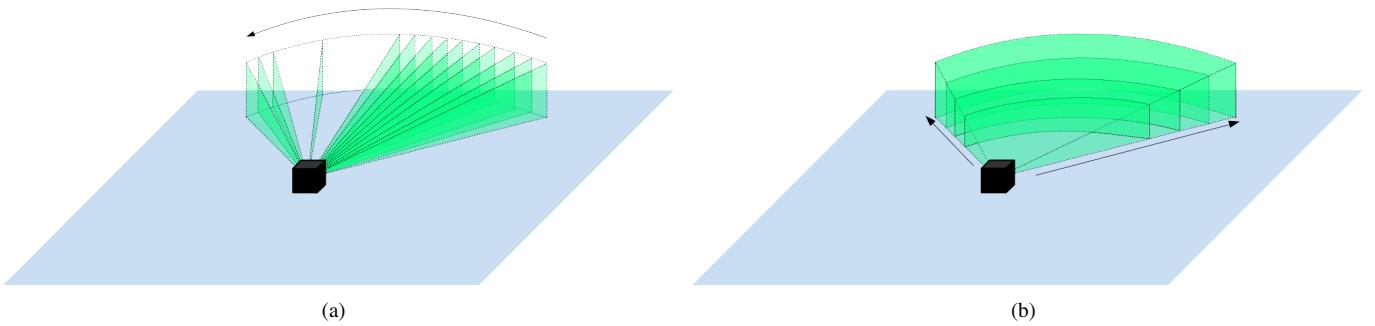


Figure 2: Different underwater sonar readings: (a) From a mechanical scanning imaging sonar and (b) from a forward-looking sonar.

154 movements. This sonar device is generally applied in obstacle  
155 avoidance [11] and navigation [12] applications.

156 As illustrated in Fig. 2(b), the whole forward view of an  
157 FLS is scanned and the current data is overwritten by the next  
158 one in a high frame rate, with  $n$  beams being read simultane-  
159 ously. This is similar to a streaming video imagery for real-  
160 time applications. This imaging sonar is commonly used for  
161 navigation [13], mosaicing [9], target tracking [14] and 3D re-  
162 construction [15].

### 163 3. GPU-based sonar simulation

164 The goal of this work is to simulate two types of underwa-  
165 ter sonar by vertex and fragment processing, with a low com-  
166 putational cost. The complete pipeline of the proposed simu-  
167 lator, from the virtual scene to the simulated acoustic data, is  
168 seen in Fig. 3 and is detailed in the following subsections. The  
169 sonar simulation is written in C++ with OpenCV [16] support  
170 as Rock packages.

#### 171 3.1. Rendering underwater scene

172 The Rock-Gazebo integration [17] provides the underwa-  
173 ter scenario, and allows real-time hardware-in-the-loop simula-  
174 tions. In this framework, Gazebo handles the physical engines,  
175 and the Rock’s visualization tools are responsible by the scene  
176 rendering. The graphical data in Rock are based on OpenScene-  
177 Graph framework, an open source C/C++ 3D graphics toolkit  
178 built on OpenGL. The osgOcean framework is used to simulate  
179 the ocean visual effects, and the ocean buoyancy is defined by  
180 the Gazebo, as described in [17].

181 All scene aspects, such as world model, robot parts (includ-  
182 ing sensors and joints) and other objects presented in the en-  
183 vironment are defined by simulation description files (SDF),  
184 which use the SDFormat [18], an XML format used to de-  
185 scribe simulated models and environments for Gazebo. Vi-  
186 sual and collision geometries of vehicle and sensor robot are  
187 also described in specific file formats. Each component de-  
188 scribed in the SDF file becomes a Rock component, which is  
189 based on the Orocros real-time toolkit (RTT) [19], providing  
190 I/O ports, properties and operations as communication layers.  
191 When the models are loaded, Rock-Gazebo creates I/O ports  
192 to allow real-world or simulated system components interact-  
193 ing with the simulated models. A resulting scene sample of this  
194 integration is seen in Fig. 4.

#### 195 3.2. Shader rendering

196 Modern graphics hardware presents programmable tasks em-  
197 bedded in GPU. Based on parallel computing, that approach  
198 can speed up 3D graphics processing, reducing the computa-  
199 tional effort of the central processing unit (CPU). The render-  
200 ing pipeline can be customized by defining programs on GPU  
201 called shaders. A shader is written in OpenGL Shading Lan-  
202 guage (GLSL) [20], a high-level language with a C-based syn-  
203 tax which enables more direct control of graphics pipeline, which  
204 avoids the use of low-level or hardware-specific languages. Shaders  
205 can describe the characteristics of either a vertex or a fragment  
206 (a single pixel). Vertex shaders are responsible by transforming  
207 the vertex position into a screen position by the rasterizer, gen-  
208 erating texture coordinates for texturing, and lighting the vertex  
209 to determine each color. The rasterization results in a set of pix-  
210 els to be processed by fragment shaders, which manipulate their  
211 locations, depth and alpha values, and interpolated parameters  
212 from the previous stages, such as colors and textures.

213 In our work, the underwater scene is sampled by a virtual  
214 camera, whose optical axis is aligned with the intended viewing  
215 direction of the imaging sonar device, as well as the covered  
216 range and opening angle. By programming the fragment and  
217 vertex shaders, the sonar data is computed as:

- 218 (a) **Depth** is the camera focal length, calculated by the Eu-  
219 clidean distance to the object surface point;
- 220 (a) **Intensity** presents the echo reflection energy based on  
221 object surface normal angle up to the camera;
- 222 (a) **Angular distortion** is the angle formed from the camera  
223 center column up to the camera boundary column, for  
224 both directions.

225 All those three informations are normalized into the interval  
226  $[0,1]$ , where zero means no energy and one means maximum  
227 echo energy for intensity data, respectively. For depth data,  
228 the minimum value portrays a close object while the maximum  
229 value represents a far one, limited by the sonar maximum range.  
230 Angle distortion value is zero in image center column which  
231 increases for both borders to present the half value of horizontal  
232 field of view.

233 In realistic sensing, real-world surfaces usually present ir-  
234 regularities and different reflectance values. To render that in

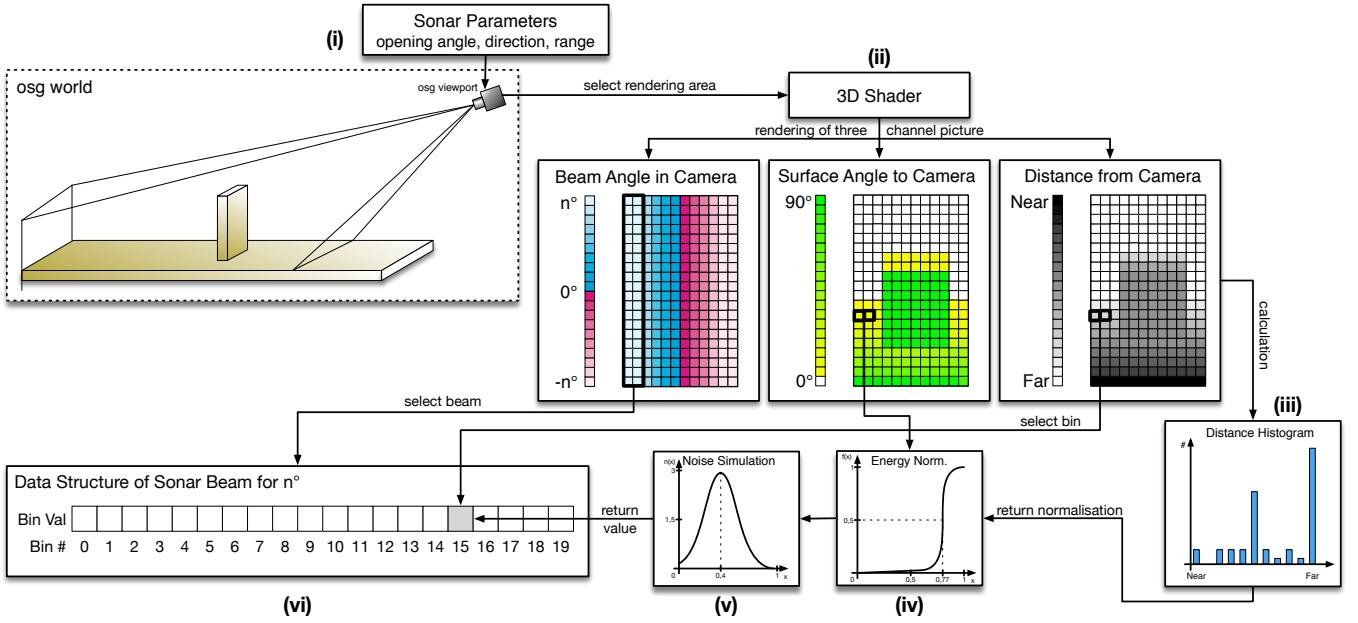


Figure 3: A graphical overview of the imaging sonar simulation process: (i) a virtual camera, specialized as the sonar device, samples the underwater scene; (ii) three components are calculated by shader rendering on GPU: The Euclidean distance from camera center, the surface normal angles, and the angular distortion; the shader information is splitted in beam parts, according the angular distortion values, and bin depth and intensity are defined by distance histogram (iii) and energy normalization (iv); (v) the speckle noise is added to the final sonar data; (vi) the simulated data is presented as Rock's data type.

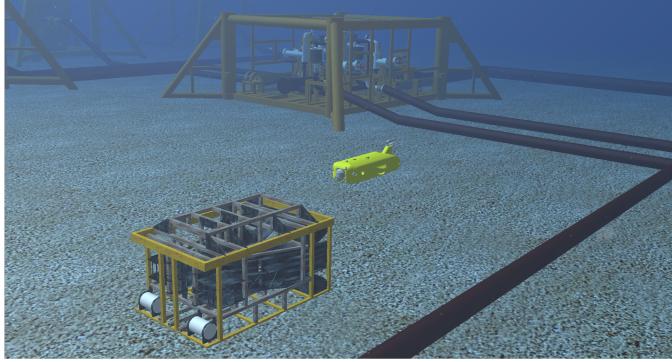


Figure 4: The AUV in Rock-Gazebo underwater scene.

a virtual scene, the intensity values can also be defined by a bump mapping and material property information. Bump mapping is a perturbation rendering technique to simulate wrinkles on the object surface by passing textures and modifying the normal directions on shaders. This approach consumes less computational resources for the same level of detail, compared to the displacement mapping technique, because the geometry remains unchanged. Since bump maps are built in tangent space, interpolating the normal vertex and the texture, a tangent, bitangent and normal (TBN) matrices are computed to convert the normal values into the world space. Representation of bump mapping process in our approach is seen in Fig. 5.

The reflectance allows to properly describe the intensity received back from observable objects in shader processing, according their material properties. For instance, aluminum has more reflectance than wood and plastic. When an object has its reflectivity defined, the reflectance value  $\rho$  is passed to frag-

ment shader and must be positive. As seen in Fig. 6, normal values are directly proportional to the reflectance value  $\rho$ . At the end, the shader process provides a matrix with three main data: Intensity, depth and angular distortion.

### 3.3. Simulating sonar device

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 a number of beam parts. Each column is correlated with its respective beam, according to sonar bearings, as seen in Fig. 3. Each beamed sub-image is converted into bin intensities using the depth and intensity data. 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 are the farthest ones. Therefore a distance histogram is evaluated to group the sub-image pixels with their respective bins, according to the depth value. This information is used to calculate the accumulated intensity of each bin.

Due to the acoustic beam spreading and absorption in the water, the final bins have less echo strength than the first ones, because the energy is lost twice, in the environment. To tackle with that issue, the 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 in each bin is normalized as

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

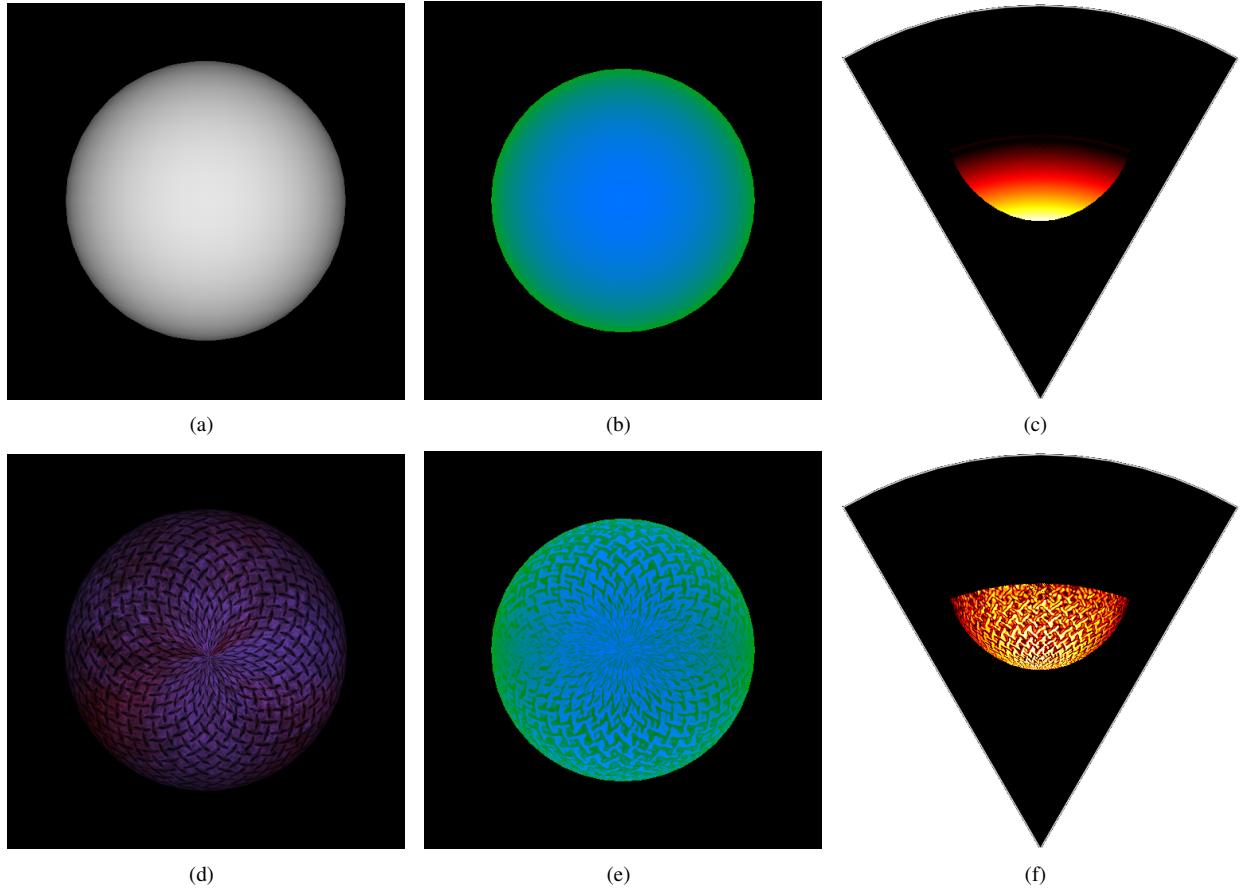


Figure 5: Example of shader rendering with bump mapping processing: A sphere without texture (a) and with texture (d); their respective shader image representation in (b) and (e), where blue is the normal channel and green is the depth one; and the final acoustic image in (c) and (f). By using bump mapping technique, the textures changes the normal directions, and the sonar image details the appearance of object surface, like in real-world sensing.

where  $I_{bin}$  is the intensity in the bin after energy normalization,  $x$  is the pixel location in the shader matrix,  $N$  is the depth histogram value (number of pixels) of that bin,  $S(i_x)$  is a 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 informations of the bins. For that, the number of bins involved is in direct proportion to the sonar image resolution.

theorem, which states that the sum of many independent and identically distributed random variables tends to behave a Gaussian random variable. A Gaussian distribution is defined by 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 performed in each frame.

### 3.4. Noise model

Imaging sonar systems are disturbed by a multiplicative noise known as speckle, and caused by coherent processing of backscattered signals from multiple distributed targets. These latter ones degrade 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 [21]:

$$y(t) = x(t) \times n(t), \quad (2)$$

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

### 3.5. Integrating sonar device with Rock

To export and display the sonar image, the simulated data is encapsulated as Rock's sonar data type, and provided as an output I/O port of a Rock's component.

## 4. Simulation results and experimental analysis

To evaluate our simulator, experiments were conducted by using a 3D model of an AUV equipped with one MSIS and one FLS, studied in different scenarios. The following devices configuration are summarized in Table 1. As the scene frames are being captured by the sonars, resulting simulated acoustic images are sequentially presented, on-the-fly.

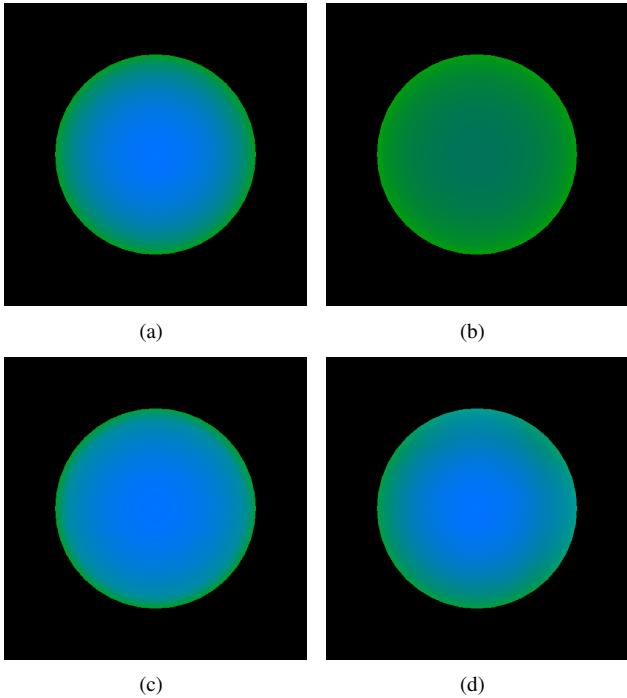


Figure 6: Examples of different reflectance values,  $\rho$ , applied in shader image representation, where blue is the normal channel and green is the depth channel: (a) raw image; (b)  $\rho = 0.35$ ; (c)  $\rho = 1.40$ ; and (d)  $\rho = 2.12$ .

Table 1: Sonar devices 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°

#### 319 4.1. Experimental evaluation

320 The virtual FLS from AUV was used to insonify the scenes  
 321 from three distinct scenarios. A docking station, in parallel  
 322 with a pipeline on the seabed, composes **the first scenario** (see  
 323 Fig. 7(a)). The target surface is well-defined in the simulated  
 324 acoustic frame (see Fig. 7(b)), even as the shadows and speckle  
 325 noise. Given the docking station is metal-made, the texture and  
 326 reflectivity were set, such as a higher intensity shape was re-  
 327 sulted in comparison with the other targets. **The second sce-  
 328 nario** presents the vehicle in front of a manifold model in a  
 329 non-uniform seabed (see Fig. 7(c)). The target model was in-  
 330 sonified to generate the sonar frame from the underwater scene.  
 331 The frontal face of the target, as well the portion of the seabed  
 332 and the degraded data by noise, are clearly visible in the FLS  
 333 image. Also, a long acoustic shadow is formed behind the man-  
 334 ifold, occluding part of the scene. **The third scenario** con-  
 335 tains a sub-sea isolation valve (SSIV) structure, connected to a  
 336 pipeline in the bottom (see Fig. 7(e)).

337 Due to sensor configuration and robot position, the initial  
 338 bins usually present a blind region in the three simulated scenes,  
 339 caused by absence of objects at lower ranges, similar to real im-

340 ages. Also, the brightness of sea-floor decreases, when farthest  
 341 from sonar, because of the normal orientation of the surface.

342 The MSIS was also simulated in three different experiments.  
 343 The robot in a big textured tank composes **the first scene** (see  
 344 Fig. 8(a)). Similar to the first scenario of FLS experiment, the  
 345 reflectivity and texture were set to the target. The rotation of  
 346 the sonar head position, by a complete 360° scanning, pro-  
 347 duced the acoustic frame of tank walls (see Fig. 8(b)). **The**  
 348 **second scene** involves the vehicle's movement during the data  
 349 acquisition process. The scene contains a grid around the AUV  
 350 (see Fig. 8(c)), an MSIS in the front of the AUV is used. This  
 351 trial induces a distortion in the final acoustic frame, because the  
 352 relative sensor position with respect to the surrounding object  
 353 changes while the sonar image is being built (see Fig. 8(d)). In  
 354 this case, the robot rotates 20° left during the scanning. **The**  
 355 **last scene** presents the AUV over oil and gas structures on the  
 356 sea bottom (see Fig. 8(e)). Using an MSIS located in the back  
 357 of the AUV, with a vertical orientation, the scene was scanned  
 358 to produce the acoustic visualization. As illustrated in Fig. 8(f),  
 359 object surfaces present clear definition in the slice scanning of  
 360 the sea-floor.

361 The experimental scenarios provided enough variability of  
 362 specific phenomena usually found in real sonar images, such as  
 363 acoustic shadows, noise interference, surface irregularities and  
 364 properties, distortion during the acquisition process and gradi-  
 365 ent of acoustic intensities. However, our sonar model also pre-  
 366 sented some limitations. For instance, the speckle noise appli-  
 367 cation is restricted to regions with acoustic intensity, as shown  
 368 in Figs. 7(f) and 8(b). This fact is due to our sonar model,  
 369 defined in Eq. 2, be multiplicative. In real sonar images, the  
 370 noise also granulates the shadows and blind regions. The sonar  
 371 simulator can be improved by inserting an additive noise to our  
 372 model. A second feature missing in simulated acoustic images  
 373 are the ghost effects caused by reverberation. This lacking part  
 374 can be addressed by implementation of a multi-path propaga-  
 375 tion model.

#### 376 4.2. Computational time

377 Performance evaluation of the simulator was determined by  
 378 considering the suitability to run real-time applications. The  
 379 experiments were performed on a laptop with Ubuntu 16.04  
 380 64 bits, Intel Core i7 3540M processor running at 3 GHz with  
 381 16GB DDR3 RAM memory and NVIDIA NVS 5200M video  
 382 card.

383 The elapsed time of each sonar data is stored to compute the  
 384 average time, standard deviation and frame rate metrics, after  
 385 500 iterations, as summarized in Tables 2 and 3. After chang-  
 386 ing the device parameters, such as number of bins, number of  
 387 beams and field of view, the proposed approach generated the  
 388 sonar frames with a high frame rate, for both sonar types. Given  
 389 the Tritech Gemini 720i, a real forward-looking sonar sensor,  
 390 with a field of view of 120° by 20° and 256 beams, presents a  
 391 maximum update rate of 15 frames per second, the results allow  
 392 the use of the sonar simulator for real-time applications. Also,  
 393 the MSIS data built by the simulator is able to complete a 360°  
 394 scan sufficiently fast in comparison with a real sonar as Tritech

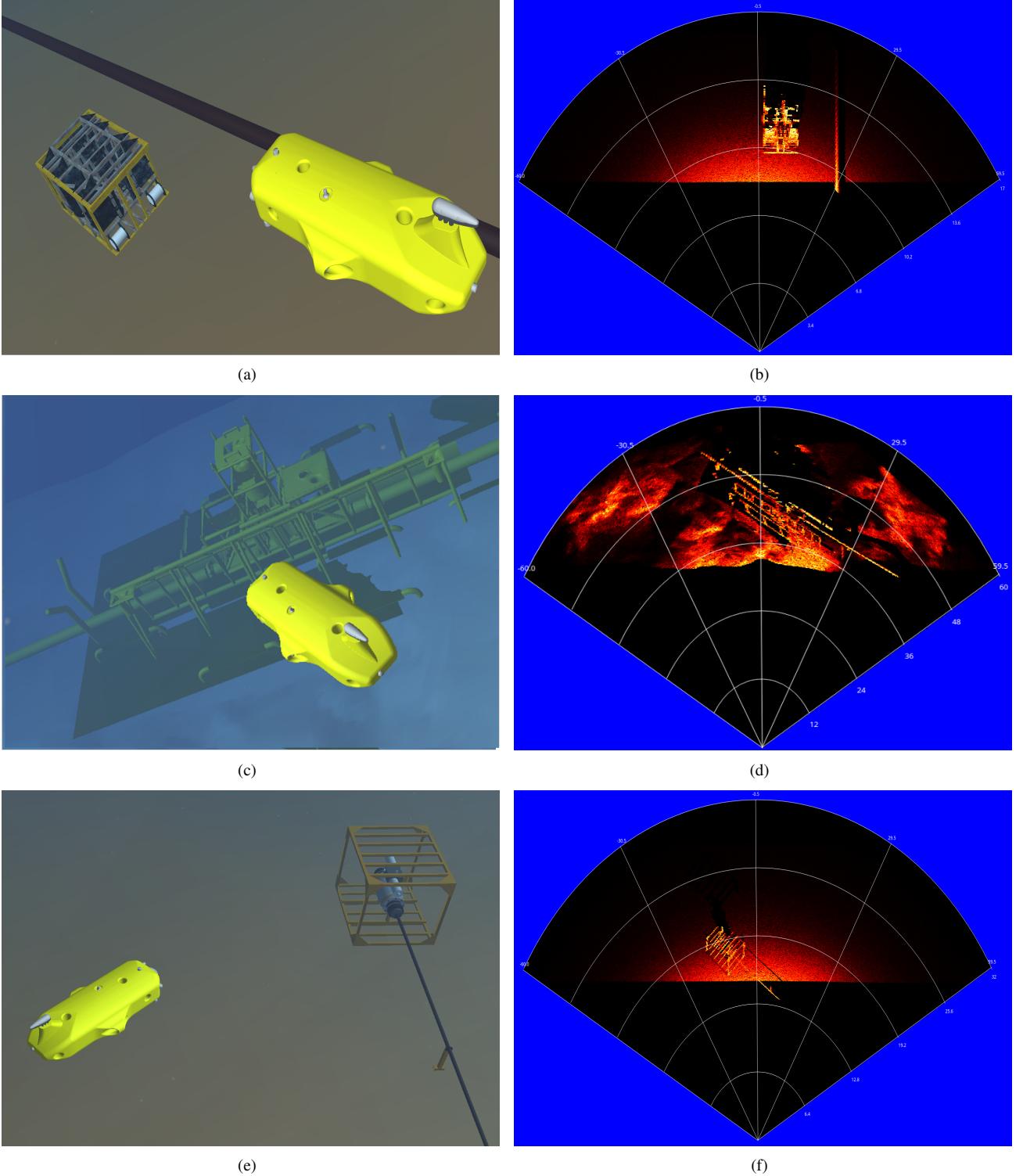


Figure 7: Forward-looking sonar simulation experiments: (a), (c) and (e) present the virtual underwater trials, while (b), (d) and (f) are the following acoustic representations of each scenario, respectively.

395 Micron DST. Moreover, for the FLS device, these rates are su-  
 396 perior to the rates lists by DeMarco *et al* [5] (330ms) and Saç  
 397 *et al* [4] (2.5min). For MSIS type, to the best of our knowledge,  
 398 there is any previous work for comparison.

399 According to previous results, since the number of bins is

400 directly proportional to sonar image resolution, as explained in  
 401 Section 3.3, this is also correlated with the computational time.  
 402 When the number of bins increases, the simulator will have a  
 403 bigger scene frame to compute, and generate the sonar data.

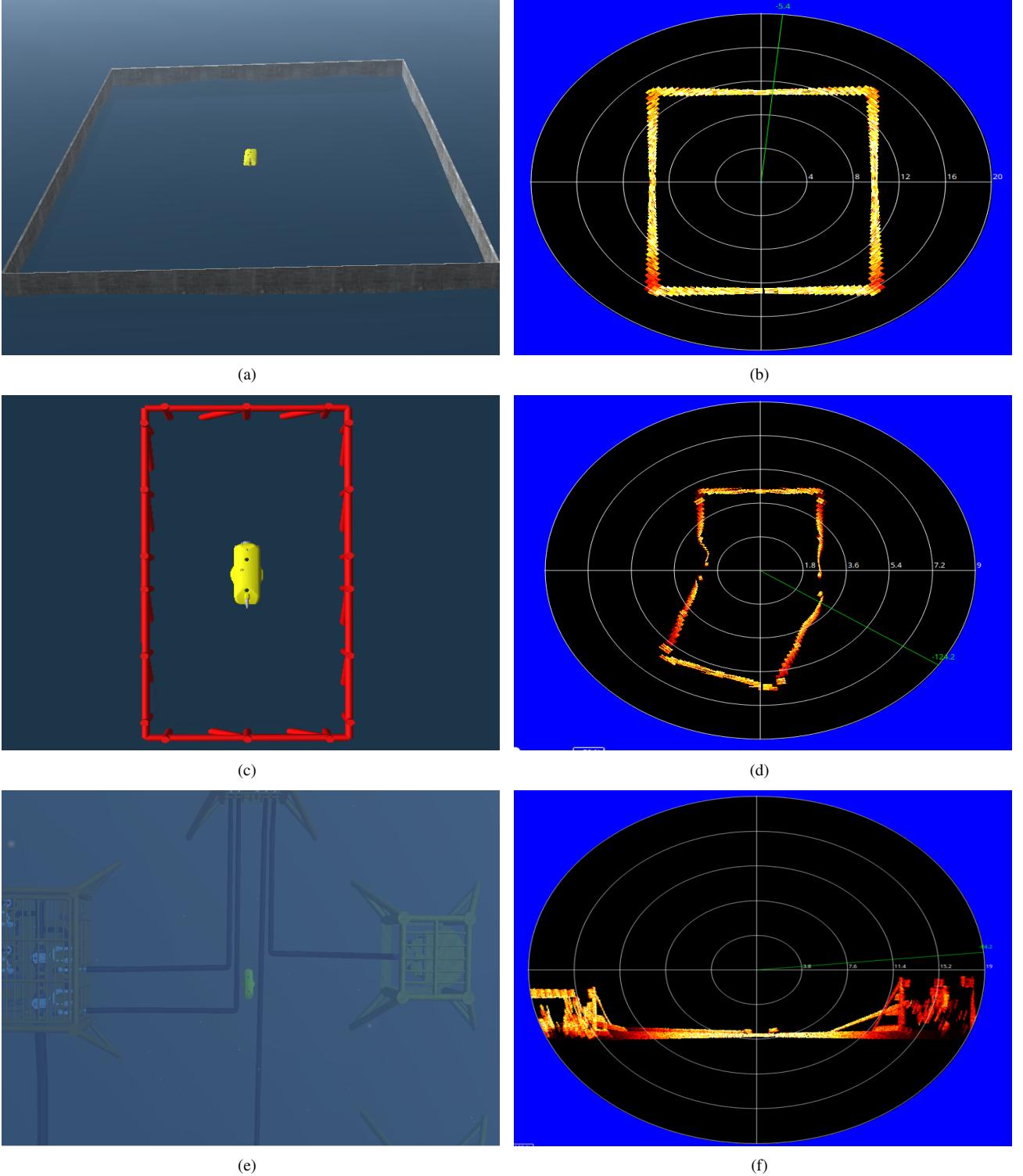


Figure 8: Experiments using mechanical scanning imaging sonar in three different scenarios (a), (c) and (e), and the respective processed simulated frames in horizontal axis in (b) and (d), and vertical axis in (f).

## 404 5. Conclusion and future work

405 A GPU-based approach for imaging sonar simulation is pre-  
 406 sented here. The same model was able to reproduce the oper-  
 407 ation mode of two different types of sonar devices (FLS and

408 MSIS) over different scenarios. The real sonar image singulari-  
 409 ties, such as multiplicative noise, surface properties and acous-  
 410 tic shadows are addressed and represented in the simulated frames.  
 411 Specially for the shadows, the acoustic representation can present  
 412 information as useful as the insonified object. Considering the

Table 2: Processing time to generate forward-looking sonar frames 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

413 qualitative results, the sonar simulator can be by feature de-  
414 tection algorithms, based on corners, lines and shapes. Also,  
415 the computation time to build one sonar frame was calculated  
416 using different device settings. The vertex and fragment pro-  
417 cessing during the underwater scene rendering accelerates the  
418 sonar image building, and the metrics, such as the average time,  
419 demonstrated that the performance is much close to real imag-  
420 ing devices. These results allowed the use of this imaging sonar  
421 simulator by real-time applications, such as obstacle detection  
422 and avoidance, and object tracking. Finally, the reverberation  
423 effect is still missing on our simulator to perform a more realis-  
424 tic sensing. The future inclusion of this phenomena will prob-  
425 ably change the reflected intensity model and the computation  
426 time must be considered again. Next steps will focus on qualita-  
427 tive and computation-efficiency evaluations with other imaging  
428 sonar simulators.

- 447 [7] Kwak S, Ji Y, Yamashita A, Asama H. Development of acoustic camera-  
448 imaging simulator based on novel model. In: IEEE International Con-  
449 ference on Environment and Electrical Engineering (EEEIC). 2015, p.  
450 1719–24.
- 451 [8] Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, et al. ROS: an  
452 open-source robot operating system. In: Workshop on Open Source Soft-  
453 ware, held at IEEE International Conference on Robotics and Automation  
454 (ICRA). 2009, p. 1–6.
- 455 [9] Hurtós N. Forward-looking sonar mosaicing for underwater environments.  
456 Ph.D. thesis; Universitat de Girona; 2014.
- 457 [10] Abbot J, Thurstone F. Acoustic speckle: theory and experimental analy-  
458 sis. Ultrasonic Imaging 1979;1(4):303–24.
- 459 [11] Ganesan V, Chitre M, Brekke E. Robust underwater obstacle detection  
460 and collision avoidance. Autonomous Robots 2015;40(7):1–21.
- 461 [12] Ribas D, Rido P, Neira J. Underwater SLAM for structured environ-  
462 ments using an imaging sonar. Springer-Verlag Berlin Heidelberg; 2010.
- 463 [13] Fallon MF, Folkesson J, McClelland H, Leonard JJ. Relocating under-  
464 water features autonomously using sonar-based SLAM. Journal of Ocean  
465 Engineering 2013;38(3):500–13.
- 466 [14] Liu L, Xu W, Bian H. A LBF-associated contour tracking method for  
467 underwater targets tracking. In: MTS/IEEE OCEANS Conference. 2016,  
468 p. 1–5.
- 469 [15] Huang TA, Kaess M. Towards acoustic structure from motion for imaging  
470 sonar. In: IEEE/RSJ International Conference on Intelligent Robots and  
471 Systems (IROS). 2015, p. 758–65.
- 472 [16] Bradski G. The opencv library. Doctor Dobbs Journal 2000;25(11):120–  
473 6.
- 474 [17] Watanabe T, Neves G, Cerqueira R, Trocoli T, Reis M, Joyeux S, et al.  
475 The Rock-Gazebo integration and a real-time AUV simulation. In: IEEE  
476 Latin American Robotics Symposium (LARS). 2015, p. 132–8.
- 477 [18] SDF. <http://sdformat.org/>; 2017. Accessed: 2017-04-23.
- 478 [19] Soetens P, Bruyninckx H. Realtime hybrid task-based control for robots  
479 and machine tools. In: IEEE International Conference on Robotics and  
480 Automation (ICRA). 2005, p. 260–5.
- 481 [20] Rost RJ, Licea-Kane B, Ginsburg D, Kessenich JM, Lichtenbelt B, Malan  
482 H, et al. OpenGL shading language. 3rd ed.; Addison-Wesley Profes-  
483 sional; 2009.
- 484 [21] Lee J. Digital image enhancement and noise filtering by use of local  
485 statistics. IEEE Transactions on Pattern Analysis and Machine Intelli-  
486 gence 1980;2(2):165–8.

## 429 References

- 430 [1] Bell JM. Application of optical ray tracing techniques to the simulation  
431 of sonar images. Optical Engineering 1997;36(6):1806–13.
- 432 [2] Coiras E, Groen J. Simulation and 3d reconstruction of side-looking sonar  
433 images. In: Silva S, editor. Advances in Sonar Technology; chap. 1. In-  
434 Tech; 2009, p. 1–15.
- 435 [3] Guériot D, Sintes C. Forward looking sonar data simulation through tube  
436 tracing. In: MTS/IEEE OCEANS Conference. 2010, p. 1–6.
- 437 [4] Saç H, Leblebicioğlu K, Bozdağı Akar G. 2d high-frequency  
438 forward-looking sonar simulator based on continuous surfaces ap-  
439 proach. Turkish Journal of Electrical Engineering and Computer Sciences  
440 2015;23(1):2289–303.
- 441 [5] DeMarco K, West M, Howard A. A computationally-efficient 2d imag-  
442 ing sonar model for underwater robotics simulations in Gazebo. In:  
443 MTS/IEEE OCEANS Conference. 2015, p. 1–8.
- 444 [6] Gu J, Joe H, Yu SC. Development of image sonar simulator for under-  
445 water object recognition. In: MTS/IEEE OCEANS Conference. 2013, p.  
446 1–6.

## 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