A Computationally-Efficient 2D Imaging Sonar Model for Underwater Robotics Simulations in Gazebo

Kevin J. DeMarco
Georgia Institute of Technology
Electrical and Computer Engineering
demarco@gatech.edu

Michael E. West
Georgia Institute of Technology
Georgia Tech Research Institute
mick.west@gtri.gatech.edu

Ayanna M. Howard
Georgia Institute of Technology
Electrical and Computer Engineering
ayanna.howard@ece.gatech.edu

***Abstract*—Divers have to perform technical underwater tasks in dangerous and unstructured environments. To reduce a diver's workload and improve overall safety, an underwater robotic assistant (UWRA) could be deployed with the diver. The UWRA could assist the diver in navigation, quickly ferry tools from the surface, and carry underwater samples. To develop and test the autonomy required for such a UWRA, autonomy developers could benefit from a simulated underwater environment. Unfortunately, a real-time simulation for one of the most important sensors in underwater robotics, a forward-looking 2D imaging sonar, does not exist in robotics simulators. We developed a simulation for 2D imaging sonars that executes in real-time in the Gazebo robotics simulator. The 2D imaging sonar simulator was assessed by comparing the sonar images that it generated with sonar images generated by real sonar systems in similar environments. For example, we tested both the simulated and real sonar systems in environments with underwater man-made structures, divers, and varying terrain. The sonar simulator was also assessed based on its computational complexity and compared to other similar simulators with respect to sonar image generation rates. We found that our 2D imaging sonar simulator generated qualitatively realistic images and that it was computationally efficient enough to execute in real-time with the Gazebo simulator.**

I. Introduction

The underwater domain is a dangerous and complex environment for human divers. Often, divers have to monitor their own life support systems as they navigate to the work site or operate dangerous machinery. Military divers have to navigate for extended periods of time without surfacing or without using localization techniques that might give away their positions. Human divers have operated under these harsh conditions for decades with few advancements in technology. In fact, a diver performs the basic task of navigation by aligning the body with a compass and counting leg kicks (i.e., human-oriented dead-reckoning). It is proposed that an Underwater Robotic Assistant (UWRA) will improve the efficiency and safety of the diver's underwater operations by providing several key capabilities. For example, the UWRA can provide navigation assistance, ferry tools from the surface, enter structures too dangerous for human divers, and carry hazardous materials. However, it is costly and sometimes dangerous to test autonomous behaviors with physical robots. Simulators, such as the Gazebo Robotics Simulator, are used in the robotics community to validate and detect issues in autonomy software. Unfortunately, Gazebo has not been heavily used by the underwater robotics community. Thus, it lacks underwater scenarios and underwater sonar sensor models. In particular, Gazebo lacks a 2D imaging sonar similar to the BlueView P900. The sonar systems in the BlueView P900 series are especially conducive to underwater robotics due to their compact sizes and low-power requirements. High-frequency sonar systems are important to underwater robotics because optical cameras can be rendered useless at long-ranges due to the water's turbidity and low-light environments. To allow for the testing of underwater robotic systems in Gazebo we developed a computationally-efficient forward-looking 2D imaging sonar sensor that can operate in real-time.

II. Related Work

There are a number of techniques that can be used to generate synthetic sonar data. A common method used is ray-tracing, in which a series of rays are advanced out of the sonar transmitter until the rays come into contact with objects or boundaries [1] [2]. If the temperature of the water is assumed to be constant, the ray is an adequate approximation of a sonar beam. This is due to the fact that if the water temperature changes along the path of the sonar beam, the beam will bend. The process of projecting rays is repeated for an array until a three dimensional point cloud is constructed from the ray detection points. In the past, ray-tracing has been too computationally expensive to generate sonar images in real-time. Thus, researchers have been looking for alternatives to ray tracing and image rasterization, which has led to a process called tube tracing [3]. Tube tracing involves using multiple rays to form a polygon or footprint on a detected boundary. Tube tracing has the advantage of being less computationally expensive compared to ray tracing and the tube volume is more characteristic of sonar beams compared to ideal rays. The feasibility of using tube tracing for forward-looking sonar simulations was shown in [4].

In our previous work, we developed a sonar simulator in the Modular OpenRobots Simulation Engine (MORSE) [5]. While we were able to generate synthetic sonar data in MORSE, due to MORSE's heavy reliance on the Blender Game Engine, the sonar data generation rate was slower than real-time. The slower than real-time performance made testing the autonomy tedious and time-consuming. Since the development of our

978-0-933957-43-5 ©2015 MTS

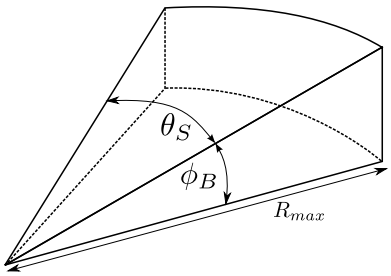

Fig. 1: 2D imaging sonar geometry.

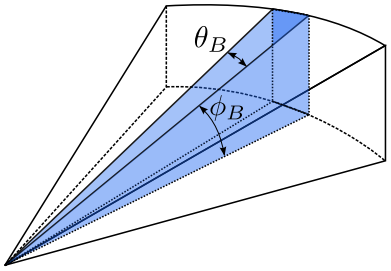

Fig. 2: A single sonar beam's geometry.

initial sonar model, a computationally efficient and high-fidelity method for generating 2D sonar images was introduced [6]. The method makes use of continuous geometric surfaces in the simulator to exactly calculate the bright and shadow zones in the sonar image. However, the time required to generate each sonar frame is large compared to other sensor models in robotics simulations: about 2.5 minutes in Matlab. The main thrust of this research was the improvement of our sonar simulator by borrowing from the concepts in [6] that provided the greatest increases in fidelity without greatly increasing the computation time.

III. Sonar Model

The aggregate sonar array geometry is shown in Fig. 1, and the geometry of a single sonar beam is shown in Fig. 2. The BlueView P900-45 2D imaging sonar has a field of view, θ_S, of 45° and a max range, R_{max}, of 100 m [7]. For a single sonar beam, the azimuth width, θ_B, is 1° and the elevation width, ϕ_B is 20°. The sonar head source, located at (x_s, y_s, z_s), concurrently emits 256 sonar beams, and when they encounter entities that reflect acoustic waves, located at (x, y, z), are reflected back towards the sonar head's receivers. Each of the sonar beams can be modeled by the acoustic version of the wave equation,

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = -\delta(x - x_s, y - y_s, z - z_s)s(t). \tag{1}$$

The wave equation relates the second-order differential equation of acoustic pressure, p, to a point source [8]. The point source is modeled by a normalized acoustic wave signal, $s(t)$, multiplied by the Dirac delta spatial distribution function, δ [2]. The wave equation assumes a constant speed of sound in water, c. If we assume there is an infinite water-mass of uniform density around the source, the acoustic pressure, p, can be expressed by (2).

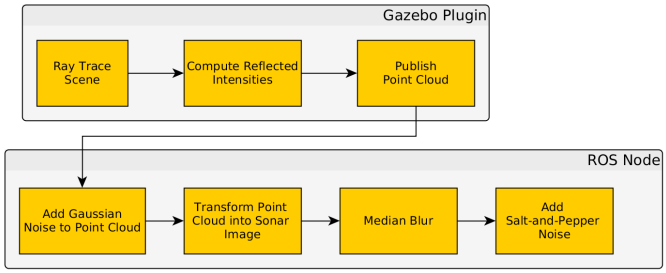

Fig. 3: Pipeline for generating the 2D sonar image.

$$p(\mathbf{x}; t) = \frac{s\left(t - \frac{r}{c}\right)}{4\pi r}. \tag{2}$$

In (2), r is the range from the source. Thus, as the acoustic wave propagates away from the source, the acoustic pressure level is attenuated proportionally to the inverse of the distance from the source. The acoustic pressure level has to be accounted for twice when using sonar heads that have co-located transmitters and receivers since the acoustic wave travels from the source to target and then back to the source. The distance between the sonar head source and targets can be calculated by applying knowledge of the speed of sound in water, and knowledge of acoustic pressure level attenuation provides a way to normalize the received acoustic signals. The bearing-to-target can be measured with the use of multiple sonar receivers arranged at different angles. Thus, the received sonar data can be transformed into a two-dimensional digital image, where each pixel is a function of the received wave's amplitude and time-delay. The pixels are plotted on a polar coordinate system, where the pixel's angle relative to the origin is determined by which receiver in the sonar array, θ_i, received the acoustic signal.

While 2D imaging sonars can provide an accurate image of the environment, they are plagued by several sources of noise. Both ambient noise in the water column and noise in the electronics of the sonar head contribute to false acoustic returns in the form of salt-and-pepper noise. Also, sonar waves are prone to multi-path reflection, which results in ghosts in the sonar images.

IV. Implementation

The objective of this work was to develop a 2D imaging sonar simulator that could generate realistic images in real-time. Other 2D imaging sonar simulators exist that can generate high-fidelity sonar images, such as the one in [6], but they require minutes of computation-time for each image. We require a sonar simulator that can generate a new image every second, at a minimum, to reduce the cost of simulating underwater autonomous systems.

The complete pipeline for the sonar simulator that we developed, shown in Fig. 3, bridges two domains: the Gazebo simulation environment and the Robot Operating System (ROS). We developed a custom Gazebo Sensor plugin that used ray-tracing to generate a point cloud based on the sonar's viewpoint

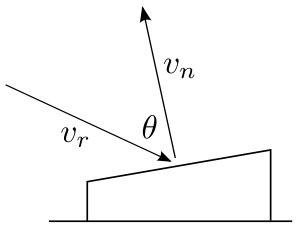

Fig. 4: The angle between the incident ray, v_r, and the vector normal to the object's surface, v_n, is θ.

of the 3D scene. This point cloud was then passed to the ROS environment, where a ROS node transformed the point cloud into a sonar image.

A. 3D Point Cloud Generation

Generating a 3D point cloud from a sensor is a common operation in the Gazebo environment. Other built-in Gazebo sensors, such as the depth camera, Kinect, and laser range-finder, make use of Gazebo's ray-tracing functionality to generate point clouds. While the ray-tracing functionality accurately generates a point cloud, the calculation of the reflected intensity values for each point makes use of an assumption that is not accurate for 2D sonar images. The Gazebo simulator assumes perfect reflectivity off of each object. To account for different objects having different absorption properties, each object's reflectivity is statically defined by the `laser_retro` in the object's SDF file. When a ray comes into contact with a simulated object, the reflected intensity value is simply the value defined by the `laser_retro` property. Unfortunately, this results in identical intensity values for all points on a single object. In order to create an accurate 2D sonar image, the reflected intensity values need to take the ray's angle-of-incidence and the distance from the sonar head into account. When a ray comes into contact with an object's surface, the reflected intensity is proportional to the cosine of the angle, θ, between the ray's vector, v_r, and the vector normal to the object's surface, v_n, as shown in Fig. 4. The angle, θ, is computed by taking the inverse cosine of the normalized dot product of v_n and v_r.

$$\theta = \cos^{-1}\left(\frac{v_n \cdot v_r}{|v_n||v_r|}\right) \tag{3}$$

To find the reflected intensity, I_R, we multiply the cosine of θ by the material's statically assigned reflectivity, I_m.

$$I_R = I_m \cos(\theta)\left(\frac{|v_r|}{R_{max}}\right)^2 \tag{4}$$

I_m has the same value as the Gazebo `laser_retro` XML tag. Finally, the reflected intensity is attenuated by taking the distance from the sonar's head into account. This is accomplished by squaring the ratio of the ray's length over the ray's possible maximum range, R_{max}.

After calculating the ray's reflected intensity, the point that represents the ray's collision with an object has to be computed. The ray's spherical coordinate position, $(|v_r|, \phi, \psi)$, is transformed into the sonar's body-fixed Cartesian position, (x, y, z), with the following:

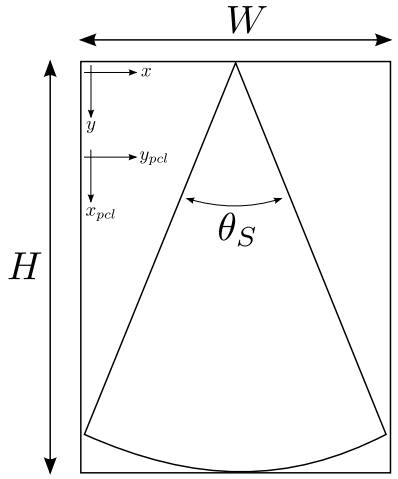

Fig. 5: Coordinate frames used during the first step of point cloud to sonar image transformation.

$$x = |v_r| \cos(\phi) cos(\psi) \tag{5}$$

$$y = |v_r| \cos(\phi) sin(\psi) \tag{6}$$

$$z = |v_r| \sin(\phi). \tag{7}$$

In the body-fixed frame, the forward direction is the x-direction, the side-to-side direction is the y-direction, and the up-and-down direction is the z-direction. Thus, in the spherical to Cartesian coordinate transformation, ϕ is the polar angle and ψ is the azimuthal angle. Finally, the ray's collision points and reflected intensity values are stored in a point cloud structure and published to the ROS environment.

B. Transformation from Point Cloud to Sonar Image

A ROS 2D sonar simulator node consumes the point cloud that is generated by the custom Gazebo sonar plugin. The ROS sonar node has access to several important characteristics of the sonar through the ROS parameter server. The characteristics are R_{max}, the sonar's maximum range, and θ_S, the sonar's field-of-view. A somewhat confusing aspect of transforming the 3D point cloud into the sonar image is the fact that the x-direction in the sonar image is actually the y-direction in the point cloud and the y-direction in the sonar image is the x-direction in the point cloud. The two coordinate systems are shown in Fig. 5 for clarification. Also, the sonar image is originally constructed with the assumption that the sonar head is located at the center-top of the sonar image. After all points in the point cloud are transformed into the sonar image, the sonar image is then rotated by 180° to represent the sonar image in its commonly viewed form.

During the point cloud to sonar image conversion, for entry i in the point cloud, (x^i_{pcl}, y^i_{pcl}), a sonar image point, (x, y) is calculated as follows:

$$x = \frac{W}{2} - \frac{y^i_{pcl}}{\sin(\theta_S) R_{max}} \sin\left(\frac{\theta_S}{2}\right) H \tag{8}$$

$$y = \frac{x^i_{pcl}}{R_{max}} H. \tag{9}$$

The sonar image is scaled based on the image's predetermined height, H, and width, W. In (8) and (9), the ratio of a point cloud position to its maximum possible value is calculated to scale the point into the sonar image's possible ranges. After the new point is calculated, a circle of radius one is drawn in the sonar image. Since the sonar image is initially constructed in grayscale, the point cloud's reflected intensity values, which range from 0 to 255, are directly used for the circle intensity values. After the entire grayscale sonar image is constructed, the Jet color map is applied to the sonar image. Finally, the completed sonar image is published to the ROS environment for other nodes to consume.

C. Noise Models

The images that 2D imaging sonars generate contain noise in the forms of Gaussian noise, salt-and-pepper noise, and multi-path propagation. Computing the multi-path propagation in a sonar simulator is computationally costly; thus, we decided to ignore multi-path at this point. However, we provided hooks for the user to add Gaussian noise to the sonar point cloud before converting the point cloud into a sonar image. Also, as shown in Fig. 3, after the sonar image is constructed, the user can modify the amount of median blur and salt-and-pepper noise that is applied to the sonar image via the ROS parameter server. We empirically determined the amount of Gaussian noise, median blur, and salt-and-pepper noise to apply to the sonar image based on our experiences with operating 2D imaging sonars in field experiments.

V. Experimental Setup

The sonar images that our 2D imaging sonar model generates is being qualitatively compared to data that was collected with a real 2D imaging sonar. In particular, the generated sonar images can be assessed based on their representations of acoustic intensity values and acoustic shadows. Also, the sonar model will be assessed based on the rate at which it can generate new sonar images. Three different underwater scenarios were created for testing the sonar simulator in the Gazebo environment. The first environment is based on literature that BlueView created to teach new users of their equipment how to interpret sonar images from 2D imaging sonars. The pylon environment is shown in Fig. 6 and consists of two bridge pylons and a box in between the two pylons. The second environment is the diver environment and consists of a simplified human diver model. As shown in Fig. 7, the diver is standing on a ground plane. This is an appropriate diver configuration for an industrial diver with weighted boots or an aquanaut. Finally, the third scenario is the ocean floor environment, as shown in Fig. 8. This environment consists of realistic ocean floor terrain, pylons, and a large sphere.

The time required to generate sonar images will be recorded and compared against another modern 2D imaging sonar simulator to assess the computational complexity of the sonar simulation.

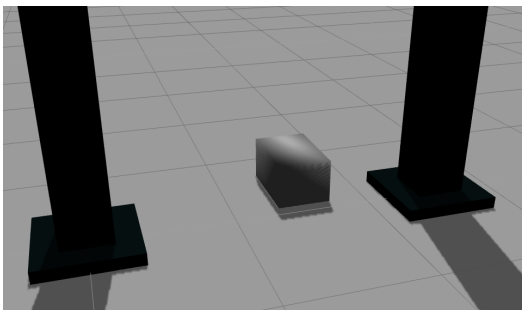

Fig. 6: The pylon environment.

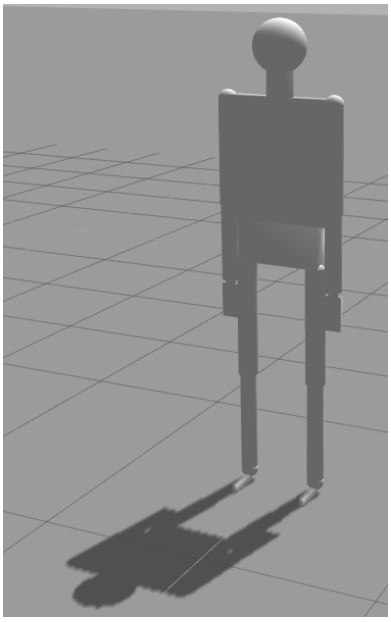

Fig. 7: The 3D model used to represent the diver in the Gazebo environment.

VI. Results

The 2D imaging sonar model that we developed was successfully used in the three simulated Gazebo environments. Sonar images from each of the scenarios are provided in this section.

A. Pylon Environment

The sonar image generated from the pylon environment is shown in Fig. 9. The bottom portions of the pylons and the box are clearly visible in the sonar image. Also, acoustic shadows created by the pylons that extend to the water's surface are present in the image. The simulated sonar image can be compared to a real sonar image, shown in Fig. 10. The sonar used to generate Fig. 10 had a field-of-view of 130°. Thus, to demonstrate the flexibility of our sonar model, we modified the sonar model's field-of-view parameter to generate the sonar image in Fig. 11.

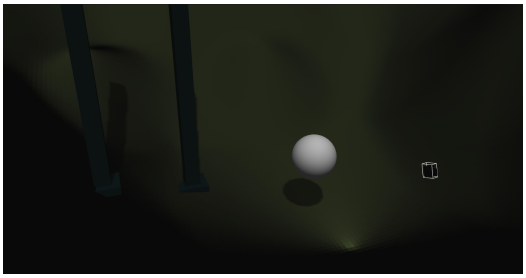

Fig. 8: The ocean floor environment.

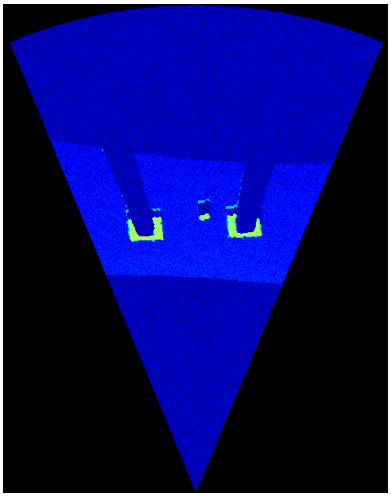

Fig. 9: Simulated 45° 2D sonar image of underwater pylons.

B. Diver Environment

The front of the diver model was ensonified in the Gazebo environment. The resulting sonar image is shown in Fig. 12. The diver is highly-reflective with a clear and long acoustic shadow. The sonar image is clean enough to show a distinction between the diver's legs. The simulated sonar image of a diver can be compared to a real sonar image of a diver standing on the ocean floor in Fig. 13. While the real sonar image consists of a highly-reflective diver and a long acoustic shadow, the real sonar image is significantly noisier. It is difficult to see the clear distinction between the diver's legs in the real sonar image. Within the diver environment, the simulated sonar was rotated by 90°, commonly referred to as the vertical configuration, to demonstrate the flexibility of our sonar model. The generated sonar model is shown in Fig. 14. The ground plane is represented by the vertical line on the left-hand side of the sonar image. The diver is visible and standing on the ground plane. Also, a gradient of reflected acoustic intensities is present across the diver's head in the sonar image. The sonar image from the vertical configuration can be compared with the real sonar image shown in Fig. 15. As in the simulated sonar image, the ground plane is represented as a vertical line on the left-hand side of the sonar image. Also, the diver is clearly standing on the ground plane. In the vertical configuration, the diver's two legs are visible and distinct.

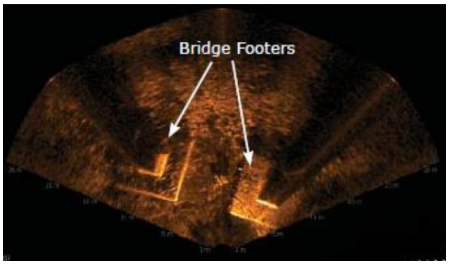

Fig. 10: A real sonar image created with a BlueView 2D imaging sonar. [7]

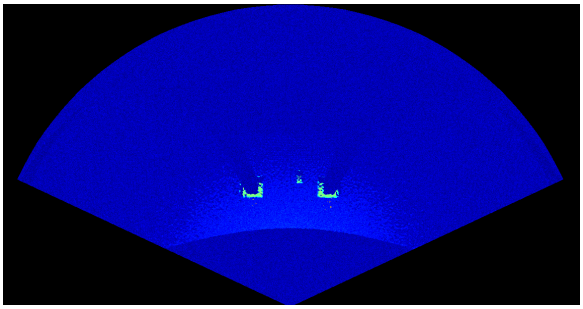

Fig. 11: Simulated 130° 2D sonar image of underwater pylons.

C. Ocean Floor Environment

The ocean floor environment was the last environment to be ensonified by the 2D imaging sonar model. The simulated sonar image is shown in Fig. 16. One of the pylons and the sphere are visible in the sonar image. Also, acoustic shadows are present behind the pylon and the sphere. As with the diver's head, a reflected acoustic intensity gradient is present across the sphere. In contrast to the other simulated environments, the background of the sonar image is distorted by the ocean floor's terrain.

D. Sonar Image Generation Rates

The rates at which the 2D imaging sonar simulator generated sonar images was determined by measuring the rates at which the ROS node published newly formed sonar images. The average times and associated standard deviations for generating sonar images in each of the three scenarios are tabulated in Table I. These rates can be compared to the rate listed by the authors of [6], which was about one frame every 2.5 minutes. An example of the sonar image created by the authors of [6] is shown in Fig. 17.

TABLE I: Sonar Image Generation Rates.

Environment	Average Rate (Hz)	Std Dev (sec)
Pylon	3.100	0.02020
Diver	3.034	0.02396
Ocean Floor	3.067	0.04119

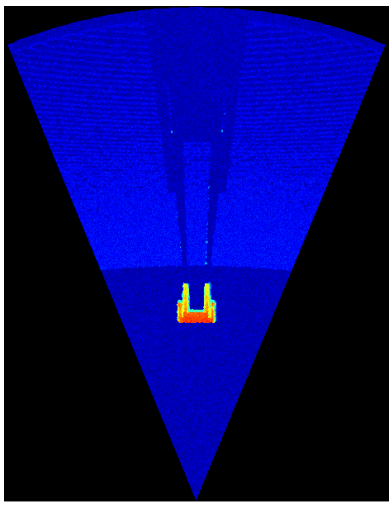

Fig. 12: A simulated 2D sonar image of the diver model in the horizontal configuration.

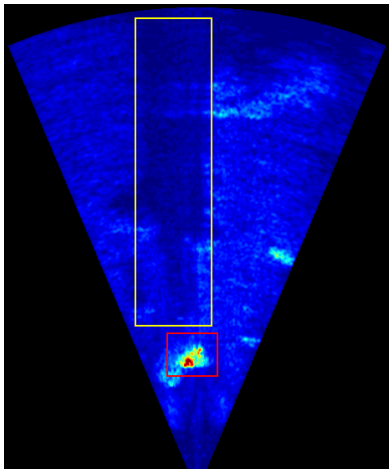

Fig. 13: A real 2D sonar image of a diver standing on the ocean floor. The diver, outlined in red, is highly-reflective. The diver's long acoustic shadow is outlined in yellow.

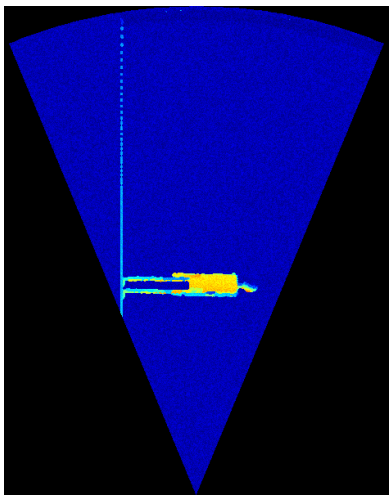

Fig. 14: A simulated 2D sonar image of the diver model in the vertical configuration.

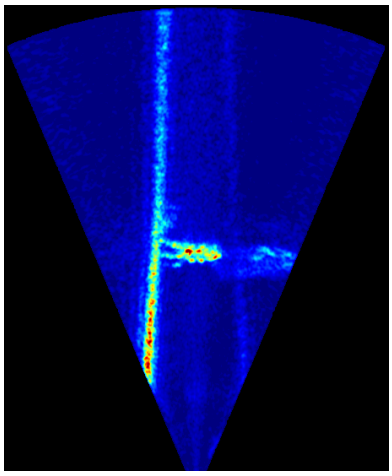

Fig. 15: A real 2D imaging sonar image of the diver in the vertical configuration.

VII. Discussion

We were successful in creating a real-time 2D imaging sonar simulator that was qualitatively similar to a real imaging sonar. The environments that were created provided enough variability to detect specific features commonly found in real sonar images, such as acoustic shadows, corners, and gradients of acoustic intensities. However, there were a few issues with our sonar model. For example, in Fig. 12, the distinction between the rays used to create the point cloud are noticeable at long-ranges. This is an inherent problem with statically-defined ray tracing algorithms. As the rays move away from the sonar head, the ray-tracing resolution decreases. The opposite problem is encountered near the sonar head of Fig. 12, where the diver's model is well-defined. However, when compared to the real sonar image of a diver in Fig. 13, it is obvious that the real sonar image produces a much noisier version of the diver's form. In the real sonar image, when the diver is standing on the ocean floor, the diver appears to be an indistinguishable blob. However, our sonar simulator produced a fairly accurate image of the diver when the sonar was in the vertical configuration in Fig. 14. When the simulated sonar image in Fig. 14 is compared to the real sonar image in Fig. 15, there are two features missing in the simulated sonar image: a ghost image of the ground plane and a highly-reflective entity above the diver's head. The lack of a ghost ground plane can be explained by the fact that we did not implement a multi-path propagation model. The entity above the diver's head in the real sonar image is actually the diver's exhaled bubbles. If we were to implement a bubble generation model in the Gazebo simulator, the simulated sonar image would also contain reflections due to exhaled bubbles. Nevertheless, the simulated sonar image in the vertical configuration was acceptably realistic. Also, the reflected acoustic intensity gradient on the diver's head is a direct result of taking a ray's angle-of-incidence into account when calculating intensity values.

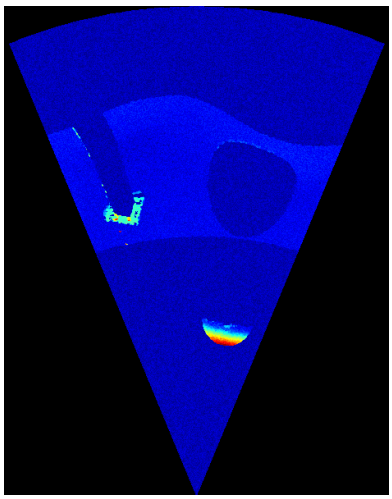

Fig. 16: A simulated sonar image of the ocean floor environment.

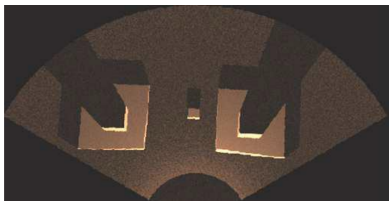

Fig. 17: A sonar image of the pylon environment created by another research group's sonar model.

While the 2D imaging sonar model we developed might not be high enough fidelity to generate synthetic data for machine learning algorithms, the sonar simulator does provide the researcher with a basic idea of how a particular 3D model will be represented when ensonified with a 2D imaging sonar. The sonar simulator we developed could even be used for early testing of visual odometry algorithms since many of those algorithms only require basic feature detection, such as corners and lines.

What is notable about this sonar simulator is the fact that it can generate a qualitatively realistic sonar image in real-time. In all three environments, the sonar simulator consistently generated new images at a rate over 3 Hz. This cannot be said about other 2D imaging sonar simulators in its class that can only generate a single frame with a time-scale on the order of minutes. Our computationally-efficient sonar simulator enables autonomy developers to quickly prototype their systems.

Future work with this 2D imaging sonar simulator could focus on a few topics. For example, a more accurate characterization of the sonar's noise model could be developed. However, for now, future simulations could probably be improved by increasing the amount of Gaussian noise that is applied to the sonar's point cloud. Also, instead of merely drawing small circles to depict sonar reflections, polygons could be drawn based on nearest-neighbors in the point cloud. By drawing polygons between point cloud neighbors, the visualization of discrete points in the point cloud will be obscured, resulting in a more realistic sonar image.

VIII. Conclusion

The original purpose for the development of this 2D imaging sonar simulator was to enable the testing of an underwater robotic diving assistant in simulation. A real-time and qualitatively realistic 2D imaging sonar simulator was successfully created and tested in Gazebo. However, we learned two important ideas when developing the 2D imaging sonar model and collecting real sonar data. The first idea was that the acoustic shadow of an object could provide just as much information about the object being ensonified as the highly-reflective foreground pixels. Second, we learned that two sonars placed in an orthogonal configuration could drastically improve diver and object classification. When the diver is stationary and in a vertical pose, the vertically-oriented sonar can detect distinctly human features. However, when the diver is moving and in a horizontal pose, the horizontally-oriented sonar has a better chance of detecting features of the diver's fins. The development of the 2D imaging sonar simulator is necessary for the future development of an underwater robotic assistant.

IX. Acknowledgment

The authors would like to thank Dr. Noel E. Du Toit from the Naval Postgraduate School (NPS) for allowing them to use his system to collect 2D imaging sonar data. Also, the authors would like to thank the Georgia Tech Research Institute (GTRI) for supporting this research.

References

[1] J. M. Bell and L. M. Linnett, "Simulation and analysis of synthetic side-scan sonar images," in *Radar, Sonar and Navigation, IEE Proceedings-*, vol. 144, 1997, pp. 219–226.

[2] E. Coiras and J. Groen, "Simulation and 3d reconstruction of side-looking sonar images," *Advances in Sonar Technology*, pp. 1–14, 2009.

[3] D. Gueriot, C. Sintes, and R. Garello, "Sonar data simulation based on tube tracing," in *OCEANS 2007-Europe*, 2007, pp. 1–6.

[4] D. Guriot and C. Sintes, "Forward looking sonar data simulation through tube tracing," in *OCEANS 2010 IEEE-Sydney*, 2010, pp. 1–6.

[5] K. J. DeMarco, M. E. West, and A. M. Howard, "A Simulator for Underwater Human-Robot Interaction Scenarios," in *OCEANS 2013*. IEEE, Sep. 2013, pp. 1–10.

[6] H. SA, K. LEBLEBCOLU, and G. B. AKAR, "2d high frequency forward-looking sonar simulator based on continuous surfaces approach."

[7] "BlueView Imaging," 2014. [Online]. Available: http://www.videoray.com/ homepage/ options/sonar/ blueview-imaging-sonars.html

[8] D. S. Drumheller, *Introduction to wave propagation in nonlinear fluids and solids*. Cambridge University Press, 1998.