UNDERWATER IMAGE COLOR RESTORATION

Prepared by

Salmane Dazine

Supervised by

Dr. Naeem Nisar Sheikh

24/04/2020

CONTENTS

ABSTRACT

Being a filmmaking and photography enthusiast, it has piqued my curiosity during my internship in Bali that the underwater footage I have taken suffers from a severe color distortion and the dominance of an unnatural blue filter, especially when trying to capture far objects. Discovering an accurate and practical method that can lower the blurriness and defects in underwater photography would be crucial for many industries, notably, the environmental and biological ones. Effectively, this could contribute to several exploration and research projects that aim to monitor the safety of sea-creatures and the preservation of sea-life. The method that I innovated in this project is inspired by two main techniques. This consists of combining a color restoration technique known as *the bright channel prior method*, with a color enhancement technique known as *the histogram equalization*. The color restoration method starts first by extracting the bright channel image from the input, which is the degraded underwater image. Then rectifying the bright channel, using the maximum channel difference image. After that, I detect the value of the atmospheric light using the rectified bright channel. At this point, I can extract the initial transmittance using the variables of the atmospheric light and rectified bright channel that I have already defined. After refining the transmittance image, I can restore the original image using the image formation equation. The work isn't finished yet by restoring the image, as I need to enhance its colors using the histogram equalization method. Testing the results can be done in different ways depending on the method of work adopted and the resources at the disposition of the designer. In this project, the testing is mainly conducted through comparison to pictures of the same scene at a close distance.

1. INTRODUCTION

This project was a source of exploration of one of my biggest passions throughout my childhood and growing up. Film making and swimming have always been there with me through travels, academic studies, and even work opportunities. One of my latest projects involved traveling to the island of Bali and interning with an international film festival, known as Balinale. This opened the door for me to expand my interests in sea life on one hand, and sharpen my photography skills on the other. I had the privilege to further explore these passions and take a lot of footage underwater using my GoPro for a film project I am planning to release. I have always been satisfied with the quality of the camera and all the options it offers. However, I have always been bugged by the very strong blue filter that I find all over the images and videos, especially when I am diving in very deep waters. Below is a screenshot from a video I have taken, while following a sea turtle over the reefs of Gili Air island on 28 July 2019.

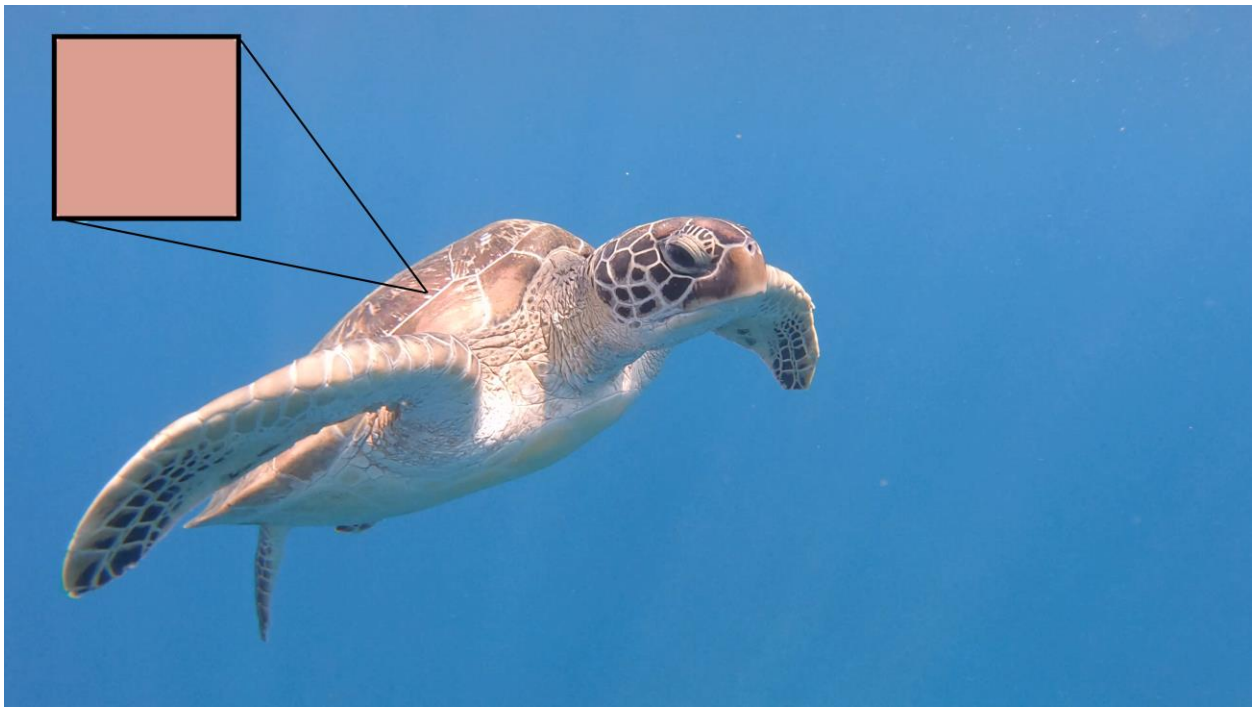

Figure 1.1 Sea Turtle Color from a Close Distance

The first screenshot shows how relatively accurate and close to reality the colors of the turtle look, and how sharp and clear they look. The square is a sample of the color of a pixel on the back of the turtle extracted by the color picker tool on Paint.net 4.1.6. The hex color code extracted from this point is DB9F92 with a blue intensity of 146 and a red intensity of 219 on a 255 scale.

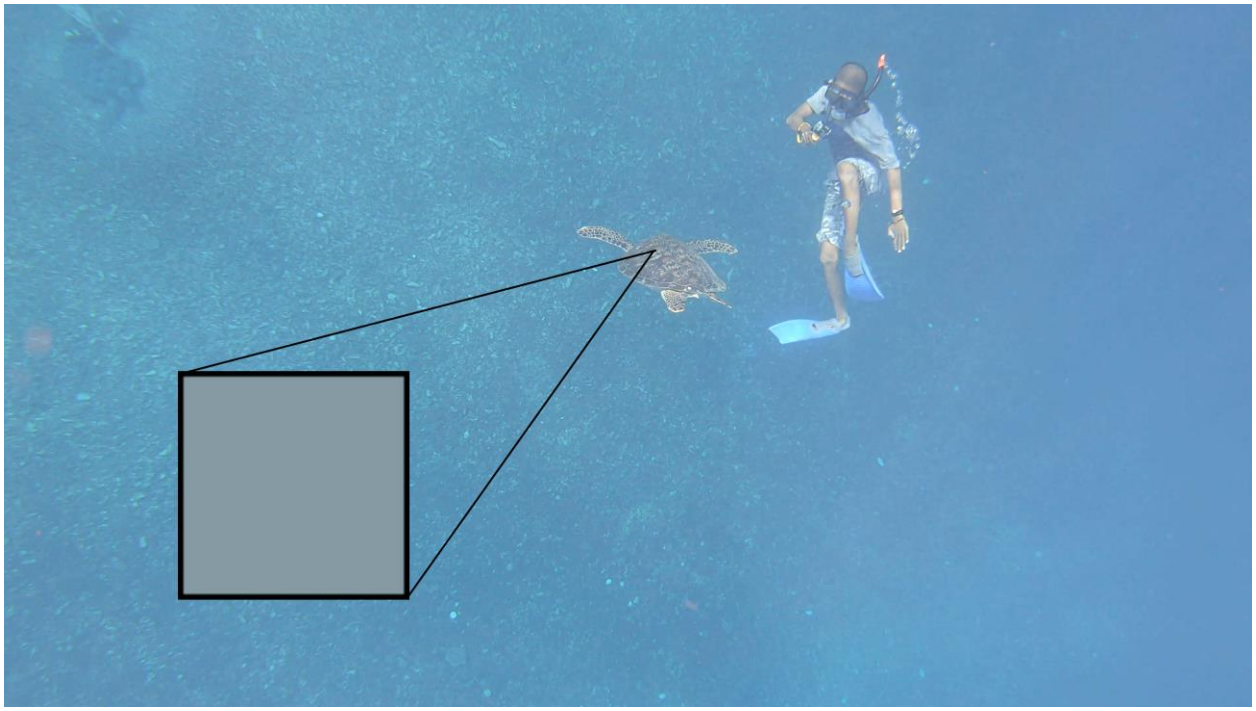

Figure 1.2 Sea Turtle Color from a Far Distance

Nevertheless, the second image, 15 seconds later when the turtle swam deep down in the sea, shows a sort of blueish and blurry version of the turtle. The second square is the color extracted from, roughly, the same area on the back of the turtle. The hex color code this time is 869AA4 with an increased blue intensity of 164 and a decreased red intensity of 134 on a 255 scale. This issue raised my interest for a long time and made me wonder about its source, and if there is anything that can be done to diminish it.

Could it be resolved only through unnatural filters and manual post-processing applications or could there be accurate decomposition methods that can restore the true natural colors of this turtle?

2. PLANNING

2.1 OVERVIEW

Underwater Image Restoration means improving the visual quality of an image taken underwater by restoring, as accurately as possible, the original colors of the objects on the image.

It is different from *Underwater Image Enhancement* which means improving the visual quality of these images, only through post-processing tools that don't necessarily restore the original colors of the objects pictured but only enhance them so they can look pleasing and natural to the human eye.

Restoring the original colors of underwater objects has been crucial for many industries, notably, the environmental and biological ones. It plays a primary role in many exploration and research projects that aim to monitor the safety of sea-creatures and the preservation of sea-life. Detecting all the necessary measures that allow divers and scientists to learn about the sea-life and help it grow would be difficult without a clear view of all the objects and creatures that exist in the scene under study.

2.2 PROBLEM DESCRIPTION

The question that I will be answering through this project is:

How to restore the original colors of a single picture taken underwater?

As it is clear from the wording of this question, it is detached from any form of restriction or limitation about the quality of the image captured, the distance of captured objects from the camera, the weather conditions, or even the pollution levels of the water environment.

Unlike different projects that have been accomplished in this field, the main objective of this project is to diminish the impact of the factors that play into the underwater vision and come up with a general solution to this problem regardless of the various circumstances that might affect it.

2.3 LITERATURE REVIEW:

Underwater image restoration has always been a challenging project, that has been met with different methodologies and implementations. Several approaches have been employed, compared, and enhanced throughout time to obtain the most accurate results possible, however, most of these methods lead to different results and have rarely been accurate. Each of the methods introduced differs significantly from each other, and all have both strengths and shortcomings in them. Therefore, designing a solution to this project professionally solely depends on doing research and identifying the most optimal and feasible method in terms of the tools in my disposition and the time available.

After reading the content of several papers, I noticed that these methods can be classified into the following categories:

2.3.1 RESTORATION USING MULTIPLE IMAGES

Examples

- "Chromatic framework for vision in bad weather" by S. G. Narasimhan and S. K. Nayar
- "Vision and the atmosphere" by S. G. Narasimhan and S. K. Nayar
- "Enhancing underwater images and videos by fusion" by C. Ancuti, C. O. Ancuti, T. Haber et al.

Summary

The authors of these papers needed to assess several pictures taken in the same scene, but under different weather conditions, to be able to remove fog and recover the true colors of the scene. These might be the most accurate on the level of visual perception, notably, the work of Ancuti, however, the pictures taken can only be restored if they are taken in specific times, positions and for the specific purpose of restoration. These methods obviously cannot work on images taken under the same weather conditions or single images found online.

2.3.2 RESTORATION USING SINGLE IMAGES

Examples

- "Visibility in bad weather from a single image" by R. T. Tan
- "Single image dehazing" by R. Fattal
- "Fast single-image dehazing using linear transformation" by G. Ge, Z. Wei, and J. Zhao

- “Single image dehazing using the change of detail prior” by J. Li, H. Zhang, D. Yuan, and M. Sun
- “Single underwater image enhancement using depth estimation based on blurriness” by Y.-T. Peng, X. Zhao, and P. C. Cosman
- “Initial results in underwater single image dehazing” by N. Carlevaris-Bianco, A. Mohan, and R. M. Eustice

Summary

The authors of these papers relied on the details of one single picture to recover its original colors, mainly by estimating the distance of the object taken, and linking the blurriness and fog levels to the distance detected. However, they all had different results and inaccuracies in the colors of the final images, for example by over enhancing the contrast or the saturation levels.

2.3.3 RESTORATION USING THE CHANNELS OF A SINGLE IMAGE

Examples

- “Single image haze removal using dark channel prior” by K. He, J. Sun, and X. Tang
- “Automatic Red-Channel underwater image restoration” by A. Galdran, D. Pardo, A. Picon, and A. Alvarez-Gila
- “Single underwater image enhancement with a new optical model” by H. Wen, Y. Tian, T. Huang, and W. Gao

- “Region-specialized underwater image restoration in inhomogeneous optical environments” by Z. Chen, H. Wang, J. Shen, X. Li, and L. Xu
- “Restoration and enhancement of underwater images based on bright channel prior” by Y. Gao, H. Li, and S. Wen

Summary

The authors of these papers relied on only one or two of the three color channels (R, G, B) in a picture to extract its original colors. However, some of these methods, like the dark channel prior, work only given certain conditions and are invalid once sky or foggy areas existed, as I will explain later in this paper.

2.4 WORK SITUATION

After understanding the content of these different research papers, and assessing the efficiency of these methods in solving the color distortion and blurriness of the pictures. I decided to go with the method introduced by Y. Gao, H. Li, and S. Wen in “Restoration and enhancement of underwater images based on bright channel prior”, as the most efficient method in my case for the following reasons:

- Ruling out multiple images methods due to the impossibility of finding a large water area and taking the necessary multiple pictures, in light of the current circumstances of the quarantine.

- Ruling out single images methods due to their inaccuracy and oversaturation of the original images.
- Ruling out most channel prior methods, except the bright channel method, due to their inadequacy in foggy environments.
- Selecting the bright channel prior method, which is an improved version of the other methods in the category of restoration by channels, given its validity in restoring images taken in all different kinds of scenes, weather conditions, and environments.

Considering the method of the bright channel prior as the main guide for my project implementation doesn't rule out resorting to other sources and research papers to solve certain issues or enhance the accuracy of my final product. As shall be seen in section 4, I will demonstrate the reasons for choosing any additional technique in my methodology.

3. ANALYSIS

3.1 STEEPLE ANALYSIS

In this analysis, I examine, in the first part, all the external factors that might impact the growth and performance of my project. And in the second part, I describe how my project can affect each of the following industries.

3.1.1 SOCIALLY

The social environment has a significant effect on the market. Noticing the current societal trends that are drawing social media followers to the world of traveling and discovering the major hubs of tourism, islands, and beaches are getting more attention than ever. Be it with visitors discovering beach sports or, professional divers monitoring sea-life. The underwater world is getting humongous populations that can have both negative and positive effects on it. These factors contribute to raising big interest in projects like this that help maintain this world as healthy and functioning as ever.

This project can promote more organic reach and populations discovering the coastal and fluvial regions and areas. Having a public and easily accessible software or hardware tool that makes underwater discovery more clear and exciting can encourage many sea-life passionate individuals to populate these areas.

3.1.2 TECHNOLOGICALLY

Perhaps the most pivotal aspect of software projects. The current generations know the existence and development of new innovative devices and super high-quality cameras that help positively with easing access to the underwater world and extracting the necessary information to get satisfactory results. However, it could also play a negative role, a concrete example of that is the gadgets that allow a natural view of the objects underwater without needing any software programs. One of these gadgets that I had the chance to try is the GoPro red filter which brings life to the attenuated red color spectrum in deep blue waters.

This project, on the other hand, can open the door to the creation of new automated software programs or devices that follow the same methodology described and program implemented to instantly show the divers the original colors of the sea-creatures, simultaneously, while they are discovering underwater.

3.1.3 ECONOMICALLY

Economically speaking, economic growth, foreign exchange rates, and investments made in the sectors of marine life sustainability and tourism are all factors that vary over time and can affect the success of the projects that promote the health of sea-life either positively or negatively.

This project can also play a significant role in promoting sea life-related businesses and increase the profit of companies. In this case, the social aspect is very much tied with the economical aspect, the more organic reach and more human interest are being shown towards the sea-life, the more the companies working in the industry will profit economically.

3.1.4 ENVIRONMENTALLY

The situation of the environment affects this project in many ways. As will be explained later, the weather conditions, size of pollution, type of water, and quantity of minuscule creatures living underwater are all factors that decide the quality and accuracy of the outcome of the method employed in this project. Although this project aims to reduce the effect of the external environment factors, the impact will always be there, especially if the water's color is very unusual and the pollution is at high amounts.

This project has a high tendency to promote the growth and sustainability of the environment if used positively.

> "The quality of underwater images plays a pivotal role in scientific missions such as monitoring sea life, taking a census of populations, and assessing geological or biological environments." [17, Page 1]

However, if used negatively, by trying to help capture large amounts of sea-creatures and damage the eco-system, it will lead to the deterioration of the environment.

3.1.5 POLITICALLY & LEGALLY

Assuring the possession of every permit needed to conduct the underwater operations, and avoid interference with the interests of other groups or individuals is necessary. The legal and political regulations of the government that owns the marine space will always dictate the rate of growth of the business either negatively or positively.

In parallel, this project might lead to the reconsideration or the establishment of new legislation or policies that control the use of tools and devices that can affect the sea environment negatively.

3.1.6 ETHICALLY

As for the ethical aspect, applying the techniques of this project implies abiding by all the ethical measures including acknowledging the work, ideas, and creations of other persons and conducting the work in complete honesty. It should also not contribute to taking part in any unethical behavior that might hurt any organization, individual, or living creatures either directly or indirectly.

This project, on the other hand, might encourage individuals to act either ethically or unethically depending on how they use the tools in their disposition towards the sea-life.

3.2 REQUIREMENTS SPECIFICATION

The work accomplished in this project includes creating a tool that restores the original colors given the input of a single image taken underwater. The tool that I created is not a software application that can be installed and run by public users, but rather a personal-use code that serves into the objective of this project which is demonstrating the solution of the color distortion problem in underwater images. This program implemented abides by the requirements stated below:

3.2.1 FUNCTIONAL REQUIREMENTS

- The program shall take one image as an input.
- The program shall print the name of the current step being processed.
- The program shall print the image dimensions in pixels.
- The program shall notify the user if the source image is large.
- The program shall save the bright channel version of the image.
- The program shall save the rectification of the bright channel image.
- The program shall print the values of the atmospheric light under the three channels (RGB).
- The program shall save the initial transmittance image.
- The program shall save the refined transmittance image.
- The program shall save the restored image.
- The program shall save the final image after the histogram equalization.

3.2.2 NON FUNCTIONAL REQUIREMENTS

- The program is a Python file that is accessible and usable through any python-inclusive integrated development environment.
- The code of the program can be read and understood with clarity using any python-inclusive integrated development environment.
- The program can be managed efficiently by Sublime Text.
- All python libraries cited at the beginning of the program, including OpenCV, NumPy, and SciPy must be installed on the operating system before running.
- The original program code and folder specification are tied to the operating system of the developer.
- If used under a different system, the user must change the folder and name of the source picture through the variable “folder” and “source” declared in the “main” function.
- The processed images will be saved in the same folder where the input image exists.
- The source image size is advised to be smaller than 600*600 pixels for smooth performance and a running time of fewer than 30 seconds.
- An image of size 800*800 takes around 4 minutes and 10 seconds on a high-performance computer as DELL XPS 15 7590 to output the final result.

3.3 FEASIBILITY STUDY:

Completing the requirements of this project imposes following a method that allows using all the time, technology resources available in an optimal manner.

3.3.1 TECHNOLOGY RESOURCES

Physical Tools

The physical resources needed for the project consisted of:

Waterproof Camera

Figure 3.3.1.1 GoPro Hero Black 7 Camera

I used my waterproof camera GoPro Hero7 Black, that I bought before my internship program in Bali, to be able to take clear and hyper smooth pictures and videos underwater. This camera guarantees 4K clear footage with the option of removing all unnatural color filters that can distort the original image.

Floating Camera Handle

Figure 3.3.1.2 GoPro Hero Black 7 Handle

I used the GoPro floating handgrip to prevent sinking the camera, especially under deep waters, and maintaining a non-slip secure grip on the camera. It also serves into aiming the camera better and allows a closer look into far objects that are difficult to reach with bare hands.

High-Performance Computer

I used my computer device, Dell XPS 15 7590 to input, process, and output the images or footage taken.

Figure 3.3.1.3 Dell XPS 15 7590 Laptop

Software Tools

As for software tools and needed to implement this project:

Languages

Python

Figure 3.3.1.4 Python Language Logo

Python is a high-level programming language that can be used for general purposes. I was introduced to this language, along with the library OpenGL, in the advanced Computer Science course Graphic Design taken during my exchange program in South Korea. Nevertheless, following the purposes of this project, I won't be using the library OpenGL which servers mainly for rendering vector graphics. I will be using the

following high-level libraries dedicated to image processing.

Libraries

OpenCV

Figure 3.3.1.5 OpenCV Library Logo

OpenCV, short for Open Source Computer Vision, is a library used in several languages like C++, Python, and Java aimed at solving computer vision problems. It serves in all kinds of images and videos analysis including facial recognition, object recognition, text reading, and many more. “Cv2” is the import name for the new Opencv version.

NumPy

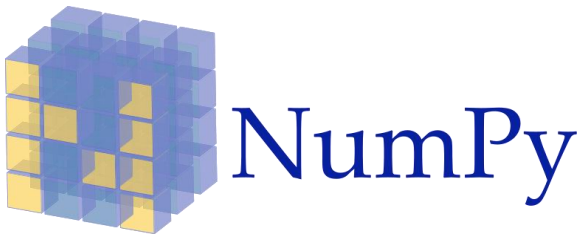

Figure 3.3.1.6 NumPy Library Logo

NumPy, initially released as Numeric, is a library used in Python for scientific computing for fields like linear algebra and matrices. It is commonly known for providing users with high-performance multidimensional array objects and the functions to work with them. It is of significant use in this project since it allows reading, writing, and manipulating the images through multidimensional arrays. Numpy is commonly imported under the abbreviation “np”.

SciPy

Figure 3.3.1.7 SciPy Library Logo

SciPy is another library used in Python for scientific computing built to work with NumPy arrays. Using both of them is powerful to process all kinds of numerical routines. I used specifically the submodule Ndimage dedicated to n-dimensional image processing, by importing it under the name “nd”.

Application Programming Interfaces

Sublime Text

Figure 3.3.1.8 Sublime Text API Logo

Sublime Text is a simple and sophisticated application programming interface for code, markup, and prose that supports several programming languages like C, C++, Java, Python, R, Ruby, and many others. I used the latest version 3.2.2 Build 3211 released in October 2019.

3.3.2 TIME RESOURCES

As for the time resources, it was feasible to complete this project, by respecting the following timetable and sticking by its deadlines.

07/02-05/03

Project Analysis

Making extensive research about the topic and identifying the most efficient methodology to be followed to solve it.

06/03-06/04

Project Design

Reading the selected papers, and transforming the acquired information into an algorithm.

07/04-16/04

Project Implementation

Transforming the algorithm into a running code through a suitable language and environment.

17/04-19/04

Project Testing

Achieving accurate and well-tested results, by applying a well-functioning code.

20/04-24/04

Project Maintenance

Presenting the finished work and recording feedback for further updates.

4. DESIGN

4.1 BACKGROUND

Underwater photography suffers from a significantly visible amount of distortion. That is due to the following two phenomenon:

4.1.1 LIGHT ATTENUATION

Also called light absorption, is due to the existence of different colors. Different colors simply mean that light is traveling at different wavelengths. For example, the blue light travels at a wavelength of 460 to 495 nm, while the red light travels at a wavelength of 620 to 750 nm. Different wavelengths result in varying degrees of attenuation. For example, the red light doesn't travel that far underwater due to its large wavelength while the blue light travels the longest due to its short wavelength, and this is what explains why the bluish tone is very strong in underwater images, as it was shown before in the pictures I captured.

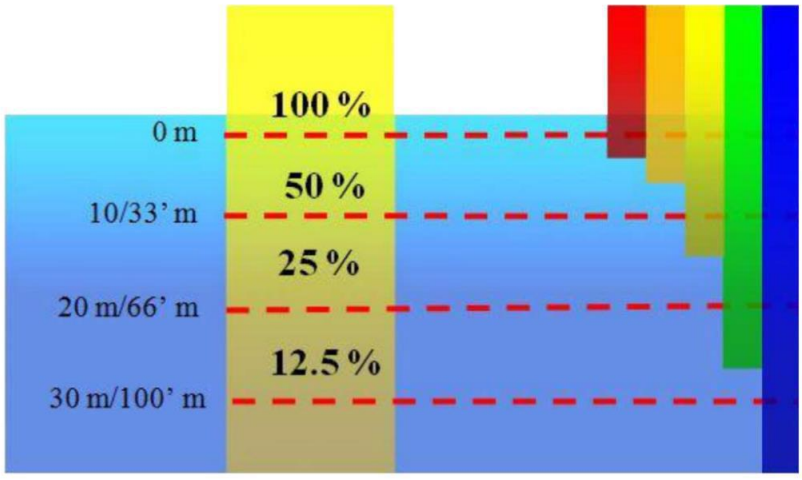

Figure 4.1.1.1 Light Attenuation Underwater

4.1.2 LIGHT SCATTERING

The light coming from the sun, also known as the atmospheric light, is reflected in the solid particles that exist in water such as sand, minerals, and plankton. The light ends up being absorbed by these objects and then re-emitted away in different directions, before reaching the camera. And this what creates the phenomenon of haze. Also due to its short wavelength, the blue color is scattered the most, which explains the strong blue color that exists both in the sky and in the water.

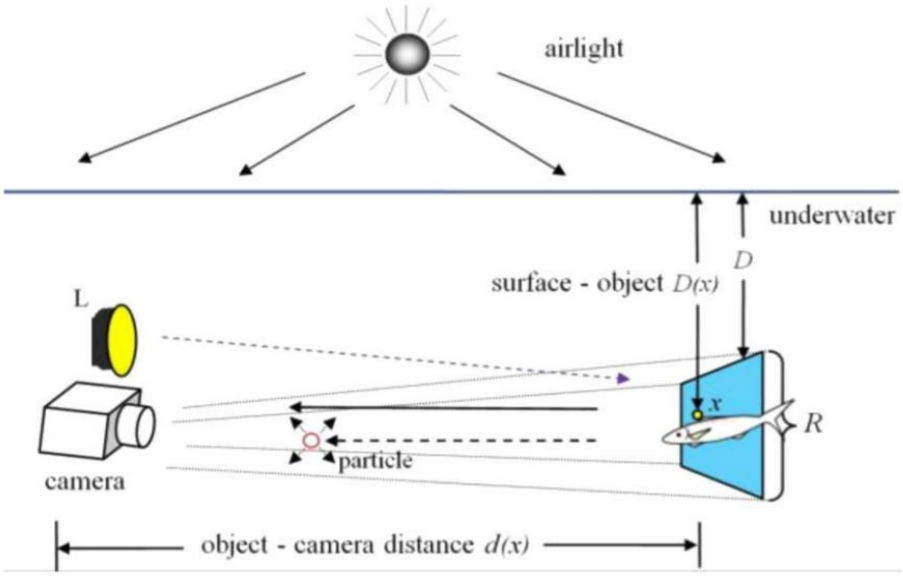

Figure 4.1.2.1 Light Scattering Underwater

This image that I have captured along the sea bed of the Gili Air beach on 28 July 2019, is a strong proof of the link between the intensity of the blue color and the distance of the objects captured.

Figure 4.1.2.2 Light Attenuation over the Length of the Sea Bed

4.2 METHODOLOGY

To ease the understanding of the solution used in the project, and clarify each stepping stone, every step of the methodology will be explained over three levels of abstraction.

- **At the lowest level**, labeled **Theory** I will explain the research and logic behind the method used.
- **At the middle level**, labeled **Execution** I will show the code suggested to implement the theory.
- **At the highest level**, labeled **Result** I will demonstrate the transformation of the input image after the execution.

4.2.1 IMAGE FORMATION

Theory

The image that is received by the camera, is a composition of several elements, among which is the original image that we are trying to deduct. The image formation model, also known as the Duntley model, is a mathematical equation that explains how the light captured by a receptor is formed.

Let **x** be a point in an underwater scene.

I(x) known as the observer intensity, is the degraded underwater image captured by the camera.

J(x) known as the scene radiance, is the original non-degraded underwater image.

t(x) known as the transmittance or the residual energy ratio of J(x), is the amount of light transmitted to the camera after traveling a distance d(x), which means the amount of light received that is nor attenuated neither scattered.

In a homogeneous atmosphere t(x) = e^-Bd(x) given B is the attenuation and scattering coefficient of the underwater medium.

Other sources define t(x) as t(x) = E residual (x) / E initial (x) given E(x) is the residual or initial energy of a point x.

Or, t(x) = Nrer^d(x) Given Nrer is the ratio of residual to initial energy for every unit of distance.

J(x).t(x) known as the attenuation term, is a multiplicative distortion that describes and summarizes the decay (attenuation and scattering) of the scene radiance J(x) in water.

A known as the atmospheric light or the air light is the combination of lights coming from the sun and/or from the camera.

A(1-t) is an additive distortion that describes the light that results from scattering by particles in water.

We assume that the light in the discussion can be of any of the existing wavelengths (red, green, and blue). This assumption serves to avoid writing an extra parameter in the equation which is the wavelength.

The formula goes like this:

$$I(x) = J(x)\,t(x) + A\,(1 - t(x))$$

It should be clear by now, that to solve this problem we have to recover J(x) from the equation.

Given I(x) as the only variable inputted by the user, we still need to estimate t(x), A, and finally J(x).

For I(x) that is an N-pixel image, there are 3N constraints and 4N + 3 unknowns which makes this problem very ambiguous.

Execution

At this point in the program, no execution is needed except reading the underwater degraded image that will be represented by I.

```
global w,bi,gi,ri, folder, source, image
w= 15 #Window size
bi,gi,ri=0,1,2 #Color channels indexes

#Reading image
print ('Reading Image...')
folder = "C:/Users/Salmane/Documents/" #Folder of reading/writing
source = "Philippines.jpeg" #Name of the file
image = cv2.imread(folder + source)

#Converting image into numpy array with 3 channels
i=np.asarray(image, dtype=np.float64)
i=i[:, :, :3]/255
```

Figure 4.2.1.1 Code of Reading an Image

Result

The picture that I will be running the program on is a picture of sardine fishes that I have taken on a discovery trip in the reefs of Moalboal, Cebu in the Philippines on 20 August

2019. I have chosen this picture due to the clarity and minimum movement of the living creatures captured. The original image read by the program is the following:

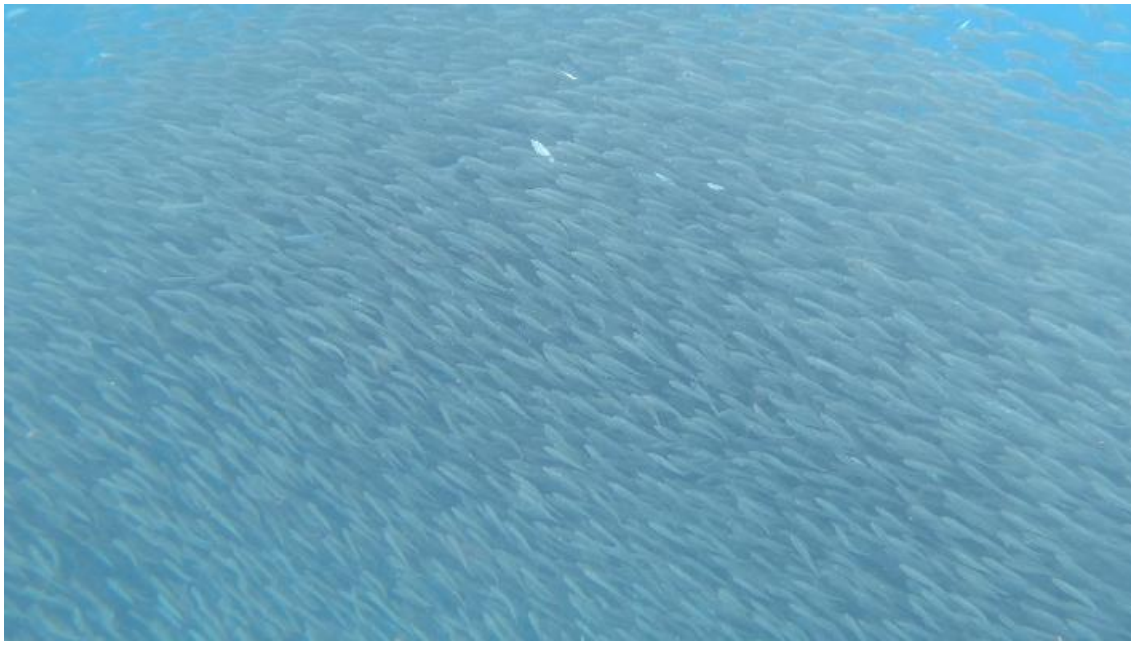

Figure 4.2.1.2 Result of Reading an Image

4.2.2 DARK CHANNEL

Theory

In an image without fog and sky areas, at least one of the three channels (R, G, B) has some pixels with a very low intensity close to 0. The dark channel of an image is simply the combination of the pixels with the lowest intensity from all the three color channels of the image. It can be represented through the following equation:

$$J^{\text{dark}}(x) = \min_{y \in \Omega(x)} \left(\min_{c \in \{r,g,b\}} J^c(y) \right),$$

Given Omega as the group of pixels in a scene, and C as the color channel.

The dark channel prior is the observation that if J is an outdoor haze-free image, its dark channel intensity tends to be 0 in a range of [0,1]. This low intensity is normally due to shadows, colorful objects, or dark objects.

The dark channel prior does not function when there is fog, because the atmospheric particle size is larger than the wavelength of the visible light, therefore the attenuation effect is more obvious than the scattering effect. The degree of scattering in air and water is similar, however in water, attenuation is much stronger. Consequently, the red channel intensity will be low and the image will be predominantly blue and/or green. And this is why the dark channel method works better in air, but not on degraded underwater images.

This method will not be used in my program, for the reasons aforementioned, and was shown solely for the purpose of clarifying its similarities and differences from the bright channel method that will be introduced next.

Execution

```python
def dark_channel(i):
    M, N, _ = i.shape #Dimensions of the image array

    #Padding an array with the values of i
    padded = np.pad(i, ((int(w/2), int(w/2)), (int(w/2), int(w/2)), (0, 0)), 'edge')

    #Creating the dark channel array
    dark = np.zeros((M, N))

    #Selecting the lowest intensity pixels out of the three channels of i
    for i, j in np.ndindex(dark.shape):
        dark[i, j] = np.min(padded[i:i + w, j:j + w, :])

    return dark
```

Figure 4.2.2.1 Code of Creating a Dark Channel Image

Result

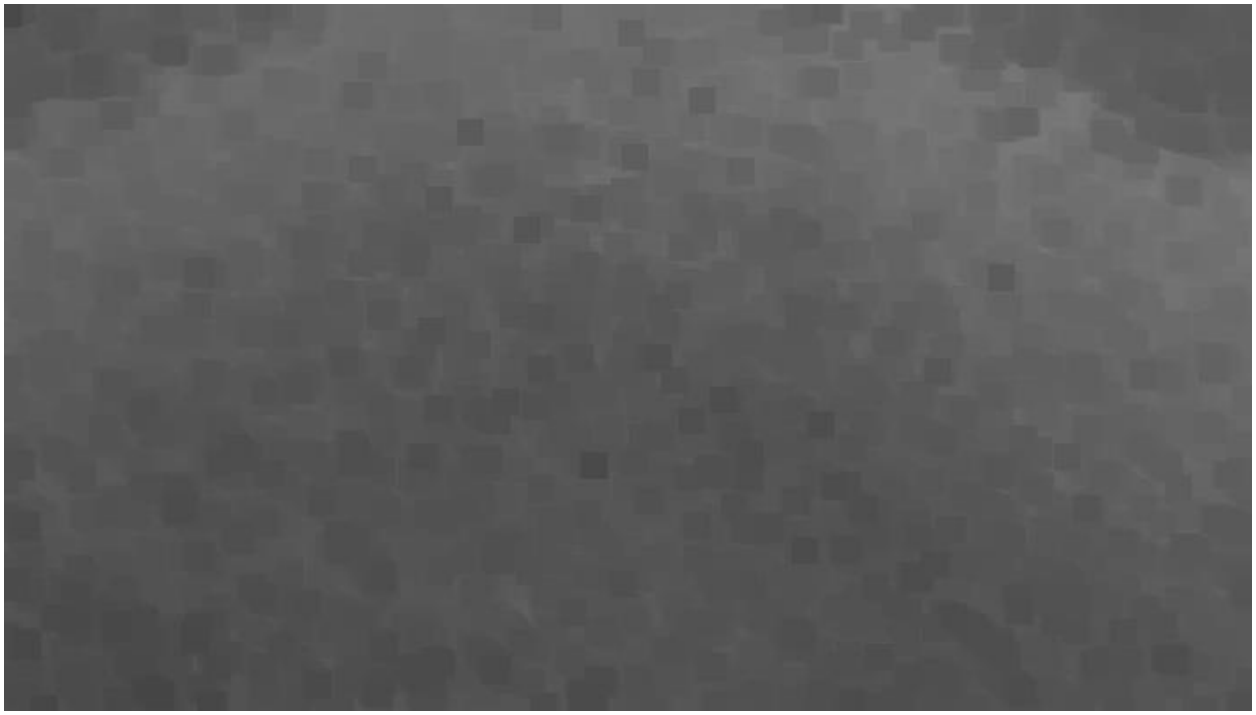

Figure 4.2.2.2 Result of Creating a Dark Channel Image

4.2.3 BRIGHT CHANNEL

Theory

As opposed to the dark channel shown previously, the bright channel of an image is the combination of the pixels with the highest intensity from all the three color channels of the image. It can be represented through the following equation:

$$J^{\text{newlight}}(x) = \max_{y \in \Omega(x)} \left(\max_{c \in \{r, g^{\text{new}}, b^{\text{new}}\}} J^{\text{new}c}(y) \right).$$

The bright channel prior is the observation that if J is an underwater image without pure water areas and distant scenes, its bright channel intensity tends to be 1 in a range of [0,1]

Execution

```
def bright_channel(i):
    M, N, _ = i.shape #Dimensions of the image array

    #Padding an array with the values of i
    padded = np.pad(i, ((int(w/2), int(w/2)), (int(w/2), int(w/2)), (0, 0)), 'edge')

    #Creating the bright channel array
    bright = np.zeros((M, N))

    #Selecting the highest intensity pixels out of the three channels
    for i, j in np.ndindex(bright.shape):
        bright[i, j] = np.max(padded[i:i + w, j:j + w, :])

    return bright
```

Figure 4.2.3.1 Code of Creating a Bright Channel Image

Result

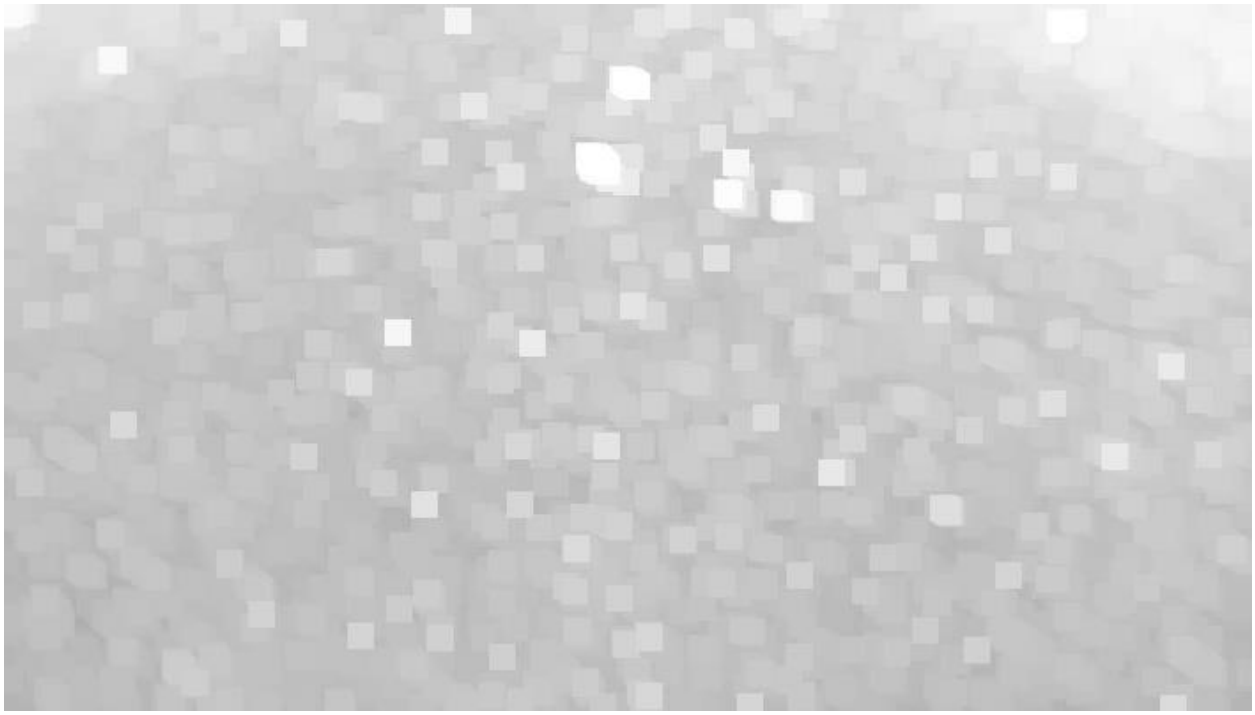

Figure 4.2.3.2 Result of Creating a Bright Channel Image

4.2.4 MAXIMUM COLOR DIFFERENCE

Theory

The maximum color difference image referred to as bgsubr(x) is an image representing the maximum difference of intensity between the three color channels of an image. The more distance separating the object captured from the camera, the bigger the difference will be since the red light attenuates fast with the increase of distance. The maximum color difference will not be used directly in the process of transforming the input image but will serve indirectly into rectifying the bright channel as I will explain next.

$$\text{bgsubr}(x) = 1 - \max\left(\max\left(C_{\max}(x) - C_{\min}(x), 0\right), \max\left(C_{\text{mid}}(x) - C_{\min}(x), 0\right)\right),$$

Execution

```
def bgsubr(i, bright):
    M, N = bright.shape #Dimensions of the image array

    #Getting the indexes of the maximum, medium and minmum color channels
    cmax, cmid, cmin, _,_,_,_,_ = channel_intensities(image)

    #Creating the maximum color difference array
    bgsubr = np.zeros((M, N))

    #Seprating i into the three color channels accordingly
    arrcmax = i[...,cmax]
    arrcmid = i[...,cmid]
    arrcmin = i[...,cmin]

    #Calculating the maximum channel difference in each pixel
    for mi in range(M):
        for ni in range(N):
            bgsubr[mi][ni] = 1 - max(max(arrcmax[mi][ni]-arrcmin[mi][ni], 0), max(arrcmid[mi][ni]-arrcmin[mi][ni],0))

    return bgsubr
```

Figure 4.2.4.1 Code of Creating a Maximum Color Difference Image

```
def channel_intensities(image):

    #Reading the 3 channels of the image
    b, g, r = cv2.split(image)

    #Calculating the mean intensity of each channel
    t = image.size / 3
    bx = float(np.sum(b)) / t
    gx = float(np.sum(g)) / t
    rx = float(np.sum(r)) / t

    #Identifying the indexes of the maximum, medium and minimum channel intensities
    var={bx:bi,gx:gi,rx:ri}
    cmax=var.get(max(var))
    cmin=var.get(min(var))
    if ((cmax==1 or cmax==2) and (cmin==1 or cmin==2)):
        cmid =0
    if ((cmax==0 or cmax==2) and (cmin==0 or cmin==2)):
        cmid =1
    if ((cmax==0 or cmax==1) and (cmin==0 or cmin==1)):
        cmid =2

    return cmax,cmid,cmin, bx,gx,rx, b,g,r
```

Figure 4.2.4.2 Code of Calculating the Different Channel Intensities and Means

Result

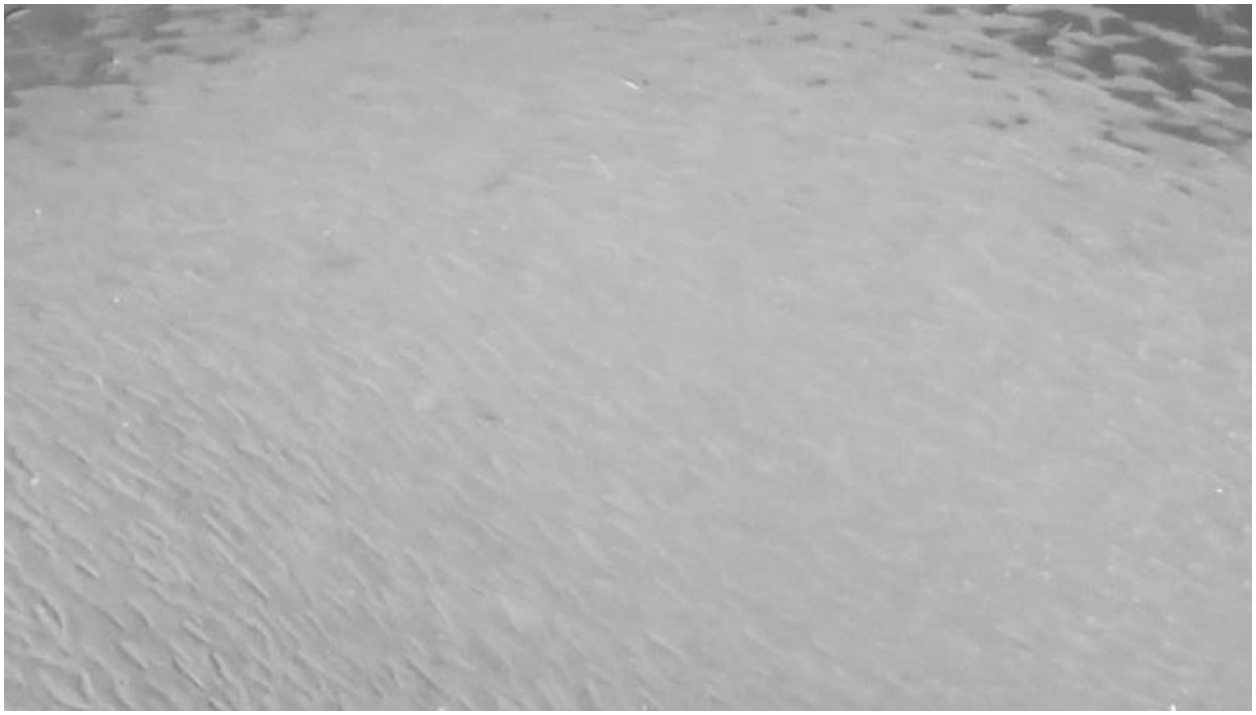

Figure 4.2.4.3 Result of Creating a Maximum Color Difference Image

4.2.5 RECTIFYING THE BRIGHT CHANNEL

Theory

As it will be demonstrated below, the transmittance of light t(x) is linearly proportional to the bright channel image I bright. Therefore, the calculated transmittance will be smaller than usual assuming that the bright channel intensity is close to 1. This necessitates rectifying the

bright channel image using the following logic:

$$\text{light}_{\text{correct}}(x) = \lambda * \text{light}(x) + (1 - \lambda) * \text{bgsubr}(x),$$

Given that lambda is a proportional coefficient that is larger than 0.5, lambda can be calculated by finding the pixel with the maximum saturation level.

$$\lambda = \max(\max(S)),$$

Given S is the saturation channel image of the degraded underwater image in an HSV color space

Execution

```
def rectify_bright(bgsubr):

    #Calculating the saturation channel of the image
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    hsv = cv2.cvtColor(rgb, cv2.COLOR_RGB2HSV)

    #Calculating the coefficient lambda
    lambd = (hsv[...,1].max())/255

    #Calculating the rectified bright channel
    ibright = (bright*lambd) + (bgsubr*(1-lambd))

    return ibright
```

Figure 4.2.5.1 Code of Rectifying a Bright Channel Image

Result

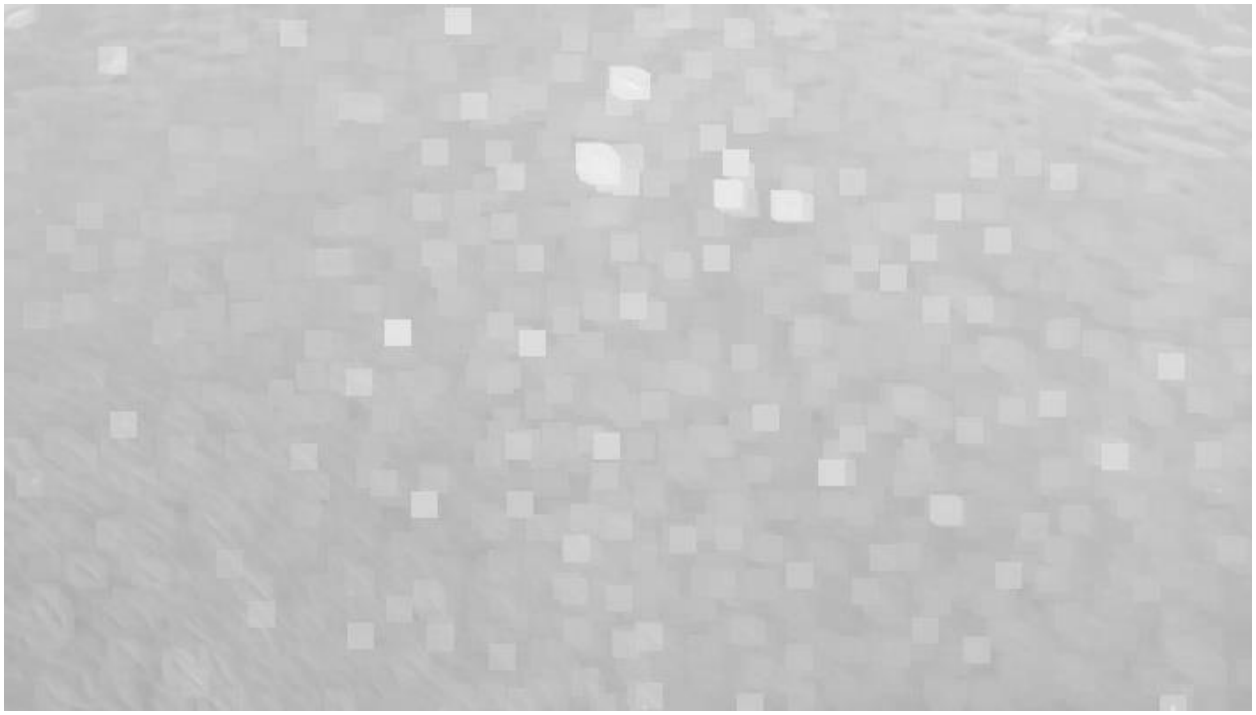

Figure 4.2.5.2 Result of Rectifying a Bright Channel Image

4.2.6 ATMOSPHERIC LIGHT

Theory

After successfully rectifying the bright channel version of I, we calculate the atmospheric light A.

First, we extract the gray version of the input image to produce the variance image V. For each pixel in the gray image we compute its variance within a block which centers this pixel. For purposes of smooth and rapid performance, I picked the size of the block to be 3*3 = 9

pixels which are the smallest block possible to form around the pixel. The variance calculated represents the evenness of the block.

Second, we pick the top 1% darkest pixels in the bright channel I.

Third, we select among these dark pixels the pixel with the lowest variance value. The intensity of this pixel over the three color channels is the atmospheric light.

Execution

```
def atmospheric_light(i, ibright):
    M, N = ibright.shape #Dimensions of the rectified bright channel array
    at = np.empty(3) #Atmospheric light
    selectvar = [] #Array used to get the variance of the darkest pixels

    #Storing the 3D input array into 2D array
    flati = i.reshape(M*N, 3)

    #Extracting the gray filter image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = np.float64(gray)/255

    #Producing the variance within a block for each pixel from gray image
    win_var = nd.generic_filter(gray, np.var, size = 3)
    minvar=256

    #Finding the top 1%*M*N darkest pixels
    flatbright = ibright.ravel()
    top = heapq.nsmallest(int(M*N*0.1),flatbright)

    #Finding the dark pixel with the miminmum variance intensity
    a = np.where(np.isin(ibright, top))
    for n in range (0, len(a[0])):
        (b,c) = (a[0][n], a[1][n])
        selectvar.append(win_var [b,c])
        if (minvar>np.amin(selectvar)):
            minvar = np.amin(selectvar)
            ib, ic = b,c
            if(minvar == 0): break

    #Getting the atmospheric light intensity
    at[0] = i[ib,ic,0]
    at[1] = i[ib,ic,1]
    at[2] = i[ib,ic,2]

    return at
```

Figure 4.2.6.1 Code of Extracting the Atmospheric Light

Result

```
Processing atmospheric light...
Atmospheric light: [0.67843137 0.57254902 0.36078431]
```

Figure 4.2.6.2 Result of Extracting the Atmospheric Light

4.2.7 INITIAL TRANSMITTANCE IMAGE

Theory

The only variable left to estimate, after knowing I(x) and A is t(x)

Given the image formation model:

$I(x) = J(x)t(x) + A(1-t(x))$

We divide by the constant A and we apply the maximum pixel out of maximum channel operators over both sides of the equation. Assuming t is constant in a scene, we get:

$$\frac{\max_{y\in\Omega(x)}\left(\max_{c\in\{r,g^{\text{new}},b^{\text{new}}\}} I^{\text{new}c}(y)\right)}{A^{\text{new}c}}$$

$$= t(x)\,\frac{\max_{y\in\Omega(x)}\left(\max_{c\in\{r,g^{\text{new}},b^{\text{new}}\}} J^{\text{new}c}(y)\right)}{A^{\text{new}c}} + 1$$

$$- t(x).$$

This leads to **max J(x) → 1**, therefore:

$$\tilde{t}(x) = \frac{1}{1 - A^{\text{new}c}} \max_{y\in\Omega(x)} \left(\max_{c\in\{r,g^{\text{new}},b^{\text{new}}\}} I^{\text{new}c}(y) \right)$$

$$- \frac{A^{\text{new}c}}{1 - A^{\text{new}c}}.$$

Therefore:

$$\tilde{t}^{c}(x) = \frac{\left(\text{light}_{\text{correct}}(x) - A^{c}\right)}{1 - A^{c}},$$

It should be noted that we have to calculate the transmittance image over each channel and solve for its average value, to get the final initial transmittance.

Execution

```
def initial_transmission(a, ibright):
    M, N = ibright.shape #Dimensions of the image array
    init = np.zeros((M,N)) #Initial transmittance

    #Calulating the transmittance over each channel
    for i in range(3):
        init = init + ((ibright-a[i])/(1.-a[i]))
        init = (init - np.min(init))/(np.max(init) - np.min(init))
    init = init/3

    #Calculating the average value of the transmittance
    return (init - np.min(init))/(np.max(init) - np.min(init))
```

Figure 4.2.7.1 Code of Extracting the Initial Transmittance

Result

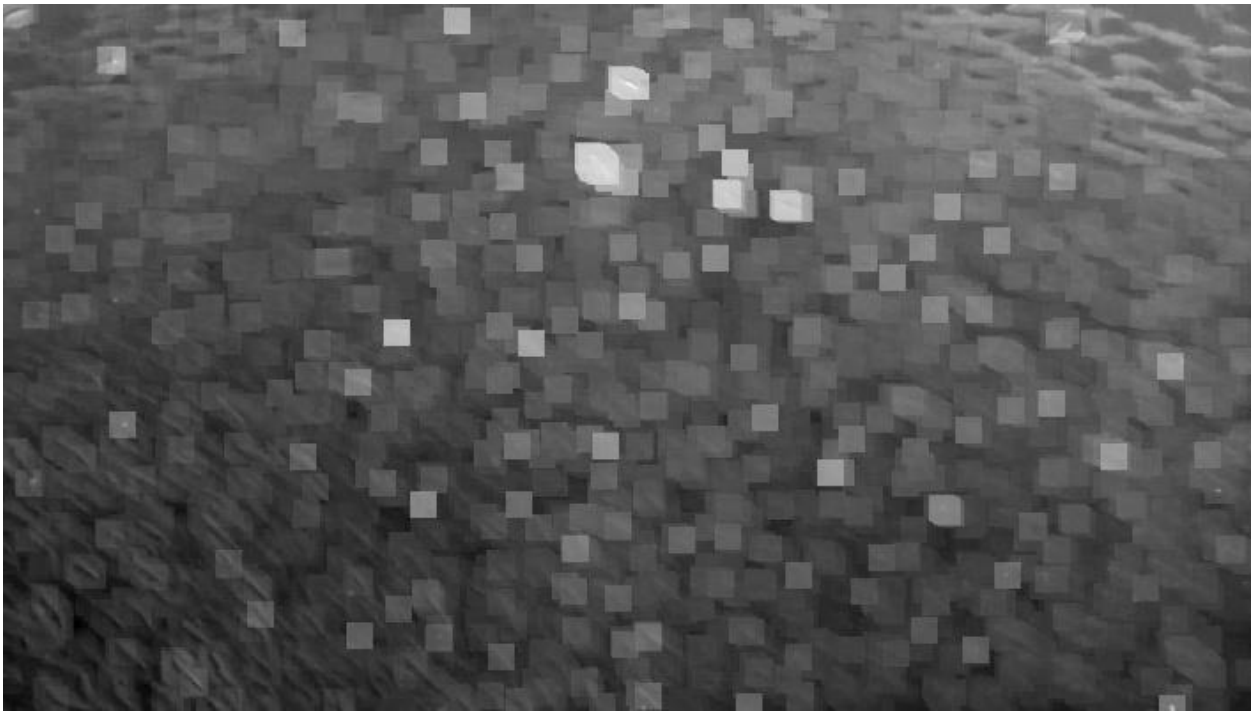

Figure 4.2.7.2 Result of Extracting the Initial Transmittance

4.2.8 REFINED TRANSMITTANCE IMAGE

Theory

The initial transmittance image demonstrates the existence of defects. To refine the image, we use the gray version of the input image as a guide image, and the initial transmittance image as an input image to perform the guided filter function.

Execution

```
def refined_transmission(init):
    refined = np.full_like(init, 0)

    #Extracting the gray filter image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = np.float64(gray)/255


    refined = guidedFilter(gray, init)

    return refined
```

Figure 4.2.8.1 Code of Refining the Transmittance

Result

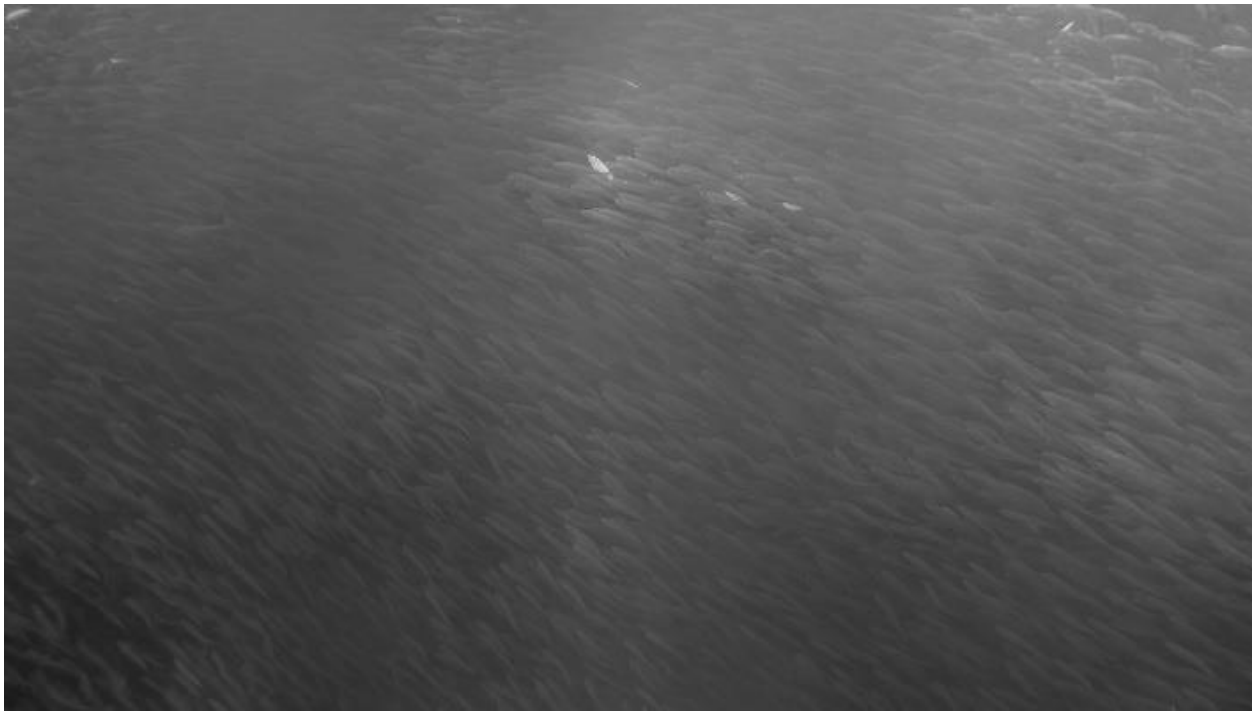

Figure 4.2.8.2 Result of Refining the Transmittance

4.2.9 RESTORED IMAGE

Theory:

After succeeding in estimating all the needed variables, it is possible at this point to restore the original image J, following this equation, for each channel:

$$J(x) = I(x) - (A / \text{refined } t(x)) + A$$

Execution

```
def restoration_image(i, a, refined):
    M, N, _ = i.shape

    #Broadcasting the refined transmission into a 3D array
    corrected = np.broadcast_to(refined[:,:,None], (refined.shape[0], refined.shape[1], 3))

    #Restoring the original image
    j = ((i-a)/corrected) + a

    return j
```

Figure 4.2.9.1 Code of Restoring the Image

Result

Figure 4.2.9.2 Result of Restoring the Image

4.2.10 HISTOGRAM EQUALIZATION

Theory

After completing the process of image restoration, it is now possible to carry on with the process of image enhancement. One of the effective methods to reduce color distortion and improve the contrast of the image is histogram equalization.

Histogram equalization is an image processing technique that aims to enhance the contrast in images by stretching the intensity range of the image, as the example below demonstrates.

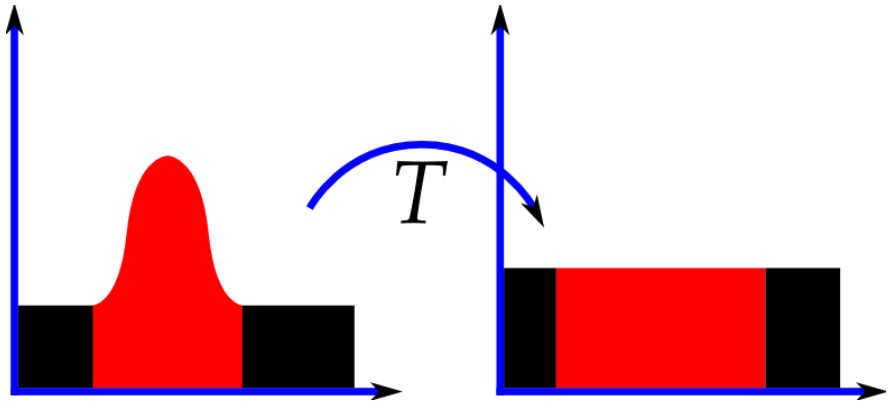

Figure 4.2.10.1 Histograms Before and After Equalization

For this purpose, I found the method followed by G. Anbarjafari in “HIS based color image equalization using iterative nth root and nth power” to be simple and effective. Moreover,

unlike the method used by Y. Gao, H. Li, and S. Wen in “Restoration and enhancement of underwater images based on bright channel prior” which only equalizes the pixels whose intensity is smaller than the mean intensity of their corresponding color channel, it takes into account all the pixels of the image which enhances the overall visual quality of the image.

The method described by G. Anbarjafari consists of an iterative algorithm based on separating the image to its three RGB channels and raising all the pixels of the channel to a specific power until the respective mean of this channel becomes equal to the wanted value. The diagram below resumes the method used:

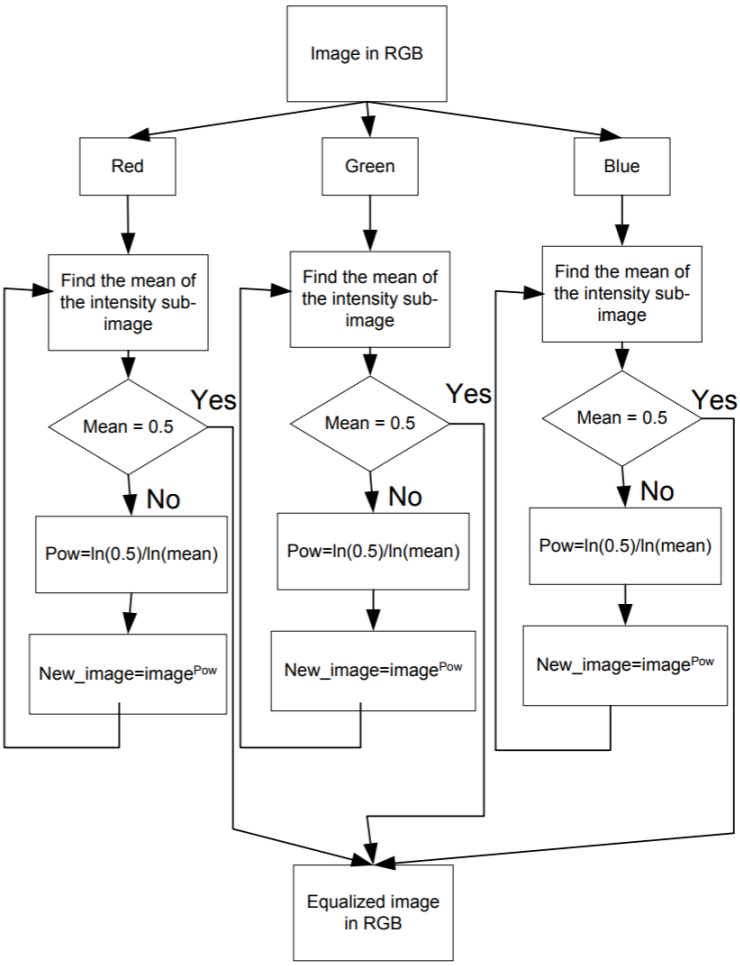

Figure 4.2.10.2 Histogram Equalization Suggested Algorithm

Assuming the intensities used in this method are converted to a [0,1] range instead of a [0,255] range, the mean of 0.5 is the standard mean used in this paper. A mean of 0.5 appeared, after I conducted several experiments on different images, to be the optimal mean for preventing a super bright image (mean → 1) and preventing a super dark image (mean → 0). Nevertheless, It is possible and even recommended by Y. Gao, H. Li, and S. Wen in their paper, manually alter the mean of each channel in the program if the final picture doesn't seem natural.

Execution

```
def histogram_equalization(j):
    M, N, _ = j.shape #Dimensions of the restored image

    #Creating the means of the color channels
    bluemean, greenmean, redmean = float("%0.5f" % (2)), float("%0.5f" % (2)) ,float("%0.5f" % (2))

    #Handling the case of negative pixels
    for mi in range(M):
        for ni in range(N):
            if (j[mi,ni,0] <= 0):
                j[mi,ni,0] = 0
            if (j[mi,ni,1] <= 0):
                j[mi,ni,1] = 0
            if (j[mi,ni,2] <= 0):
                j[mi,ni,2] = 0

    #Getting the means and arrays of each channel
    _, _, _, b,g,r, barr,garr,rarr = channel_intensities(j*255)

    #Converting the intensity range to [0,1]
    barr=barr/255
    garr=garr/255
    rarr=rarr/255

    #Assigning the wanted means from each channel
    bidx=0.5
    gidx=0.5
    ridx=0.5
```

Figure 4.2.10.3 Code of Histogram Equalization #1

```
#Equalizing the blue channel
if (bidx>0):
    bint = float("%0.5f" % (bidx))
    while bluemean != bint:
        bluemean = float("%0.5f" % (float((np.sum(barr))) / (M*N)))
        powb = np.log(bint)/np.log(bluemean)
        barr = (barr) ** (powb)

#Equalizing the green channel
if (gidx>0):
    gint = float("%0.5f" % (gidx))
    while greenmean != gint:
        greenmean = float("%0.5f" % (float((np.sum(garr))) / (M*N)))
        powg = np.log(gint)/np.log(greenmean)
        garr = (garr) ** (powg)

#Equalizing the red channel
if (ridx>0):
    rint = float("%0.5f" % (ridx))
    while redmean != rint:
        redmean = float("%0.5f" % (float((np.sum(rarr))) / (M*N)))
        powr = np.log(rint)/ np.log(redmean)
        rarr = (rarr) ** (powr)

#Combining the three channels into the new restored image
for mi in range(M):
    for ni in range(N):
            j[mi,ni,0]=barr[mi,ni]
            j[mi,ni,1]=garr[mi,ni]
            j[mi,ni,2]=rarr[mi,ni]

return j
```

Figure 4.2.10.4 Code of Histogram Equalization #2

Result

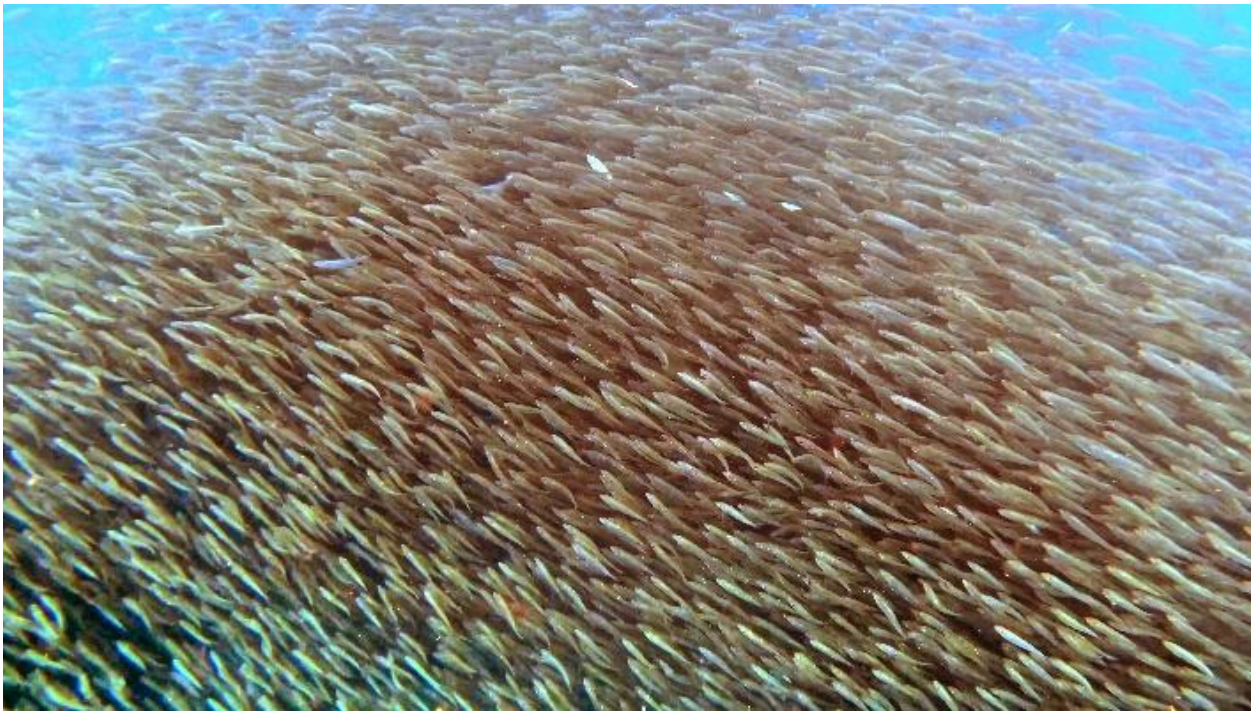

Figure 4.2.10.5 Result of Histogram Equalization

4.3 ALGORITHM

To summarize the theory aspect of this project, the diagram below resumes all the steps described to achieve the original colors of a degraded underwater image.

First, we input the degraded underwater image taken by the camera.

Second, we extract the bright channel and the maximum color difference versions of the image.

Third, we use both of these pictures to rectify the bright channel image.

Fourth, we use the bright channel image to estimate the value of the atmospheric light and then use both the bright channel image and the atmospheric light to obtain the initial transmittance image and to refine it.

Fifth, we extract the restored image using all of the rectified bright channel image, the atmospheric light, and the refined transmittance image.

And last, we apply the histogram equalization method on the restored image to obtain the final result.

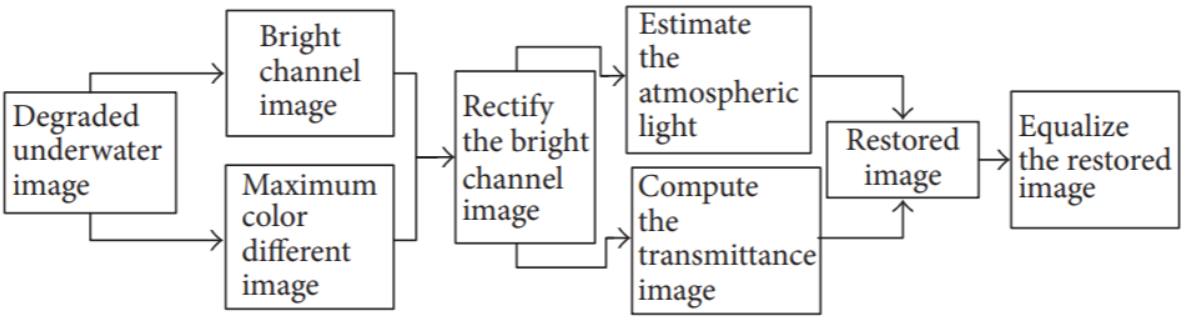

Figure 4.3.1 Algorithm of Bright Channel Method

5. IMPLEMENTATION

To summarize the software execution aspect of this project, the screenshot below demonstrates the main function of my program, which combines under a specific order, all the sub-functions explained earlier.

```
if __name__ == '__main__':

    global w,bi,gi,ri, folder, source, image
    w= 15 #Window size
    bi,gi,ri=0,1,2 #Color channels indexes

    #Reading image
    print ('Reading Image...')
    folder = "C:/Users/Salmane/Documents/" #Folder of reading/writing
    source = "Philippines.jpeg" #Name of the file
    image = cv2.imread(folder + source)

    #Converting image into numpy array with 3 channels
    i=np.asarray(image, dtype=np.float64)
    i=i[:, :, :3]/255

    #Priting the image dimensions
    height, width, _ = i.shape
    print ('Image dimensions: ( Height:', height, ', Width:', width, ')')
    if (height >600) and (width >600):
        print('Your source image is large. The program might risk running for a long time.')
    print('\n')

    #Writing the input image
    cv2.imwrite(folder + 'original_image.jpeg', image)


    #Getting the bright channel
    print ('Processing bright channel...')
    bright=bright_channel(i)
    cv2.imwrite(folder + 'bright_channel.jpeg', bright*255)
    print('Bright channel saved successfully')
    print('\n')


    #Getting the bright channel
    print ('Procession the bright channel rectification...')
    bgsubr = bgsubr(i, bright)
    ibright = rectify_bright(bgsubr)
    cv2.imwrite(folder + 'rectified_bright.jpeg', ibright*255)
    print('Rectified bright channel saved successfully')
    print('\n')
```

Figure 5.1.1 Code of Main Function #1

```
#Getting the atmospheric light
print ('Processing atmospheric light...')
a = atmospheric_light(i, ibright)
print ('Atmospheric light: {}'.format(a))
print('\n')

#Getting the initial transmission
print ('Processing initial transmission...')
init = initial_transmission(a, ibright)
white = np.full_like(bright, 255)
cv2.imwrite(folder + 'initial_transmission.jpeg', init*white)
print('Initial transmission saved successfully')
print('\n')

#Getting the refined transmission
print ('Processing refined transmission...')
refined = refined_transmission(init)
cv2.imwrite(folder + 'transmission_corrected.jpeg', refined*white)
print('Refined transmission saved successfully')
print('\n')

#Getting the restored image
print ('Processing image restoration...')
j = restoration_image(i, a, refined)
cv2.imwrite(folder + 'restored_image.jpeg', j*255)
print('Restored image saved successfully')
print('\n')


#Getting the histogram equalization
print ('Processing histogram equalization...')
result = histogram_equalization(j)
cv2.imwrite(folder + 'final_result.jpeg', result*255)
print('Final result saved successfully')
print('\n')
```

Figure 5.1.2 Code of Main Function #2

6. TESTING

To go more in detail with the visual aspect of this project, I test different pictures on the same program to ensure its validity.

As mentioned in section 2, underwater image restoration is a field that has known various methods and different results. The reason behind this is that it is difficult and even nearly impossible to measure the parameters, like the transmittance image, in an absolute and totally precise manner. Therefore it is perfectly natural to expect different results for different methods and with different input pictures.

Although testing the efficiency of the program professionally, and guarantying accurate results, requires sticking by hard facts and having reliable references to resort to be able to judge. The field of underwater image processing knows

> "no ground truth or uniform measure standard available". [18, Page 8]

Nevertheless, each author resorts to different references to establish the legitimacy of their method. The references that I resorted to, in order to ensure the validity of my project are the following:

6.1 TESTING BY COMPARISON TO A COLOR BOARD

Due to the current circumstances of the quarantine, my initial plan of relying on conducting personal experiments in the Olympic swimming pool at Al Akhawayn University had to be ruled out. As inspired by J. Chiang and Y. Chen in their paper "Underwater Image

Enhancement by Wavelength Compensation and Image Dehazing", the purpose of the experiment was to take a picture of a board with six color patches before diving and after diving underwater at a specific depth, and trying to compare the output of the program with the original colors of the board captured before diving. This leads me to satisfy with the next testing methods.

Figure 6.1.1 Color Board before Diving, at a Normal Depth, and Deep Depth

6.2 TESTING BY COMPARISON TO A CLOSE DISTANCE OBJECT

A similar alternative to comparing with color patches is to compare the colors of the objects taken underwater, to the same objects but at a significantly close distance from the camera. Following the theory explained in Section 2 and 4, at a large distance the phenomenon of light attenuation and scattering increase, leading to painting the original colors with a blue filter. However, at a very small distance, the impact of these phenomena tend to be negligible.

Looking through my files, I found a video, taken in the same environment of the ones in the introduction, that captures a sea bed full of orange coral plants both from a very close distance (around 20 cm) and long-distance (around 3 meters). Applying the restoration program on the picture from the long-distance leads to the following result

Figure 6.2.1 Degraded Underwater Image of the Coral Plants from a Long Distance

Extracting the color form the orange coral plant, using Paint.net 4.1.6 shows the following information:

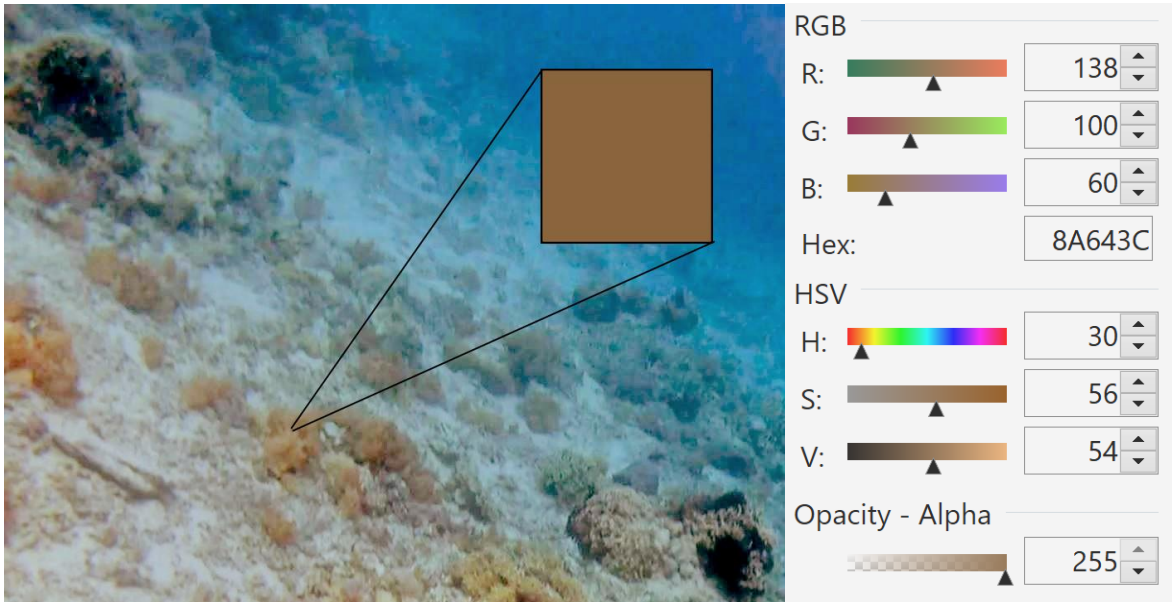

Figure 6.2.2 Restored and Enhanced Image of the Coral Plants from a Long Distance

In parallel, extracting the color from the orange coral plant in the picture from a close distance shows the following information:

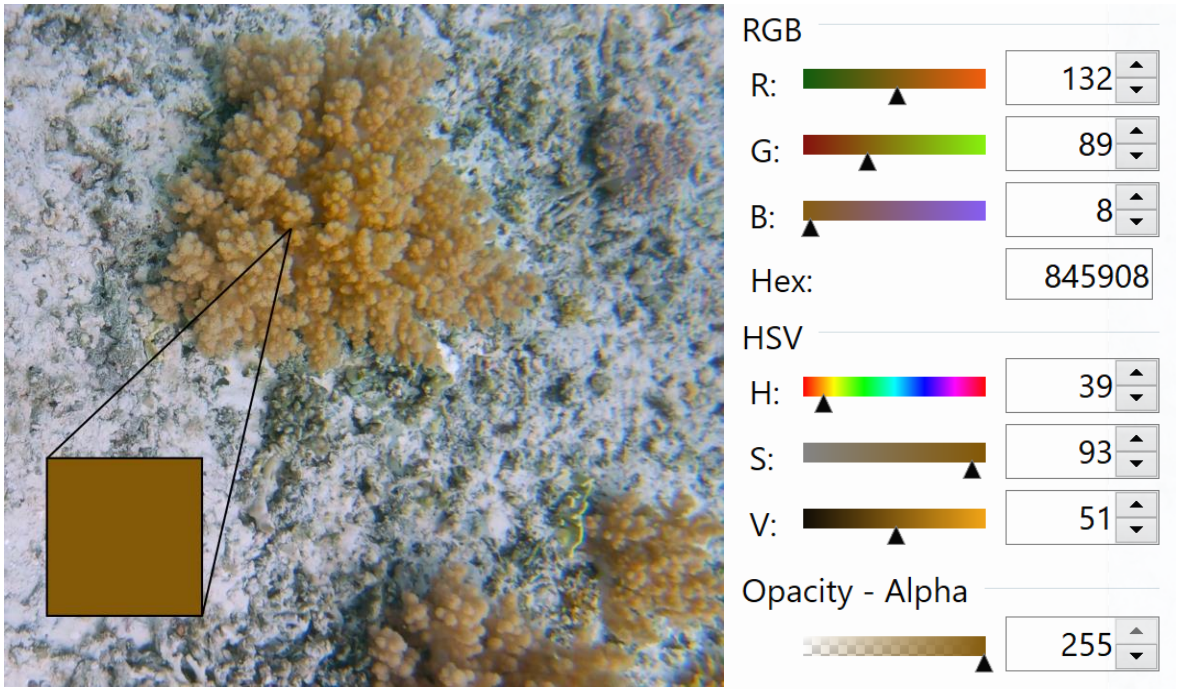

Figure 6.2.3 Clear Underwater Image of the Same Coral Plants from a Short Distance

As it appears, there is a significant similarity between the two colors which can tell us that, on a visual aspect, the image restored by the program is relatively similar to the real-life scene captured.

6.3 TESTING BY VISUAL SATISFACTION

The last resort that can be employed to ensure valid results is to simply settle for visual judgment. As the example below shows, noticing the blue color of the small fish, the orange color of the coral plants, and the brown color of the rocks, are all factors that conform to the common sense of the viewers and can vouch for the validity of the program. Nevertheless,

this testing method is not accurate, as the visual perception is subjective among the viewers and not absolute and objective for all kinds of images under all conditions.

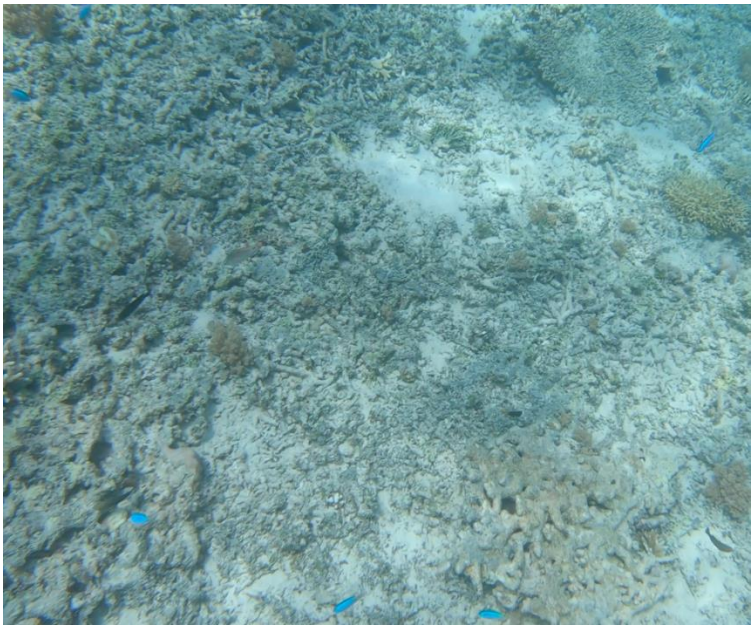

Figure 6.3.1 Degraded Underwater Image of Sea Bed

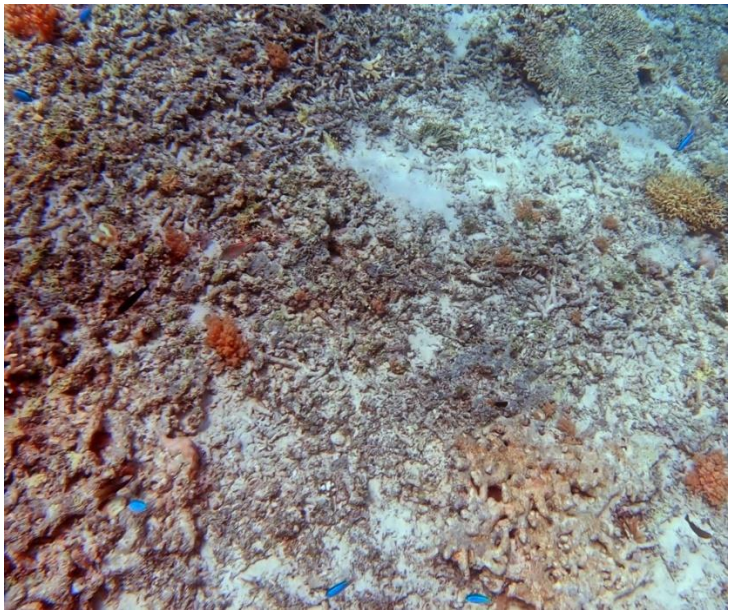

Figure 6.3.2 Restored and Enhanced Image of Sea Bed

7. MAINTENANCE

7.1 CORRECTION

Maintaining the project has been, first, accomplished simultaneously along with the development of the code. Corrections have been made on the methods used, functions selected and coefficients chosen based on the similarity of the result achieved by me to the result achieved by Y. Gao, H. Li, and S. Wen on the guide paper. The implementation of the program has been completed once a satisfactory result, which is similar enough to the result shown by the paper, has been achieved.

After finishing the implementation of the program, and after making enough research to learn about the testing methods possible, the testing methods aforementioned has been operated on a large number of pictures to correct any missing details in the program.

7.2 FUTURE PERSPECTIVES

This project is still in its initial stages and needs more future work to be perfect. The future work that will be accomplished will consist of:

7.2.1 ENHANCING THE TESTING METHODS

- Testing by comparison to a color board after the end of the quarantine period.

- Testing by numerical methods, rather than visual ones, by doing research and searching for an accurate and representative numerical figure to confirm the accuracy of the result.

7.2.2 LAUNCHING A PUBLIC APPLICATION

Transforming the personal program into a public mobile application on Android and/or iOS. The application would be publicly accessible and downloadable on public application stores as the Play Store and/or Apple Store.

8. CONCLUSION

This project has been very beneficial in assessing and enhancing the prior knowledge I accumulated in my Bachelor's degree both in the Computer science field through courses like Computer Graphics, Algorithms Analysis, and Data Structures, and also the Mathematical field through courses like Linear Algebra and the three levels of Calculus.

Some of the main constraints that have been encountered during the work on this project include selecting the research papers with the right level of difficulty of implementation and understanding. This issue has been met by resorting to my supervisor who guided me through the research step of this project and selecting together the most efficient and implementable method, in regards to the short time of the semester.

Going through all the steps of research, implementation, and testing allowed me to achieve a visually acceptable result following the underwater image restoration using the bright channel method. What can still be improved by future work is applying other types of testing on different inputs to leave no doubts about the accuracy of the program. Nevertheless, the field of underwater image restoration knows different solutions that are all valid despite the differences of the color, saturation, and contrast intensities between the applied methods. This field is still under development and research, and until now there is no absolute ground truth standard to go back to.

9. REFERENCES

[1] S. G. Narasimhan and S. K. Nayar, “Chromatic framework for vision in bad weather,” in Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR ’00), vol. 1, pp. 598–605, Hilton Head Island, SC, USA, June 2000.

[2] S. G. Narasimhan and S. K. Nayar, “Vision and the atmosphere,” International Journal of Computer Vision, vol. 48, no. 3, pp. 233–254, 2002.

[3] R. T. Tan, “Visibility in bad weather from a single image,” in Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’08), pp. 1–8, June 2008.

[4] R. Fattal, “Single image dehazing,” ACM Transactions on Graphics, vol. 27, no. 3, article 72, pp. 1–9, 2008.

[5] K. He, J. Sun, and X. Tang, “Single image haze removal using dark channel prior,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 12, pp. 2341–2353, 2011.

[6] G. Ge, Z. Wei, and J. Zhao, “Fast single-image dehazing using linear transformation,” Optik, vol. 126, no. 21, pp. 3245–3252, 2015.

[7] J. Li, H. Zhang, D. Yuan, and M. Sun, “Single image dehazing using the change of detail prior,” Neurocomputing, vol. 156, pp.1–11, 2015.

[8] Y.-T. Peng, X. Zhao, and P. C. Cosman, “Single underwater image enhancement using depth estimation based on blurriness,” in Proceedings of the IEEE International Conference on Image Processing (ICIP ’15), pp. 4952–4956, September 2015.

[9] C. Ancuti, C. O. Ancuti, T. Haber et al., “Enhancing underwater images and videos by fusion,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’12), pp. 447–456, June 2012.

[10] N. Carlevaris-Bianco, A. Mohan, and R. M. Eustice, “Initial results in underwater single image dehazing,” Oceans, vol. 27, no. 3, pp. 1–8, 2010.

[11] A. Galdran, D. Pardo, A. Picon, and A. Alvarez-Gila, “Automatic ´Red-Channel underwater image restoration,” Journal of Visual Communication & Image Representation, vol. 26, pp. 132–145, 2015.

[12] H. Wen, Y. Tian, T. Huang, and W. Gao, “Single underwater image enhancement with a new optical model,” in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS ’13), pp. 753–756, May 2013.

[13] Z. Chen, H. Wang, J. Shen, X. Li, and L. Xu, “Region-specialized underwater image restoration in inhomogeneous optical environments,” Optik, vol. 125, no. 9, pp. 2090 2098, 2014.

[14] S. Q. Duntley, A. R. Boileau, and R. W. Preisendorfer, “Image transmission by the troposphere I,” Journal of the Optical Society of America, vol. 47, no. 6, pp. 499–506, 1957.

[15] K. He, J. Sun, and X. Tang, “Guided image filtering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 6, pp. 1397–1409, 2013.

[16] S. Bazeille, I. Quidu, L. Jaulin, and J.-P. Malkasse, “Automatic underwater image pre-processing,” in Proceedings of the Caracterisation du Milieu Marin (CMM ’06), pp. 145–152, October 2006.

[17] J. Y. Chiang and Y.-C. Chen, “Underwater Image Enhancement by Wavelength Compensation and Dehazing,” *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1756–1769, 2012.

[18] Y. Gao, H. Li, and S. Wen, “Restoration and Enhancement of Underwater Images Based on Bright Channel Prior,” *Mathematical Problems in Engineering*, vol. 2016, pp. 1–15, 2016.

[19] Anbarjafari, G. (2014). ”HSI based colour image equalization using iterative nth root and nth power,” eprint arXiv:1501.00108

[20] CoveyCS, “How to Do PESTLE Analysis For Software Projects,” *Medium*, 03-Apr-2019. [Online]. Available: https://medium.com/@coveycs/how-to-do-pestle-analysis-for-software-projects-c2841aaec684. [Accessed: 21-Apr-2020].

[21] “Histogram equalization,” *Wikipedia*, 03-Mar-2020. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_equalization. [Accessed: 21-Apr-2020].