Underwater image processing

David FAGET CAÑO
ENS Paris-Saclay
david.faget_cano@ens-paris-saclay.fr

Gabriel SINGER
ENS Paris-Saclay
gabrielsinger2@gmail.com

***Abstract*—The challenge of recovering accurate colors and shapes in underwater scenes has prompted various imaging techniques. Correcting colors in underwater images is crucial for applications like 3D imaging and navigation. Existing methods, not reliant on knowledge of physical parameters, utilize manual adjustments or computational color constancy approaches. The conventional gray-world hypothesis, rooted in achieving a gray average image, proves inefficient in most cases. Consequently, a pressing need arises for alternative algorithms and methods to overcome these limitations. Thus, we will start by presenting classical algorithms such as gray-world and white patch. Then, we will present two more sophisticated algorithms: ACE and All-in-One. Finally, we will compare all the studied algorithms.**

***Index Terms*—Underwater Imaging, Color Correction, Image Enhancement**

I. Introduction

The human eye is able to distinguish the color of an illuminated object in an illuminated scene. If, for example, you look at a white cloth in a room where the ambient light is slightly yellow, your eyes will see the cloth as white, whereas a digital camera, a priori, will see it as yellow. To compensate this, white balance is used. For images presenting a wide range of colors, the gray-world algorithm -which we have implemented- gives satisfactory results. However, for images that do not exhibit a great deal of color variability (such as images taken in an underwater environment), this algorithm is insufficient, and specific algorithms for underwater imaging must be used. Underwater image acquisition has many applications, for example in surveillance and underwater navigation.

This represents a real challenge for image processing. Depth, salinity, the nature of the sand and the polluted nature of the water are just some of the parameters that influence and complicate image acquisition. Added to this is the water's ability to diffract light rays and reduce their intensity. According to [5], the poor quality of images acquired is mainly due to an exponential decrease in light as a function of the distance it travels. On the one hand, this is due to diffraction by particles in the water (waste, grains of sand), and on the other, to the dissipation of light energy. Added to this is the loss of color information: the deeper the image, the fewer the colors. The disappearance of colors is a function of wavelength; blue-violet, with the longest wavelength in the visible spectrum, is often the color most often found, [5].

All these problems create the need for algorithms to deal with underwater images. Two types can be distinguished: restoration methods and enhancement methods. Image restoration often involves understanding and reversing the degradation factors by applying methods such as deconvolution and noise reduction. Therefore, it requires a solid physical understanding of degradation factors. The goal is to recover the original image as closely as possible, and it aims for accuracy and fidelity to the original scene.

On the other hand, image enhancement is about improving the visual appearance of an image or making it more suitable for a specific task. It is a more subjective process and does not necessarily aim to restore the original image. Techniques in image enhancement include adjusting brightness and contrast, applying spatial or frequency domain filters, and histogram manipulation. These methods are widely used in photography, satellite imagery, consumer electronics (to make images more appealing or informative) and for sure in underwater imaging. Image enhancement is thus driven by the desired visual outcome, which can vary based on individual preferences or specific application needs.

Recent advancements in neural networks have significantly enhanced their role in image processing tasks such as contour detection, deblurring, image enlargement, and color modification. However, a common challenge with many neural network models, including widely-used ones like Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), is their limited ability to generalize. Typically, an algorithm trained on images captured at a particular depth may not perform well on images taken at a significantly different depth. This lack of adaptability has been a notable concern. Paper [1] addresses this issue, presenting developments that improve the adaptability of these neural networks to work effectively across images of varying depths.

In this work, we will focus on enhancement methods. We will start by presenting classical algorithms such as gray-world and white patch. Then, we will present two more sophisticated algorithms: ACE and All-in-One. Finally, we will compare all the studied algorithms.

II. Classical algorithms

In this section we will extend two algorithms seen in class and apply them to underwater images: gray-world and white patch.

A. The gray-world assumption

First of all, we are going to present and implement the simplest approach we could think of: the gray-world assumption. This is based on the idea that, on average, the colors in a typical scene tend to balance out to a shade of gray. In other

words, the assumption is that the sum of the intensities of the red, green, and blue channels across an entire image should be roughly equal.

This assumption is rooted in the observation that natural scenes often have a balanced distribution of colors, and the illumination in outdoor settings, for example, is usually white or grayish. By assuming that the overall color balance in an image is neutral, algorithms can correct for variations in lighting conditions and color cast, making the image appear more natural.

Mathematically, the gray-world assumption can be expressed as:

$$R_{avg} = G_{avg} = B_{avg}$$

where R_{avg}, G_{avg} and B_{avg} are the average intensities of the red, green, and blue channels, respectively.

The main drawback of gray-world is that it fails to produce correct colors in images with strong color biases or under non-standard lighting conditions, which is the case for underwater images. We have therefore implemented the algorithm and we have tested it with some images. Figure 1 shows that this algorithm has many limitations. In fact, we observe that the red and green colours appear in the image. This is what we expected, as they compensate for the blue, which is the only colour seen in the original image. This observation can be extended to most underwater images, which show blue or even greenish tones, but rarely red.

Fig. 1: A ship before and after gray-world processing. We observe that the obtained colors are not realistic.

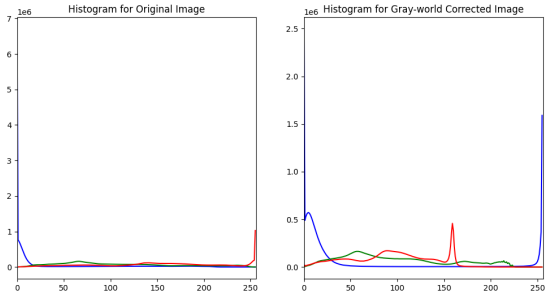

Fig. 2: Histograms before and after gray-world processing.

Figure 2 shows histograms before and after gray-world processing. We indeed observe that red and green colors, which where almost absent on the original image, are present on the modified one. This highlights the need for new, more sophisticated algorithms.

B. *The white patch algorithm*

Another simple white balance algorithm is the white patch algorithm -which we have also implemented-. The essence of this method lies in its approach to scaling the color channels, ensuring that the brightest pixels in each channel are transformed to white. This technique hinges on the assumption that the brightest pixel within an image is, in fact, white.

The simplicity of the white patch algorithm is one of its main advantages, making it easy to implement across various applications. It shows particular effectiveness in scenarios where an image contains a dominant white or neutral gray area, efficiently correcting white balance issues. Moreover, in instances where an image features a distinctly bright area, this algorithm proves to be especially beneficial.

However, the algorithm is not without its limitations. Its foundational assumption that the brightest color in an image equates to white can sometimes lead to inaccuracies. If this assumption fails to hold true, the algorithm may over-correct, resulting in unnatural coloration or visible artifacts within the image. Additionally, it can inadvertently cause color shifts or artifacts in certain areas of the image, particularly in those lacking a clear reference point.

The white patch algorithm is most effective in controlled environments, particularly where the brightest point is known to be white, and the lighting conditions are consistent throughout the image. Conversely, its effectiveness diminishes in situations involving multiple light sources, mixed lighting conditions, or when the reference area does not represent the overall lighting conditions.

Applying the white patch algorithm to underwater photography introduces unique considerations. Underwater images often suffer from color distortion due to the selective absorption and scattering of light in water. This typically results in a blue or green tint overshadowing the true colors of the underwater environment. By applying the white patch algorithm, a reference point for true white under normal lighting is established, enabling a rudimentary form of color correction. This assists in balancing the color spectrum, reducing the overpowering blue or green hues, and bringing the colors closer to what would be perceived in natural light.

Nevertheless, it's crucial to acknowledge that the white patch algorithm, while helpful, may not fully address the complexities associated with underwater color distortion.

The use of a percentile in this algorithm involves selecting a point in the image's brightness distribution as a reference for what should be considered white. For instance, if the 95th percentile is used, it means the algorithm identifies the brightness level below which 95% of the image's pixels fall. This level is then assumed to be white, and the rest of the colors in the image are adjusted relative to this point.

The choice of percentile is crucial because it determines which part of the image is used as the reference white point. A high percentile (like 95th or 98th) will pick a very bright part of the image, which could be effective in scenes with a clear white reference but might lead to overexposure in others. On the other hand, a lower percentile might not accurately

identify the intended white point, especially in images with less pronounced white areas.

We have tested it on the same image than gray-world. We can clearly see that the higher the percentile, the less change there is with the original image. In fact, if we choose the percentile equal to 100, we get exactly the same image. This is exactly what we expected.

Fig. 3: A ship before and after white patch processing (percentile=25).

Fig. 4: A ship before and after white patch processing (percentile=50).

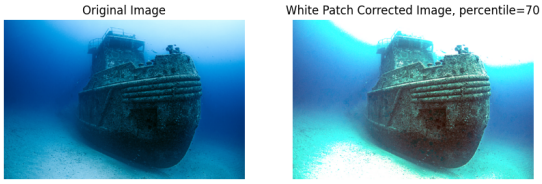

Fig. 5: A ship before and after white patch processing (percentile=70).

Fig. 6: A ship before and after white patch processing (percentile=90).

We can be even more precise in this comparison. Visually, it is clear that the higher the percentile, the greater the difference. However, we can use objective metrics to measure this difference. One of them is the structural similarity index (SSIM) which we will present later. Intuitively, the higher the SSIM, the less difference there is between images. This is coherent with what we observe in Table I.

Fig. 7: A ship before and after white patch processing (percentile=100).

Percentile Value	25	50	70	90	100
SSIM	0.32	0.42	0.54	0.90	1.00

TABLE I: SSIM metric: Higher values mean similar results. The reference image is the original one.

C. Combining both algorithms

Finally, we have decided to combine both algorithms to see what we obtain. This also allows us to introduce ACE algorithm, presented in the next section, which uses those two assumptions (among others). The main observation is that both operations are not commutative: it is not the same to apply first gray-world and then white patch than to first apply white patch and then gray-world.

When we first apply the white-patch technique, we adjust the brightest pixels, potentially altering the overall color balance. If we then apply the gray-world assumption, we're adjusting this already modified balance towards a neutral gray, which could lead to different results compared to starting with the gray-world assumption initially. Each method impacts the image's color distribution in its own way, and the sequence of these adjustments can lead to different enhancements.

We have observed this by applying both algorithms to the same image. Results are shown in Figures 8 and 9.

Fig. 8: A ship before and after white patch + gray-world processing.

Fig. 9: A ship before and after gray-world + white patch processing.

III. ACE and its Fast Implementation

A. ACE

The "ACE and its Fast Implementation" [2] paper discusses the Automatic Color Enhancement (ACE) method for color image enhancement. This method, introduced by Gatta, Rizzi, and Marini, is based on modeling several low-level mechanisms, including the gray-world assumption and the white patch algorithm presented before. The paper describes two fast approximations of ACE. The first uses a polynomial approximation of the slope function to decompose the main computation into convolutions, significantly reducing the computational cost. The second algorithm, based on interpolating intensity levels, also reduces the main computation to convolutions. Both approaches aim to make ACE more efficient for practical use in image enhancement and color correction.

We are going to start by reformulating the ACE algorithm with our words. The technique operates on a grayscale image or a color channel of an image, denoted as I. The domain of this image is represented by Ω, with intensity values scaled between 0 and 1. In the case of color images, this operation is independently applied to each of the RGB channels.

The contrast enhancement is achieved through a formula for $R(x)$, which involves a summation over the domain Ω, excluding the point x. This formula takes the intensity difference between $I(x)$ and $I(y)$, scales it using the function s_α, and divides it by the Euclidean distance $\|x-y\|$ between the points x and y. This approach enhances the local contrast in the image.

The slope function s_α is pivotal in this process. It scales intensity differences, effectively amplifying small differences to enhance contrast and saturating large differences to maintain overall balance. The function is designed to keep the output within the range of [-1, 1].

A significant aspect of this technique is the role of the parameter α. As α approaches infinity, the expression for $R(x)$ simplifies, resulting in a sum of positive contributions from points where $I(x) > I(y)$ and a subtraction of contributions from points where $I(y) > I(x)$. This parameter is likely influential in modifying the image's histogram.

Following the contrast enhancement, the enhanced channel is computed. This is done by stretching R to fit within the [0, 1] range using the formula $L(x) = \frac{R(x)-\min R}{\max R-\min R}$. This step is crucial in adapting the image after its local contrast has been enhanced.

The process can be summarized in two stages. The first stage adapts local image contrast by simulating lateral inhibition. It does this through the weighting of neighbor differences according to their distance, a process significantly influenced by the value of α, which dictates the amplification or saturation of differences. The second stage adjusts the globally enhanced image to achieve balanced overall contrast and brightness, ensuring effective utilization of the image's entire dynamic range.

This method provides a sophisticated approach to image enhancement, particularly beneficial for images that require both local and global adjustments in contrast and color balance, making it suitable for a wide range of applications in image processing.

B. Link between ACE and histogram equalization

We wanted to go further and we investigated about the link between ACE and histogram equalization. Indeed, paper [2] explains that ignoring the spatial weighting $\omega(x, y)$, uniform histogram equalization corresponds to $\alpha = \infty$, but it was not very clear for us. Therefore, we found paper [7] where this step is clearly explained. We are going to explain here the idea behind the proof.

As α approaches infinity, the slope function $s_\alpha(t)$ in the ACE method converges to the signum function $s_\infty(t) = \text{sign}(t)$. The signum function essentially captures the sign of the difference between pixel intensities, ignoring their magnitude.

In the ACE method, local image contrast is adapted by amplifying small differences and saturating large differences between neighboring pixel intensities. This process, influenced by the slope function s_α, effectively modifies the dynamic range of the image based on local content.

Paper [7] shows that uniform histogram equalization emerges as a minimizer when $\alpha = \infty$ and the weight function $\omega(x, y) = 1$. This indicates that, under these conditions, the ACE method aligns with the goals of uniform histogram equalization, which is to redistribute the image's intensities so that they are uniformly spread across the available range.

To sum up, when α is set to infinity in the ACE method, the slope function becomes a simple signum function, and the ACE's process of adapting local contrast and redistributing intensities aligns with the principle of uniform histogram equalization. This results in an image enhancement effect that is similar to applying uniform histogram equalization, where the goal is to achieve a uniform distribution of pixel intensities across the image.

C. Fast Implementation

Paper [2] presents a fast implementation of the ACE algorithm introduced before. Since the goal of this project is not reformulating the paper, we will just sum up in a couple of sentences the two methods presented by [2].

The first method consists in approximating s_α by polynomial functions of odd degrees.

The second variant consists in approximating $R(x)$ by a kind of first-order limited expansion of the function $R(x, \bullet)$ defined by

$$R(x; L) = \sum_{y \in \mathbb{T}^2} \omega(x-y) s_\alpha(L - I(y)).$$

ω designates either the Euclidean norm or a Gaussian.

D. ACE Results

We are going to present some results obtained from the IPOL demo of paper [2]. First, we vary only α, the slope of s_α. The greater the α, the more intensely the colors appear.

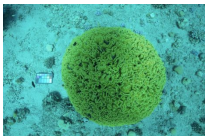

Fig. 10

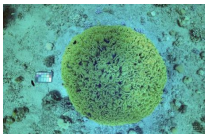

Fig. 11

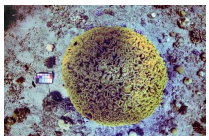

Fig. 12

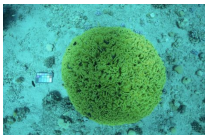

Fig. 13

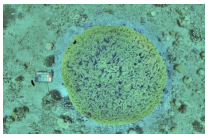

Fig. 14

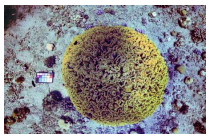

Fig. 15

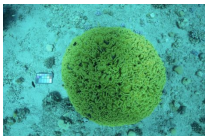

Fig. 16

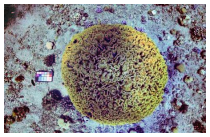

Fig. 17

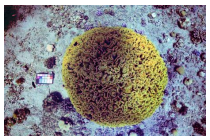

Fig. 18

Fig. 19

Fig. 20

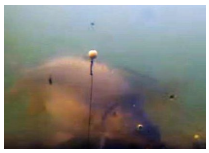

Fig. 21

Fig. 22: ACE results

Original	$(\frac{1}{\sqrt{x^2+y^2}}, 9, 1)$	$(\frac{1}{\sqrt{x^2+y^2}}, 9, 8)$
Original	$(\exp -\frac{x^2}{\sigma^2}$,9,1,20)	$(\exp -\frac{x^2}{\sigma^2}$,9,1,60)
Original	$(\exp -\frac{x^2}{\sigma^2}$,9,1,50)	$(\exp -\frac{x^2}{\sigma^2}$,9,1,60)
Original	$(\frac{1}{\sqrt{x^2+y^2}}, 9, 1)$	$(\frac{1}{\sqrt{x^2+y^2}}, 9, 8)$

TABLE II: ACE's parameters (polynomial interpolation) corresponding to the figures above : $(\omega, d(s_\alpha), \alpha, \sigma)$.

Observations: For the first photo, we can see that as α increases, so does the colorization. For $\alpha = 8$, for Gaussian and inverse weights, the color pallet has intense colors. This is not the case for $\alpha = 1$. Nevertheless, the algorithm acts only very locally and loses effectiveness at the edge, which is understandable given its construction, as the gradient explodes at the edge. We can see the halo effect surrounding the coral, and above all, the edges are colored red, see figure 15 for instance. Note that no blur is added.

For the picture with the fish, we can see that the original is blurred and so are the processed images. The sand grains are colorized and stand out much more on the processed images. We can see that the algorithm is insufficient if the photo is of poor quality, but that it works very well, at least locally, for white balance.

IV. Deep Learning: A Modern Approach

The "All-In-One" (A.I.O) article introduces a novel approach to underwater image enhancement using domain-adversarial learning. The algorithm proposed by the authors effectively learns to enhance images in a way that is agnostic to the varying conditions of underwater environments. Here, we are going to sum up the article with our words and to discuss its implementation.

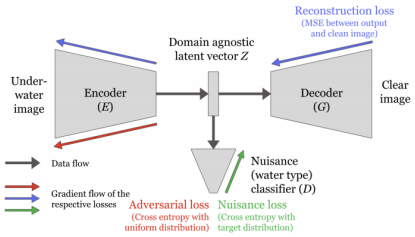

Fig. 23: Architecture of the model

A. How the algorithm works

First, the CNNs are employed to analyze underwater images and differentiate between essential content and water-type-specific characteristics. This distinction is one of the keys to effectively enhancing the images by retaining important content while mitigating distortions.

In the same time, Generative Adversarial Networks (GANs) are utilized to generate realistic, enhanced versions of underwater images. The process involves a generator network, which creates enhanced images, and a discriminator network, which evaluates the realism of these images. Through this adversarial process, the GAN learns to produce images that closely resemble natural, undistorted underwater scenes.

The model architecture consists of an encoder (E), a generator (G), and a nuisance classifier (D). The encoder's task is to produce a latent vector Z that is agnostic to the water type, while the generator reconstructs a clear image from Z.

B. Model Architecture

The model's core components are:

- **Encoder (E)**: Extracts the latent vector Z from the input image.
- **Generator (G)**: Reconstructs the clear image from Z.
- **Nuisance Classifier (D)**: Classifies the water type from Z, with the aim of being made uncertain by the encoder through adversarial training.

C. Loss Functions

The model is guided by three loss functions:

1) Reconstruction Loss: The reconstruction loss L_R ensures the fidelity of the reconstructed image to the ground truth:

$$L_R(X, Y) = \frac{1}{N} \sum_{i=1}^{N} |G(Z)_i - Y_i|^2 \quad (1)$$

where $Z = E(X)$ and N is the number of pixels in the image.

2) Nuisance Loss: The nuisance loss L_N is the cross-entropy loss used to update the nuisance classifier:

$$L_N(X, C) = -\sum_{c=1}^{M} y_c \log D(Z)_c \tag{2}$$

where $y_c = 1$ if $c = C$ else $y_c = 0$, $Z = E(X)$, and M is the number of water type classes.

3) Adversarial Loss: The adversarial loss L_A aims to increase the uncertainty of the nuisance classifier:

$$L_A(X) = \sum_{c=1}^{M} D(Z)_c \log D(Z)_c \tag{3}$$

where $Z = E(X)$ and M is the number of classes.

These loss functions collectively ensure the model generates clear images while discarding features specific to water types, thus achieving consistent image enhancement across different underwater conditions.

D. Performance of this architecture

A standout feature of this model is its ability to generalize across different water types. This capability is vital, considering the varying conditions of underwater environments, ranging from clear coastal waters to murky deep-sea regions. The model adapts to these conditions, ensuring consistent image enhancement regardless of the water type.

The approach has shown promising results in practical applications such as underwater navigation and marine research.

Note that in the article they go one step further and show that they can recognize on the image rendered by their algorithm objects that are not recognizable on the original image.

E. Implementation

The most time-consuming stage of this project was trying to get the official code of the paper [], which was on GitHub, to work. We note that the GitHub already contains the checkpoints (already trained models), so here we will limit ourselves to the inference stage on our photos. However, this has also been a challenge for us. The steps (and modifications) we have followed are as follows:

- Modify the .py files to work on a notebook (.ipynb format).
- Download the UIEB dataset (the NYU dataset originally used in the paper can no longer be downloaded in the desired format).
- Modify the code to be compatible with our versions of the Python libraries (we tried to download the environment, but some versions were too old and we got errors).
- Modify the Dataset class (the whole part of loading clear images was missing).
- Modify the code to run on the CPU.

Following the previous steps we were able to run the code without errors. Here are some outputs when testing the code on some of the images of the UIEB dataset mentioned above.

Fig. 24: From left to right - Input image, output of the model, ground-truth (reference) image.

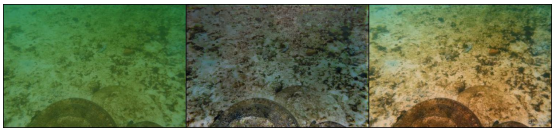

Fig. 25: From left to right - Input image, output of the model, ground-truth (reference) image

V. Comparison

A. Qualitative comparison

Fig. 26: Gray-World

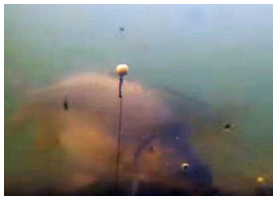

Fig. 27: ACE, s_8

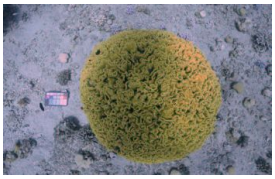

Fig. 28: Gray-World

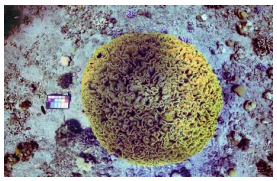

Fig. 29: ACE

Qualitatively, we can see that in all three cases, ACE gives an image where colors are much better. In the case of the freshwater photo (fig23,fig24), we can see that ACE has a higher contrast than Gray-World, and also deflates the image. The limitations of gray-world can also be seen in Fig (27), where the original image is essentially green, whereas ACE renders a less blurred image with better contrast.

However, we believe that this qualitative analysis is subjective: some people may prefer photos with colors that other people may not like. Since only two people (Gabriel and David) have done this project, we decided to survey to assess people's preferences. The results were indeed very interesting.

The survey took the form of a Google form, and we got 20 responses. The aim was to see whether users qualitatively prefer A.I.O or one of the other algorithms studied. The first question shows two photos of the same scene, one processed by ACE, the other by A.I.O. The question asks whether these photos were taken underwater. One person in two said that at least one of the two photos had not been taken underwater. This shows just how good both algorithms are at color enhancement. Unexpectedly, people prefer the rendering of the ACE image. This must surely be because ACE renders the colors of the image more vividly, whereas A.I.O more faithfully renders the colors that are often less vivid in marine environments. Again, quite unexpectedly, when asked which photo people prefer between the original photo or WP+ACE or GW+ACE, people prefer the original photo with WP+ACE in second place. This must be due to the fact that the combination of the WP, ACE and GW, ACE algorithms results in less color, with a lot of white overall, where green would be expected, for example.

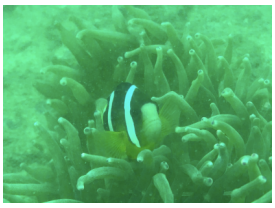

Fig. 30: Original

Fig. 31: ACE

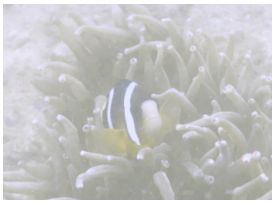

Fig. 32: GW

Fig. 33: WP

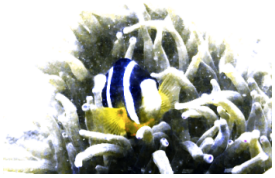

Fig. 34: WP+ACE

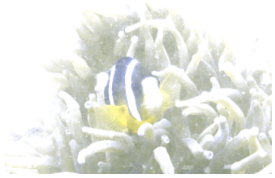

Fig. 35: GW+ACE

Fig. 36: WP: White Patch, Percentile=70
GW: Gray-World
ACE : $\alpha = 8 \quad \omega(x, y) = \frac{1}{\sqrt{x^2+y^2}}$.

B. Quantitative comparison

By construction, it is impossible for us to have a "perfect" digital image of the underwater scene. So we use the SSIM measurement with the image rendered by the ALL-IN-ONE algorithm as reference.

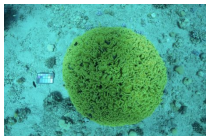

Fig. 37

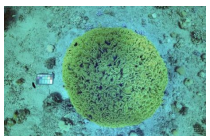

Fig. 38: $\alpha = 1$

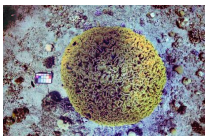

Fig. 39: $\alpha = 8$

Fig. 40: Original

Fig. 41: ACE

Fig. 42: A.I.O

The Structural Similarity Index (SSIM) is a metric used to measure the similarity between two images. Unlike simpler, pixel-based difference measures, SSIM takes into account perceptual phenomena, making it more aligned with how humans perceive visual similarity. SSIM evaluates three key aspects of comparison: luminance, contrast, and structure. Luminance assesses the closeness of the average brightness levels between images, contrast compares the variance in pixel values to evaluate texture and depth, and structure examines the spatial distribution of pixels to determine the similarity of patterns or features. A high SSIM score indicates high similarity across these aspects, while a lower score suggests noticeable differences. SSIM, therefore, offers a comprehensive and human-centric way to assess image similarity, proving particularly useful in fields like image processing, computer vision, and quality assessment.

Gray World	White Patch	ACE	All-In-One	Picture
0.58	0.39	0.54	1	Vases
0.57	0.46	0.34	1	Algae
0.61	0.46	0.58	1	Boat
0.77	0.61	0.65	1	Fish
0.68	0.50	0.50	1	Ground

TABLE III: SSIM metric:
Higher values mean better results. The reference image is the one done by All-In-One.

We have chosen to use ALL-IN-ONE. In our opinion, because of the way camera sensors are constructed, the "ground truth" photo has necessarily been processed, and so choosing it as a reference means comparing it to another algorithm that we're not studying.

We can see that for all three images, Gray-World has the highest SSIM, which contradicts what we see. Normally, it would be ACE that would have the highest SSIM, since SSIM is a metric that is supposed to measure proximity to what we see [4]. These differences can be explained by the fact that gray-world tends, in the three cases above, to produce an image with much lower contrast than ACE.

Fig. 43: Original

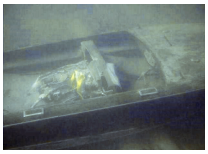

Fig. 44: ACE

Fig. 45: A.I.O

Fig. 46: ACE vs CNN ALL IN ONE

Fig. 47: Or

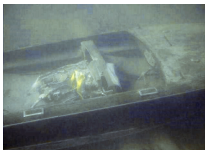

Fig. 48: ACE

Fig. 49: A.I.O

Fig. 50: ACE vs A.I.O

VI. Conclusion

In this work we explore the complexities of underwater image processing, particularly focusing on color correction due to the unique light absorption and scattering properties of water. We compare traditional algorithms like Gray-World and White Patch with more advanced techniques such as ACE and an AI-based method, All-in-One, which integrates Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs).

A non-negligible portion of our time was devoted to adapting the All-In-One algorithm's code from GitHub to suit our specific requirements. This involved extensive modifications to ensure compatibility with our software environment and the data we intended to process. We needed to restructure Python files for effective use in a notebook format, incorporate the UIEB dataset in place of the originally used NYU dataset, and make critical adjustments to align with our Python library versions, including modifying the code for CPU execution.

We show the limitations of Gray-World in handling uniformly colored underwater scenes and the effectiveness of ACE in enhancing contrast and color intensity. The ACE algorithm is described in detail, focusing on its role in local contrast enhancement and overall image adjustment. Additionally, the paper explores the relationship between ACE and histogram equalization, explaining how ACE aligns with uniform histogram equalization under specific conditions. The All-in-One method shows its adaptability across different water types, ensuring consistent image quality regardless of varying underwater conditions. To evaluate those algorithms, we employ the Structural Similarity Index (SSIM). Based on the responses to a survey we have elaborated, we have chosen to take as reference the image given by the All-In-One algorithm, for quantitative comparison. We illustrate, quantitatively and qualitatively, that more complex approaches like All-in-One can outperform "classical" methods in underwater image enhancement.

TABLE IV: GW, WP (Per=70), ACE, A.I.O

References

[1] Pritish Uplavikar, Zhenyu Wu, Zhangyang Wang. *All-In-One Underwater Image Enhancement using Domain-Adversarial Learning*. 2019. https://arxiv.org/abs/1905.13342. *arXiv preprint*, cs.CV.

[2] Pascal Getreuer. *Automatic Color Enhancement (ACE) and its Fast Implementation*. *Image Processing On Line*, 2:266–277, 2012. https://doi.org/10.5201/ipol.2012.g-ace.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org

[4] Zhou Wang, A.C. Bovik, H.R. Sheikh, et E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 10.1109/TIP.2003.819861.

[5] Raimondo Schettini, Silvia Corchs. *Underwater Image Processing: State of the Art of Restoration and Image Enhancement Methods*. *EURASIP Journal on Advances in Signal Processing*, 2010. https://doi.org/10.1155/2010/746052.

[6] L. Abril Torres-Méndez and Gregory Dudek. *Color Correction of Underwater Images for Aquatic Robot Inspection*. In *Proceedings of the Conference*, 2005, pp. 60-73. ISBN: 978-3-540-30287-2. https://doi.org/10.1007/11585978$_5$.

[7] M. Bertalmio, V. Caselles, E. Provenzi and A. Rizzi, "Perceptual Color Correction Through Variational Techniques," in IEEE Transactions on Image Processing, vol. 16, no. 4, pp. 1058-1072, April 2007, doi: 10.1109/TIP.2007.891777.