Obstacle detection and tracking in maritime environments

Isabella-Sole Bisio, Serena Roncagliolo, Francesca Odone, Enrico Simetti

Abstract—**Environment perception represents a vital role in autonomous systems. In this study, scene understanding in maritime environments is performed through an obstacle detection and tracking procedure based on a video camera and LIDAR acquisition.**
Point clouds are clustered and tracked in 3D, with a Kalman filter estimating the object's size and position. The general objective was to reinforce and improved the previous work, focusing on the link between the vision and LIDAR processing pipelines. The point cloud of an obstacle boat acquired by the LIDAR varies quite a lot to the sea motion and the tilt angles of the LIDAR sensor. Using filtering techniques it is possible to recover a more precise size of the boat.

I. Introduction

The investigation on the development of Unmanned Surface Vehicles (USVs) has improved considerably during the last decade. This expansion is due to an increasing interest in their applications such as remote sensing, surveillance, coastal patrolling, navigation and communication support to Unmanned Underwater Vehicles (UUVs) [1]. However, they still present many limitations, such as the simultaneous detection and avoidance of static and dynamic obstacles.
Generally, UVSs can be controlled by a remote location and operated to carry out a specific task in open or confined waters [1]. These vehicles should function without any manual control and their performance depends on the navigation, guidance and control (NGC) system design, which should handle every onboard components. Frequently, even a progressive and advance NGC design may not be able to perform full autonomy without the use of a system that can perform obstacle detection and avoidance (ODA). In many studies, it was shown that when dealing with manned vessels, 60% of mortal incidents at sea are caused by collisions [2].
Researchers have been strongly motivated to develop automatic collision detection and avoidance systems for manned and unmanned vehicles because it has been proved that many incidents at sea were caused by human error [1]. Consequently, the scientific community focused their research on the automatic detection of surrounding objects and their motion estimation to obtain a reliable control that could be applied in a wide range of environmental scenarios.

II. Related Work

Obstacle detection and avoidance for marine vehicles is still today an explored topic and during the last decades, researchers have tried to develop various types of USV systems for several missions and applications [3], [4]. When dealing with USV platforms, it is common to detect and track ships using marine radars. We may find several types of these radars, such as S-band and X-band radar, mainly used for collision detection and avoidance. Indeed, when using X-band radars, it is possible to estimate the motion and size of the target using a joint probabilistic data association tracker [5], apply a Bayesian network-based method on multiple frame-by-frame radar images [6] and incorporate an alpha-beta tracking filter onto multiple motion models [7].
In [8] was presented a new method for object detection and tracking vision-based to deal with the challenges present in maritime environments. In 2017, it was suggested the combined use of cameras and LIDAR sensors to perform motion analysis of close-range obstacles in marine environments [9]. The monocular camera gives bearing data from which we can estimate relative range data from the bearing changes in the framework of bearing-only tracking, while 3D LIDARs can detect dynamic and static obstacles in close range [3]. It was also suggested to apply multi-sensor tracking approaches using radars and LIDARs measurements combined into a tracking filter [10], while camera and LIDAR measurements can be fused into a radar-based detection algorithm used to improve the general performance and minimize the radar's blind zone. Furthermore, radars, cameras and LIDARs can be combined with AIS (Automatic Identification System) to detect the surrounding environment more efficiently.

III. Previous work

We start from the work of Sorial et al. [11] where they detect obstacles in the image place using an object detection system based on CNN (Convolutional Neural Network) YOLO. In Sorial et al. it is shown a procedure which uses LIDAR point clouds to locate obstacle in the world reference frame and derive a scene map which can be used to initialize a path planning algorithm.

A. System architecture

As mentioned before, multiple sensory channels can be used to produce a map of the surrounding environment for safe and efficient navigation. The employment of different types of sensors is useful to ensure optimal obstacles tracking in situations where one of the sensors could be damaged or not able to collaborate for the mapping task.
In Sorial et al. [11], is proposed the ULISSE USV system, developed by the Interuniversity Research Center on Integrated System for the Marine Environment (ISME) and by the University of Genoa, as shown in Fig. (1). It is equipped with a wide range of sensors:

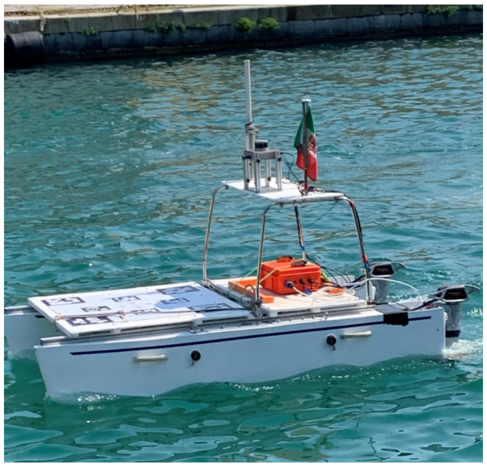
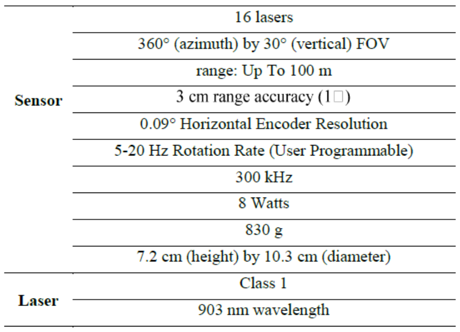


Fig. 1: ULISSE catamaran [11].



Sensor	16 lasers
	360° (azimuth) by 30° (vertical) FOV
	range: Up To 100 m
	3 cm range accuracy (1□)
	0.09° Horizontal Encoder Resolution
	5-20 Hz Rotation Rate (User Programmable)
	300 kHz
	8 Watts
	830 g
	7.2 cm (height) by 10.3 cm (diameter)
Laser	Class 1
	903 nm wavelength

Fig. 2: Specifications of the Velodyne VLP-16 [12]

- a video camera Basler CCD;
- a Velodyne VLP-16 LIDAR sensor;
- Gyro sensor;
- Accelerometer sensor;
- Compass sensor;

The first two sensors are placed and fixed together above the hull, while the others are contained in an internal waterproof box.
The camera can acquire images at 31 fps, recording high-definition videos (1032x778 pixels), whereas the LIDAR sensor can cover 360 degrees horizontally and 30 degrees vertically, which are sufficient to investigate the enclosing environment and especially to deal with the wave oscillations. The output data is then memorised and managed by a separate ROS middleware.
The Velodyne VLP-16 LIDAR is widely used in the mapping industry for its low requirements and reduced cost. It is feasible to collect high-resolution 3D models when integrated on unmanned aerial vehicles, indoor mapping platforms, and autonomous vehicles [12]. The VLP-16 is manufactured by Velodyne LiDAR of Morgan Hill and its specifications are shown in Fig. (**??**). The Velodyne VLP-16 LIDAR has 16 beams, positioned with a certain angle to each other, which come out from the centre of the sensor. The rays reflections are captured by the sensor and can detect points of nearby objects. Once the sensor has completed a full rotation, the points are collected. Unfortunately, if the acquisition is done at a certain distance, fewer lines of points are collected.

B. Object Detection

Vision is mainly used to extract information from the surrounding environment. Sorial et al. proposed to acquire data from a camera as the first step, using a CNN whose architecture is feasible to derive objects and classification information found in the surrounding environment. In particular, they chose to use the per-frame object detector YOLOv3 (You Only Look Once), a CNN-based detector capable to understand if obstacles are present or not in the observed scene.
YOLOv3 is an improved version of the classic YOLO which can compute predictions at three different scale levels and detect also small objects. YOLO usually takes an entire image as input and simultaneously generates multiple bounding-boxes and class probabilities. The image is divided into an SxS grid, where each cell is responsible for checking whether or not an object is present in its centre. Each cell predicts several bounding boxes which are described by 5 parameters [13]:

- four values giving the x, y coordinates of a vertex, the height and the width of the shape of the rectangular bounding box;
- one parameter giving a measure of the confidence in the prediction that the object is contained or not inside the specific bounding box.

C. Object Tracking

Sorial et al. use a ”tracking-by-detection” approach, where the tracking algorithm works using both predictions and outputs of an object detection algorithm [11]. Each of the tracks is assigned to an ID, which can be used to recognise them at different time instants. We can summarise their approach as follow:

- *Data Association*: it defines if tracked instances and detected ones have matching attributes using the approach of Bewley et al. [14], defining an appropriate matching cost matching between tracked and detected values;
- *Tracks addition and deletion policy*: unmatched tracks can cause prediction errors or they can affect the objects persistence in the tracking, giving false positives in both cases. It is fundamental to take into account the detector, the prediction model and the environment. Therefore, tracks are added if they correspond to a high-confidence detection and deleted if they are missed for several consecutive frames;
- *Prediction Models*: the system should give a prediction of a new position of the tracked object using information from previous observations; in Sorial et al. [11], they rely on the *Kalman filter*, which uses a combination of a defined motion model and a sequence of measurements updates, or *optical flow*, which uses motion information across frames to estimate the displacement of obstacles.

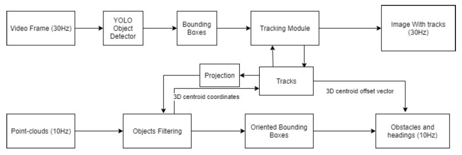


Fig. 3: Proposed pipeline block diagram of Sorial et al. [11].

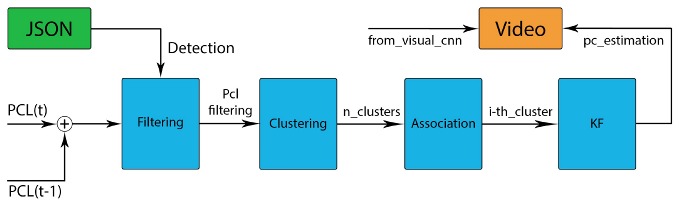


Fig. 4: Proposed new pipeline

D. Sensor Fusion

In Sorial et al. [11] it is proposed a new approach to have a real-time detection of surrounding obstacles. This result can be achieved by combine information and observation coming from multiple detection sensors, obtaining a fusion-based tracking filter as shown in Blackman et al. [15].
However, this approach can present some complications when n associating data of heterogeneous sensors and their relatively large computational load. It is important to fuse accurately information coming from the LIDARs and the cameras.
The main problem to deal with will be the LIDAR-CAMERA calibration since this process is fundamental to determine a correspondence between multiple heterogeneous sensors and provide lidar-camera geometric relationships between the sensors [16]. As shown in [17], it is advisable to use the approach implemented in Park et al. [18], which shows how to calibrate a low-resolution 3D LIDAR combined with a limited number of vertical sensors using 2D-3D correspondences. The 3D points are estimated using a polygonal (triangle or diamond shape) planar board with adjacent sides. Knowing the lengths of adjacent sides, the vertices of the board are estimated as the meeting points of two projected sides of the board. The 3D locations of the vertices come from scanned laser data and they are then compared with their corresponding corners in the 2D image of the camera. This way it is possible to find point-to-point corresponding pairs between 2D images and 3D point clouds, which are then used to solve the linear equations to obtain a suitable calibration matrix [18].

IV. Our Contribution

In the precedent work of Sorial et al. [11], they implemented a simple, but efficient procedure for object tracking-by detection which relies only on motion and appearance properties using data coming from the camera. Their pipeline is shown in Fig. (3) and it is based on camera and video data only. This pipeline was tested offline, and its implementation online is still in development on embedded architectures.
In Sorial et al. [11] was therefore suggested an alternative fusion-based approach using data coming from both LIDAR and cameras. The feasibility of this procedure was tested on a limited number of frames using the software MATLAB. Since they were able to validate its feasibility, our main objective was to implement it as a reliable and efficient system being able to detect an obstacle in real-time using this aforementioned approach. We started working on the skeleton of the code architecture, already implemented, by reorganising its structure and adding some important functionality.
The whole project was developed in ROS using C++.

A. PCL Library

When robots need to work in unstructured environments, they must be able to perceive their surroundings. A high-quality 3D representation of the environment may be provided by *point clouds* obtained using sensors like the Velodyne LIDAR [19].
For handling point clouds efficiently, it is possible to make use of the open-source Point Cloud Library (PCL), which can be fully integrated with ROS [19]. Our algorithm is mainly based on the use of functions from this library.

1) Architecture: PCL is a C++ library used for point cloud processing. Mathematical operations are based on *Eigen*, a open-source library used for linear algebra [20].
PCl includes several 3D processing algorithms operating on point clouds including filtering and feature estimation. Basic use of PCL can be summarised as: first create the processing object such as a filter or an estimator, secondly, set the input point cloud data set to the processing module, lastly, set the required parameters and call the processing function. PCL includes some smaller libraries which can be used for:

- data filters (i.e downsampling, index estraction etc);
- 3D features (i.e. surface normals and curvatures)
- I/O operations;
- cluster extraction;
- surface reconstruction techniques;
- point cloud registration (i.e. ICP).

2) PCL and ROS: Each algorithm from PCL is a standalone building block capable to connect to other blocks the same way nodes are communicating in ROS.

3) Visualization: In PCL, we can find its own custom *visualisation library*, integrated with VTK [21], which offers many functionalities for visualising 3D point clouds and surface data [19]. It is possible to use the library to visualise effectively hyper-dimensional data, set visual properties such as colours and point size and draw basic 3D shaped on-screen for any n-D point cloud dataset.

B. Proposed Pipeline

Starting from the pipeline shown in Fig. (3), we developed our pipeline which uses the output of the visual tracking done by Sorial et al. [11]. We aimed to implement a system capable of performing efficient tracking using 3D point clouds. In Fig. (4) we show a schematic diagram of our pipeline, which is an

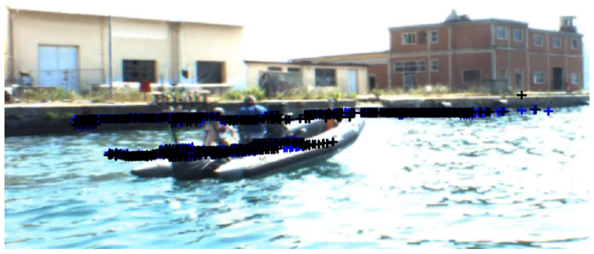


Fig. 5: Plot of point clouds belonging to subsequent frames

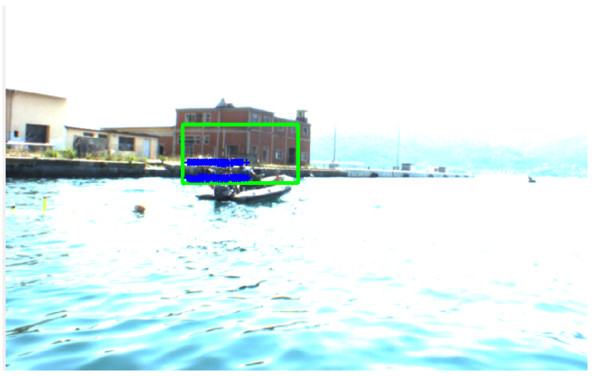


Fig. 6: Output detection using JSON file [22]

expansion of the *Object Filtering* block in Fig. (3).
The general functioning of our system can be summarised as follow:

- it should use the results of the detection made in Sorial et al. [11] on the image plane and read the parameters describing the Bounding Boxes of the obstacles;
- secondly, it should use the four vertices of the Bounding Boxes on the image plane to create a pyramidal shape from the pinhole of the camera by drawing the edges of the shape passing through the vertices;
- it should then select and use only the portion of the point cloud contained inside the volume of the pyramid; this selected 3D volume is projected on the inertial plane, removing the z component of the points;
- it should then create clusters using Euclidean distances and compute their centroids; they will be used to distinguish the different obstacles from the starting point cloud, and track their position in time;
- finally, the output of each cluster centroid is given to a Kalman Filter to observe the variation of its position over time; the Kalman Filter is used to predict the position of the obstacle's bounding box inside the point cloud, not on the image plane.

1) Point Cloud: All operations are performed using two-point clouds belonging to subsequent frames and merged. To check the consistency of this merging operation, we plotted the point clouds at instant *t* and *(t-1)* on the image place using different marker colours as shown in Fig (5). We decided to use this approach because merging point clouds of successive acquisitions together makes sure that, even if some fluctuations are present, what we are looking at remains the same, giving a denser cloud. Furthermore, by doing so, the size of the obstacle is overestimated, which is always a better procedure than underestimating it.

2) Detection: We decided to attempt to design our system such that it could function in real-time. At the current state, it was not possible to test its functioning in a real situation or environment. Thus we used the result of the detection coming from the precedent work of Sorial et al. [11] obtained from the feature extractor YOLO V3 on the image plane and read the parameters describing the Bounding Boxes of the obstacles.
The prior architecture of the code was designed to be functioning using a single Bounding Box whose dimensions corresponded to the ones of the image plane. We decided to replace this single Bounding Box using the *n* Bounding Boxes coming from the result of the prior detection. The data describing their parameters were saved in a JSON file, where for each bounding box we could find the following information:

- *image_id*: containing the id of the i$^{\text{th}}$ frame;
- *bbox*: a vector belonging to the correspondent i$^{\text{th}}$ bounding box containing the *(x,y)* coordinates of the top left vertex, the *width* and the *length*.

This file is then parsed and read by our code. In Fig. (6) we show the result of our improved detection: we can observe that using different bounding boxes, the pyramidal volume used to select a portion of the Point Cloud will be of reduced dimensions.
The main difficulty of this implementation was to synchronise the reading from the JSON file and the detection of the obstacle in the video frame. It was necessary to read the messages written on the camera topic and from the obtained information insert a control that starts the parsing of the JSON data at the exact moment when the relative obstacle appears on the camera range.
Another complication was given by the fact that for several frames, the result of the detection did not compute any bounding box. On the other hand, for further frames, the detection provided more than one bounding box, since more obstacle was present in the camera range. This issue has led us to use the bounding box relative to the nearest obstacle only.

3) Filtering: When localising obstacles, it is possible to project the point cloud directly on the image plane and subsequently, filters point located inside the bounding box. This approach is effective, but it has a high computational cost since the LIDAR provides a considerable number of points.
In Sorial et al. [11] it is suggested to use a more feasible output by locating the bounding boxes in the 3D reference frame of the camera by rotating it with a feasible transformation matrix. Afterwards, our code computes a pyramidal volume whose edges pass through the four vertices of the relative bounding box. This shape is used on the volume of the point cloud to select only the portion of the volume contained

inside the pyramid. The resulting volume is then rotated again using the telemetry of the catamaran to express it on the inertial reference frame (along which the motion happens). The volume is then manipulated using specific functions of the PCL library to project it as a 2D volume. This projection is obtained by removing the z component of the points, providing a footnote of the obstacle, which is then visualised in RViz.

4) Clustering: At this point a clusterisation is performed on the investigated object; this step is fundamental since essential information can be extracted from this result, such as the centre, the direction and the size of the obstacle, which will be later used. A simple clustering algorithm was used in our code, which exploits the information obtained from the calculation of the Euclidean distance between the catamaran and the obstacle. Based on the value obtained, a division is made between one cluster and another, thus being able to distinguish several objects. Within this algorithm, various parameters have been set that can be changed according to the needs of the detection:

- setClusterTolerance(): it is a very sensitive value if a very small value is taken, an actual object can be seen as multiple clusters. On the other hand, if the set value is too high, multiple objects might be seen as one cluster. So our choice of this value was made by a trial and error session since the perfect tuning value which suits our dataset was found;
- setMinClusterSize(): we impose that the clusters found must have at least this size;
- setMaxClusterSize(): we impose that the clusters found must have at maximum this size;
- setSearchMethod(): we decided to use the tree search method but also other methods were checked, such as the brute force searching method.

5) Kalman Filter: A linear Kalman filter was implemented, with a constant velocity model, which takes into account the fact that the length, width and orientation angle of the obstacle does not change over time. The model that we considered has a (7x1) state vector $\underline{\hat{x}}$:

$$\underline{\hat{x}} = \begin{bmatrix} x & y & w & l & vx & vy & \psi \end{bmatrix}^T \tag{1}$$

where the values of *x*, *y*, *w* and *l*, that represent the centre coordinate of the obstacle, its width and length are directly retrieved from the cluster information, whereas the ψ angle, representing the obstacle's orientation angle is evaluated using the major eigenvector of the cluster in this way:

$$\psi = atan(\frac{major\_vector.y}{major\_vector.x}) \tag{2}$$

The *vx* and *vy* linear velocities of the obstacle's centre are only estimated during each iteration. It is also present a (5x1) measurement vector $\underline{y}$, that provides information about:

$$\underline{y} = \begin{bmatrix} x & y & w & l & \psi \end{bmatrix}^T \tag{3}$$

The equations describing our model, in discrete time, are the following (the model does not keep into account noise):

$$\underline{x}(k+1) = A\underline{x}(k) + B\underline{u}(k) \tag{4}$$

$$\underline{y}(k) = C\underline{x}(k) \tag{5}$$

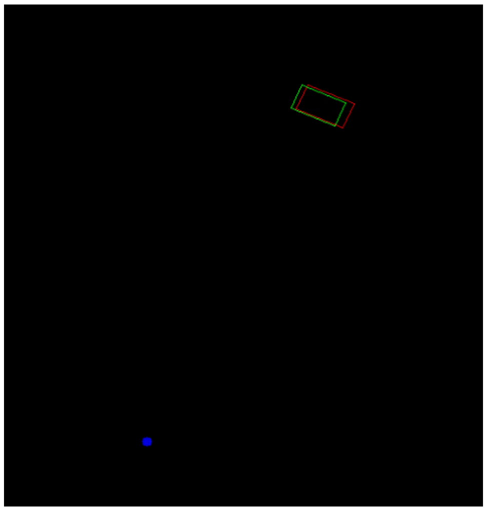


Fig. 7: Kalman filter output

with a (7x7) transition matrix A and a (5x7) measurement matrix C:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ dt & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & dt & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{7}$$

Regarding the covariance values, the *measurement noise covariance* was mainly set by trial and error, the same was done for the *a posteriori error estimate covariance*, to set the initial uncertainty. Whereas for the *process noise covariance*, we also kept into account that the model that we were using to describe the reality (constant length, width and orientation angle) has a small value of uncertainty. The result was significantly improved; in fact, the output of the Kalman filter (red rectangle) faithfully follows the position of the cluster over time (green rectangle) as shown in Fig. (7).

C. Results

For a better understanding of our work and improvements, we decided to plot some of our results using the software MATLAB and compare them with the prior results of the code we started from. Therefore, we selected values of specific variables such as the position of the catamaran and the obstacle and the output of the Kalman Filter. The data was recorded on a *.txt* file which can be easily loaded and visualised using MATLAB.

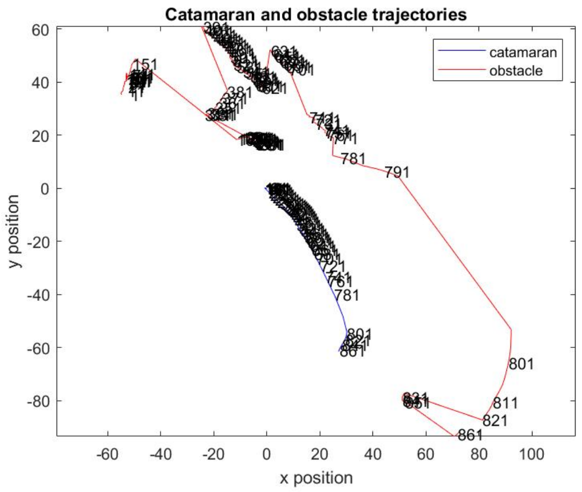


Fig. 8: Obstacle and catamaran trajectories before our improvements

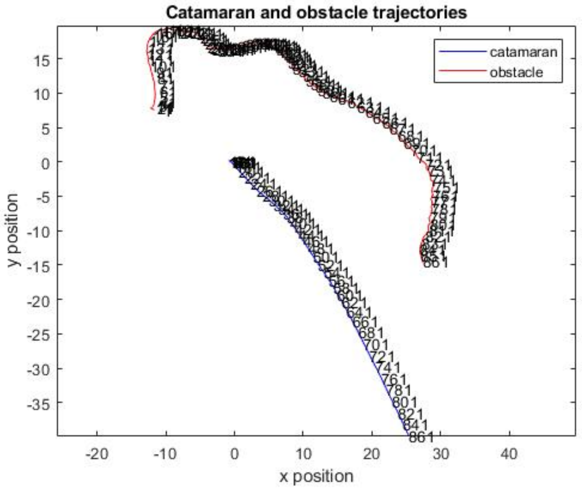


Fig. 9: Obstacle and catamaran trajectories after our improvements

1) Obstacle and catamaran trajectories: Looking at the data values, we assumed the LIDAR on the catamaran having the following reference system:

- Y positive toward the front of the vehicle;
- X positive toward the right-hand side of the vehicle;
- Z positive upwards.

The catamaran has instead:

- X positive forward;
- Y positive to the right;
- Z positive down.

From our code, we obtain latitude and longitude describing the position of the catamaran. They can be transformed into Cartesian coordinates using specific function, generally the Universal Transverse Mercator (UTM) [23], with East-North-Up (ENU) notation, where *x=east*, *y=north*,*z=up*.
In our case, the telemetry of the robot has a different reference system following instead the notation North-East-Down (NED), thus after we convert latitude and longitude using UTM, the resulting Cartesian coordinates should be projected in the NED reference frame, by switching x and y, and inverting z (from up to down).
The previous version of the work did not detect the moving obstacle correctly, because it was also detecting the pier edges, as shown in Fig. (8). Thus we can observe that the trajectory of the obstacle is not continuous, but appears erratic and discontinuous since for some time interval the moving object is not tracked, but it reappears later and further of many meters, as shown in Fig. (8) for the values on the x-axes between 40 and 60. After out implementations, shown in Fig. (9) the system can detect the moving obstacles and ignore the static elements of the environment, resulting in a continuous and neat trajectory.

D. Width and Length

We have also implemented a script to plot the variation of the width and length of the obstacle when filtered using the Kalman Filter. In the represented situation, the tracked object first appears partially in the range of the camera, as shown in Fig. (10), thus the width and length of the resulting cluster assume a small value. When moving, the object at first is closer to the catamaran, thus it appears bigger, and later it appears smaller since it moves away from the USV. If represented graphically, this behaviour should assume the following pattern: first of all, length or width assumes a small value, which subsequently grows. When they reached a peak value, corresponding to the whole object observed in the camera frame and closest to the catamaran, they finally start assuming decreasing values.
In Fig. (11) and Fig. (12), correspondent to the output of our code, this behaviour is shown correctly. In the precedent version, the system is not able to detect the moving object, thus the plot in Fig. (13) and Fig. (14) does not represent correctly the behaviour described previously.

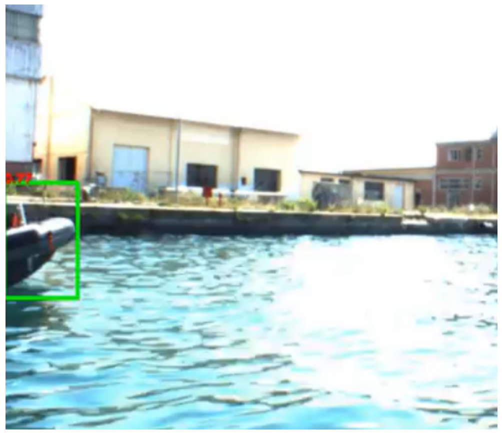


Fig. 10: The obstacle appears partially in the camera

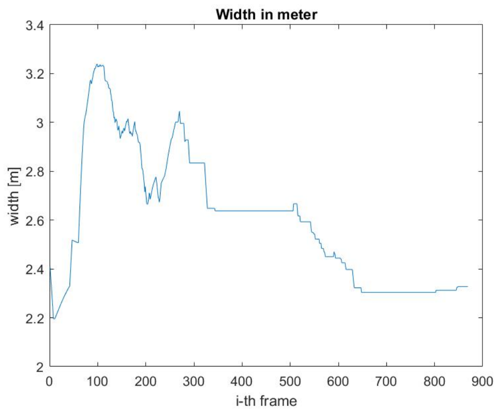


Fig. 11: Width variation

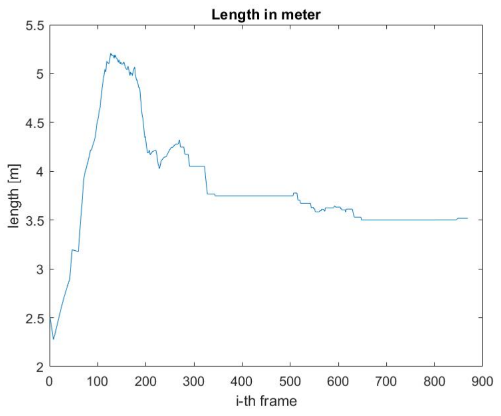


Fig. 12: Length variation

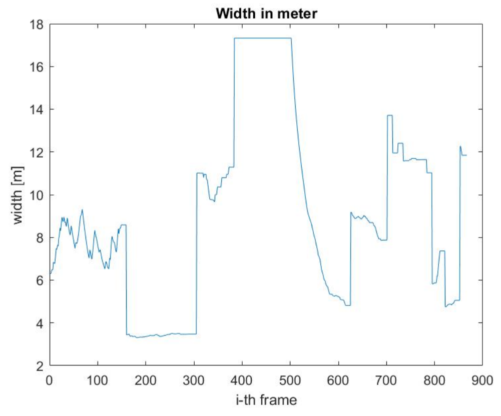


Fig. 13: Width variation - previous version

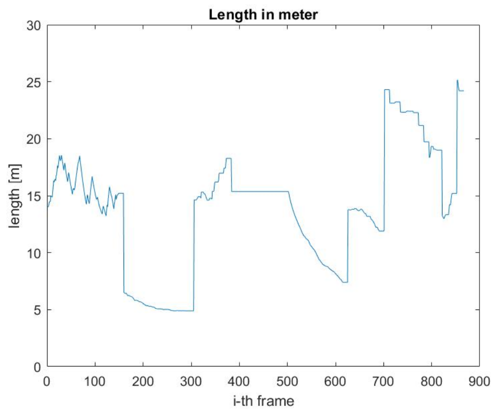


Fig. 14: Length variation - previous version

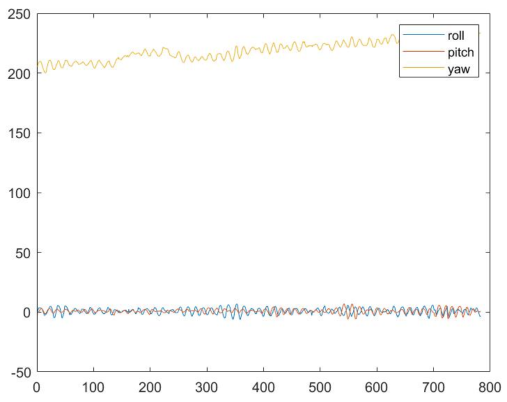


Fig. 15: Yaw, pitch, roll oscillation

V. Future work

Our work still possesses limitations and inaccuracies, which render the system not able to work effectively in real-time.
One of the main limitations was the provided dataset, which consisted of a set of files of extension *".bag"* containing data coming from the different sensors fused. The resulting output appears to be slightly fluctuating because of the motion of the waves affecting the robot's telemetry and the inaccurate data acquisition deriving from the sensors. Being aware of this behaviour, it is possible to plot the observed oscillation (see Fig. (15)) and apply a filter on the yaw angle of the sensors. The oscillation is evident because Sorial et al. [11] used a sensor that lacked gyroscopes and a large bandwidth, thus fast movements could not be detected correctly. Therefore, roll and pitch values are estimated incorrectly, obtaining the wrong value of yaw.
It is advisable to use a more accurate and complete dataset, such as the *MIT Sea Grant Marine Perception Dataset* provided by the MIT Sea Grant College Program of the Massachusetts Institute of Technology (MIT) [22]. This public dataset is used by MIT for autonomous surface vessels research in the marine domain. It contains a large multi-modal sensor dataset collected by a mobile marine autonomy laboratory built on a marine vessel. It provides data collected in several scenarios to study the complexity of operating an autonomous marine vessel in a coastal environment as shown in Fig. (16). It also provides a scenario of a highly trafficked harbour environment.

The dataset contains data acquired from sensors such as a VLP-16 Velodyne 3D Lidar, a 4G Broadband Radar and

Fig. 16: Harbor Traffic with Snow [22]

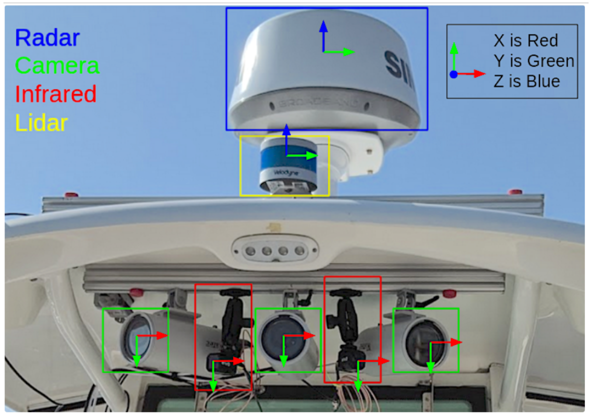


Fig. 17: Sensors MIT Sea Grant Marine Perception Dataset [22]

multiple cameras, as shown in Fig. (17).

VI. Conclusion

This project describes a work in progress towards a reliable implementation of a real-time obstacle detection system for USV. We presented a pipeline relying on both camera and LIDAR data. Our system was tested offline using data acquired from an unmanned surface vessel and typical metrics in object detection. The results have shown the achieved improvements, but the system still requires additional adjustments to be able to detect obstacle in real-time.

Acknowledgment

The authors would like to thank our supervisors for the great support and the Italian Interuniversity Research Center of Integrated Systems for Marine Environment (ISME).

References

[1] S. Campbell, M. Abu-Tair, and W. Naeem, "An automatic COLREGs-compliant obstacle avoidance system for an unmanned surface vehicle," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 228, no. 2, pp. 108–121, 2014, tex.eprint: https://doi.org/10.1177/1475090213498229. [Online]. Available: https://doi.org/10.1177/1475090213498229

[2] Z. Jingsong, W. Price, and P. Wilson, "Automatic collision avoidance systems: Towards 21st century," *Department of Ship Science*, vol. 1, no. 1, pp. 1–5, 2008.

[3] L. Elkins, D. Sellers, and W. R. Monach, "The autonomous maritime navigation (amn) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles," *Journal of Field Robotics*, vol. 27, no. 6, pp. 790–818, 2010.

[4] V. Bertram, "Unmanned surface vehicles-a survey," *Skibsteknisk Selskab, Copenhagen, Denmark*, vol. 1, pp. 1–14, 2008.

[5] G. Vivone, P. Braca, and B. Errasti-Alcala, "Extended target tracking applied to x-band marine radar data," in *OCEANS 2015-Genova*. IEEE, 2015, pp. 1–6.

[6] F. Ma, Y.-w. Chen, X.-p. Yan, X.-m. Chu, and J. Wang, "A novel marine radar targets extraction approach based on sequential images and bayesian network," *Ocean Engineering*, vol. 120, pp. 64–77, 2016.

[7] W. Kazimierski, "Determining of marine radar target movement models for the needs of multiple model neural tracking filter," in *2011 12th International Radar Symposium (IRS)*. IEEE, 2011, pp. 611–616.

[8] T. Cane and J. Ferryman, "Saliency-based detection for maritime object tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2016, pp. 18–25.

[9] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 1993–2016, 2017.

[10] J. Han, J. Kim, and N.-s. Son, "Persistent automatic tracking of multiple surface vessels by fusing radar and lidar," in *OCEANS 2017-Aberdeen*. IEEE, 2017, pp. 1–5.

[11] M. Sorial, I. Mouawad, E. Simetti, F. Odone, and G. Casalino, "Towards a Real Time Obstacle Detection System for Unmanned Surface Vehicles," in *OCEANS 2019 MTS/IEEE SEATTLE*. Seattle, WA, USA: IEEE, Oct. 2019, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8962685/

[12] C. Glennie, A. Kusari, and A. Facchin, "Calibration and stability analysis of the vlp-16 laser scanner." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 9, 2016.

[13] K. Li, W. Ma, U. Sajid, Y. Wu, and G. Wang, "Object Detection with Convolutional Neural Networks," *arXiv:1912.01844 [cs]*, Dec. 2019, arXiv: 1912.01844. [Online]. Available: http://arxiv.org/abs/1912.01844

[14] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA: IEEE, Sep. 2016, pp. 3464–3468. [Online]. Available: http://ieeexplore.ieee.org/document/7533003/

[15] S. Blackman and R. Popoli, "Design and analysis of modern tracking systems(book)," *Norwood, MA: Artech House, 1999.*, 1999.

[16] J. S. Berrio, M. Shan, S. Worrall, and E. Nebot, "Camera-lidar integration: Probabilistic sensor fusion for semantic mapping," *arXiv preprint arXiv:2007.05490*, 2020.

[17] M. Sorial, I. Mouawad, E. Simetti, F. Odone, and G. Casalino, "Towards a real time obstacle detection system for unmanned surface vehicles," in *OCEANS 2019 MTS/IEEE SEATTLE*, 2019, pp. 1–8, tex.organization: IEEE.

[18] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim, "Calibration between color camera and 3d lidar instruments with a polygonal planar board," *Sensors*, vol. 14, no. 3, pp. 5333–5353, 2014.

[19] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[20] G. Guennebaud, B. Jacob *et al.*, "Eigen," *URl: http://eigen. tuxfamily. org*, 2010.

[21] W. J. Schroeder, B. Lorensen, and K. Martin, *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[22] "AUV Lab – Datasets – Marine Perception – Overview – MIT Sea Grant."

[23] "What does the term UTM mean? Is UTM better or more accurate than latitude/longitude?" https://www.usgs.gov/faqs/what-does-term-utm-mean-utm-better-or-more-accurate-latitudelongitude?qt-news_science_products=0#qt-news_science_products.