HoloOcean: Realistic Sonar Simulation

Easton Potokar, Kalliyan Lay, Kalin Norman, Derek Benham, Tracianne B. Neilsen,
Michael Kaess, and Joshua G. Mangelson

***Abstract*— Sonar sensors play an integral part in underwater robotic perception by providing imagery at long distances where standard optical cameras cannot. They have proven to be an important part in various robotic algorithms including localization, mapping, and structure from motion. Unfortunately, generating realistic sonar imagery for algorithm development is difficult due to the high cost of field trials and lack of simulation methods. To remove these obstacles, we present various upgrades to the sonar simulation method in HoloOcean, our open-source marine robotics simulator. In particular, we improve the noise modeling using a novel cluster-based multi-path ray-tracing algorithm, various probabilistic noise models, and material dependence. We also develop and integrate simulated models for side-scan, single-beam, and multibeam profiling sonars.**

I. Introduction

Many algorithms for autonomous underwater vehicles (AUVs) leverage the use of sonar imagery to localize, detect objects, generate maps, or perform 3D reconstruction where standard optical imagery can be unusable due to water turbidity. These algorithms are useful for many applications of AUVs such as marine infrastructure inspection, exploration of oceans, and many others. These applications have the potential to drastically increase safety, quality of life, and general scientific knowledge.

Unfortunately, generating the required realistic sonar imagery for initial algorithm development can be difficult due to field trials being expensive and high-risk. While simulation has the potential to help with this, current simulation methods lack the heavy noise present in real-world sonar imagery. In real-world scenarios, this noise is derived from a variety of sources including, but not limited to, reverberation, inaccurate sensor readings, multi-path acoustic wave propagation, and diffusion.

HoloOcean [1] is an open-source underwater simulator built upon Unreal Engine 4 (UE4) [2]. It includes a novel sonar simulation method leveraging an octree representation of the environment as compared to computationally expensive ray-tracing approaches. We have augmented the sonar simulation in HoloOcean with the following features:

1) A novel cluster-based ray-tracing method to efficiently compute significant multi-path noise found in sonar imagery,
2) Implementation of various probabilistic noise models that more accurately represent reality and produce realistic looking sonar imagery,
3) Easily editable material dependence built into the octree structure, and
4) Implementations of echosounder, side-scan, and multibeam profiling sonars all built upon the octree representation.

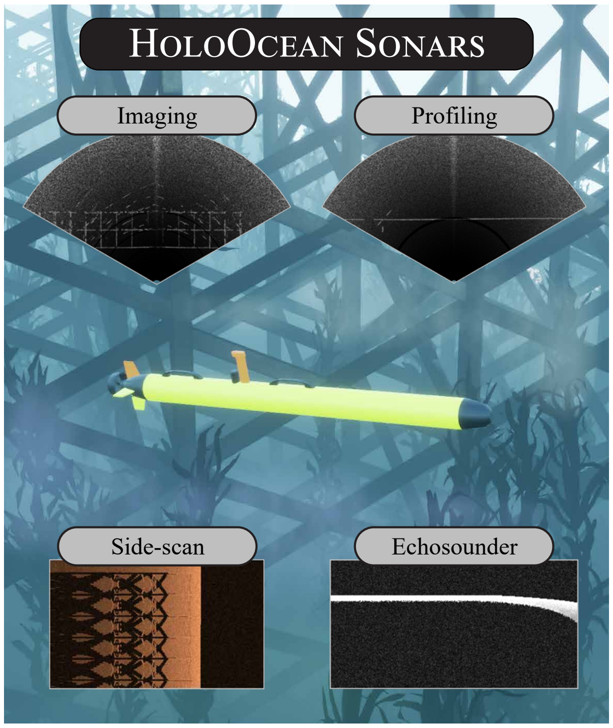


Fig. 1: HoloOcean has been upgraded to include implementations of multibeam imaging, multibeam profiling, side-scan, and echosounder sonars. Further, the noise models have been significantly improved to provide more realistic imagery.

The paper proceeds as follows. Section II reviews current sonar simulation methods and models. In Section III, we describe our novel cluster-based multi-path algorithm, as well as demonstrate our implementation of acoustic reflection models based on material properties. We also describe the various probabilistic models chosen to perturb the data in the same section. This is followed by our side-scan sonar implementation details in Section IV, multibeam profiling sonar in Section V, and the echosounder in Section VI. Finally, Section VII concludes the article and proposes future work. More information about HoloOcean can be found at `https://holoocean.readthedocs.io/`.

This work relates to Department of Navy awards N00014-21-1-2435 and N00014-21-1-2272 issued by the Office of Naval Research.

E. Potokar, K. Norman, K. Lay, D. Benham, T. Neilsen and J. Mangelson, are at Brigham Young University; M. Kaess is at Carnegie Mellon University.

{eastonpots, kalliyan, kalinnorman, derek.benham, tbnbyu, joshua_mangelson}@byu.edu, kaess@cmu.edu

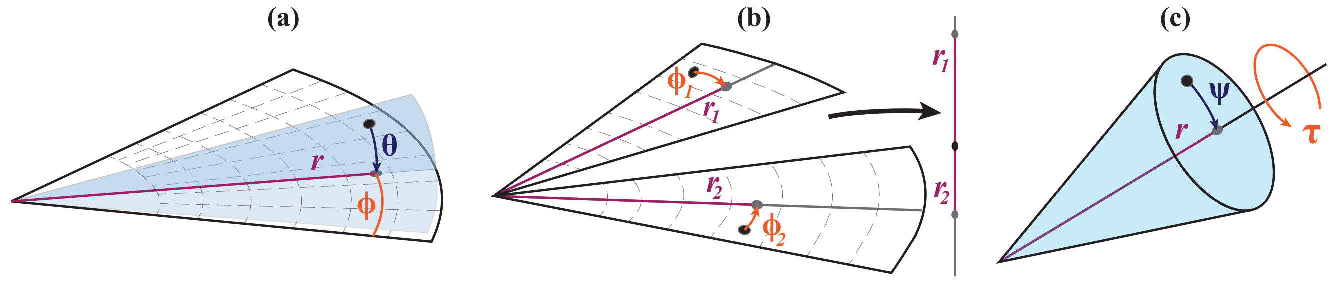


Fig. 2: Geometric representation of various sonar types. (a) Geometry of a multibeam imaging sonar, where the intensities are projected onto the range r, azimuth ϕ plane and elevation θ is lost. A multibeam profiling sonar functions similarly, with the exception that the elevation θ is so tight it's negligible. (b) Side-scan sonar geometry, where elevation angle is negligible, and the intensities are projected onto the r axis, and then concatenated. (c) Cone shape of an echosounder ping, where intensities are projected onto the r axis, and semi-vertical angle ψ is lost. Central angle τ is used for shadowing purposes.

II. Related Work

Creating a realistic underwater sonar simulation is a difficult problem, requiring many pieces for the image to be usable as a fill-in for real-world data. These include accurate modeling of the heavy noise often found in sonar imagery, modeling of high fidelity environments, and modeling of the various types of sonar sensors.

Current sonar simulation methods include ray-tracing [3, 4] or GPU-based methods [5], but little work has been done on modeling the heavy noise present in sonar imagery. This noise is a product of multi-path reverberations, acoustic scattering, general sensor noise, ambient waves, and many other phenomena. Many simulation methods model speckle noise using additive and multiplicative methods, but ignore other forms of noise [6, 7]. Attempts have been made to leverage a generative adversarial network [8, 9] or style transfer networks [10] to recreate this noise, but those lack a clear way to extrapolate to more complex environments or between varying sonar parameters.

Under our own observation, one of the primary noises present in sonar imagery is largely due to multi-path reflections off of objects or structures in the environment. This noise is difficult to recreate in real time due to the heavy computational cost. Preliminary attempts to selectively ray-trace multi-path reverberations have been done using a virtual camera [11], but are limited due to the camera field of view not matching that of a true sonar.

The intensities measured by sonars are dependent on properties of the materials the acoustic waves reflected off of, specifically acoustic impedance that depends on the material speed of sound and material density [12]. While this acoustic impedance has been included in sonar simulations previously [4, 13], it has generally been in simple environments with limited material variety.

Work has been done in simulating a side-scan sonar [14–17], profiling sonar [18], and to a lesser extent echosounder [19]. However, these approaches are all spread out across numerous simulators, are generally ray-tracing-based methods that can be computationally expensive, and are often not open-source.

HoloOcean [1] has an octree-based sonar simulation method, allowing for fast simulation by avoiding ray-tracing, and easy access to the structure of the environment. We augment its simulation method by adding in a clustering-based multi-path propagation algorithm; material dependence in large, complex UE4-based environments; various probabilistic models for noise introduction; and implementations of the sidescan, profiling, and echosounder sonars.

III. Modeling Noise in Imaging Sonar

A multibeam imaging sonar is a common sensor in underwater robotics used for generating imagery of the environment. It generates these images by sending out acoustic pings, which upon return are categorized by their intensity, range that is calculated from the time of flight, and azimuth angle that is estimated via beamforming methods. Results are used to create a 2D image where each pixel value is the intensity for a range and azimuth interval. This geometry is shown in Fig. 2(a).

A. Simulation Model

HoloOcean's sonar simulation method is based upon an octree structure of a UE4 environment [1]. At each timestep, this octree is recursively searched for all the leaves that are in the field of view. These leaves are then sorted in azimuth ϕ, elevation θ bins which are then sorted by range in ascending order. The leaves beyond the first cluster are removed as they are in shadows, and all the resulting leaves have their intensities calculated using surface normals. Intensities are then averaged over each r, ϕ bin. We have built on this simulation model by adding multi-path contributions and various noise improvements, as explained in the following subsections.

B. Multi-path

Building upon the octree structure that HoloOcean provides, after locating and sorting the octree leaves in the sonar field of view into r, ϕ, θ bins, multi-path contributions are then ready to be computed. Note each leaf has unit normal n, location p represented in the sonar frame, and a return unit vector v pointing to the sonar stored as part of its structure, all of which are used throughout. We denote which leaf these belong to through their subscript, for example leaf k has values n_k, p_k, v_k.

First, each leaf is sorted into a cluster. Clusters are made by choosing an octree leaf o and selecting all corresponding leaves within ϵ bins that have normals that are within δ

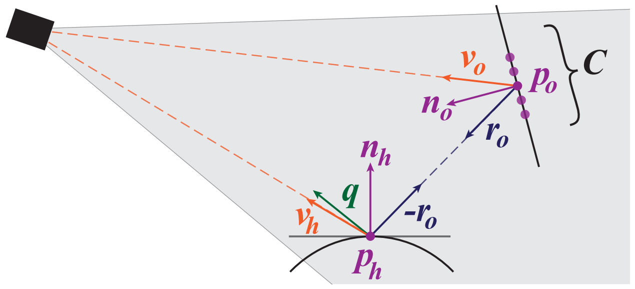


Fig. 3: Visualization of cluster-based multi-path algorithm. A cluster C is created around an octree leaf o, which has normal, position, reflection, and return vector of n_o, p_o, r_o, v_o, respectively. Leaf o is ray-traced to leaf h along r_o, where when impact is found $-r_o$ is reflected into q. The surface at leaf h is then approximated using a plane created from n_h, p_h, off of which the remaining leaves in cluster C will be reflected.

degrees of the original leaf. In practice, we have found values of $\epsilon = 5, \delta = 15°$ have given good results.

The original leaf is then ray-traced. This is done by rotating the return vector v_o by 180° about its normal n_o as follows [20]

$$r_o = -v_o + 2(n_o \cdot v_o)n_o. \tag{1}$$

We then step along the bins in the direction of r_o until either we step out of the field of view of the sonar or hit another leaf h.

If we hit a leaf h, we make the assumption that the small region about h is planar and defined by the location p_h, normal n_h, and return vector v_h. This assumption implies that any leaf i in the cluster will bounce off this same plane. Ray-tracing i then simplifies to finding the intersection between the line given by p_i, r_i and the plane p_h, n_h reducing computations significantly. A closed form solution is given by [21]

$$p = p_i + r_i\left(\frac{(p_h - p_i)\cdot n_h}{r_i \cdot n_h}\right). \tag{2}$$

These steps are all visualized in Fig. 3. From this p, a final azimuth angle can be computed, as well as the resulting distance as follows

$$d = \frac{||p_i|| + ||p - p_i|| + ||p_i||}{2}. \tag{3}$$

We compute q by reflecting $-r_i$ off of n_h as in eq. (1). Using q, the resulting intensity value is found by dotting the return ray of p with q, and scaling by both materials' power reflection coefficients as in Section III-C. This intensity value is then averaged along with all the other leaf and multi-path contributions in the same r, ϕ bin. These steps are summarized in Fig 4.

C. Material Dependence

The intensities measured by the sonar are dependent on the materials that the acoustic waves reflected off of. Given normal incidence, the power reflection coefficient off leaf k is given by [12]

$$R_{\Pi} = \frac{I_r}{I_i} = \left(\frac{z_k - z_w}{z_k + z_w}\right)^2 \tag{4}$$

$$z_k = \rho_k c_k, \tag{5}$$

Algorithm 1: Clustering-Based Ray-Tracing

```
1  F ≜ Leaves found in field of view;
2  C_i ≜ Cluster i;
3  i = 0;
   // Notation
4  $n$ represents normals;
5  $p$ represents positions;
6  $r$ represents reflection vector;
7  $v$ represents impact normal;
8  $idx$ represents $r, \phi, \theta$ bin index;
9  while F is not empty do
10     A = Pop F;
11     foreach B in F do
12         if $||idx_A - idx_B|| < \epsilon$ and $n_A \cdot n_B > \cos(\delta)$ then
13             Add B to C_i;
14             Remove B from F;
15     i += 1;
16 foreach $C_i$ in $C$ do
17     A = Pop C_i;
18     H = Raytrace from $p_A$ in direction $r_A$;
19     if H then
20         foreach B in $C_i$ do
               // Find intersection plane & line
21             $t = (p_H - p_B) \cdot n_H / r_B \cdot n_H$;
22             $p = p_B + n_B t$;
               // Find return bounce & intensity
23             $q = r_B - 2(n_H \cdot r_B)n_H$;
24             $i = q \cdot v_H$;
               // Find total length
25             $d = (||p_B|| + ||p|| + ||p_B - p||)/2$;
26             Add $i$ to $r, \phi$ bin;
```

Fig. 4: Pseudocode for clustering-based multi-path algorithm. Lines 4-10 correspond to the actual basic clustering algorithm, lines 11-14 to the ray-tracing of a single cluster element, and lines 15-21 to the ray-tracing of the rest of the cluster.

where I_r, I_i are the reflected and incident intensities respectively; z_k, z_w the specific acoustic impedance of leaf k and water respectively; and ρ_k, c_k are the density and the speed of sound, respectively, of leaf k.

Using this relation, we have the following approximation for the returned intensity of leaf k as follows

$$I_k = \left(\frac{z_k - z_w}{z_k + z_w}\right)^2 (v_k \cdot n_k) \tag{6}$$

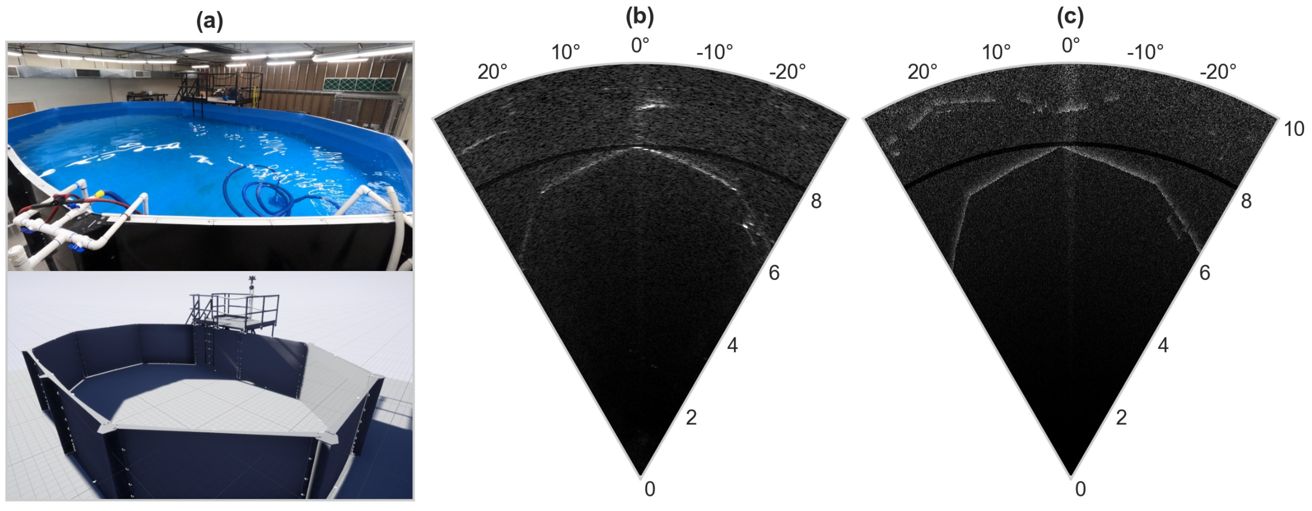
where I_k is the return intensity, v_k the return vector, and n_k the surface unit normal all of the octree leaf k. Note $v_k \cdot n_k$ results in the angle between the two unit vectors.

Material names are pulled directly from UE4 and the material properties ρ_k, c_k for each material are saved in a csv file for easy editing without the need for recompilation. This allows for on-the-fly tweaks to materials' properties between simulation missions.

D. Probabilistic Contributions

Apart from the noise contributed by multipath, A number of other sources of noise are present in sonar imagery. We chose to perturb our imagery using various distributions to introduce these noise sources.

The time of flight is measured by the sonar and is proportional to the reflection distance. This measured value often suffers from diffusion, meaning it may result in some



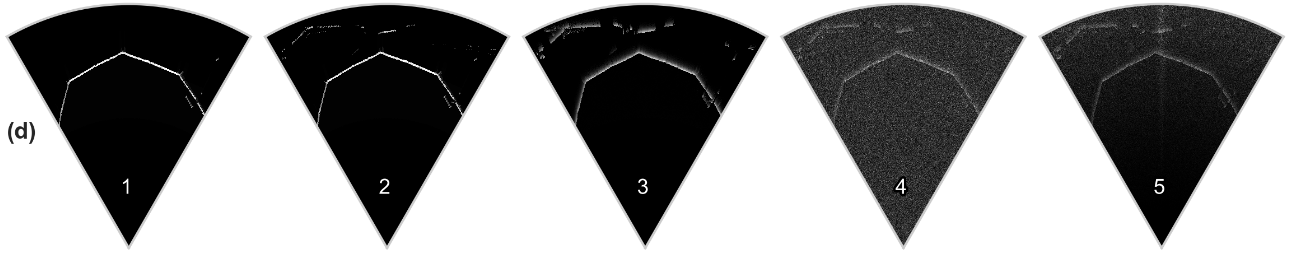


Fig. 5: Comparison of real world and simulation sonar imagery taken in our testing tank. (a) Comparison of our real tank, and the UE4 environment of our tank. (b) Sonar image taken from a Blueprint Subsea Oculus M1200d [22] on the HF setting with a 60° azimuth, 12° elevation, and 10m max range and (c) our simulation method with identical parameters and a cluster size of $\epsilon = 5$. The multi-path differences in (b) and (c) at the 9m marks are likely due to differences in sonar pose between real world and simulation. (d) Computation steps of our model from (1) initial octree image, (2) multi-path, (3) range diffusion, (4) additive and multiplicative noise, (5) scaling the additive noise, and finally the azimuth artifact is shown in (c) at about the 8m mark. The figures in (d) are saturated to more easily visualize the changes.

returns longer than the actual time of flight, but none shorter since that would be physically impossible. This phenomenon has previously been modeled with a K-distribution [23], but for computational efficiency, we approximately perturb r with the similar, easy to sample from exponential distribution

$$\tilde{r} = r + \mathrm{w}^r, \qquad \mathrm{w}^r \sim \mathrm{Exp}(\lambda). \tag{7}$$

Not only is the time of flight perturbed by the exponential distribution, but the intensity returns diminish the more r is perturbed. We model this by scaling the intensity using a scaled probability density function f_{W^r}

$$\tilde{I}_k = I_k f_{\mathrm{W}^r}(\mathrm{w}^r)\lambda = I_k \exp(\frac{-\mathrm{w}^r}{\lambda}). \tag{8}$$

These intensities are then averaged over all leaves in a bin to produce a single pixel intensity,

$$I_{ij} = \frac{1}{n}\sum_{k=0}^{n} \tilde{I}_k. \tag{9}$$

These resulting intensities are often scaled onboard the sonar to produce uniform pixel intensities. Since acoustic intensity is proportional to $1/r^2$ [12], it is often normalized on board the sonar using r^2 to get uniform pixel density. This processing method has the effect of increasing the magnitude of the noise as range increases. Also, the lobe shape of the transducer is often used as a normalization factor. Since our intensity is already normalized, these scale factors will only influence the speckle noise. We thus modify the additive noise w^{sa},

$$\tilde{\mathrm{w}}^{sa} = \frac{r_{ij}^2}{r_{max}^2}\big(1 + 0.5\exp(-\phi_{ij})\big)\mathrm{w}^{sa} \tag{10}$$

$$\mathrm{w}^{sa} \sim \mathcal{R}(\sigma^{sa}), \tag{11}$$

where ϕ_{ij}, r_{ij} are the azimuth and range of the i, j bin; r_{max} is the maximum range of the sonar; and $\mathcal{R}$ is the Rayleigh distribution. This additive noise is then combined with multiplicative noise w^{sm},

$$\tilde{I}_{ij} = I_{ij}(0.5 + \mathrm{w}^{sm}) + \tilde{\mathrm{w}}^{sa} \tag{12}$$

$$\mathrm{w}^{sm} \sim \mathcal{N}(0, \sigma^{sm}). \tag{13}$$

E. Azimuth Streaking

Occasionally, a sonar can receive too many reflections from normal incidences with the same time of flight and make the beamforming problem ill-conditioned. In these scenarios, sonars often return intensities at all azimuth angles for a given range, resulting in an azimuth streak. Some sonars attempt to subtract out these differences, but in doing so can cause more artifacts. We estimate when this ill-conditioning

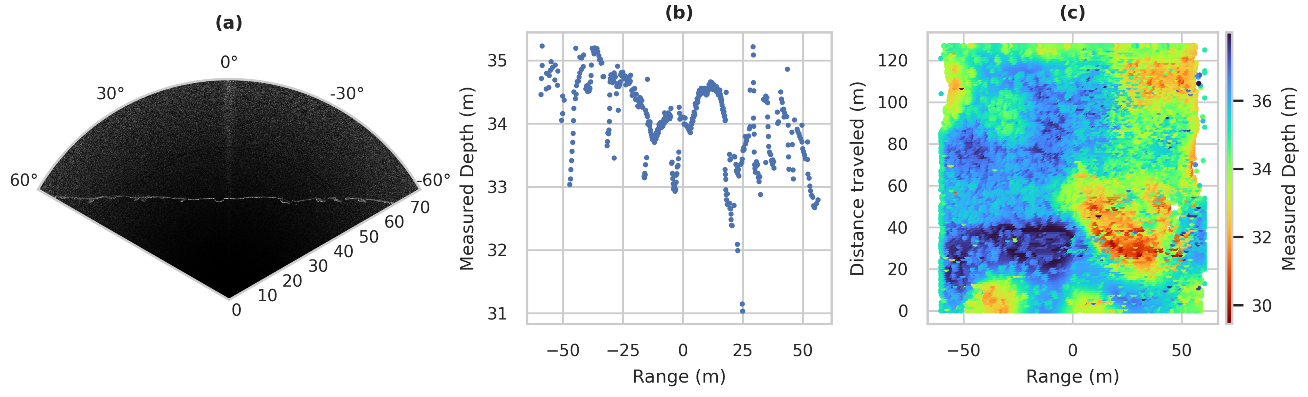


Fig. 6: Example of multibeam sonar simulation and uses. (a) Example of simulated multi-beam sonar imagery taken over an ocean floor looking down. (b) The x,y coordinates extracted from (a), and (c) 120 of these scans placed side by side to create a bathymetry map. Note the techniques used to extract x,y coordinates were rather simple, and the added noise naturally impacted the results. A more advanced bathymetry algorithm is beyond the scope of this paper.

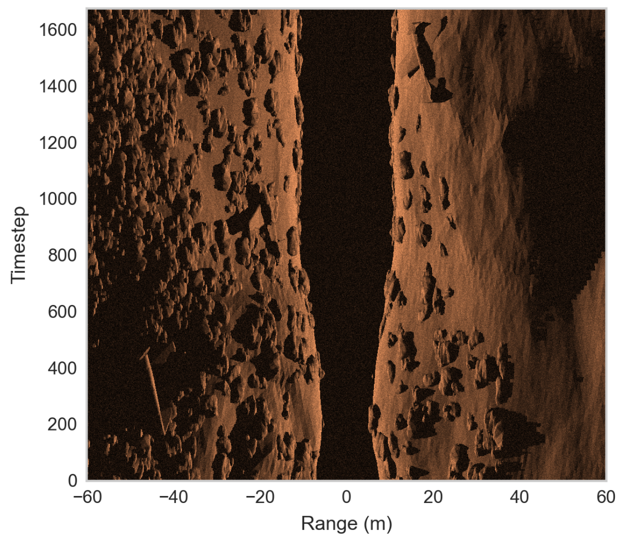


Fig. 7: Example of output of the side-scan sonar simulation. This was taken over 1600 timesteps, each of which results in a row of the image.

occurs by checking if a certain percentage of leaves found in a given range interval with near parallel n and v exceeds a given threshold ζ. This range interval is then modified as follows to introduce the azimuth artifact,

$$\hat{I}_{ij} = 1 - (1 - \tilde{I}_{ij})^2, \tag{14}$$

and if the sonar has on-board gain reduction as follows instead,

$$\hat{I}_{ij} = \tilde{I}_{ij}^2. \tag{15}$$

An example of the resulting image with the contributions of multi-path, diffusion, additive noise scaling, and removal of azimuth streaks can be seen in Fig. 5. Our approach generates approximately 12 images per second at ranges of 10m in the tank environment.

IV. Side-scan Sonar

There are other types of sonar sensors that are also common in underwater robotics, such as the side-scan sonar. In order to increase the data generation capabilities of HoloOcean, we have leveraged the existing robust octree structure to simulate side-scan sonar as well.

A. Operation

Side-scan sonars also send out acoustic waves to capture imagery of their environment. They generally operate by pointing down at the seafloor and emitting two separate pulses, one off each side of the sensor. Upon reflection and return, time of flight and intensity are both measured. In contrast to the imaging sonar sensor, angle of arrival is not estimated, so no azimuth is measured. These pulses are generally sent over a large azimuth angle and over a much tighter elevation. This results in a known range and elevation, but unknown azimuth.

The resulting intensities are then binned from max range to 0m on the left and appended to the array of 0m to max range on the right [17]. A blank portion always occurs in the middle of this data since no objects exist within a certain distance of the sonar.

When visualizing side-scan imagery, each time-step is stitched together to form a waterfall plot; where the y-axis denotes the time-step of the ping, x-axis is the range from max range on the left to 0 in the center to max range on the right, and each pixel value is the given intensity. This geometry is summarized in Fig. 2(b).

B. Simulation Model

The side-scan sonar simulation model is built upon the existing octree structure and the imaging sonar implementation including the existing recursive leaf searching method, shadowing algorithm, and noise models.

It does differ in how the found leaves are sorted. If $\phi < 0$, a leaf is added to the right beam, and to the left beam if $\phi > 0$. The left beam's leaves are binned from largest to smallest range on the left, and the right beam's leaves from smallest to largest range. These beams are then concatenated, and the resulting intensity computed as in eqs. (6) and (9). Resulting imagery can be seen in Fig. 7.

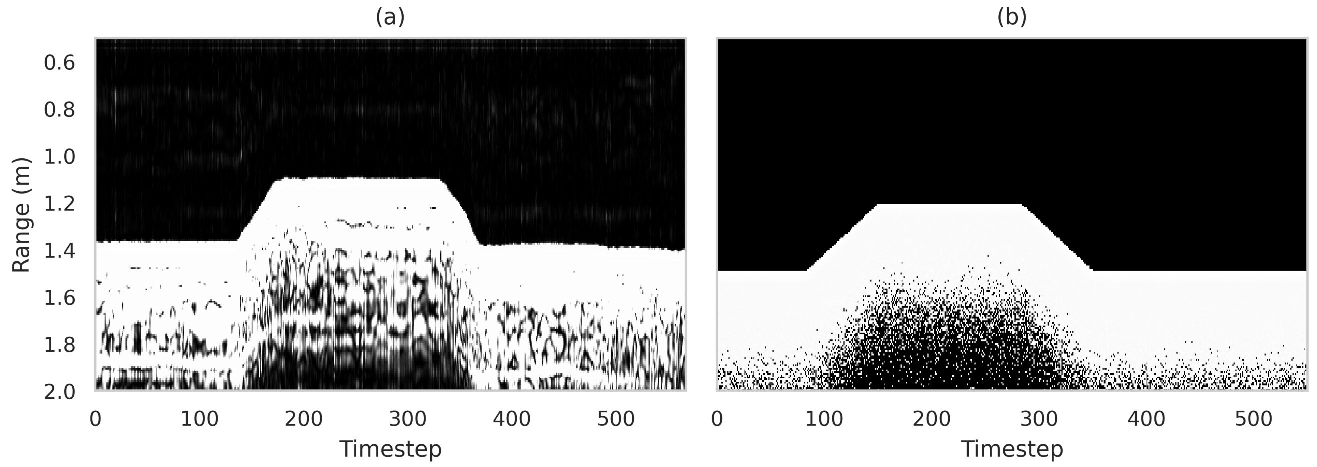


Fig. 8: Comparison of real-world and simulation data of an echosounder in our tank. Simulation and real-world data were taken pointing at the bottom of the tank from the surface, submerging about 0.5m, then returning to the surface. (a) The single-beam data from the BlueRobotics Ping Sonar [24], where the x-axis is the timestep and y-axis is range and (b) the simulation data. Note that these images appear to have extremely heavy noise, but this is mostly due to the small max range used to measure the bottom of the tank.

V. Multibeam Profiling Sonar

For mapping large areas of the seafloor, a multibeam profiling sonar is used for generating low resolution bathymetry maps quickly. Due to its excessive size, weight, and high power consumption, profiling sonars are typically mounted to surface vessels instead of AUVs. The sonar is mounted to the underside of the vehicle pointing downwards towards the seafloor where it scans a broad sweep of the seafloor directly beneath. A systematic survey pattern can then be used for mapping larger areas.

A. Operation

The sonar is comprised of multiple physical sensors, called a transducer array, that send and receive sound pulses that reflect off the seafloor or other present objects. Similar to the imaging sonar, the profiling sonar has a wide azimuth angle that allows a wide field of view, but differs in that the elevation angle is much narrower. With a narrow elevation angle, typically 1 to 3 degrees, the profiling sonar can estimate the true distance and return an array of measurements of the environment scanned below. While surveying an area, by saving the sonar measurements with the estimated pose of the robot at each time step, a 3D representation of the environment can be created from these 2D slices.

B. Simulation Model

Due to the similar characteristics of the imaging sonar, the profiling simulation model implementation is nearly identical, utilizing the recursive leaf searching method and noise models. The models differ in that the profiling only utilizes one elevation bin for shadowing. To demonstrate the use of the profiling sonar in simulation, a simple mission was planned gathering bathymetry data shown in Fig. 6. An example profiling sonar image is shown, along with extracted x, y coordinates, and a resulting bathymetry map.

VI. Echosounder

Echosounders, or single-beam sonar sensors, are common systems used for determining water depth, which is useful for mapping and navigation. Echosounders are often cheaper than other sonars and have wide field of view condensed in a single beam. Although echosounders do not produce maps as detailed as those from a multibeam sonar, they can detect schools of fish and even plankton. They can also be used to prevent collisions with large objects beneath ships. Similar to the multibeam sonar, echosounders are usually mounted to the bottom of vehicles pointed towards the seafloor or on the nose of an AUV pointing forward. However, single-beam sonars take longer to map an area than a multibeam sonar because the resolution of the data depends on the speed of the vehicle on which the sonar is mounted.

A. Operation

The echosounder generally uses a single transducer to transmit and receive acoustic signals in the shape of a cone. Because the speed of sound in water is known, echosounders can use the round-trip time of a signal to calculate the distance traveled. This distance can then be divided in two to determine the distance from the boat to the seafloor, and returned in an array of intensities where each array element corresponds to a different range interval. This geometry is shown in Fig. 2(c).

B. Simulation Model

The echosounder simulation also heavily inherits from the previously mentioned imaging sonar sensor in terms of octree leaf searching and shadowing. However, because of the inherent cone shape of the echosounder, it does not have an elevation or azimuth angle. Instead, only the range and semi-vertical angle ψ of the cone are used to determine if a leaf is within the field of view of the sonar. An additional angle, which we will call the central angle τ, is also used to create bins for shadowing purposes. These angles are visualized in Fig. 2(c). The signal intensities are sorted into ψ, τ bins, and shadowing is computed with these bins in an identical manner as the imaging sonar ϕ, θ bins, as described in Subsection III-A. After shadowing, the remaining leaves are sorted into range bins, the intensities in each bin are averaged, and then returned.

The range calculations are perturbed using the exponential distribution, similar to the imaging sonar case. The intensities are not scaled by the probability density function though, as there is little intensity drop-off for this diffusion.

Fig. 8 shows comparison of real-world data and our simulation model, with an opening angle of 30 degrees, or a semi-vertical angle of 15 degrees.

VII. Conclusion

Based on the octree simulation techniques developed in our previous work [1], we augment HoloOcean, our open-source marine robotics simulator, with a side-scan, multi-beam profiling, and echosounder sonar implementations. These are all built upon the existing octree structure to provide realistic imagery of the complex environments in UE4 engine. Further, we improve the noise modeling through a cluster-based ray-tracing method for multi-path generation, as well as various probabilistic contributions for bin diffusion, additive noise scaling, material dependence, and azimuth streaking. The resulting imagery is realistic and can be used for underwater algorithm development. Future work includes a GPU implementation of the sonar algorithms, additional agents including autonomous surface vessels, and additional custom environments.

References

[1] E. Potokar, S. Ashford, and J. Mangelson, "HoloOcean: An underwater robotics simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2022.

[2] Epic Games, "Unreal engine," https://www.unrealengine.com, 2021.

[3] J.-H. Gu, H.-G. Joe, and S.-C. Yu, "Development of image sonar simulator for underwater object recognition," in *2013 OCEANS - San Diego*, Sep. 2013, pp. 1–6.

[4] S. Kwak, Y. Ji, A. Yamashita, and H. Asama, "Development of acoustic camera-imaging simulator based on novel model," in *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, Jun. 2015, pp. 1719–1724.

[5] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez, and L. Oliveira, "A novel GPU-based sonar simulator for real-time applications," *Computers & Graphics*, vol. 68, pp. 66–76, Nov. 2017.

[6] K. J. DeMarco, M. E. West, and A. M. Howard, "A computationally-efficient 2D imaging sonar model for underwater robotics simulations in Gazebo," in *OCEANS 2015 - MTS/IEEE Washington*, Oct. 2015, pp. 1–7.

[7] A. Rascon, "Forward-Looking Sonar Simulation Model for Robotics Applications," Thesis, Monterey, CA; Naval Postgraduate School, Sep. 2020.

[8] M. Sung, J. Kim, J. Kim, and S.-C. Yu, "Realistic Sonar Image Simulation Using Generative Adversarial Network," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 291–296, Jan. 2019.

[9] D. Liu, Y. Wang, Y. Ji, H. Tsuchiya, A. Yamashita, and H. Asama, "CycleGAN-based realistic image dataset generation for forward-looking sonar," *Advanced Robotics*, vol. 35, no. 3-4, pp. 242–254, Feb. 2021.

[10] S. Lee, B. Park, and A. Kim, "Deep Learning from Shallow Dives: Sonar Image Generation and Training for Underwater Object Detection," *arXiv:1810.07990 [cs]*, Oct. 2018.

[11] R. Cerqueira, T. Trocoli, J. Albiez, and L. Oliveira, "A rasterized ray-tracer pipeline for real-time, multi-device sonar simulation," *Graphical Models*, vol. 111, p. 101086, Sep. 2020.

[12] S. L. Garrett, *Understanding Acoustics*. S.l.: Springer Nature, 2020.

[13] J. Kim, M. Sung, and S.-C. Yu, "Development of Simulator for Autonomous Underwater Vehicles utilizing Underwater Acoustic and Optical Sensing Emulators," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2018, pp. 416–419.

[14] D.-H. Gwon, J. Kim, M. H. Kim, H. G. Park, T. Y. Kim, and A. Kim, "Development of a side scan sonar module for the underwater simulator," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jun. 2017, pp. 662–665.

[15] Y. Pailhas, Y. Petillot, C. Capus, and K. Brown, "Real-time sidescan simulator and applications," in *OCEANS 2009-EUROPE*, May 2009, pp. 1–6.

[16] E. Coiras, A. Ramirez-Montesinos, and J. Groen, "GPU-based simulation of side-looking sonar images," in *OCEANS 2009-EUROPE*, May 2009, pp. 1–6.

[17] J. M. Bell, "A model for the simulation of sidescan sonar," Thesis, Heriot-Watt University, Sep. 1995.

[18] T. Henderson and S. Lacker, "Seafloor profiling by a wideband sonar: Simulation, frequency-response optimization, and results of a brief sea test," *IEEE Journal of Oceanic Engineering*, vol. 14, no. 1, pp. 94–107, Jan. 1989.

[19] B. R. Biffard, "Seabed remote sensing by single-beam echosounder: Models, methods and applications." Thesis, University of Victoria, 2011.

[20] C. GRUBIN, "Derivation of the quaternion scheme via the Euler axis and angle," *Journal of Spacecraft and Rockets*, vol. 7, no. 10, pp. 1261–1263, 1970.

[21] E. W. Weisstein, "Line-Plane Intersection," From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/Line-PlaneIntersection.html.

[22] Blueprint, "Blueprint Subsea Oculus M1200d," https://www.blueprintsubsea.com/oculus/oculus-m-series, 2021.

[23] D. Abraham and A. Lyons, "Novel physical interpretations of K-distributed reverberation," *IEEE Journal of Oceanic Engineering*, vol. 27, no. 4, pp. 800–813, Oct. 2002.

[24] BlueRobotics, "BlueRobotics Ping Sonar," https://bluerobotics.com/store/sensors-sonars-cameras/sonar/ping-sonar-r2-rp/, 2021.