Adaptive Image Restoration and Object Detection and Tracking using Semantic Segmentation and Cross-Correlation based trackers for light-weight embedded devices

Shubh Jain[1] Dipak Kumar Giri[2]

***Abstract*— We introduce an innovative strategy for object detection and tracking based on semantic segmentation masks for detection and cross-correlation trackers specifically designed for light weight embedded-devices, enabling real-time operations even in environments with limited resources, including open-source robotics initiatives. An adaptive image enhancement algorithm has also been developed to tackle the dynamically changing underwater conditions. The outcomes highlight the efficacy of our tracking method, allowing Autonomous Underwater Vehicles (AUVs) to pursue objects swiftly on light-weight embedded devices, achieving comparatively high frame rates.**

***Index terms*- AUV/ROV , Underwater Vehicle, Image enhancement, Object detection, Object Tracking.**

I. INTRODUCTION

The ocean is crucial for modern development, with the exploration and utilization of marine resources closely linked to technological advancements [1]. In this context, underwater imaging plays a vital role, particularly for navigation tasks that require accurate and reliable visual information. Key underwater tasks such as Autonomous Underwater Vehicle (AUV) operations, marine exploration [7], environmental monitoring [6], [8], underwater archaeology [9], offshore oil and petroleum exploration [10], oceanographic research [11], and defense operations [12] depend heavily on high-quality image capture [3]. Accurate visual data is essential for effective obstacle avoidance, path planning, and data collection. However, the unique properties of seawater pose significant challenges to obtaining clear images. Issues such as blurred details, low contrast, color distortion, poor clarity, and uneven lighting impede efficient marine exploration and navigation [2].

Overcoming these challenges requires innovative image processing and object detection techniques that adapt to the variable and often challenging underwater conditions. Although recent advancements in underwater image enhancement and traditional object detection methods frequently utilize computationally demanding deep learning frameworks [4] [5], these approaches necessitate substantial hardware resources. They can lead to problems such as overheating, especially in resource-limited underwater environments. Furthermore, the need for cost-effective, open-source solutions in underwater research adds another layer of complexity to the deployment of advanced image processing and detection technologies. Our proposed methodology aims to address these issues by focusing on efficiency while maintaining accuracy, thereby minimizing computational demands and mitigating overheating and power consumption concerns. By integrating image processing with deep learning, this study seeks to establish a robust framework for object detection and tracking on lightweight embedded devices.

Through Adaptive Image Enhancement Techniques and an integrated analysis of division-sized algorithms and various deep learning models, this study aims to optimize object detection & Tracking performance for low-end, lightweight devices, providing a comprehensive understanding of how these methods can be effectively combined to enhance efficiency and accuracy. The rest of the paper is structured as follows: Section 2 outlines the proposed framework for image processing, object detection, and tracking, while Section 3 covers the results and their interpretation.

II. PROPOSED FRAMEWORK

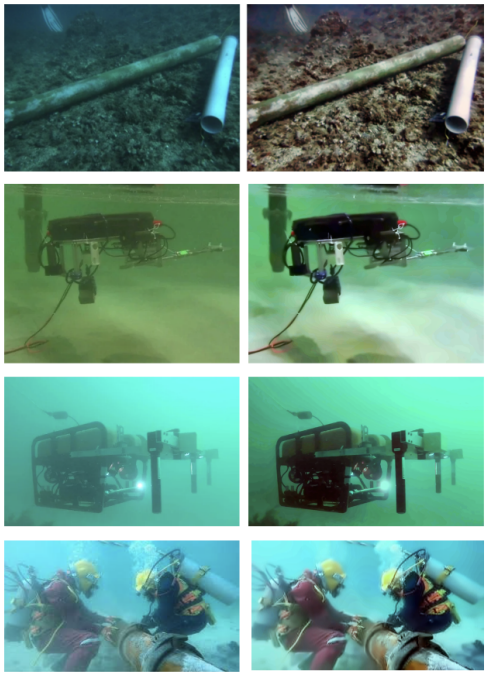

Fig. 1: **Figure on the Left: Raw image; Figure on the Right: Processed image.**

A. Adaptive Image Restoration

The saying "Water flows with an eternal grace, never truly still, forever in motion, reflecting the ever-changing tapestry

[1]Author is with department of Aerospace Engineering, Indian Institute of Technology, Kanpur shubhj21@iitk.ac.in

[2]Author is Faculty with the department of Aerospace Engineering, Indian Institute of Technology, Kanpur dkgiri@iitk.ac.in

of existence" encapsulates the dynamic nature of water, mirroring the perpetual changes in life. Similarly, underwater conditions remain in constant flux. This section considers the adaptive image processing algorithm used for advanced image enhancement and color restoration. The algorithm has been designed keeping in mind computer efficiency and dynamically changing underwater conditions.

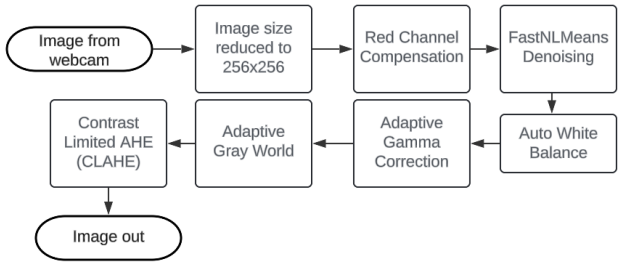

Fig. 2: **Adaptive Image Enhancement Overview**

1) Red Channel Compensation: The underwater environment affects light transmission differently than typical media due to seawater's absorption of wavelengths. Red light, with the longest wavelength, is absorbed first, while green and blue wavelengths are better preserved. This means that the green and blue channels contain complementary color information to offset the significant attenuation of the red channel. Consequently, in underwater images, the green and blue channels are used to compensate for the red channel for each pixel $I(x)$.

Given an input image in BGR format, we split the image into it's three separate channels: blue (B), green (G), and red (R). We then calculate the mean intensity values of the red and green channels, denoted as μ_R and μ_G, respectively. The red channel is adjusted by adding a scaled difference between the mean green and mean red intensities.

Mathematically, the compensated red channel $R_{\text{compensated}}$ is computed as follows:

$$R_{\text{compensated}} = R + \alpha(\mu_G - \mu_R) \tag{1}$$

where α is a constant scaling factor.

To ensure the pixel values remain within the valid range $[0, 255]$, the compensated red channel is clipped:

$$R_{\text{clipped}}(x, y) = \min(\max(R_{\text{compensated}}(x, y), 0), 255) \tag{2}$$

By applying these compensations, we enhance the color balance in underwater images, addressing the common issue of color distortion due to the absorption properties of water.

2) Fast Non Linear Means Denoising: Fast Non-Local Means (FastNLMeans) denoising is effective for underwater images, which often suffer from noise due to the scattering and absorption properties of water. Unlike traditional methods that only consider local neighborhoods, FastNLMeans compares each pixel with its neighbors and averages them based on similarity, preserving important details while reducing noise. The algorithm adapts to heterogeneous noise patterns, enhancing image clarity and visibility. In OpenCV, the `fastNlMeansDenoisingColored` function is used. Mathematically, the denoised value $\hat{I}(p)$ of a pixel p is given by:

$$\hat{I}(p) = \frac{\sum_{q \in \mathcal{N}(p)} I(q) \cdot \exp\left(-\frac{\|I(p)-I(q)\|^2}{h^2}\right)}{\sum_{q \in \mathcal{N}(p)} \exp\left(-\frac{\|I(p)-I(q)\|^2}{h^2}\right)}$$

where $I(p)$ is the pixel intensity, $\mathcal{N}(p)$ is the neighborhood of p, and h is the filtering parameter.

3) Adaptive White Balancing Algorithm: The adaptive white balancing algorithm adjusts the color balance of an image by dynamically correcting color imbalances. This is achieved through an analysis of the color distribution and applying statistical corrections. The algorithm uses histograms and cumulative distribution functions (CDFs) to identify and adjust color values, ensuring accurate color representation. *Algorithm Description:-*

1) **Histogram Calculation:** Compute the histograms for the Red, Green, and Blue channels:

   $H_c(v) =$ number of pixels with intensity v in channel c

   where $c \in \{\text{R}, \text{G}, \text{B}\}$ and v represents pixel intensity. Normalize histograms to obtain the probability distribution functions (PDFs):

   $$P_c(v) = \frac{H_c(v)}{N_c}$$

   where N_c is the total number of pixels in channel c.
2) **Compute Cumulative Distribution Function (CDF):** Calculate the CDF from the normalized histograms:

   $$\text{CDF}_c(v) = \sum_{i=0}^{v} P_c(i)$$

3) **Determine Percentile Values:** Identify the intensity values corresponding to the desired percentiles for each channel. Using a default percentile of perc $= 0.007$ (0.7%):

   $$\text{Percentile}_c(\text{perc}) = \text{argmin}_v \{\text{CDF}_c(v) \geq \text{perc}\}$$

   Calculate the minimum and maximum pixel values:

   $$\min_c = \text{Percentile}_c(\text{perc})$$

   $$\max_c = \text{Percentile}_c(1 - \text{perc})$$

4) **Construct Look-Up Tables (LUTs):** Build LUTs to map original pixel values to corrected values:

   $$\text{LUT}_c(v) = 255 \times \frac{v - \min_c}{\max_c - \min_c}$$

   This normalization maps pixel values from the range $[\min_c, \max_c]$ to the full intensity range $[0, 255]$.
5) **Apply LUTs to Image:** Adjust each pixel value using the corresponding LUT:

   $$I_{\text{out}}(x, y, c) = \text{LUT}_c(I_{\text{in}}(x, y, c))$$

   where $I_{\text{in}}(x, y, c)$ is the original pixel value at position (x, y) in channel c, and $I_{\text{out}}(x, y, c)$ is the corrected pixel value.

6) **Output the Corrected Image:** Generate and return the image with adjusted color balance, ensuring improved accuracy in color representation.

This approach ensures that the image colors are adjusted dynamically based on the statistical properties of the image, resulting in a more accurate white balance correction that adapts to varying lighting conditions.

4) Adaptive Gamma Correction with Weighting Distribution (AGCWD) : The adaptive gamma correction algorithm is used to enhance image contrast by dynamically adjusting the gamma value based on the image's histogram. This method is based on [15].

Algorithm Description: The algorithm follows these steps:

1) **Image Conversion:** The image is converted to grayscale if it has more than one channel.
2) **Histogram Calculation:** The histogram of the grayscale image is calculated and normalized to obtain the probability density function (PDF):
$$P(v) = \frac{H(v)}{N}$$
where $H(v)$ is the histogram value at intensity v and N is the total number of pixels.
3) **PDF Weight Calculation:** Weights for the PDF are calculated using the alpha value:
$$W(v) = P_{\max} \left(\frac{P(v) - P_{\min}}{P_{\max} - P_{\min}} \right)^{\alpha}$$
where $P_{\max}$ and $P_{\min}$ are the maximum and minimum values of the PDF, respectively.
4) **CDF Calculation:** The cumulative distribution function (CDF) is computed from the PDF weights:
$$CDF(v) = \sum_{i=0}^{v} W(i)$$
5) **Gamma Correction:** A look-up table (LUT) is built to adjust the pixel values:
$$LUT(v) = \left(\frac{v}{255} \right)^{1-CDF(v)} \times 255$$
6) **Apply LUT:** Each pixel value in the image is adjusted using the LUT. The gamma correction is applied to each channel (Red, Green, Blue) of the original color image using the computed gamma values.

To enhance video contrast efficiently while reducing computational demands, we incorporate a temporal-based (TB) technique into the Adaptive Gamma Correction with Weighting Distribution (AGCWD) method.

The process starts by storing the first video frame to establish a reference mapping curve for the AGCWD method. For subsequent frames, we use an entropy-based model to evaluate changes between consecutive frames. The entropy H for each frame is calculated using:

$$H = -\sum_{l=0}^{L-1} \text{pdf}(l) \log(\text{pdf}(l)) \tag{3}$$

where $\text{pdf}(l)$ denotes the probability density function of pixel intensities.

If the difference in entropy between frames exceeds a threshold (0.05), the frame storage and mapping curve are updated. Otherwise, the existing curve is used for the new frame. This approach, introduced and detailed by Huang et al. [15], improves computational efficiency while maintaining high contrast quality.

5) Adaptive Gray World Algorithm for Image Color Compensation: The adaptive gray world algorithm is designed to improve color constancy in images by compensating for illumination variations. This method combines both global and local means to achieve a more balanced color correction, making it particularly effective for images with varying lighting conditions.

Algorithm Description : The adaptive gray world algorithm operates as follows:

1) **Global Mean:** Compute the global mean color value:
$$\text{global\_mean} = \frac{1}{N} \sum_{i=1}^{N} I(i)$$
where N is the total number of pixels and $I(i)$ is the color value of pixel i.
2) **Local Mean:** Compute the local mean using a window size:
$$\text{local\_mean}(x, y) = \frac{1}{w \times w} \sum_{i=-\frac{w}{2}}^{\frac{w}{2}} \sum_{j=-\frac{w}{2}}^{\frac{w}{2}} I(x+i, y+j)$$
where w is the window size.
3) **Combine Means:** Adjust each pixel using a weighted combination:
$$\begin{aligned} &\text{adjusted\_pixel}(x, y, c) = \\ &I(x, y, c) \times \left(\text{global\_weight} \times \frac{\text{global\_mean}(c)}{\text{global\_mean}(c)} \right. \\ &\left. + (1 - \text{global\_weight}) \times \frac{\text{local\_mean}(x, y, c)}{\text{global\_mean}(c)} \right) \end{aligned} \tag{4}$$
,where c represents the color channels (Red, Green, Blue).
4) **Normalize and Convert:** Normalize the adjusted image:
$$\begin{aligned} &\text{normalized\_image} = \\ &\frac{\text{compensated\_image} - \min(\text{compensated\_image})}{\max(\text{compensated\_image}) - \min(\text{compensated\_image})} \\ &\times 255 \end{aligned} \tag{5}$$

6) Apply CLAHE: Contrast Limited Adaptive Histogram Equalization (CLAHE) is employed to enhance image contrast by dividing the image into small, non-overlapping tiles and calculating the histogram for each tile. A clip limit is applied to these histograms to constrain contrast enhancement and mitigate noise amplification. The clipped histograms are then used to compute the Cumulative Distribution Function (CDF), which remaps the grayscale values to enhance local contrasts. CLAHE is applied to each color channel (RGB)

independently, improving overall image contrast without introducing color artifacts. This method is particularly effective for images with varying lighting conditions.

B. Object Detection

We employ a method to execute the object recognition algorithm in real time on a lightweight embedded board by writing a lightweight and divisional-sized algorithm [13]. Two deep learning-based systems were used to reduce the amount of computation and increase the accuracy and overall structure. First, as shown in Fig. 3, semantic segmentation is performed using ENet to mask out the region of interest (ROI) from an image or frame. Using trained weights, any object in an image can be segmented. Currently, many highly diverse models are being developed. However, it was difficult to find a model that could be executed simultaneously in different operating systems, taking into account the differences in the board and the host PC and their corresponding execution methods. Since the background around an object in underwater scenario is not as complex as the outside world (in our application case), ENet segments the images way faster and with high accuracy (an example of an underwater gate in very poor conditions can be seen in Fig 4. To create a masked image, the original image is converted into an array, then the mask and the original image are combined using the bitwise and operation functions in OpenCV.

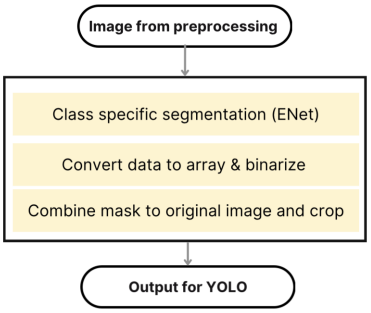

Fig. 3: **Algorithm used for masking the ROI**

The processed image with masks is subsequently inputted into the YOLO algorithm for object identification, revealing both the bounding box and the likelihood of the identified object. Our initial focus in deploying detection algorithms is on achieving rapid and precise performance with minimal computational load. To optimize for both efficiency and accuracy, we specifically utilized the YOLO-v7-tiny version, tailored to our objective of detecting underwater gates and buoys within murky water conditions.

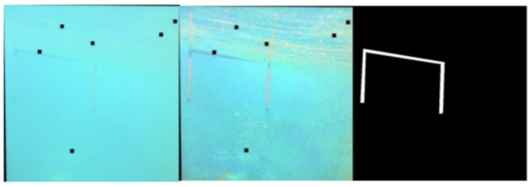

Fig. 4: **Figure on the left: Raw image; Figure in the middle: Restored image; Figure on the right: Segmented Region of Interest (ROI) of a gate.**

The decision to separate segmentation and object detection into distinct models was intentional, and aimed at optimizing code deployment. This separation offers several advantages. Firstly, ENet along with pre-image postprocessing can function efficiently on a small FPGA board, while YOLO can be executed on a more capable FPGA. Merging segmentation and detection into a single model demands an embedded board with significantly higher performance, which is unnecessary when the tasks are run separately. Utilizing two less powerful boards proves to be far more cost-effective than relying on a single high-performance board.

Having successfully detected the objects of interest, the next critical step involves tracking these identified objects over time to monitor their movement and interactions.

C. Object tracking

After establishing the bounding boxes around objects via YOLO, they are forwarded to the CSR-DCF [14], known as the Discriminative Correlation Filter with Channel and Spatial Reliability tracker. The CSR-DCF performs comparably to state-of-the-art trackers that utilize computationally demanding high-dimensional features, but it runs considerably faster and delivers top tracking performance among real-time trackers. While ideally suited for short-term tracking tasks, certain factors, such as changes in lighting conditions or the temporary disappearance and reappearance of objects within the frame, can pose challenges. To address these issues, a closed-loop control strategy is adopted, using the image preprocessing algorithm to detect changes in image parameters and the YOLO algorithm to detect targets and form a tracking closed loop to re-track lost targets. When these parameters surpass a predetermined threshold or the target is lost from the frame, the tracker is updated with the new bounding box from YOLO. The thresholds are determined by experimentation with CSR-DCF.

Mathematically, the CSR-DCF optimization problem can be formulated as follows:

$$\min_{w} \sum_{c=1}^{C} \|x_c * w_c - y\|^2 + \lambda \sum_{c=1}^{C} \|w_c\|^2 + \sum_{c=1}^{C} \left(\|(1 - R_c) \odot x_c * w_c\|^2 + \|(1 - R_c) \odot w_c\|^2 \right) \quad (6)$$

where x_c represents the feature map of the c-th channel, w_c is the filter for the c-th channel, y is the desired output (Gaussian-shaped response), λ is the regularization parameter, and R_c is the spatial reliability map for the c-th channel. The first term ensures the correlation filter w_c matches the desired response y, while the second term is a regularization term to prevent overfitting. The third term incorporates the spatial reliability map to adaptively weigh different regions of the feature map, thus improving robustness against occlusions and background clutter.

III. RESULTS AND DISCUSSIONS

In this section, we assess our framework's performance in tracking an underwater gate, illustrated in Fig. 4. Our setup employs a Logitech C920 webcam with a 78-degree field of view and a 30fps video output. The ENet model runs on a Linux-based Intel NUC featuring an 8th-gen Core i7 CPU, while YOLO operates on a Jetson TX2 with a quad-core ARM Cortex-A57 MPCore processor. To gauge the effectiveness on lightweight embedded devices, all tests are conducted on the CPU. Changes are quantified in terms of FPS and memory usage while transforming images of various gates under different conditions into a real-time webcam slideshow.

When YOLO-Tiny operates independently without any image preprocessing, the fps and processing time exhibit fluctuations in response to varying underwater conditions and viewing angles. However, integration of Adaptive Image Restoration, ENet, and YOLO-Tiny yields stable fps and processing time irrespective of changing underwater conditions and viewing angles, with an average **fps** of **9.07**.

Algorithm	Average FPS
YOLO-V7-Tiny	**7.9**
Enhanced Images + ENet + YOLO-v7-tiny	**9.07**

TABLE I: Average FPS measurement

When using YOLO-Tiny alone, the average frames per second (FPS) achieved was 7.9. However, with the combined use of ENet and YOLO-Tiny, the FPS increased to 9.07. Although the average accuracy for both models showed little change, there was a notable increase in precision and a decrease in error values with the ENet-YOLO-Tiny combination. Additionally, the difference in memory usage between ENet with Tiny-YOLO and Tiny-YOLO alone is minimal, making the former a more efficient option without significant additional resource demands.

While Tiny-YOLO showed lower average accuracy compared to the standard YOLO, which offers significantly higher accuracy, our primary focus was on achieving higher FPS and minimizing resource consumption. These considerations led us to opt for Tiny-YOLO, as it better meets our requirements for real-time performance and efficiency.

Upon passing the bounding box to the CSR-DCF tracker, the system maintained gate tracking at an average rate of nearly 10 FPS while using minimal system resources. The CSR-DCF tracker proved to be highly efficient, especially when the entire gate was visible within the frame. However, it encountered challenges when parts of the gate were obscured or when the gate temporarily exited and re-entered the frame. To mitigate these issues, a closed-loop control strategy was implemented. In this strategy, if the CSR-DCF tracker failed to maintain the bounding box due to occlusions or frame exits, the YOLO model would immediately reinitialize the bounding box.

This dynamic interaction between YOLO and the CSR-DCF tracker ensured continuous and reliable tracking, thereby enhancing the robustness of the overall tracking system under varying underwater conditions. The closed-loop control strategy allowed for quick recovery from tracking failures, maintaining the system's effectiveness even in challenging scenarios where parts of the gate might be intermittently obscured or out of the frame.

IV. CONCLUSION

This study highlights a significant enhancement in image restoration using light-weight image processing and object recognition efficiency achieved through two deep learning models. The findings suggest potential applications in Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs), particularly in scenarios requiring recognition of gates, pipes, tools, and similar objects, while operating at lower speeds. Furthermore, the feasibility of implementing this method on lighter and more cost-effective boards compared to those traditionally used for object recognition presents promising opportunities for initiating open-source underwater projects.

In future work, a more diverse dataset will be created to further test and refine the approach. This will involve collecting a wide range of underwater images under different conditions and scenarios to ensure the robustness and generalizability of the framework. By addressing the variability and complexity of real-world underwater environments, the aim is to develop more reliable and accurate object detection and tracking systems. Additionally, expanding the dataset will enable benchmarking against other state-of-the-art methods, providing a clearer picture of its performance and identifying areas for improvement. The goal is to continue enhancing the methodology and validating it across diverse datasets, advancing the capabilities of lightweight embedded devices for underwater applications.

V. ACKNOWLEDGEMENTS

We thank IIT Kanpur for funding our project. We would also like to thank our faculty advisor Dr. Indranil Saha for his support. We also extend our gratitude towards the Institute administration and numerous staff members at the swimming pool.

REFERENCES

[1] Raveendran, S.; Patil, M.D.; Birajdar, G.K. Underwater image enhancement: A comprehensive review, recent trends, challenges and applications. Artif. Intell. Rev. 2021, 54, 5413–5467.

[2] Jian, M.; Liu, X.; Luo, H.; Lu, X.; Yu, H.; Dong, J. Underwater image processing and analysis: A review. Signal Process.-Image Commun. 2021, 91, 116088.

[3] Zhang, S.; Zhao, S.; An, D.; Liu, J.; Wang, H.; Feng, Y.; Li, D.; Zhao, R. Visual SLAM for underwater vehicles: A survey. Comput. Sci. Rev. 2022, 46, 100510.

[4] Li, C.; Guo, J.; Guo, C. Emerging from water: Underwater image color correction based on weakly supervised color transfer. IEEE Signal Process. Lett. 2018, 25, 323–327.

[5] Guo, Y.; Li, H.; Zhuang, P. Underwater image enhancement using a multiscale dense generative adversarial network. IEEE J. Ocean. Eng. 2019, 45, 862–870.

[6] G Acar and AE Adams. Acmenet: an underwater acoustic sensor network protocol for real-time environmental monitoring in coastal areas. IEE Proceedings-Radar, Sonar and Navigation, 153(4):365–380, 2006.

[7] Russell B Wynn, Veerle AI Huvenne, Timothy P Le Bas, Bramley J Murton, Douglas P Connelly, Brian J Bett, Henry A Ruhl, Kirsty J Morris, Jeffrey Peakall, Daniel R Parsons, et al. Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience. Marine geology, 352:451–468, 2014.

[8] Fabiana Di Ciaccio and Salvatore Troisi. Monitoring marine environments with autonomous underwater vehicles: a bibliometric analysis. Results in Engineering, 9:100205, 2021.

[9] David Gibbins and Jonathan Adams. Shipwrecks and maritime archaeology. World Archaeology, 32.

[10] Haibo Niu, S Adams, Kenneth Lee, T Husain, and Neil Bose. Applications of autonomous underwater vehicles in offshore petroleum industry environmental effects monitoring. Journal of Canadian Petroleum Technology, 48(05):12–16, 2009.

[11] Mark Grasmueck, Gregor P Eberli, David A Viggiano, Thiago Correa, Glenda Rathwell, and Jiangang Luo. Autonomous underwater vehicle (auv) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the straits of florida. Geophysical Research Letters, 33(23), 2006.

[12] Marius Rogobete, Stefania L Nita, Marius I Mihailescu, and Christian Voelker. An unmanned underwater vehicle defence system. Scientific Bulletin" Mircea cel Batran" Naval Academy, 25(1):16–22, 2022.

[13] Yun, H.; Park, D. Efficient Object Detection Based on Masking Semantic Segmentation Region for Lightweight Embedded Processors. Sensors 2022, 22, 8890. https://doi.org/10.3390/s22228890

[14] Lukežič, A.; Vojíř, T.; Čehovin Zajc, L.; Matas, J.; Kristan, M. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. International Journal of Computer Vision 2018, 126, 671–688.

[15] Huang, S.-C.; Cheng, F.-C.; Chiu, Y.-S. Efficient Contrast Enhancement Using Adaptive Gamma Correction With Weighting Distribution. IEEE Transactions on Image Processing 2013, 22, 1032–1041. https://doi.org/10.1109/TIP.2012.2226047.