AUTONOMOUS UNIVERSITY OF BAJA CALIFORNIA
Faculty of Engineering, Architecture and Design
Master and Doctorate in Sciences and Engineering

Evaluation of underwater image enhancement algorithms on embedded systems

Thesis
for obtaining the degree of
Doctor in Sciences

that presents:

OSCAR ADRIAN AGUIRRE CASTRO

Director of thesis

Dr. Everardo Inzunza González

Ensenada, Baja California, México. November, 2022.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

Evaluation of underwater image enhancement algorithms on embedded systems

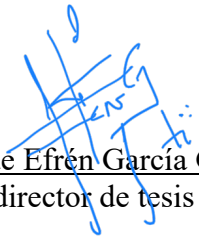
TESIS

Que para obtener el grado de Doctorado en Ciencias presenta:

Oscar Adrian Aguirre Castro

Aprobada por:

Dr. Everardo Inzunza González
Director de tesis



Dr. Enrique Efrén García Guerrero
Co-director de tesis

Dr. Oscar Roberto López Bonilla
Miembro del comité

Dr. José Ricardo Cárdenas Valdez
Miembro del comité

Dr. Jesús Everardo Olguín Tiznado
Miembro del comité

Ensenada Baja California, México, noviembre, 2022

RESUMEN de la tesis de **Oscar Adrian Aguirre Castro**, presentada como requisito parcial para obtener el grado de DOCTOR EN CIENCIAS, del programa de Maestría y Doctorado en Ciencias e Ingeniería de la UABC. Ensenada, B. C. México, Noviembre de 2022.

Evaluación de algoritmos de mejoramiento de imágenes submarinas en sistemas embebidos

Resumen aprobado por:

Dr. Everardo Inzunza González
Director de Tesis

Dr. Enrique Efrén García Guerrero
Co-director de Tesis

La mejora de las imágenes subacuáticas ha avanzado considerablemente debido a su contribución a la ingeniería marina y la exploración submarina. Este hecho se ha visto reflejado en los últimos años con la propuesta de numerosos algoritmos que mejoran la calidad de las imágenes submarinas. En este trabajo de tesis doctoral se presenta una evaluación comparativa de tres algoritmos basados en los modelos Retinex implementados en cinco sistemas embebidos de alto rendimiento. Estos algoritmos son el Modelo Retinex de Escala Única (SSR), el Modelo Retinex de Escala Múltiple (MSR) y el Modelo Retinex de Escala Múltiple con Restauración del Color (MSRCR). Estos algoritmos realizan la ecualización del histograma para distribuir los píxeles, reducir el color predominante, realizar la corrección del color y el contraste, y lograr un balance de blancos automático para mejorar la iluminación. Esta tesis emplea cinco dispositivos de borde como Beagle Board, Odroid-XU4, Raspberry Pi 4, Jetson Nano y Jetson TX2 para mejorar las imágenes subacuáticas y comparar su rendimiento. Para evaluar la calidad de las imágenes subacuáticas mejoradas se utilizan cuatro métricas de calidad sin imagen de referencia, como son UIQM, UCIQUE, BRISQUE y Entropía. El algoritmo MSRCR consigue los mejores resultados de calidad cuando se implementa en el sistema embebido Jetson TX2. Tiene una diferencia de 0,46 segundos en el tiempo de procesamiento de las imágenes de 147×196 píxeles respecto a un ordenador personal (PC) de alto rendimiento. La implementación de estos algoritmos en sistemas embebidos ofrece una excelente relación costo-beneficio frente a un PC tradicional, teniendo en cuenta las métricas de calidad de imagen, precisión, exactitud, consumo energético, precio, ligereza, tamaño, portabilidad y fiabilidad. Estos resultados son prometedores para los vehículos submarinos no tripulados y autopropulsados con visión artificial para la exploración

Palabras clave: Mejoramiento de imágenes; imágenes submarinas; Retinex; computación de frontera; sistema embebido.

ABSTRACT of the thesis of **Oscar Adrian Aguirre Castro**, presented as a partial requirement to obtain the degree of DOCTOR IN SCIENCES, of the Master and Doctorate program in Sciences and Engineering of UABC. Ensenada, B. C., Mexico, November, 2022.

Evaluation of underwater image enhancement algorithms on embedded systems

Abstract approved by:

Dr. Everardo Inzunza González
Director of thesis

Dr. Enrique Efrén García Guerrero
Co-director of thesis

The enhancement of underwater imaging has advanced significantly due to its contribution to marine engineering and underwater exploration. This fact has been reflected in recent years with the proposal of numerous algorithms that improve the quality of underwater images. This thesis presents a benchmarking of three algorithms based on the Retinex models implemented on five high-performance embedded systems. These algorithms are the Single Scale Retinex Model (SSR), Multi-Scale Retinex Model (MSR), and the Multi-Scale Retinex Model with Color Restoration (MSRCR). These algorithms perform the histogram equalization to distribute pixels, reduce the predominant color, perform color and contrast correction, and achieve an automatic white balance to improve illumination. This thesis employs five edge devices such as Beagle Board, Odroid-XU4, Raspberry Pi 4, Jetson Nano, and Jetson TX2 to enhance underwater images and benchmark their performance. Four quality metrics without a reference image such as UIQM, UCIQUE, BRISQUE and Entropy are used to evaluate the quality of the enhanced underwater images. The MSRCR algorithm achieves the best quality results when it is implemented on Jetson TX2 embedded system. It has a difference of 0.46 seconds in the processing time of 147×196 pixels images concerning a high-performance personal computer (PC). Implementing these algorithms on embedded systems offers an excellent cost-benefit ratio versus a traditional PC, considering image quality metrics, precision, accuracy, energy consumption, price, lightweight, size, portability, and reliability. These findings hold great promise for unmanned and self-propelled underwater vehicles with artificial vision for exploration.

Keywords: Image enhancement; underwater image; Retinex; edge computing; embedded system.

A mi Familia

Agradecimientos

*A **DIOS**, por tenerme siempre bajo su protección, por las bendiciones que me ha dado todos los dias de mi vida.*

*Al **Dr. Everardo Inzunza González**, por su amistad, por haberme dirigido en la realización del presente trabajo de tesis de doctorado, la atención y el apoyo incondicional que me ha brindado.*

*Al **Dr. Enrique Efrén García Guerrero**, por su amistad, atención, por sus enseñanzas que me permitieron desarrollar con éxito el presente trabajo de tesis de doctorado.*

*Al **Dr. Oscar Roberto López Bonilla**, por su amistad, sus comentarios y enseñanzas, que me permitieron desarrollar el presente trabajo de tesis de doctorado.*

*Al **Dr. Jóse Ricardo Cárdenas Valdes , Dr. Jesús Everardo Olguin Tiznado**, un especial agradecimiento por sus comentarios muy acertados en las presentaciones y revisiones del presente trabajo de tesis de doctorado.*

*A mi esposa **Yurivia Miranda Zavala**, por ser siempre mi apoyo incondicional en todos los proyectos de mi vida, por su motivación en los momentos dificiles y ser un pilar de mi familia.*

*A mis hijos **Pamela Aguirre, Oscar Santiago Aguirre, Vania Aguirre** , que con su alegria y cariño son un motivo para seguir adelante preparandome.*

*A mis padres **Francisco Oscar Aguirre González, Ana Elia Castro Lozoya**, por estar siempre alientandome a cumplir todos mis sueños y que mas se puede agradecer a las dos personas que siempre han visto por mi, me han enseñado a crecer tanto personal, como profesionalmente, gracias por TODO!!!.*

*A mis hermanas, **Eliana Aguirre y Paloma Aguirre** , por ser ejemplos de vida, por su apoyo y sus consejos.*

*A mis **familiares y amigos**, que siempre con sus consejos y comentarios, me hicienron crecer personalmente, gracias por su apoyo incondicional.*

*Al **CONACyT**, que con su programa de becas me proporcionaron los recursos necesarios durante esta etapa importante en mi vida profesional.*

A nuestra alma mater: ***Universidad Autónoma de Baja California****, que es como mi segundo hogar y donde se me ha permitido desarrollarme profesionalmente.*

Al ***coordinador de posgrado****, Dr. Priscy Alfredo Luque Morales, por sus asesorías administrativas.*

Table of Contents

Table of Contents (Continuation)

List of figures

List of figures (Continuation)

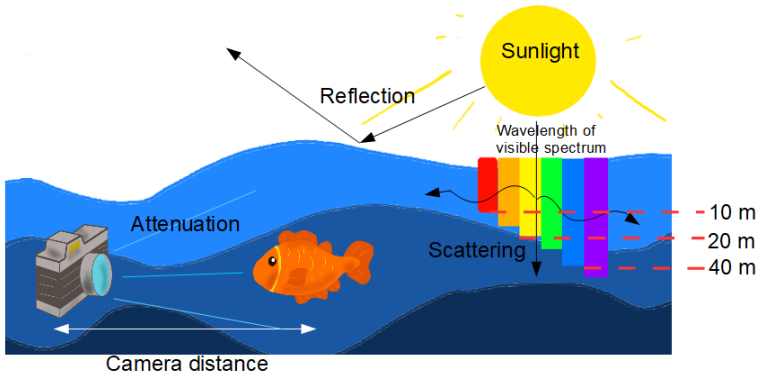
List of tables

Chapter I

Introduction

Compared to rivers, lakes, and other types of waters, marine resources are abundant but have not yet been fully explored. The marine environment below the ocean's surface is more complicated and hazardous, and the risk coefficient associated with artificial exploration and development is excessively high Hu et al. (2021). In modern times, there are a variety of approaches to both the exploration and protection of the seafloor Prasath and Kumanan (2020).The improvement of underwater images is a significant contribution to marine engineering and underwater robotics. It can be used for various applications, including underwater surveillance and the inspection of pipes located on the seafloor, as well as other applications with a shorter range Jian et al. (2021). According to Anwar and Sahu Anwar and Li (2020); Sahu et al. (2014), deteriorating the quality and characteristics of the underwater image can be identified by a lack of contrast, blurring, color aberrations, and noise Hou et al. (2019); Wang et al. (2019, 2020). Practical applications such as feature extraction, target recognition, and structure matching are severely hindered Chun Zhou et al. (2020). In these applications, underwater cameras play an essential part in the gathering of data by photographing and filming underwater survey operations carried out with remotely operated vehicles (ROV) and autonomous underwater vehicles (AUV) Liu et al. (2020); Tang et al. (2018). These vehicles have inside high-performance embedded systems and cameras for short-range scanning as shown in Hmue and Pumrin (2019); He et al. (2020); Tolstonogov et al. (2019); Aguirre-Castro et al. (2019).

In the research that has been done on the topic, histogram equalization is the



Figures 1: Figure of the wavelength striking the sea, inspired from Li et al. (2019)

technique that has been found to be the most successful in terms of improving color and contrast. Underwater picture enhancement also makes a significant contribution to information retrieval and filtering Liang et al. (2021); Li et al. (2016). Ancuti Ancuti et al. (2012) proposes an underwater image enhancement algorithm based on fusion, in which four weights were used to determine which pixels appear in the final image. Rizzi Rizzi et al. (2003) proposes an image enhancement technique that performs automatic color equalization (ACE); this algorithm is based on the adaption of human vision and aims to replicate the internal human visual system. ACE is an abbreviation for automatic color equalization. Iqbal Iqbal et al. (2007) performed a linear contrast normalization in the RGB color space, saturation and intensity are stretching in the HSI color space. Drews Drews et al. (2016) proposed the traditional dark channel, a method initially used for single-image haze removal, to enhance underwater images. Ancuti Ancuti et al. (2018) showed that the green channel is the counterpart of the red channel, as it compensates the attenuation of the red channel and the blue channel, and attenuation of the green channel. W. Zhang Zhang et al. (2021b) shows an underwater image enhancement algorithm with color correction and adaptive contrast enhancement. Yang Yang et al. (2020) A solution to the poor contrast and color distortion

present in underwater photos was provided in the form of a method for enhancing underwater photographs based on local contrast correction (LCC) and multi-scale fusion. Deep neural networks are utilized for various purposes, including identifying distinct species, within the field of underwater picture enhancement Cao et al. (2020); Jin and Liang (2017). Y. Xu Xu et al. (2017) the usage of neural networks for categorization of underwater photos was demonstrated to be inferior to that of airborne photographs, but improved with the addition of data generated by generative adversarial networks.

Nowadays, Retinex models are being widely used in computer vision for underwater image enhancement Zhang et al. (2022); Oladi et al. (2022); Muniraj and Dhandapani (2021); Zhou et al. (2021); Hassan et al. (2021); Caizhen et al. (2021); Hu (2021); Sun et al. (2021); Tang et al. (2021); Zhuang et al. (2021). The most common algorithms based on the Retinex model are Single Scale Retinex Model (SSR) Parthasarathy and Sankaran (2012), Multi-Scale Retinex Model (MSR) Zhou et al. (2021); Zhang et al. (2017), and the Multi-Scale Retinex Model with Color Restoration (MSRCR) Palacios et al. (2020); Jobson et al. (1997a). Their main advantages can be summarized as follows: They are derived from color constancy theory, and thus must simultaneously exhibit the properties of dynamic range compression, color independence concerning the spectral distribution, and color and luminosity reproduction. These properties are achieved by applying logarithmic transformations to the image, eliminating the illuminance component, and representing each pixel as a logarithmic transformation Parthasarathy and Sankaran (2012). These algorithms are characterized by their low computational load, which are suitable to be implemented on embedded systems.

On the other hand, the primary metrics commonly used in the literature to measure the quality of underwater images are classified as follows, (i) with a reference image, and (ii) without reference image; for the former, they are the mean squared

error (MSE), the peak signal-to-noise ratio (PSNR) and structured similarity indexing method (SSIM) Padmavathy and Priya (2020); for the latter, they are blind or reference-less image spatial quality evaluator (BRISQUE) Mittal et al. (2012), Entropy Li et al. (2019, 2016), underwater image quality measure (UIQM), and underwater color image quality evaluation (UCIQUE) Panetta et al. (2016); Yang and Sowmya (2015); Xiang et al. (2021); Zhang et al. (2021a). Recently, some authors Li et al. (2020a); Hou et al. (2020); Sara et al. (2019) have focused their efforts on benchmarking dataset and algorithms related to underwater image enhancement.

According to the reviewed literature, only one paper Tang et al. (2019) reports the use of the Jetson TX2 embedded system for the enhancement of underwater images. However, there are still open research problems, such as the benchmarking of algorithms implemented on different edge devices, optimization of underwater image quality, efficiency, performance, and energy consumption. This doctoral thesis aims to evaluate the quality and performance of underwater image enhancement algorithms based on Retinex models, as well as to present a benchmarking of five embedded systems suitable for the development of underwater exploration vehicles, such as AUV. The chosen algorithms are based on the Retinex theory such as Single Scale Retinex (SSR) Parthasarathy and Sankaran (2012), Multi-Scale Retinex (MSR) Zhou et al. (2021); Zhang et al. (2017), and Multi-Scale Retinex with Color Restoration (MSRCR) Palacios et al. (2020); Jobson et al. (1997a). These algorithms are implemented on five high-performance embedded systems commonly used in research and applications that work in real-time. The benchmarking of these algorithms is performed on Beagle Board, Odroid-XU4, Raspberry Pi 4, Jetson Nano, and Jetson TX2 embedded systems. Additionally, these algorithms are implemented on a personal computer (PC) to compare the quality metrics and performance of the different hardware. The proposed algorithms perform histogram equalization by distributing pixels, reducing the dominant color, improving color and

contrast, and applying an automatic white balance to improve lighting conditions. The algorithms are coded in Python language, achieving competitive results regarding performance and quality metrics versus a PC.

I.1 Problem Statement

Although many land, air, and oceanic teams have the ability to roam and investigate the environment in which they work, adding a sense of vision to these gadgets will increase their sensory capacity. The vast majority of current artificial vision systems operate in controlled light conditions, which is an open research challenge in the area. In order to carry out the exploration in real surroundings, a novel approach is proposed in this thesis protocol that is invariant to fluctuations in lighting, scale, rotation, and position of the items of interest. In this proposal, we will work with underwater environments.

I.1.1 Description of the Problem

Although there are various land, air and oceanographic teams with the ability to travel and explore the environment in which they work, giving them a sense of vision will expand the sensory capacity of these devices. An open research problem in the field of artificial vision is that the vast majority of current systems work under controlled light environments. In this thesis protocol, a new method is proposed that is invariant to variations in lighting, scale, rotation, and position of the objects of interest in order to carry out exploration in real environments, in this proposal we will work with underwater environments.

I.1.2 Research questions

In the development of this thesis, a series of questions can be formulated, which will be answered during the reading. Then some of the questions are proposed, which were answered in the methodology and results section.

- ¿Which high-performance embedded system has the best performance for running image enhancement algorithms?
- ¿Which underwater image enhancement algorithms will have better performance in embedded systems?
- ¿How much information will be lost when using embedded systems with less computational capacity?

I.2 Project Objectives

In this thesis, the following general objective is proposed. in which the specific objectives achieved are mentioned

I.2.1 General objective

Develop a new artificial vision algorithm capable of adapting to unmanned aquatic vehicles using a high-performance embedded system with multiprocessing capacity and allowing target tracking in real aquatic environments.

I.2.2 Specific objectives

- Select high-performance embedded systems with optimal multiprocessing for the execution of artificial vision algorithms.
- Evaluate algorithms for underwater image enhancement and pattern recognition.

- Implement machine learning algorithms for pattern recognition in an embedded system with multiprocessing.
- Evaluate metrics for quality measurement of enhanced underwater images (MSE, Histogram, PSNR, Correlation).
- Release benchmark performance and quality metrics of images obtained on personal computers and high-performance embedded systems.

Chapter II

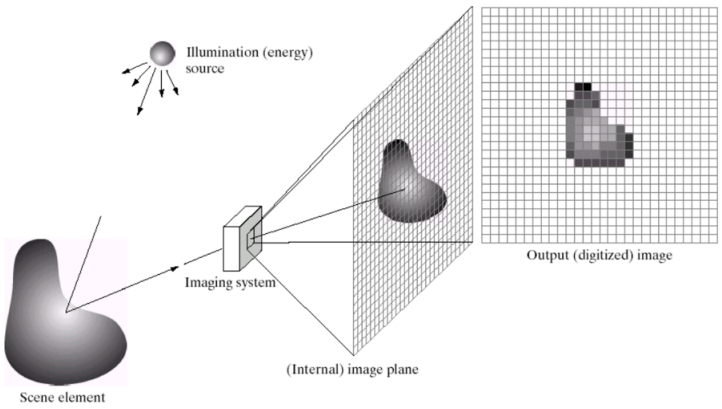
Basic principles of Enhancement Underwater Images

In this second chapter, some key ideas of the methodology used in the development of this thesis, as well as the materials required to answer the provided problem, will be discussed. In addition, this chapter will discuss the materials needed to solve the problem. The process of image improvement begins with the gathering of pictures at the very first phase.

II.1 Processing Digital image

The purpose of digital image processing is to improve the appearance of images and to make certain details more evident. The usefulness of image processing is very broad and covers many fields. One example is images obtained for medical diagnostic purposes. This is because an image can be defined as a two-dimensional function $f(x, y)$ where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates is called the intensity or gray level of the image at the point. Digital image processing, in general terms, involves 2D, 3D, and image sequence recognition, analysis, manipulation, transmission, and other related areas, this represented in the Figure 2. To apply the gamma correction algorithm to the image in Figure 30, digital image processing is used to perform pixel-by-pixel spatial operations.

Picture acquisition is the initial step in digitally processing an image, and this step could involve receiving an image that is already in digital form. Figure 2 provides a



Figures 2: Example of image acquisition, image taken from Gonzalez and Woods (2008)

representation of one example of the process of acquiring a digital image.

This scene must have a source of illumination or energy for the imaging system to record it, and the scene must be projected onto the picture plane. The element of the scene is shown here. This image is going to be digitized, and then it will be displayed as the output. The primary purpose of enhancement is to modify an image so that the final product is better suited for a specific use than the original image. The application that is being discussed in this thesis is for underwater photographs.

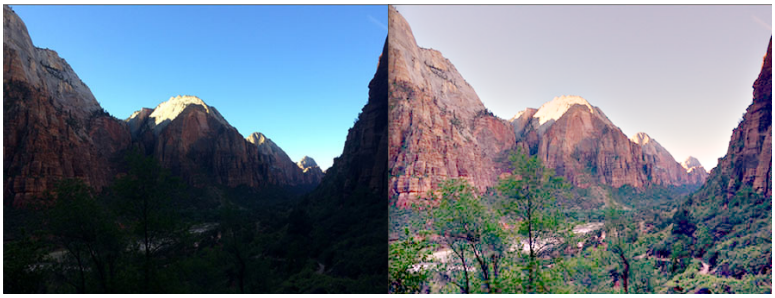
II.2 Image enhancement

Improving an image's overall visual quality is referred to as "image enhancement," and it describes the process itself. There are no tried-and-true methods for enhancing photographs at this time because varied lighting circumstances, camera models, and other factors can result in considerable differences in the appearance of different images. In general respects, enhancing procedures are of nature.

For instance, if the original image is not sharp, a suitable algorithm known as a sharpening mask can be designed to improve the subjective quality of the original image by increasing its sharpness without altering any of the image's other qualities, as

mentioned by Thyagarajan (2006) in his book. This can be accomplished by improving the quality of the original image. In image enhancement, digital image processing is used to make corrections and recover most of the information lost or degraded due to capture hardware, natural phenomena, and low lightness and brightness.

In the Figure 30 shows an example of image enhancement, which uses gamma correction, where the pixel intensities change range, making the image darker or lighter than the original image.

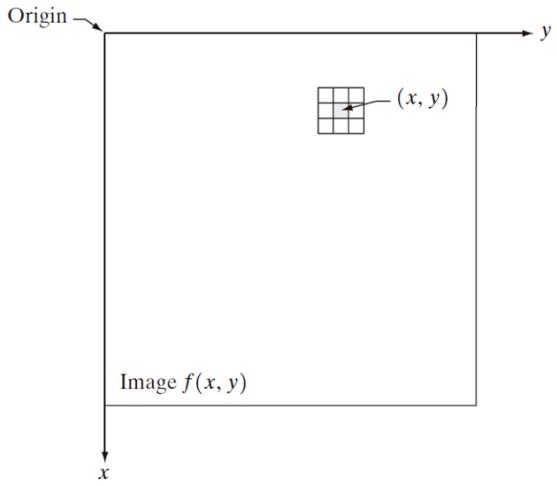


Figures 3: Example of an image with low light intensity applying the gamma correction algorithm using OpenCV.

Methods that improve images can be broken down into two primary groups: those that work in the frequency domain and those that work in the spatial domain. The word "spatial domain" refers to the image plane itself, and the techniques that fall into this category are those that directly manipulate the pixels that make up an image. The Fourier transform of a picture can be modified using techniques that fall under the purview of frequency domain processing. The collective pixels that make up a picture are what's meant to be referred to as an image's "spatial domain." Methods belonging to the spatial domain are processes that work directly on these pixel coordinates. The expression of the Equation1 will be used to denote the processes that occur in the spatial domain.

$$g(x, y) = T[f(x, y)] \tag{1}$$

where, $f(x, y)$ is the input image, $g(x, y)$ is the processed image, and T is an operator on f,defined over some neighborhood of (x, y).



Figures 4: A $3X3$ neighborhood about a point (x, y) in an image, image taken from Gonzalez and Woods (2008)

By employing these operations in the spatial domain, it is possible to rectify some flaws in the images obtained from the capture systems. As was just discussed, a wide variety of natural occurrences can have an effect on the systems used to collect underwater photos, and these can be found both above and below the sea. The low brightness, scattering, reflection, and other natural phenomena combine to produce an image of poor quality or one that has deteriorated. Because of this, image enhancement has a significant role to play in the processing of underwater photographs. Specifically, this is because of the reason stated above.

II.3 Underwater image

One of the primary challenges associated with taking accurate underwater photographs using electronic equipment, such as a camera or camcorder, arises from the underwater environment. This is because the light and the tone of the light source change depending on the depth of the scene at the time the photograph is taken. This is because the filtering of different wavelengths occurs at different degrees of depth, which results in different color tones depending on the lighting conditions. A color change is brought on by uniformity, caused by the scattering of incident light and the distinct wavelengths received by the device sensor. It has been suggested that the application of the Retinex model, which is predicated on the knowledge of the human visual system (SVH-HVS) and makes use of the phenomenon of color constancy, be used as a solution in order to cut down on the efficiency response that is brought on by the issues that have been outlined above. This model will perform the function of a preprocessing step for the scene that will be processed, and the outcome will be displayed by modifying the color content of the image, primarily in places with low levels of light. In this thesis, we will assess the experimental results of two models applied to underwater photos. These models are the Simple-Scale Retinex model, the Multi-Scale Retinex model, and the Multi-Scale Retinex model with Color Restoration.

Sahu et al. (2014) explains that processing the captured underwater image is necessary because the quality of underwater images affects, and these images present some severe problems compared to images from a 1clearer environment. Mentions that processing of the captured underwater image is necessary because of this. Low contrast, poor viewing circumstances (absorption of natural light), non-uniform lighting, minor color fluctuations, pepper noise and blurring all contribute to a significant increase in

the amount of noise produced in underwater photographs. As a result of these factors, there are a few different approaches to treating these issues. Quality underwater photographs by reducing noise particles using various techniques, and the other way that has been used is expanding the RGB color level. Both of these techniques have been utilized. Image enhancement can also be accomplished using the direct SSR, MSR and MSRCR techniques.

II.3.1 Underwater images algorithms

The Human Vision System (HVS) model has as its characteristic the possibility of having both the chromaticity components of the objects and separately, the wavelength of the illumination source. These components can be processed independently without affecting the features in which it is not required to manipulate the brightness and chromaticity information at the same time. A desirable feature for underwater image enhancement is to have color constancy.

II.3.2 Color Constancy

Color constancy is based on the field of radiometry where it describes how the receptors present in the retina measure the total energy captured in each of the sensors present in the retina. Color constancy theory states that, from a photometric point of view, a red object illuminated by a green light source should produce the same spectral distribution as a green object illuminated by a red light source. But even under these conditions, the color of the object remains stable regardless of changes in the wavelength of the light source, and this is due to a theory is known as "Color Constancy". This behavior is described in the Equation 2.

$$f(x, y) = G(x, y)R_i(x, y)I_i, \tag{2}$$

where, I_i is the ilumination source, $f_i(x, y)$ is the intensity of each pixel of an image

at position (x, y), $G(x, y)$ is a factor of scene characteristics, $R_i(x, y)$ is the reflectance, and sub-index i corresponds to the color channel of the image, red, green and blue.

$$Q_i = \int S_i(\lambda)E(\lambda)d\lambda \tag{3}$$

The color constancy can be described by the energy captured by each of the sensors represented in Equation 3. Where Q_i is the measured energy, $S_i(\lambda)$ it responds to a curve of three receivers, the index i represents each of the color channels (red, green, blue), and $E(\lambda)$ the irradiance, i.e. the incident power on the receivers per unit area for a given wavelength, considering that the color channels must be in the visible magnetic spectrum. The phenomenon of constancy color separates the reflectance from the scene illumination, which can define the spectral characteristic of the light source. With these values, we can calculate the approximate value of the reflectance as shown in Equation 4.

$$E(\lambda) = R(\lambda)L(\lambda) \tag{4}$$

The function of an electronic video sensor is to convert the scene from a photometric point of view, which captures the product of reflectance and illumination. What is done with the Retinex model is to mimic the mechanism of the Color Constancy phenomenon that the HVS possesses.

II.3.3 Model Retinex

Dr. Land developed the Retinex model theory of color perception. According to this model, three different receptors measure energy at different points in the visible spectrum, and each receptor acts as a unit to process the measured energy. Initially, Dr. Land hypothesized that the process of color interpretation begins in the retina with that of the visual cortex. This retinal brain system that processes part of the visible

spectrum is called Retinex, where the operators are part of the Retinex model and have the quality of photoreceptor simulation in HVS. Described by Dr. Land, the Retinex model is described by biological light acquisition operators that are repeatedly applied to an image along a path or group of paths, and the value of each pixel is compared to neighboring pixels. The differences that can be found in the literature for this model are in the selection of comparison points and the order of comparison. To find the closest approximation to the target, a search for the maximum intensity in each channel is performed, as shown in Equation 5.

$$I_i = max\{f_i(x, y)\} \tag{5}$$

Equation 5 can be changed by calculating the histogram for each color component, considering the light source as the point where the high color component, considering the light source as the point where there is a high energy accumulation. The simple Retinex algorithm takes into account the maximum value in each color component, and the maximum value in each color component, the maximum value in each color component, $maxf_i(x, y)$, for the whole image, and uses it as the representative value of the white color to be used in the respective image component.

II.3.4 Single-Scale Retinex

The Retinex image enhancement technique comes from the color constancy theory, therefore, it must simultaneously achieve the properties of i) dynamic range compression, ii) color independence regarding the spectral distribution, and iii) color and luminosity reproduction, which are achieved by applying logarithmic transformations on the image, eliminating the illuminance component, representing each pixel as a logarithmic transformation of the image, and representing each pixel as a product of illuminance and reflectance Parthasarathy and Sankaran (2012).

The mathematical model of the SSR algorithm Li et al. (2020b) can be expressed by (6).

$$R_{SSR_i}(x,y) = log_{10}[I_i(x,y)] - log_{10}[F(x,y) * I_i(x,y)], \tag{6}$$

where $I_i(x,y)$ is the image distribution, i is the spectral color band, the convolution operation, $F(x,y)$ (sorround function) is the Gaussian surrounding function as shown in (7), the standard deviation σ is a scale parameter that controls the trade-off between color fidelity and dynamic range compression, $R_{SSR}(x,y)$ is the obtained Retinex output.

$$F(x,y) = Ke^{-(x^2+y^2/2\sigma^2)}, \tag{7}$$

where K is a factor to constrain by 8:

$$\int\int F(x,y)dxdy = 1. \tag{8}$$

This model improves the image quality by sacrificing the dynamic range at the time of compression.

II.3.5 Multi-Scale Retinex

Because it is difficult for a single scale to achieve a balance between color fidelity and detail preservation, unlike SSR, this model applies a Gaussian kernel with a weight for each iteration. Multiple scales can handle the different details at multiple levels of the image. In this case, the algorithm is developed with three scales that are appropriate enough to represent a small scale, a large scale and an intermediate scale respectively; generally, three for multichannel images and RGB images Zhang et al. (2017). This algorithm has better detail retrieval and color retention capabilities, as shown in Figure 17 (c), which is denoted by (9).

$$R_{MSR_i}(x,y) = \sum_{n=1}^{K} \omega_n[log_{10}[I_i(x,y)] - log_{10}[F_n(x,y) * I_i(x,y)]], \tag{9}$$

where: K is the number of scaling parameters, ω_n is the weight and when $\omega_n = 1$, is an SSR algorithm. Wang Wang et al. (2021) mentions that previous researchers have experimentally verified that if K is larger than 3, the effect has no significant improvement in quality, obtaining an exponential increase in the amount of computation, so K is generally taken as 3, in other words, three Gaussian filters at different scales are used to filter the original image, as shown by (10). When the three Gaussian filter outputs are obtained, they are equated and a weighted sum is performed to obtain the final output of the multi-scale Retinex Palacios et al. (2020).

$$F_n(x, y) = K_n e^{-(x^2+y^2/2\sigma_n^2)}. \tag{10}$$

The MSR algorithm contains a good recovery effect on grayscale images and has a higher robustness. However, during processing, the image noise may be amplified, which will cause color distortion and will not be able to show the real color of the object Jobson et al. (1997a).

II.3.6 Multi-Scale Retinex Color Restoration

This Retinex algorithm was developed to be applied in images containing global or regional de-saturations, but it is imperative to be careful not to compromise color constancy in the pursuit of color reproduction, since color constancy is one of the main goals of the Retinex algorithm Jobson et al. (1997a). This color restoration is described by (11),

$$R_{MSRCR_i}(x, y) = G[C_i(x, y)R_{MSR_i}(x, y) + b], \tag{11}$$

where $C_i(x, y)$ is the position of the pixel and is obtained by using (12),

$$C_i(x, y) = f[I_i^{'}(x, y)], \tag{12}$$

where $I_i^{'}$ represents the chromacity coordinates,

$$I^{'}(x,y) = I_i / \sum_{i=1}^{K} I_i(x,y). \tag{13}$$

The function that provides the best overall color restoration is represented by (14),

$$C_i(x,y) = \beta log[\alpha I_i^{'}(x,y)], \tag{14}$$

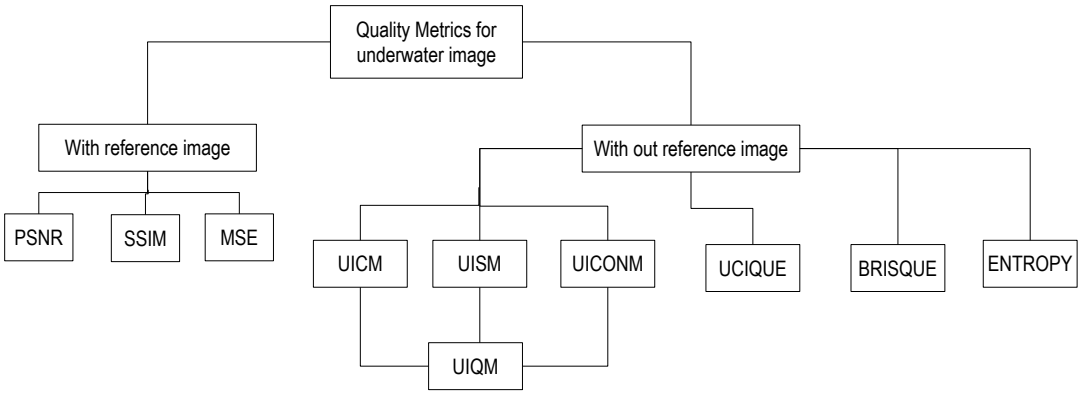
where $C_i(x,y)$ represents color restoration and is obtained by using (15),

$$C_i(x,y) = \beta log_{10}[\alpha I_i^{'}(x,y)] - \beta log_{10}[\sum_{i=1}^{K} I_i(x,y)], \tag{15}$$

where β is a constant gain, α controls the nonlinearity, and G and b are gain and offset values. The values of these constants are specified by the author Zia et al. Jobson et al. (1997b) which are $\beta = 46$, $\alpha = 125$,$b = -30$, $G = 192$.

II.4 Quality Metrics

The quantification of picture distortion is necessary for many areas of image processing, and evaluation methods that quantify the quality of an image are directly tied to the comparison of an original version or under some ideal image criterion presented in the literature. The quality of the image may suffer due to distortions that occur throughout the process of acquiring and processing images. Noise, blurring, ringing, and other artifacts caused by compression can be considered examples of distortion.



Figures 5: Quality metrics.

II.4.1 Quality Metrics with Reference Image

Mean squared error (MSE)

MSE is the most common estimator of image quality measurement metric. It is a full reference metric and the values closer to zero are the better. The MSE is the variance of the estimator in case of unbiased estimator. It has the same units of measurement as the square of the quantity being calculated like as variance. The MSE introduces the Root-Mean-Square Error (RMSE) or Root-Mean-Square Deviation (RMSD) and often referred to as standard deviation of the variance, as it shows Sara et al. (2019).

Mean Squared Error (MSE) between two images such as $g(x, y)$ and $\hat{g}(x, y)$is defined as:

$$MSE = \frac{1}{MN} \sum_{m=0}^{M} \sum_{m=1}^{N} [\hat{g}(n, m) - g(n, m)]^2 \tag{16}$$

Peak signal-to-noise ratio (PSNR)

PSNR is used to calculate the ratio between the maximum possible signal power and the power of the distorting noise which affects the quality of its representation. This ratio between two images is computed in decibel form. The PSNR is usually calculated as the logarithm term of decibel scale because of the signals having a very wide dynamic range. This dynamic range varies between the larg- est and the smallest possible values which are changeable by their quality. The Peak signal-to-noise ratio is the most commonly used quality assessment technique to measure the quality of reconstruction of lossy image compression codecs. The signal is considered as the original data and the noise is the error yielded by the compression or distortion. The PSNR is the approximate estima- tion to human perception of reconstruction quality compared to the compression codecs, an is expressed as:

$$PSNR = 10 \log_{10}(peakval^2)/MSE \tag{17}$$

Here, peakval (Peak Value) is the maximal in the image data. If it is an 8-bit unsigned integer data type, the peakval is 255

Structural Similarity Index Measure(SSIM)

The SSIM index method, a quality measurement metric is calculated based on the computation of three major aspects termed as luminance, contrast and structural or correlation term. This index is a combination of multiplication of these three aspects.

Structural Similarity Index Method can be expressed through these three terms as:

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \tag{18}$$

Here, l is the luminance (used to compare the brightness between two images), c is the contrast (used to differ the ranges between the brightest and dark- est region of two images) and s is the structure (used to compare the local luminance pattern between two images to find the similarity and dissimilarity of the images) and α, β and γ are the positive constants. Again luminance, contrast and structure of an image can be expressed sepa- rately as:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{19}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{20}$$

$$s(x,y) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{21}$$

where μ_x and μ_y are the local means, σ_x and σ_y are the standard deviations and $\sigma_x y$ is the cross-covariance for images x and y sequentially.

II.4.2 Quality Metrics without Reference Image

Underwater Image Quality Measures (UIQM)

The new underwater image quality measure UIQM comprises three attribute measures, namely, the underwater image colorfulness measure (UICM), the underwater image sharpness measure (UISM), and the underwater image contrast measure (UIConM). It has been demonstrated that underwater images can be modeled as a linear superposition of absorbed and scattered components. Besides, it is known that the absorption and

scattering effects cause color, sharpness, and contrast degradation. Therefore, it is reasonable to use the linear superposition model for generating the overall underwater image quality measure as well. The overall underwater image quality measure is then given by

$$UIQM = c_1 \times UICM + c_2 \times UISM + c_3 \times UIConM \tag{22}$$

where the colorfulness, sharpness, and contrast measures are linearly combined together. It is worth noting that the UIQM has three parameters c_1, c_2, and c_3. The selections of these parameters are application dependent.

Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE)

A new model of the statistics of pair-wise products of surrounding (locally adjusted) luminance values is presented by BRISQUE. The naturalness of the image can be quantified even further using the characteristics of this model. Our contention is that defining the locally normalized brightness coefficient in this manner is adequate not only to define naturalness, but also to quantify quality in the presence of distortion, and this is the proposition that we are putting out. Mittal et al. (2012). The BRISQUE algorithm makes use of an NSS model framework consisting of locally normalized luminance coefficients. It then assesses 'naturalness' by making use of the model's parameters. A new model of the statistics of pair-wise products of surrounding (locally adjusted) luminance values is presented by BRISQUE. The naturalness of the image can be quantified even further using the characteristics of this model. Our contention is that describing the locally normalized luminance coefficient in this manner is adequate not just to define naturalness, but also to assess quality in the presence of distortion. This is the argument that underpins our position.

II.5 Embedded Systems

A computer system designed to execute specific functions and whose components are integrated into a motherboard is an embedded system. The term "integrated system" is another name for an embedded system. A microcontroller, also known as a microprocessor that incorporates input/output interfaces and a small memory on the same chip, is responsible for the system's primary processing tasks. A microcontroller is a type of microprocessor.

Programming for these systems can either be done directly in the assembly language of the microcontroller or the microprocessor, or it can be done using other languages such as C or C++, Python, Java, or Cuda, or it can be done using specialized compilers.

In general, they are intended for application in real-time computation activities, and their primary focus is on designing and developing applications and prototypes using embedded systems from graphical environments. However, they can also be used in other contexts. For the sake of this thesis project, many of the models that have received the most attention in the relevant body of research were chosen, and a summary of each model's salient features is provided below.

II.5.1 Beagle Board

BeagleBone AI bridges the gap between more modest single-board computers (SBCs) and more robust industrial computers by relying on the open-source Linux methodology pioneered by BeagleBoard.org. Because it is based on the Texas Instruments AM5729, developers get access to the highly capable SoC while still benefiting from the simplicity of the BeagleBone Black header and mechanical compatibility. BeagleBone AI makes it simple to investigate the practical applications of artificial intelligence (AI) thanks to the TI C66x digital-signal-processor (DSP) cores and embedded-vision-engine (EVE) cores that are supported by an optimized TIDL machine learning OpenCL API along

with pre-installed tools. Dedicated to the automation of routine tasks in many settings, including homes, businesses, and factories.

Figures 6: Image of BeagleBoard.

In the Figure 31 shows the motherboard of the Beagle board embedded system. Table I shows the main characteristics of the embedded system obtained directly from the datasheet published by its manufacturer.

Tables I: Main features of Beagle Board embedded system

CPU	GPU	RAM	Connectity	Storage	Energy consumption	Video output	Operating voltage	Operaing System	Cost [USD]
ARM Cortex-A8 AM3359	Not available	512 MB DDR	Bluetooth, Wifi	2 GB eMMC	1-2.3 Wh	micro-HDMI	5 V	Debian	$40.00

II.5.2 Odroid

The ODROID-XU4 is a new generation of computing device that has hardware that is both more powerful and more energy efficient, and it also has a more compact form factor. The board offers support for open source software and is compatible with multiple distributions of Linux, including the most recent version of Ubuntu (20.04), as

well as three different versions of Android (4.4 KitKat, 5.0 Lollipop, and 7.1 Nougat).

Amazing data transfer speeds are one of the many features that the ODROID-XU4 has to offer thanks to the implementation of eMMC 5.0, USB 3.0, and Gigabit Ethernet interfaces. This is a feature that is becoming increasingly necessary to support enhanced computing capability on ARM devices.

Users are able to experience a genuine improvement in their computing capabilities as a result of this, particularly in terms of speedier boot times, online surfing, networking, and 3D gaming.

Figures 7: Image of ODROID-XU4Q.

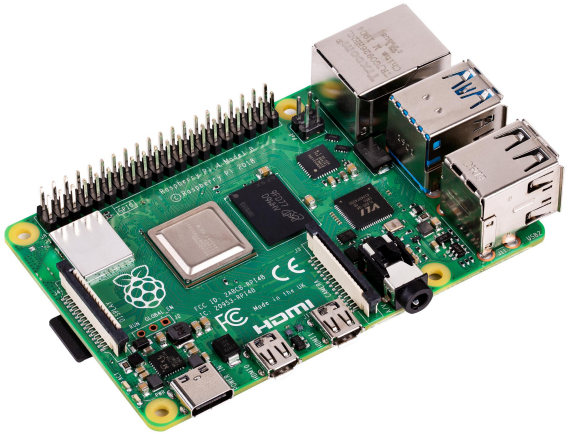
Tables II: Main features of ODROID-XU4 embedded system

CPU	GPU	RAM	Connectity	Storage	Energy consumption	Video output	Operating voltage	Operaing System	Cost [USD]
Exynos5 Octa-core	Power VR SGXX544MP3	2 GB DDR3	Bluetooth, Wifi	32 GB SD-Card	5.2-5.3 Wh	DHDMI	5 V	Ubuntu	$75.00

II.5.3 Raspberry Pi 4

When compared to previous Raspberry Pi models, the speed and performance of the brand-new Raspberry Pi 4 is a significant improvement. We have, to our knowledge, created the very first comprehensive desktop experience. You'll find the experience to

be smooth and very familiar, regardless of whether you're editing documents, browsing the web with a bunch of tabs open, juggling spreadsheets, or drafting a presentation; all of these things can be done on a machine that is smaller, more energy-efficient, and much more cost-effective.



Figures 8: Image of Raspberry pi 4.

Tables III: Main features of Raspberry Pi 4 embedded system

CPU	GPU	RAM	Connectity	Storage	Energy consumption	Video output	Operating voltage	Operaing System	Cost [USD]
Broadcom BCM2711 Quad-core	Video Core VI	4 GB DDR4	Bluetooth, Wifi, Ethernet	32 GB SD-Card	3-7 Wh	HDMI	5 V	Raspbian	$55.00

II.5.4 Jetson Nano

The NVIDIA Jetson Nano Developer Kit is a compact and potent piece of hardware that is suitable for students interested in gaining knowledge regarding artificial intelligence. The Jetson Nano paves the way for robots and brings deep learning to the edge for real-time image classification, object identification, segmentation, audio processing, and a variety of other applications. Students, makers, and developers will find it to be the ideal tool for beginning their first artificial intelligence project and gaining knowledge of popular machine learning frameworks such as PyTorch and TensorFlow.

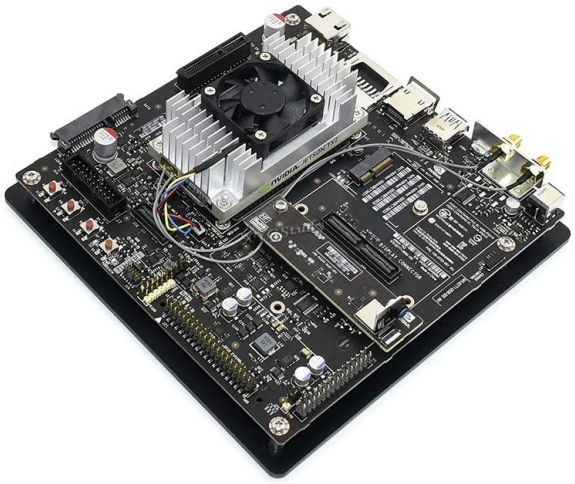
Figures 9: Image of Jetson Nano embedded system.

Tables IV: Main features of Jetson NANO embedded system.

CPU	GPU	RAM	Connectity	Storage	Energy consumption	Video output	Operating voltage	Operaing System	Cost [USD]
MPCore processor Quad-core	128-core Maxwell 1098 MHz	4 GB DDR4	Wifi, Ethernet	32 GB SD-Card	6.5-10 Wh	HDMI	5 V	Ubuntu	108.00

II.5.5 Jetson TX2

NVIDIA Jetson TX2 series modules bring you exceptional speed and power efficiency in an integrated AI computing device. Each supercomputer-on-a-module brings true artificial intelligence to the edge with an NVIDIA Pascal GPU, up to 8GB of memory and 59.7GB/s of memory bandwidth, and a wide range of standard hardware interfaces.



Figures 10: Image of NVIDIA Jetson TX2.

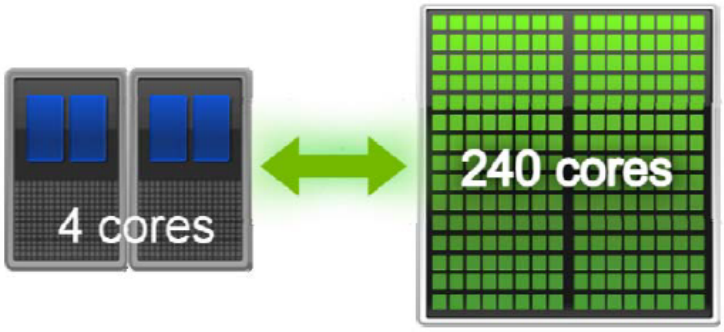
Tables V: Main features of Jetson TX2 embedded system.

CPU	GPU	RAM	Connectity	Storage	Energy consumption	Video output	Operating voltage	Operaing System	Cost [USD]
Denver2, 2.5GHz Quad-core ARM	256-core Pascal GPU	8 GB DDR4	Wifi, Ethernet Bluetooth	32 GB eMMC	7.5-15 Wh	HDMI	19 V	Ubuntu	479.00

At this time, most image enhancement algorithms have been evaluated on various types of cutting-edge computers with a large amount of processing capacity. On the other hand, very few writers have reported the implementation of these algorithms on embedded systems. The current needs do not support large onboard computers for short-range underwater exploration and inspection based on ROVs and AUVs. This is due to several constraints, including, but not limited to, limited space, load, and power consumption. Tang et al. (2019); Zamora-Arellano et al. (2021); Flores-Vergara et al. (2019); Tedesco-Oliveira et al. (2020). These cars utilize high-performance embedded systems to control many of their tasks. Raspberry Pi 4, Jetson Nano, and Jetson are the embedded systems that have been referred to the most in the literature concerning their applicability to various challenges. Tang et al. (2019); Zamora-Arellano et al. (2021); Flores-Vergara et al. (2019); Tedesco-Oliveira et al. (2020).

II.6 Artificial vision using deep learning

The artificial vision is have been applied to such as computer vision and processing digital image. This application is used for the objects detection and submarine species identification as shown by Liu et al. (2019). This models to deep learning have been used in different applications as shown by Cao et al. (2020) for identification and detector for underwater live crabs. Burguera (2020) focused on the detection of Posidonia oceanic in underwater images. The input image is split into a set of patches that are classified as depicting Posidonia or not. Two different Neural Networks are proposed to perform the classification. This models of deep learning have a big data

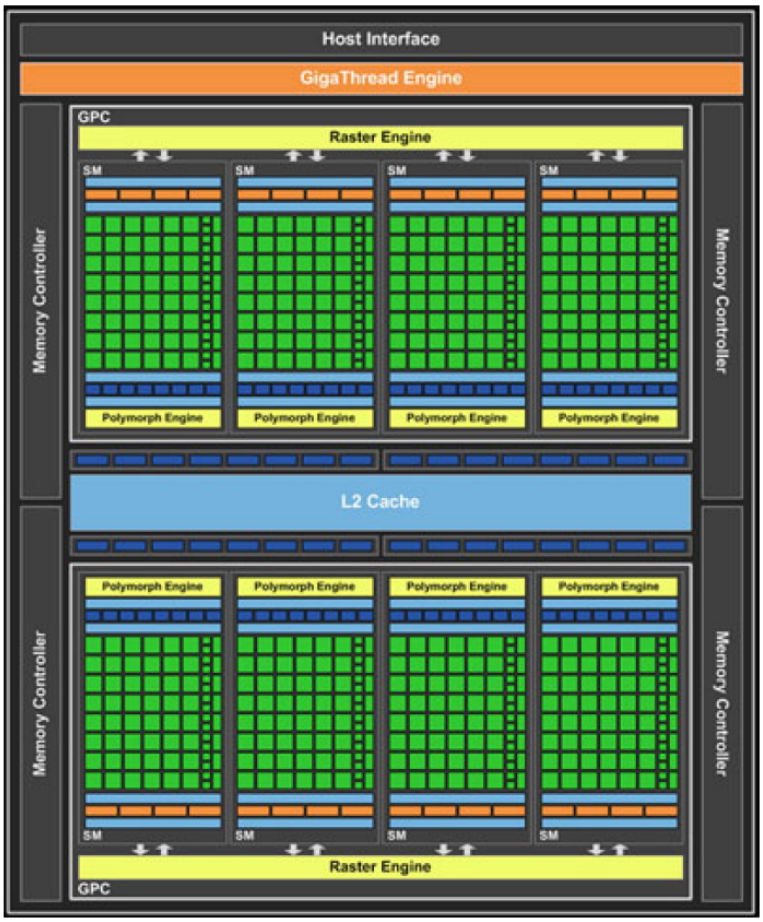


Figures 11: Image of the elements that make up a heterogeneous system

collection for the different trainingś models and features extraction. In this thesis work for the handling this data are use different libraries for the data processing using parallel programming o concurrent programming. In this case using optimized libraries with CUDA Architecture or computing in heterogeneous systems. These heterogeneous computing systems can be defined as computing on graphic cards called, Graphic Processing Unit (GPU). Cheng et al. (2013) mentions that in a heterogeneous computing system consists of using together a Central Processing Unit (CPU), which carries the sequential part of an application commonly called "Host" and the more expensive part of the computation or data handling is executed in the Graphic Processing Unit (GPU) called "Device", as shown in the Figure 11. Although there are different architectures for the use of heterogeneous systems, in this thesis work we will work with CUDA, since some of the embedded systems are optimized to work with this architecture, such as the Jetson Nano and Jetson TX2.

II.6.1 CUDA Architecture

In the classic architecture of a graphics card we can find the presence of two types of processors, vertex processors and fragment processors, dedicated to different and independent tasks within the graphics pipeline, and with different instruction repertoires. This presents two important problems, on the one hand the load imbalance that



Figures 12: Architecture of a CUDA graphics card

appears between both processors and on the other hand the difference between their respective instruction repertoires. CUDA architecture was born out of this need to unify an architecture that would be able to balance the threads the CPU and GPUs.

Figure 12 shows the architecture of a CUDA-compliant graphics card. Here you can see the presence of execution units called Streaming Multiprocessors (SM); each SM is composed of computational cores called "CUDA cores" or Streaming Processors (SP), which are in charge of executing the instructions of the executing the instructions dedicated for more expensive part of the computation. Thanks to this CUDA architecture, using neural networks in digital image processing is one of the tools that today has solved problems of execution times, enanhcement image, object detection and species

recognition, among other applications.

II.6.2 Neuronal Networks

Neural networks have been around for many years, and they've gone through several periods during which they've fallen in and out of favor. But recently, they have steadily gained ground over many other competing machine learning algorithms. This resurgence is due to having computers that are fast, the use of GPU versus the most traditional use of CPUs, better algorithms and neural net design, and increasingly larger datasets. A neural network can describe as a mathematical model for information processing. The characteristics of a neural network is a Information processing occurs in its simplest form, over simple elements called neurons. Neurons are connected and they exchange signals between them through connection links, the connection links between neurons can be stronger or weaker, and this determines how information is processed. Each neuron has an internal state that is determined by all the incoming connections from other neurons. Each neuron has a different activation function that is calculated on its state, and determines its output signal Okatani (2015). The model mathematical the neuron is defined as the Equation 23

$$y = f(\sum_{i} x_i \omega_i + b) \tag{23}$$

First element we compute the weighted sum $\sum x_i \omega_i$ of the inputs x_i and the weights ω_i (also known as an activation value). Here, x_i is either numerical values that represent the input data, or the outputs of other neurons, the weigths ω_i are numerical values that represent either the strength of the inputs or, alternatively, the strength of the connections between the neurons and the weight b is a special value called bias whose input is always 1. Then, we use the result of the weighted sum as an input to the activation function f, which is also known as transfer function. There are many types

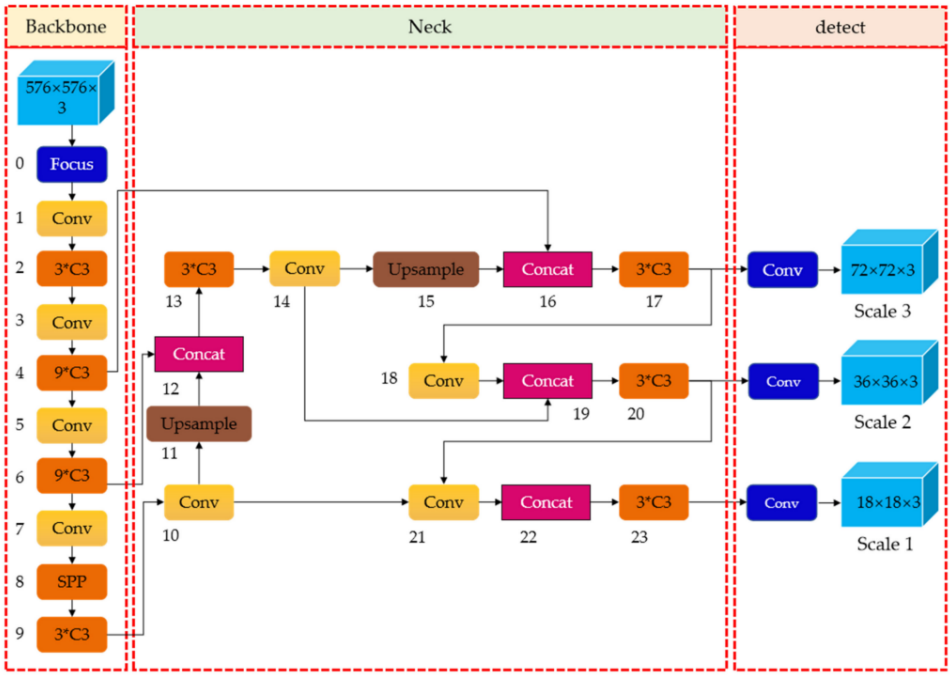
of activation functions, but they all have to satisfy the requirement to be non-linear.

II.6.3 Deep learning algorithms

We could define deep learning as a class of machine learning techniques, where information is processed in hierarchical layers to understand representations and features from data in increasing levels of complexity Okatani (2015). In practice, all deep learning algorithms are neural networks, which share some common basic properties. They all consist of interconnected neurons that are organized in layers. Some of the most common architectures used in a deep learning algorithm are Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), Recurrent Networks and Autoencoders. In this thesis work, the enhancement of underwater images with a CNN architecture called YOLO V5 was elucidated.

II.6.4 YOLO Deep Learning Object Detection Model

Object detection is among the classical computer vision problems to identify which objects are in the image and their corresponding locations. The object detection issue is more complex than the classification problem that consists of recognizing objects but without indicating their locations in the image. Moreover, images containing more than one object cannot be classified. YOLO-v5, a recent update of the YOLO family. YOLO was the first object recognition model to merge bounding box estimation and object identification in one end-to-end differentiable network. Darknet is the environment under which YOLO is written and maintained. In comparison to previous YOLO models, YOLO-v5 is the first YOLO model developed with the PyTorch framework, and it is more lightweight and simple to use compared to previous YOLO variants Couturier et al. (2021).



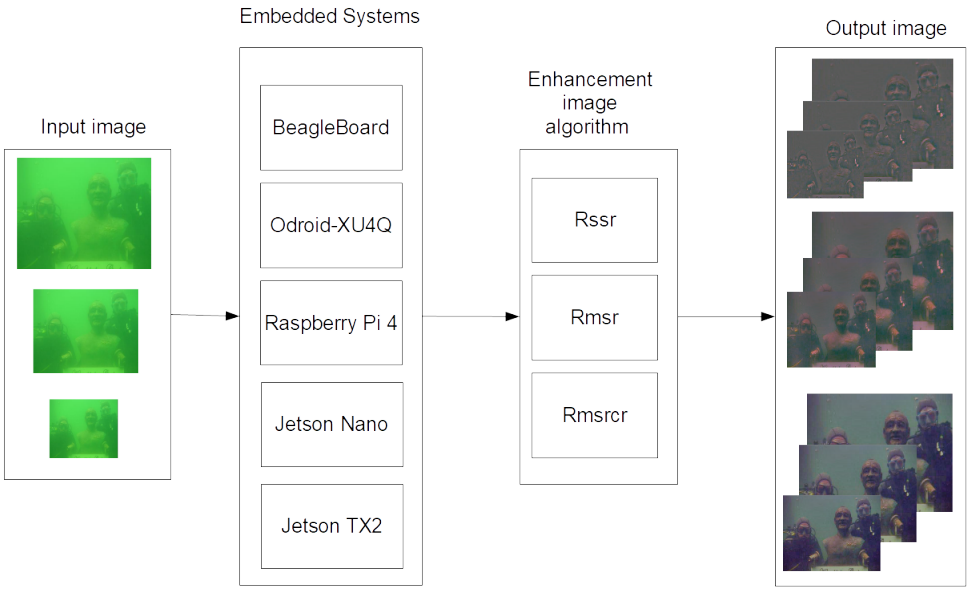
Figures 13: Architecture of YOLO V5 Li et al. (2022)

Figure 13 shows the architecture of the YOLO V5 algorithm, being a convolutional neural network for object detection. YOLO V5 is based on the YOLO detection architecture and uses the excellent algorithm optimization strategy in the field of convolutional neural networks in recent years, such as auto learning bounding box anchors, mosaic data augmentation, the cross-stage partial network, and so on; they are responsible for different functions in different locations of the YOLOv5 architecture. In the architecture, YOLOv5 consists of four main parts: input, backbone, neck, and output.

Chapter III

Methodology

Chapter III of this thesis work describes the methodology utilized in the development and experimentation of this thesis work. Additionally, the various underwater image enhancement algorithms, the quality metrics that were utilized, and the various high-performance embedded systems selected are discussed, in addition to the model used in the experimentation.



Figures 14: Methodology used in the development of the experiment.

Figure 14 illustrates the method that was developed during the course of this thesis work. In this method, the input images are displayed in a variety of sizes in order to locate a benchmark in the processing of the image enhancement algorithms that will be utilized. This benchmark was located by measuring the processing time in

each of the embedded systems that were chosen and by sampling the quality or loss of information of the various image enhancement algorithms that were executed in each of the embedded systems.

III.1 Study and Selection of underwater image enhancement algorithms

According to the research that was done on the algorithms used to enhance underwater images, the majority of underwater photographs have a strong color or light degradation. This investigation demonstrates the application of Retinex models for the purpose of improving these photos.

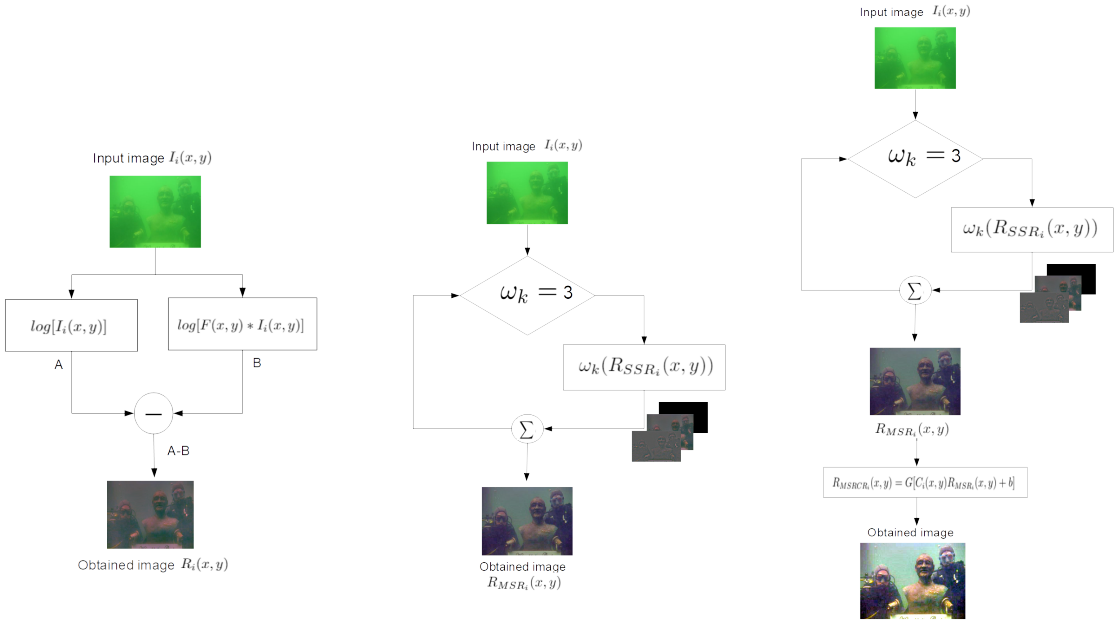
(a) Wrecked ship (b) Fishes (c) Fish

(d) Divers (e) Diver (f) Coral

Figures 15: Additional underwater images for benchmarking the Retinex algorithms implemented on embedded systems, the original images were taken from Li et al. (2020a)

Figure 12 shows some of the images used in the experiments described in the following chapters. These images contain different types of degradations that are presented in underwater images. Figure 16(a) shows the model single-scale Retinex, this model improves the visual quality of the input image, removing the predominant color, and

enhancing the colors by distributing. However, details such as edges and color enhancement are improved by employing MSR, increasing the brightness of the image and making colors sharper, as shown in the Figure 16(b).



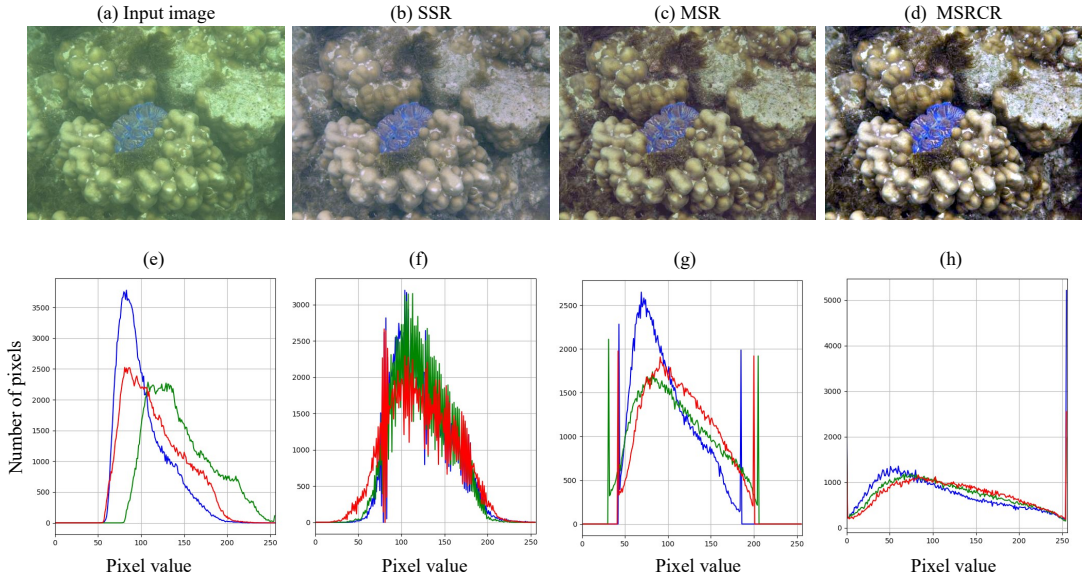
(a) Model Single-Scale Retinex (b) Model Multi-Scale Retinex (c) Model Multi-Scale Retinex Color Restoration

Figures 16: Models for enhancement underwater image.

In the development of this thesis work the use of the different models of underwater image enhancement, a color restoration algorithm was used to highlight the colors of the images, this restoration is given by means of the coefficients shown in the previous chapters in the equation 15, and shown in Figure 16(c).

This color restoration is shown in the histograms of the resulting images, the MSRCR algorithm, and shown in the Figure 17 . Initially, the histogram using the SSR algorithm compresses the dynamic range, fading out the predominant colors, and recovering some original colors. Using the MSR algorithm the histogram values are expanded throughout the dynamic range, increasing the color intensity of the image.

In Aguirre-Castro et al. (2022) shown in the Figure 17(a) shows as input a Coral



Figures 17: Coral.png image of size 375×500 pixels, taken from Li et al. (2020a). (a) input image, (b) image enhanced with SSR algorithm, (c) image enhanced with MSR algorithm, (d) image enhanced with MSRCR algorithm, (e) histogram of the original image, (f) histogram of image enhanced with SSR, (g) histogram of the image enhanced with MSR, (h) histogram of the image enhanced with MSRCR.

image of 375×500 pixels, in which low color definition, low contrast, and blur can be observed. Figure 17(e) shows its corresponding histogram with the distribution of the red, green, and blue channels. Figures 17(b-d) show the original image processed under SSR, MSR and MSRCR image enhancement algorithms respectively. The input image processed under the SSR algorithm and shown in Figure 17(b), loses the predominant green color, however, from its histogram displayed in Figure 17(f) it can be deduced that the algorithm causes that the dynamic range becomes compressed and that the RGB channels have much variation in amplitude, generating that the image presents a noise-like effect and low contrast. On the other hand, Figure 17(c) shows the original image under the MSR algorithm, where a better contrast and a higher luminosity effect derived from the concentration and uniformity in its dynamic range can be observed as shown in the histogram displayed in Figure 17(g). Figure 17(d) shows the image obtained by

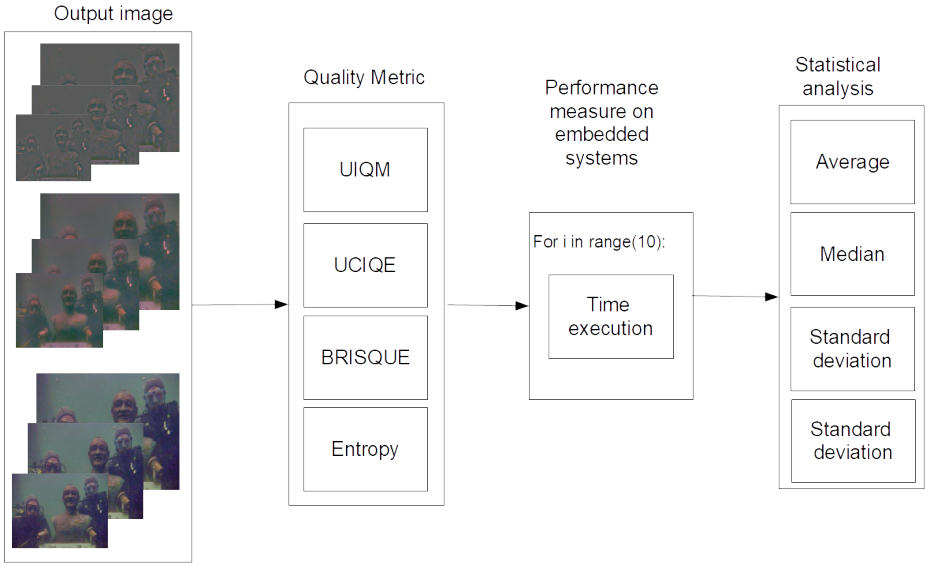
using the MSRCR algorithm, which, compared to the previous ones, exhibits a better contrast, a greater brightness effect and color definition. These effects are the product of the equalization in its dynamic range as evidenced in its histogram displayed in Figure 17(h).

III.2 Study and Selection of Quality Metrics

In the study of the different quality metrics for digital image processing there are several and for different applications, as seen in previous chapters. Some of them use reference images to obtain an estimation of the amount of information lost or modified. In underwater image enhancement the use of metrics such as PSNR, SSIM, and MSE, can provide an approximate estimate of how much the enhanced image has changed, given that the original image has a color degradation or noise. However, the use of these metrics can be used in object recognition and species detection.

In the study of the various underwater image enhancement algorithms presented in the literature, they use non-reference quality metrics. With these metrics, a study of image information such as UIQM is obtained. The UIQM employs the HVS model only and does not require a reference image; hence, it is a better candidate for evaluating underwa- ter images. UIQM is dependent on three attribute measures the underwater images, which are (1) image colorfulness measure (UICM), (2) sharpness measure (UISM), and (3) contrast measure (UIConM), shown in the Figure 5.

Figure 18 shows the methodology used to measure the image quality and performance of the output images obtained by the different embedded systems. These measurements provide information to know if losses are depending on the size of the image. In the next block, the execution time of the underwater image enhancement algorithms based on the Retinex model is obtained, after 10 executions, the third block is followed by a statistical analysis that provides the best execution times of the embedded



Figures 18: Methodology for measuring quality and performance of embedded systems.

systems.

III.3 Selection of Embedded Systems

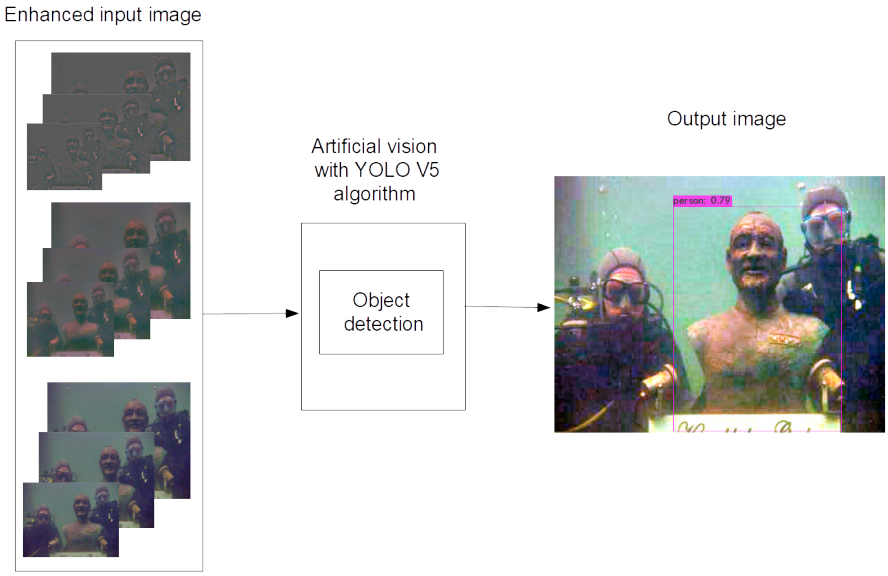
Currently, most image enhancement algorithms have been tested on different models of advanced computers with significant processing power; however, few authors have reported the deployment of these algorithms on embedded systems Tang et al. (2019). Present-day requirements for short-range subsea exploration/inspection based on ROVs and AUVs do not support large onboard computers due to space, load, and power consumption constraints, among others. These vehicles employ high performance embedded systems for the control of many of their functions Aguirre-Castro et al. (2019), being the most referenced in the literature with application to various problems, Raspberry Pi 4, Jetson Nano and Jetson TX2 Tang et al. (2019); Zamora-Arellano et al. (2021); Flores-Vergara et al. (2019); Tedesco-Oliveira et al. (2020). Table VI shows the main characteristics and differences of these high-performance embedded systems.

Tables VI: Comparison of embedded systems with high-performance versus personal computer.

Hardware	Beagle Board	Odroid	Raspberry Pi 4	Jetson Nano	Jetson TX2	Personal Computer (PC)
CPU	ARM Cortex- A8	Exynos5 Octa-core	Broadcom BCM2711	MPCore processor	Denver 2, 2.5 GHz	Intel Core i7-8750
GPU	Not available	PowerVR SGX544MP3 GPU	Video Core VI 500 MHz	128-core Maxwell	256-core NVIDIA	Nvidia GeForce GTX1050ti
RAM	512 MB DDR	2 GB DDR3	4GB	4 GB DDR4	8 GB DDR4	16 GB DDR4
Connectity	Bluetooth, Wifi	Bluetooth, Wifi		Bluetooth, Wifi, Ethernet		
Storage	2GB eMMC	32 GB SD-Card	32 GB SD-Card	32 GB SD-Card	32GB eMMC 5.1	1 TB SSD
Energy consumption	210-460 mA	5.2-5.3 W	3 W - 7 W	6.5 W - 10 W	7.5 W - 15 W	180 W â€“ 200 W
Video output	micro-HDMI	HDMI	mini HDMI	HDMI	HDMI	HDMI
Operating voltage	5 V	5 V	5 V	5 V	19 V	19 V
Operating System	Debian	Ubuntu	Raspbian	Ubuntu	Ubuntu	Ubuntu/Windows
Cost [USD]	$40.00	$75.00	$55.00	$108.00	$479.00	$1,399.00

III.4 YOLO V5 Methodology

The use of deep learning algorithms helps in detecting how good the digital image enhancement can be with the detection of targets with the weights set in the YOLO V5 architecture. The system is tested with the weights and characteristics of the YOLO V5, this model not have a trained system for the detection of divers or underwater images, this model is trained only for people detection This test will be performed with



Figures 19: Person detection methodology applied to divers with YOLO V5.

the image diver.png shown in Figure 15(d). This image will provide the similarity score that a diver has to a person, initially in the original image without any image processing or enhancement, and with the resulting image after processing with the SSR and MSR models.

Figure 19 shows the methodology to be used with the YOLO v5 person recognition algorithm. This algorithm will be applied to the images enhanced with the SSR, MSR, MSRCR algorithms. With this type of procedures we will be able to see which of the images has a considerable improvement in the detection of objects, species and other applications.

Chapter IV

Development of image enhancement models

In this Chapter IV, we will describe the underwater image enhancement models described above, the development elaborated in Python language, which provides us with a series of libraries used for digital image processing. These libraries make the calculations and matrix operations optimized and facilitate the result. Libraries such as NumPy, OpenCV, and pandas provide algorithms necessary for the development of digital image processing. The following lines denote the algorithms of Retinex models.

IV.1 Models to Enhancement Image

In digital image processing, there are different algorithms to obtain a better quality image due to image enhancement, as has been seen previously the study of underwater images is an open topic with great potential for study, finding different models or providing improvements in those used in the literature, this time we provide a study which contains algorithms based on the Retinex model. This model is one of the most used in the literature and probes a gain of information restoring the original color of the objects in scene. The following are the underwater image enhancement algorithms developed in Python language

IV.1.1 Model Simple-Scale Retinex

The first algorithm used is the SSR, shown in the Chapter II Equation 6 and following the methodology of the previous Chapter. In this algorithm using libraries for image

management such as OpenCV, using numpy for the optimization of matrix operations and matplotlib to visualize the results obtained.

```
# Libraries for PDI
import matplotlib.pyplot as plt
import cv2
import numpy as np
import time
# Function of SSR
def SingleScaleRetinex(img, sigma):
    retinex = np.log10(img) - np.log10(cv2.GaussianBlur(img, sigma))
return retinex
# Import image
img = cv2.imread('Coral.png')
img_SSR_SN = SingleScaleRetinex(img)
img_SSR = Normalize(img_SSR_SN)
histogram_SSR = cv2.calcHist([img_SSR], None, [256], [0, 256])
# Show image and Histogram
cv2.imshow("Image SSR",img_SSR)
plt.plot(histogram_SSR, color = c)
plt.show(plt.title('Histogram_SSR'))
```

Listing IV.1: Program SSR Python

Lines 2-6 show the aforementioned libraries, with which the information obtained from the images can be processed. Lines 10-12 show the SSR image enhancement function. In line 16 cv2.imread() is used to read an image found inside a working directory. It is imported for processing by saving it in the variable "img" being an array type data, this is a data structure that can contain more than one value at a time. This img variable is used to execute the "SingleScaleRetinex" function. At the end of this function, the results are normalized to be able to display the processed image stored in the variable "img_SSR" this resulting image is shown in Figure 17(b). Once the resulting image is obtained, its histogram is obtained and displayed using the Matplotlib library, this histogram is shown in Figure 17(f).

IV.1.2 Model Multi-Scale Retinex

The algorithm used in the MSR model and following the Equation 9 of the Chapter II starts with the SSR algorithm, this is due to the limited SSR algorithm since it

compresses the low-scale dynamic range, amplifying the dark parts and weakening the bright parts. However, large-scale SSR can provide more natural images. This scale is the variation of the "*sigma*" value. To combine these advantages and disadvantages of varying this value MSR is used where it proposes a scale variation by summing the outputs provided by the SSR. In the following lines of code shown below, the MSR algorithm is used to improve underwater images.

```
# Libraries for PDI
import matplotlib.pyplot as plt
import cv2
import numpy as np
import time
#   Single-Scale Retinex
def singleScaleRetinex(img, sigma):
  retinex = np.log10(img) - np.log10(cv2.GaussianBlur(img, (0, 0),
   sigma))
return retinex
#   Multi-Scale Retinex
def multiScaleRetinex(img, sigma_list):
  retinex = np.zeros_like(img)
  for sigma in sigma_list:
    retinex += singleScaleRetinex(img, sigma)
    retinex = retinex / len(sigma_list)
return retinex
# Import image
img = cv2.imread('Coral.png')
# Sigma values
sigma_list=[15, 80, 240]
img_MSR_SN = multiScaleRetinex(img,sigma_list)
img_MSR = Normalize(img_MSR_SN)
histogram_MSR = cv2.calcHist([img_MSR], None, [256], [0, 256])
# Show image and Histogram
cv2.imshow("Image MSR",img_MSR)
plt.plot(histogram_MSR, color = c)
plt.show(plt.title('Histogram_MSR'))
```

Listing IV.2: Program MSR Python

In this algorithm for the improvement of underwater images based on MSR, it is shown that, unlike SSR, it takes more advantage of the dynamic range, increasing the contrasts of the image by decompressing the dynamic range, this result is shown in Figure 17(g) that a comparison of Figure 17(f) in which the SSR algorithm is applied.

This improvement can be seen visually in Figure 17(c).

IV.1.3 Model Multi-Scale Retinex Color Restauration

Although the SSR and MSR models improve the visual representation of an image when lighting conditions are not good. Our eyes can see colors correctly in low light, cameras and camcorders do not get it right. The MSRCR algorithm is based on the Retinex filter, which is inspired by the biological mechanisms of the eye to adapt to these conditions. With MSR you get a great improvement of the image since it takes better advantage of the dynamic range by increasing the contrast however in very dark images the colors are not well defined, as the MSR algorithm tends to make the image lighter, the use of color restoration has this slider and allows you to adjust the color saturation contamination around the new average color. A higher value means less color saturation. This is the parameter you want to modify for optimal results because its effect is highly dependent on the image. In Chapter II in Equation 11 the behavior of the MSRCR algorithm is described, this algorithm probes an improvement in the color saturation adjustment, eliminating the multi-scale aspect that leaves the result of the MSR algorithm.

```
# Libraries for PDI
import matplotlib.pyplot as plt
import cv2
import numpy as np
import time

#    Single-Scale Retinex
def singleScaleRetinex(img, sigma):

#   Multi-Scale Retinex
def multiScaleRetinex(img, sigma_list):

#   Multi-Scale Retinex Color Restoration
def MSRCR(img, sigma_list, G, b, alpha, beta, low_clip, high_clip):
  img_retinex = multiScaleRetinex(img, sigma_list)
  img_color = colorRestoration(img, alpha, beta)
  img_msrcr = G * (img_retinex * img_color + b)
return img_msrcr
```

```
# Constants values
G=float(192.0)
alpha= float(125.0)
b= float(-30.0)
beta=float(46.0)
high_clip=float(0.99)
low_clip=float(0.01)
sigma_list=[15, 80, 250]

img_MSRCR_SN =  MSRCR(img, sigma_list, G, b, alpha, beta, low_clip,
    high_clip)
img_MSRCR = Normalize(img_MSRCR_SN)
histogram_MSRCR = cv2.calcHist([img_MSRCR], None, [256], [0, 256])
# Show image and Histogram
cv2.imshow("Image MSRCR",img_MSRCR)
plt.plot(histogram_MSRCR, color = c)
plt.show(plt.title('Histogram_MSRCR'))
```

Listing IV.3: Program MSRCR Python

In the Python code of the MSRCR algorithm, we start with the same previous functions the SSR and the MSR. In this code some of the operations to be performed are shown, the result of this MSRCR image enhancement algorithm is shown in Figure 17(d), this image shows a visual improvement concerning the result of the other algorithms. Figure 17(h) shows the histogram of the resulting image where a uniform distribution is seen in comparison with the others.

IV.2 Quality Metrics

The quality metrics used were chosen after conducting a literature review; this will provide the information needed to compare the resulting images in image enhancement algorithms that are visually equal. These quality metrics provide an analysis of color, brightness and contrast in the images, obtaining metrics on the loss of information in small images and being able to make a statistical comparison with larger images.

```
def metricsimg(img,img_ssr,img_msr,img_msrcr):
  #Peak Signal-to-Noise Ratio
  PSNR0 = cv2.PSNR(img_ssr,img)
  PSNR1 = cv2.PSNR(img_msr,img)
  PSNR2 = cv2.PSNR(img_msrcr,img)
```

```
#Mean Squared Error
MSE0 = mse(img,img_ssr)
MSE1 = mse(img,img_msr)
MSE2 = mse(img,img_msrcr)

#The Structural Similarity Index Measure
SSIM0 = getMSSISM(img,img_ssr)
SSIM1 = getMSSISM(img,img_msr)
SSIM2 = getMSSISM(img,img_msrcr)

#Blind/Referenceless Image Spatial Quality Evaluator
BRQ0 = brisque.score(img_ssr)
BRQ1 = brisque.score(img_msr)
BRQ2 = brisque.score(img_msrcr)

#Metrics for underwater image quality evaluation
uiqm,uciqe,uicm,uism,uiconm = nmetrics(img_ssr)
uiqm1,uciqe1,uicm1,uism1,uiconm1 = nmetrics(img_msr)
uiqm2,uciqe2,uicm2,uism2,uiconm2 = nmetrics(img_msrcr)

#Entropy Metric
entim1= entropy(img_ssr,3)
entim2= entropy(img_msr,3)
entim3= entropy(img_msrcr,3)

  #SAVE QUALITY METRICS

  data = {'PSNR':[PSNR0,PSNR1,PSNR2],
    'MSE':[MSE0,MSE1,MSE2],
    'SSIM':[SSIM0,SSIM1,SSIM2],
    'BRISQUE':[BRQ0,BRQ1,BRQ2],
    'UICM':[uism,uism1,uism2],
    'UISM':[uism,uism1,uism2],
    'UICONM':[uiconm,uiconm1,uiconm2],
    'UIQM':[uiqm,uiqm1,uiqm2],
    'UCIQUE':[uciqe,uciqe1,uciqe2],
    'ENTROPY':[entim1,entim2,entim3]}

  return data
```

Listing IV.4: Program from Qualitys Metrics Python

This quality measures algorithm imports of both the original and improved photos. These are utilized to acquire the various metrics used in the study; the sizes of the photos must be the same to prevent mistakes in the matrix computations. These are used to calculate the PSNR is the first step in determining these metrics. An OpenCV library is utilized for this calculation since the PSNR is a standard measure for digital

image processing. The MSE, or the mean square error, is a metric used to evaluate a picture's overall quality and is indicated and defined in Equation 16 above. The SSIM is a metric which considers three aspects of the image, luminance, contrast and structure described by Padmavathy and Priya (2020), and denoted in Equation 18. The following metric BRISQUE makes an analysis of the image characteristics described by Mittal et al. (2012), being a metric without reference image. In addition to completing an examination of the values within each pixel, the underwater picture quality evaluation metrics specify particular features, such as contrast, brightness, color, and structure. These metrics have been discussed in earlier chapters, and various writers cite them throughout the scholarly literature Panetta et al. (2016); Yang and Sowmya (2015); Xiang et al. (2021); Zhang et al. (2021a).

These measures are then recorded in a variable before being exported to an Excel file. Within the Excel file, tables containing all of the photos' metrics have been enhanced using methods based on Retinex. The outcomes of these algorithms are described in the Chapter that comes after this one (Chapter I V), which also contains the outcomes of the algorithms covered in Chapter IV.

Chapter V

Experimental results

In this Chapter V, the findings acquired by following the technique described in the previous chapter are shown here. Additionally, the results that were obtained by utilizing several underwater image improvement algorithms such as SSR, MSR, and MSRCR are presented in this chapter. These photographs were used to compare performance and test if there is a loss of information when applying the image enhancement method. The images used in this work come in various sizes, which was necessary to complete this study. The following table details the results that were achieved.

V.1 Algorithms enhancement image results

The images used in this thesis work are of different sizes in order to have a better comparison of image quality as well as processing or execution time in the different high-performance embedded systems. The images shown in Figure 15 show some of the images used that were used to have a result with a greater number of test images, thus obtaining a greater number of elements in obtaining image quality metrics. Figure V.1 shows three results of images of different sizes, this was done to see if any of them could have a considerable visual difference or if the Retinex image enhancement algorithm had an image size restriction. As can be seen, the images shown have a relatively similar visual appearance. To check whether the image quality is improved by its size or degraded, a series of quality metrics mentioned above are implemented to get a better insight into the displayed MSR algorithm. In the Table VII you can see the differences in the quality metrics of the images obtained using the same MSR image enhancement

algorithm.

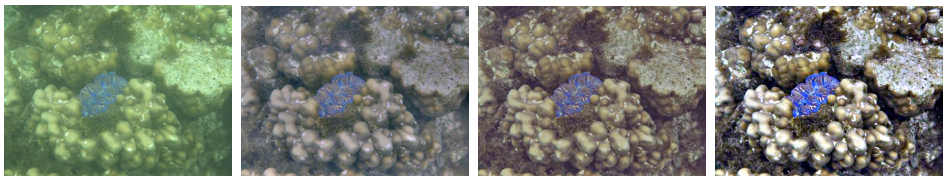
(a) MSR on Diver image 186×138 (b) MSR on Diver image 940×703 (c) MSR on Diver image 1092×809

Figures 20: Image Diver.png of different sizes, with enhancement MSR on Jetson TX2 embedded system.

These results shown in Figure V.1, Figure 21, Figure 22, Figure 23 and Figure 24 using a single input image in these figures shown by the medium scale *coral.png* image, it can be shown at a glance that the images are the same or similar since the results of the operations given by the different embedded systems are very similar or practically identical. Given this, we proceed to obtain a more efficient comparison by applying a series of image quality metrics to observe their behavior and whether some embedded systems may have relevant information losses. In this study was also implemented to see if the loss of information between small or large images would have a great relevance, these images are used to test the quality and performance of embedded systems.

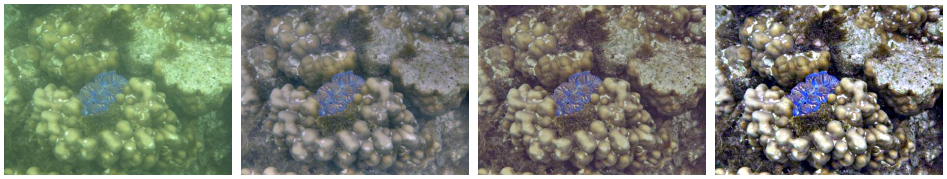
V.2 Quality metrics results

The above results visually identify that processing images with the MSRCR algorithm yields the best result, however, since visual perception is relative, it is required to use different metrics to classify and quantify the quality in images, although some of them employ a reference image, but others do not. Some reference image metrics such as MSE, PSNR, SSIM Padmavathy and Priya (2020); Li et al. (2016) and referenceless metrics



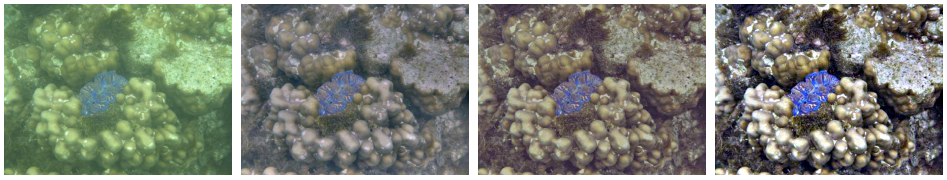
(a) Original image (b) Model SSR (c) Model MSR (d) Model MSRCR

Figures 21: Models for enhancement underwater image on BeagleBone .



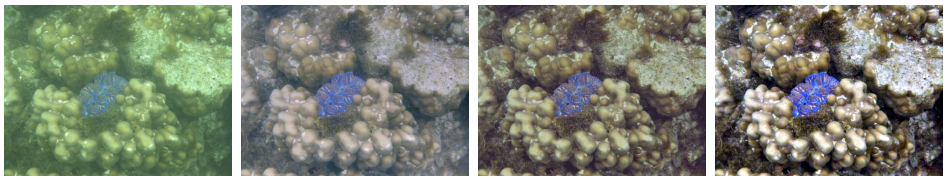
(a) Original image (b) Model SSR (c) Model MSR (d) Model MSRCR

Figures 22: Models for enhancement underwater image on Odroid-XU4.



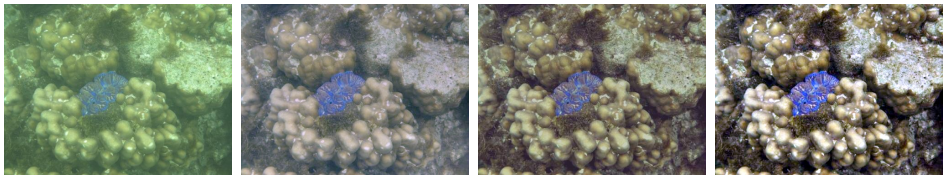
(a) Original image (b) Model SSR (c) Model MSR (d) Model MSRCR

Figures 23: Models for enhancement underwater image on RaspBerry Pi 4 .



(a) Original image (b) Model SSR (c) Model MSR (d) Model MSRCR

Figures 24: Models for enhancement underwater image on Jetson NANO.



(a) Original image (b) Model SSR (c) Model MSR (d) Model MSRCR

Figures 25: Models for enhancement underwater image on Jetson TX2.

such as BRISQUE Mittal et al. (2012) or Entropy Li et al. (2019, 2016) are employed to quantify the quality of low-light images, although, on the other hand, metrics such as UICM, UISM, UICONM, UCIQE Panetta et al. (2016); Yang and Sowmya (2015);

Xiang et al. (2021); Zhang et al. (2021a) are employed to quantify quality for underwater images. This quality metrics show us how good the improvement is in some of them by using the original image as a reference to show how much it has changed or how similar it is to the same original image. Table VII displays the quality metrics for the images depicted in Figure V.1; this table describes the quality metrics for the same image of various sizes. In it is demonstrated that there are only minor differences between the images after the processing to algorithm MSR due to the matrix operations in the original image.

Tables VII: Results of Diver.png image with MSR on Jetson TX2

Quality Metrics	Diver.png image		
	Image size		
	138X186	703X940	809X1092
PSNR	13.3662889	13.9925048	13.4612316
MSE	8986.13059	7779.49023	8791.81333
BRISQUE	19.9956931	48.0747978	51.0154382
UICM	11.0264967	18.6316031	20.4205815
UISM	11.0264967	18.6316031	20.4205815
UICONM	-0.24848228	-0.09977996	-0.08681065
UIQM	2.71453716	5.53663028	6.17277206
UCIQUE	3.06070022	2.39757693	2.2595618

Tables V.2 show respectively the results of these assessments, these being very similar between the embedded systems and the PC. Additionally, the different experimental results show that the embedded systems do not generate loss of information in the images (neither has the image's quality degraded) regarding those generated on a PC with robust processing. Analyzing Tables V.2 and V.2 corroborate that the quality metrics of the underwater images of different sizes are similar in the different hardware tested. Furthermore, respect the Jetson TX2, can be observed that the MSRCR model obtained better results in the UIQM, UCIQUE, BRISQUE, and Entropy metrics, which are the referenceless metrics.

In this section it was shown that the images of different sizes and these algorithms

Tables VIII: Quality metrics of the Coral.png image shown in Figure 17(a) of size 375×500 pixels.

Algorithm	Quality metrics	Hardware					
		Beagle Board	Odroid	Raspberry Pi 4	Jetson Nano	Jetson TX2	PC
SSR	PSNR	20.753527	20.755273	20.757278	20.687329	20.773366	20.757278
	MSE	1634.236875	1638.603435	1638.603435	1665.208875	1632.544277	1638.603435
	SSIM	0.118171	0.163464	0.163194	0.167025	0.159057	0.163455
	BRISQUE	14.723666	14.814764	14.804712	15.003332	14.716483	14.804712
	UICM	8.138513	8.153518	8.158520	8.022891	7.530429	8.158520
	UISM	8.082566	8.157523	8.158520	8.022891	7.530429	8.158520
	UICONM	-0.227306	-0.226125	-0.227060	-0.227412	-0.225989	-0.227060
	UIQM	2.024119	2.075839	2.079850	2.041170	1.897229	2.079850
	UCIQUE	0.464095	0.466712	0.461688	0.465095	0.466808	0.461688
	ENTROPY	4.437088	4.436879	4.437487	4.451084	4.468893	4.437487
MSR	PSNR	20.487951	20.487951	20.487952	20.487952	20.483494	20.487952
	MSE	1743.437722	1743.437723	1743.437723	1743.437723	1745.228299	1743.437723
	SSIM	0.304823	0.304971	0.304971	0.304971	0.311433	0.304971
	BRISQUE	15.387715	15.389765	15.389766	15.389766	15.387706	15.389766
	UICM	9.767856	9.799285	9.799286	9.799286	7.658658	9.799286
	UISM	9.798285	9.799928	9.799286	9.799286	7.658658	9.799286
	UICONM	-0.237789	-0.237837	-0.237838	-0.237838	-0.237752	-0.237838
	UIQM	2.327195	2.327195	2.327195	2.327195	1.690642	2.327195
	UCIQUE	0.459048	0.459048	0.459048	0.459048	0.489734	0.459048
	ENTROPY	4.559365	4.559365	4.559365	4.559365	4.574188	4.559365
MSRCR	PSNR	16.028347	16.028347	16.028347	16.028347	16.092639	16.028347
	MSE	4868.183765	4868.183765	4868.183765	4868.183765	4796.646715	4868.183765
	SSIM	0.745142	0.745241	0.745242	0.745242	0.738272	0.745242
	BRISQUE	15.688138	15.688138	15.688138	15.688138	16.074704	15.688138
	UICM	6.435618	6.435618	6.435618	6.435618	6.620804	6.435618
	UISM	6.435618	6.435618	6.435618	6.435618	6.620804	6.435618
	UICONM	-0.196600	-0.196600	-0.196600	-0.196600	-0.201581	-0.196600
	UIQM	1.574567	1.574567	1.574567	1.574567	1.608241	1.574567
	UCIQUE	0.695743	0.695743	0.695743	0.695743	0.697495	0.695743
	ENTROPY	4.959972	4.959972	4.959972	4.959972	4.979988	4.959972

tested in different embedded systems, their quality in the output images do not have a variety of great consideration, since the resulting images were of good quality concerning the original image in all embedded systems and PC, The next variable to consider is how much the execution time varies in the embedded systems with respect to a PC, these results will provide if these algorithms can have a real-time application for underwater exploration inside ROVs and AUVs, the results of this benchmark are shown in the following sections.

Tables IX: Quality metrics of the Coral.png image shown in Figure 17(a) of size 147×196 pixels.

Algorithm	Quality Metrics	Hardware					
		Beagle Board	Odroid	Raspberry Pi 4	Jetson Nano	Jetson TX2	PC
SSR	PSNR	20.75727755	20.77336647	20.83773719	20.83856697	20.8250476	20.8377372
	MSE	1638.603435	1632.544277	1608.525302	1608.217999	1613.23209	1608.5253
	BRISQUE	14.80471178	14.71648296	18.13086674	18.82786795	17.6279976	18.1308667
	UICM	8.158519813	7.530428607	7.255937517	7.597779132	7.10906827	7.25593752
	UISM	8.158519813	7.530428607	7.255937517	7.597779132	7.10906827	7.25593752
	UICONM	-0.227060189	-0.225989019	-0.321279078	-0.32161498	-0.32259153	-0.32127908
	UIQM	2.079849903	1.897229378	1.453742148	1.560392543	1.4061551	1.45374215
	UCIQUE	2.425484645	2.421304677	2.715433848	2.713501252	2.67213797	2.71543385
MSR	PSNR	20.48795177	20.48349369	20.2533375	20.2533375	20.2140508	20.2533375
	MSE	1743.437723	1745.228299	1840.21203	1840.21203	1856.9343	1840.21203
	BRISQUE	15.38976581	15.38770577	21.57373475	21.57373475	22.5431403	21.5737347
	UICM	9.799285545	7.658657837	8.932868697	8.932868697	7.48569881	8.9328687
	UISM	9.799285545	7.658657837	8.932868697	8.932868697	7.48569881	8.9328687
	UICONM	-0.237837797	-0.23775178	-0.304840351	-0.304840351	-0.30349693	-0.30484035
	UIQM	2.327195151	1.690641647	1.601747466	1.601747466	1.1810536	1.60174747
	UCIQUE	2.16577331	2.158002684	2.496653905	2.496653905	2.43551838	2.49665391
MSRCR	PSNR	16.02834652	16.09263883	16.52906248	16.52906248	16.5703678	16.5290625
	MSE	4868.183765	4796.646715	4338.058135	4338.058135	4296.99483	4338.05814
	BRISQUE	15.68813762	16.07470372	22.11810333	22.11810333	24.0077052	22.1181033
	UICM	6.43561794	6.620804339	6.88610758	6.88610758	6.99561125	6.88610758
	UISM	6.43561794	6.620804339	6.88610758	6.88610758	6.99561125	6.88610758
	UICONM	-0.196600453	-0.201580696	-0.225153506	-0.225153506	-0.23004996	-0.22515351
	UIQM	1.574566512	1.608240534	1.436051291	1.436051291	1.42310795	1.43605129
	UCIQUE	4.117604697	4.070516352	3.710334772	3.710334772	3.62545899	3.71033477

V.3 Benchmark of embedded systems

In this section, the results of the processing time of embedded systems are discussed, obtaining a point of comparison with respect to a personal computer. In Tables X-XI, the results of the ten tests that were carried out on each embedded system are displayed, together with the average execution time, median execution time, standard deviation of the ten processing times, and variance of the ten processing times. After making the discovery that the embedded systems did not have a significant impact on the execution of the picture enhancement algorithms, we came to the following conclusion: These studies are carried out using the five underwater photos depicted in Figures V.1 and 15, and their purpose is to ascertain the typical amount of time that an embedded system requires to process an image by employing images of varying sizes. The results of the ten experiments executed on each embedded system are shown in Tables , together with the average execution time, median, standard deviation, and variance of the ten processing times. Observing that the embedded systems did not have a considerable difference in the execution of the image enhancement algorithms. These experiments are conducted with the five underwater images shown in Figures V.1-15 to determine the average execution time of the embedded systems by using images of different sizes.

Table X shows the performance results of the embedded systems by processing an underwater image of 375×500 pixels from Figure 17(a). It is evident that the PC is in the first place presenting the shortest execution times with the SSR, MSRCR and MSR algorithms, with average times of 1,060, 1,619 and 1,643 respectively. Regarding embedded systems, the Jetson Nano has the best average time (1.6279 seconds) when the SSR algorithm is implemented. However, the Jetson TX2 has the best average time when the MSRCR and MSR algorithms are implemented, with values of 8.3375 seconds and 8.4103 seconds respectively. Raspberry Pi 4 has averaging times of 1.8287,

14.7071, and 15.0038 with the SSR, MSRCR, and MSR algorithms respectively. This is followed by the Odroid-XU4 with average times greater than 2.79 seconds and finally the Beagle board with times greater than 34 seconds, this device being the slowest of the embedded systems.

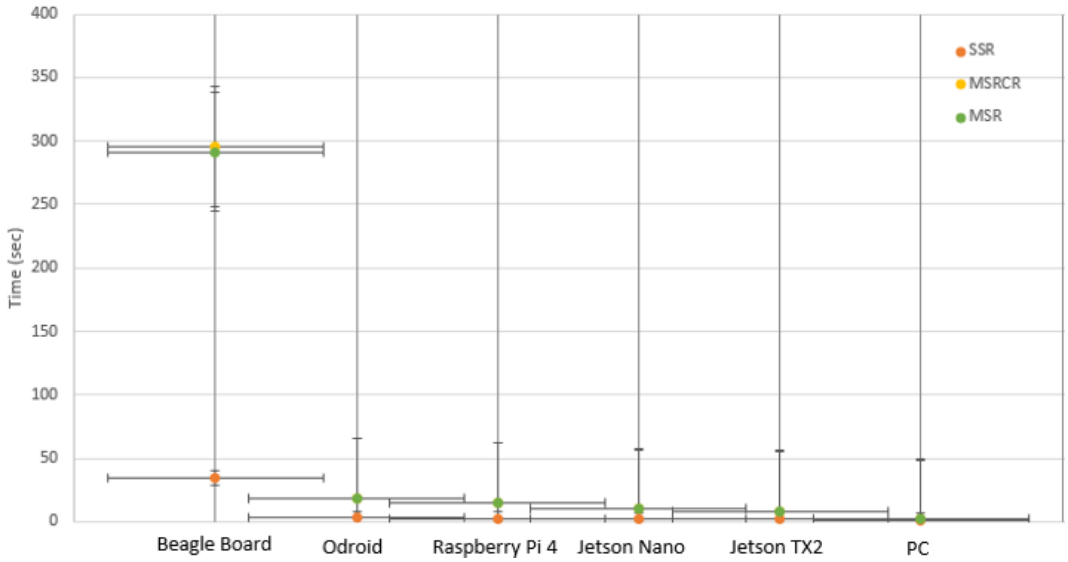
Figures 26 and 27 depict a performance comparison of the Beagle Board, Odroid-XU4, Raspberry Pi 4, Jetson Nano and Jetson TX2 embedded systems, as well as a PC when processing the SSR, MSR and MSRCR algorithms, using as input the original Coral.png image depicted in Figure 17(a) of 375×500 and 147×196 pixels respectively. Figures 26 and 27 shows the average execution time in seconds obtained from the 10 experiments of Tables X and XI respectively.

Tables X: Performance results of the embedded systems by processing an underwater image of 375×500 pixels from Figure 17(a).

Experiment	Beagle Board			Odroid-XU4			Raspberry Pi 4		
	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)
Run #1	291.042936	295.250216	34.6502587	18.2496523	17.2305201	2.68952312	15.0301521	14.679297	1.80478811
Run #2	291.093658	295.195874	33.3596052	18.5890252	17.9632054	2.89631153	14.9825261	15.2939839	1.79388213
Run #3	291.0569687	293.256305	36.2558021	18.2045638	17.2025123	2.78623614	15.4341819	15.404484	1.78590488
Run #4	292.369412	294.502451	35.0250587	16.2596325	18.1025325	2.78056126	14.4861012	14.391314	1.78093386
Run #5	290.596345	296.365025	33.3650252	17.3690215	19.0025368	3.03652089	15.6682382	14.4678428	1.96179891
Run #6	291.036425	294.258014	35.1095801	18.2055089	18.0253652	2.75026369	14.6604319	14.5160811	1.78358793
Run #7	290.996541	296.358021	34.8052304	19.3652051	17.9852364	2.69354205	15.466754	15.1158299	1.86452913
Run #8	290.4536985	295.195202	34.6802141	19.2103025	18.0215042	2.90251823	14.765239	14.5905209	1.785357
Run #9	290.258025	295.452026	35.1902051	18.9836502	18.0325695	2.76859522	14.186692	14.0626609	1.79563403
Run #10	292.366512	296.254058	34.6093252	18.2468025	17.9985605	2.63205305	15.3577192	14.5498118	1.93097806
Average	291.1270521	295.208719	34.7050305	18.2683364	17.9564543	2.79361252	15.0038035	14.7071826	1.8287394
Median	291.0396805	295.223045	34.7427222	18.2482274	18.0100324	2.77457824	15.0063391	14.5701663	1.79475808
Standard deviation	0.715567448	1.00147198	0.8521192	0.91635885	0.49760059	0.12068872	0.48050083	0.42761552	0.06696931
Variance	0.512036772	1.00294612	0.72610713	0.83971353	0.24760634	0.01456577	0.23088105	0.18285504	0.00448489

Experiment	Jetson Nano			Jetson TX2			PC		
	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)
Run #1	9.8623178	9.30115819	1.73680186	7.97462201	7.83880997	1.78580403	1.61472344	1.64181638	1.02601719
Run #2	9.77333212	9.335454941	1.58006811	8.45683503	8.55268097	1.82544398	1.66808963	1.7339561	1.13046885
Run #3	9.68043995	9.945575953	1.59277797	8.64562917	8.55706501	1.90012598	1.6923337	1.65063596	1.08518863
Run #4	10.3228209	10.99919796	1.57111216	8.55375218	8.90598297	1.92607784	1.60207582	1.67330003	1.05632973
Run #5	9.68583393	9.805238009	1.57411194	8.20692801	8.41194105	1.94449019	1.6231122	1.60749745	1.06057525
Run #6	9.98035908	10.66847587	1.7334621	8.39587116	7.94574499	1.77406597	1.60483265	1.55103898	1.06815267
Run #7	11.0370612	10.88404799	1.64543605	8.31183314	8.29025793	1.93327594	1.63978791	1.57450986	1.03220224
Run #8	9.97905111	10.82937884	1.60813594	8.66434216	8.15114999	1.8631289	1.68646622	1.59353614	1.05570412
Run #9	9.95843697	9.937731981	1.61404705	8.43233609	8.40344191	2.09076405	1.67576003	1.55924606	1.10006046
Run #10	10.3721018	9.54602313	1.62305403	8.46111894	8.31815505	1.96317887	1.62290335	1.60615706	0.98811722
Average	10.0651755	10.12522829	1.62790072	8.41032679	8.33752298	1.90063558	1.64300849	1.6191694	1.06028163
Median	9.96874404	9.941653967	1.61109149	8.44458556	8.36079848	1.91310191	1.63145006	1.60682726	1.05845249
Standard deviation	0.41491584	0.661539614	0.06106095	0.20731319	0.31038872	0.09425931	0.03458749	0.05656736	0.03999691
Variance	0.17215515	0.43763466	0.00372844	0.04297876	0.09634116	0.00888482	0.00119629	0.00319987	0.00159975

The amount of time required by the SSR, MSR, and MSRCR algorithms to complete a processing cycle is shown in Figure 26. The SSR algorithm is basically the same on



Figures 26: Performance of the embedded systems by processing an underwater image of 375×500 pixels from Figure 17.

all of the hardware that was tested, although the performance of the algorithm on the PC was somewhat better by a fraction of a second as compared to the other devices. On the other hand, the execution time of the MSR and MSRCR algorithms remains almost the same on each embedded system and on the PC despite the varying hardware, with the exception of the Beagle board, which is slower. This is the case for all of the embedded systems. On the other hand, there are certain minor distinctions that set this embedded system apart from the others. When compared to the other embedded systems, the Jetson Nano is able to complete the SSR method for the enhancement of an underwater image with 375$times$500 pixels in the shortest amount of time. When it comes to the execution of the MSR and MSRCR algorithms, it is clear that the Beagle board is the piece of hardware that is the slowest, lasting up to approximately 300 seconds.

Table XI shows the performance results of the embedded systems by processing an underwater image of 147×196 pixels from Figure 17(a). Again, here it is observed that

the PC obtains the smallest times. Regarding embedded systems, the Raspberry Pi 4 has the best average time of 0.1574 with the SSR algorithm, then followed by the Jetson Nano with an average time of 0.1674 seconds, and the Jetson TX2 with 0.3323 seconds. Regarding the execution of the MSRCR algorithm, the Jetson TX2 obtained the lowest processing time of 0.6297, followed by Jetson Nano and Raspberry Pi 4 respectively. Similarly, regarding the execution of the MSR algorithm, the Jetson TX2 obtained the best performance with an average time of 0.6295, followed by the Jetson Nano with an average time of 0.7992 and Raspberry Pi 4 with 0.8315 seconds. In addition, it can be seen that the Odroid-XU4 and Beagle board are the slowest devices to execute the SSR, MSR and MSRCR algorithms. Regarding the detail that the time of the SSR algorithm is higher on the PC versus the other algorithms (MSR and MSRCR), it is because the operating system (OS) of the PC is multitasking and sometimes other applications that are running in the background demand CPU time. However, on embedded systems, the CPU runs fewer processes which did not have important effects during the execution of the Retinex algorithms.

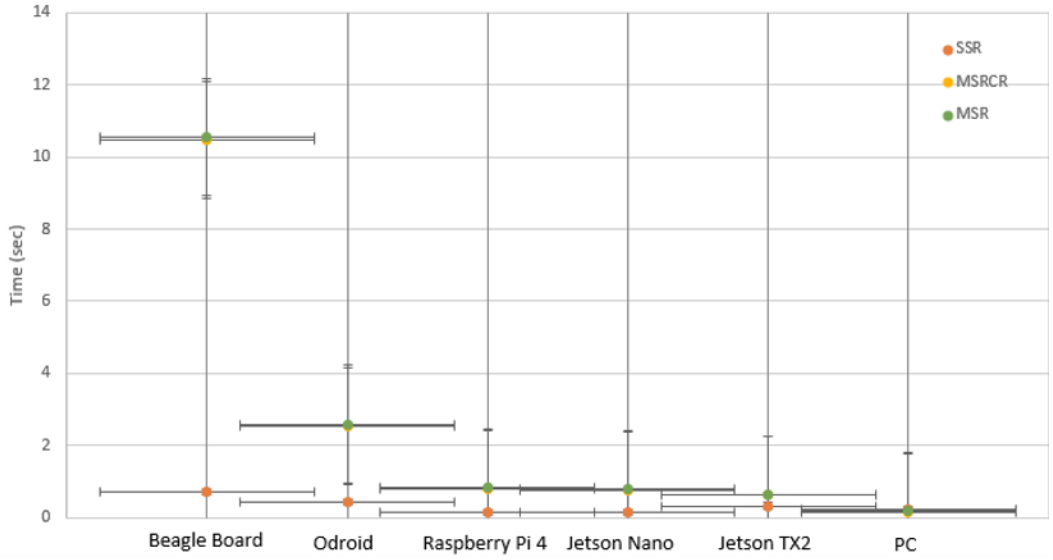
As for Figure 17(a) of 147×196 pixels size, and considering the MSRCR algorithm with the best results in quality levels, the average execution time is again achieved by the Jetson TX2 system compared to the embedded systems, although it is 0.4616 seconds slower than the PC, as one can see in Figure 27. The results shown in both figures (Fig. 26 and Fig. 27) are consistent with each other. Moreover, the reduction in image size does not degrade image quality, and it also improves processing times.

Table XII shows the processing time of each embedded system to run the algorithms (SSR, MSR and MSRCR) with underwater images of several sizes. As can be seen in this Table, embedded systems such as the Raspberry Pi 4, Odroid-XU4, Jetson Nano, and Jetson TX2 are comparable with the PC when running small images of 320×240 resolution. However, the Jetson TX2 and Jetson Nano achieved processing times between

Tables XI: Performance results of the embedded systems by processing an underwater image of 147×196 pixels from Figure 17(a).

Experiment	Beagle Board			Odroid-XU4			Raspberry Pi 4		
	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)
Run #1	10.5417902	10.4679586	0.69896554	2.59821258	2.52392154	0.43685215	0.84834909	0.79629183	0.15628099
Run #2	10.5398763	10.4645214	0.70251254	2.58145282	2.51598475	0.43125436	0.82643104	0.79833293	0.16988897
Run #3	10.5352146	10.4625865	0.70922538	2.59225476	2.51425368	0.44525635	0.82199717	0.79737306	0.15448093
Run #4	10.5417256	10.4687589	0.70982541	2.60525865	2.55263584	0.43565126	0.82389212	0.78408599	0.15447998
Run #5	10.5425652	10.4725424	0.71366969	2.59985878	2.52386525	0.43254896	0.83448195	0.78306389	0.15439796
Run #6	10.5428145	10.4836521	0.71095237	2.61258685	2.52996523	0.43678546	0.82039785	0.79719496	0.15575004
Run #7	10.5535687	10.4614528	0.70912536	2.60365285	2.51995254	0.43726162	0.82767296	0.791713	0.15450597
Run #8	10.5425874	10.4652365	0.68325874	2.65463215	2.51885452	0.42998584	0.82630515	0.79577088	0.15484595
Run #9	10.5435893	10.4625419	0.70785896	2.56352487	2.51258456	0.43985872	0.82632995	0.79398394	0.15465713
Run #10	10.5425415	10.4785214	0.72036589	2.55358213	2.52398541	0.43896524	0.85967994	0.82500005	0.16550493
Average	10.5426273	10.4687772	0.70657599	2.59650164	2.52360033	0.436442	0.83155372	0.79628105	0.15747929
Median	10.5425534	10.4665976	0.70917537	2.59903568	2.5219089	0.43681881	0.82638049	0.79603136	0.15475154
Standard deviation	0.00453036	0.00739788	0.01001933	0.02779682	0.01148152	0.00448536	0.01270329	0.01145443	0.00551823
Variance	2.0524E-05	5.4729E-05	0.00010039	0.00077266	0.00013183	2.0118E-05	0.00016137	0.0001312	3.0451E-05

Experiment	Jetson Nano			Jetson TX2			PC		
	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)	T_{MSR} (sec)	T_{MSRCR} (sec)	T_{SSR} (sec)
Run #1	0.84492397	0.78538299	0.16999602	0.64984107	0.618855	0.33645701	0.17902684	0.16656876	0.20777464
Run #2	0.73674417	0.7727499	0.16510701	0.64549899	0.62479901	0.37665105	0.19012403	0.16455507	0.23842144
Run #3	0.75866199	0.76325512	0.16577697	0.59853101	0.63925099	0.33147311	0.1775279	0.1760776	0.20242596
Run #4	0.79090905	0.78715301	0.17126703	0.63270497	0.62172198	0.32629108	0.16555882	0.1615994	0.24186015
Run #5	0.80360389	0.81674218	0.17101407	0.66169715	0.64604378	0.33013201	0.16555882	0.1615994	0.24186015
Run #6	0.82382393	0.76074314	0.16701412	0.63531494	0.61655402	0.32628298	0.18618131	0.17455149	0.21841359
Run #7	0.78035307	0.77465296	0.1665051	0.6050148	0.65839911	0.34235096	0.16308427	0.16171241	0.23613405
Run #8	0.82763195	0.7817421	0.165838	0.58231306	0.63455892	0.31761909	0.1686306	0.16636395	0.24569798
Run #9	0.82879305	0.74731088	0.16739917	0.64345193	0.61869311	0.31830215	0.18839073	0.17996287	0.25857139
Run #10	0.79747891	0.76547289	0.1644938	0.6411221	0.61863899	0.31821609	0.17104745	0.16805601	0.20944095
Average	0.7992924	0.77552052	0.16744113	0.629549	0.62975149	0.33237755	0.17551308	0.1681047	0.23006003
Median	0.8005414	0.77370143	0.16675961	0.63821852	0.6232605	0.32821155	0.17428768	0.16646636	0.23727775
Standard deviation	0.03394673	0.01898682	0.00246002	0.0255286	0.01424842	0.01755239	0.01016252	0.00657635	0.01904323
Variance	0.00115238	0.0003605	6.0517E-06	0.00065171	0.00020302	0.00030809	0.00010328	4.3248E-05	0.00036264



Figures 27: Performance of the embedded systems by processing an underwater image of 147×196 pixels from Figure 17.

Tables XII: Processing time of the embedded systems using different images size.

Hardware	SSR	MSR	MSRCR	SSR	MSR	MSRCR	SSR	MSR	MSRCR
	1280×960 pixels			640×480 pixels			320×240 pixels		
Odroid-XU4	16.4114	118.3579	117.0881	2.8689	17.0789	16.8952	0.5563	2.7533	2.7093
Raspberry Pi 4	15.5508	99.1016	123.3246	2.7837	18.2468	17.9911	0.4368	2.5982	2.5239
Jetson Nano	10.6466	79.8163	79.1382	2.3189	14.6238	13.9837	0.40248	2.6347	2.6254
JetsonTX2	4.9881	40.2617	38.9944	2.7923	16.1186	15.8298	0.3044	0.5268	0.5377
Beagle board	120.3052	362.2038	357.3328	34.6942	291.0526	295.2033	0.7923	10.5428	10.4681
PC	2.9626	17.4351	17.1753	1.8083	3.8934	3.5463	0.37589	0.8163	0.7824

2.31 and 16.11 seconds when processing images of 640×480 pixels using the SSR, MSR, and MSRCR algorithms. Regarding the high-resolution images of 1280×960 pixels, the PC takes between 2.96 and 17.43 seconds to execute the Retinex algorithms, however, the Jetson TX2 takes between 4.98 and 40.26 seconds.

V.4 Deep Learning results

The performance and quality results shown in the previous section show how functional embedded systems are in underwater image enhancement. When looking for a quality metric, which would indicate a potential improvement of the different image enhancement models used, the use of deep learning was proposed. Figure V.4 shows the results of YOLO v5 with an original image with a predominantly green color and an image with the result of the SSR and MSR models. In these, it can be seen how the percentage of similarity to a person improves and the better the image enhancement model is, the better the quality of similarity of the deep learning model.

The YOLO V5 model is not trained for the detection of divers or statues, the weights with which the neural network is trained are the detection of people, regarding this, Figure V.4a) shows that it only detects with a percentage of 41, not detecting the divers that are next to the statue, Figure V.4b) shows the result of the original image improved with the SSR algorithm, this image detects the divers that are next to the statue improving its probability of being a person, Figure V.4c) shows an improvement

(a) Model YOLO V5 application on the Divers original image (b) Model YOLO V5 application on the Divers image with SSR (c) Model YOLO V5 application on the Divers image with MSR

Figures 28: YOLO V5 results with image Divers.png

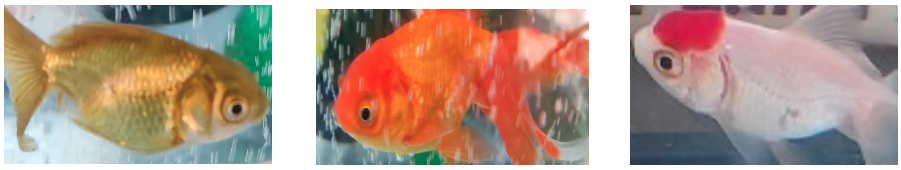
in the original image using MSR in this image shows an improvement in the prediction of people. Although the objective was not to have a metric that would show which image has better quality, having a better prediction denotes that the image has better visual quality.

V.4.1 Ornamental fish detection using YOLO V5

The classification of ornamental fish provides valuable information for researchers to carry out various tasks or applications, one of which is the identification of different species on real underwater environments, as well as the optimal ecological study for the growth and development of the various species Shah et al. (2019).

The data-set was collected using a digital camera (VEMONT) equipped with a 2× wide-angle lens, 12 Mega pixels of resolution, and 1080P video recording of upto 30 frames per second. In video camera mode, a resolution of 640×480 pixels was employed, and because it is an action camera, it provides useful information on how to work with underwater shots. This is the size that is used throughout the entire image collection. By capturing a variety of images, it is possible to monitor the behavior of various species, in their natural environment. Researchers can collect images of the fish from various angles, allowing them to extract traits for use in several applications, thus giving the aquaculturist a better solution to the different problems that can affect their

environment.



(a) Common Goldfish (b) Oranda Goldfish (c) Oranda RedHat Goldfish

Figures 29: Example of Goldfish found in the proposed dataset. (a) Goldfish Common, (b) Goldfish Oranda RedHat, and (c) Goldfish Oranda.

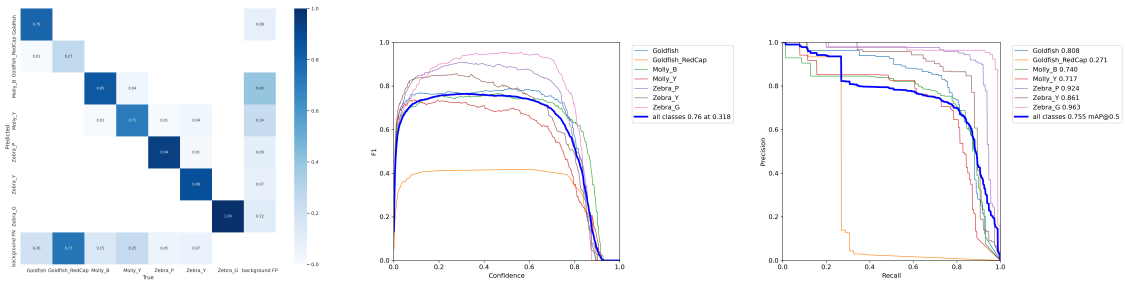
(a) Yellow Molly (b) Black Molly (c) Bicolor Molly

Figures 30: Example of Molly fish with different colors. (a) Yellow Molly, (b) Black Molly, and (c) Bicolor Molly.

(a) Green Zebra (b) Pink Zebra (c) Yellow Zebra

Figures 31: Example of Zebra fish with different colors. (a) Green Zebra, (b) Pink Zebra, and (c) Yellow Zebra.

This dataset contains three ornamental fish species. It has 820 images in PNG format with a resolution of 640×480 pixels. Different classes and colors of the three fish species can be obtained from this dataset; for example, in the goldfish section are: Common Goldfish, Oranda RedHat goldfish, and Oranda Goldfish, as shown in Figure 29; this section of the dataset contains 307 images. Molly's section contains 264 images of different colors of the same species as shown in Figure 30, including black, yellow,



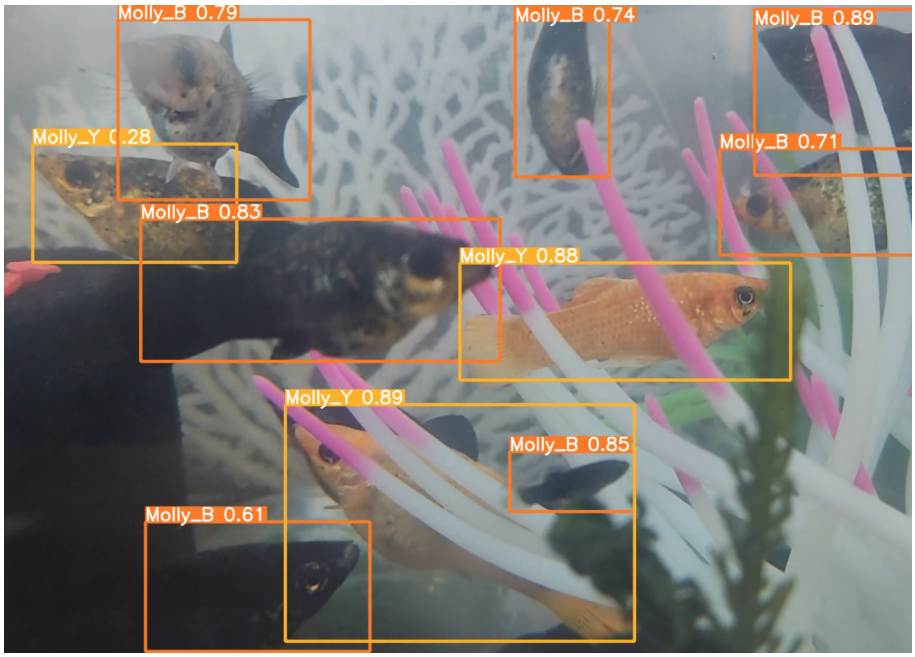
(a) Matrix confusion (b) F score (c) Precision and recall

Figures 32: Deep learning metrics to visualize the performance of a supervised learning algorithm.

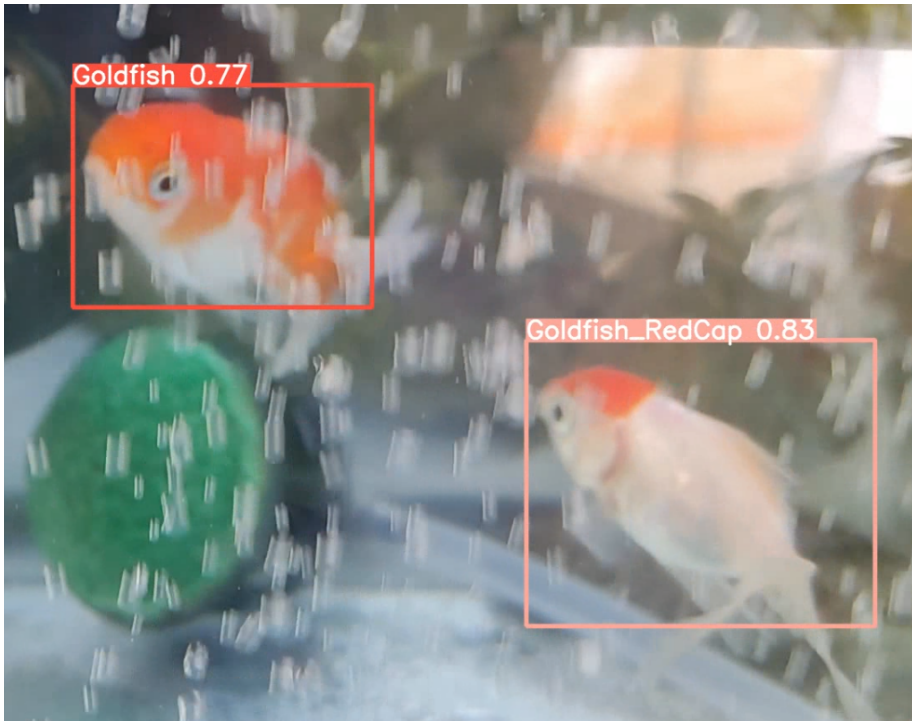
Figures 33: Result of the trained model for Zebra-fish detection.

and bicolor species. There are different colors of fish in the Zebra section, as shown in Figure 31, with 249 images captured in real scenarios. The images in the data set are in a real environment that is conducive to reproduction and growth.

The results are shown in the Figures 33,34, and 35, in which the identification of each one is observed, distinguishing the colors and shapes. The trained model for the detection of ornamental fish has precision metrics in the detection of species, these metrics are described in Figure 32, in this Figure 32(a) the confusion matrix is shown, and in it, the false positives and false are displayed. negative, as well as the detection



Figures 34: Result of the trained model for Molly-fish detection.



Figures 35: Result of the trained model for Gold-fish detection.

probability of the different species to be detected. Figure 32(b) shows the F-score, which is a measure of the precision of a test. It is calculated from the precision and recall of the test, this is presented in Figure 32(c), where precision is the number of true positive results divided by the number of all positive results, including those that were not correctly identified, and recall is the number of true positive results. true positives divided by the number of all samples that should have been identified as positive.n

Figure 33 shows the trained model is result for identifying zebra, pink, green, and yellow fish. This model has an accuracy of 95 percent. In which the green zebra-fish are the ones that can best identify, this precision displayed in the confusion matrix shown in Figure 32(a). Figure 34 shows the identification of the Molly species, this species labeled in two colors, yellow and black, for this species the trained model obtains an average accuracy of 85 percent. In Figure 35, the result of different species of goldfish is obtained, which contains a low precision due to the similarity with other species detected. Although the detection of ornamental fish is low in some species, the use of these detection techniques with the improvement of underwater images will have a better performance in the detection of species.

V.5 Conclusions

In this thesis, a comparative benchmarking for underwater image improvement models based on the Retinex model was demonstrated. The Single Scale Retinex (SSR), multi-scale Retinex (MSR), and multi-scale Retinex with Color Restoration (MSRCR) models were implemented on the Raspberry Pi 4, Jetson Nano, and Jetson TX2 embedded systems, achieving a performance highly comparable to those obtained with a high-performance PC. The MSRCR algorithm was found to achieve the best visual results on underwater images when quality metrics are considered. According to quality metrics without a reference image, such as UIQM, UCIQUE, BRISQUE and ENTROPY.

The MSRCR algorithm achieves the best quality results when is implemented on Jetson TX2 embedded system, and it has a difference of 0.46 seconds in the processing of 147×196 pixels images concerning a high-performance PC. Particularly, the MSRCR algorithm implemented on the Jetson TX2 system was the closest to the execution times obtained with a PC, having an average processing time of approximately 8.33 s for 375×500 pixels images and 0.6297 s for 147×196 pixels images. Similarly, for this type of small image, as in the case of the Raspberri Pi 4 and Jetson Nanosystems, the processing time is approximately 0.7962 s and 0.7755 s, respectively. Since these embedded systems have low power consumption, are small in size, are portable, and image processing that require real-time identification., they have a lot of potential for use in unmanned short-range underwater exploration vehicles. This opens up many research opportunities. In addition, the monitoring of fuel, hydrocarbon, or communication pipelines, as well as in high-tech crop farms, are just a few examples of the applications of this technology. Implementing these algorithms on high-performance embedded systems offers an excellent cost-benefit ratio compared to a conventional personal computer. When image quality metrics, precision, accuracy, power consumption, price, lightweight, size, portability, and reliability are considered, the embedded system implementation offers significant advantages.

V.6 Future work

As part of future work, deep learning or machine learning model should be used to optimize the amount of time spent on image improvement. In a similar vein, an identification of the objects should be carried out with increased accuracy. When it comes to the process of extracting the features that need to be categorized, having the ability to use an underwater picture enhancement model will provide additional information for the different models of deep learning.

Bibliography

Aguirre-Castro, O., García-Guerrero, E., López-Bonilla, O., Tlelo-Cuautle, E., López-Mancilla, D., Cárdenas-Valdez, J., Olguín-Tiznado, J., and Inzunza-González, E. (2022). Evaluation of underwater image enhancement algorithms based on Retinex and its implementation on embedded systems. *Neurocomputing*, 494:148–159.

Aguirre-Castro, O. A., Inzunza-González, E., García-Guerrero, E. E., Tlelo-Cuautle, E., López-Bonilla, O. R., Olguín-Tiznado, J. E., and Cárdenas-Valdez, J. R. (2019). Design and construction of an ROV for underwater exploration. *Sensors*, 19(24).

Ancuti, C., Ancuti, C. O., Haber, T., and Bekaert, P. (2012). Enhancing underwater images and videos by fusion. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 81–88.

Ancuti, C. O., Ancuti, C., De Vleeschouwer, C., and Bekaert, P. (2018). Color Balance and Fusion for Underwater Image Enhancement. *IEEE Transactions on Image Processing*, 27(1):379–393.

Anwar, S. and Li, C. (2020). Diving deeper into underwater image enhancement: A survey. *Signal Processing: Image Communication*, 89(August):115978.

Burguera, A. (2020). Segmentation through patch classification: A neural network approach to detect Posidonia oceanica in underwater images. *Ecological Informatics*, 56(January):101053.

Caizhen, Z., Binlong, K., Ying, L., and Yuan, C. (2021). Underwater image enhancement based on differential channel gain and improved retinex. *Laser and Optoelectronics Progress*, 58(14).

Cao, S., Zhao, D., Liu, X., and Sun, Y. (2020). Real-time robust detector for underwater live crabs based on deep learning. *Computers and Electronics in Agriculture*, 172(February):105339.

Cheng, J., Grossman, M., and McKercher, T. (2013). *Professional CUDA C Programming*, volume 53.

Chun Zhou, J., Huan Zhang, D., and Shi Zhang, W. (2020). Classical and state-of-the-art approaches for underwater image defogging: a comprehensive survey. *Frontiers of Information Technology and Electronic Engineering*, 21(12):1745–1769.

Couturier, R., Noura, H. N., Salman, O., and Sider, A. (2021). A Deep Learning Object Detection Method for an Efficient Clusters Initialization.

Drews, P. L. J., Nascimento, E. R., Botelho, S. S. C., and Montenegro Campos, M. F. (2016). Underwater Depth Estimation and Image Restoration Based on Single Images. *IEEE Computer Graphics and Applications*, 36(2):24–35.

Flores-Vergara, A., Inzunza-González, E., García-Guerrero, E. E., López-Bonilla, O. R., Rodríguez-Orozco, E., Hernández-Ontiveros, J. M., Cárdenas-Valdez, J. R., and Tlelo-Cuautle, E. (2019). Implementing a chaotic cryptosystem by performing parallel computing on embedded systems with multiprocessors. *Entropy*, 21(3).

Gonzalez, R. C. and Woods, R. E. (2008). *Digital image processing*. Prentice Hall, Upper Saddle River, N.J.

Hassan, N., Ullah, S., Bhatti, N., Mahmood, H., and Zia, M. (2021). The retinex based improved underwater image enhancement. *Multimedia Tools and Applications*, 80(2):1839–1857.

He, Y., Wang, D. B., and Ali, Z. A. (2020). A review of different designs and control models of remotely operated underwater vehicle. *Measurement and Control*, 53(9-10):1561–1570.

Hmue, P. M. and Pumrin, S. (2019). Image Enhancement and Quality Assessment Methods in Turbid Water: A Review Article. *2019 IEEE International Conference on Consumer Electronics - Asia, ICCE-Asia 2019*, pages 59–63.

Hou, G., Pan, Z., Wang, G., Yang, H., and Duan, J. (2019). An efficient nonlocal variational method with application to underwater image restoration. *Neurocomputing*, 369:106–121.

Hou, G., Zhao, X., Pan, Z., Yang, H., Tan, L., and Li, J. (2020). Benchmarking Underwater Image Enhancement and Restoration, and Beyond. *IEEE Access*, 8:122078–122091.

Hu, K., Zhang, Y., Weng, C., Wang, P., Deng, Z., and Liu, Y. (2021). An Underwater Image Enhancement Algorithm Based on Generative Adversarial Network and Natural Image Quality Evaluation Index. *Journal of Marine Science and Engineering*, 9(7).

Hu, X. (2021). Underwater image enhancement method based on wavelet transform and Retinex. pages 86–90.

Iqbal, K., Salam, R., Osman, A., and Talib, A. (2007). Underwater image enhancement using an integrated colour model.

Jian, M., Liu, X., Luo, H., Lu, X., Yu, H., and Dong, J. (2021). Underwater image processing and analysis: A review. *Signal Processing: Image Communication*, 91:116088.

Jin, L. and Liang, H. (2017). Deep learning for underwater image recognition in small sample size situations. In *OCEANS 2017 - Aberdeen*, pages 1–4.

Jobson, D. J., Rahman, Z., and Woodell, G. A. (1997a). A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976.

Jobson, D. J., Rahman, Z., and Woodell, G. A. (1997b). Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6(3):451–462.

Li, C., Guo, C., Ren, W., Cong, R., Hou, J., Kwong, S., and Tao, D. (2020a). An Underwater Image Enhancement Benchmark Dataset and Beyond. *IEEE Transactions on Image Processing*, 29:4376–4389.

Li, C., Guo, J., Cong, R., Pang, Y., and Wang, B. (2016). Underwater Image Enhancement by Dehazing With Minimum Information Loss and Histogram Distribution Prior. *IEEE Transactions on Image Processing*, 25(12):5664–5677.

Li, C., Quo, J., Pang, Y., Chen, S., and Wang, J. (2016). Single underwater image restoration by blue-green channels dehazing and red channel correction. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016-May(October):1731–1735.

Li, C., Tang, S., Kwan, H. K., Yan, J., and Zhou, T. (2020b). Color Correction Based on CFA and Enhancement Based on Retinex with Dense Pixels for Underwater Images. *IEEE Access*, 8:155732–155741.

Li, H., Zhuang, P., Wei, W., and Li, J. (2019). Underwater image enhancement based on dehazing and color correction. *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, (2):1365–1370.

Li, Z., Tian, X., Liu, X., Liu, Y., and Shi, X. (2022). A Two-Stage Industrial Defect Detection Framework Based on Improved-YOLOv5 and Optimized-Inception-ResnetV2 Models. *Applied Sciences (Switzerland)*, 12(2).

Liang, Z., Wang, Y., Ding, X., Mi, Z., and Fu, X. (2021). Single underwater image enhancement by attenuation map guided color correction and detail preserved dehazing. *Neurocomputing*, 425:160–172.

Liu, P., Wang, G., Qi, H., Zhang, C., Zheng, H., and Yu, Z. (2019). Underwater Image Enhancement with a Deep Residual Framework. *IEEE Access*, 7:94614–94629.

Liu, X., Gao, Z., and Chen, B. M. (2020). IPMGAN: Integrating physical model and generative adversarial network for underwater image enhancement. *Neurocomputing*, pages 538–551.

Mittal, A., Moorthy, A. K., and Bovik, A. C. (2012). No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708.

Muniraj, M. and Dhandapani, V. (2021). Underwater image enhancement by combining color constancy and dehazing based on depth estimation. *Neurocomputing*, 460:211–230.

Okatani, T. (2015). *On Deep Learning*, volume 33.

Oladi, M., Ghazilou, A., Rouzbehani, S., Zarei Polgardani, N., Kor, K., and Ershadifar, H. (2022). Photographic application of the Coral Health Chart in turbid environments: The efficiency of image enhancement and restoration methods. *Journal of Experimental Marine Biology and Ecology*, 547.

Padmavathy, V. and Priya, R. (2020). Low light image contrast enhancement using advanced perfusion technique. *Materials Today: Proceedings*.

Palacios, J. A., Caro, V., Durán, M., and Figueroa, M. (2020). A hardware architecture for multiscale retinex with chromacity preservation on an FPGA. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 73–80.

Panetta, K., Gao, C., and Agaian, S. (2016). Human-visual-system-inspired underwater image quality measures. *IEEE Journal of Oceanic Engineering*, 41(3):541–551.

Parthasarathy, S. and Sankaran, P. (2012). An automated multi scale retinex with color restoration for image enhancement. In *2012 National Conference on Communications (NCC)*, pages 1–5.

Prasath, R. . and Kumanan, T. (2020). Application of Different Techniques for Underwater Image Processing- A Systematic Review. *IOP Conference Series: Materials Science and Engineering*, 925:012034.

Rizzi, A., Gatta, C., and Marini, D. (2003). A new algorithm for unsupervised global and local color correction. *Pattern Recognition Letters*, 24(11):1663–1677. Colour

Image Processing and Analysis. First European Conference on Colour in Graphics, Imaging, and Vision (CGIV 2002).

Sahu, P., Gupta, N., and Sharma, N. (2014). A Survey on Underwater Image Enhancement Techniques. *International Journal of Computer Applications*, 87(13):19–23.

Sara, U., Akter, M., and Uddin, M. S. (2019). Image Quality Assessment through FSIM, SSIM, MSE and PSNRâ€"A Comparative Study. *Journal of Computer and Communications*, 07(03):8–18.

Shah, S. Z. H., Rauf, H. T., IkramUllah, M., Khalid, M. S., Farooq, M., Fatima, M., and Bukhari, S. A. C. (2019). Fish-Pak: Fish species dataset from Pakistan for visual features based classification. *Data in Brief*, 27:104565.

Sun, Z., Li, F., Chen, W., and Wu, M. (2021). Underwater image processing method based on red channel prior and Retinex algorithm. *Optical Engineering*, 60(9).

Tang, C., von Lukas, U. F., Vahl, M., Wang, S., Wang, Y., and Tan, M. (2019). Efficient underwater image and video enhancement based on Retinex. *Signal, Image and Video Processing*, 13(5):1011–1018.

Tang, C., Wang, Y., Wang, S., Wang, R., and Tan, M. (2018). Floating Autonomous Manipulation of the Underwater Biomimetic Vehicle-Manipulator System: Methodology and Verification. *IEEE Transactions on Industrial Electronics*, 65(6):4861–4870.

Tang, Z., Jiang, L., and Luo, Z. (2021). A new underwater image enhancement algorithm based on adaptive feedback and Retinex algorithm. *Multimedia Tools and Applications*, 80(18):28487–28499.

Tedesco-Oliveira, D., Pereira da Silva, R., Maldonado, W., and Zerbato, C. (2020). Convolutional neural networks in predicting cotton yield from images of commercial fields. *Computers and Electronics in Agriculture*, 171:105307.

Thyagarajan, S. K. (2006). *Digital Image Processing with Application to Digital Cinema*. Elsevier Inc.

Tolstonogov, A. Y., Dzyaman, M. A., Sebto, A. Y., Filonov, I. V., and Chemezov, I. A. (2019). The Compact ROV with Variable Center of Gravity and its Control. In *2019 IEEE Underwater Technology (UT)*, pages 1–7.

Wang, F., Zhang, B., Zhang, C., Yan, W., Zhao, Z., and Wang, M. (2021). Low-light image joint enhancement optimization algorithm based on frame accumulation and multi-scale retinex. *Ad Hoc Networks*, 113:102398.

Wang, W., Wu, X., Yuan, X., and Gao, Z. (2020). An Experiment-Based Review of Low-Light Image Enhancement Methods. *IEEE Access*, 8:87884–87917.

Wang, Y., Song, W., Fortino, G., Qi, L. Z., Zhang, W., and Liotta, A. (2019). An Experimental-Based Review of Image Enhancement and Image Restoration Methods for Underwater Imaging. *IEEE Access*, 7:140233–140251.

Xiang, T., Xiao, H., and Qin, X. (2021). Quality-distinguishing and patch-comparing no-reference image quality assessment. *Multimedia Tools and Applications*, pages 1–24.

Xu, Y., Zhang, Y., Wang, H., and Liu, X. (2017). Underwater image classification using deep convolutional neural networks and data augmentation. In *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–5.

Yang, M., Hu, K., Du, Y., Wei, Z., Sheng, Z., and Hu, J. (2020). Underwater image enhancement based on conditional generative adversarial network. *Signal Processing: Image Communication*, 81.

Yang, M. and Sowmya, A. (2015). An underwater color image quality evaluation metric. *IEEE Transactions on Image Processing*, 24(12):6062–6071.

Zamora-Arellano, F., López-Bonilla, O. R., García-Guerrero, E. E., Olguín-Tiznado, J. E., Inzunza-González, E., López-Mancilla, D., and Tlelo-Cuautle, E. (2021). Development of a Portable, Reliable and Low-Cost Electrical Impedance Tomography System Using an Embedded System. *Electronics*, 10(1).

Zhang, S., Wang, T., Dong, J., and Yu, H. (2017). Underwater image enhancement via extended multi-scale retinex. *Neurocomputing*, 245:1–9.

Zhang, W., Dong, L., and Xu, W. (2022). Retinex-inspired color correction and detail preserved fusion for underwater image enhancement. *Computers and Electronics in Agriculture*, 192.

Zhang, W., Dong, L., Zhang, T., and Xu, W. (2021a). Enhancing underwater image via color correction and Bi-interval contrast enhancement. *Signal Processing: Image Communication*, 90(January):116030.

Zhang, W., Pan, X., Xie, X., Li, L., Wang, Z., and Han, C. (2021b). Color correction and adaptive contrast enhancement for underwater. *Computers and Electrical Engineering*, 91(January):106981.

Zhou, J., Yao, J., Zhang, W., and Zhang, D. (2021). Multi-scale retinex-based adaptive gray-scale transformation method for underwater image enhancement. *Multimedia Tools and Applications*.

Zhuang, P., Li, C., and Wu, J. (2021). Bayesian retinex underwater image enhancement. *Engineering Applications of Artificial Intelligence*, 101:104171.

Appendix A

Articles derived from thesis work

Publications in JCR journals

Aguirre-Castro, O., Garcia-Guerrero, E., Lopez-Bonilla, O., Tlelo-Cuatle, E., López-Mancilla, D., Cárdenas-Valdez, J., Olguín-Tiznado, J., and Inzunza-González, E. (2022). *Evaluation of underwater image enhancement algorithms based on Retinex and its implementation on embedded systems.* Elsevier, *Neurocomputing*, 494:148-159, `https://doi.org/10.1016/j.neucom.2022.04.074`.

Aguirre-Castro, O., Garcia-Guerrero, E., Lopez-Bonilla, O., Tlelo-Cuatle, E., López-Mancilla, D., Cárdenas-Valdez, J., Olguín-Tiznado, J., and Inzunza-González, E. (2019), *Design and Construction of an ROV for Underwater Exploration*, MDPI, *Sensors*, `https://doi.org/10.3390/s19245387`.

Demonstrative videos

Aguirre-Castro, O., Garcia-Guerrero, E., Lopez-Bonilla, O., Inzunza-González, E. (2021), Visión Artificial para Identificación de peces ornamentales, XXVIII Jornadas de Ingeniería Arquitectura y Diseño, Universidad Autónoma de Baja California, `https://www.youtube.com/watch?v=6ED1zZNBgP8&list=PLe7t__ZW3tJyNQDtdU2ch6cXnH58ZRE79&index=5`.

Aguirre-Castro, O., Garcia-Guerrero, E., Lopez-Bonilla, O., Inzunza-González, E. (2020), Robótica submarina, XXVII Jornadas de Ingeniería Arquitectura y Diseño, Universidad Autónoma de Baja California, `https://www.youtube.com/watch?v=CYGSwwiqn3Q`.