25th International Conference on Harbor, Maritime and Multimodal Logistic Modeling & Simulation
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0339 © 2023 The Authors.
doi: 10.46354/i3m.2023.hms.007

Efficient Maritime Object Detection and Validation for Enhancing Safety of Uncrewed Marine Systems

Ahmet Saglam[1,*] and Yiannis Papelis[1,*]

[1]Virginia Modeling, Analyses, and Simulation Center, Old Dominion University, 1030 University Blvd., Suffolk, VA, 23435, USA

[*]Corresponding author. Email address: asaglam@odu.edu, ypapelis@odu.edu

Abstract

Safe operation of uncrewed maritime systems is a major concern in the presence of other vehicles or obstacles. Typically, perception algorithms utilize sensor data to identify obstacles that must be avoided, and AI algorithms are used to interpret raw sensor data for use in navigation and object avoidance algorithms. However, perception algorithms are typically computationally expensive. In this paper, we present an efficient method for detecting obstacles using raw lidar data in the form of range or Point Cloud, employing computationally efficient techniques that do not depend on trained models or AI matching. The approach converts the sensor readings into the robot's local coordinate system, projecting it onto an occupancy map, and applying efficient image processing techniques to detect obstacles. As a rapid and easy to implement algorithm, the proposed work provides a practical solution for lidar-based maritime perception applications. This paper further focuses on detection of near-by objects with simple shapes, such as buoys or totems, which are commonly used in near-shore and near-harbor maritime environments. With the ability to detect obstacles efficiently, our algorithm can help ensure safe navigation when maneuvering these environments. Results show that the algorithm can accurately detect buoys and totems with minimal false positives.

Keywords: Obstacle detection; lidar-based algorithm; unmanned systems; safe teleoperation

1. Introduction

Uncrewed maritime systems, such as autonomous underwater vehicles (AUVs) and unmanned surface vessels (USVs), have revolutionized various marine applications (Molfino et al., 2014), ranging from underwater exploration to maritime surveillance. However, ensuring the safe operation of these systems in the presence of other vehicles or obstacles remains a critical challenge (Ahmed & Naamane, 2021; Bruzzone et al., 2019). To address this concern, efficient and reliable maritime object detection and validation techniques are crucial for enhancing the safety of these systems.

Traditionally, perception algorithms have been employed to utilize sensor data, including lidar, to identify obstacles and potential risks that must be avoided. These algorithms often leverage AI techniques to interpret raw sensor data, enabling accurate detection and interpretation of the surrounding environment. While effective, the computational demands associated with these algorithms can limit their practicality, especially for smaller vessels that are space and power limited.

In this paper, we propose an alternative approach for detecting obstacles in uncrewed maritime systems that utilizes raw lidar data in the form of range or Point Cloud. Our method deviates from the traditional reliance on trained models or AI matching, instead employing computationally efficient and

straightforward techniques. By converting the sensor readings into the local coordinate system of the robot, projecting them onto an occupancy map, and applying image processing techniques, we extract and validate obstacle locations with high accuracy and minimal computational overhead.

The remainder of this paper is structured as follows. Section 2 provides an overview of the related work in the field of LIDAR obstacle detection, highlighting the existing approaches and their limitations. In Section 3, we present our proposed LIDAR obstacle detection algorithm, explaining its underlying principles and key components. Section 4 presents the preliminary results obtained from our experimental evaluation, showcasing the performance and effectiveness of our algorithm. Finally, in Section 5, we draw conclusions based on our findings and discuss potential future work to further improve the proposed approach.

2. Related Work

Numerous studies (Molina-Molina et al., 2021; Qiao et al., 2021; X. Zhang et al., 2021) have investigated the use of AI techniques for maritime obstacle detection, aiming to enhance the safety and efficiency of uncrewed marine systems.

One common approach in the literature is the utilization of deep learning techniques for maritime object detection. Convolutional neural networks (CNNs) have been widely applied to process sensor data, such as lidar or sonar, and extract features for object recognition (Fu et al., 2021; Ma et al., 2019; Pan et al., 2020; Shi et al., 2022; Xu et al., 2020). These models are trained on large datasets containing annotated maritime objects, enabling them to detect various obstacles with high accuracy. However, the computational complexity of deep learning models poses challenges for real-time implementation, especially in resource-constrained onboard systems.

Another line of research focuses on employing AI matching algorithms for maritime obstacle detection. These techniques utilize machine learning algorithms, such as support vector machines (SVMs) (Gupta & Gupta, 2021; Kaido et al., 2016) or random forests (Stanislas & Dunbabin, 2018; C. Zhang et al., 2020), to match sensor data with pre-defined obstacle patterns or templates. By training the algorithms on labeled datasets, they can identify obstacles based on similarities between the sensor readings and the patterns. Although effective in certain scenarios, these approaches may suffer from limited adaptability to varying environmental conditions and require substantial computational resources for matching.

While AI-based techniques have shown promise in maritime obstacle detection, they often suffer from computational complexity, which can limit their real-time implementation and practicality. In this paper, we propose an alternative approach that circumvents the reliance on trained models or AI matching. By utilizing raw lidar data and employing computationally efficient and straightforward techniques, our method offers a practical solution for efficient maritime object detection and validation.

3. LIDAR Obstacle Detection Algorithm

The overall algorithm is presented in Figure 1. The input is either 2D or 3D laser data. The data is preprocessed and then apply image processing techniques are applied to detect buoy or totems.

3.1. Sensor Data and Preprocessing

The input to our method is 3D or 2D laser scans. Although any sensor such as depth or ultrasound that measures distances can still work with the proposed method, we focused on specifically LIDAR devices because they are more accurate and faster than others.

When using a 3D LiDAR sensor, the distance measurements provided are in spherical coordinates: radius (r), elevation (ω), and azimuth (α). However, to work with the data more easily, it is necessary to convert these spherical coordinates (r, ω, α) to Cartesian coordinates (x, y, z). Typically, this conversion is handled internally, and the resulting data is reported as a point cloud. A point cloud represents a collection of data points within a specific coordinate system. In our case, these points offer measurements relative to the sensor's frame.

A 2D LiDAR sensor, on the other hand, produces a set of measurements, commonly referred to as a "scan". Each measurement within the scan provides information about the distance to an object or obstacle at a specific angle. These angle values are usually evenly distributed, resulting in a dense set of measurements across the sensor's field of view. Similar to 3D, the readings are relative to the sensor and converted in Cartesian coordinates (x, y) to later process them. It is worth noting that 2D is subject to body motion and works best when the boat is flat while 3D is more resilient to such movements.

In order to accurately perceive and respond to obstacles or plan trajectories that avoid collisions, we need to compensate for the boat's motion, and thus to orient the point cloud generated by the sensor. To do that, we transform the sensor data from local (LIDAR sensor) frame to a global frame.

3.2. Occupancy Images for 3D LIDAR

An occupancy image is a 2D grid representation of the environment that represents the occupancy or occupancy likelihood of each cell in the grid. In this case, we are interested in a slice of points that are parallel to the sea surface. Each cell in the occupancy image contains a value that indicates the occupancy state of that area. This value is usually binary, representing whether the cell is occupied (1, one) or unoccupied (0, zero). These images are generated by

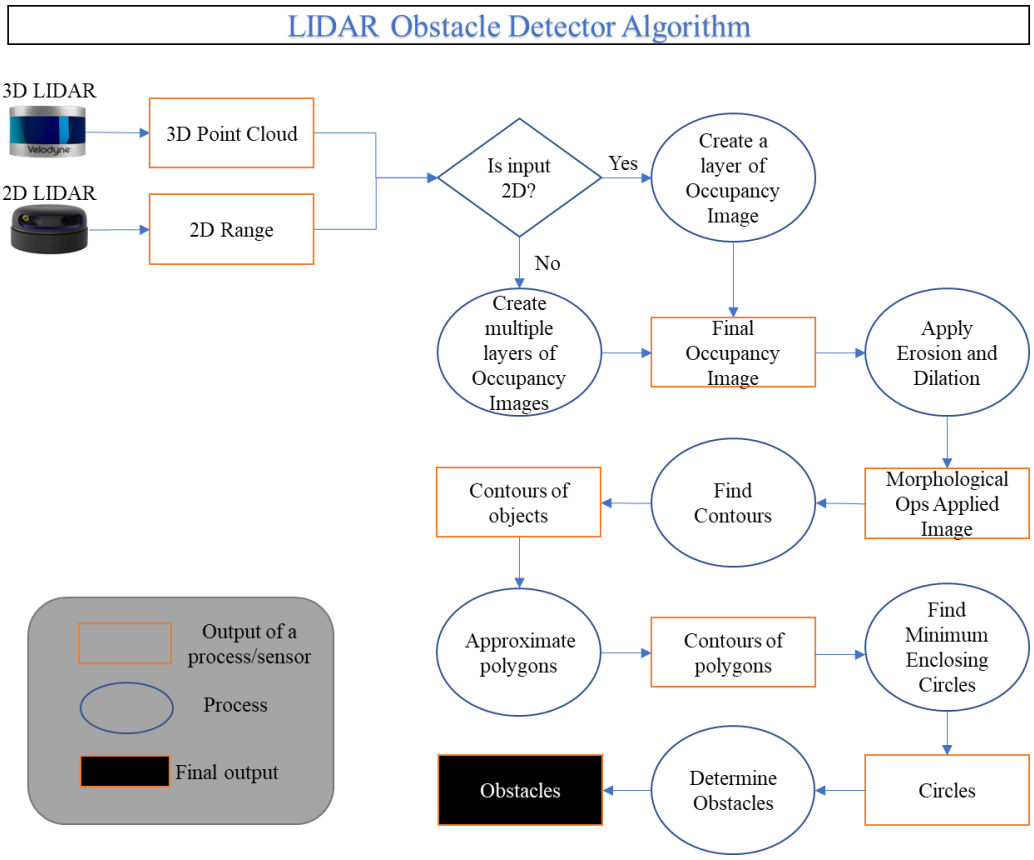


Figure 1. Implementation of the proposed obstacle detection algorithm.

mapping the sensor data onto them. The mapping of sensor data to the occupancy grid involves associating each measurement with its corresponding cell in the grid.

In our algorithm, a two-dimensional occupancy image is generated based on a user-defined area of interest (ROI) around the boat. The ROI is defined by setting the boundaries of the environment in x, y, and direction.

Let:

grid_size: cell size of the occupancy image

$\mathbf{x_{min}}$: minimum boundary in the x direction

$\mathbf{y_{min}}$: minimum boundary in the y direction

$\mathbf{x_{max}}$: maximum boundary in the x direction

$\mathbf{y_{max}}$: maximum boundary in the y direction

$\mathbf{z_{min}}$: lower limit of the height range

$\mathbf{z_{max}}$: upper limit of the height range

$\mathbf{x_{index}}$: x grid index of a specific point in the occupancy image

$\mathbf{y_{index}}$: y grid index of a specific point in the occupancy image

$\mathbf{x_{resolution}}$: x resolution (number of cells)of the occupancy image

$\mathbf{y_{resolution}}$: y resolution (number of cells)of the occupancy image

Then, occupancy image dimensions can be found via:

$$x_{resolution} = \left(\frac{(x_{max} - x_{min})}{grid\_size}\right)$$

$$y_{resolution} = \left(\frac{(y_{max} - y_{min})}{grid\_size}\right)$$

$$x_{index} = x_{resolution} - floor\left(\frac{(point.x - x_{min})}{grid\ size}\right) \quad (1)$$

$$y_{index} = y_{resolution} - floor\left(\frac{(point.y - y_{min})}{grid\ size}\right) \quad (2)$$

The floor function is used to round down the division result to the nearest integer, ensuring that the resulting indices correspond to valid grid cells.

Once we define the ROI and create the occupancy image, we initialize all the values in the cells as not occupied (0). By iterating over all points in the point cloud and filtering out the ones not in the ROI, we find the grid indices ($\mathbf{x_{index}}$, $\mathbf{y_{index}}$) of a point (x, y) on the image using the equations (1) and (2) and then update the corresponding cell value from 0 to 1 (occupied).

In our implementation, we experimented with different parameters in order to capture the features as much as possible with such a huge data with a lot of noise. For example, choosing a bigger grid size resulting in losing small objects in the image, while a

smaller grid size increased the computational complexity. Eventually, considering the size of the objects, e.g., buoys, totems, we want to detect, and the range and accuracy of our sensor, we chose the following parameters that captures objects effectively:

grid_size: 0.03 meters

$\mathbf{x_{min}}$: − 20 meters, $\mathbf{y_{min}}$: − 20 meters

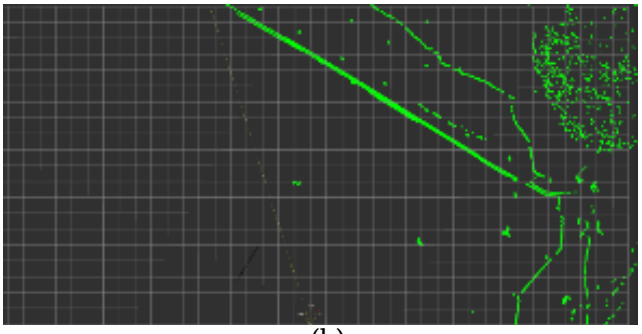
$\mathbf{x_{max}}$: 20 meters, $\mathbf{y_{max}}$: 20 meters

$\mathbf{z_{min}}$: 0.1 meters, $\mathbf{z_{max}}$: 1.5 meters from sea surface

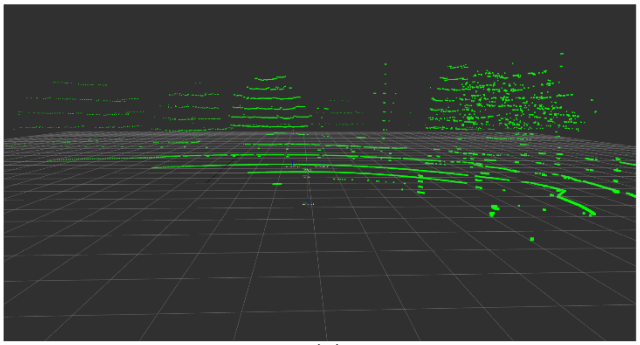
$\mathbf{x_{resolution}}$: 1333 $\mathbf{y_{resolution}}$: 1333

An example of occupancy image created from a 3D LIDAR is shown in Figure 2. The topmost image (Figure 2-a) captures the scene from a camera. Figure 2-b and 2-c depict screenshots from a top down and a horizontal view of the Point Cloud data visualized in RViz, a robotics visualization tool. Lastly, Figure 2-d is binary occupancy image, where the white pixels show the occupied cells. Please note that the resolution does not reflect the actual size of the created image.

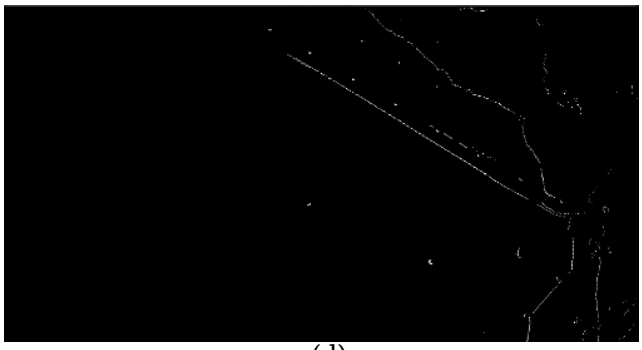
(a)



(b)



(c)



(d)

Figure 2. (a) Image of the environment with a buoy and totem. (b, c) Top down and horizontal view of the captured LIDAR data represented as Point Cloud. (d) Binary occupancy image created from the Point Cloud in this scene.

3.3. Morphological Operations

LIDAR data contains noise due to sensor limitations or environmental factors such as white capping. After creating the binary image in the previous section, it is necessary to perform morphological operations to remove small, isolated noise points and obtain a cleaner binary image before proceeding with further analyses.

Morphological operations are a set of mathematical functions, known as non-linear filters in image processing. Two basic morphological operators are Dilation and Erosion. Dilation expands the boundaries of objects, while erosion removes pixels from object boundaries. These operations are performed on binary images using a small binary filter or kernel known as a structuring element. The structuring element scans the image and modifies the pixels based on its size and shape. Commonly used shapes for structuring elements include rectangles, ellipses, and crosses, as depicted in Figure 3, which showcases 5x5 structuring elements with different shapes.

Rectangle 5x5				
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Ellipse 5x5				
0	0	1	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	1	0	0

Cross 5x5				
0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Figure 3. Structuring element shapes of size 5x5

To achieve cleaner binary images, it is crucial to analyze various sizes and shapes of structuring elements while considering the order of applying dilation and erosion. In Erosion, the size of the element determines the extent of shrinking performed, with larger shapes resulting in greater shrinkage. In dilation, as the structuring element increases in size, the resulting areas of the objects also become larger, and isolated islands of pixels also increase in size.

After experimenting with numerous operations, such as applying erosion followed by dilation or vice versa, using different structuring element shapes (rectangle, ellipse, cross), and sizes (2x2, 3x3, 5x5, 7x7, 9x9), we found out that the combination of dilation with a 5x5 cross-shaped element, followed by erosion with a 3x3 rectangle-shaped element, and another erosion with a 2x2 rectangle-shaped element, yielded the best results for objects located within a 20-meters radius. With this order, we first group nearby areas into a single object and then remove isolated noisy regions. Also, using a larger structuring element for dilation allows for capturing neighboring pixels effectively, and a smaller structuring element for erosion helps preserve larger areas.

Figure 4 shows the result of the morphological operations applied on a binary image created from 3D LIDAR. In all images in Figure 4, white pixels depict occupied cells. However, for clarity, we put red solid lines around the pixels indicating totems and yellow dashed lines around noise pixel. As we can see in Figure 4 bottom-right image, the noise shown in top-left image is cleared successfully.

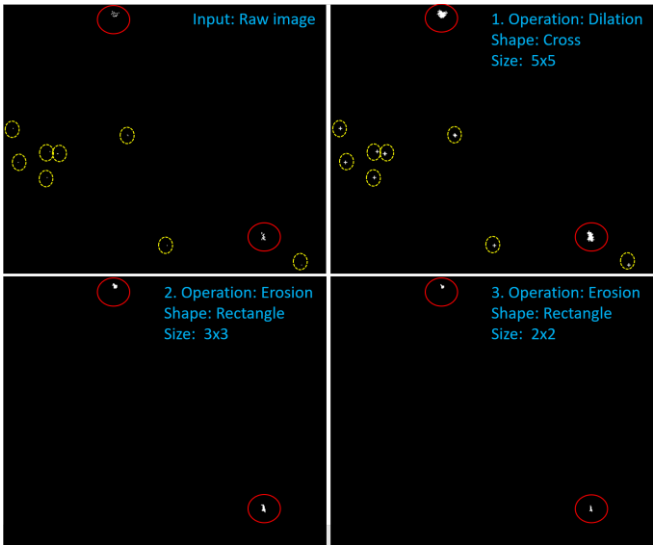


Figure 4. Morphological operations

3.4. Finding Contours

Now that we have a better binary image, we can extract contours. Contours refer to the continuous curves or boundaries that delineate the shape of objects within an image. Similar to implementation of morphological operations, we retrieve contours with the help of OpenCV function implementing the algorithm of (Suzuki, 1985). To find contours using this method, we need to specify:

1. Contour retrieval mode, which determines the hierarchical relationship between contours. Options include:
    a. retrieving only the outermost contours
    b. retrieving all contours in a flat list
    c. retrieving all contours in a hierarchical tree structure
2. Contour approximation method, which determines how the contour points are approximated and compressed. One option is to compress horizontal, vertical, and diagonal segments into their respective end points. The other option is to store all the contour points without approximation.

Regarding the retrieval mode, we focus solely on the outermost contours since we do not need to analyze parent-child relationships or inner object parts. As for the approximation method, although the second option preserves detailed contour information, it generates a large number of points, consuming more memory and slowing down subsequent processing. Since we did not observe significant improvements compared to the first option, we decided to compress the points.

Upon applying the Suzuki's algorithm, we simplify the extracted contours while preserving their overall shape by employing the Douglas-Peucker algorithm (Saalfeld, 1999). This algorithm simplifies a curve or polygon by recursively dividing it into smaller segments and then approximating each segment with a line. Therefore, reducing the number of vertices leads to computational efficiency and better generalization of the shape.

Figure 5 illustrates the outcome of contour extraction from the image resulting from the previously applied morphological operations (Figure 4 - bottom left). The left image displays the result of the initially obtained contours, while the right image demonstrates the application of Douglas-Peucker's algorithm. Please note that red circles are drawn around the extracted contours (pink) to better emphasize them.

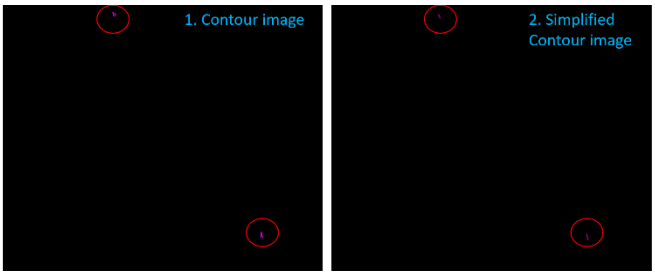


Figure 5. Finding Contours

3.5. Detection of Obstacles

To detect buoys, totems, or similar objects, we compute the minimum enclosing circles based on the previously retrieved contours. We utilize the Smallest Enclosing Disks method (Welzl, 2005) implemented in OpenCV to find the centers and radii of the smallest circles that completely enclose the input contours.

After finding all the circles, we reject circles that are either too small or too big compared to the size of totem-like obstacles. Finally, we mark the remaining circles as obstacles. Figure 6 (left) visualizes the source sensor data, Point Cloud, which was used to create the binary images shown earlier in Figures 4 and 5. Figure 6 (right) displays the detected obstacles after applying all the aforementioned image processing techniques.

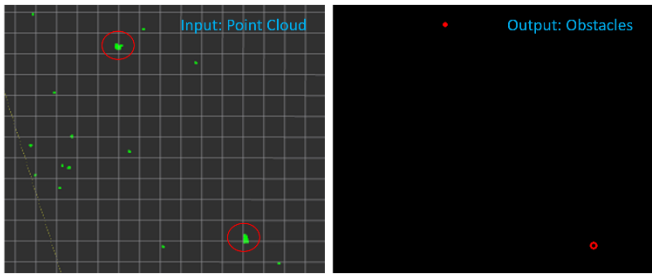


Figure 6. Detected obstacles from the previous scene

4. Preliminary Results

We have implemented the proposed work in Ubuntu 18.04 environment using ROS (Robotics Operating System) and OpenCV libraries with C++ programming language.

The algorithm is tested using sensor data captured from Velodyne's VLP-16 lidar sensor mounted on our Wave Adaptive Modular Vessel (WAM-V) boat (Figure 7). We recorded the data using rosbag, a set of tools for recording from and playing back to ROS data including Point Cloud. The sensor sends out Point Cloud at the rate of 10 Hz.

Figure 7. WAM-V with Velodyne LIDAR

The tests are conducted on a Dell computer with 12Th Gen Intel Core i7-12700Hx20, NVIDIA GeForce RTX 3080 Ti GPU, and 64 GB RAM. We analyzed a sequence of 30 lidar samples, all of which contained two obstacles. Our algorithm was able to detect these totems 28 times successfully. In two frames, it detected only one of the obstacles. In average, it took only 0.0042 secs per frame to output the obstacles.

In Figure 8, we show another application our algorithm on a real data obtained from the same device. Figure 8(a) captures the environment from a camera mounted exactly on top of the lidar with the same horizontal angle. The laser data is visualized in Rviz in Figure 8(b). Figure 8(c) depicts the binary image created from this reading. While the result of processed image is presented in Figure 8(d), the detected obstacles are displayed in Figure 8(e).

5. Conclusions and Future Work

We proposed an image-processing based obstacle detection algorithm that aims to overcome the limitations of existing AI-based approaches, offering a rapid and easy-to-implement solution for lidar-based maritime perception applications. By focusing on the detection of nearby objects with simple shapes, such as buoys or totems, our algorithm provides a practical and efficient solution for enhancing safety and security in uncrewed marine systems. The results of our experiments demonstrate the accuracy and effectiveness of our algorithm, with minimal false positives, thereby showcasing its potential for real-world deployment.

Our method, which is based on Point Cloud processing and image segmentation is computationally very effective. However, determining the buoy-like objects in a complex environment using this approach is challenging because other items would interfere the detection of them.

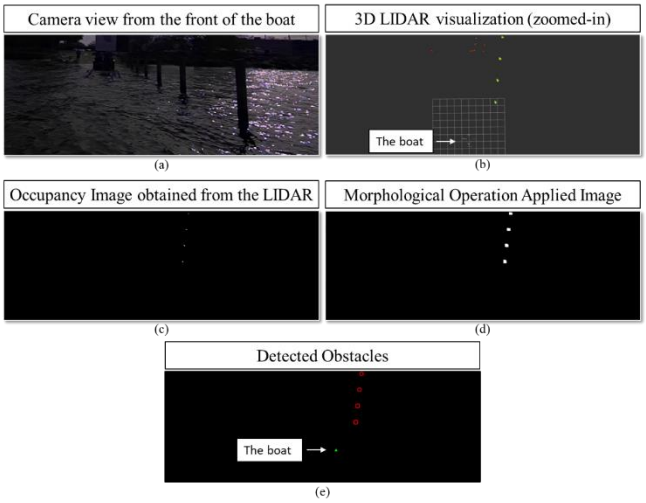


Figure 8. Another example of totem-like obstacle detection using our technique.

Acknowledgements

A portion of this work was supported by a grant from the Commonwealth Cyber Initiative of Coastal Virginia (COVA CCI).

References

Ahmed, M. M. M., & Naamane, A. (2021). *Decentralized navigation control of multiple vehicles with obstacle avoidance*. 152–161.

Bruzzone, A., Massei, M., Sinelshchikov, K., Fadda, P., Fancello, G., Fabbrini, G., & Gotelli, M. (2019). *Extended reality, intelligent agents and simulation to improve efficiency, safety and security in harbors and port plants*. 88–91.

Fu, H., Song, G., & Wang, Y. (2021). Improved YOLOv4 marine target detection combined with CBAM. *Symmetry*, *13*(4), 623.

Gupta, V., & Gupta, M. (2021). *Automated object detection system in marine environment*. 225–235.

Kaido, N., Yamamoto, S., & Hashimoto, T. (2016). *Examination of automatic detection and tracking of ships on camera image in marine environment*. 58–63.

Ma, L. Y., Xie, W., & Huang, H. B. (2019). Convolutional neural network based obstacle detection for unmanned surface vehicle. *Mathematical Biosciences and Engineering: MBE*, *17*(1), 845–861.

Molfino, R., Zoppi, M., Dinale, A., & Muscolo, G. (2014). *A robotic vehicle for freight delivery in urban areas*. 10–12.

Molina-Molina, J. C., Salhaoui, M., Guerrero-González, A., & Arioua, M. (2021). Autonomous marine robot based on AI recognition for permanent surveillance in marine protected areas. *Sensors*, *21*(8), 2664.

Pan, M., Liu, Y., Cao, J., Li, Y., Li, C., & Chen, C.-H. (2020). Visual recognition based on deep learning for navigation mark classification. *IEEE Access*, *8*, 32767–32775.

Qiao, D., Liu, G., Lv, T., Li, W., & Zhang, J. (2021). Marine vision-based situational awareness using discriminative deep learning: A survey. *Journal of Marine Science and Engineering*, *9*(4), 397.

Saalfeld, A. (1999). Topologically consistent line simplification with the Douglas-Peucker algorithm. *Cartography and Geographic Information Science*, *26*(1), 7–18.

Shi, B., Su, Y., Lian, C., Xiong, C., Long, Y., & Gong, C. (2022). Obstacle type recognition in visual images via dilated convolutional neural network for unmanned surface vehicles. *The Journal of Navigation*, *75*(2), 437–454.

Stanislas, L., & Dunbabin, M. (2018). Multimodal sensor fusion for robust obstacle detection and classification in the maritime RobotX challenge. *IEEE Journal of Oceanic Engineering*, *44*(2), 343–351.

Suzuki, S. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46.

Welzl, E. (2005). *Smallest enclosing disks (balls and*

ellipsoids). 359–370.

Xu, Z., Huo, Y., Liu, K., & Liu, S. (2020). Detection of ship targets in photoelectric images based on an improved recurrent attention convolutional neural network. *International Journal of Distributed Sensor Networks*, *16*(3), 1550147720912959.

Zhang, C., Bin, J., Wang, W., Peng, X., Wang, R., Halldearn, R., & Liu, Z. (2020). AIS data driven general vessel destination prediction: A random forest based approach. *Transportation Research Part C: Emerging Technologies*, *118*, 102729.

Zhang, X., Wang, C., Jiang, L., An, L., & Yang, R. (2021). Collision-avoidance navigation systems for Maritime Autonomous Surface Ships: A state of the art survey. *Ocean Engineering*, *235*, 109380.