Imaging sonar simulator for assessment of image registration techniques

José E. Almanza-Medina
Department of Electronic Engineering
University of York
York, UK. YO10 5DD
jeam502@york.ac.uk

Benjamin Thomas Henson
Department of Electronic Engineering
University of York
York, UK. YO10 5DD
benjamin.henson@york.ac.uk

Yuriy V. Zakharov
Department of Electronic Engineering
University of York
York, UK. YO10 5DD
yury.zakharov@york.ac.uk

***Abstract*—This work focuses on building a Forward-Looking sonar simulator capable of generating large volumes of ground truth data to test algorithms such as: novel image registration techniques for trajectory estimation, three-dimensional reconstruction or to be used as training data for machine learning algorithms. The simulator is capable of generating realistic data sets of images and providing ground truth data with the exact position and attitude of the sonar related to objects in a test scenario. The sonar simulator is developed using the Unity software platform. This work also shows an example application. Simulated image data sets for different sonar trajectories and seabed textures were created. These data sets are used by an attitude-trajectory estimation method and a quantitative analysis of the method is presented.**

***Index Terms*—forward-looking sonar, image registration, sonar simulator**

I. Introduction

In recent years there has been a growing interest in underwater environments and considerable effort in creating and developing technologies to explore and exploit them. Accurate detection, identification and localization of underwater objects is difficult due to the conditions experienced underwater, such as poor visibility, poor propagation of radio waves and constant motion of the water body.

The utilization of acoustic imaging techniques presents advantages in scenarios where optical cameras have a poor performance due to scarce illumination or turbidity of water [1]. Two-dimensional (2D) Forward-Looking Sonars (FLSs) can generate high-resolution images. Regardless the technical specifications defined by each manufacturer, the image formation process is mostly identical [2]. This type of image is created when an acoustic pulse is emitted from the sonar and reflected by objects in the scene. The reflections are collected by a sensor in the sonar from different azimuth (θ) and elevation (ϕ) directions, revealing the distance, position and acoustic reflectivity of the objects. The information is compressed into a 2D image by losing the elevation dimension in the process (Fig. 1). Using polar coordinates, the intensity of a pixel on a sonar image can be represented by [3]

$$I_s(r,\theta) = \int_{\phi_1}^{\phi_2} \beta(\phi) V_s(r,\theta,\phi) D_s(r,\theta,\phi) d\phi, \qquad (1)$$

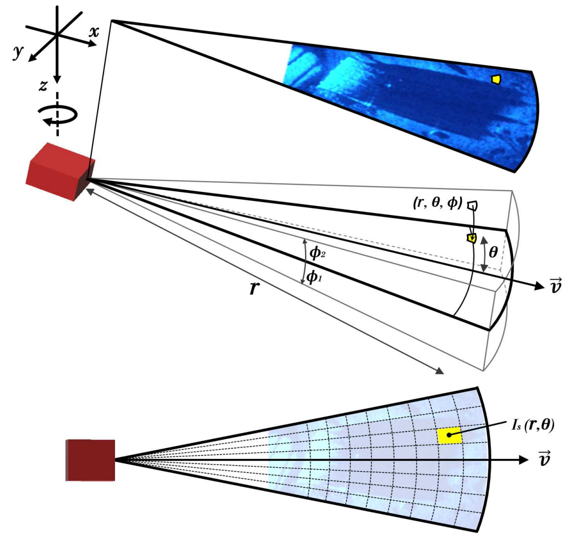


Fig. 1: Image formation geometry.

where a single pixel is formed by the contribution of all the intensities from points in three-dimensional (3D) space (r, θ, ϕ) over the aperture $[\phi_1, \phi_2]$. $\beta(\phi)$ is a function related to the beam pattern, $V_s(r,\theta,\phi)$ is a measure related to objects reflectivity and $D_s(r,\theta,\phi) = \frac{\vec{v}\cdot\vec{n}_{r\theta\phi}}{\|\vec{v}\|\|\vec{n}_{r\theta\phi}\|}$, is the cosine of the angle between the direction of the beam $\vec{v}$ and the surface normal $\vec{n}$ at point (r, θ, ϕ).

The development of new models and techniques based on sonar imaging often requires large collections of data taken from a controlled test scenario and accurate knowledge of position and attitude of objects and sensors in the scene. For that reason underwater simulators have been developed where all these features can be easily manipulated and tuned.

Simulating aquatic environments can be a challenging task. The process of recreating realistic data sets becomes complex when including such factors as object motion.

In [4], a novel open source simulator is presented. It uses the OpenSceneGraph (OSG) programming interface and specialist libraries for realistic underwater rendering. This simulator uses Robot Operating System (ROS) for interaction between the user, vehicles and sensors. The authors created different simulation sensors, but none of them is for an imaging sonar.

In order to create test samples for a 3D reconstruction algorithm, [5] built an imaging sonar sensor using the underwater simulator from [4]. The simulator uses a ray tracing technique which obtains the angle of incidence between the sonar beam and the normal to the object (Lambertian model [6], [7]). It considers a constant reflection coefficient for all the objects, i.e., the intensity of the reflected beam is only affected by the angle of incidence and not by the reflectivity of the material. The beam pattern is shaped as a sinc function and noise is added according to a noise model for the BlueView P900 imaging sonar [8]. Although this simulator is open source software, one of difficulties in its use is the background knowledge required to operate it, specifically, as it requires advanced skills in C++, Python and XML languages besides some experience with ROS. Furthermore, it only works on Linux systems.

The *Rock-Gazebo framework* developed in [9] is used by [10] to build a simulator for a Mechanical Scanning Imaging Sonar (MSIS) and a FLS. The simulated sensor emits pulses into the scene and stores three parameters in memory using the 8-bit RGB channels: (i) the distance to the object; (ii) the intensity of the reflected signal, which is proportional to the incidence angle of the beam and normal to the surface; and (iii) the angle of arrival. Speckle noise (modelled by a Gaussian distribution) is applied to the final image. However, noise is added to every pixel with no distinction of the origin of the pixel, e.g., if it was generated by an acoustic shadow or an object, they are treated equally.

The work [11] proposes an approach to development of an imaging sonar simulator that is focused on adding three types of distortions to the images. It considers the speed of sonar motion, the navigation course deviation and the optical noise on the pixels of the image. The imaging procedure uses the ray tracing approach and no environment conditions are considered such as different values of reflectivity on the surface of the objects.

Another image simulation approach consists in a less conventional combination of ray tracing with a frequency domain algorithm [1]. Similar to other methods, three parameters are collected to construct the image: (i) Euclidean distance from the source to points in the scene; (ii) reflected intensity; and (iii) a number associated with each of the polygonal surfaces that make the objects. The number is used to avoid overlapping (summation) of rays that return from the same area when producing the image. This reduces the computational cost. However, if an object presents a large surface or a low number of polygons, most of its surface will be ignored. There is a risk of missing information and producing incomplete images. Another feature to remark is the angle ray separation which can choose one of two configurations: (i) the same angular separation or (ii) the same separation on the seabed. In a similar way, another scanning strategy is described in [12]. Rays in elevation angle are non-uniformly distributed to scan only the points on the surface of the objects that offer more information of the shape. However, this ray separation approach may require greater computation when analyzing the object shape to chose the right scanning pattern.

The simulator in [13] uses two methods for sonar imaging: (i) a ray tracing method and (ii) a method that uses the rays to create *tubes* for modelling the propagation of acoustic waves. When a tube is emitted from a transmitter, it produces a footprint on the surface of the objects. The intersection of two or more footprints from different tubes represents one path of the ray traveling in the space, allowing the consideration of multipath in simulations. This simulator also defines two concepts for textures in the objects that, when scanned, help to produce more realistic images: macro-textures, which are small deformations in the surface of the objects, and micro-textures, which are represented by variations in the color or image attached to the surface of the object.

Summarizing the features of previous works and considering that the aim of the new desired simulator is to supply large volumes of samples to feed image registration techniques, the following characteristics are needed to be considered:

- **Reflectivity:** It should consider material properties on the surface of the objects in the scene. Therefore macro and micro-textures similar to that in [13] should be used.
- **Angle of incidence:** Total reflectivity should consider the angle between rays and surface of objects.
- **Noise:** Model of noise that is present on FLS images. It should be able to be adapted or customized to the simulated sonar.
- **Computational and time consuming cost:** Since it is expected to generate large volumes of data.
- **Adaptability:** It can be easily modified to simulate different types of FLS.
- **Mobility:** Sonar motion paths must be set and controlled easily in order to scan the scenes from multiple points of view.
- **Multi-platform:** The simulator can be used in different operating systems.

The sonar simulator proposed in this work considers aspects mentioned above in order to produce realistic sonar images from underwater environments. The generated data set can be used for testing image processing techniques while having a low computational cost. Equal separation between rays is used to reduce the processing time when scanning the scene rather than a non-equally spaced scheme. The present work focuses on building an imaging sonar simulator using the increasingly popular and low-cost platform Unity [14]. Unity is a well-tested development platform capable of creating 3D scenarios, providing a large programming toolset, an intuitive workspace and realistic manipulation effects controlled by its own physics engine [15]–[17]. The physics engine is a software program that enables simulating motions and reaction of objects under the laws of physics, in a similar way to the real world. Unity initially focuses on the creation of videogames. However, it has also been used in education, medical, animation, construction design and industrial applications [18]. Project testing and editing can be done *on-the-fly*, i.e., while a simulation is running. Programming codes are written in C# language.

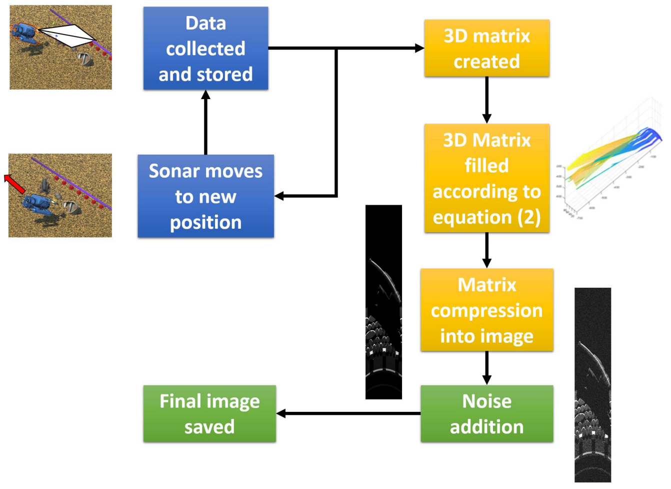
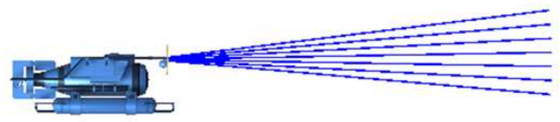
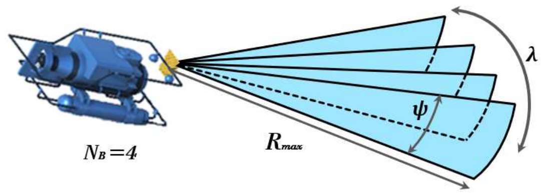

Data collected and stored
3D matrix created
3D Matrix filled according to equation (2)
Sonar moves to new position
Matrix compression into image
Final image saved
Noise addition

Fig. 2: Image formation process.



(a) Multiple rays forming a beam. Only seven rays are shown here but a larger number is used to generate images, e.g., 700.



(b) Multiple beams forming the FOV of the sonar. For demonstration purposes only four beams are displayed, the exact number N_B depends on the sonar being simulated. λ represents the azimuth angle or aperture, ψ the elevation angle and R_{max} is the maximum measurable range from the sonar.

Fig. 3: Description of the sonar FOV composition.

They can operate directly with the objects in the scenes allowing a highly efficient management of the environment. Moreover, Unity platform can be run on Windows, Mac and Linux operating systems.

The remainder of this paper is organized as follows. Section 2 describes the design of the imaging sonar simulator. Section 3 presents results and evaluation of an attitude-trajectory estimation method using data sets generated by the simulator. Finally, conclusions are given in Section 4.

II. Sonar simulator design

The sonar simulator is divided into three parts. The first part emulates the underwater scenario, defining the sonar and its motion and acquiring the data required to produce images. The second part processes the information collected in the first part to generate a set of images. The third part adds noise to the images. The whole process is summarized in Fig. 2.

A. Generation of the underwater scenario and data acquisition

The first part is based on the Unity platform. The simulator developed in this work is based on a ray tracing technique where a series of rays are emitted from the sonar at certain angles that then collide with the objects in the scene. When a ray hits an object, it stores in memory three features needed to produce the image: (i) Euclidean distance from the sonar to the collision point, i.e., the length of the ray when it hits an object; (ii) the pixel color information (in RGB color system) of the point being hit. The color represents a measure of the acoustic reflectivity of the material; and (iii) the angle of incidence between the ray and the vector normal to the surface of the object following the Lambertian model.

Multiple rays are equally spaced in elevation angle over a beam (Fig. 3a). In order to obtain more realistic imagery within a low simulation time, a trade-off between resolution and computation/storage cost can be established.

The sonar field of view (FOV) is formed by a set of beams uniformly separated in the azimuth aperture (Fig. 3b). The number of beams N_B and the aperture angle λ are set according to specifications of the sonar that is emulated. The elevation angle ψ, the number of rays per beam N_R, the maximum measurable range R_{max}, position and orientation of the sensor, defined by six degrees of freedom are defined in a Unity dashboard created to control the sonar.

The motion-scanning algorithm of the sonar defines a trajectory by specifying start and end points in the 3D scenario ($P_i = [x_i, y_i, z_i]$ and $P_o = [x_o, y_o, z_o]$, respectively) and the number of image frames to be acquired N. N points equally spaced on the trajectory are calculated and an image is obtained from each of these points. The sonar can rotate while it moves by specifying initial and final orientations angles ($P_{rot_i} = [\alpha_i, \gamma_i, \epsilon_i]$ and $P_{rot_o} = [\alpha_o, \gamma_o, \epsilon_o]$, respectively). For each position in the trajectory, an orientation angle is calculated using the points P_{rot_i}, P_{rot_o} and N. For simplicity, these angles are also equally spaced.

For each position, the sonar stores the data that was collected by the rays in three matrices of size $N_B \times N_R$. The first matrix $\mathbf{R}$ contains the Euclidean distance from the sonar to each of the points hit by the rays. The second matrix $\mathbf{C}$ stores a value between 0 and 1. For this simulator, only red component from RGB baseline is saved, this color information is used as a measure of reflectivity on the surface of the object, the redder a pixel is, the more reflective it is, and if it is darker, it absorbs more energy and its reflectivity is low. The third matrix $\mathbf{L}$ keeps the cosine of the angle of incidence between the ray and a vector normal to the surface of the object. These matrices represent the functions in (1), where $V_s(r, \theta, \phi)$ and $D_s(r, \theta, \phi)$ correspond to $\mathbf{C}$ and $\mathbf{L}$, respectively, and for simplicity, the beam pattern $\beta(\phi)$ is set to one. $\mathbf{R}$ is used to map the information of the objects to the correct position in the image. The information about the position and orientation of the sonar at each point of the trajectory is also stored to be used as the ground truth.

B. Image formation

For each position in the trajectory, an image is generated from the data acquired in the previous step. This process is described for a single frame, but it is repeated for each of N desired images. A 3D matrix $\mathbf{I}$ of size $N_R \times M \times N_B$ is created (elevation, range and aperture, respectively). M is defined by the emulated sonar and represents the number of different possible range values, i.e., the number of pixel rows in the final image. Initially, the matrix is filled with zero values and only some of its elements are replaced by an intensity value at specific coordinates that are defined by

$$I\left(\left\lceil \frac{R(i,j)M}{R_{max}} \right\rceil, i, j\right) = C(i,j)L(i,j), \qquad (2)$$

where i and j are indexes that represent coordinates in the image matrix and $\lceil a \rceil$ is the ceiling operation that returns the nearest integer greater than or equal to a number a.

The matrix $\mathbf{I}$ is compressed into a square matrix to create the final image $\tilde{I}$ of size $M \times N_B$. Compression is performed by adding all the values in each single column:

$$\tilde{I}(m,n) = \sum_{k=1}^{N_R} I(k,m,n). \qquad (3)$$

Therefore the elevation axis of the 3D matrix is lost and only aperture and range axes remain. If resulting values from the summation exceed a threshold, the value is truncated to that threshold, e.g., 255, since pixel values in this type of images have a scale from 0 to 255 (from dark to bright, respectively).

C. Adding noise to images

After the image has been produced, its pixels are distorted by adding noise according to a model using the following criteria. If the pixel lacks the information about objects in the scene, i.e., it is of a zero value, then it is considered to be produced by an acoustic shadow or the water column and being affected only by electrical noise inherent to the sonar; thus the pixel value is modified according to a Gaussian distribution model [19]–[21]. Otherwise, if the pixel created using the information from object surfaces, the Rayleigh distribution is applied, since it corresponds to a surface represented by many small scatterers [20]–[23]. The parameters of Gaussian and Rayleigh distributions are chosen according to measures obtained from real imaging sonar devices. The result of adding noise is the final image $\hat{I}$. Examples of simulated images obtained can be seen in Fig. 4.

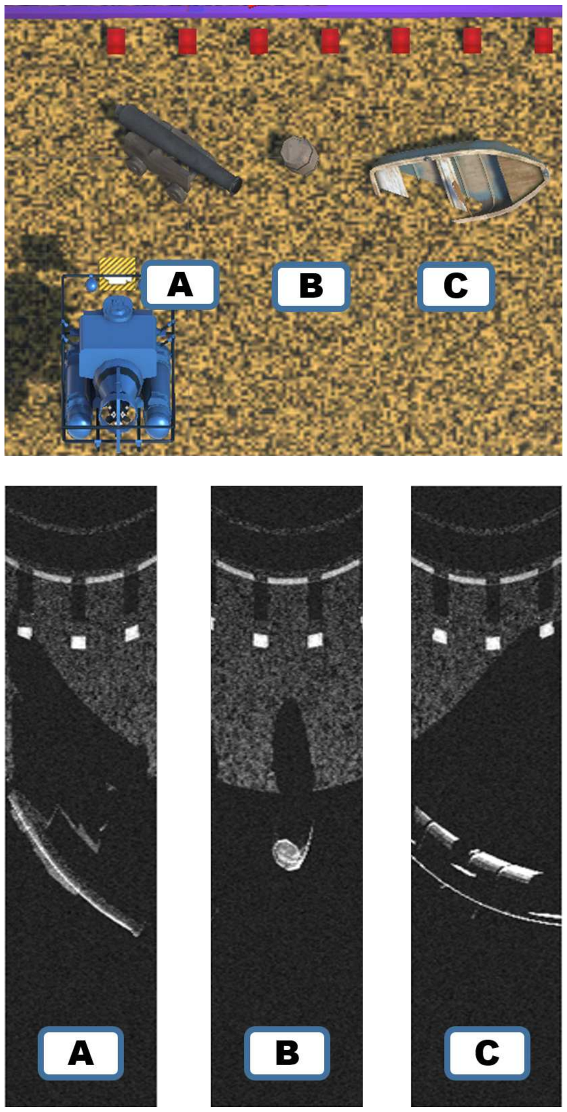


Fig. 4: 3D testing scenario and resulting images acquired from positions A, B and C in front of a cannon, barrel and boat respectively, imitating DIDSON 300 sonar features [24].

III. Image registration using the simulated images

This section presents results of applying an image registration method on multiple data sets of simulated images. Image registration is a process that compares a reference image with one or more images of a scene from different points of view at different moments in order to obtain information of the change in the imaging conditions [25]. This paper uses the method for trajectory estimation and mosaicing developed in [26], which is capable of estimating pixel displacements between two frames with subpixel accuracy. The displacement information is used to estimate the attitude and trajectory of the imaging sonar sensor. In this estimation, three parameters of the sensor motion are calculated: rotation around z axis and translation in x and y axes according to directions displayed in the coordinate system of Fig. 2. Finally, a mosaic is built using the image data set and the attitude-trajectory estimates are obtained. This method was applied on the data sets for validating its performance using the ground truth provided by the simulator.

The data sets were constructed using three different scanning trajectories over a scene defined as follows.

1) *Alternation between translation in x and y directions*: 250 frames along 5 m in y direction, 100 frames along 1.2 m in x direction, 250 frames along 5 m in $-y$ direction, 100

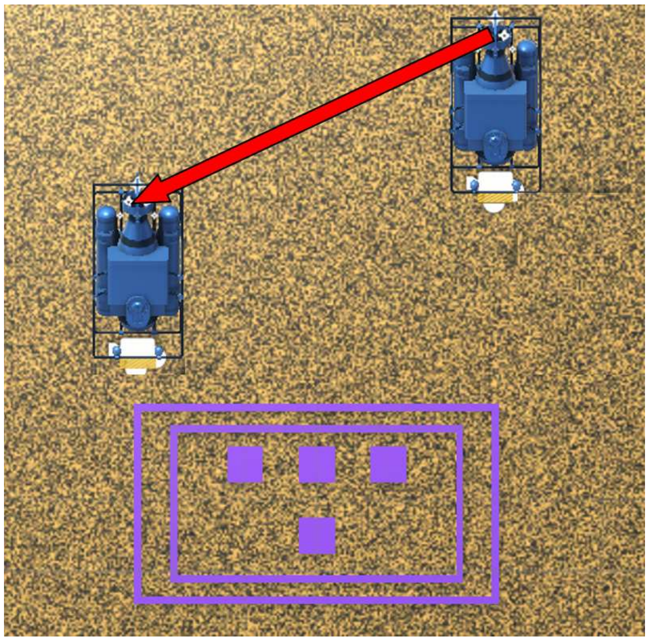


Fig. 5: Testing scenario seen from above. Gaussian noise as seabed texture. Red arrow represents the trajectory of the sonar moving 5 m in y axis and 2.3 m in x axis with no rotation.

frames along 1.1 m in x direction and 250 frames along 5 m in y direction, 950 frames in total.

2) *Translation in xy direction*: 300 frames along 2.3 m and 5 m in x and y directions respectively.
3) *Rotation over z axis*: 300 frames over $120°$ around z axis on the same position.

The testing scenario can be seen in Fig. 5. It was built using 3D geometric shapes. Four cuboids of size $50 \times 50 \times 2.5$ cm forming a "T" shape and four pairs of parallel bars enclosing the cuboids were set over the seabed. This pattern was chosen since it is easy to recognize when the mosaic is built.

Six seabed textures (see Fig. 6) were set on the bottom, which, when combined with the three trajectories, make a total of 18 data sets. Three of these seabed textures were produced using random generators: the first one was produced by a Gaussian noise generator with a pixel value independent from its neighbours. Two textures were produced using a Perlin random process [27], [28], which generates natural appearing textures. An image produced using this method provides a correlation between a pixel and its neighbours that can be seen as smooth transitions. For this work, the Perlin textures are produced with a low and a high correlation. The other three seabed textures were created by taking photographs from real surfaces. The purpose of using real surface textures is to obtain realistic representations of the seabed on the images.

The sensor to be simulated is the DIDSON 300 [24] with FOV of $29° \times 14°$ (aperture and elevation angles, respectively), 96 beams, image size of 512×96 pixels and intensity range between 0 and 255. The images generated by the sonar are truncated in the range to zoom into the objects that appear in the scene only. This interval is chosen because it

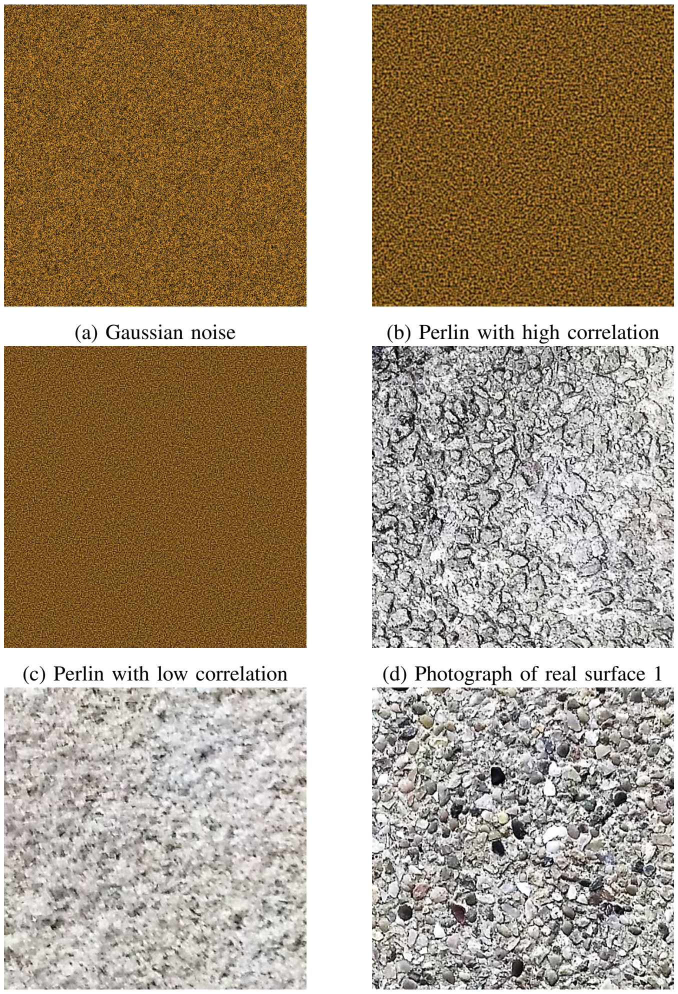

(a) Gaussian noise
(b) Perlin with high correlation
(c) Perlin with low correlation
(d) Photograph of real surface 1
(e) Photograph of real surface 2
(f) Photograph of real surface 3

Fig. 6: Seabed textures used in the simulator.

encloses the area in front of the sonar where the objects are. Noise parameters are set according to measurements from real images in [29]. The Gaussian noise added to the pixels in the image has a mean of 35 and a standard deviation of 8, whilst the Rayleigh distribution has an offset of 50 and its peak is reached at 70.

Fig. 7 shows the absolute error of motion estimates, which represents the difference between the estimated and actual displacement between two consecutive images. This was obtained using the Gaussian seabed texture and the sonar trajectory type 2.

Tables I, II and III summarize the root mean square error (RMSE) obtained for each of the data sets for the three types of motion. These results show that the trajectory estimation method has a higher performance when the bottom texture presents more independence between pixels, which is the case of the Gaussian noise texture. The image registration process identifies motion of a pixel from one frame to another. However, if a pixel value is very similar to the values of pixels around it and considering that noise has been added, it is

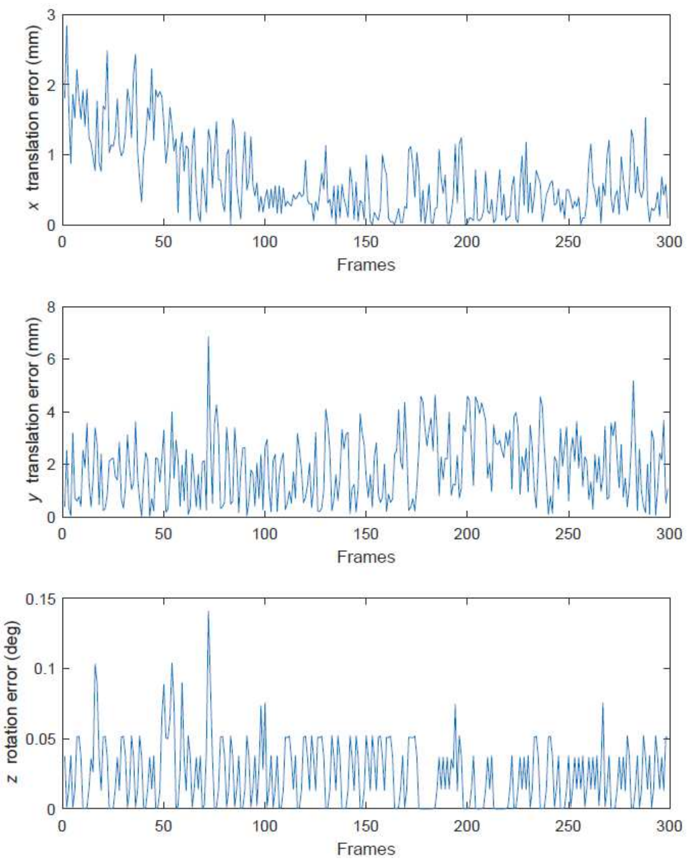


Fig. 7: Absolute errors from displacement estimation between consecutive frames for data set with motion motion type 2 and Gaussian noise in the seabed texture.

possible that the displacement is estimated inaccurately. This problem appears in textures with the Perlin noise, where pixel values have a dependance to their neighbours. Another point to remark is that estimation is more accurate in x translation compared to y translation. This is due to the similarity between y translation and z rotation when pixels move from one point to another between consecutive frames in the polar coordinate system. In this system, when the object pixels change their positions in x direction, they move vertically. However z rotation of objects can be seen as horizontal movement while y translation can be seen as a rotation as it is shown in Fig. 8. Therefore, it is more likely that the image registration algorithm interprets the sonar movement in y direction as a z rotation, producing a higher estimation error.

A mosaic is built when the attitude-trajectory estimation is completed. Fig. 9 shows an example of such a mosaic for the data set with the Gaussian seabed texture and motion type 2.

IV. Conclusions

The imaging sonar simulator presented in this paper is able to generate realistic data sets of images from 3D scenarios, exploiting capabilities of the Unity platform. Sonar features such as FOV, maximum range and number of beams can be adjusted in accordance with real sensors. A large number of

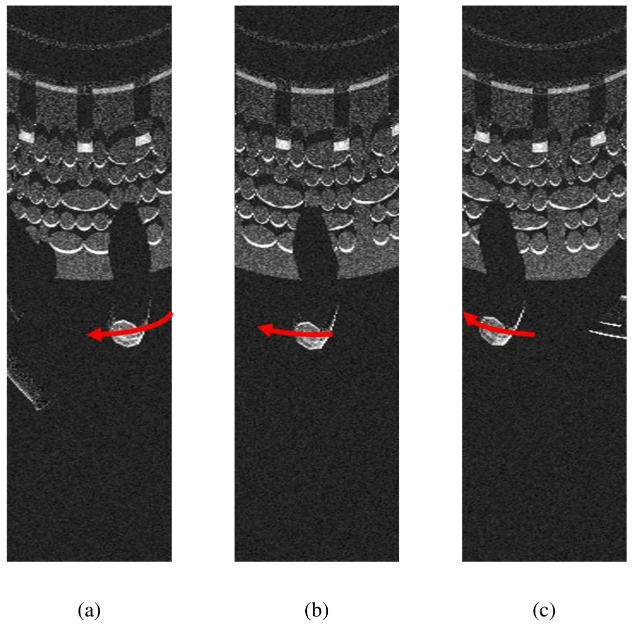


Fig. 8: Sequence of displacement of an object between image frames in polar coordinates. When the sonar moves in y direction, the objects in the images seem to rotate rather than displacing horizontally.

TABLE I: Errors in trajectory with alternation between x translation and y translation

Seabed texture	RMSE from motion estimation		
	x-trans (mm)	*y-trans (mm)*	*z-rot (deg)*
Gaussian noise	0.95	4.40	0.01
Perlin high correlation	3.33	11.86	0.06
Perlin low correlation	2.45	9.03	0.04
Photograph 1	3.93	12.29	0.05
Photograph 2	4.83	15.98	0.05
Photograph 3	3.95	13.78	0.05

TABLE II: Errors in trajectory with xy translation

Seabed texture	RMSE from motion estimation		
	x-trans (mm)	*y-trans (mm)*	*z-rot (deg)*
Gaussian noise	0.86	2.27	0.03
Perlin high correlation	1.70	6.19	0.08
Perlin low correlation	1.28	4.29	0.06
Photograph 1	2.49	5.93	0.07
Photograph 2	4.69	12.59	0.10
Photograph 3	4.24	11.56	0.09

TABLE III: Errors in trajectory with rotation around z axis

Seabed texture	RMSE from motion estimation		
	x-trans (mm)	*y-trans (mm)*	*z-rot (deg)*
Gaussian noise	2.88	10.63	0.31
Perlin high correlation	1.13	13.06	0.41
Perlin low correlation	1.17	13.39	0.38
Photograph 1	0.55	11.00	0.39
Photograph 2	0.71	7.55	0.41
Photograph 3	0.78	7.44	0.41

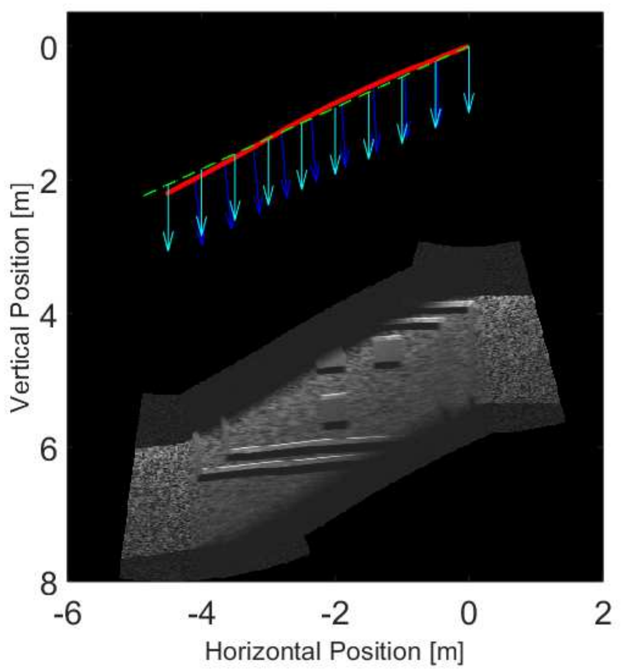


Fig. 9: Mosaicing of a data set of 300 simulated frames. Red line and green lines represent estimated and simulated trajectories respectively. Dark blue and cyan arrows represent the attitude of the sonar for estimated and simulated data, respectively.

different imaging sonars can be simulated using this tool as long as their parameters are known and set on the simulator. Also, other noise models can be implemented and customized for different sonars. Attitude and position parameters of the sonar are provided by the simulator, allowing the simulated images to represent a ground truth, e.g., for image registration techniques. As an example, when the images were used as data source for the attitude-trajectory estimation and mosaicing, it was possible to make a quantitative analysis of this method.

Acknowledgment

JEAM acknowledges financial support from CONACyT.

References

[1] H. Saç, M. K. Leblebicioğlu, and G. Bozdaği Akar, "2D high-frequency forward-looking sonar simulator based on continuous surfaces approach," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, no. 1, pp. 2289–2303, 2015.

[2] N. Hurtós, D. Ribas, X. Cufí, Y. Petillot, and J. Salvi, "Fourier-based registration for robust forward-looking sonar mosaicing in low-visibility underwater environments," *Journal of Field Robotics*, vol. 32, no. 1, pp. 123–151, 2015.

[3] T. Guerneve, K. Subr, and Y. Petillot, "Three-dimensional reconstruction of underwater objects using wide-aperture imaging sonar," *Journal of Field Robotics*, vol. 35, no. 6, pp. 890–905, 2018.

[4] M. Prats, J. Perez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 2577–2582.

[5] T. Guerneve and Y. Petillot, "Underwater 3D reconstruction using BlueView imaging sonar," in *IEEE OCEANS*, 2015, pp. 1–7, Genova, Italy.

[6] H. Ragheb and E. R. Hancock, "Lambertian reflectance correction for rough and shiny surfaces," in *IEEE Proceedings. International Conference on Image Processing*, vol. 2, 2002, pp. 553–556.

[7] M. Oren and S. K. Nayar, "Generalization of Lambert's reflectance model," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, 1994, pp. 239–246.

[8] *P900 Series 2D Imaging Sonar*, BlueView Technologies, Inc., 2011.

[9] T. Watanabe, G. Neves, R. Cerqueira, T. Trocoli, M. Reis, S. Joyeux, and J. Albiez, "The rock-gazebo integration and a real-time auv simulation," in *12th IEEE Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 2015, pp. 132–138.

[10] 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.

[11] M. Borawski and P. Forczmański, "Sonar image simulation by means of ray tracing and image processing," in *Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems*. Springer, 2005, pp. 209–214.

[12] M. D. Aykin and S. Negahdaripour, "Efficient ray-casting of quadric surfaces for forward-scan sonars," in *MTS/IEEE OCEANS*, 2016, pp. 1–7, Monterey.

[13] D. Gueriot and C. Sintes, "Sonar data simulation," in *Sonar Systems*. IntechOpen, 2011.

[14] "Unity," https://unity.com/, accessed: 2019-04-23.

[15] S. L. Kim, H. J. Suk, J. H. Kang, J. M. Jung, T. H. Laine, and J. Westlin, "Using unity 3D to facilitate mobile augmented reality game development," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 21–26.

[16] J. Ribeiro, J. E. Almeida, R. J. Rossetti, A. Coelho, and A. L. Coelho, "Using serious games to train evacuation behaviour," in *The 7th IEEE Iberian Conference on Information Systems and Technologies (CISTI 2012)*, 2012, pp. 1–6.

[17] A. Indraprastha and M. Shinozaki, "The investigation on using Unity3D game engine in urban design study," *Journal of ICT Research and Applications*, vol. 3, no. 1, pp. 1–18, 2009.

[18] "Unity solutions," https://unity.com/solutions, accessed: 2019-05-17.

[19] P. Coupé, P. Hellier, C. Kervrann, and C. Barillot, "Nonlocal means-based speckle filtering for ultrasound images," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2221–2229, 2009.

[20] F. Schmitt, M. Mignotte, C. Collet, and P. Thourel, "Estimation of noise parameters on sonar images," in *Statistical and Stochastic Methods for Image Processing*, vol. 2823. International Society for Optics and Photonics, 1996, pp. 2–13.

[21] A. Abu and R. Diamant, "Robust image denoising for sonar imagery," in *MTS/IEEE OCEANS Kobe Techno-Oceans (OTO)*, 2018, pp. 1–5.

[22] D. Kuan, A. Sawchuk, T. Strand, and P. Chavel, "Adaptive restoration of images with speckle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 373–383, 1987.

[23] F. Maussang, J. Chanussot, A. Hétet, and M. Amate, "Mean–standard deviation representation of sonar images for echo detection: Application to SAS images," *IEEE Journal of Oceanic Engineering*, vol. 32, no. 4, pp. 956–970, 2007.

[24] *DIDSON 300 Standard Version Specifications*, Sound Metrics Corp., 2009.

[25] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[26] B. T. Henson and Y. V. Zakharov, "Attitude-trajectory estimation for forward-looking multibeam sonar based on acoustic image registration," *IEEE Journal of Oceanic Engineering*, pp. 1–14, 2018, early access.

[27] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[28] K. Perlin, "Improving noise," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 681–682.

[29] B. T. Henson, "Image registration for sonar applications," Ph.D. dissertation, University of York, 2017, accessed: 2019-05-17. [Online]. Available: http://etheses.whiterose.ac.uk/19536/1/thesis.pdf