Design and fabrication of a low cost autonomous surface vehicle for assessing marine environment

Abstract

1. Introduction

Design and develop an underwater vehicle capable of conducting comprehensive marine life studies and surveillance operations, aiming to enhance understanding of aquatic ecosystems, monitor marine species behaviour, and contribute to environmental conservation efforts. In the present study embedded a raspberry pi camera for Observing and documenting the behaviours, social interactions, and migration patterns of marine species. Figure 1 presents the conceptual idea and applications. Monitoring population sizes, distributions, and the health of different marine species to assess biodiversity and ecosystem health. Integrating the Raspberry Pi camera with the underwater vehicle allows for detailed visual documentation, which is crucial for conducting comprehensive fish studies and contributing valuable data to marine biology research. Using high-resolution cameras and imaging systems to identify and catalogue various marine species, including rare and endangered ones. Creating educational materials and outreach programs by sharing captivating underwater footage to raise awareness about marine life and conservation efforts.

Investigating different marine habitats, such as coral reefs, seagrass beds, and deep-sea environments, to understand their structure, composition, and biodiversity. Surface vehicle assist underwater vehicle to transport to the observatory location. It is carried with various water assessment sensors for the Measuring and recording environmental parameters such as water temperature, salinity, pH levels, and dissolved oxygen to study their effects on marine life. In addition, ping sonar is incorporated at the bottom side of the hull at 90° to the seabed surface for mapping and analysing different marine habitats, including coral reefs, seagrass beds, and deep-sea environments.

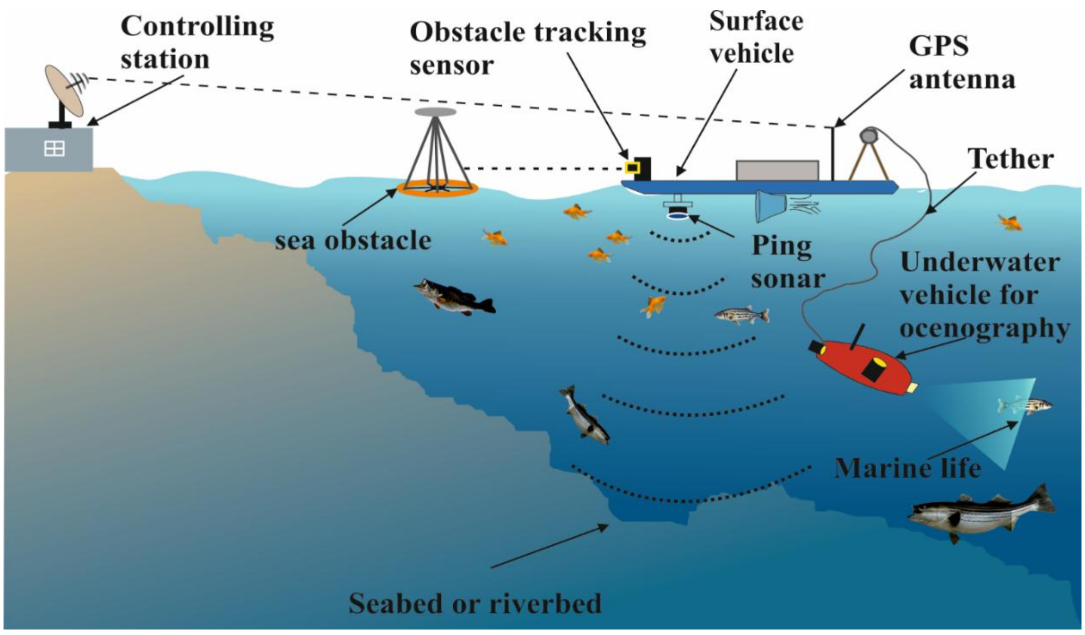


Fig. 1 operating concept of the integrated platform developed in this study

2. Design of the autonomous surface vehicle

- In design of the autonomous surface vehicle comprises hull, control unit, propulsion unit, and sensors unit. Framework of complete system is shown in figure 2.
- Two thruster are used to propel the vehicle and control the directions, t-200 thruster is used which capable to generate the 14 kg thrust at the payload condition. Same motor can generate the reverse thrust that make to vehicle more easy to control at 2 degree at freedom each thruster has 12V - 22.2 V operating rate.
- Surface vehicle design and developed for the water quality assessment. Multiple sensors Viz, temperature, TDS, PH and turbidity sensor are incorporated on the hull to collect the data from river, pond and lakes.
- It comprises of hull structure, measuring unit, propulsion motors, sensors and control unit as shown n Figure 2 Material used in the hull body is Polypropylene foam and dimension is 100 cm × 50cm × 5.7cm. At the bottom side two T200 thruster is fixed to propel the vehicle and its operating thrust for single thruster is 3.71 - 6.7 kg per force at the voltage range 10v - 22v. This vehicle takes turn by varying the speed between two thrusters, also can move in backward direction.
- Hull body is capable to carry the total payload 68 kg without disturbing the stability of the vehicle, control unit and data acquiring unit is situated on the board at the center of the vehicle. Sensors are placed at the front side, so that it deployed underwater.
- Vehicle controlled by the remote transmitter manually and its receiver is situated into the control unit waterproof casing. Receiver and thrusters are connected with teensy 4.1 microcontroller board. Thruster speed control using ESC (Electronic speed controller) wire connection of the thruster such as positive, ground and signal should properly connects with ESC's connections.
- This autonomous surface vehicle can follow the predefined path by analysing the GPS, magnetometer and acceleration sensors data

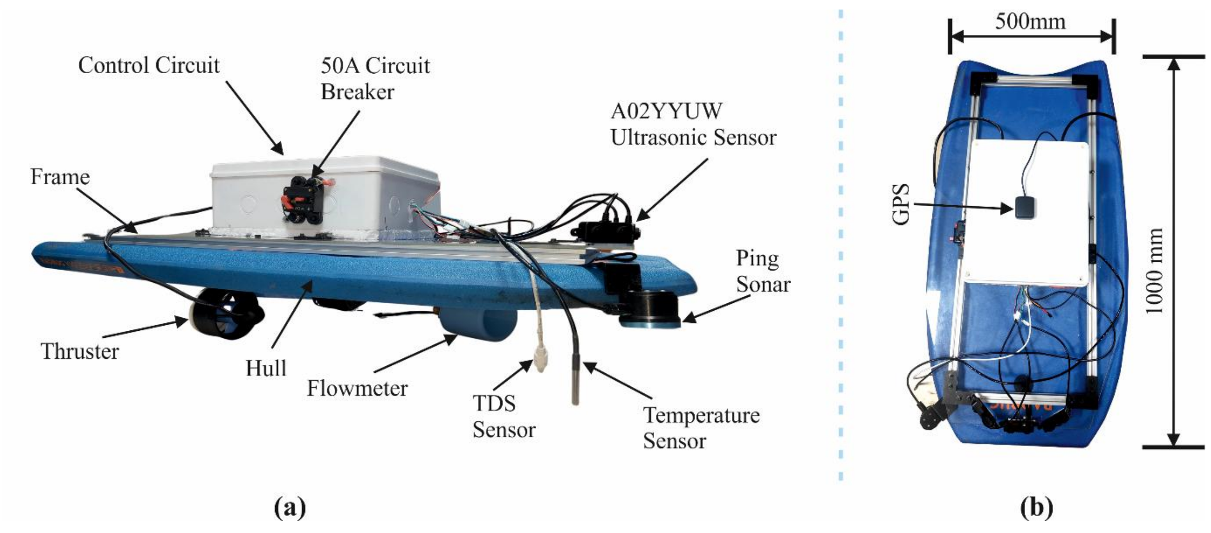


Fig. 2 Layout of the autonomous surface vehicle (a) component of the surface vehicle (b) dimension of the vehicle hull

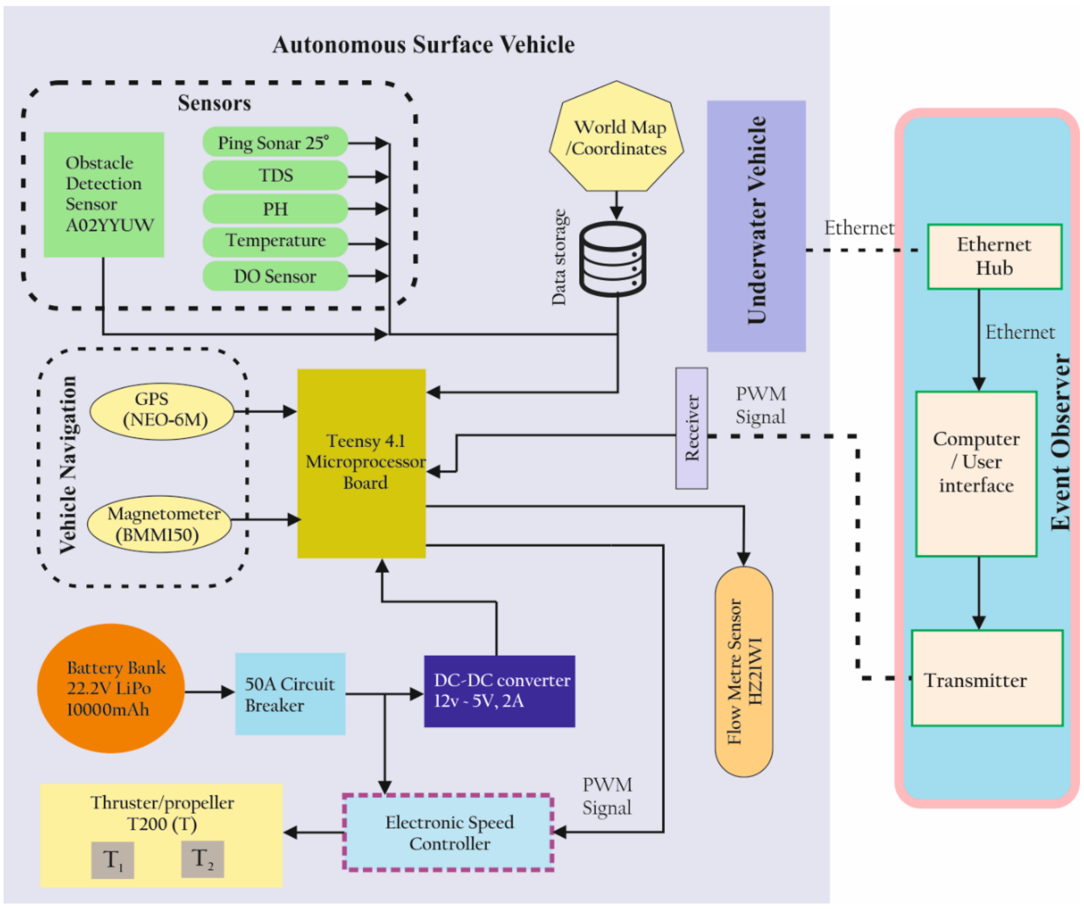


Fig.3 Block diagram of autonomous surface vehicle

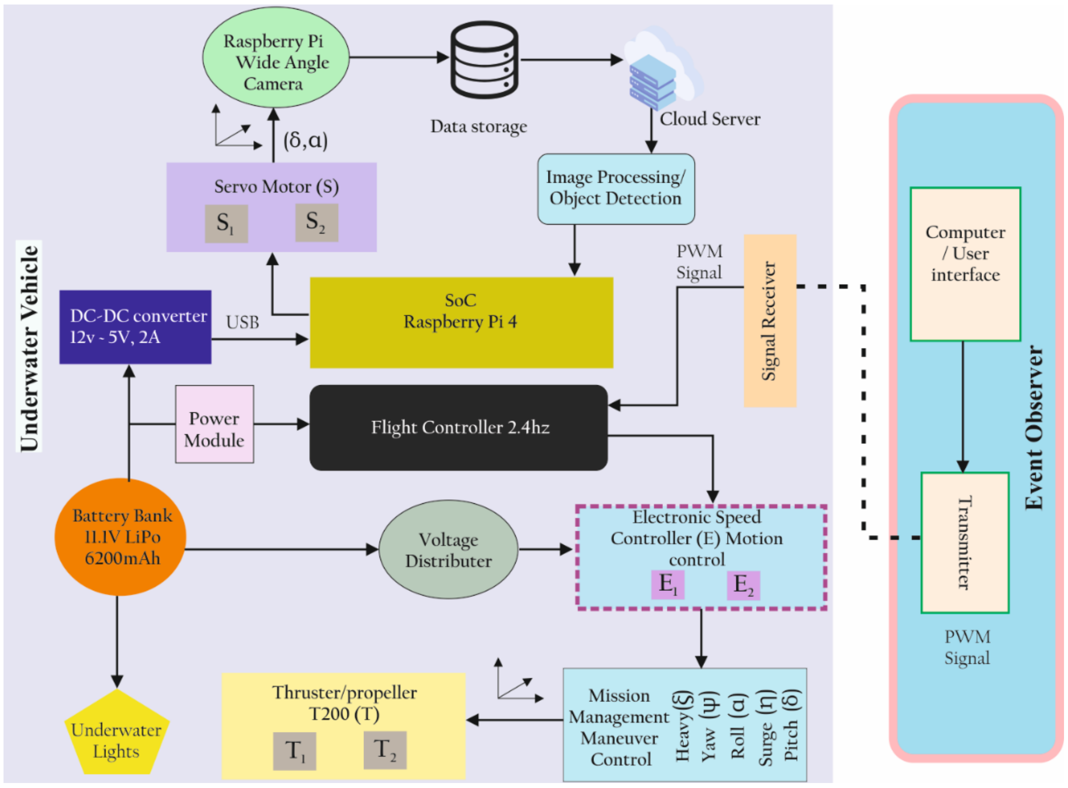


Fig.4 Block diagram of underwater vehicle

2.1 Components and features

- **Microcontroller: Teensy 4.1**

- The Teensy 4.1 serves as the brain of the Autonomous Surface Vehicle, orchestrating all operations including autonomous navigation and real-time sensor data collection. It boasts high processing power, approximately 10 times faster than the Arduino Mega, and low latency for real-time decision-making. The Teensy 4.1 is capable of running multiple functions simultaneously, such as collecting data while autonomously navigating the surface vehicle using multiple ultrasonic sensors. Figure 3 and 4 illustrates the schematic of internal connection of autonomous surface vehicle and underwater vehicle

2.2 Navigation System

2.3.1 Autonomous Mode

- **Waypoint Navigation**
  - Reads goal coordinates from the SD card.
  - Calculates paths to reach waypoints by using gps and magnetometer data. Additional, Adapts to changing conditions (river currents, wind).

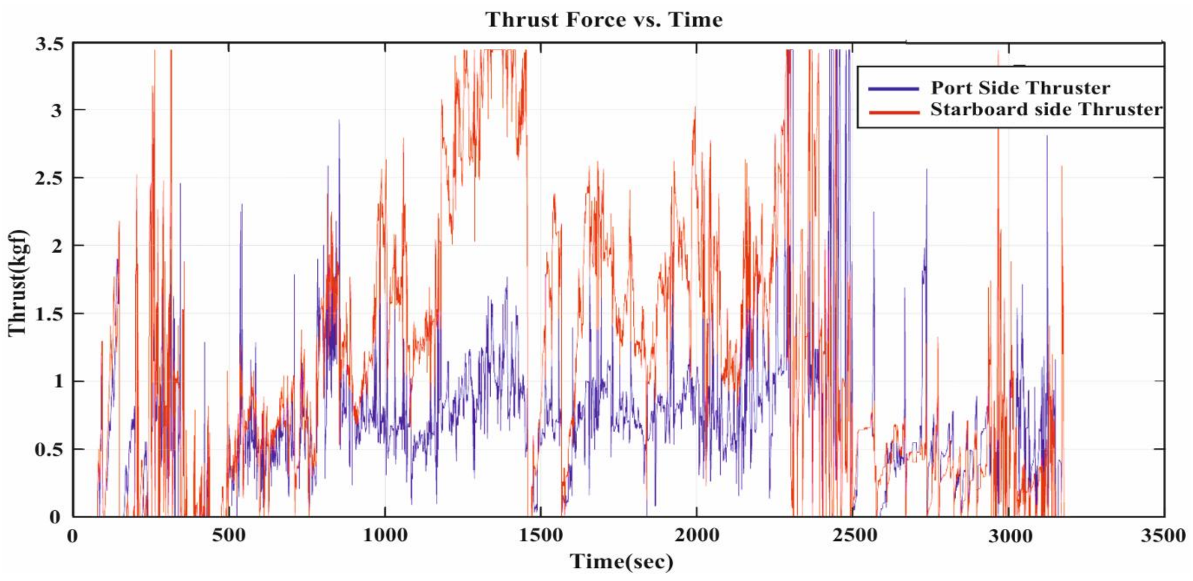


Fig. 5 Graph of operating thrust versus time of the autonomous surface vehicle thruster

Figure 5 shows the power drop could indicate changes in the performance of the thruster motor over time or under different operational conditions. Studying this can provide insights into how the vehicle performs in different environments or during different phases of operation. Power drop typically occurs due to various factors such as weed stuck inside the propellers, current draw, and battery condition.

2.3.2 Manual Mode

- Allows remote control by an operator.
- Useful for testing, calibration, or emergency situations.

2.3 Propulsion System

- **T200 Thrusters**

- Two powerful thrusters:
- **Port Side Thruster** Clockwise orientation
- **Starboard Side Thruster** Counter clockwise orientation
- Differential thruster speed adjustment for steering motion.

Algorithm 1: Executing the autonomous path generation algorithm for ASV's (autonomous surface vehicle)

```
Require current.Lat, current.Long, $C_{\text{array}}$; {Current GPS coordinates, waypoint array}
Require $W_p$; {Numbers of waypoints}
Require $N_{\text{coords}}$; {Total number of waypoints}
1: Initialize goalLat, goalLong, goalheading, $H^{diff}$, goalport, goalstarb, d;
2: goal.Lat ← $C_{\text{array}}$ [0][$W_p$]; {Retrieve latitude of current Waypoint}
3: goal.Long ← $C_{\text{array}}$ [1][$W_p$]; {Retrieve longitude of current Waypoint}
4: goal.heading ←TinyGPSPlus::courseTo( current.Lat, current.Long, goal.Lat, goal.Long); {Calculate desired heading to Waypoint}
5: if current.heading < 0 then
6:     current.heading ← 360 + current.heading;
7: end if
8: $H^{diff}$ ← current.heading – goal.heading; {Calculate heading difference}
9: if $H^{diff} > 180$ then
10:     $H^{diff} \leftarrow H^{diff} - 360$;
11: end if
12: if $H^{diff} < -180$ then
13:     $H^{diff} \leftarrow H^{diff} + 360$;
14: end if
15: if $H^{diff} \geq 0$ then
16:     goalstarb ←$\text{€}_{\text{high}}$; {Set starboard motor throttle to maximum}
17:     goalport ← $\text{€}_{\text{high}} - (\text{sgain}\times( H^{diff}/90) \times (( \text{€}_{\text{high}}) - \eta_{\text{low}})) \text{€}_{\text{high}} - (\text{sgain} \times (H^{diff} / 90) \times ((\text{€}_{\text{high}}) - \eta_{\text{low}})) \text{€}_{\text{high}} - (\text{sgain}\times ( H^{diff}/90) \times (( \text{€}_{\text{high}}) - \eta_{\text{low}}))$; {Calculate port motor throttle}
18: if goalport< $\eta_{\text{low}}$ then
19:     goalport ← $\eta_{\text{low}}$;
```

```
20: end if
21: else
22: goalport ←$€_{high}$;
23: goalstarb ← $€_{high}$ +(sgain× ( $H^{diff}$ /90) × (($€_{high}$)− $\eta_{low}$)) $€_{high}$ + (sgain × ($H^{diff}$ / 90) × (($€_{high}$) − $\eta_{low}$)) $€_{high}$ +(sgain× ( $H^{diff}$ /90) × (( $€_{high}$)− $\eta_{low}$)); {Calculate starboard motor throttle}
24: if goalstarb < $\eta_{low}$ then
25: goalstarb ← $\eta_{low}$;
26: end if
27: end if
28: if | $H^{diff}$ | < 15 then
29: goalport ← $€_{high}$;
30: goalstarb ← $€_{high}$;
31: end if
32: adjust_PWM(goalport,0); {Set port motor to calculated throttle}
33: adjust_PWM(goalstarb,1); {Set starboard motor to calculated throttle}
34: $d$ ←gps.DistanceBetween(current.Lat, current.Long ,goal.Lat ,goal.Long); {Calculate distance to Waypoint}
35: if ($d$<10) and ($W_p$< $N_{coords}$) then
36: $W_p$ ←$W_p$ + 1; {Move to the next waypoint if close enough}
37: end if
```

Algorithm comprehensive description:

- **Initialize**: Sets initial values for variables used in the algorithm.
- **TinyGPSPlus::courseTo**: Calculates the heading (goal.heading) from the current position to the next waypoint (goal.Lat, goal.Long).
- **Adjust Heading**: Ensures current.heading is within the range of 0 to 360 degrees and calculates H^{diff} as the difference between currentheading and goalheading.
- **Motor Throttles**: Adjusts goalport and goalstarb based on the calculated H^{diff}, ensuring motor speeds ($€_{high}$) are adjusted proportionally using sgain but constrained by pwmlow.
- **Pointing-to-Target Check**: Sets both motor throttles to maximum ($€_{high}$) if the heading difference (H^{diff}) is less than 15 degrees.

- **Adjust_PWM**: Sets the PWM (Pulse Width Modulation) denoted as (η) values for motor control based on calculated throttle values (goalport, goalstarb).
- **Distance Check**: Calculates the distance d to the waypoint using GPS coordinates and increments the waypoint index (waypoint) if the vehicle is sufficiently close to the waypoint ($d < 10$ meters) and there are more waypoints ($W_p < \boldsymbol{N}_{\text{coords}}$).

3. Experimental trials

3.1 River Flow Meter

Flow meter shown in figure 6 with its schematic use to track the river current at the specific location that data will helps to analysis AUVs (autonomous surface vehicle) path deviation from the actual path. It is very essential for the AUVs due to change in the current the vehicle can drag form it's predetermine path USVs must navigate in dynamic environments where river currents can be strong and unpredictable. Without prior knowledge of the current flow, the USV's sensor-based navigation becomes more challenging. The interaction between currents and the USV's control system introduces uncertainty in USV's path planning.

River boundaries continuously transform due to tidal patterns and atmospheric forces, making it difficult for autonomous systems to track them accurately.

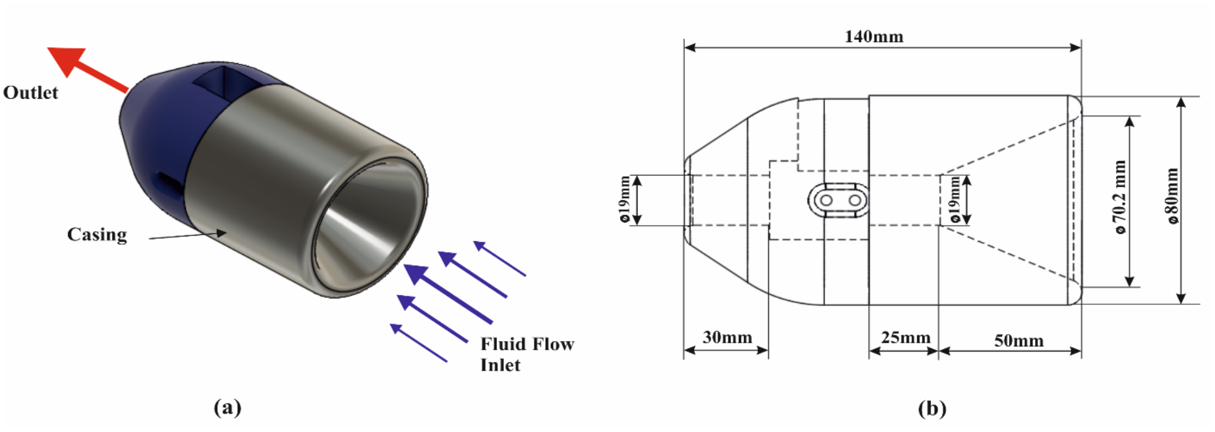


Fig.6 (a) 3D model of flow meter, (b) schematic of flow meter

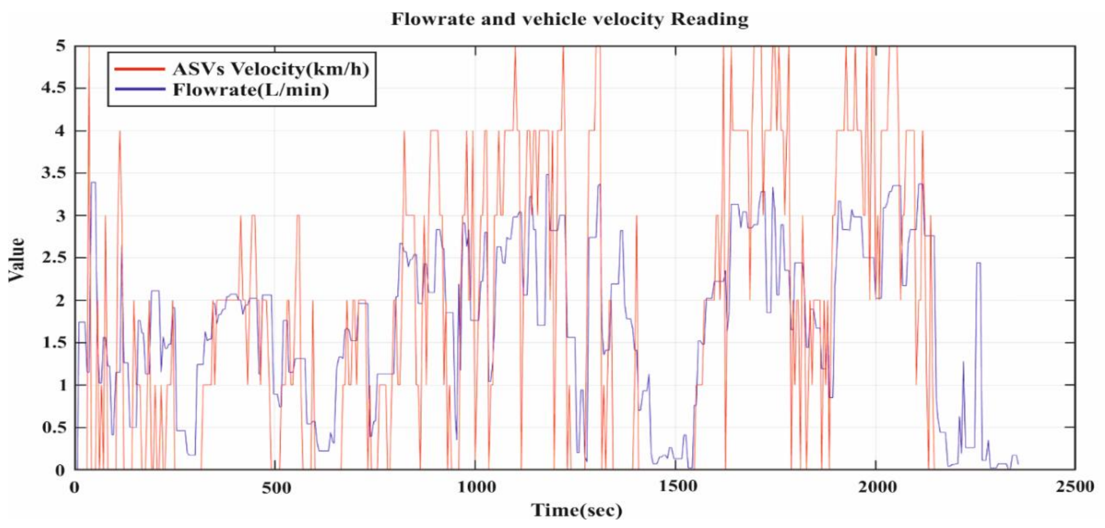


Fig.7 Graph of flowrate versus vehicle velocity

The overlaid line in the image illustrate in figure 7 two key variables over time: boat velocity obtained from GPS (in kilometres per hour, km/h) and flowrate measured by the flowmeter (in litres per minute, L/min) during the trial.

1. **Velocity of the Surface Boat**: The blue line represents the boat's velocity. As time progresses, the graph shows fluctuations in the boat's speed, resulting in peaks and troughs. These variations likely correspond to changes in the boat's movement or external factors influencing its velocity.
2. **Flowrate through the Flowmeter**: The red line represents the flowrate measured by the flowmeter. Similar to the boat's velocity, the flowrate also varies throughout the trial. Peaks and valleys in this graph indicate changes in the flow of water through the flowmeter.

From the graph, we observe a correlation between these two variables. Both velocity and flowrate exhibit similar patterns, suggesting a relationship between them.

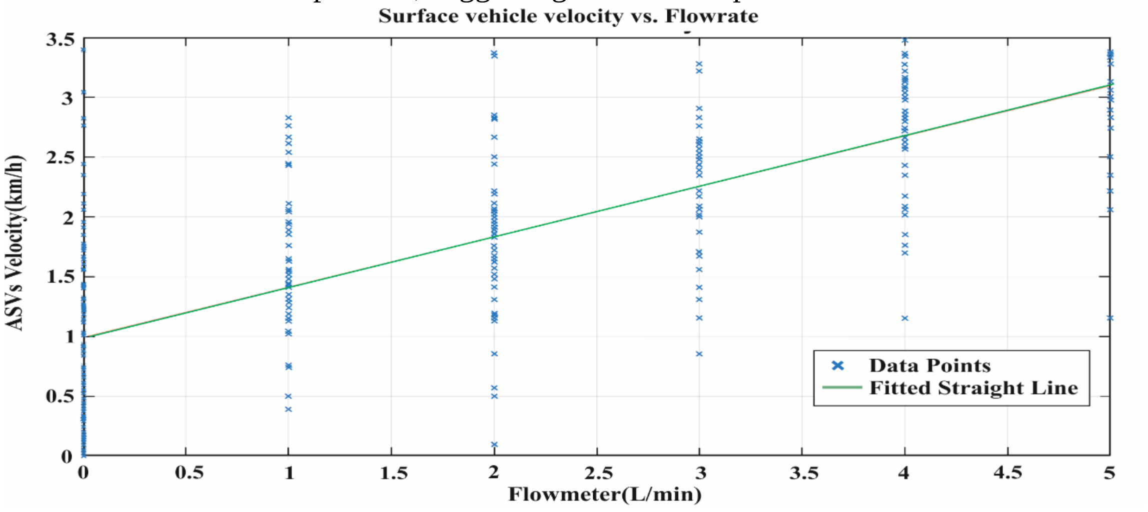


Fig.8 Graph of flowrate versus vehicle velocity GPS correlation

This is a graph between velocity of the surface vehicle and the flowmeter reading. The graph shown in figure 8 represents the relationship between the velocity of a surface boat and the reading from a flowmeter attached to it. The flowmeter measures the velocity of water flow around the boat. As the boat's velocity increases, the flowmeter reading also increases. This relationship allows us to monitor the boat's speed based on the flowmeter data.

Linear Relationship Attempt:

- Additionally, we've attempted to establish a linear relationship between the flowmeter reading and the boat's velocity.
- This involves fitting a straight line (linear regression) to the data points.
- If successful, such a linear model would allow us to predict the boat's speed based on flowmeter readings.

3.2 River mapping

Ping sonar is connected with Arduino Uno and with the code ping sonar start scanning. After acquiring the data is stored into the system. Data consist of distance measure from the river surface level to river bed. Data are as follow

1.1 Ping Sonar

- The topography of the river bed is crucial for study about the underwater environment, such data very helpful in underwater infrastructure. Bathymetric map is necessary for building the bridges, oil and network underwater lines.
- River mapping has being use to acquire the information about the lost ships. It helps scientists study
- Geological processes such as seafloor spreading, subduction, and sedimentation. It also provides insights into ocean circulation patterns, water quality dynamics, and the distribution of marine species. In rivers, bathymetric mapping aids in understanding channel morphology, erosion, and sediment transport processes.

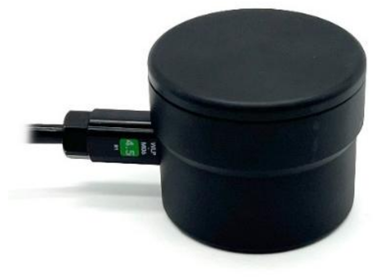


Fig. 9 Ping sonar altimeter and echo sounder

- River mapping is critical for safety of navigation and is used for many other application for example marine and riverine resource management, environmental monitoring.
- In rivers, bathymetric maps assist in maintaining navigable channels, especially in areas prone to sedimentation or changing water levels.
- The vast majority of rivers near the areas where population is living are still virtually unmapped and unexplored.

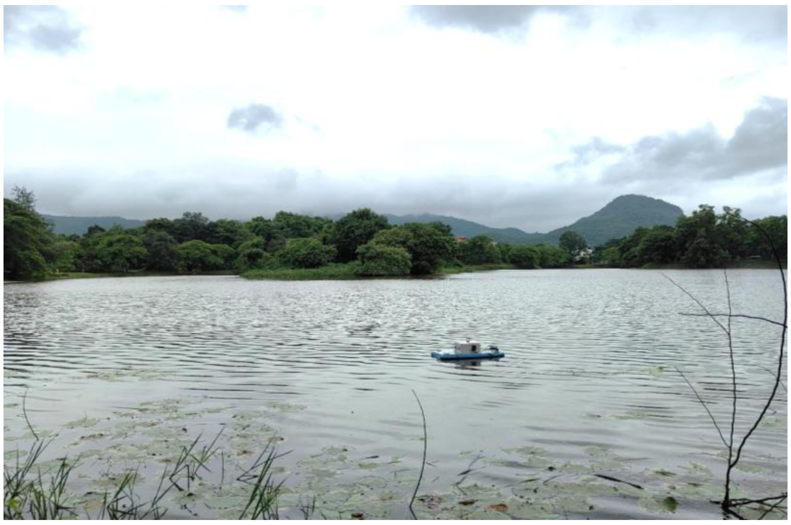


Fig.10 Trail of the Autonomous surface vehicle in the serpentine lack at IITG

The trail has conducted using the ping sonar sensor incorporate on the surface vehicle to perform mapping of the Serpentine Lake over selected area shown in figure 10, duration of the mapping is 150 min approx. Next data has interpolate with the GPS data which are given below. Figure 11 shows the path that was followed to measure the depth data.

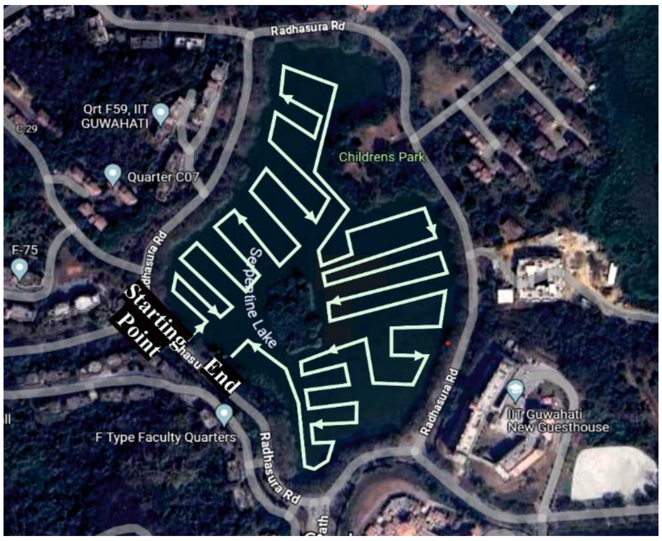


Fig.11 Path followed by the autonomous surface vehicle for mapping

- The surface vehicle followed the predetermined path at constant speed which is 2 knot.
- Maintaining a constant speed ensures that the data collected during the mapping process is consistent. Variations in speed can affect the quality and resolution of the bathymetric data, leading to inaccuracies in depth measurements and spatial information.

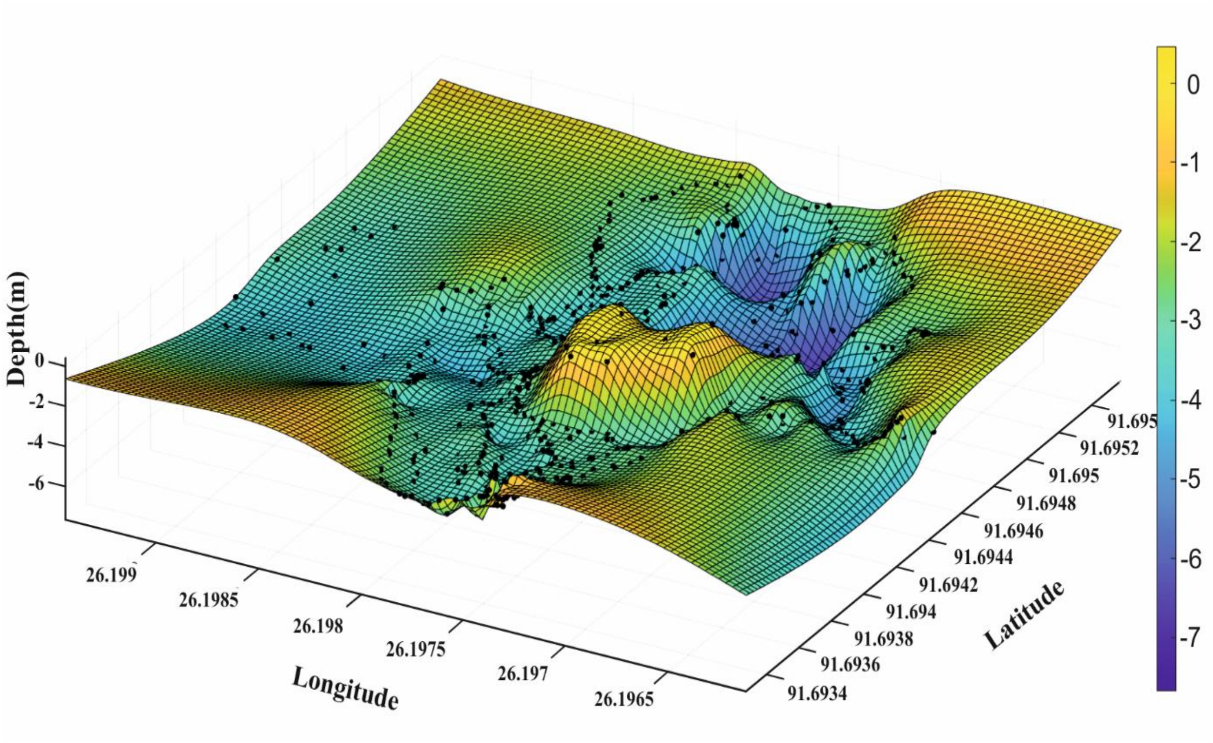


Fig. 12 Bathymetric map of the serpentine lake graph depict Depth verses longitude verses latitude

3.3 Water quality assessment

Assessing the impacts of pollutants, such as plastics, chemicals, and noise, on marine life and their habitats. Detecting and studying the prevalence and spread of diseases among marine organisms.

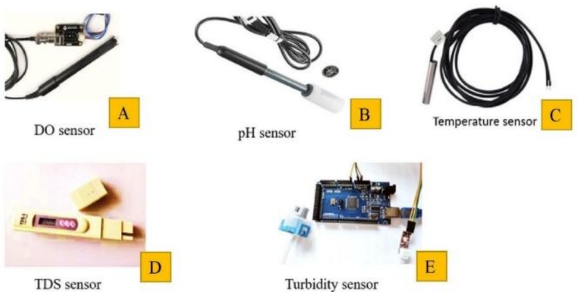


Fig.13 Types of sensor utilize for water assessment

- Quality assessment of contaminated water is essential, particularly in areas close to human habitation.
- Many industries now, dump waste slurries directly into lakes and rivers.
- Measuring and monitoring of quality of water contained in ponds, lakes, and rivers is crucial.
- The essential attributes for assessing the quality of water are pH, turbidity, conductivity, temperature, total dissolved solids (TDS), dissolved oxygen and others.

TDS Sensor (Total Dissolved Solids)

- Monitors water quality by measuring dissolved solids.
- Detects changes in salinity or pollution levels.

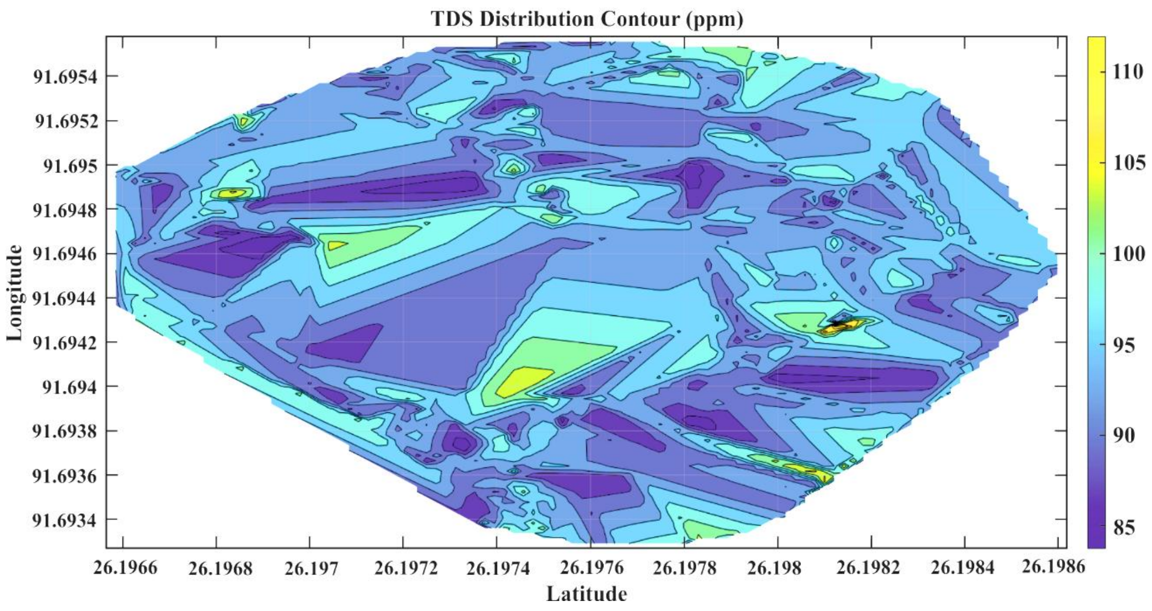


Fig.14 Total dissolve solid at different point on predetermined path of autonomous surface vehicle

Temperature Sensor

- Records water temperature variations.
- Important for environmental studies and habitat assessment.

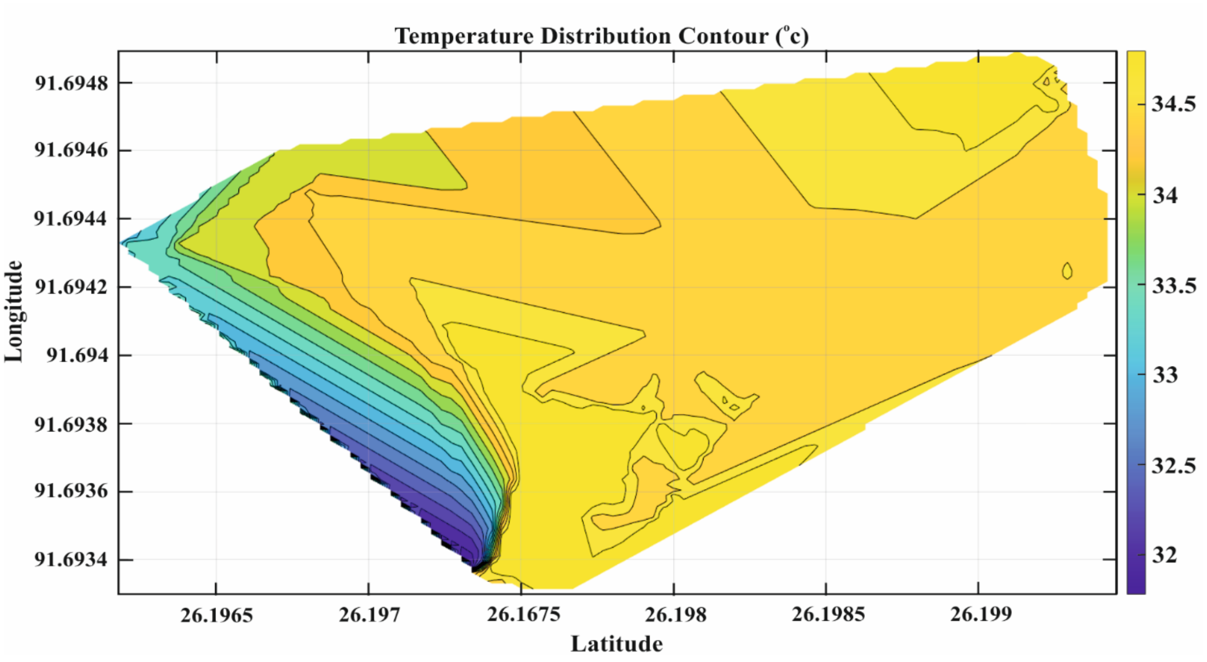


Fig.15 Temperature distribution at the different point at predetermined surface vehicle path

3.4 Obstacle avoidance

USVs need intelligent collision avoidance algorithms that takes into account turbulences and counteract them, to avoid obstacles such as islands, reefs, and other ships.

4. Underwater object detection and tracking

Underwater vehicle has implement cameras with machine learning algorithms for efficient and accurate surveillance. Machine learning algorithm assist to identify the specific special that required to search, therefore objected detection model is incorporated into the processing system of the underwater vehicle. The motive of object detection and tracking is to locate objects of interest within a given frame or series of frames in a video, and subsequently follow their movement across frames. Underwater marine life is very mystical, so it crucial to track and analyse the movement patterns of marine animals to understand their behaviours, migration routes, and habitat preferences for the research purpose.

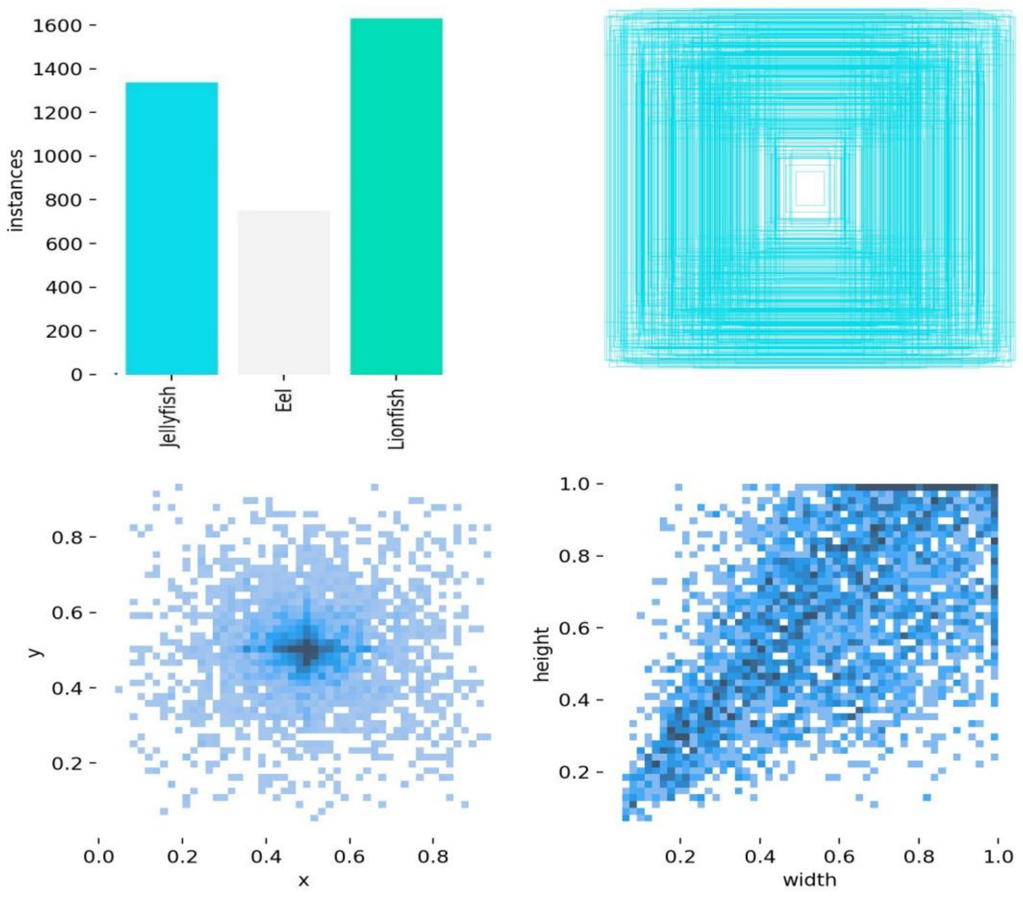


Fig.16 Instances labels for a sample of the dataset

Study interactions between different species, such as predator-prey relationships or symbiotic interactions. The YOLOv8 model with python 3.9 codding, used for training and testing, runs on an 11th Gen Intel Core i7-11700K 3.60GHz processor with a 64-bit operating system. 200 epochs is completed in 126.64 hours using CPU only. In additional, for same number of epoch yolov5 took 48.184 hours to train the model using python – 3.12.2. During the training process initially data set images pre-processed and stretch to 640×640 pixel size.

This model architecture able to achieve 49 FPS (Frame per second) using the Pytorch at the 53.1 mAP@5.0 (mean average precision). YOLOv8 model predict the Class using the bounding box, oriented bounding box prediction gives more precision and confidence over recall. In the oriented bounding box data set the images are rotated at the random angle. Figure

16 illustrates the annotations or ground truth labels for objects of interest in the image. These labels typically represent bounding boxes encompassing different categories of marine fish visible in the sample picture. Each bounding box corresponds to a specific fish class, identifying the type of fish (e.g., jellyfish, eel, and lionfish) present in that area of the image.

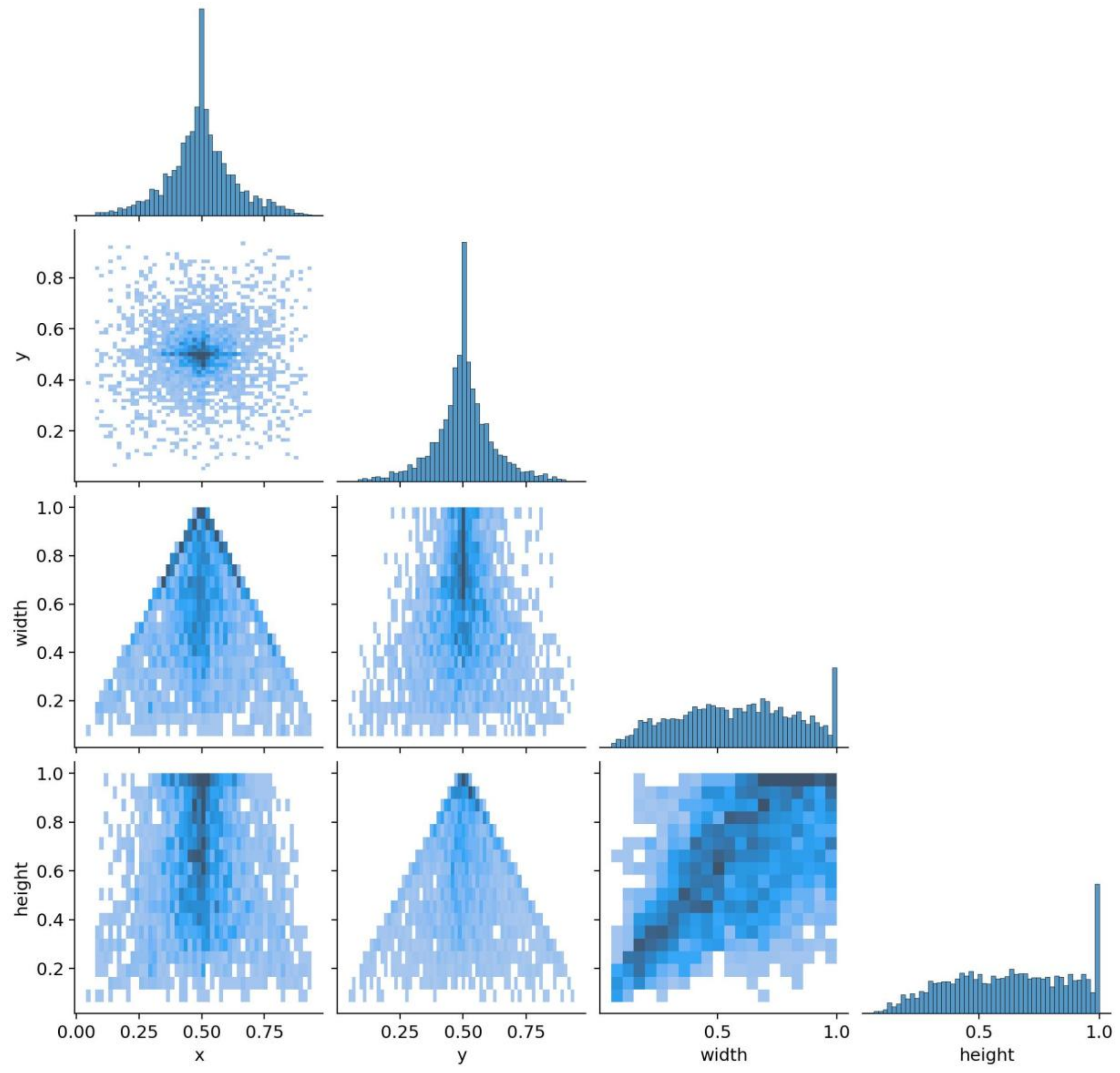


Fig. 17 Correlogram labels for three different classes' sample dataset

A samples from the dataset figure featuring a correlogram typically presents a square matrix. In this matrix, each cell denotes the correlation coefficient between two variables, as illustrated in Fig. 17. The variables (features) of the dataset are listed on both the rows and columns of the matrix.

4.1 Data set

Underwater water vehicle is developed mainly for marine habitat survillance and observe specific species, accordinly the data set is elicted for traning and evauation the model. The data set comprise three classes such as jellyfish, eel, and lionfish. Eels, jellyfish, and lionfish are marine species that divers should be cautious of. Moray eels, known for their sharp teeth and defensive behavior, may bite if provoked or approached too closely. Jellyfish can sting with their tentacles, causing pain and, in severe cases, allergic reactions. Lionfish have venomous spines that can inflict painful stings if touched. In this paper, the total number of annotated images is 4,450. Of these, 3,115 (70%) images constitute the training set, while 889 (20%) and 446 (10%) are allocated to the validation and test sets, respectively.

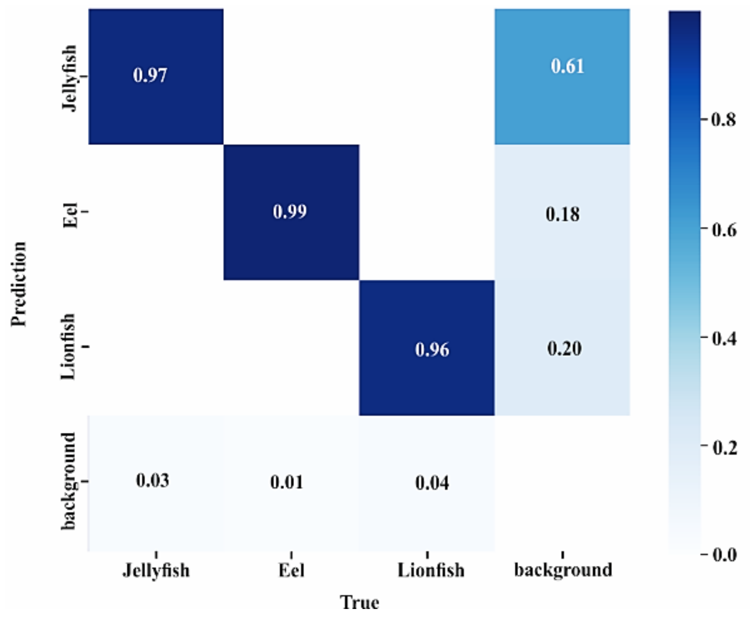
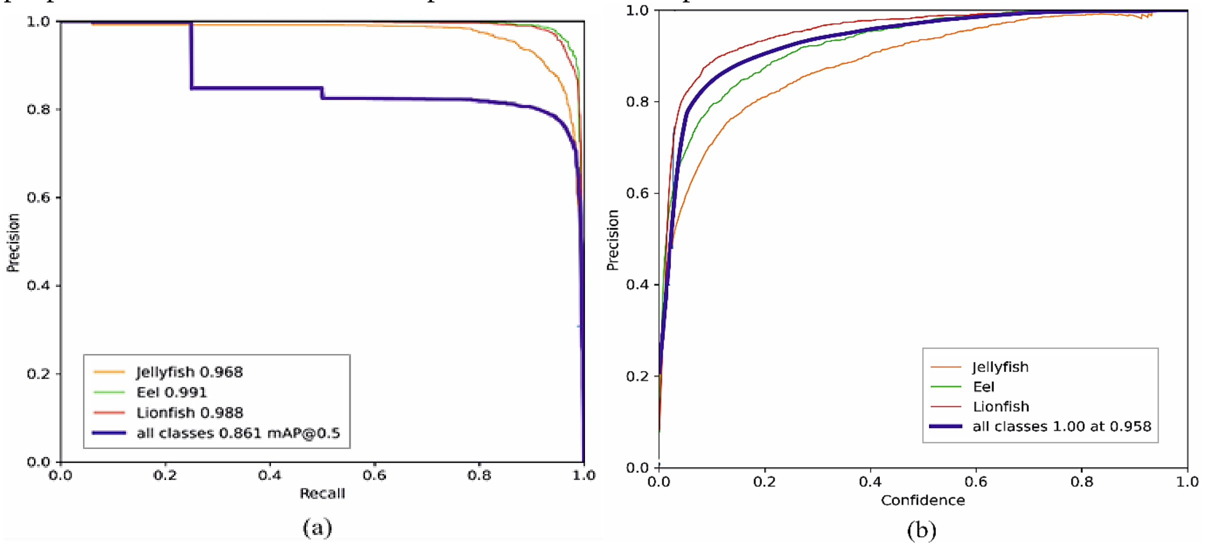
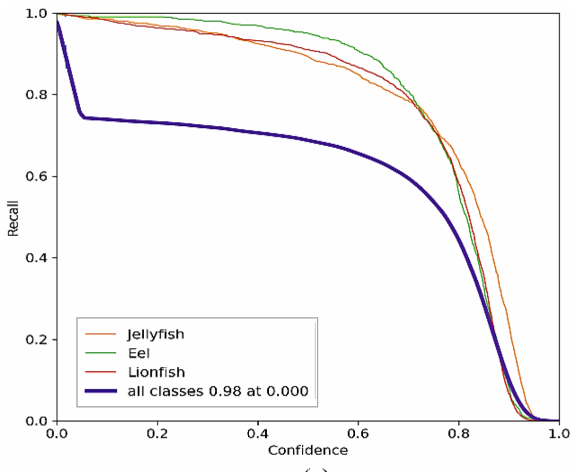


Fig. 18 Normalized confusion matrix diagram

A normalized confusion matrix was constructed to evaluate marine life object recognition, as illustrated in Figure 18. Each row and column in the matrix corresponds to actual and predicted categories, respectively. The diagonal elements represent the accuracy percentages for each category: jellyfish (0.92), eel (0.94), and lionfish (0.96), indicating the proportion of correct predictions compared to their true values.





(c)

Fig.19 YOLOv8 model training results (a) Precision-Recall curve, (b) Precision-Confidence curve (c) Recall-Confidence curve

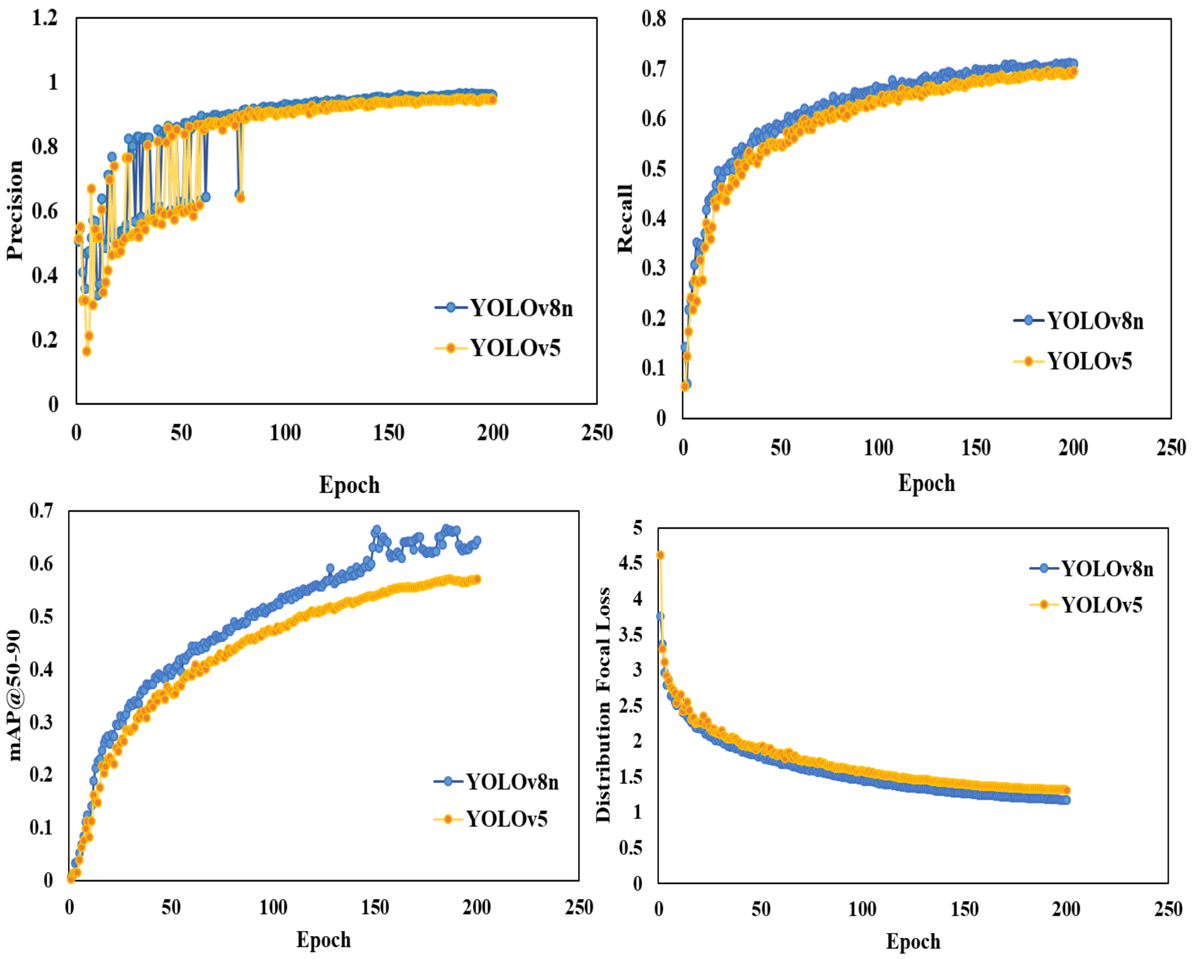
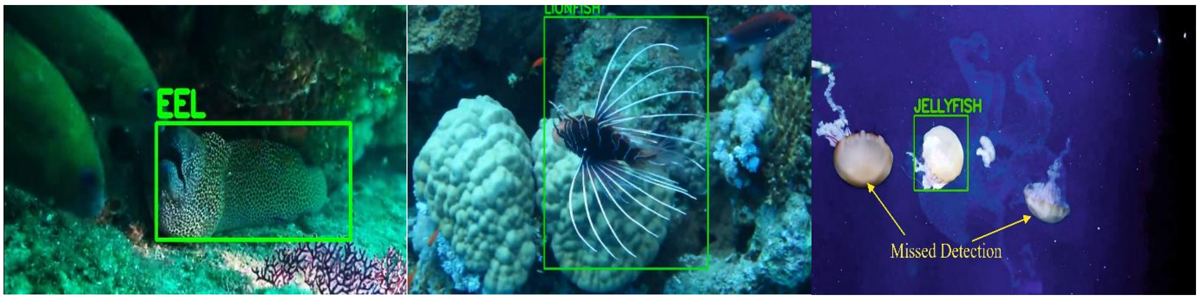
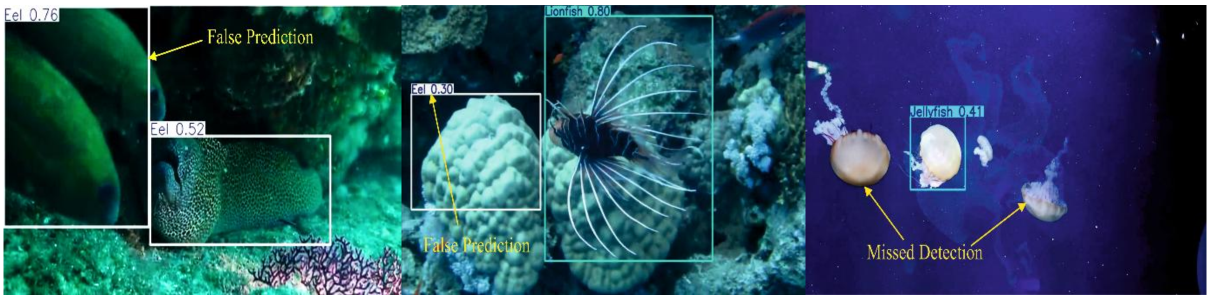


Fig. 20 Net analysis of Precision, Recall, Mean average precision (mAP@50-95), Distribution focal loss (DFL) for YOLOV8 with valid dataset.

The training process incorporates the Learning Rate Decay method with an initial learning rate (lr0) of 0.01 and a coefficient learning rate float (lrf) of 0.01, resulting in a final learning rate that gradually decreases over 200 epochs. Input resolution is set to 640 x 640, with additional parameters including weight decay of 0.0005, and training conducted over 100 epochs with a batch size of 16.



(a) Object detection result of YOLOv8n



(b) Object detection result of YOLOv5

Fig. 21 Comparison of sample detection in between YOLOv8n and YOLOv5 trained model

In Figure 21(b), false predictions and missing detections were observed for lionfish and starfish in many frames per second (FPS) of the video when using the YOLOv5 model. These issues did not occur with the YOLOv8 model, which demonstrated higher precision in detection. However, missed detections for the jellyfish class occurred in both YOLOv5 and YOLOv8 models. This was primarily due to the low resolution, low quantity, and poor quality of the images in the dataset, which affected the models' ability to accurately identify multiple jellyfish. The fact is, when comparing the performance of YOLOv5 and YOLOv8 on the Raspberry Pi 4B, it is important to note that both models have different computational demands and performance characteristics.

YOLOv5 on the Raspberry Pi 4B demonstrates detection with a mean average precision (mAP) of approximately 99.86% and can achieve around 10 to 14 frames per second (FPS) when optimized.

YOLOv8, while offering improved accuracy and advanced features such as multi-scaled object detection and enhanced segmentation capabilities, exhibits slower inference times. Users report an inference time of around 400ms for YOLOv8, which translates to roughly 2.5 FPS at best on this hardware setup

YOLOv5 is expected to run more efficiently due to its lighter architecture, allowing for smoother and faster detections, making it favourable for real-time applications on the Raspberry Pi 4B.

With YOLOv8's demanding architecture and increased complexity, users may face challenges in maintaining real-time performance without optimizations. The model's advanced capabilities would potentially require additional processing power and memory. The lower latency and decent accuracy make it preferable for applications where real-time performance is crucial. While YOLOv8 may excel in scenarios requiring higher accuracy and advanced features, the trade-off in terms of speed and responsiveness would likely be detrimental on the Raspberry Pi’s architecture capable of running YOLOv5 more effectively

Conclusion

References