Kraken: Open-Source Computer Vision for Underwater Fish Detection and Measurement

Pedro Gohl
Federal University of Amazonas
Manaus - AM, Brazil
pedro.gohl@icomp.ufam.edu.br

Herbert Rocha
Federal University of Roraima
Boa Vista - RR, Brazil
herbert.rocha@ufrr.br

Felipe Lobo
Federal University of Roraima
Boa Vista - RR, Brazil
felipe.lobo@ufrr.br

Leandro Balico
Federal University of Roraima
Boa Vista - RR, Brazil
leandro.balico@ufrr.br

Abstract—**Brazil's extensive river networks, particularly in the Amazon Basin, are characterized by high turbidity and low underwater visibility, presenting significant challenges for aquatic monitoring and fish farming operations. This study introduces Kraken, an open-source computational system designed for automated fish detection, counting, and size estimation in turbid freshwater environments. Kraken integrates state-of-the-art computer vision frameworks—including YOLO, OpenCV, and TensorFlow—with open hardware platforms such as Arduino and Raspberry Pi, enabling cost-effective and scalable deployment. The system employs a pre-trained Inception convolutional neural network for robust fish classification and utilizes image preprocessing techniques to enhance detection accuracy under challenging conditions. Experimental evaluation demonstrates Kraken's reliability, achieving an average size estimation at a fixed distance of 30 cm and a classification accuracy of up to 90%. By automating fish counting and measurement, Kraken reduces manual labor and operational costs, while its open-source architecture fosters collaborative innovation and adaptability. The results confirm Kraken's effectiveness for real-world aquatic monitoring, highlighting its potential to advance research and practical applications in fish farming and ecosystem management.**

Keywords—**computer system; computer vision; aquatic ecosystem.**

I. Introduction

Brazil's river networks are divided into several hydrographic regions, among which the Amazon Basin stands out as the largest in the world [1]. The Amazon Basin's waters are classified as white, black, or clear [2]. White-water rivers, as described by Barthem and Fabré [3], exhibit high turbidity, which hinders underwater visibility. Santos and Santos [4] attribute the white coloration to the abundance of minerals, a characteristic commonly observed in rivers of northern Brazil.

The demand for identifying submerged objects has grown due to advancements in ocean observations, resulting in the processing of large amounts of visual data from underwater environments. Recognizing fish species and solving object detection problems are crucial tasks in ocean and river observations, benefiting scientists, biologists, and commercial fish farming applications [5].

According to Van Damme [6], it is possible to capture data in an aquatic ecosystem using elementary techniques such as scale drawings, trilateration with a tape measure, displacement measurements, and simple photography. These methods are ideal for reconnaissance of aquatic archaeological sites. However, apart from photography, these methods are not accurate and take a long time to implement, making them prone to human error. Additionally, these techniques only produce two-dimensional and three-dimensional representations with a low level of detail of the studied location.

According to Huimin et al. [7], sound was used to map environments, for example, by emitting a pulse that reflects off the ocean floor and creates a sonogram. A sonogram is a graphic representation of sound frequencies. The images obtained through this sonar technique resemble optical images but with much higher levels of detail. The reflections created by the sonar form a fan-shaped pattern as the pulse moves. These reflections then generate a series of image lines perpendicular to the beam. Depending on the environment being studied, the sonogram can be confusing, often requiring extensive experience to identify the images.

In the Van Damme [6] study, it is stated that there are advanced submerged three-dimensional collection techniques, which are efficient and highly accurate. According to Dianne et al. [8], one of these techniques is the remote stereo-video technique, which consists of two remotely controlled cameras on the bottom of the ocean. Van Damme [6] also describes another technique as Computer Vision Photogrammetry, which allows a series of images to be loaded into dedicated software to generate a three-dimensional model of the scene or object.

The difficulty of carrying out operations (e.g., size estimation) in an environment with turbid waters is usually encountered in fish farming. Therefore, it is necessary to use methods to improve visibility of submerged fish and reduce the required manpower. For instance, one method involves removing fish suitable for consumption in a given fish farm based on technical reports. In this context, the problem addressed in this study can be expressed in the following question: How can a computational system be designed to identify fish from the northern region of Brazil in an aquatic ecosystem with high turbidity and partial observability, enabling estimation of their distance and dimensions?

Aiming to contribute to the development of computational systems for image processing, the context of this study is focused on designing and developing a computational system capable of detecting fish in turbid water ecosystems, called Kraken. The proposed system is controlled from a computer and will estimate the distance and dimensions of the fish through image classification. This way, it facilitates the identification of fish in fishing reservoirs.

The Kraken system integrates a comprehensive suite of open-source technologies and open hardware platforms to provide a robust and cost-effective solution for underwater fish detection and measurement. The primary software components include TensorFlow [9], an open-source machine learning framework that supports the development and deployment of advanced neural network models. OpenCV [1], a widely used open-source library, offers extensive tools for image processing, feature extraction, and template matching, enabling effective enhancement and analysis of underwater images under challenging conditions. Additionally, YOLO (You Only Look Once) [10], a state-of-the-art real-time object detection algorithm, enhances Kraken's ability to accurately identify and localize fish in video streams with high efficiency.

From a hardware perspective, Kraken employs platforms such as Arduino [2] and Raspberry Pi [3], which exemplify the principles of open hardware. Arduino microcontrollers are characterized by their flexibility, affordability, and ease of integration with various sensors and actuators, making them well-suited for embedded control tasks within the buoy infrastructure. Raspberry Pi single-board computers provide greater computational power and support for full operating systems, facilitating complex processing and wireless communication. The use of open-source software and hardware not only reduces development and operational costs but also promotes collaborative innovation, scalability, and adaptability.

II. Microcontrollers vs. Microprocessors

According to Heath [11], modern computers—such as desktops and laptops—are based on microprocessors, which enable them to perform a wide variety of functions. In contrast, systems based on microcontrollers are typically designed to execute a single, repetitive task. Microprocessors were initially developed to replace early calculators, featuring chips that could be reprogrammed. This innovation allowed developers to update functionality by modifying the software, rather than designing new hardware for each application.

Heath [11] further explains that the term "embedded systems design" encompasses a broad spectrum of microprocessor-based projects. These are not limited to simple microcontrollers but may also include personal computers running dedicated software. Microcontrollers, on the other hand, integrate several essential components—such as RAM, input and output interfaces, and reprogrammable memory—into a single chip [12].

A representative example of a microcontroller-based system is the Arduino Uno. The Arduino Uno features 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, USB connectivity, and a dedicated power supply input. Its microcontroller can be reprogrammed multiple times via a computer, and, if necessary, the microcontroller itself can be replaced at a low cost. This flexibility and affordability make Arduino a popular choice for prototyping and educational purposes [13].

In contrast, Single Board Computers (SBCs), such as the Raspberry Pi, integrate both the processor and memory on a single board, blurring the line between microcontrollers and general-purpose computers [14]. SBCs offer advantages such as low cost, reduced energy consumption, high processing capability, and a large number of general-purpose input/output (GPIO) ports. Unlike typical microcontrollers, SBCs are capable of running full operating systems and supporting more complex applications, making them suitable for a broader range of embedded and computational tasks.

In summary, while microcontrollers like Arduino are ideal for dedicated, real-time control applications with limited resources, SBCs such as the Raspberry Pi provide greater computational power and versatility, supporting more complex software stacks and multitasking environments.

III. Computer vision

Human vision has no difficulties in identifying small variations in translucency and shading in images, distinguishing objects from the background image. For human vision, it is easy to identify three-dimensional structures, differentiate formats, textures, colors, and people in images. It can even discern the emotions a person might be feeling just by looking at pictures [15]. Computer vision aims to emulate human vision by taking images as input and providing an interpretation of those images as output [16].

According to Marengoni and Stringhini [16], the pre-processing step is usually adopted when we want to extract

[1]https://opencv.org
[2]https://www.arduino.cc
[3]https://www.raspberrypi.com

information. According to Gonzalez et al. [17], there are three levels of image processing where low-level processing involves primitive operations, such as processing for noise reduction, contrast enhancements, and sharpness enhancements. Mid-level image processing involves tasks such as segmentation of the image into regions and classifying objects. At this level, images are received as input and attributes extracted from the input images, such as contours, are returned. High-level image processing involves the computer's understanding of object recognition and performing cognitive functions usually associated with human vision.

IV. Frameworks for Image Identification

TensorFlow is a machine learning framework that has a library allowing development for different platforms, from server clusters to mobile devices [9]. Machine learning solutions have made progress in recent years, reaching levels of image abstraction that surpass human perception. One type of model that has achieved satisfactory results, even surpassing human capacity in image recognition tasks, is the Convolutional Neural Network (CNN) [18]. Convolutional networks are at the heart of most solutions to image recognition problems [19]. The classification model that will be adopted in this study is Inception, which, according to Szegedy et al. [19], has a much lower computational cost than other models, making its use viable for scenarios with limited resources, such as mobile processing.

OpenCV is a library that allows the use of functions for data manipulation. The library contains over 2500 optimized algorithms for various functions, including object tracking and optical sensor motion tracking [20]. In the experimental analysis of this study, some transformation functions were tested, which will serve to manipulate the data obtained by the optical capture sensors. Additionally, according to Ayushi et al. [21], OpenCV contains many modules, including a module for image processing, object identification, and machine learning. A straightforward approach to pattern recognition, known as Template Matching, involves verifying if a pattern appears in the source image. This method includes conducting a comprehensive search of the template within the source image and marking each location where the pattern is detected. OpenCV offers different methods to perform template matching, such as square difference, correlation (see Equation 1, where i and j are the coordinates of the points inside the region and $f(i,j)$ is the intensity level of the binary image at position i and j), and correlation coefficient [22].

$$R_{ccorr} = \sum_{ij} [t(i,j) * f(x+i, y+j)]^2 \tag{1}$$

Furthermore, YOLO is a real-time object detection system that, according to Joseph et al. [10], calculates probabilities for the detected boxes. Jiang et al. [23] introduced YOLO as a viral and widely used algorithm. YOLO exhibits strong generalization ability, as it can learn highly generalized features that can be transferred to other domains. The original YOLO network was pre-trained on ImageNet using 224×224 pixels and later fine-tuned for object detection on the VOC dataset using 448×448 pixels. According to Prakhar et al. [24], YOLO operates as follows: each image frame is divided into a small matrix, and that section is responsible for detection. The model is trained using both dependent and independent variables. The independent variable represents the image on which object detection is performed, while the dependent variables consist of position coordinates. The dependent feature comprises five variables: the x-position, y-position, width, height, and confidence score, which can be either 0 or 1.

V. Related Work

A framework based on a convolutional neural network was proposed by Hongwei et al. [5] for fish recognition in videos recorded underwater. In our study, background images are removed using methods based on sparse and low-level matrix decomposition, as adopted by Hongwei et al. [5], thus eliminating the background and isolating only the fish. The extracted data are then passed through convolutional layers to a non-linear layer, followed by classification using Support Vector Machine (SVM) [25]. The learning rate decreases as the number of parameters increases, necessitating optimization in the learning filter. Similar to the study by Hongwei et al. [5], our method also adopts a CNN for image detection.

Che et al. [26] propose a framework for underwater fish recognition, consisting of a fully unsupervised learning technique and an error-resistant classifier. The framework analyzes certain characteristics of the input data, such as the shape of the fish's outline, and then applies sorting criteria based on these characteristics. The unsupervised approach, as proposed in [26], generates a binary hierarchical class structure, where each node represents a classifier. The results demonstrate the accuracy of the proposed framework in both conditions: when using a public database and in controlled environments with high uncertainty.

The study by Foresti and Gentili [27] proposes a system (an autonomous vehicle) for the detection and tracking of objects submerged in water. In this case, the system focuses on detecting pipes, which can extend for kilometers, and autonomously positioning based on these detections. Furthermore, the authors [27] presented a technique that applies color compensation by modifying values in the color channels to account for light disturbances and image scaling. After scaling, the images are reduced to sizes of $1/16$ for edge detection and $1/32$ to locate anodes within the pipes.

In this context, object identification typically employs neural networks to classify images in real time and detect obstacles. Consequently, the vehicle's automatic navigation traces routes and paths based on processed images and geometric resonance. Gentili and Foresti [27] proposed a system capable of detecting pipes and other structures, even in challenging conditions such as low luminosity, light dispersion and attenuation, and issues like sand and debris on pipes.

Stephane et al. [28] propose an algorithm for processing and restoring underwater environment images that suffer from loss of capture distance, light distortion, low contrast, low color saturation, and other problems. Current image processing methods focus on distortion and attenuation caused by light and need knowledge of the environment being studied. The algorithm proposed in [28] is an algorithm that automates the pre-processing of underwater images. The results presented by Stephane et al. [28] shown that the algorithm reduces disturbances in underwater images, and improves image quality. It is worth noting that the algorithm adopts techniques that will be important for the pre-processing of the images of the computational system proposed in our study, techniques such as:

1) Moiré Pattern: removal through a spatial analysis by detecting peaks in the Fourier transforms and deleting them, assuming they represent Moiré patterns.
2) Image resizing: this transformation standardizes the image, facilitating the application of algorithms such as Fourier transforms.
3) Homomorphic filter: used to correct lighting and improve image contrast.

The comparative analysis in Table I underscores the unique strengths and limitations of the Kraken system relative to existing underwater detection frameworks. Distinctive Advantages of Kraken:

1) Open-Source Integration: Kraken exclusively leverages open-source technologies (TensorFlow, OpenCV, YOLO) and open hardware platforms (Arduino, Raspberry Pi), significantly reducing deployment costs while enabling collaborative development and community-driven improvements.
2) Practical Fish Farming Applications: Unlike existing solutions that primarily focus on research or marine exploration, Kraken specifically targets real-world fish farming operations by providing automated counting and size estimation capabilities that directly reduce manual labor requirements and operational costs.
3) Scalable Wireless Buoy Architecture: The distributed buoy-based deployment model enables simultaneous monitoring across multiple aquatic locations through wireless connectivity, offering superior operational flexibility compared to fixed underwater systems or autonomous vehicles that require direct intervention for repositioning.
4) Optimized Single-Class Detection: The focused approach on fish detection enables optimized model performance and reduced computational complexity compared to multi-class underwater object detection systems, resulting in faster processing times and improved accuracy for the specific target application.

Key Limitations Compared to Existing Solutions:

1) Operational Range Constraints: Kraken's size estimation module is calibrated exclusively for objects at a fixed distance of 30 cm, which significantly restricts its effectiveness in scenarios involving variable water depths or dynamic positioning relative to target organisms.
2) Distance-Dependent Calibration: The accuracy of dimensional measurements relies on a predetermined calibration constant (2.75), which may limit the system's adaptability across diverse aquatic environments, camera configurations, or varying underwater visibility conditions.
3) Limited Training Dataset Scope: The neural network was trained on a relatively modest dataset comprising 319 images of Amazonian freshwater fish species. While adequate for proof-of-concept validation, this dataset size may constrain the system's generalization capabilities when encountering diverse fish species, morphological variations, or novel environmental conditions not represented in the training data.
4) Single-Class Detection Framework: Although optimized for computational efficiency, the current implementation focuses exclusively on fish detection without species-level classification, potentially limiting its applicability in biodiversity monitoring or selective aquaculture management scenarios.

Despite these limitations, Kraken represents a significant advancement in accessible underwater monitoring technology, demonstrating how open-source frameworks and cost-effective hardware can be effectively integrated to address real-world challenges in aquatic ecosystem management. The identified constraints provide clear directions for future system enhancement, particularly in adaptive calibration methodologies and expanded training datasets.

TABLE I
COMPARISON OF UNDERWATER DETECTION SYSTEMS ACROSS LEARNING TECHNIQUES AND DEPLOYMENT CHARACTERISTICS.

Study	Learning Technique	Operating Environment	Accuracy
[5]	Sparse and low-rank matrix decomposition for background removal; CNN feature extraction; SVM classification	Controlled underwater environments with moderate clarity	98.75%
[26]	Fully unsupervised learning with binary hierarchical class structure; Error-resistant classifiers	Controlled aquarium environments with high uncertainty conditions	93.8%
[27]	Color compensation and edge detection algorithms; Geometric reasoning for autonomous navigation	Deep ocean environments with low luminosity and debris, and structural elements	97%
[28]	Automated preprocessing algorithms: Moiré pattern removal, homomorphic filtering, contrast enhancement	Various underwater conditions with focus on image quality degradation	With additional synthetic degradations, 71.9% improvement in quality enhancement
Kraken (proposed)	Transfer learning with Inception; R-CNN based fish detection; Single class classification	Turbid freshwater environments; Amazon Basin rivers; Fish farming operations	90%

VI. PROPOSED METHOD

This study presents a computational system called Kraken for classifying, counting, and estimating the size of fish within the aquatic ecosystem with high turbidity. The proposed computational system consists of several interconnected modules that communicate via a wireless network and a physical infrastructure structured as a water buoy. The buoy is composed of sensors for optical capture and can be an embedded system, as shown in Figure 1. An example of an embedded system that could be used to acquire underwater images is a smartphone coupled to a buoy. The smartphone sends the data to the central processing unit for tasks such as image classification and preprocessing using a homomorphic filter to correct lighting and improve image contrast. The system generates a report containing the count and size of objects classified for the end user.

Figure 1 presents the structure of the Kraken system, which is used for collecting underwater images of fish while connecting to a central processing station. The auxiliary solar panel is optional and serves to charge the on-board system, which is coupled to the buoy. Interaction with the Kraken is accomplished through a wireless connection to the central processing. The data gathered from the processing station can be accessed through a web app.

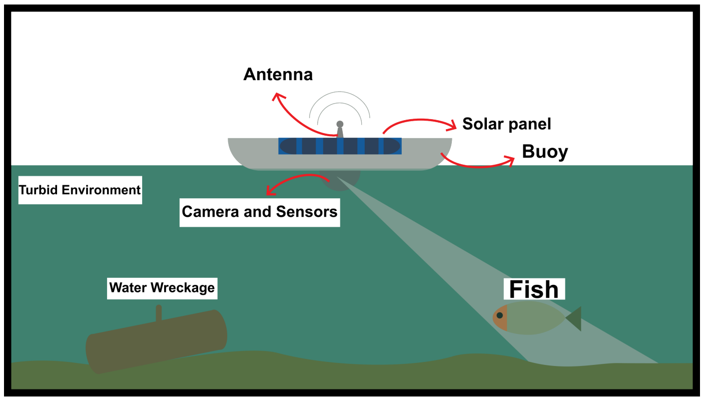

Fig. 1. Experimental configuration of an instrumented buoy for aquatic ecosystem studies.

VII. COMPUTATIONAL SYSTEM FOR COLLECTING AND PROCESSING FISH IMAGES IN AN UNDERWATER ENVIRONMENT WITH LOW VISIBILITY

The physical infrastructure of the water buoy proposed for the Kraken system will include microcontrollers (e.g., Arduino UNO R3) that will be responsible for managing the integrated systems, such as motors, stabilizers, and sensors. These sensors will detect fish and capture images within the underwater environment, even in high turbidity conditions. Furthermore, the buoy will classify the fish and estimate their size. Its adoption aims to facilitate fish farming activities, which typically require a significant amount of labor for fish sorting.

Classified objects will have their sizes speculated. The underwater image is preprocessed before being sent to the classifier, ensuring that it is enhanced for input into the image classification process. In addition to the auxiliary application, the system has an image classification module that utilizes the Tensorflow framework, which yielded better results in preliminary tests compared to other frameworks using Yolo [4].

The Kraken flow starts running on the web application, where the modules will be displayed, giving the user options to open the application, classify, and generate the image report. Additionally, the buoy is connected to the central station, and an image is requested from the buoy. If there is a disconnection with the optical sensor, a fault message is sent. Once the image is sent to the central station, the classification module will preprocess it if necessary. When the image is suitable, it will be processed, and a report will be generated and displayed to the user through the web application.

The sequence diagram presented in Figure 2 details the interaction flow among five main system components: User, Mobile Application (App), Classification Module, Control Board, and Camera. The process initiates with the user's interaction with the application, which generates a request for image capture and classification. This request is forwarded to the classification module, which determines whether a new image acquisition is necessary. A conditional block in the diagram represents this decision, evaluating if the available data are sufficient or if a new capture should be triggered.

If image capture is required, the classification module sends a command to the control board, which executes its own conditional check to decide whether to activate the camera. Upon confirmation, the camera is triggered to perform the capture, returning the raw data to the control board, which then forwards them to the classification module. The classification module processes the image, including steps such as preprocessing, feature extraction, and application of a pre-trained machine learning model. After inference, the result is sent back to the application, which presents the output to the user.

From a software engineering perspective, the diagram emphasizes critical validation points for system robustness and efficiency: (i) the preliminary assessment in the classification module to avoid redundant captures, optimizing hardware usage and bandwidth; (ii) the conditional decision on the control board to activate the camera only when necessary, reducing energy consumption and extending device lifespan; (iii) the assurance of data integrity and synchronization during transmission between the camera, control board, and classification module; and (iv) the confirmation of synchronous result delivery, ensuring immediate and complete user feedback.

Architecturally, the flow illustrates a hybrid edge-cloud architecture, where embedded devices (control board and camera) perform acquisition and preprocessing tasks close to the data source, while the application layer centralizes the classification logic. This decoupling reduces latency, optimizes resource usage, and increases resilience to connectivity failures. The presence of conditional blocks at strategic points demonstrates an event-driven and resource-optimized design, which is essential for embedded systems with constraints on energy, bandwidth, and processing capacity. Furthermore, the modularity of the architecture supports scalability and maintainability, allowing updates to components such as classification models or capture hardware without significant impact on other layers.

In summary, the system fulfills the functional requirements of image capture and analysis while incorporating principles of scalable, sustainable, and resilient design, making it suitable for continuous monitoring in industrial and scientific applications.

VIII. Classification Model

The main part of the Kraken system is the object classification model that was generated from training an Inception neural network. In this section, we will describe how the dataset was created and how the model was generated to integrate into the proposed system. For training the classification model, hardware resources were required. For instance, the GPU card used was the RTX 2060 with 6 Gigabytes of GDDR6 memory and 1920 CUDA cores running at 1365Mhz. These resources assisted in the training of the proposed model.

A. Model Training

Aiming to generate the model classification, a dataset was created with images of freshwater fish from the Amazon fauna, totaling 319 images obtained from various websites through a search for fish in the Amazon region. The training was done using an R-CNN network as a base, employing Inception v3 divided into two stages and separated into three training sets.

Before conducting the training, the dataset of images was labeled with the classes that would be fed into the neural network. For instance, in the Kraken model classification, there was only the fish class, which included various fish species from the Amazonian fauna. The LabelImg [5] tool was used to label these images, indicating the location of objects in the input image for training purposes. This allowed for the training of a new class in the neural network's final layer.

The model training process involved conducting 80,000 iterations using TensorFlow. Due to hardware resource constraints,

[4]https://pjreddie.com/darknet/yolo/

[5]https://github.com/tzutalin/labelImg

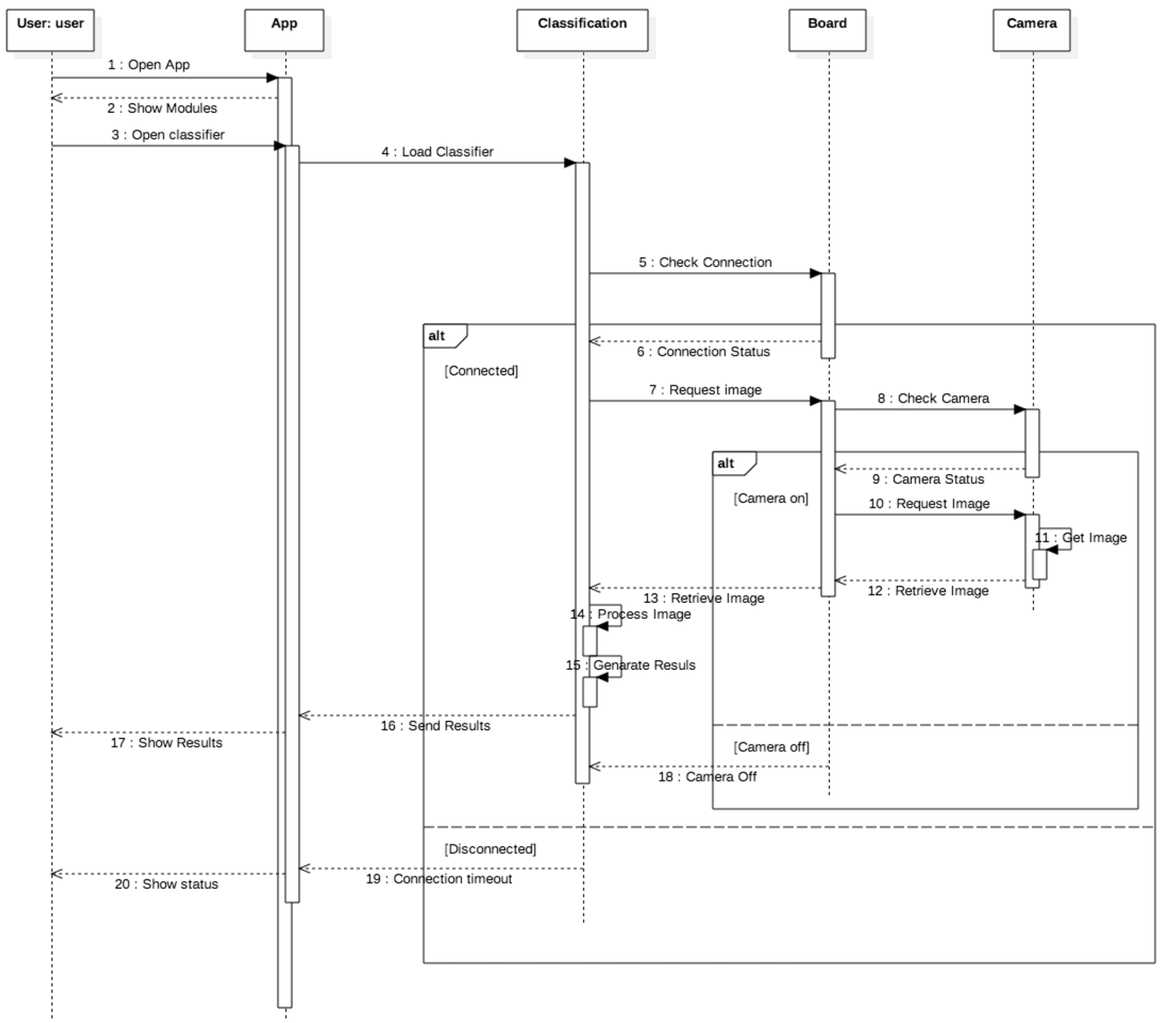

Fig. 2. Sequence diagram of the automated image classification system via embedded device.

the input image size was standardized to 300 pixels, ensuring aspect ratio preservation to maintain the morphological features of the fish. Upon completion of training, the classifier achieved an accuracy of 96%.

B. Data Analysis in the Central Processing Station

In Kraken, after collecting the images using capture sensors (e.g., cameras), the images are sent to the Central Station either via a wireless network or through file upload. At the Central Station, the images are processed by the classification object module, which utilizes OpenCV for pre-processing filters to enhance the quality of the collected images. The classification model is then applied to the processed images to perform counting and size estimation. This process employs Tensorflow, which was previously modeled using the pre-trained Inception model to retrain the last layers of its neural network.

C. Preprocessing Module

The preprocessing step consists of checking the image quality to decide whether it should undergo certain processing steps to improve it and focus on the contour of the fish. Kraken applies pretesting of the image to determine if it can be classified. In situations where the observed environment does not have turbid waters, the Kraken process sends the image as input to the classification module.

The pre-processing module was developed using the Python (3.6) programming language, which utilizes the OpenCV [6] (version 3.2) and PILLOW [7] (version 6.0) libraries, integrating the image classification software proposed in this study. To evaluate the results of the Kraken preprocessing module, we conducted tests using a dataset of images containing five different classes (e.g., Fish and Wrecks), with each class having 20 different samples. The experiment was divided into two parts: one using pre-processed images and the other using non-pre-processed images.

Analyzing the preliminary test results (see Table II) of preprocessing images, using the Tensorflow framework with the Inception library v3, and employing the pre-trained model with images from the ImageNet database, which contains images of North American contexts, reduced the classifier's accuracy related to fish from the Amazon fauna. We observed a subtle difference between the experiment using the pre-processed samples and the unprocessed images. However, we noted that the number of correct results in both approaches was greater than the number of incorrect results.

TABLE II
RESULTS OF THE PRE-PROCESSING MODULE.

Labels	**Pre-processed Images**		**Non-preprocessed Images**	
	CORRECT	INCORRECT	CORRECT	INCORRECT
Fish	14	6	18	2
Mammal	13	7	17	3
Stone	11	9	15	5
Wreckage	19	1	19	1
Snake	12	8	14	6

D. Counting and Size Estimation Module

The counting is performed by receiving the video file sent by the user, then processing the analyzed video using the classification model. The object counting is executed during the video's execution, with frames divided to facilitate processing. Size estimation is performed by collecting data from the identification paths of each object detected in the image. Data is collected by the same classification script, as can be seen in Figure 3. The size data is then transformed into size estimates based on a normalization constant, calibrated to calculate sizes up to 30cm away. The results are displayed in a web interface that provides information such as the total amount, average size, average accuracy, and a graph illustrating the size distribution into small ($< 13.9cm^2$), medium ($\leq 27.8cm^2$), and large ($> 27.8cm^2$) categories. The interface also allows for making new classifications based on video files or streaming video.

[6]https://opencv.org
[7]https://pillow.readthedocs.io

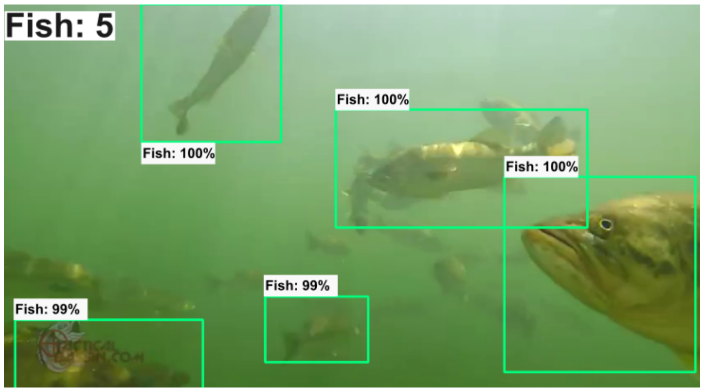

Fig. 3. Fish Counting Results: Automated detection and size estimation of fish in underwater video frames.

IX. Experimental Evaluation

This section aims to present the planning, execution, and analysis of the results obtained from testing the Kraken as a proposed solution. The evaluation focuses on the functioning of the Kraken prototype through experiments in a controlled scenario, using external samples, while considering the system's effectiveness and efficiency in nominal use situations.

A. Experiment Planning and Design

This experimental evaluation investigates the ability of the proposed solution to classify and estimate the size of fish in an aquatic environment with high turbidity. In this sense, the proposed system was implemented in a software called Kraken [8]. The following research questions were defined:

RQ1: Is the accuracy and precision of the Kraken solution sufficient to prove its reliability?
RQ2: Is the Kraken system able to estimate the unit size of fish via video capture when, in an underwater environment, there are more than one unit?
RQ3: Is the Kraken system able to count the number of fish in an underwater environment via video capture?

To answer the research questions, the experiments were designed to validate the modules. For these experiments, seven videos with varying durations (between 3 and 6 seconds) and a resolution of 640x360 pixels were selected. The videos used to validate the Kraken system were captured in a controlled environment and extracted from snippets of underwater fishing recordings obtained from YouTube [9].

[8]https://github.com/danielgohl13/Kraken
[9]https://www.youtube.com/watch?v=HxjLc9OJNqU&t=94s

B. Experiments Execution and Analysis of the Results

In this section, the results obtained by the experiments are analyzed, trying to answer the research questions created for the validation of the Kraken system modules. Answering the second research question (**RQ2**), tests were performed with two videos recorded in a controlled environment, with each video scene previously defined to test aspects of the proposed system's functionality. For example, images outside the aquatic environment were included.

Analyzing the results obtained by the Kraken system, it is noted that the system was able to identify the fish. To calibrate the size estimation module in Kraken, a constant value of 2.75 was defined. This constant is multiplied by the pixel value representing the size of the object in the classification video, resulting in the estimated size of the object in centimeters. It should be noted that this calibration process does not affect the classification accuracy; it only changes the size estimate. The estimation module is efficient when the object is at a distance of 30 cm from the camera (as defined in the video recording), and the average detection accuracy was 99%. Therefore, this proof of concept demonstrates the correct execution of the main functions of the Kraken system.

C. Analysis of the counting module and size estimation

Using a sample of five videos, the statistical test of Mann-Whitney [29] was adopted to compare the central tendencies of two independent samples of equal sizes from two unpaired groups. This comparison aimed to verify whether or not they belong to the same population. To address the third question (**RQ3**) proposed in the section, an experiment was defined. It consists of three stages: (1) human counting of the videos; (2) analysis of the Kraken classifier results; and (3) application of the Mann-Whitney statistical test to the results obtained in stages (1) and (2), using the data represented in Table III.

TABLE III
NUMBER OF FISH ACCOUNTED FOR IN THE EXPERIMENTS.

	Amount of Fish Human Counting	**Amount of Fish Kraken Counting**
Vídeo 1	9	7
Vídeo 2	4	4
Vídeo 3	3	3
Vídeo 4	10	8
Vídeo 5	6	3

After executing the proposed test and applying the Mann-Whitney statistical analysis, a p-value of 0.22 was obtained. Since this value exceeds the significance threshold $\alpha = 0.05$, the test results suggest that there is no statistically significant difference between the two samples—human counting and Kraken counting—indicating similar central tendencies. However, in 60% of the cases (3 out of 5 videos), Kraken underestimated the fish count compared to human observation, suggesting a potential systematic bias that warrants further investigation. For future work, we propose the following improvements: (i) expanding the sample size to enable more robust validation; (ii) analyzing the underlying causes of Kraken's systematic undercounting; (iii) refining the detection algorithms to reduce false negatives; and (iv) implementing continuous performance monitoring across diverse environmental scenarios.

To further enhance the reliability of Kraken's fish counting module, we recommend expanding the experimental scenarios to include challenging conditions such as overlapping fish, variable lighting, and increased turbidity. These additions will help evaluate and improve the system's robustness in real-world environments. Periodic retraining and validation with newly collected data are essential to maintain high accuracy as Kraken is deployed across diverse aquatic settings. Moreover, integrating multi-frame tracking algorithms—such as Kalman filtering combined with Hungarian assignment for associating detections across video frames [30]—can substantially reduce systematic undercounting by linking individual fish detections over time. This approach mitigates false negatives caused by occlusions or rapid movement, resulting in more consistent and accurate fish counts.

D. Analysis of the Complete System Operation

To answer the first research question (**RQ1**) proposed in this section, data from Section IX-A was analyzed. The solutions were found to be satisfactory for the efficiency of the proposed system, Kraken. By analyzing the gathered data from the experiments, it was possible to perceive that the accuracy in both experiments was above 90%, both in a controlled environment and in excerpts collected from underwater fishing videos. Considering the results found, we can highlight the advantages and disadvantages of using the proposed system:

- **Advantages**:
  - Automate the task of counting fish in ponds, reducing the need for manpower and facilitating the separation of fish, making them ready for consumption.
  - Demonstrating promising results in fish classification, counting, and size estimation.
  - It offers a low-cost solution compared to the labor and time required for manually removing, measuring, and counting fish.
- **Disadvantages**:
  - To calculate the size of objects based on distance.
  - High processing time when used with a central station that has a performance equal to or lower than the market average.

X. Conclusion and Future Works

This study presents the development of Kraken, a computational system based on a water buoy equipped with a single-board computer or smartphone for fish detection in fluvial environments with high turbidity. Leveraging machine learning techniques, Kraken assists fish farming operations by automating fish detection and size estimation tasks. The system is designed with modularity in mind, allowing for future integration of additional sensors—such as oximeters for water quality monitoring or motors for locomotion and stabilization. The buoy's architecture facilitates easy attachment of new components, minimizing wear and supporting long-term extensibility.

For future work, we plan to: (i) refine and optimize Kraken's modules to improve efficiency in real-world deployments; (ii) conduct extensive field experiments to validate system performance under diverse environmental conditions; (iii) collaborate with dam operators and fish farming professionals to assess practical interest and gather user feedback; and (iv) identify and implement further enhancements or new functionalities to address specific needs in aquaculture, thereby increasing the system's impact and applicability.

References

[1] W. J. Junk, M. G. M. Soares, and P. B. Bayley, "Freshwater fishes of the amazon river basin: their biodiversity, fisheries, and habitats," *Aquatic Ecosystem Health & Management*, 2007.
[2] H. Sioli, *The Amazon: limnology and landscape ecology of a mighty tropical river and its basin*. Springer Science & Business Media, 2012.
[3] R. B. Barthem and N. N. Fabré, "Biologia e diversidade dos recursos pesqueiros da Amazônia," *A pesca e os recursos pesqueiros na Amazônia brasileira*, vol. 1, pp. 17–62, 2004.
[4] G. M. d. Santos and A. C. M. d. Santos, "Sustentabilidade da pesca na Amazônia," *Estudos Avançados*, vol. 19, no. 54, pp. 165–182, 2005.
[5] H. Qin, X. Li, J. Liang, Y. Peng, and C. Zhang, "DeepFish: Accurate underwater live fish recognition with a deep architecture," *Neurocomputing*, vol. 187, pp. 49–58, 2016.
[6] T. Van Damme, "Computer vision photogrammetry for underwater archaeological site recording in a low-visibility environment," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 5, p. 231, 2015.
[7] H. Lu, Y. Li, and S. Serikawa, "Computer vision for ocean observing," *Studies in Computational Intelligence*, vol. 672, pp. 1–16, 2017.
[8] D. L. Watson, E. S. Harvey, M. J. Anderson, and G. A. Kendrick, "A comparison of temperate reef fish assemblages recorded by three underwater stereo-video techniques," *Marine Biology*, vol. 148, no. 2, pp. 415–425, 2005.
[9] Martin~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, Greg~S.~Corrado, Andy~Davis, Jeffrey~Dean, Matthieu~Devin, Sanjay~Ghemawat, Ian~Goodfellow, Andrew~Harp, Geoffrey~Irving, Michael~Isard, Y. Jia, Rafal~Jozefowicz, Lukasz~Kaiser, Manjunath~Kudlur, Josh~Levenberg, Dandelion~Mané, Rajat~Monga, Sherry~Moore, Derek~Murray, Chris~Olah, Mike~Schuster, Jonathon~Shlens, Benoit~Steiner, Ilya~Sutskever, Kunal~Talwar, Paul~Tucker, Vincent~Vanhoucke, Vijay~Vasudevan, Fernanda~Viégas, Oriol~Vinyals, Pete~Warden, Martin~Wattenberg, Martin~Wicke, Yuan~Yu, and Xiaoqiang~Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/
[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[11] S. Heath, *Embedded Systems Design*. Elsevier Science, 2002.
[12] E. White, *Making Embedded Systems: Design Patterns for Great Software*. O'Reilly Media, 2011, vol. 2011.
[13] J. Boxall, *Arduino for Arduinians: 70 Projects for the Experienced Programmer*. No Starch Press, 2023.
[14] G. Halfacree, *The Official Raspberry Pi Beginner's Guide: How to use your new computer*. Raspberry Pi Press, 2023.
[15] R. Szeliski, *Computer Vision : Algorithms and Applications*. Springer Science & Business Media, 2010, vol. 5.
[16] M. Marengoni and S. Stringhini, "Tutorial: Introdução à visão computacional usando opencv," *Revista de Informática Teórica e Aplicada*, 2009.
[17] R. C. Gonzalez, R. E. Woods, and others, *Digital image processing*. Addison-wesley Reading, 1992.
[18] TensorFlow, "Image classification," 2023. [Online]. Available: https://www.tensorflow.org/tutorials/images/classification
[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *CoRR*, vol. abs/1512.0, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567
[20] OpenCV, "Opencv," 2023. [Online]. Available: https://opencv.org
[21] A. Sharma, J. Pathak, M. Prakash, and J. N. Singh, "Object detection using opencv and python," in *3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021.
[22] M. Marengoni and D. Stringhini, "High level computer vision using opencv," in *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, 2011, pp. 11–24.
[23] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, 2022, the 8th International Conference on Information Technology and Quantitative Management: Developing Global Digital Economy after COVID-19.
[24] P. Agrawal, G. Jain, S. Shukla, S. Gupta, D. Kothari, R. Jain, and N. Malviya, "Yolo algorithm implementation for real time object detection and tracking," in *2022 IEEE Students Conference on Engineering and Systems (SCES)*, 2022, pp. 01–06.
[25] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[26] M. C. Chuang, J. N. Hwang, and K. Williams, "A feature learning and object recognition framework for underwater fish images," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1862–1872, 2016.
[27] G. L. Foresti and S. Gentili, "A Vision Based System for Object Detection in Underwater Images," *International Journal of Pattern Recognition and Artificial Intelligence*, no. 02, 2000.
[28] S. Bazeille, I. Quidu, L. Jaulin, and J.-P. Malkasse, "Automatic underwater image pre-processing," in *Caracterisation Du Milieu Marin (CMM)*. HAL open science, 2006. [Online]. Available: https://hal.science/hal-00504893v1
[29] N. Nachar, "The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution," *Tutorials in quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008.
[30] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2016.