Multimodal perception for obstacle detection for flying boats - Unmanned Surface Vehicule (USV)

Ronan Douguet, Dominique Heller, Johann Laurent

Multimodal perception for obstacle detection for flying boats - Unmanned Surface Vehicle (USV)

Ronan Douguet
Lab-STICC UMR 6285
CNRS
Lorient, France
ronan.douguet@univ-ubs.fr

Dominique Heller
Lab-STICC UMR 6285
Université Bretagne Sud
Lorient, France
dominique.heller@univ-ubs.fr

Johann Laurent
Lab-STICC UMR 6285
Université Bretagne Sud
Lorient, France
johann.laurent@univ-ubs.fr

***Abstract*—Unmanned surface vehicles (USVs) are increasingly used in bathymetric survey, maritime surveillance, and maintenance applications. However, detecting obstacles in the maritime environment poses significant challenges due to sea clutter, background variability, and other factors. Although various sensors such as radar, AIS, LiDAR, and camera have been used for obstacle detection, they have limitations in terms of range and effectiveness, especially at different USV speeds. In this paper, we present a real-time obstacle detection system for USVs in the maritime environment by explaining the used sensors, the architecture and the implemented algorithms. We focus on evaluating the performance of the YOLOV7 network, which forms the basis of our fusion algorithm. Our results demonstrate that our system can effectively detect maritime objects in real-time, providing improved safety and efficiency for USV operations. Additionally, we provide an open solution for visualizing the detected targets on a chart plotter navigation.**

***Index Terms*—Sensor fusion, obstacle detection, USV, YOLOv7, ROS**

I. Introduction

The USV encompasses a wide range of boats, with sizes ranging from several decimetres to more than twenty metres, depending on their mission requirements. In our specific case, we are working on a seven-metre rigid inflatable boat (RIB) that is equipped with two foils (cf. Fig. 1). There are two main advantages to using a flying boat. Firstly, the foils enable a reduction in fuel consumption, thus increasing the vessel's operational range. Secondly, the boat offers increased stability, resulting in reduced sensor vibrations, which is crucial for accurate obstacle detection. The vessel is capable of reaching speeds of up to 30 knots in calm sea conditions, up to sea state 2.

To operate a USV safely, it is essential to accurately detect obstacles on the sea surface. Many reviews [1] are available for autonomous vehicles about self-driving cars, and it is useful to seek inspiration from them since we use similar sensors, such as LiDAR, camera, and radar, in maritime vehicles. Moreover, other sensors are vital on vessels, such as the Automatic Identification System (AIS), Inertial Measurement Unit (IMU), and maritime radar. These sensors are now being commonly used on USVs [2], [3], and their usefulness depends on the navigation area (port, offshore, etc.) and boat speed.

Despite weather conditions such as haze, fog, rain, bright sunlight, and twilight which affecting camera vision, it remains

Fig. 1. Flying boat using for testing (*built by the SEAir company)*

one of the most commonly used sensors for obstacle detection due to its relatively low cost. Different neural networks have often been tested and compared to find the best solution for detecting maritime objects [4], [5]. However, fusing this detection with other sensors requires synchronization and calibration of these sensors.

In this work, we present a solution for implementing real-time obstacle detection. We detail the performance of our trained neural network based on the You Only Look Once (YOLOv7) networks [6].

This paper is organized as follows: Section II presents the sensors, synchronization, and calibration steps. Section III describes the architecture we chose for implementing real-time detection. Section IV presents the performance we obtained with the trained YOLOv7 network. Section V shows the initial results of sensor fusion, while Section VI proposes a solution to integrate the detected targets on a open navigation chart. Finally, Section VII concludes the paper.

II. Sensors

A. overview

The sensor selection and placement on the USV are critical for successful obstacle detection. The sensors must be positioned to provide maximum visibility without interfering

TABLE I
LIST OF SENSORS

	Sensors	Company	Model	Range (m)	HFOV (°)	VFOV (°)
Detection	AIS	McMurdo	Smartfind M15	37000	-	-
	Camera RGB	Basler	acA1920-50gc	400	40,1	30,3
	Camera monochrome	Hikrobot	MV-CA020-20GM	400	74	57
	Camera thermal	Teledyne Flir	FLIR ADK	300	75	60
	LiDAR 32-channel	Hesai	PandarXT-32	120	360	31
	LiDAR 64-channel	Ouster	OS1-64	120	360	45
	Maritime radar	Simrad	Halo20+	40000	360	25
Position	GPS	u-blox	ZED-F9P	-	-	-
	IMU	SBG	Ellipse-E	-	-	-

Notes: The acronyms form left to right (in fist row) are Horizontal Field Of View (HFoV), Vertical Field Of View (VFoV). For cameras these parameters depend on the variant of focal length.

with each other. In our project, we have developed a sensor platform (cf. Fig. 2) that can be easily integrated into different types of vessels. This platform is mounted at the front of the flying boat and is equipped with several sensors, including AIS, GPS, cameras (RGB, monochrome, and thermal), IMU, and LiDAR. Due to space constraints, the maritime radar is installed separately from the platform.

During our project, we tested two LiDAR sensors: the PandarXT-32 with 32 beams from Hesai company, and the OS1-64 with 64 beams from Ouster company. The OS1-64 has better resolution with its 64 beams, but both sensors have similar maximum ranges near to 120m. Additionally, we integrated several RGB and monochrome cameras to test the impact of color on our detection algorithm and to determine the best field of view for our application.

Fig. 2. Sensor platform

Each sensor has a different range and coverage (cf. Tab. I), but they are complementary. For instance, when the USV is traveling at 25 knots, the essential sensors for detecting targets are the radar and AIS. However, when the USV is traveling between boats at anchor at speeds of 3-5 knots, detection by cameras and LiDAR is highly effective. It is important to consider the range of the sensors when integrating obstacle avoidance in future works. The sensors can be categorized into two groups based on their range: close detection for short-term avoidance and far detection for long-term avoidance which can be integrated with a prediction path planning algorithms that comply with the COLREGs (*Convention on the International Regulations for Preventing Collisions at Sea*).

B. Synchronisation

A synchronization step is necessary to be correctly merge data provided from different sensors. Without synchronization, the detected target may be incorrect, and false detection can create hazards.

As the sensors do not have the same communication link, there are several ways to synchronize them using hardware or software [7] [8]. In our system, the atomic clock time transmitted via GPS is used as the basis to synchronize all the sensors and the processor. The principle is detailed in Figure 3. The Pulse Per Second (PPS) signal coupled with the GPS NMEA frame synchronize the LiDAR, IMU, and processor. Then, the hardware trigger of the camera is used to start each image acquisition. The trigger clock, which is derived from the PPS signal, is provided by the IMU. The maritime radar is also synchronized with GPS through the OpenCPN software.

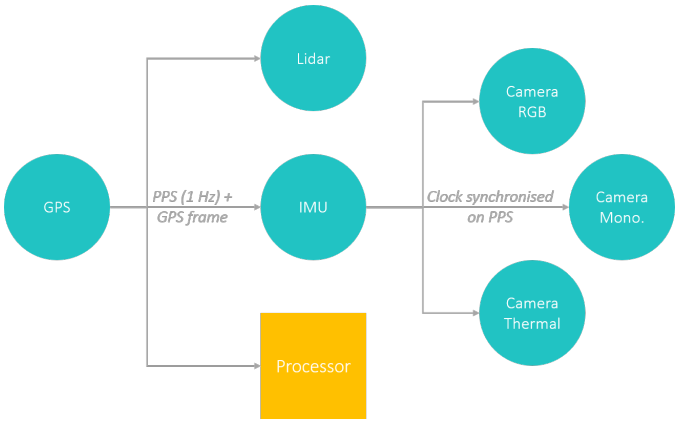

Fig. 3. Synchronization diagram

C. Calibration

In order to detect targets accurately, it is necessary to properly calibrate the sensors used in the detection process.

There are two main steps involved in the calibration process: intrinsic calibration and extrinsic calibration. Intrinsic calibration involves adjusting each individual sensor and extrinsic calibration, on the other hand, involves adjusting the sensors to ensure that they are all in the same coordinate system.

As the intrinsic parameters of LiDAR and radar are already calibrated during the factory process, we have only to calibrate the cameras. Indeed, they often require manual calibration to ensure accurate measurements. This calibration allows to correct the lens distortion and to define the projection matrix from 3D camera coordinates into 2D image coordinates.

Among the various methods for camera calibration, Zhang's Method [9] is one of the most widely used techniques. This approach involves utilizing a checkerboard pattern to capture images of the camera from different angles and orientations. The underlying principle of this method is based on the pinhole camera model, which mathematically describes the relationship between the coordinates of a point in three-dimensional space and its projection onto the image plane. This relationship is expressed using homogeneous coordinates and is detailed by the following equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

where (u, v) represents the coordinates of the projection point in pixels; f_x, f_y are the focal lengths; c_x, c_y are the coordinates of principal point (which is typically the center of the image); (X, Y, Z) are the coordinates of 3D points.

In addition to modeling the pinhole camera model, it is important to account for lens distortion in order to obtain accurate calibration results. Distortion coefficients are calculated as part of this process and are subsequently used to project 3D point cloud data from lidar or radar sensors into 2D image coordinates. To calibrate our cameras, we utilized the camera calibration ROS package [1]. This tool enabled us to obtain intrinsic and distortion parameters for our calibrated camera, which were crucial for achieving high-quality imaging results.

The second step, extrinsic calibration, is important for ensuring that the data from each sensor can be accurately combined to create a complete picture of the target. Several automatic methods are available for extrinsic calibration [10], [11], [12], including those based on Aruco markers and calibration panels with circles [13]. This last method has been tested for our project to calibrate the couple LiDAR-camera (cf. Figure 4). This method consists of identifying the centers of the four calibration panel circles in both the point cloud and the image. For cameras, the Aruco markers are used to determine these centers, while for lidar, the circle edges are used instead. Once the centers have been located, a mathematical model is utilized to compute the precise rotation and translation matrix required to align the two sensors. This result has been used to project the point cloud onto the image. The following section explains the architecture on which we have deployed these calibration methods.

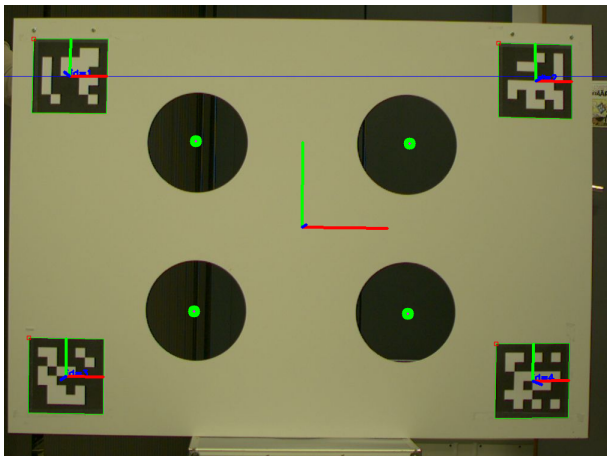

Fig. 4. Panel Calibration LiDAR-Camera

III. Architecture

The architecture of our system is built around the Jetson AGX Xavier developer kit. This board is an ideal choice due to its powerful system-on-chip, which includes an eight-core ARM CPU clocked at 2.26 GHz, a Volta GPU clocked at 1.37 GHz with 512 CUDA cores, 64 Tensor cores, and a deep learning accelerator. In addition, the board has 16 GB of memory and multiple connectivity options, such as Gigabit Ethernet, USB 3.0, CAN, and HDMI, making it easy to integrate various sensors.

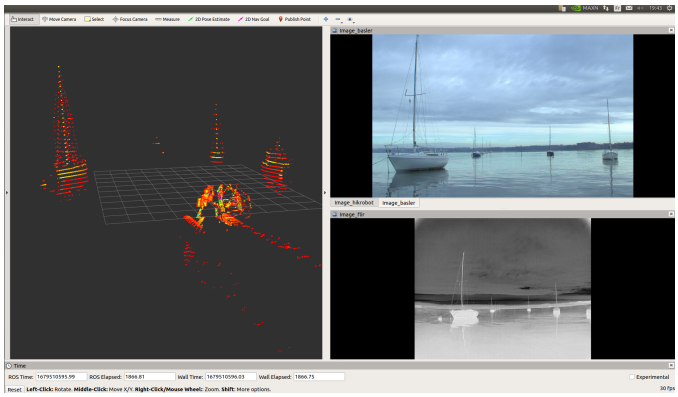

Fig. 5. Rviz tool to view point cloud and video camera

To ensure modularity and facilitate rapid software development, we have chosen to use the Robotic Operating System (ROS) framework. Although initially developed for research and education, ROS is increasingly being used in industrial applications such as robotics, machine vision, and data acquisition. One of the main advantages of using ROS is its support

[1] camera$_c alibration$: $http://wiki.ros.org/camera_calibration$

for modular and scalable software development. It also offers a range of useful tools and libraries, such as RVIZ (shown in Figure 5), which makes it easy to visualize early results. Additionally, many sensor manufacturers (e.g., SBG, Hesai, Ouster, Basler) provide ROS drivers for their devices. There are several versions of ROS available, and our project operates to ROS Noetic.

The ROS framework provides another advantage, including the use of the ROS bag tool for recording all ROS messages in a specific format for later playback and analysis. In our project, we leveraged this tool to create a database for our neural network. This approach enabled us to refine our fusion algorithms during post-processing and ensure optimal performance prior to real-time deployment.

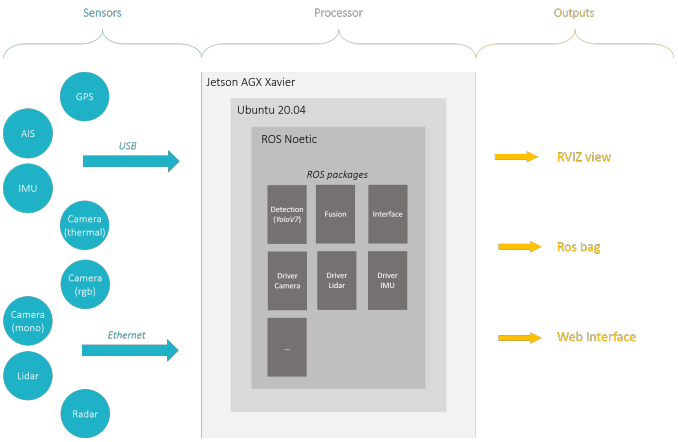

Fig. 6. Architecture

The block diagram of our system architecture is depicted in Figure 6. All sensors are connected to the Jetson Xavier through Ethernet or USB, and the necessary software components such as drivers, detection algorithms, fusion algorithms, and user interface have been developed as ROS packages. The system is capable of logging all sensor data in rosbag format, and can display both raw data and detected targets using Rviz or a custom application. Additionally, the system is designed to display detected targets on a nautical chart using the OpenCPN software, which will be further explained in the subsequent section. All electronic components are housed in a waterproof box equipped with Li-ion batteries that provide approximately 800 Wh of power, allowing for up to 8 hours of autonomous operation. We utilized this platform to implement the detection algorithm developed in the next section.

IV. Detection

A. Overview

As explained at the beginning of this article, each sensor in our system covers a specific portion of the guard zone, as indicated in Table I. The radar and AIS provide the position, speed, and course of the detected target directly, which the system can readily read and parse. In contrast, we need to develop algorithms to detect targets from the image camera and point cloud data. In this section, we will present the neural network that we use and a basic fusion of camera and LiDAR data to obtain the target's 2D coordinates.

B. YOLOv7

In our previous article [14], we proposed a deep learning-based solution for detecting and classifying maritime objects using convolutional neural networks (CNNs). During our study, we evaluated the performance of several networks and selected YOLOv4 due to its high detection performance and rate. We compared the performance of YOLOv4 across various platforms. We continued our work by updating the model with YOLOv7 [6], which offers improved accuracy and scalability compared to its predecessors.

In particular, we compared the performance of YOLOv7 and YOLOv4 on our dataset with 416x416 resolution images. The evaluation showed that YOLOv7 improved the accuracy by 2% and provided a 46% increase in frames per second (FPS) compared to YOLOv4. Therefore, we selected YOLOv7 for our real-time implementation.

In addition to updating our model, we have continued to enhance our maritime object dataset. This dataset comprises RGB images and consists of 40,876 images with 59,386 annotations across 41 classes (see Figure 7). It has one of the most extensive collections of classes and annotations among available datasets. We included various classes from the AIS ship type, as well as specific subtypes that we identified on marine traffic [2]. Moreover, for research and rescue mission [15], we have incorporated a new class, "Swimming Person," to accurately represent individuals swimming on the sea surface. This addition enhances the specificity of our dataset, which also includes other objects such as jet-skis, kayaks, small boats, and submarines [15].

TABLE II
Performance and results on Jetson AGX Xavier

Model	Image Resolution	mAP	FPS (FP32 model)	FPS (FP16 model)
YOLOv7-tiny	416x416	0.819	48	140
YOLOv7	416x416	0.861	35	50
YOLOv7	608x608	0.868	31	45
YOLOv7	640x640	0.860	30	39

This dataset has been used for the training process; it has been conducted by using the official YOLOv7 tools on a Nvidia RTX 3090. We trained several version of YOLOv7 networks to compare their performances : YOLOv7-tiny, YOLOv7-416, YOLOv7-608 and YOLOv7-640. Each model is trained for 300 epochs during 17.713 hours for the YOLOv7-Tiny up to 54 hours for the YOLOv7 using 640x640 resolution. Each trained models are validated by using the validation dataset. The mean average precision (mAP) is computed for each 300 epochs based on the AP@[0.5:0.95] metric (the Intersection over Union "IoU" values from 50% to 95%, at a step of 5%).The results are show in the table II.

[2]MarineTraffic: https://www.marinetraffic.com/

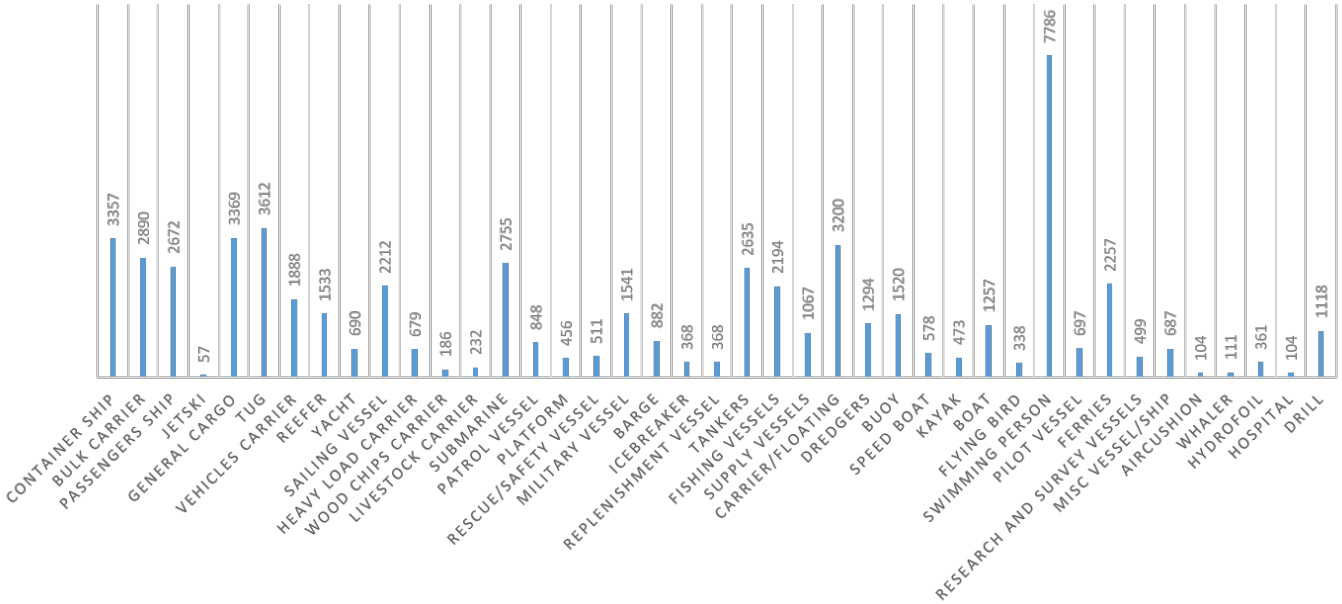

Fig. 7. Distribution of Image per class

Then we compared the inference speed of the trained networks on the Jetson Xavier AGX board. To optimize and accelerate the inference of our deep neural networks on the GPUs, we deployed each trained model using TensorRT. TensorRT is an software development kit (SDK) developed by Nvidia that provides a high-performance deep learning inference engine. We converted each model to a TensorRT engine, first to an FP32 representation and then to an FP16 one. This approach is beneficial for improving inference speed and reducing power consumption.

We have provided detailed speed inference results in Table II. The FPS obtained for each model are more than sufficient for our needs, as we aim to operate the inference at 20 Hz. The cameras are synchronized to 20 Hz, allowing us to detect targets every 50 ms. This time interval is not critical for our application, even when the boat is sailing at 30 knots. The major challenge in this case is being able to detect targets beyond 200 meters, in order to have enough time to avoid obstruction by taking appropriate actions such as stopping the USV or changing its course. Therefore, in fast speed mode, the radar and AIS are prioritized for detecting targets at longer ranges. To enable real-time inference, we have developed a ros_object_detect package based on the TensorRT library. This package subscribes to an image topic and provides two outputs: the image result and a specific ROS message that sends all detections for each frame. This package will be used for data fusion

C. Fusion Data

In this section, we will explain the data fusion between the detected targets in the image and the LiDAR. The same principle can also be used to merge detected targets from the radar or AIS. The aim is to project the 3D coordinate points from the LiDAR sensor onto the image and, specifically, only

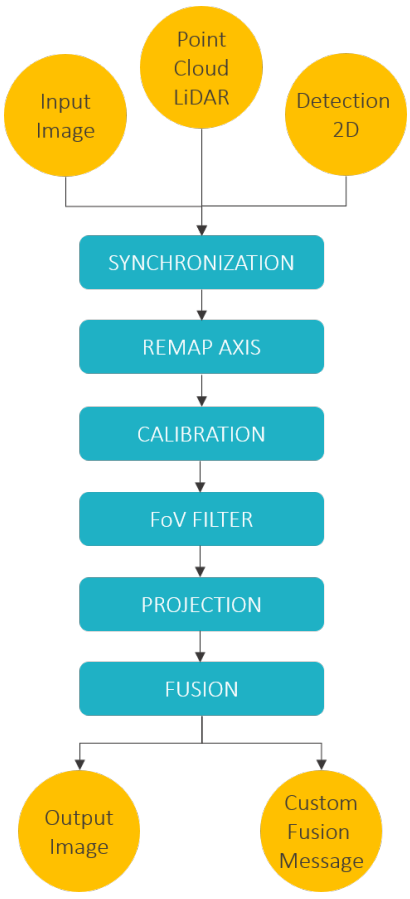

Fig. 8. Flow chart of fusion camera LiDAR

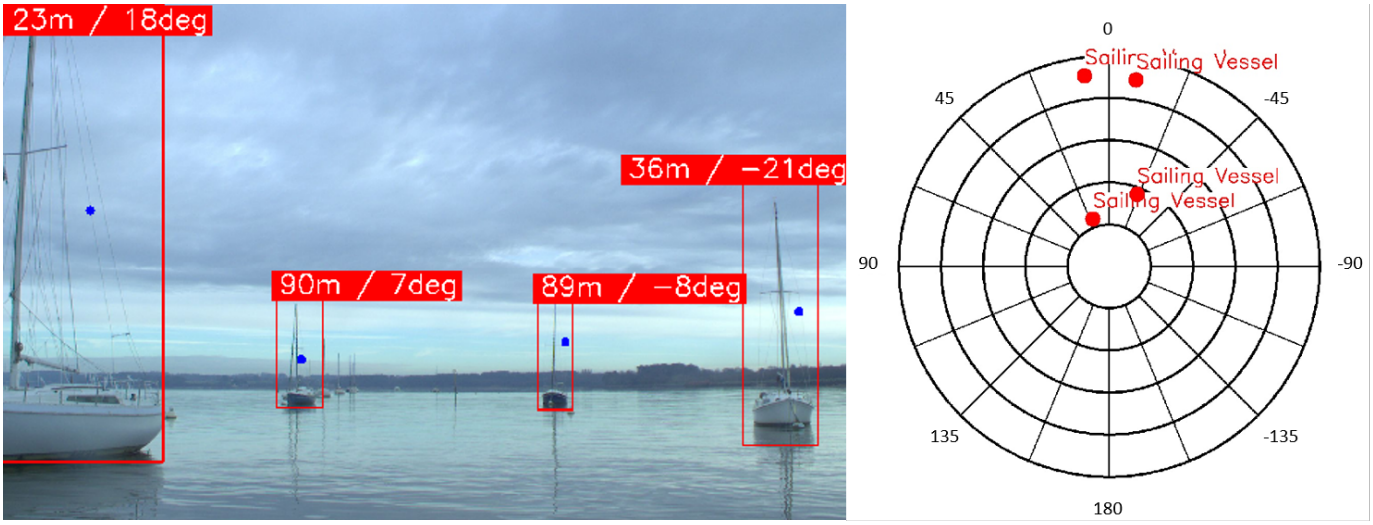

Fig. 9. Example of detection by merging camera-LiDAR

within the detected target boxes. By doing so, we are able to obtain the average distance and azimuth between the detected targets and the USV.

To fuse the detected targets from the LiDAR and image data, several steps are required, as illustrated in Figure 8. The first step is to synchronize the 2D detection, which consists of a list of detected target box coordinates, with the correct image and the closest cloud points in time. The next step involves remapping the LiDAR axis to use the same reference as the camera axis, followed by applying the translation and rotation matrix (extrinsic calibration matrix) to put the point cloud in the camera coordinates. A filter step is then used to retain only the point cloud that is in the camera's field of view, before projecting all 3D point clouds into the 2D image. Finally, the fusion step involves averaging the mean distance and azimuth of each detected box. This algorithm has been implemented in C++ using a ROS package, which generates an image displaying the distance and angle of any detected objects. In addition, a custom message containing the coordinates of identified targets is also produced as output. An example of the output is shown in Figure 9, which presents the target on both a picture and classic radar view. Furthermore, the custom message is utilized to display the target on a navigational chart, as explained in the next section.

V. Navigation Chart

Visualization of detected targets is a crucial aspect of obstacle detection for USV operators, as it enables them to navigate safely in open waters with the aid of navigation charts. In addition to the camera detection views and custom radar view, we propose that users view targets through a navigation chart. To achieve this, we selected the OpenCPN software, which is a free and open-source navigation tool that provides users with a wide variety of features and tools to help them to navigate waters safely. The software is compatible with multiple platforms, including Windows, Linux, and macOS, and supports a range of chart formats such as S-57, S-63, and BSB/KAP. Additionally, it offers several plugins, including the radar_pi plugin, which enables users to configure the maritime radar and define a guard zone to detect and track obstacles using Automatic Radar Plotting Aid (ARPA).

In our project, this software is used to display the USV position and predicted route, as well as all detected targets by the AIS, the radar, and the fusion of camera-LiDAR. Displaying the USV position and predicted route allows operators to navigate with confidence and avoid obstacles. To achieve this, a UDP communication link is established between the Jetson Xavier AGX and the navigation computer. The data are exchanged through the NMEA protocol, which is a standardized communication protocol in marine electronics defined by the National Marine Electronics Associations (NMEA).

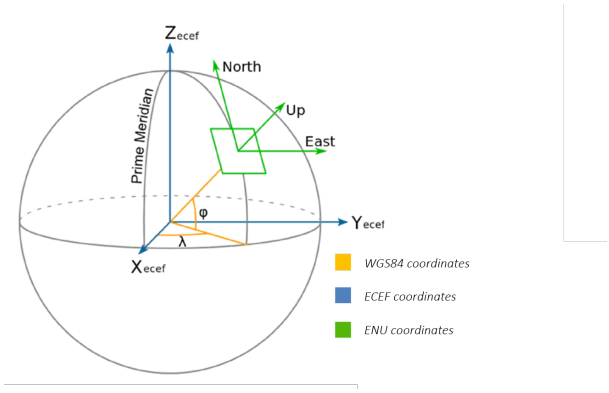

Fig. 10. The different coordinates

In order to display detected targets on the navigation chart, they must be in the World Geodetic System (WGS) coordinates. The AIS and the radar coupled with the GPS already provide the data in these coordinates. However, the detected targets from the fusion of camera-LiDAR are expressed the

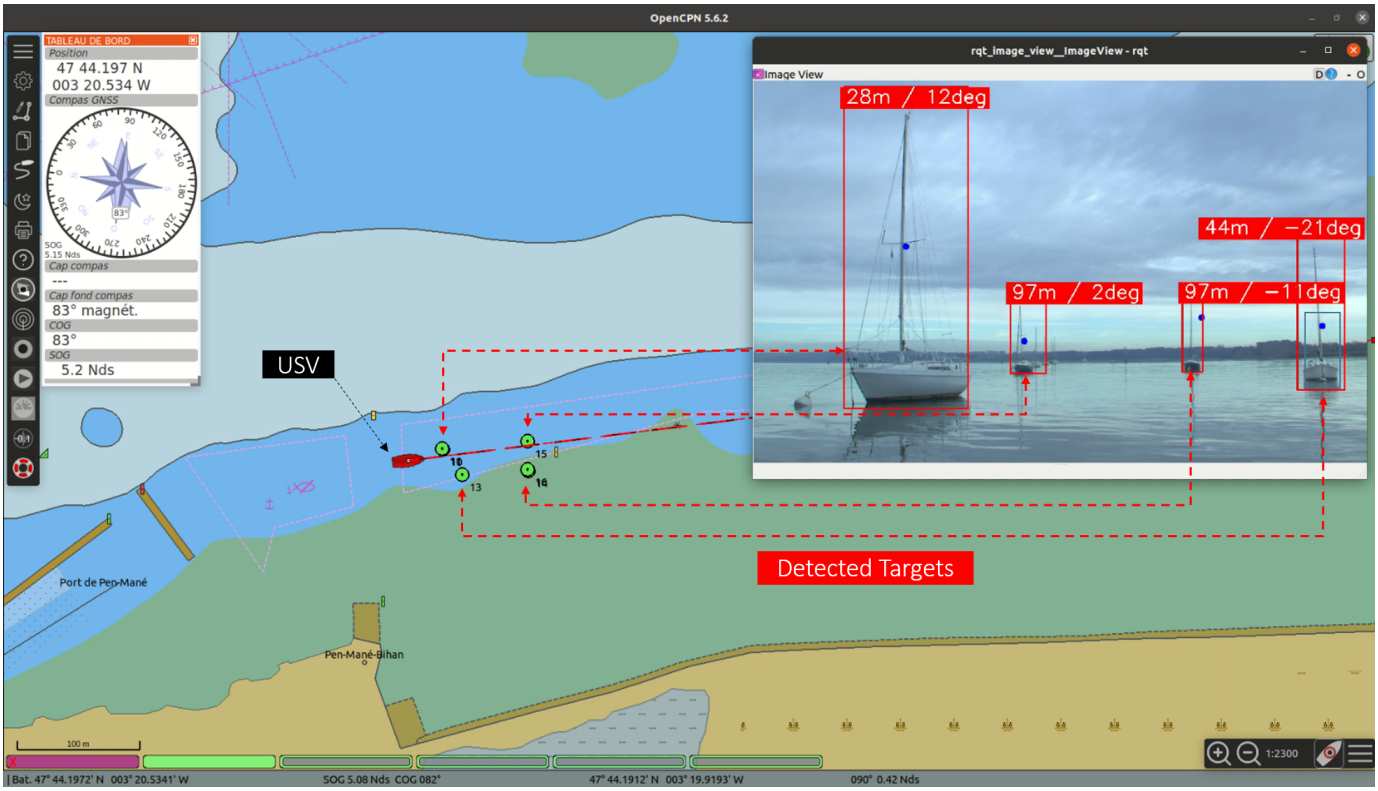

Fig. 11. Example View of detected target display on OpenCPN

East-North-Up (ENU) system which is a local plan. Therefore, a conversion step is necessary to transform these data into the WGS coordinates. This involves transforming the coordinates from the ENU system to the Earth-Centered-Earth-Fixed (ECEF) system, and then to the WGS84 coordinate system. These different coordinates are shown in Figure 10. Once this data is converted, it is possible to display them on OpenCPN, as shown in Figure 11. By integrating detected targets into the navigation chart, operators can make informed decisions about how to navigate based on factors such as water depth, rocks, and other hazards.

VI. Conclusion

In conclusion, this paper proposes a novel solution for real-time obstacle detection on the sea surface using a fusion of classical maritime sensors. Leveraging the ROS framework and Jetson AGX Xavier device for calibration, sensor synchronization, and fusion algorithms implementation, our approach demonstrates the benefits of fusing camera-LiDAR or radar for detecting targets in close proximity. We have successfully trained and tested YOLOv7 models with different configurations to identify the optimal network for our application. Furthermore, we have integrated navigational chart software into our solution to provide real-time visualization of detected targets in the environment, aiding USV operators in making informed decisions about avoiding obstacles. The effectiveness of our solution has been validated through real-world testing on the foiling RIB.

The results of this research show promising potential for real-time obstacle detection on the sea surface, with applications in various marine industries. Future work will focus on further improving the precision of the fusion process by incorporating point cloud classification techniques from the created kitti point cloud dataset for marine object detection, as well as exploring semantic instance segmentation methods. Additionally, we are working on developing a trained dataset using thermal images to enable detection in challenging conditions such as low light or nighttime scenarios. The ultimate goal is to integrate all detected targets into the obstacle avoidance algorithm to enable complete automation of the drone system.

References

[1] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.

[2] J. Han, Y. Cho, J. Kim, J. Kim, N.-s. Son, and S. Y. Kim, "Autonomous collision detection and avoidance for aragon usv: Development and field tests," *Journal of Field Robotics*, vol. 37, no. 6, pp. 987–1002, 2020.

[3] M. DeFilippo, M. Sacarny, and P. Robinette, "Robowhaler: A robotic vessel for marine autonomy and dataset collection," in *OCEANS 2021: San Diego–Porto*. IEEE, 2021, pp. 1–7.

[4] F. Farahnakian and J. Heikkonen, "Deep learning based multi-modal fusion architectures for maritime vessel detection," *Remote Sensing*, vol. 12, no. 16, p. 2509, 2020.

[5] W. Zhang, F. Jiang, C.-F. Yang, Z.-P. Wang, and T.-J. Zhao, "Research on unmanned surface vehicles environment perception based on the fusion of vision and lidar," *IEEE Access*, vol. 9, pp. 63 107–63 121, 2021.

[6] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.
[7] J. Wu, Z. Xiong *et al.*, "A soft time synchronization framework for multi-sensors in autonomous localization and navigation," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2018, pp. 694–699.
[8] P. Merriaux, R. Boutteau, R. Rossi, G. Coru, V. Vauchey, and X. Savatier, "Synchronisation et calibrage entre un lidar 3d et une centrale inertielle pour la localisation précise d'un véhicule autonome," in *Journées scientifiques d'URSI-France (Géolocalisation et Navigation dans l'Espace et le Temps)*, 2018.
[9] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
[10] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 1–6.
[11] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "Lidar-camera calibration using 3d-3d point correspondences," *arXiv preprint arXiv:1705.09785*, 2017.
[12] J. Peršić, I. Marković, and I. Petrović, "Extrinsic 6dof calibration of a radar–lidar–camera system enhanced by radar cross section estimates evaluation," *Robotics and Autonomous Systems*, vol. 114, pp. 217–230, 2019.
[13] J. Beltrán, C. Guindel, A. de la Escalera, and F. García, "Automatic extrinsic calibration method for lidar and camera sensor setups," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 677–17 689, 2022.
[14] D. Heller, M. Rizk, R. Douguet, A. Baghdadi, and J.-P. Diguet, "Marine objects detection using deep learning on embedded edge devices," in *IEEE International Workshop on Rapid System Prototyping (RSP), part of Embedded Systems Week (ESWEEK)*, 2022.
[15] M. Rizk, D. Heller, R. Douguet, A. Baghdadi, and J.-P. Diguet, "Optimization of deep-learning detection of humans in marine environment on edge devices," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2022, pp. 1–4.