Vision-Based Autonomous Navigation

1. Project Overview

Competition: Singapore Autonomous Underwater Vehicle Challenge 2026 (SAUVC 2026)
Objective: To design and build a fully Autonomous Underwater Vehicle (AUV) capable of efficiently performing underwater tasks, starting with Gate Navigation and Flare Avoidance.
Core Approach: Utilizing advanced Computer Vision (CV) techniques, including Machine Learning (ML) and traditional image processing, for accurate target detection and autonomous navigation in the challenging underwater environment.

2. Technical Approach for Initial Stage (Navigation Task)

The initial stage requires the AUV to successfully pass through a gate and avoid a critical orange flare without touching it. We propose an integrated vision system combining robust ML for primary object identification and fast, reliable traditional CV for confirmation and avoidance.

2.1. ML-Based Object Detection (YOLO)

Application: Primary detection and localization of key static targets like the Gate and Drums.

- **Model:** A lightweight, highly efficient model such as **YOLO (You Only Look Once)** or EfficientDet will be employed to ensure fast inference speeds on the embedded system.
- **Training Data:** The model will be trained using a dataset comprising simulated underwater images, augmented with effects like blur, distortion, and varied lighting, alongside images from previous competition rulebooks.
- **Output:** The model provides a bounding box and the precise center point coordinates (C_G) of the detected Gate.
- **Significance:** ML models provide high resilience against poor lighting, turbidity, and partial occlusions common in underwater settings.

2.2. Traditional Color Detection (HSV)

Application: Rapid and precise detection and avoidance of the mandatory Orange Flare, and verification of the Gate's markings (Red/Green stripes).

- **Technology:** The **HSV (Hue, Saturation, Value) color space** is superior to RGB for underwater processing as it separates intensity (Value/Lightness) from color (Hue), making it more stable under changing illumination.
- **Process:**
    1. The live camera feed is converted to HSV.
    2. Specific HSV ranges are defined for the target colors (e.g., Orange for the flare).
    3. A binary mask is created, highlighting only the pixels within the target range.
    4. Contour analysis is performed on the mask to determine the flare's center point (C_F).

- **Avoidance Mechanism:** If the center point C_F of the Orange Flare is detected within a high-risk zone relative to the AUV's path, the navigation logic immediately initiates an evasive maneuver (lateral shift) to prevent contact.

3. Hardware and Software Stack

3.1. Hardware Components

Component	Description	Purpose
On-board Computer	NVIDIA Jetson Nano/Xavier NX (Preferred) or Raspberry Pi 5	High-performance execution of ML models and autonomous decision-making.
Underwater Camera	High-resolution, low-light USB or CSI camera	Acquiring video feed for target detection and environment monitoring.
Thrusters	6 or 8 Thruster configuration	Providing precise control over movement in X, Y, Z axes, and rotation (Yaw).
Depth Sensor	Pressure Sensor (e.g., MS5837)	Maintaining stable depth for consistent image acquisition and navigation.
Watertight Hull	Custom or Commercial Pressure Vessel	Protecting all electronic components from the water environment.

3.2. Software Tech Stack

Category	Technology	Role
Operating System (OS)	Ubuntu/Linux	Foundational OS for robotics development.
Programming Language	Python	Primary language for CV, ML integration, and control logic.
Computer Vision (CV)	OpenCV	Image preprocessing, traditional color masking, and feature extraction.
ML Framework	TensorFlow Lite / PyTorch	Running the pre-trained YOLO/EfficientDet model efficiently.
Robotics Framework	ROS (Robot Operating System) - Optional but Recommended	Integration and modular communication between sensors, control, and vision modules.

4. Implementation Flow

Phase 1: Initialization and Depth Control

1. The AUV stabilizes at a pre-defined depth (e.g., 50cm).
2. The vision system starts processing the camera feed.

Phase 2: Gate and Flare Detection

1. **ML:** The YOLO model actively searches for the Gate. If detected, its central coordinates C_G are recorded.
2. **CV:** HSV masking is run simultaneously to detect the precise location and extent of the Orange Flare. The flare's center point C_F is recorded.

Phase 3: Decision Logic

Condition	AUV Action
Gate Detected (\text{YOLO } > \text{Threshold}) AND No High-Risk Flare Detected	Begin maneuvering towards the Gate's center point C_G.
Gate Detected, BUT Orange Flare C_F is directly in the path/high-risk zone	Initiate a lateral avoidance maneuver away from C_F (e.g., shift 1m to the left/right), then re-engage navigation to C_G.
Gate Not Detected	Initiate a slow, controlled search pattern (e.g., slow panning motion) until the target is acquired.

Phase 4: Navigation and Entry

1. Error Calculation: Calculate the positional error (distance in pixels/meters) between the AUV's current center and the Gate's center C_G.
2. PID Control: Use a **PID controller** based on the error to command the thrusters, ensuring the AUV is centered before moving forward to cross the Gate.
3. After passing the Gate, the AUV transitions to the search pattern for the next stage's objective.

5. Conceptual Python Pseudocode

The following conceptual code demonstrates the core logic for processing the camera input and executing control commands based on the detection results.