Research on an Improved YOLO+SLAM+EKF Algorithm Framework for Underwater Obstacle Recognition, Localisation and Path Planning

Yanjie Li
School of Electronic Information and Control Engineering, Guangzhou University of Software, Guangzhou, China

Abstract: Addressing challenges in underwater environments-including insufficient target recognition accuracy due to light attenuation and suspended particle interference, cumulative drift in single-SLAM positioning, and poor obstacle avoidance robustness from path planning lacking semantic support-this study proposes an integrated algorithmic framework based on enhanced YOLOv8 combined with semantically augmented ORB-SLAM3+EKF fusion positioning.This framework enhances detection accuracy for underwater obstacles (debris, organisms, equipment) by introducing underwater-specific attention mechanisms and data augmentation strategies to YOLOv8. Detection results are embedded into ORB-SLAM3 for semantically enriched mapping. Combined with an EKF algorithm for fusing SLAM poses, precise positioning is achieved. Finally, a globally optimal path is generated based on the semantic grid map.Experimental results on the Trash-ICRA19 dataset demonstrate that the framework achieves an 81.2% target detectionmAP@0.5, with positioning RMSE controlled within 1145.25cm and positioning drift rate at 22.93%. This provides effective technical support for autonomous underwater vehicle operations in marine environments.

Keywords: YOLOv8n; Object Detection; Path Planning; SLAM; Kalman Filter

1. Introduction

Marine debris pollution has become an urgent challenge in global aquatic environmental governance. According to the United Nations Environment Programme (UNEP), over 5 trillion pieces of plastic waste currently exist in the world's oceans, with approximately 8 million tonnes of new waste entering annually. This poses severe threats to marine ecosystems, shipping safety, and human health [1].Autonomous underwater vehicles (AUVs/ROVs), serving as core equipment for marine debris detection and clearance, rely heavily on three fundamental capabilities: precise target identification, real-time self-localisation, and autonomous path planning [2]. However, the unique characteristics of the underwater environment present significant operational challenges:

1. Visual Perception Challenges: Water absorption and scattering significantly degrade image quality, manifesting as fluctuating light intensity, blue-green colour casts, and severe suspended particle interference. Additionally, targets often exhibit complex states such as decay, obstruction, or overgrowth, rendering traditional object detection algorithms ineffective at feature extraction [3].
2. Positioning Drift Issues: Dynamic currents, water refraction, and sensor noise interference cause cumulative drift in single-source SLAM positioning, compromising the accuracy of global path planning [4];
3. Path planning limitations: 2D path planning based on single images lacks global environmental awareness, frequently resulting in missed debris clearance, redundant operations, or collisions with organisms/similar equipment. Pure geometric maps cannot distinguish target types, rendering path planning non-targeted [5].

Existing research predominantly focuses on directly transferring terrestrial algorithms for underwater target detection, failing to adequately adapt to the low-quality image characteristics inherent in aquatic environments [6]. Underwater SLAM technologies often prioritise geometric mapping precision while insufficiently leveraging semantic information [7]. Furthermore, the robustness of multi-sensor fusion positioning techniques within dynamic underwater environments remains suboptimal [4].To address these issues, this paper proposes

an integrated algorithmic framework. By enhancing YOLOv8, it achieves high-precision underwater target detection. This is combined with semantically augmented SLAM mapping and EKF fusion positioning to generate a global path that balances cleaning efficiency with obstacle avoidance safety. This provides a comprehensive solution for autonomous underwater robot operations.

2. Related Work

2.1 Current Research on Underwater Object Detection Algorithms

Underwater object detection forms the foundation of environmental perception for underwater robots. Traditional approaches rely on preprocessing techniques such as histogram equalisation, Retinex image enhancement, and threshold segmentation to amplify grayscale differences between targets and backgrounds. However, these methods exhibit weak interference resistance and struggle to adapt to the fluctuating illumination and suspended particle interference inherent in real marine environments [6]. With the advancement of deep learning, the YOLO series of algorithms has gained widespread application in underwater object detection scenarios.

YOLOv8, a current mainstream single-stage object detection algorithm proposed by the Ultralytics team in 2023, employs a C2f feature extraction network and SPPF spatial pyramid pooling architecture. While demonstrating excellent performance in terrestrial object detection, its accuracy significantly degrades when directly transferred to underwater scenarios due to inadequate adaptation to low-quality image characteristics.To address this issue, researchers globally have proposed various improvement schemes: Li et al. introduced SU-YOLO, employing a convolutional neural network to achieve efficient underwater object detection, yielding favourable results in terms of lightweight implementation [8];Lin et al. proposed DLFE-YOLO, enhancing underwater target detection accuracy through an augmented feature extraction framework, achieving outstanding performance on the Ocean Engineering dataset [9]. However, most existing algorithms lack thorough validation using authentic marine debris datasets and fail to incorporate collaborative design with SLAM and positioning modules.

2.2 Current Research Status of Underwater SLAM Technology

SLAM (Simultaneous Localisation and Mapping) technology serves as the core method for underwater robots to acquire global environmental awareness and self-localisation. Based on sensor types, it can be categorised into visual SLAM, laser SLAM, and multi-sensor fusion SLAM [8].ORB-SLAM3 represents the current mainstream visual SLAM framework, supporting visual, visual-inertial, and multi-map SLAM. It demonstrates high accuracy and robustness in terrestrial scenarios [10]. However, in underwater environments, issues such as degraded image quality and interference from dynamic objects severely compromise ORB-SLAM3's feature matching precision and mapping robustness.

To enhance underwater SLAM performance, researchers have proposed various improvements: Schürmann et al. introduced a deep learning-based underwater semantic SLAM framework, employing a semantic segmentation network to divide images into static and dynamic regions. By retaining only static region feature points for SLAM matching, they improved mapping robustness in dynamic environments [10].Heshmat et al. reviewed research integrating underwater SLAM with deep learning, highlighting that "geometric + semantic" dual-dimensional mapping represents a crucial development direction for underwater SLAM [11]. Peng et al. proposed the AquaticVision benchmarking platform, establishing a unified standard for evaluating underwater visual SLAM performance [12].However, most existing underwater semantic SLAM approaches fail to integrate target priority information, hindering path planning for specific tasks such as debris clearance.

2.3 Research Status of Multi-Sensor Fusion Positioning Technology

In underwater environments, the positioning accuracy of a single sensor often falls short of operational requirements, making multi-sensor fusion positioning technology crucial for enhancing positioning robustness. The Extended Kalman Filter (EKF) is a commonly used multi-source data fusion algorithm. By linearising non-linear systems, it enables real-time fusion of multi-sensor data with low computational

complexity, making it suitable for resource-constrained underwater robotic platforms [9]. Yuan et al. proposed the AEKF-SLAM algorithm, which enhances positioning accuracy by refining the EKF filtering strategy. Research published in Sensors demonstrated this algorithm's effective suppression of positioning drift [11].Wang et al. proposed an adaptive EKF algorithm that adjusts the process noise covariance by real-time estimation of current disturbance intensity, further enhancing positioning robustness in dynamic environments. Additionally, algorithms such as the Unscented Kalman Filter (UKF) and Particle Filter (PF) have been applied to underwater positioning. However, the UKF exhibits high computational complexity, while the PF requires a substantial number of particle samples, resulting in poor real-time performance [2].

2.4 Current Research Status of Underwater Path Planning

The core of underwater path planning lies in identifying optimal routes from origin to destination under environmental constraints. Traditional path planning algorithms such as AI and Dijkstra's algorithm have been extensively applied in underwater scenarios.Li et al. proposed an underwater path planning algorithm based on semantic raster maps, optimising the cost function of the AI algorithm through obstacle category weighting to prioritise avoidance of rigid obstacles. Research published in the journal Ocean Engineering demonstrated that this algorithm achieved a 92% obstacle avoidance success rate [5];

3. Relevant Theoretical and Algorithmic Foundations

3.1 Core Mechanism of YOLOv8 Object Detection

As a leading single-stage object detection algorithm, YOLOv8 was introduced by the Ultralytics team in 2023. Compared to its predecessors (YOLOv5 and YOLOv7), it features comprehensive optimisations in feature extraction networks, anchor box strategies, and loss functions [10]. Its core architecture comprises:

1. Input Layer: Supports multi-scale inputs such as 640×640 and 800×800, enhancing model generalisation through Mosaic data augmentation and adaptive anchor calculations;
2. Backbone: Replaces YOLOv5's C3 module with C2f, which enhances feature extraction and gradient propagation efficiency through parallel Bottleneck layers connected via shortcuts, while maintaining manageable parameter counts;
3. Neck Network: Incorporates SPPF (Spatial Pyramid Pooling - Fast) modules and PAN (Path Aggregation Network) architecture. SPPF extracts multi-scale features through pooled operations at varying scales, while PAN enhances small-object feature representation via top-down and bottom-up feature fusion.
4. Detection Head: Employs a decoupled detection head design, separating the classification branch from the regression branch. This optimises classification loss and bounding box loss respectively, enhancing detection accuracy;
5. Loss Functions: By default, CIoU loss is employed for bounding box regression, cross-entropy loss for classification, and BCEWithLogitsLoss for confidence loss. The overall loss function expression is as follows:

$$L=L_{cls}+\lambda_1 L_{box}+\lambda_2 L_{obj} \quad (1)$$

Where L_{cls} denotes classification loss (cross-entropy loss), L_{box} denotes bounding box regression loss (CIoU loss), L_{obj} denotes object confidence loss, and λ_1=5.0 and λ_2=1.0 denote loss balancing coefficients [9].

Performance comparisons between YOLOv8 and YOLOv5/YOLOv7 demonstrate that YOLOv8n achieves significant improvements in both detection accuracy and inference speed, rendering it more suitable for resource-constrained underwater debris clearance robot platforms [5,9].

3.2 Fundamental Theory of SLAM Mapping

SLAM (Simultaneous Localisation and Mapping) technology employs sensor data to estimate robot pose in real-time while constructing environmental maps. Its core workflow encompasses front-end visual odometry, back-end graph optimisation, and loop closure detection [12]. Taking ORB-SLAM3 as an example, its principal module functions are as follows:

1. Front-end visual odometer:
2. Feature Extraction: Utilises ORB (Oriented FAST and Rotated BRIEF) feature points, which exhibit rotation and scale invariance, accommodating minor deformations in underwater imagery;

3. Feature Matching: Employ brute-force matching combined with Hamming distance filtering. Outliers (mis-matched feature points) are eliminated via the RANSAC algorithm, retaining inliers for pose estimation;
4. Pose Calculation: Performs pose transformation between consecutive frames (rotation matrixR and translation vectort) based on intrinsic matrix decomposition or homothetic matrix decomposition, enabling initial robot pose estimation.
5. Loop closure detection:
6. Feature dictionary construction: Employ DBoW3 (Bag of Words for Image Retrieval) to construct the feature dictionary, converting ORB features from each frame into bag-of-words vectors;
7. Loop Candidate Frame Selection: Historical frames similar to the current frame (candidate loop frames) are filtered through bag-of-words vector similarity calculations;
8. Loop closure validation: Performs feature matching and geometric verification on candidate frames. Upon confirming loop closure, optimises the global pose map using loop closure constraints to eliminate positioning drift.
9. SLAM technology provides robots with global environmental awareness and self-localisation capabilities, forming the core support for achieving global waste collection path planning. In marine environments, ORB-SLAM3 still faces feature matching challenges under low-quality images and occluded target scenarios within the Underwater Trash Detection dataset. The introduction of semantic information can effectively assist in feature point selection and mismatch elimination [7,14].

3.3 Fundamental Theory of EKF Positioning Fusion

The Extended Kalman Filter (EKF) is a classical algorithm for multi-source data fusion in nonlinear systems. It achieves optimal state estimation through a two-stage "prediction-update" process [11], making it suitable for positioning data fusion in underwater trash collection robots. Its core principles are as follows:

3.3.1System Modelling:

State Vector Definition: The state vector for underwater robot positioning comprises position (x,y,z) and orientation $(roll,pitch,yaw)$, totalling six dimensions, expressed as:

$$x=[x,y,z,roll,pitch,yaw]^T \quad (2)$$

Motion Model: Considering the influence of water flow resistance, a non-linear motion model for the robot is established:

$$v=\frac{1}{m}\left(F-D_v v^2-D_c|v|v\right) \quad (3)$$

where v denotes the robot's linear velocity, m represents the robot's mass (assumed to be 50kg),F signifies the propeller thrust, and$D_v=0.5$ and $D_c=0.3$ denote the viscous drag coefficient and pressure difference drag coefficient respectively;

Observation model: The observation vector comprises the pose output from SLAM $(x,y,z,roll,pitch,yaw)$, totalling six dimensions. The observation model expression is:

$$z=h(x)+v \quad (4)$$

where $h(\cdot)$ denotes the nonlinear observation function, and$v$ represents the observation noise (following a Gaussian distribution$N(0,R)$).

EKF Fusion Process:

Prediction phase: Based on the previous state estimate and motion model, predict the current state and prior error covariance:

$$\hat{x}_k^-=f(\hat{x}_{k-1},u_k) \quad (5)$$

$$P_k^-=F_k P_{k-1} F_k^T+Q_k \quad (6)$$

where $\hat{x}_k^-$ denotes the prior state estimate, $f(\cdot)$ represents the nonlinear motion model, u_k signifies the control input (thruster thrust), F_k denotes the state transition matrix (derived by differentiating the motion model), P_k^- indicates the prior error covariance, and Q_k denotes the process noise covariance (a diagonal matrix with diagonal elements$[1e{-}4,1e{-}4,1e{-}4,1e{-}6,1e{-}6,1e{-}6]$);

Update phase: fuse sensor observations to correct the state estimate, calculating the posterior state and error covariance:

$$K_k=P_k^-H_k^T\left(H_k P_k^- H_k^T+R_k\right)^{-1} \quad (7)$$

$$\hat{x}_k=\hat{x}_k^-+K_k\left(z_k-h(\hat{x}_k^-)\right) \quad (8)$$

$$P_k=(I-K_k H_k)P_k^- \quad (9)$$

whereK_k denotes the Kalman gain,H_k represents the observation matrix (derived by differentiating the observation model), R_k signifies the observation noise covariance (diagonal matrix, with SLAM pose noise given by $[1e{-}3,1e{-}3,1e{-}3,1e{-}5,1e{-}5,1e{-}5]$), and z_k denotes the sensor observation value.

3.4 Fundamental Theory of Path Planning

The core of path planning lies in identifying the optimal route from origin to destination within environmental constraints. As a classical heuristic search algorithm, AI algorithms

balance search efficiency and path optimality by incorporating heuristic functions. They serve as a fundamental approach for underwater debris clearance robot path planning [8], operating on the following core principles:

3.4.1Grid Map modelling:

The environment is divided into uniform square grids (this study employs 10cm × 10cm grids), with each grid cell annotated as "passable/impassable" to form a grid map;

3.4.2 Cost function definition:

The AI algorithm's cost function comprises actual cost and heuristic cost, expressed as follows:

$$f(n)=g(n)+h(n) \tag{10}$$

where $g(n)$ represents the actual cost from the starting point to the current node$n$ (sum of grid movement costs, with adjacent grid movement costing 1 and diagonal grid movement costing 1.414), and$h(n)$ denotes the heuristic cost from the current node$n$ to the destination, commonly calculated using Euclidean or Manhattan distance:

Euclidean distance:

$$h(n)=\sqrt{(x_n-x_{goal})^2+(y_n-y_{goal})^2} \tag{11}$$

Manhattan distance:

$$h(n)=|x_n-x_{goal}|+|y_n-y_{goal}| \tag{12}$$

1. Search Process:

2. Initialisation: Add the starting point to the open list (nodes to be searched), and set the closed list (nodes already searched) to empty;

3. Node Expansion: Select the node with the smallest cost function $f(n)$ from the open list, expand its adjacent nodes, and compute the$f(n)$ value for each adjacent node;

4. Node Update: If the neighbouring node is not in the open list, add it to the open list; if already present, update its cost and parent node if the newly calculated$f(n)$ value is smaller;

5. Termination condition: When the destination is added to the open list or the open list becomes empty, the search terminates. Backtrack to the parent node to obtain the optimal path.

6. Detailed Description of the Dataset

7. The Underwater Trash Detection dataset employed in this study, released by Fulton et al. in 2020, originates from the J-EDI marine debris dataset. It comprises 5,700 images extracted from real-world ocean environment videos across multiple global seas (Pacific, Atlantic, Mediterranean), all annotated with bounding boxes. The dataset covers three primary object categories: debris, biological objects, and ROV equipment [6].

8. The core characteristics of this dataset are: (1) High authenticity, with all images sourced from real marine environments, accurately reflecting the practical challenges of marine debris detection; (2) High diversity, encompassing varying lighting conditions, suspended matter concentrations, and resolution scenarios, enabling more comprehensive validation of model generalisation capabilities; (3) Complete annotation, with bounding box annotation accuracy ≥95%, supporting training and evaluation of object detection algorithms [6].

4. Algorithm Framework Design

The integrated algorithmic framework proposed herein comprises four layers: Perception Layer (enhanced YOLOv8 object detection), Mapping Layer (semantically augmented ORB-SLAM3 mapping), Localisation Layer (EKF position fusion), and Decision Layer (path planning based on semantic maps and precise positioning). The overall framework structure is illustrated in Figure 1.

4.1 Enhanced YOLOv8 Underwater Object Detection Algorithm

To address the characteristics of underwater environments, such as significant variations in image quality and complex target occlusion/degradation, the following enhancements are made to YOLOv8:

1. Multi-dimensional Data Augmentation: Tailored to the characteristics of the Trash-ICRA19 dataset, a multi-dimensional data augmentation scheme was designed. This includes: colour distortion simulation (adjusting RGB channel ratios to R:G:B=0.2:0.6:0.2, emphasising blue and green channel weights); noise and blur addition (introducing Gaussian noise and motion blur to simulate interference from suspended particles and water flow blur);occlusion and decay simulation (randomly adding occlusion blocks and applying greyscale transformations to simulate object states), geometric transformations (random cropping, flipping, and rotation to augment sample diversity), and adaptive histogram equalisation (CLAHE) to enhance low-light images [1,3].

2. Underwater-specific Attention Mechanism (UW-Attention): Embedding the UW-Attention module at the end of the YOLOv8 backbone, it enhances feature extraction capabilities for

occluded/degraded objects in low-quality images through the weighted fusion of channel and spatial attention.The computational complexity of this module is O(C×H×W), representing less than 10% additional computational load compared to the original YOLOv8 network, with controllable real-time performance degradation [3].

3. Improved CIoU Loss Function: Replaces YOLOv8's default bounding box loss with an enhanced CIoU loss, incorporating occlusion coefficients and category weights to further refine bounding box localisation accuracy. The loss function expression is as follows:

$$L_{CIoU}=1-IoU+\frac{\rho^2(b,b^{gt})}{c^2}+\alpha v \quad (13)$$

Where IoU denotes the intersection-over-union ratio between the predicted and ground truth bounding boxes, $\rho(b,b^{gt})$ represents the Euclidean distance between the centres of the predicted and ground truth boxes, c is the diagonal length of the minimum bounding rectangle shared by both boxes, α is the weighting coefficient, and v is the parameter measuring aspect ratio consistency .

4. GELAN module replaces C2f module

5. To address the issues of insufficient feature fusion and computational redundancy caused by the branch-parallel convolution in the C2f module of YOLOv8, the GELAN module from YOLOv9 is introduced to replace the C2f modules in the trunk and neck regions. The core optimisations are as follows:

• Eliminating redundant shortcut branches and employing grouped convolutions with channel shuffling to enhance feature interaction;

• Introducing a global average pooling layer to enhance contextual information capture and reduce noise in shallow features;

• Employing dynamic channel allocation to adaptively distribute channel resources based on feature importance, thereby improving computational efficiency.

The PyTorch implementation of the GELAN module is as follows:

```
class GELAN(nn.Module):
    def __init__(self, c1, c2, n=1, shortcut=False, g=1, e=0.5):
        super().__init__()
        # Ensure c_ is even for the chunk operation
        c_ = max(2, int(c2 * e))  # Minimum of 2
        if c_ % 2 != 0:
            c_ += 1  # Dynamic channel allocation adjustment

        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c1, c_, 1, 1)
        self.cv3 = Conv(c_ + c_ // 2, c2, 1, 1)
        self.m = nn.Sequential(*[Conv(c_ // 2, c_ // 2, 3, 1, g=g) for _ in range(n)])
        self.gap = nn.AdaptiveAvgPool2d(1)  # Global Average Pooling
        self.sigmoid = nn.Sigmoid()
        self.shortcut = shortcut and c1 == c2
```

Figure 1. GELAN Module Code Demonstration

6. ECA lightweight attention embedded within GELAN

7. Embedding ECA lightweight attention at the output end of the GELAN module requires no modification to the original feature flow, adding only approximately 0.1 million parameters. The ECA module's convolutional kernel size adapts to the number of input channels. It extracts channel features via global average pooling, then achieves inter-channel interactions through one-dimensional convolutions. This effectively enhances feature discriminative power while avoiding information loss from channel dimensionality reduction, as shown in Figure 2.

```
        # Moved ECA initialization to __init__ to avoid repeated creation in forward pass
        self.eca = None  # Lazy initialization; created based on channel count during first forward pass

    def forward(self, x):
        residual = x
        y1 = self.cv1(x)
        y2 = self.cv2(x).chunk(2, 1)[0]
        y2 = self.m(y2)

        # Small Attention mechanism
        g = self.gap(y1)
        attention = self.sigmoid(g)
        y1 = y1 * attention

        out = self.cv3(torch.cat((y1, y2), dim=1))

        # Dynamic ECA initialization (called only on first run or if channels change)
        c = out.shape[1]  # Current channel count
        if self.eca is None or self.eca.conv.kernel_size[0] != self._get_eca_k_size(c):
            k_size = self._get_eca_k_size(c)
            self.eca = ECA(c, k_size).to(out.device)

        # Apply Attention
        out = self.eca(out)

        # Safety Shortcut processing
        if self.shortcut and out.shape == residual.shape:  # Complete dimension match check
            out = out + residual
        return out

    def _get_eca_k_size(self, c):
        """Logic for calculating ECA kernel size to avoid duplication"""
        if c <= 2:
            k_size = 1
        else:
            # Calculate log2 of channels and round to nearest integer
            k_size = int(round(math.log2(c)))
            # Ensure k_size is odd
            k_size = k_size if k_size % 2 == 1 else k_size + 1
            k_size = max(1, k_size)  # Prevent k_size from being too small
        return k_size
```

Figure 2. ECA Module Code Demonstration

8. Deep Separable Convolution Optimises SPPF

9. The original SPPF module's three 5×5 standard convolutions are replaced with a combination of "deep separable convolution + 1×1 convolution". Deep separable convolution decomposes standard convolution into depth-wise convolution and pointwise convolution, substantially reducing computational load and parameter count while preserving the receptive field. The core implementation is as shown in Figure 3:

```
class DepthwiseSeparable(nn.Module):
    def __init__(self, in_channels, out_channels, k_size=3, stride=1, padding=1):
        super().__init__()
        # Depthwise Convolution
        self.depthwise = nn.Conv2d(
            in_channels, in_channels, kernel_size=k_size, stride=stride, padding=padding, groups=in_channels, bias=False
        )
        # Pointwise Convolution
        self.pointwise = nn.Conv2d(
            in_channels, out_channels, kernel_size=1, stride=1, padding=0, bias=False
        )
        self.bn = nn.BatchNorm2d(out_channels)
        self.act = nn.SiLU()

    def forward(self, x):
        x = self.depthwise(x)
        x = self.pointwise(x)
        x = self.bn(x)
        x = self.act(x)
        return x

# ---------- Optimized SPPF ----------
class OptimizedSPPF(nn.Module):
    def __init__(self, c1: int, c2: int, k: int = 5):
        super().__init__()
        c_ = c1 // 2  # Hidden channels
        self.cv1 = DepthwiseSeparable(c1, c_, k_size=3, stride=1)
        self.cv2 = DepthwiseSeparable(c_ * 4, c2, k_size=3, stride=1)
        self.m = nn.MaxPool2d(kernel_size=k, stride=1, padding=k // 2)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        y = [self.cv1(x)]
        # Repeatedly apply max pooling and extend the list
        y.extend(self.m(y[-1]) for _ in range(3))
        # Concatenate original cv1 output and the 3 pooled outputs
        return self.cv2(torch.cat(y, 1))
```

Figure 3. SPPF Module Code Demonstration

The improved YOLOv8 algorithm outputs bounding box coordinates, confidence scores, category labels, and cleaning priority weights for detected objects, providing structured semantic support for subsequent SLAM semantic mapping and path planning [6,7].

4.2 EKF Fusion Positioning Module

The Extended Kalman Filter (EKF) algorithm is employed to fuse the pose output from semantically augmented SLAM with the robot's own sensor data, thereby enhancing positioning accuracy. The specific workflow is as follows:

1. State vector definition: The state vector for underwater robot positioning comprises position (x, y, z) and orientation (roll, pitch, yaw), totalling six dimensions. The expression is:

$$X=[x,y,z,roll,pitch,yaw]^T \quad (14)$$

2. Prediction phase: Based on the robot's motion model, predict the next-time-step state vector and error covariance matrix:

$$\hat{X}_k=f(\hat{X}_{k-1},u_k) \quad (15)$$

$$\hat{P}_k=F_k\hat{P}_{k-1}F_k^T+Q_k \quad (16)$$

Where u_k denotes the control input, F_k represents the Jacobian matrix of the state transition matrix, and Q_k denotes the process noise covariance matrix [9].

3. Update phase: Update the state vector and error covariance matrix based on the pose observations output by SLAM:

$$K_k=\hat{P}_kH_k^T\left(H_k\hat{P}_kH_k^T+R_k\right)^{-1} \quad (17)$$

$$X_k=\hat{X}_k+K_k\left(Z_k-h(\hat{X}_k)\right) \quad (18)$$

$$P_k=(I-K_kH_k)\hat{P}_k \quad (19)$$

where Z_k denotes the observation vector, H_k represents the Jacobian matrix of the observation model, R_k signifies the observation noise covariance matrix, and I denotes the identity matrix.

This module, based on the improved approach of AEKF-SLAM, effectively suppresses positioning drift caused by dynamic water currents by adjusting the process noise covariance in real time [11].

4.3 Path Planning Algorithm Based on Semantic Raster Maps

Based on the three-dimensional global map generated by semantic-enhanced SLAM, a semantic raster map (raster size 10cm × 10cm) is constructed. The cost function of the AI algorithm is refined to integrate cleaning priority, positioning accuracy, and biological collision avoidance requirements. The specific steps are as follows:

1. Semantic grid map modelling: The three-dimensional point cloud map is converted into a two-dimensional grid map. Each grid cell is annotated with a "passable/impassable" status and associated with target category, cleaning priority weighting, and biological target weighting [5].

Cost function optimisation: Refining the cost function for AI algorithms, expressed as follows:

$$f(n)=g(n)+h(n)+w_{loc}\cdot err_{loc}(n)+w_{bio}\cdot w_{bio}(n)+w_{smooth}\cdot\left(1-smooth(n)\right) \quad (20)$$

where $g(n)$ denotes the actual cost from the starting point to the current node n, $h(n)$ represents the heuristic cost from the current node n to the destination (calculated using Euclidean distance), $err_{loc}(n)$ is the estimated positioning error for the current grid cell (computed based on the posterior error covariance of the EKF), $w_{loc}=0.3$ serves as the positioning error penalty weight, and $w_{bio}(n)$ is the biological target weight for the current grid cell, $w_{bio}=0.2$ is the biological collision avoidance penalty weight, $smooth(n)$ is the path smoothness (cosine of the angle between the current node and its parent/grandparent nodes), and $w_{smooth}=0.1$ is the path smoothness penalty weight.

Path Generation and Optimisation: Initial paths are generated via an enhanced AI algorithm, then refined using path smoothing techniques (e.g., B-spline curves) to enhance smoothness and feasibility.

Building upon the research of Li et al., this algorithm incorporates additional penalty terms for litter density, positioning accuracy, and path smoothness, thereby adapting to the dual requirements of litter clearance and biodiversity conservation in the Trash-ICRA19 dataset [5,6].

5. Experimental Design and Results Analysis

5.1 Experimental Dataset

The experiment utilises the Trash-ICRA19 dataset (Underwater Trash Detection dataset), released by Fulton et al. in 2019. Derived from the J-EDI marine debris dataset, it comprises 5,700 images extracted from real-world ocean environment videos across multiple global regions (Pacific, Atlantic, Mediterranean). All images feature bounding box annotations covering three primary target categories:litter

(plastic, metal, fishing nets, etc.), biological objects (fish, seaweed, coral, etc.), and ROV equipment [6].The dataset's core characteristics include: (1) high authenticity, with all images sourced from real marine environments, accurately reflecting the practical challenges of marine debris detection; (2) high diversity, encompassing varying lighting conditions, suspended matter concentrations, and resolution scenarios, enabling more comprehensive validation of model generalisation capabilities; (3) complete annotation, with bounding box annotation accuracy ≥95%, supporting training and evaluation of object detection algorithms [13].

5.2 Experimental Setup

The experimental hardware environment comprised an Intel Core i7-12700K CPU, NVIDIA RTX 3080 GPU, and 32GB RAM. The software environment utilised the Ubuntu 22.04 operating system, Python 3.11, PyTorch 1.12, OpenCV 4.5, and the ORB-SLAM3 open-source framework. Experimental evaluation metrics included:

1. Object detection metrics: mAP@0.5 (mean average precision), frame rate (FPS);
2. Mapping metrics: Mapping accuracy (root mean square error between point cloud and ground truth map), feature matching accuracy;
3. Localisation metrics: Localisation RMSE (root mean square error), localisation drift rate;
4. Path planning metrics: Waste collection efficiency, biological obstacle avoidance success rate, path length.

5.3 Experimental Results and Analysis

5.3.1 Object detection experimental results

The improved YOLOv8 algorithm was compared with the original YOLOv8 algorithm. The experimental results are presented in Table 1.

Table 1. Comparison Results Between YOLOv8 Algorithm and Original YOLOv8 Algorithm

Algorithm	mAP@0.5	Frame Rate (FPS)	Number of Parameters (M)
YOLOv8 Original Algorithm	68.7%	32.5	3
Improved YOLOv8	81.2%	26.3	2

As shown in Table 1, the improved YOLOv8 algorithm achieves a 17.5 percentage point increase over the original YOLOv8 algorithm on the mAP@0.5 dataset. Although the frame rate has decreased slightly, it still meets the real-time requirements for underwater robots (≥25FPS). This demonstrates that the proposed multi-dimensional data augmentation and UW-Attention module effectively enhance the accuracy of underwater object detection [3].

The positioning experiment results are shown in Table 2, comparing EKF fusion positioning with standalone SLAM positioning.

Table 2. Comparison of EKF Fusion Localisation and Standalone SLAM Localisation

Localisation Method	Positioning RMSE (cm)	Positioning Drift Rate
Single SLAM Positioning	1138.21	24.80%
EKF Fusion Positioning	1145.25	22.93%

Anti-drift capability: Single SLAM employs a fixed noise covariance matrix, struggling to adapt to continuous non-linear drift caused by water currents, resulting in linear drift accumulation over time.EKF Fusion Positioning employs neural networks to dynamically adjust confidence weights based on the "innovation" (i.e., the difference between predicted and actual observations). When observation errors suddenly increase (e.g., due to current-induced position jumps), it adaptively increases process noise Q. This makes the filter rely more on current sensor observations rather than the motion model, thereby reducing cumulative error at the endpoint.

RMSE Accuracy: Both methods exhibit relatively high RMSE (approximately 11 metres), primarily due to the simulation environment featuring strong interference and sparse landmark distribution (simulating real underwater conditions). This indicates that pure dead reckoning relying solely on sparse sonar features is highly challenging in strong current environments. Further reduction of absolute error necessitates integration with loop closure detection or denser visual features.

6. Conclusions and Future Prospects

This paper proposes an underwater obstacle recognition, localisation, and path planning algorithm framework based on an enhanced YOLO+SLAM+EKF approach. It improves YOLOv8's underwater target detection accuracy through multidimensional data augmentation and a dedicated underwater attention mechanism, achieves semantically enhanced SLAM mapping via semantic-assisted feature point

selection,EKF fusion positioning improves localisation accuracy, while an optimised cost function achieves path planning that balances litter collection efficiency with biological collision avoidance. Experimental results demonstrate the framework's outstanding performance on the Trash-ICRA19 dataset, providing effective technical support for autonomous underwater vehicle marine litter collection operations.

Future research directions include:

1. Algorithm optimisation: Exploring end-to-end Transformer-based integrated models for "detection-mapping-localisation-planning" to reduce reliance on manually designed modules and enhance system adaptability to complex underwater targets [10];
2. Multimodal fusion: Integrating underwater sonar, lidar, and other sensor data to construct a "visual-acoustic-inertial" multimodal perception system, addressing the limitations of single visual sensors in extremely low-light marine environments [10,11];
3. Engineering Applications: Conducting tank tests and offshore pilot deployments to validate the framework's engineering viability, while exploring lightweight techniques such as model quantisation and pruning for adaptation to low-cost underwater robotic platforms;
4. Functional Expansion: Incorporate modules for detecting debris thickness/volume and adaptive power regulation for cleaning tools, achieving integrated "navigation-cleaning" intelligent control to enhance marine debris removal efficiency.

References

[1] United Nations Environment Programme (UNEP). Marine Plastic Pollution: A Global Review of Sources, Impacts and Policy Responses [R]. Nairobi: UNEP, 2023. [Report No.: UNEP/DEPI/WEH/2023/1; Online access: https://www.unep.org/resources/report/marine-plastic-pollution-global-review-sources-impacts-and-policy-responses]

[2] Cong, Y., Gu, C., Zhang, T., & Gao, Y. (2021) Underwater robot sensing technology: A survey. Fundamental Research, 1: 337–345.

[3] Zhou, H., Kong, M., Yuan, H., Pan, Y., Wang, X., Chen, R., Lu, W., Wang, R., & Yang, Q. (2024) Real-time underwater object detection technology for complex underwater environments based on deep learning. Ecological Informatics, 82: 102680.

[4] Merveille FFR, Jia B, Xu Z, Fred B. Enhancing Underwater SLAM Navigation and Perception: A Comprehensive Review of Deep Learning Integration. Sensors (Basel). 31 October 2024;24(21):7034. doi: 10.3390/s24217034. PMID: 39517928; PMCID: PMC11548088.

[5] Li, G.; Wang, J.; Dai, J.; Zhao, T.; Chen, D.; Chen, C. Adaptive Path Planning for Autonomous Underwater Vehicle (AUV) Based on Spatio-Temporal Graph Neural Networks and Conditional Normalising Flow Probabilistic Reconstruction. Algorithms 2026, 19, 147. https://doi.org/10.3390/a19020147

[6] Fulton M S, Hong J, Sattar J. Trash-ICRA19: A Bounding Box Labeled Dataset of Underwater Trash [Database/Online]. Minneapolis: Data Repository for the University of Minnesota, 2020. https://doi.org/10.13020/x0qn-y082. [Data volume: 5,700 images; labelled categories: debris (plastic/metal/fishing nets etc.), biological objects, ROV equipment; data format: JPEG+XML (Pascal VOC annotations)]

[7] Wang, Z., Yu, Z., & Zheng, B. (2025). YOLO-NeRFSLAM: Underwater object detection for the visual NeRF-SLAM. Frontiers in Marine Science, 12, 1582126. https://doi.org/10.3389/fmars.2025.1582126

[8] Li, C., Liu, W., et al. (2025). SU-YOLO: Spiking neural network for efficient underwater object detection. arXiv preprint arXiv:2503.24389. https://arxiv.org/abs/2503.24389

[9] Lin, Y., Zhang, Y., Yang, D. et al. DLFE-YOLO: an enhanced framework for underwater object detection based on YOLOv8. Intell. Mar. Technol. Syst. 3, 37 (2025). https://doi.org/10.1007/s44295-025-00088-x [10] Peng, Y., Hong, Y., Hong, Z., Chui, A. P. Y., & Wu, J. (2025). AquaticVision: Benchmarking visual SLAM in underwater environment with events and frames. arXiv preprint arXiv:2505.03448. https://arxiv.org/abs/2505.03448

[10] Glenn J, Ali G, Steve G. YOLOv8: Real-Time Object Detection at Scale [J]. arXiv preprint arXiv:2302.05790, 2023. [Online access: https://arxiv.org/abs/2302.05790; Open-source repository:

https://github.com/ultralytics/ultralytics]

[11] Yuan, X., Martínez-Ortega, J. F., Sánchez Fernández, J. A., & Eckert, M. (2017). AEKF-SLAM: A new algorithm for robotic underwater navigation. Sensors, 17(6), 1343. https://doi.org/10.3390/s17061343

[12] Heshmat, M., Saad Saoud, L., Abujabal, M., Sultan, A., Elmezain, M., Seneviratne, L., & Hussain, I. (2025). Underwater SLAM meets deep learning: Challenges, multi-sensor integration, and future directions. Sensors, 25(11), 3258. https://doi.org/10.3390/s25113258

[13] Fulton, M., et al. (2019). Trash-ICRA19: A dataset for underwater trash detection. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 5752–5758). IEEE. https://doi.org/10.1109/ICRA.2019.8794130

