DeepSonar: GPU-Accelerated Sonar Simulation Enhanced with Natural Language Prompting

Puneeth Sarma KAV*[1], Aadi Palnitkar*[1], Arjun Suresh[1], E. Baker Herrin[2], Jane Shin[2], Yiannis Aloimonos[1], Xiaomin Lin[1,3,†]

***Abstract*—Sonar technology plays a crucial role in underwater exploration, autonomous navigation, and robotic perception. However, the development of AI-based sonar models is often constrained by a scarcity of training data.**

To overcome this limitation, we introduce *DeepSonar*, a sonar simulation with integrated natural language processing functionality via the usage of Large Language Models (LLMs). This approach allows users to navigate underwater simulation in tools like Blender with natural language, and produce realistic sonar imagery. Additionally, our pipeline is fully integrated with Objaverse, a large community curated labeled dataset of 3-D models, enabling users to place objects of their choosing into the simulator via natural language prompting. In aggregate, this scalable and automated pipeline is presents a highly customizable framework for the rapid generation of realistic synthetic sonar data, fully rooted in physics-based principles.

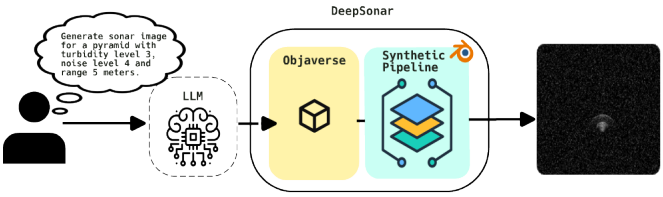


Fig. 1: Overview of the DeepSonar pipeline : The system takes a user natural language prompt, processes it through a Large Language Model (LLM) to extract object and simulation parameters, and generates synthetic sonar images using a physics-based simulation pipeline in Blender. The final output is a realistic sonar image suitable for AI-based sonar perception tasks.

I. Introduction

Sonar technology is fundamental in marine exploration, underwater navigation, and robotic perception due in large part to its imaging applications spanning the domains of oceanographic mapping, search and recovery missions, environmental monitoring, and defense systems [1]. Owing to its ubiquity in underwater applications, sonar data is often used to train vision models [2]–[4], however, datasets for the sonar modality is lacking in availability, due in part to the logistical difficulties and challenges inherent to the domain. As such, sonar simulators which are capable of bridging the sim2real gap and generating realistic sonar data are required.

To this end, we introduce ***DeepSonar***, a pipeline which integrates an underwater robotics simulator based in Blender [5] with the natural language processing capabilities afforded by large language models to enable a realistic sonar simulation. Additionally we use NVIDIA's WARP framework [6] to leverage GPUs for performance, similar to traditional physics-based sonar simulators, thus granting it the ability to generate high-fidelity sonar data using prompting in a computationally efficient manner.

Our research builds on the framework established by *ChatSim* [7] with the ability to automate the creation of high-quality synthetic sonar datasets. Natural language input prompts from the user are used to add objects and change various parameters within a larger underwater scene. Subsequently, realistic sonar imagery is captured through the simulated forward look sonar device. For the 3D meshes in the scene, we utilize *Objaverse* [8], a large, open-source, and curated collection of 3D objects. Users can interact with the scene through API calls initiated directly by the LLM. This methodology enables for a scalable, automated means of generating highly configurable sonar data in a computationally efficient manner.

In summary, we make the following novel contributions:

- A full fledged end-to-end pipeline for sonar image generation
- Fully integrated large language model functionality for scene customization
- GPU acceleration via NVIDIA's state-of-the-art WARP framework

II. Related Work

A. Sonar Operations

Sonar (Sound Navigation and Ranging) utilizes the principles of sound wave propagation to detect objects underwater. In typical sonar operation, the sonar device creates a sound wave, which propagates over an area demarcated in spherical coordinates by some maximum azimuthal angle $\theta_{\max}$, some maximum radial angle $\psi_{\max}$ over some range R, with a maximum and minimum range of $r_{\min}$ and $r_{\max}$ respectively. Upon striking an object, the returning echo is sampled by the sonar, and the strength of the backscattered energy determines the intensity of the object within the final sonar image. At the

* Equal Contributors, † Corresponding Author

[1]Maryland Robotics Center, University of Maryland, College Park, MD 20742, USA {akondamu, apalnitk, arjsur, jyaloimono}@umd.edu

[2] Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA {eherrin, jane.shin}@ufl.edu

[3] Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218, USA {xlin52}@jhu.edu

high level, this is how a typical sonar device operates, and researchers are building upon this foundational understanding of sonar operations to create equivalent computer simulations.

B. Sonar Simulations

Sonar simulations are an active area of research, seeking to better approximate the behavior of this nigh-ubiquitous sensor within the underwater domain. Wang et al. [9] proposed a novel method for simulating two-dimensional forward-looking sonar images, with a particular focus on accurately modeling ground echo effects.

Shin et al. [10] introduced a novel method for producing underwater object images that align acoustically with side-scan sonar data, utilizing the capabilities of Unreal Engine. Although these techniques offer useful resources for generating synthetic sonar data, they frequently necessitate extensive manual adjustment of parameters and may not efficiently accommodate a variety of datasets.

Cerquiera et al. [11], a continuation of [12], adopt a rasterization based approach where they leverage GPU integrated rasterized ray-tracer to achieve highly computationally efficient sonar simulations. In contrast, Choi et al. [13], utilizes a virtual GPU ray sensor to emit rays into the scene to interact with the objects present. The rays are subsequently combined into beams, from which the simulator obtains the relevant range intensity data needed for the final sonar image output. This approach differs from the prior work in that it aims to address the key limitations of being unable to properly replicate speckle noise patterns and other features of realistic sonar output.

The capability to simulate accurate sonar is also a feature touted by many larger underwater simulation papers, such as HoloOcean [14], DAVE [15], and most recently OceanSim [16]. All three feature expansive and realistic underwater environments for robotics simulations. Notably, OceanSim and HoloOcean both share the same sonar implementation, introduced by Potokar et al [17], recursively searching a constructed octree data structure to obtain the points within the sonar field of view (FOV).

C. Large Language Models (LLMs) in Data Generation

With the advent of LLMs, prompt based generation presents an attractive paradigm for the generation of synthetic data. LLM generated data pipelines can be easily used to create tailor made synthetic data for diverse use-cases even by non-technical users, owing to their advanced natural language processing capabilities. Their applications in the synthetic generation of text data is well explored; Brown et al. [18] illustrated the capability of LLMs to engage in few-shot learning across various natural language processing applications, which has led to their adoption in the augmentation and generation of datasets. Hämäläinen et al. [19] evaluated the capabilities of LLMs to generate useful synthetic text responses to questionnaires for human computer interaction (HCI) related endeavors. Li et al. [20] explored the applications of LLM generated data in training text-classification models. Gao et al. [21] showcased the application of LLMs in agent-based modeling, where they enable extensive simulated interactions. Natarajan et al. [22] featured a full end-to-end pipeline leveraging diffusion models and VLM (vision language models) in tandem to dynamically generate synthetic sonar data, including a wide range of commonly seen underwater objects. All of these works showcase the sheer variety of the use cases for LLMs within the realm of synthetic data generation.

D. Large Language Models and Underwater Simulation

The application of large language models to the domain of underwater simulations remains comparitively unexplored. A notable example of a simulator with integrated LLM functionality is *ChatSim* [7], which combines ChatGPT with an underwater simulation, *OysterSim* [23], enabling users to control underwater simulations through conversational prompts. The user is given the ability to dynamically add objects and control the scenery using GPT, which enables researchers to construct custom environments as per the task at hand. Another simulator of similar concept is *OceanChat* [24], which leverages LLM prompting to control an autonomous underwater vehicle (AUV) in a created scene. This simulated AUV is a digital twin of the real-life EcoMapper, and replicates its functionality.

In this work, we extend ChatSim with sonar functionality and GPU acceleration, to create a completely configurable sonar simulation with natural language input that is computationally efficient and produces realistic sonar output.

III. Methods

This paper introduces an extensive simulation framework aimed at emulating the essential physical processes that underlie acoustic sonar imaging in real-world scenarios. In functioning sonar systems, acoustic waves are projected in a conical volume, interact with environmental surfaces, and return echoes whose intensity is influenced by various factors, including surface orientation, material properties, and distance of propagation. Drawing inspiration from these principles, our framework replicates sonar operations by generating rays across discretized azimuth and elevation fields, tracing their interactions with three-dimensional scene geometry, and modeling the strength of the returning signals based on the physical attributes of these interactions. The resulting synthetic fan-shaped sonar images are created by projecting recorded surface interactions onto a two-dimensional Cartesian plane, while also incorporating signal attenuation and noise modeling to enhance realism.

A crucial aspect of this framework is the incorporation of ChatSim [7], which facilitates simulation control by converting natural language commands into executable operations within Blender. This feature enables researchers to intuitively navigate the autonomous underwater vehicle (AUV), adjust its orientation, and alter scene components without the need for manual scripting. By merging high-level verbal commands with physics-accurate sonar rendering, ChatSim expedites

dataset generation, lowers technical barriers, and allows for the scalable and efficient production of realistic synthetic sonar imagery. The following subsections elaborate on the integration of ChatSim for natural language-driven simulation control, the ray generation process, the projection methodology, and the noise modeling techniques utilized to produce high-fidelity synthetic sonar datasets.

A. ChatSim

We employ ChatSim [7] to enhance and simplify the traditionally intricate process of managing code-based interactions within Blender. ChatSim effectively merges large language models (LLMs) such as ChatGPT with the simulation environment, allowing users to issue commands in natural language that are transformed into simulation actions, thereby eliminating the necessity for manual scripting or expertise in Blender. In the framework of our DeepSonar project, we utilize ChatSim to guide the movements of the AUV within the simulated underwater environment, enabling navigation towards areas of interest based solely on user instructions. This integration facilitates intuitive specification of focal areas, dynamic scene modification through the addition or removal of objects, and precise control over the AUV's pose (including position, yaw, pitch, and roll) using straightforward language commands. Once the AUV is positioned, it captures the designated environment with its simulated sonar sensor, generating realistic fan-shaped sonar imagery that accurately represents both the geometric configuration and material characteristics of the scene. This integration offers two significant benefits: (1) it democratizes the customization of underwater simulation environments, eliminating the requirement for specialized knowledge in Blender or Python, and (2) it accelerates the experimental iteration process, enabling researchers to swiftly configure, test, and produce sonar datasets by merging high-level verbal commands with low-level physics-based sonar rendering. In summary, ChatSim's natural language prompting serves as a potent facilitator within DeepSonar, effectively connecting human intent with simulation execution for efficient and scalable sonar data generation.

B. Ray generation

The generation of sonar rays draws inspiration from the methodology outlined in [25]. To effectively simulate the volumetric coverage of a sonar sensor, the dimensions of azimuth, elevation, and radial distance are divided into a predetermined number of bins. Specifically, the azimuthal angle is segmented between θ_{min} and θ_{max} into a defined quantity of azimuth bins, while the elevation is discretized in a similar manner between φ_{min} and φ_{max}. Within each discrete angular sector, a uniform grid of directions is generated through a meshgrid of azimuth and elevation angles. Each direction sampled is subsequently transformed into a unit 3D vector by utilizing spherical-to-Cartesian conversion techniques.

$$\mathbf{d}(\theta, \varphi) = \begin{bmatrix} \cos(\varphi)\cos(\theta) \\ \cos(\varphi)\sin(\theta) \\ \sin(\varphi) \end{bmatrix}. \tag{1}$$

The set of direction vectors is modified by the local pose, which includes both rotation and translation, of the sonar sensor, thereby aligning the rays with the sensor's frame of reference.

Subsequently, the entire array of rays is traced utilizing Blender's integrated ray tracing engine to determine the intersections (hit points) with the surrounding environment, thereby effectively simulating the return signals of an actual sonar sensor. This approach guarantees a systematic and thorough coverage of the sonar field of view while ensuring computational efficiency for simulation purposes.

C. Shadows

An essential aspect of authentic sonar simulation is the precise representation of sonar shadows, areas where acoustic signals do not return due to obstruction by environmental objects. In actual sonar systems, shadows occur when objects obstruct the transmission of sonar waves, leading to regions devoid of signals on the surface below. To simulate this phenomenon, we employ our ray-based simulation projected in a volumetric distribution throughout the scene. When these rays encounter objects, their propagation is halted, leaving the areas obscured by the objects darkened, thus forming shadow regions. By ensuring that the underwater environment features a continuous surface, such as a seabed, we can replicate shadows that closely mimic those found in genuine sonar imagery. This significantly improves the visual and structural authenticity of the produced data, offering critical insights for subsequent perception tasks.

D. Sonar Projection

In the process of generating a simulated sonar image, valid surface hits identified through raycasting are projected from three-dimensional coordinates into a two-dimensional Cartesian framework. For each intersection point, two key parameters are derived: the radial distance r from the sonar's origin to the intersection point, measured within the horizontal plane, and the azimuthal angle θ that indicates the direction of the originating ray. These parameters, namely radial distance r and azimuth θ, facilitate the mapping of three-dimensional intersections onto a fixed-size two-dimensional grid that represents the sonar projection. This mapping process discretizes the spatial arrangement in accordance with the specified sonar resolution, thereby enabling the creation of a fan-shaped image structure.

The intensity contribution I for each grid cell is calculated based on the physical characteristics of the surface interaction. The resultant signal strength is influenced by three primary factors: the cosine of the angle between the incident ray and the surface normal, expressed as $(\hat{\mathbf{n}} \cdot (-\hat{\mathbf{d}}))$ in accordance with Lambertian reflectance principles [26]; the inherent reflectivity of the material at the point of impact, denoted by ρ; and an attenuation factor $\alpha(r)$ that models the frequency based reduction of sonar signal strength over increasing distances. This factor is detailed further in the subsequent section.

$$I = \left(\hat{\mathbf{n}} \cdot (-\hat{\mathbf{d}})\right) \rho \, \alpha(r) \tag{2}$$

Ultimately, the total intensities across all grid cells are normalized by the number of contributing hits, resulting in a fan-shaped two-dimensional sonar projection. This synthetic projection effectively encapsulates both the spatial and material-dependent attributes of the environment, while also considering signal degradation with distance, thus providing realistic data for the training and assessment of sonar perception systems.

E. Frequency Based Attenuation

We follow the schema detailed in [11], in which the attenuation factor α is expressed as:

$$\alpha = \alpha_B + \alpha_M + \alpha_F \tag{3}$$

These additive components respectively denote the relaxation of boric acid molecules below 1 kHz, the relaxation of magnesium sulfate molecules below 100 kHz and the viscosity of pure water. The equations take into account the pH, salinity (S) and temperature (T) of the water.

The boric acid component is calculated as follows:

$$\alpha_B = 0.106 \frac{f_1 f^2}{f^2 + f_1^2} e^{\frac{pH-8}{0.56}} \tag{4}$$

where f_1 is

$$f_1 = 0.78 \sqrt{\frac{S}{35}} e^{\frac{T}{26}} \tag{5}$$

Similarly, the magnesium sulfate component is:

$$\alpha_s = 0.52(1 + \frac{T}{43}) \frac{S}{35} \frac{f_2 f^2}{f^2 + f_2^2} e^{\frac{-z}{6}} \tag{6}$$

where f_2 is

$$f_2 = 42 e^{\frac{T}{17}} \tag{7}$$

and z is the depth of the water.

Lastly, the freshwater attenuation is expressed as:

$$\alpha_f = 0.00049 f^2 e^{\frac{T}{27} + \frac{z}{17}} \tag{8}$$

When additively combined, we get our final attenuation factor α. Our final attenuation is

$$I = I_0 e^{-0.023 \alpha d} \tag{9}$$

Where I is the final intensity plotted onto the output sonar image, I_0 is the raw intensity computed by the points where the sonar rays hit the objects in the simulation, and d is the distance travelled by the rays. We scale alpha by 0.023 for the purpose of unit conversion (from decibels/km to Nepers/km) and to take into account the proportionality relation of intensity and pressure, $I \propto p^2$. When combined with the speckle noise detailed below, we obtain highly realistic, physics based output.

F. Speckle Noise Addition

As noted by [11], speckle noise is an unavoidable feature of realistic sonar output. While speckle noise typically consists of two factors, one additive and one multiplicative, the additive factor is typically ignored as the multiplicative term dominates. We therefore apply the following equation to model the speckled noise, where g(x,y) refers to the final noised image, f(x,y) refers to the original image and n(x,y) refers to the noising function to generate the speckled noise.

$$g(x, y) = f(x, y) + f(x, y)\, n(x, y) \tag{10}$$

G. Medium Backscatter Noise Modeling

In order to replicate the impact of volumetric scattering in underwater sonar imaging, we integrate medium backscatter as a noise component that varies with distance. This approach reflects the physical reality in which acoustic energy is progressively scattered by suspended particles within the medium, leading to heightened interference at greater distances from the sonar source.

The intensity of noise at each pixel is adjusted according to its relative position along the propagation axis, with minimal noise present near the source and a linear increase towards the maximum range of interest. The resultant noise field is generated by scaling samples from a stochastic distribution of Rayleigh using a normalized attenuation factor that is dependent on range.

We define the noise contribution $N(x, y)$ at pixel location (x, y) as:

$$N(x, y) = \eta(x, y) \cdot \left(\frac{r(y) - r_{\text{near}}}{r_{\text{far}} - r_{\text{near}}} \right) \tag{11}$$

where $\eta(x, y)$ is a sample from a zero-mean noise distribution, and $r(y)$ is the radial distance associated with the image row y. The linear scaling term increases the noise amplitude with depth, bounded between r_{near} and r_{far}.

This formulation effectively emulates range-dependent acoustic interference, improving the realism of synthetic sonar imagery for perception and learning tasks.

IV. Results

We qualitatively analyze the results of our sonar simulation, DeepSonar, by evaluating it on both performance and interactivity.

We use a complex scene in Blender representing a rocky shallow water region with grass (similar to an underwater seabed) to test our simulator. The underlying ChatSim framework is used to navigate the scene, and DeepSonar is used to capture any areas of interest.

A. Performance

We run our experiments on NVIDIA RTX-A6000 GPU's to enhance the speed of our setup. DeepSonar is accelerated due to GPU utilization by Blender and Warp. The summarized processing times for each DeepSonar with and without GPU's

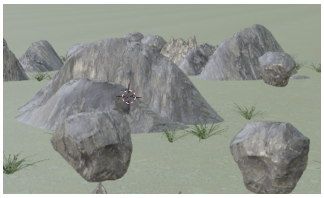
as well as Blainder [27] - a state-of-the-art Blender-based sonar plugin are elaborated in table I. As shown in the table, we achieve dramatically faster processing times in generating convincing imagery for complex scenery.

TABLE I: Processing times of different methods

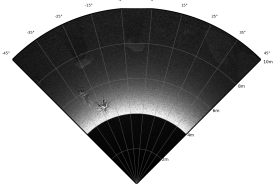
DeepSonar + Warp	DeepSonar	Blainder
2 ± 1 mins	10 ± 1 mins	12 ± 2 mins

B. Interactivity

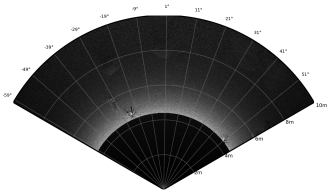
The ability to interact and dynamically control our simulation and sonar output is one of the primary goals for DeepSonar. Using ChatSim which also has LLM assistance built-in, allows DeepSonar to analyze and control the scene in Blender and even infer material properties which enhance the performance of sonar. A few examples of this include LLMs dynamically importing objects into the scene as the simulation progresses while also controlling the reflectivity of the mesh, we started the test by using our LLM to infer the ideal sonar parameters required to obtain the most realistic sonar output possible based on the scene. Figures 2a, 2b and 2c represent the image at three types of the simulation during the tests.



(a) RGB underwater scene in Blender



(b) Sonar Image with 2.1MHz frequency



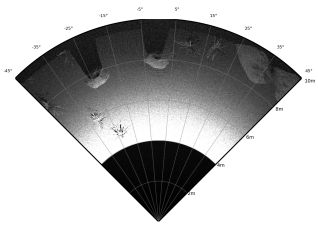
(c) Sonar Image with 3.1MHz frequency

Fig. 2: Pipeline for sonar image generation from 3D scenes.

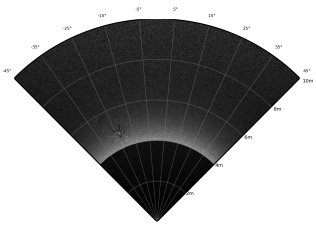
C. Environmental Configuration and Sonar Response Visualization

To evaluate DeepSonar's adaptability in simulating underwater environments, we conducted experiments by varying physical parameters like temperature, salinity, pH, and sonar frequency. These factors influence sonar signal propagation and are incorporated into our simulation using physics-based models. In each setup, we positioned a virtual sonar sensor in a customized Blender scene and traced volumetric rays across specific angles. The sonar intensities were calculated based on local surface interactions and adjusted for environmental parameters. By changing water properties and sonar frequency (e.g., 2.1 MHz), we observed variations in acoustic return strength, signal loss, and shadow formation, consistent with real sonar systems. We also added stochastic noise models, including speckle and medium backscatter noise, to enhance realism. Backscatter noise increases with range to simulate scattering from suspended particles, especially in turbid water, where distant surfaces become less distinct due to increased signal attenuation and scattering.

Figure 3 illustrates the sonar outputs produced from an identical scene under two distinct turbidity levels. The simulation characterized by elevated turbidity demonstrates diminished visibility and heightened haziness, especially in areas distant from the sonar source. This visual differentiation underscores DeepSonar's capability to authentically replicate underwater settings with varying water clarity and acoustic characteristics.



(a) Low turbidity



(b) High turbidity

Fig. 3: Comparison of sonar outputs from the same scene under low and high turbidity conditions. Higher turbidity increases volumetric backscatter and attenuation, reducing effective sensing range and contrast.

Overall, our results demonstrate that DeepSonar is capable of producing physically plausible sonar responses under a variety of environmental configurations. This flexibility makes it a valuable tool for generating domain-relevant training datasets and benchmarking perception algorithms in simulation environments that closely emulate real-world underwater conditions.

V. Future Work

For future work, we intend to capitalize on the highly flexible nature of DeepSonar to add physics-based simulation of intensity and noise using backscatter, reverberation and multi-path interference using advanced models like Born approximation and Bellhop [28] for beam tracing and acoustic pressure prediction respectively. This will allow users to access the full suite of tools currently only available to higher end simulations which are limited due to their choice of software.

We also plan to augment our simulation with more configuration options, allowing users or LLMs in their stead to choose the kind of physics models used in the sonar, based on the environment they aim to simulate - an option not available

to most current simulators, as well as adding additional graphical utilities to view the scene not reliant on Blender, allowing for a complete standalone experience. Currently, our DeepSonar pipeline depends on Blender-specific functions for scene processing. We will instead convert each scene into a 3D point cloud, enabling flexible, independent input processing.

REFERENCES

[1] L. Bjørnø, "Developments in sonar technologies and their applications," in *2013 IEEE International Underwater Technology Symposium (UT)*, 2013, pp. 1–8.

[2] M. Valdenegro-Toro, D. C. Padmanabhan, D. Singh, B. Wehbe, and Y. Petillot, "The marine debris forward-looking sonar datasets," 2025. [Online]. Available: https://arxiv.org/abs/2503.22880

[3] M. Valdenegro-Toro, A. Preciado-Grijalva, and B. Wehbe, "Pre-trained models for sonar images," 2021. [Online]. Available: https://arxiv.org/abs/2108.01111

[4] Y. Steiniger, D. Kraus, and T. Meisen, "Survey on deep learning based computer vision for sonar imagery," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105157, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622002718

[5] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[6] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).

[7] A. Palnitkar, R. Kapu, X. Lin, C. Liu, N. Karapetyan, and Y. Aloimonos, "Chatsim: Underwater simulation with natural language prompting," in *OCEANS 2023 - MTS/IEEE U.S. Gulf Coast*, 2023, pp. 1–7. [Online]. Available: https://arxiv.org/abs/2308.04029

[8] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi, "Objaverse-xl: A universe of 10m+ 3d objects," 2023. [Online]. Available: https://arxiv.org/abs/2307.05663

[9] Y. Wang, C. Wu, Y. Ji, H. Tsuchiya, H. Asama, and A. Yamashita, "2d forward looking sonar simulation with ground echo modeling," *arXiv preprint arXiv:2304.08146*, 2023. [Online]. Available: https://arxiv.org/abs/2304.08146

[10] J. Shin, S. Chang, M. J. Bays, J. Weaver, T. A. Wettergren, and S. Ferrari, "Synthetic sonar image simulation with various seabed conditions for automatic target recognition," in *OCEANS 2022 Conference Proceedings*, 2022, pp. 1–6.

[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, 2020.

[12] 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, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849317301371

[13] W.-S. Choi, D. R. Olson, D. Davis, M. Zhang, A. Racson, B. Bingham, M. McCarrin, C. Vogt, and J. Herman, "Physics-based modelling and simulation of multibeam echosounder perception for autonomous underwater manipulation," *Frontiers in Robotics and AI*, vol. Volume 8 - 2021, 2021. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.706646

[14] E. Potokar, S. Ashford, M. Kaess, and J. G. Mangelson, "Holoocean: An underwater robotics simulator," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3040–3046.

[15] M. M. Zhang, W.-S. Choi, J. Herman, D. Davis, C. Vogt, M. McCarrin, Y. Vijay, D. Dutia, W. Lew, S. Peters, and B. Bingham, "Dave aquatic virtual environment: Toward a general underwater robotics simulator," 2022. [Online]. Available: https://arxiv.org/abs/2209.02862

[16] J. Song, H. Ma, O. Bagoren, A. V. Sethuraman, Y. Zhang, and K. A. Skinner, "Oceansim: A gpu-accelerated underwater robot perception simulation framework," 2025. [Online]. Available: https://arxiv.org/abs/2503.01074

[17] E. Potokar, K. Lay, K. Norman, D. Benham, T. Neilsen, M. Kaess, and J. Mangelson, "HoloOcean: Realistic sonar simulation," in *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems, IROS*, Kyoto, Japan, Oct 2022.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[19] P. Hämäläinen, M. Tavast, and A. Kunnari, "Evaluating large language models in generating synthetic hci research data: a case study." New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3544548.3580688

[20] Z. Li, H. Zhu, Z. Lu, and M. Yin, "Synthetic data generation with large language models for text classification: Potential and limitations," in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. [Online]. Available: https://openreview.net/forum?id=MmBjKmHIND

[21] C. Gao, X. Lan, N. Li, Y. Yuan, J. Ding, Z. Zhou, F. Xu, and Y. Li, "Large language models empowered agent-based modeling and simulation: A survey and perspectives," *arXiv preprint arXiv:2312.11970*, 2024. [Online]. Available: https://arxiv.org/abs/2312.11970

[22] P. Natarajan, K. Basha, and A. Nambiar, "Synth-sonar: Sonar image synthesis with enhanced diversity and realism via dual diffusion models and gpt prompting," 2024. [Online]. Available: https://arxiv.org/abs/2410.08612

[23] X. Lin, N. Jha, M. Joshi, N. Karapetyan, Y. Aloimonos, and M. Yu, "Oystersim: Underwater simulation for enhancing oyster reef monitoring," in *OCEANS 2022, Hampton Roads*. IEEE, 2022, pp. 1–6.

[24] M. Hou, R. Yang, and F. Zhang, "Oceanchat: Piloting autonomous underwater vehicles in natural language," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1–7. [Online]. Available: https://www.mengxuehou.com/publication/icra_llm/

[25] 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, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070320300278

[26] S. J. Koppal, *Lambertian Reflectance*. Boston, MA: Springer US, 2014, pp. 441–443. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_534

[27] S. Reitmann, L. Neumann, and B. Jung, "Blainder—a blender ai add-on for generation of semantically labeled depth-sensing data," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2144

[28] S. Gul, S. S. H. Zaidi, R. Khan, and A. B. Wala, "Underwater acoustic channel modeling using bellhop ray tracing method," in *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2017, pp. 665–670.