UNDERWATER IMAGE PROCESSING AND ENHANCEMENT

Submitted in partial fulfillment of the requirements

for the award of

Bachelor of Engineering degree in Computer Science and Engineering

by

Pennama.Hemasagarreddy (39110764)

S.Keerthivas (39110486)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI - 600119

APRIL - 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **S.KEERTHIVAS (Reg.No:37110486) and PENNAMA.HEMASAGARREDDY (Reg.No – 39110764)** who carried out the project entitled "**UNDERWATER IMAGE PROCESSING AND ENHANCEMENT"** under my supervision from January 2023 to April 2023.

Internal Guide
P.KABITHA,M.E

Head of the department
Dr.L.LAKSHMANAN,M.E.,Ph.D.

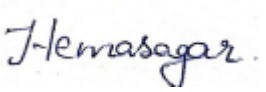
Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I,**PENNAMA.HEMASAGARREDDY (Reg No:39110764),** hereby declare that the Project Phase-2 Report entitled "**UNDERWATER IMAGE PROCESSING AND ENHANCEMENT**" done by me under the guidance of **P.KABITHA,M.E.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering.**



DATE:

PLACE:Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, **Dean**, School of Computing, **and Dr. L. Lakshmanan, M.E., Ph.D., Head of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.P.Kabitha,M.E ,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

ABSTRACT

The oceans contain unknown creatures and vast energy resources, playing an important role in the continuation of life on earth. Hence significant effort has been dedicated worldwide, since the middle of the 20th century, to actively engage in high-tech marine exploration activities.

Restoring clear and real video of underwater image is of great importance to marine ecology, underwater archaeology, underwater biological recognition, and underwater robot vision. Before reaching the camera, the light is reflected and deflected multiple times by particles present in the shallow water.

Thus, extracting valuable information for underwater scenes requires effective methods to correct color, improve clarity and address blurring and background scattering, which is the aim of image enhancement and restoration algorithms. These are particularly challenging due to the complex underwater environment, where images are degraded by the influence of water turbidity and other various factors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This is a general observation that the quality of an image taken with in water is always degraded. It loses the actual tonal quality and the contrast necessary for distinguishing the object of interest present in the image. The situation becomes more challenging when the neighboring objects have very minor differences in pixel intensity values. This situation poses a serious challenge to extract finer details from the data and reduces the performance of the algorithms used to extract information from the images. Therefore, there is a pressing demand for images taken underwater much be processed in such away that they should represent their true tonal details.

Underwater imagery has wide range of applications for example investigation of aquatic life, quality of water, defense and security purposes etc. therefore images or videos obtained for meeting these objectives much carry the exact details. Image enhancement improves an image appearance by increasing dominance of some features or by decreasing ambiguity between different regions of the image. Histogram based image enhancement technique is mainly based on equalizing the histogram of the image and increasing the dynamic range corresponding to the image.

The objectives of this research is to develop an improved single image enhancement technique with is able to work under a variety of conditions and using a standard formulation for assessment of underwater image quality. Specifically, the goals of this research are as follows.

- Analyze and study the effects on light propagation under water and its impacton thedegradation on image quality.
- Comparing the results of the proposed solution using established techniques withother stateof the art solutions to the problem of enhancing underwater images.

Software based underwater image enhancement techniques are usually work by controlling some aspect of the mathematical model of underwater to compensate for the degrading effects introduced by the water‟s light absorption and the presence of organic and inorganic particles in water. Current state of the art method for underwater image restoration are typically designedfor a single image input as using multiple images for the processing usually require more computational resources and may not be suitable for the real-time applications.

Among all Enhancement techniques, Relative Global Histogram Stretching gives better results compared to Rayleigh Distribution and Histogram Equalization. The general principle in RGHS follows two main steps : (i) Contrast Correction and (ii) color correction.

In Rayleigh Distribution Contrast correction, we decompose image and apply Modified VonKreis Hypothesis to it. Then we will stretch histogram and divide into lower histogram and upper histogram based on mid point. Resultant sides we will apply Rayleigh distribution function and then compose it by averaging. Where as, In Color Correction, We will convert RGB model into HSV Model, then S and V components are stretched. Finally we will convert into RGB model. We will get output. But we observe that Red color is more due to saturation. In order to overcome we are using Relative Global Histogram Stretching

In RGHS Contrast Correction, We decompose RGB values and we equalize G and B values only but not R value because in order to prevent saturation state. Then we apply Adaptive stretching and filters. For color correction, We are using CIE-Lab model.

1.1 CHARACTERISTICS OF UNDER WATER IMAGES

Unlike conventional imaging taken above sea in open air, underwater photography shows a strong dominance of bluish and greenish colors. On the other hand, the strong attenuation of light in the water with respect to the air and a greater diffusion of the incident light have the consequence of considerably reducing the visibility.

Thus, objects at distant distances from the acquisition system or the observer but also at medium distances, or even relatively short in some cases, are hardly visible and poorly contrasted with respect to their environment.

In addition, in the presence of particles suspended in water (sand, plankton, algae, etc.), the incident light is reflected by these particles and forms a kind of inhomogeneous mist that addsto the scene observed. This turbidity of the water, most often white, also affects the visibilitybut also the color dynamics of the objects contained in the image by tarnishing or veiling them. On the other hand, the formation of an underwater image is highly dependent on the nature of the water in which it was acquired. Natural waters can have very varied constitutions in terms ofplants or minerals dissolved or suspended in water. The behaviour of the propagation of light in such a medium is strongly governed by this factor**.**

1.2 UNDER WATER MODEL

A well-known haze image function model is often used to approximate the propagation equation of underwater scattering in the background light. The equation is as follows:

$$I_\lambda(x) = J_\lambda(x)\, t_\lambda(x) + (1 - t_\lambda(x))\, B_\lambda(x) \quad \text{----(1.b)}$$

Where the light wavelength $\lambda \in \{red, green, blue\}$, x represents the pixel point in the underwater image $I_\lambda(x)$, $J_\lambda(x)$ is the scene radiance at point x, $t_\lambda(x)$ is the residual energy ratio of after reflecting from point x in the underwater scene and reaching the camera, B_λ is the uniform background light. $J_\lambda(x)\, t_\lambda(x)$ describes the direct attenuation of scene radiance in thewater. Note that the residual energy ratio $t_\lambda(x)$ is a function of both λ and the scene camera distance (x), which reflects the overall effects for both light scattering and color change suffered by light with wavelength traveling the underwater distance $d(x)$. Thus, $t_\lambda(x)$ can be represented as

$$t_\lambda(x) = \text{Nrer}(\lambda)^{d(x)} \quad \text{----------(1.c)}$$

(λ) is the normalized residual energy ratio, which refers to the ratio of residual to initial energy for every unit of distance propagated. As showed in the Fig. 1, where the green and blue lights process shorter wavelength and high frequency thereby attenuates extraordinarily lower than the red counterpart. This is why the deep sea image appears the bluish tone prevalent but the performance of the shallow-water image is not obvious. The dependency of (λ) on the light wavelength can be defined based on that of Ocean Type I as (3).

Where Nrer(λ) = 0.8~0.85 if λ = 650nm ~750nm (red)
= 0.93~0.97 if λ = 490nm ~550nm (green)
= 0.95~0.99 if λ = 400nm ~490nm (blue)

These equations will be considered to decide the maximum range of R- G-B channel in the RGHS.

CHAPTER 2
LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

Zhang, Weidong, et al [1] proposed an efficient and robust underwater image enhancementmethod, called MLLE to locally adjust the color and details of an input image according to aminimum color loss principle and a maximum attenuation map-guided fusion strategy.

Liu, Risheng et al [2] proposed an object-guided twin adversarial contrastive learning based underwater enhancement method to achieve both visual-friendly and task-orientated enhancement. first develop a bilateral constrained closed-loop adversarial enhancement module, which eases the requirement of paired data with the unsupervised manner and preserves more informative features by coupling with the twin inverse mapping.

Zhou et al [3] proposed a visual quality enhancement method for underwater images based on multi-feature prior fusion (MFPF), achieved by extracting and fusing multiple feature priors of underwater images. And designed a color correction method based on self-adaptive standard deviation, which realizes the color offset correction based on the dominant color of the underwater image.

Ding, Xueyan, et al [4] proposed a novel unified total variation method based on an extended underwater imaging mode. In the proposed variational framework, we transform underwater image enhancement into two subproblems and construct different prior knowledge-guided optimization functions for them. The two subproblems aim to remove the light attenuation along the scene-sensor and surface-scene paths.

Jiang, Qun, et al [5] propoed an underwater image enhancement method that does not require training on synthetic underwater images and eliminates the dependence on underwater ground- truth images it transfers in-air image dehazing to real-world underwater image enhancement.

Zhuang, Peixian, et al [6] proposed a hyper-laplacian reflectance priors inspired retinex variational model to enhance underwater images. Specifically, the hyper-laplacian reflectance priors are established with the l1/2 -norm penalty on first-order and second-order gradients of the reflectance. As a result, they turned a complex underwater image enhancement issue into simple subproblems that separately and simultaneously estimate the reflection and the illumination that are harnessed to enhance underwater images in a retinex variational model.

Sun, Kaichuan, et al [7] proposed a progressive multi-branch embedding fusion network (PMEFN) to improve image quality. The distorted images and its sharpened versions are used as the input, which are fused to learn the contextualized features based on a two-branch hybrid encoder–decoder module () combined with the triple attention module to focus on the noise region. They used the multi-stage refining framework to decompose the image enhancement or marine snow removal tasks into multiple stages and progressively learn the nonlinear functions from the distorted inputs.

Kang, Yaozu, et al [8] presented a perception-aware decomposition and fusion framework for underwater image enhancement (UIE). A general structural patch decomposition and fusion (SPDF) approach is introduced. SPDF is built upon the fusion of two complementary pre- processed inputs in a perception-aware and conceptually independent image space. First, a raw underwater image is pre-processed to produce two complementary versions including a contrast-corrected image and a detail-sharpened image. Then, each of them is decomposed into three conceptually independent components, i.e., mean intensity, contrast, and structure, via structural patch decomposition (SPD).

Sharma, Shobha, and Tarun Varma. [9] proposed two new methods based on graph signal processing (GSP) techniques to enhance underwater images. The proposed schemes utilizethe graph Fourier transform (GFT) and graph wavelet filterbanks in place of the conventional Fourier and wavelet transforms. Initially, the raw images are represented on a chosen graph structure, and then the root-filtering is applied in the GFT and graph wavelet transform (GWT) domains in the present work. The root filtering method highlights the high frequency terms, which pertain to edges .

2.2 EXISTING SYSTEM

2.2.1 HISTOGRAM EQUALIZATION

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areasof lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequentintensity values.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

2.2.2 Calculation of Histogram Equalization

$$v = round\ \frac{cdf(v) - cdf_{min}}{(M*N) - cdf_{min}} * (L - 1)$$ --- (1.a)

cdf(v) – Cumulative Distribution Function of the histogram table. cdf_{min} – Minimum Cumulative Distribution Frequency.

M – Width of the gray scale Image. N – Length of the gray scale Image.

L – Number of grey levels used (Usually 256)

NOTE : The cdf must be normalized to [0, 255].

2.2.3 For Color Images

Histogram equalization can also be used on color images by applying the same method separately to the Red, Green and Blue components of the RGB color values of the image. However, applying the same method on the Red, Green, and Blue components of an RGB image may yield dramatic changes in the image's color balance since the relative distributions ofthe color channels change as a result of applying the algorithm. However, if the image is first converted to another color space, Lab color space, or HSL/HSV color space in particular, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image. There are several histogram equalization methods in 3D space. Trahanias and Venetsanopoulos applied histogram equalization in 3D color space However, it results in "whitening" where the probability of bright pixels are higher than that of dark ones. Han et al. proposed to use a new cdf defined by the iso-luminance plane, which results in uniform gray distribution.

For Example :

Small Image of 8-bit gray scale image shown below

52	55	61	59	79	61	76	61
62	59	55	104	94	85	59	71
63	65	66	113	144	104	63	72
64	70	70	126	154	109	71	69
67	73	68	106	122	88	68	68
68	79	60	70	77	66	58	75
69	85	64	58	55	61	65	83
70	87	69	68	65	73	78	90

Fig -2.1-bit gray scale Image

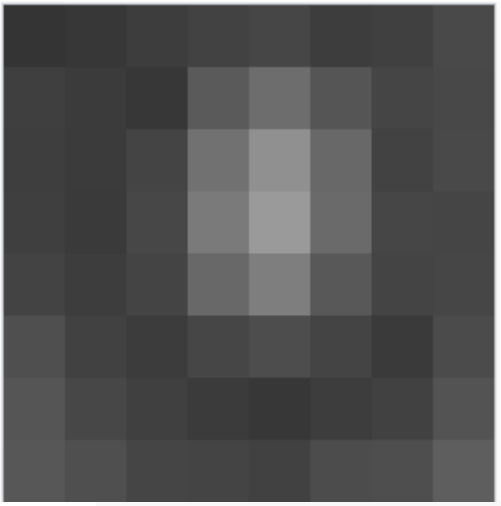


Fig -2.2The 8×8 sub-image shown in 8-bit grayscale

The histogram for this image is shown in the following table. Pixel values that have a zero.

Value	Count	Value	Count	Value	Count	Value	Count	Value	Count
52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
61	4	69	3	78	1	104	2		
62	1	70	4	79	2	106	1		
63	2	71	2	83	1	109	1		

Table 1 Pixel Value Vs count

This cdf shows that the minimum value in the subimage is 52 and the maximum value is 154. The cdf of 64 for value 154 coincides with the number of pixels in the image. The cdf must be normalized to [0,255]. The general histogram equalization formula is:

$$(v) = round\ (\frac{cdf(v) - cdf_{min}}{(M*N) - cdf_{min}} * (L-1)) \quad \text{---(1.a)}$$

V, Pixel Intensity	cdf(v)	h(v), Equalized v
52	1	0
55	4	12
58	6	20
59	9	32
60	10	36
61	14	53
62	15	57
63	17	65
64	19	73
65	22	85
66	24	93
67	25	97
68	30	117
69	33	130
70	37	146
71	39	154
72	40	158
73	42	166
75	43	170

v, Pixel Intensity	cdf(v)	h(v), Equalized v
76	44	174
77	45	178
78	46	182
79	48	190
83	49	194
85	51	202
87	52	206
88	53	210
90	54	215
94	55	219
104	57	227
106	58	231
109	59	235
113	60	239
122	61	243
126	62	247
144	63	251
154	64	255

Once this is done then the values of the equalized image are directly taken from thenormalized cdf to yield the equalized values:

0	12	53	32	190	53	174	53
57	32	12	227	219	202	32	154
65	85	93	239	251	227	65	158
73	146	146	247	255	235	154	130
97	166	117	231	243	210	117	117
117	190	36	146	178	93	20	170
130	202	73	20	12	53	85	194
146	206	130	117	85	166	182	215

Fig - 2.3 Equalized Image

Notice that the minimum value (52) is now 0 and the maximum value (154) is now 255

With Image and Result:

Fig -9 Before Histogram Equalization

Fig -10 After Histogram Equalization

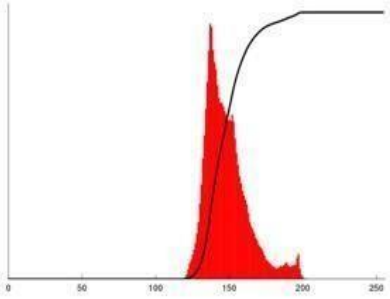


Fig -11 Corresponding histogram and Cumulative histogram

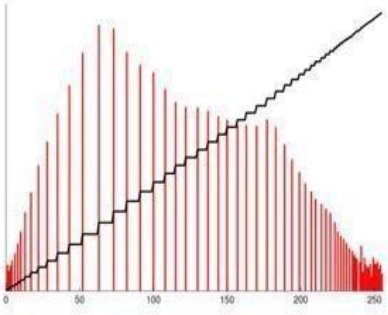


Fig -12 Corresponding histogram and Cumulative histogram

Results of example :

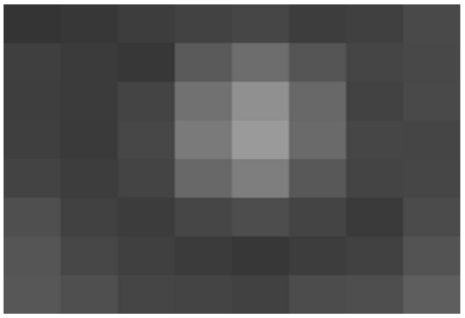


Fig -5 Original

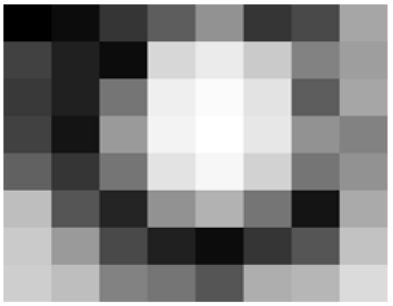


Fig - 6 Equalized

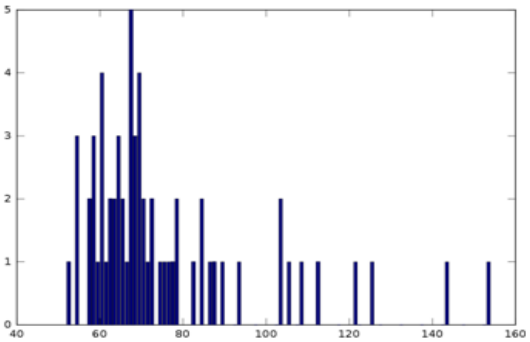


Fig -7 Histogram of Original Image

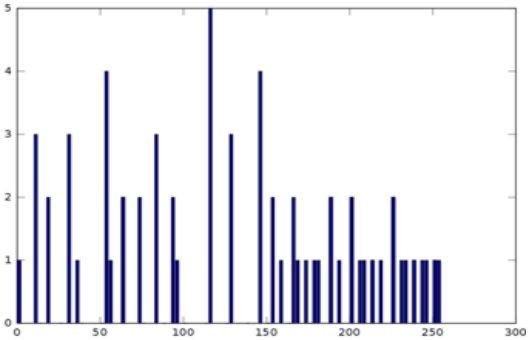


Fig -8 Histogram of Equalized Image

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 SCOPE OF THE PROJECT

The project's goal is to provide everyone in the globe the wonderful ease of viewing bright
and contrast under water images.

3.2 OBJECTIVE

The objectives of this research is to develop an improved single image enhancement technique with is able to work under a variety of conditions and using a standard formulation for assessment of underwater image quality. Specifically, the goals of this research are as follows.
Analyze and study the effects on light propagation under water and its impacto n the degradation on image quality.
Comparing the results of the proposed solution using established techniques withother state of the art solutions to the problem of enhancing underwater images.

Hardware Interfaces

Since the application must run over the internet, this brings out the requirement of a network interface on the device. User should have a device with valid internet connection, Wi-Fi or 3G/4G/5G.

Software Interfaces

The data of the project is organized in a relational database as it makes it easier to curate data with a large number of attributes. Python 3.9.6 will be used as the predominant programming language for this project. Since the project is based on Machine Learning Python is the clear as it has built in modules for handling and cleansing data eg. Pandas and numpy and for training the recommendation model eg. ScikitLearn.

Hardware requirements

RAM	: 4GB or above 4GB
Processor of Frequency	: 1.5GHz or above
Processor	: Intel Pentium 4 or higher

RAM : 4GB or above 4GB

Software requirements

Operating System	: Windows 8 and above
Languages	: Python, Deep Learning
Tools used	: Visual Studio Code, Jupyter Notebook

3.3 SOFTWARE REQUIREMENTS

The experiment or project is worked on python software. Python contain huge packages. Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large- scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured and procedural, object-oriented and functional programming. Python is a popular high-level programming language used for a wide range of applications, including web development, data analysis, artificial intelligence, and scientific computing. It is known for its simple syntax, readability, and ease of use, making it a great language for beginners to learn. It has a vast library of modules and packages that can be used for various tasks, making it a versatile language. Python is compatible with Windows 10, and many developers use it to build applications on this operating system.

Package	Command to install
pip	python –m pip install --upgrade pip
numpy	python –m pip install numpy
openCV	python –m pip install opencv-python (or) python –m pip install opencv-contrib-python
xlwt	python –m pip install xlwt
scikit-image or skimage	python –m pip install scikit-image
natsort	python –m pip install natsort
datetime	python –m pip install datetime
pandas	python –m pip install pandas
matplotlib	python –m pip install matplotlib

- ***3.3.1 Python***

  Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

  Python Features

  Python's features include −

- **Easy-to-learn** − Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is relatively simple to maintain..
- **A broad standard library** − The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode** − Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable** − Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable** − The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python can construct and port GUI programs to numerous system calls, libraries, and Windows systems, including Windows MFC, Macintosh, and Unix's X Window system.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Python Libraries

Python libraries include:

- ***3.3.2 Scikit-learn***

It is also known as Sklearn is a free software machine-learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting and k-means. It was originally called ***scikits. learn*** and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project to another level and made the first public release (v0.1 beta) on 1st Feb. 2010.v It is a community, where everyone can contribute to it. Various organizations such as Booking.co, JP Morgan, Evernote, Inria, Spotify and much more use Sklearn.

The prerequisites of using Sklearn, including the installation of Python, NumPy, Scipy, Joblib, Matplotlib and Pandas. There are two ways to install scikit-learn –

1. Using pip: pip install -U scikit-learn

2.Using Conda: conda install scikit-learn

Features of Scikit-learn

- **Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are part of scikit-learn.
- **Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, and PCA (Principal Component Analysis) to unsupervised neural networks.
- **Clustering** − This model is used for grouping unlabelled data.
- **Cross Validation** − It is used to check the accuracy of supervised models on unseen data.
- **Dimensionality Reduction** − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
- **Ensemble methods** − As name suggests, it is used for combining the predictions of multiple supervised models.
- **Feature extraction** − It is used to extract the features from data to define the attributes in image and text data.
- **Feature selection** − It is used to identify useful attributes to create supervised models.
- **Open Source** − It is an open source library and also commercially usable under a BSD license.

▪ ***3.3.3 NumPy***

NumPy(Numerical Python) is a library for the Python programming language, that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It contains various features including:

- A powerful N-dimensional array object

- Sophisticated functions
- Tools for integrating C/C++ and Fortan code
- Useful linear algebra, Fourier transform and random number capabalitites.

- ***3.3.4 Pandas***

  pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

  Features

- It has a fast and efficient Data Frame object with default and customized indexing.
- Used for reshaping and pivoting data sets.
- Group by data for aggregations and transformations.
- It is used for data alignment and integration of missing data.
- Provide the functionality of Time Series.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, and time series.
- Handle multiple operations of the data sets such as sub setting, slicing, filtering, group By, re-ordering, and re-shaping.
- It integrates with other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and If you want to speed it, up even more, you can use **Python**.

- ***3.3.5 Matplotlib***

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

  Features

- Create publication-quality plots.

- Make interactive figures that can zoom, pan, and update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib**.**

CHAPTER-4
DESCRIPTION OF PROPOSED SYSTEM

4.1 PROPOSED SYSTEM

4.1.1 GLOBAL HISTOGRAM STRETCHING

Histogram Stretching is mainly used for contrast purpose i.e. to increase the contrast. Contrast is the difference between maximum and minimum pixel intensity.

$$g(x,y,z) = \frac{f(x,y,z) - fmin}{} * (255-0)$$

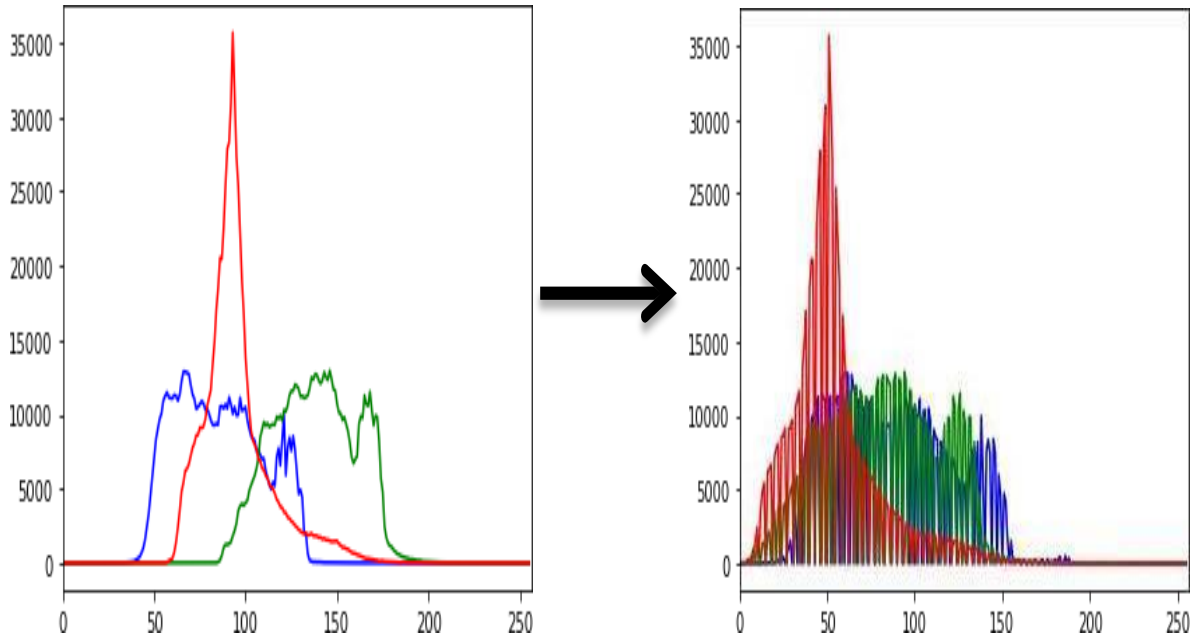
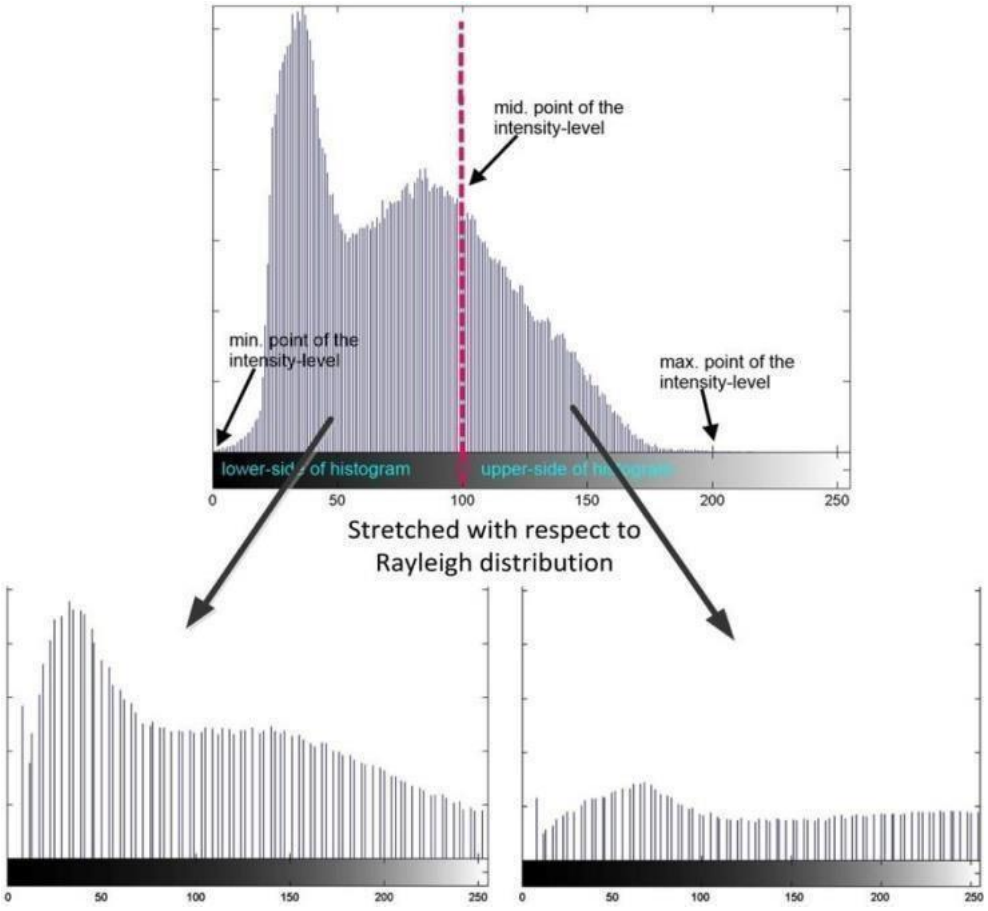


Fig -4.1 Global Histogram Stretching

4.1.2 DIVISION INTO LOWER AND UPPER SIDES

From Stretched Histogram we are dividing into two stretched histogram. We consider three points minimum point, mid point and maximum point. We will construct the lower

stretched histogram by lower side of histogram i.e. in between minimum point and mid point. Similarly, wecan construct upper stretched histogram by upper side of histogram i.e. in between mid point and maximum point.



4.1.3 HSV COLOR MODEL OR COLOR SPACE

HSV (hue, saturation, value) is one of several color systems used by people to select colors from a color wheel or palette. This color system is considerably closer than the RGB system to the way in which humans experience and describe color sensations. In artists‟ terminology,hue, saturation, and value refer approximately to tint, shade, and tone.

The HSV color space is formulated by looking at the RGB color cube along its gray axis (the axis joining the black and white vertices), which results in the hexagonally shaped color palette. As we move along the vertical (gray) axis, the size of the hexagonal plane that is perpendicular to the axis changes, yielding the volume depicted in the figure. Hue is expressed as an angle around a color hexagon, typically using the red axis as the reference (0°) axis. The value component is measured along the axis of the cone.

The V = 0 end of the axis is black. The V = 1 end of the axis is white, which lies in the center of the full color hexagon. Thus, this axis represents all shades of gray. Saturation (purity of the color) is measured as the distance from the V axis.

The HSV color system is based on cylindrical coordinates. Converting from RGB to HSV entails developing the equations to map RGB values to cylindrical coordinates. This topic is treated in detail in most texts on computer graphics so we do not develop the equations here.

The MATLAB function for converting from RGB to HSV is rgb2hsv, whose syntax is

hsv_image = rgb2hsv(rgb_image)

The input RGB image can be of class uint8, uint16, or double; (The input image must be of class double. The output is of class double also) the output image is of class double.

The function for converting from HSV back to RGB is hsv2rgb:

rgb_image = hsv2rgb(hsv_image)

- H = Hue / Color in degrees $\in$[0,360]
- S = Saturation $\in$[0,1]
- V = Value $\in$[0,1]

4.1.4 Conversion of RGB to HSV

V = max = max(R,G,B)

min = min(R,G,B)

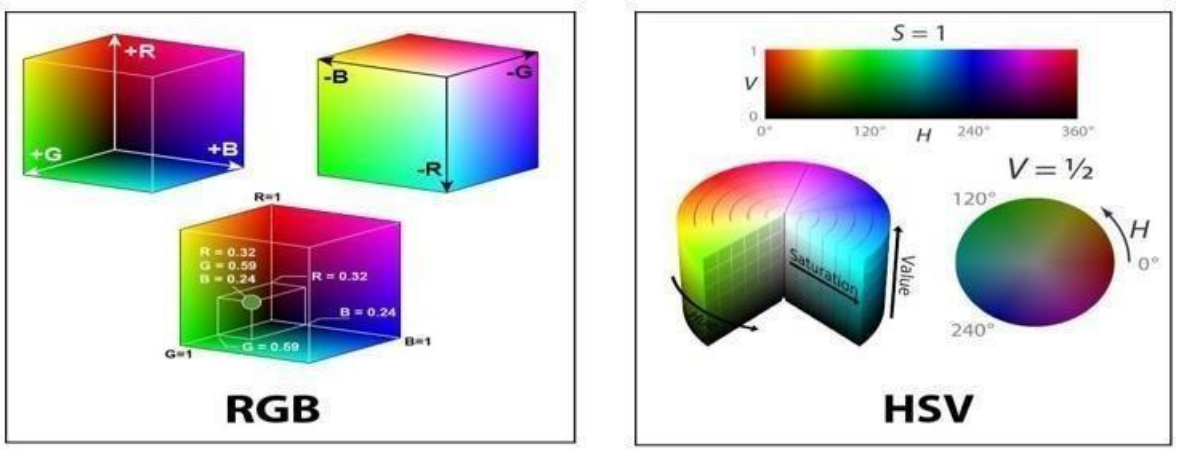
$$S = \frac{(\max - \min)}{\max} \quad (\text{or } S=0, \text{ if } V=0)$$

$$H = 60 * (0 + \frac{G-B}{max-min}), \text{ if max} = R$$

$$60 * (2 + \frac{B-R}{max-min}), \text{ if max} = G$$

$$60 * (4 + \frac{R-G}{max-min}), \text{ if max} = B$$

$$H = H+360, \text{ if } H<0$$



RGB And HSV Color space

Fig 4.2 RGB and CSV color space

4.1.5 STRETCHING OF S AND V VALUES

we use the histogram stretching technique of min-max based sub-image-clipped introduced. We obtain the histograms of components S and V of the image processed in the first stage. Then, we and the maximum and minimum points in each histogram. The minimum intensity point i_{min} refers to the lowest intensity value in the image histogram, whereas the maximum intensity value i_{max} refers to the highest intensity value. The mid-point of the intensity level i_{mid} of a histogram is calculated using below equation.

Since brightness of image is shown in component V, we process the V component to overcome under enhanced or over enhanced areas possibly brought forth in the first step. The S component is also processed for improving the color information of the image. Hence, eachhistogram is divided into two regions, namely lower and upper. Each region is then stretched as follows :

- The lower region, from i_{min} to the mid-point i_{mid}, is stretched towards the higher- intensity level with the minimum output value of 1% from the minimum intensity value of the dynamic range.

- The upper region, from i_{mid} to the maximum intensity value i_{max}, is stretchedtowards the lower intensity level with the maximum output value of 1% from the maximum intensity value of the dynamic range.

  The above limits are set in order to prevent the S and V components from exceeding their minimum and maximum values, which can lead to under or over brightness of the image. The stretching (1%) is done to decrease the effect of dominant color channels and it is empirically chosen. The resultant images with a bigger limiting value are very over-enhanced

  The same integration process is applied to component V, where the lower and upper stretched histograms are integrated. Then, H, S, and V components are composed to produce an image in HSV color model. Finally, the enhanced image is converted to RGB by inverse transform. Figure 5 shows the resultant images after performing each step on the image. The effect of stretching 1% on the histograms is clear in this figure.

  Image contrast is enhanced based on the chosen target image and this process may interfere with the color purity in some regions. Therefore, in the second stage, we simultaneously change the histograms of both components S and V of the image produced in the first stage inorder to solve this potential problem. Indeed, the process in the second stage resolves the problem of under- and over-enhanced regions.

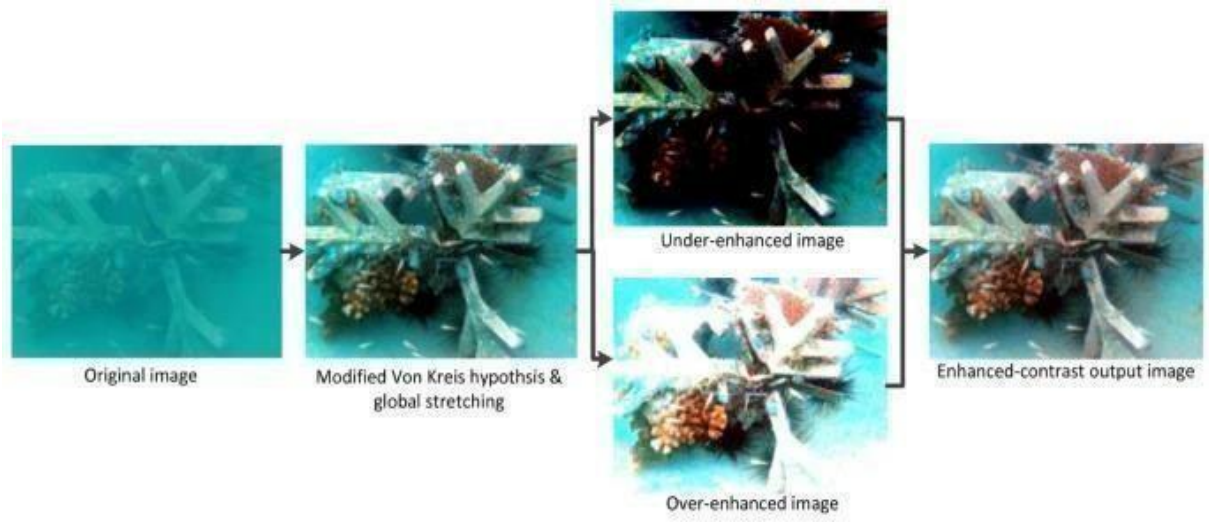


Fig 4.3- Illustration of Dual Intensity and Rayleigh stretching

4.1.6 CONVERSION OF HSV TO RGB

max =V

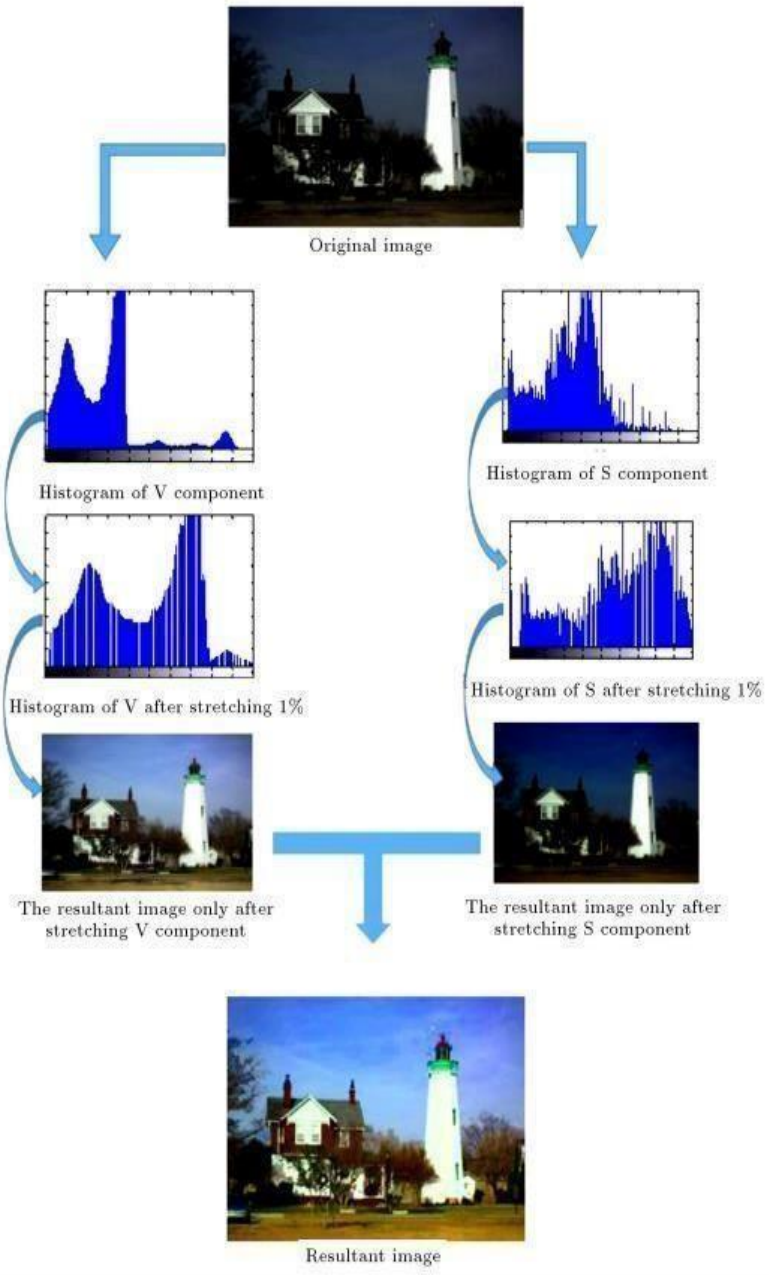
C = S*V

min = max-C

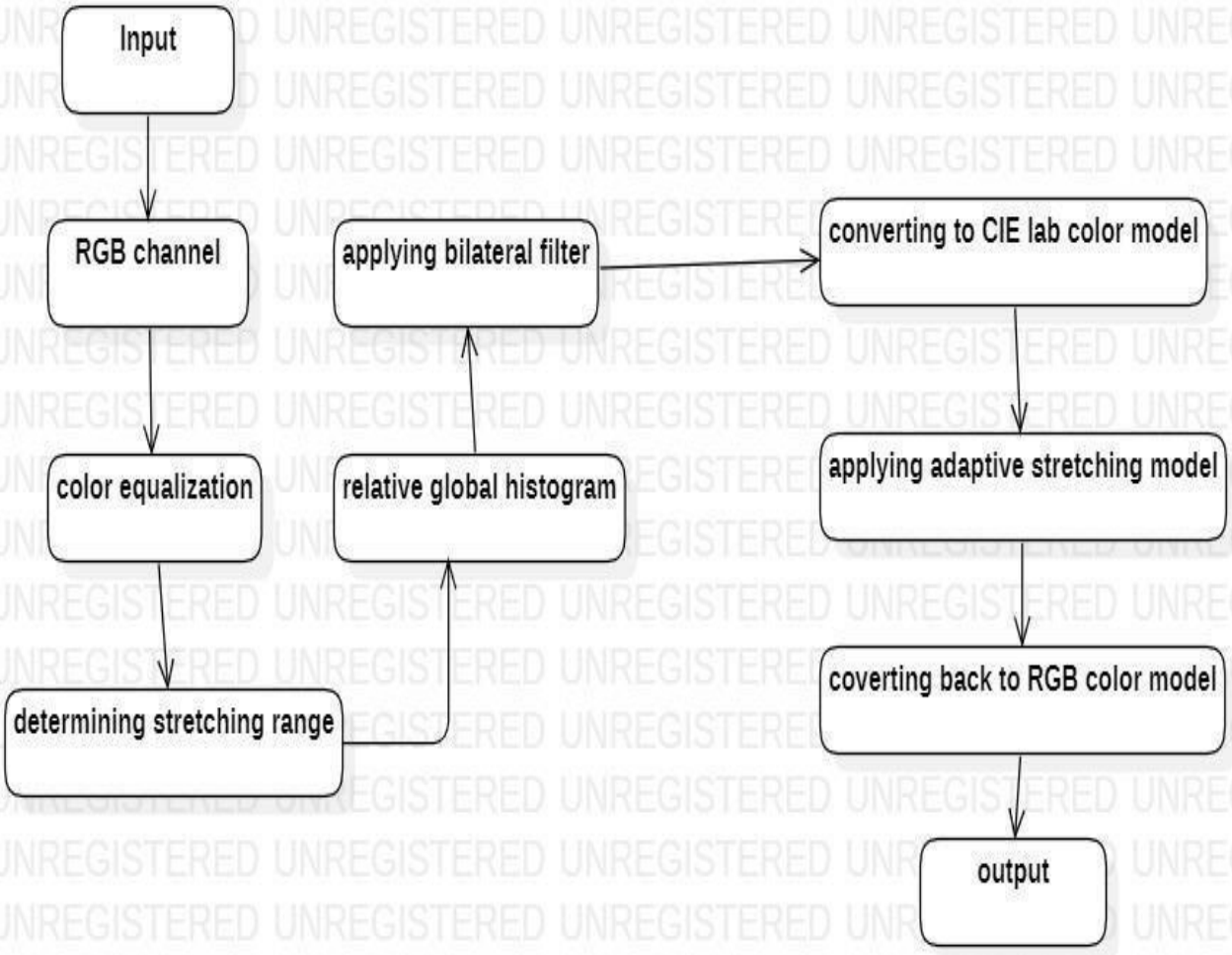
- If H > = 300 then H' = $\frac{H-360}{60}$

  else if H < 300 then H' = $\frac{H}{60}$

1. If H" is in range [-1,1)
   - ✓ If H" < 0
     - R = max
     - G = min
     - B = G-H"*C
   - ✓ If H" >= 0
     - R = max
     - G = B-H"*C
     - B = min
2. If H" is in range [1,3)
   - ✓ If H" < 2
     - R = B - (H" - 2) * C
     - G =max
     - B =min
3. If H" is in range [3,5)
   - ✓ If H" < 4
     - R = min
     - G = R - (H" - 4) * C
     - B =max
   - ✓ If H" >= 4
     - R = G + (H" - 4) * C
     - G = min
     - B = max



4.2 SYSTEM ARCHITECTURE



4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System

4.3.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

4.3.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

4.3.3 Getting Python

The most up-to-date and current source code, binaries, documentation, news,

etc., is available on the official website of Python https://www.python.org.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer python-XYZ. msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

4.3.4 First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python
Python2.4.3(#1, Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type"help","copyright","credits"or"license"for more information.

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −
print"Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

4.3.5 What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

4.3.6 What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.
-

4.3.7 Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

  Good to know
- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Net beans or Eclipse which are particularly useful when managing larger collections of Python files.

  Python Syntax compared to other programming languages
- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

4.4 Testing Plan

4.4.1 Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

4.4.2 Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

4.4.3 Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

4.5 Project management plan

A project management plan for an under water image processing and enhancement would typically include the following components:

1. Scope: This section would define the goals and objectives of the project, as well as the deliverables and milestones that need to be achieved. It would also outline the requirements and specifications for the image processing technology, as well as the integration with the under water image processing.

2. Schedule: This section would include a detailed timeline for the project, with tasks and activities broken down into smaller, manageable components. It would also include deadlines and milestones, as well as resource allocation and dependencies.

3. Budget: This section would outline the financial resources required for the project, including the costs of hardware and software, development and testing, and any other expenses related to the project. It would also include contingency plans and risk mitigation strategies.

4. Quality management: This section would outline the quality standards and procedures for the project, including testing and validation protocols, as well as acceptance criteria and user feedback mechanisms.

5. Human resources: This section would outline the roles and responsibilities of the project team, as well as their skills and experience requirements. It would also include recruitment and training plans, as well as performance evaluation and feedback mechanisms.

6. Communication: This section would outline the communication channels and protocols for the project, including regular status updates, progress reports, and stakeholder engagement. It would also include conflict resolution and escalation procedures.

7. Risk management: This section would outline the risks and uncertainties associated with the project, as well as the risk management strategies and contingency plans to mitigate them. It would also include a risk register and risk assessment methodology.

Overall, a project management plan for an under water image processing and enhancement would be a comprehensive and detailed document that outlines the goals, objectives, and requirements of the project, as well as the resources, schedules, and risks associated with it. It would also include mechanisms for quality assurance, communication, and stakeholder engagement, to ensure that the project is completed successfully and within budget and timeline constraints.

CHAPTER 5
IMPLEMENTATION DETAILS

5.1 DEVELOPMENT SETUP

The following steps should be followed to set up the development and deployment environment for this project:

5.1.1 Install Python:

Download and install python from the official Python website. Make sure to select the latest version of the Python. Click the appropriate link for your system to download the executable file.

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

5.1.2 Python QuickStart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

5.1.3 The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile.py

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment.

5.1.4 INSTALL REQUIRED LIBRARIES:

Open the command prompt or terminal and navigate to the project directory. Install the required libraries by running the following command: “pip install -r requirements.txt”. This command will install all the required libraries mentioned in the requirements.txt file.

5.2 ALGORITHMS

5.2.1. HISTOGRAM EQUALISATION

In underwater situations, images are rarely color balanced correctly. After R-G-B channel decomposition, we first conduct color equalization for the underwater image. In Rayleigh Stretching adjusted the color values of the RGB components based on the modified Von Kries hypothesis, which keeps the dominant color cast channel constant.

We correct the G and B channels following the GW assumption theory. Here, R channel is not considered because the red light in the water is hard to compensate by simple color equalization, which may bring about red over-saturation.

We are not equalizing the R color i.e. Red because we observe in Rayleigh Stretching that the Red color is appeared in the image because of the saturation state and it has the shortest wavelength i.e. if we equalize then it will be in the saturation region itself again it will appear on the image.

5.2.2. RAYLEIGH DISTRIBUTION

After the contrast correction in the RGB color model, the image will undergo the color correction process. In this process, the underwater image is converted into a CIE-Lab color model to improve color performance. In the CIE-Lab color model, „L‟ component, which is equivalent to the image lightness, represents the brightest value at L = 100 and the darkest value at L= 0. When a = 0 and b = 0, the color channel will present true neutral gray values. Hence, the output color gradations of the „a‟ and „b‟ components are modified to acquire color correction accurately, meanwhile, „L‟ component is used to adjust the brightness of the entire image.

Firstly, the shallow-water image in the CIE-Lab color model is decomposed into

respective channels. Color and brightness in an image are the important parameters of clearness and visibility. Therefore, the objects in the image can be clearly differentiated from the background. After the adaptive stretching process of „L", „a" and „b" components in the CIE- Lab color model, the channels are then composed and converted back into the RGB color model. A contrast-enhanced and color-corrected output image can be generated as the perceivable and visible final output image

5.2.3. RELATIVE GLOBAL HISTOGRAM STRETCHING

The blind global histogram stretching usually uses the same parameters for all R-G-B channels of the images, ignoring the histogram distribution characteristics of different channels and in different images. When the fixed values (e.g., 0, 255) are applied in the relative histogram stretching P0 equation, it may over-stretch or under-stretch certain color channels and damage the details of the original image. According to the propagation law of underwater light, we need to apply the contrast correction method to modify the distorted images. The histogram distribution rule of RGB channels on the following observation in shallow water images: In most of the hallow-water images, the histogram of red light is focused in such values [50, 150], while G channel and B channel have the most of numerical concentrated in the range [70, 210]. This indicates that histogram stretching should be sensitive to channels

5.2.4. STRETCHING RANGE

Adaptive parameter acquisition for stretching range: For the underwater images, global histogram stretching, range [0, 255], often brings excessive blue-green illumination. In order to achieve an optimal desired range of stretching, we dynamically determine the minimum (Omin) and maximum (Omax) intensity level values for each R-G-B channels. We first calculate the standard deviation values of the Rayleigh distribution $\sigma\lambda$. For the maximum parameters of the desired range, because of the different degrees of attenuation of the different light bands in the water, we must take a separate analysis of RGB channels to calculate.

5.2.5. COLOR EQUALIZATION ON G-B CHANNELS

In the underwater situation, images are rarely color balanced correctly. After R-G-B

channel decomposition, we firstly conduct color equalization for underwater image.We correct the G and B channels following the GW assumption theory. Here, R channel is not considered because the red light in the water is hard to compensate by simple color equalization, which may bring about red over-saturation. Equations of Gavg, Bavg , θ_b, θg is used to calculate the G- B channel of color equalization coefficient θb, θg, where M and N are the spatial resolution of an image. In Rayleigh Stretching adjusted the color values of the RGB components based on modified Von Kries hypothesis, which keeps the dominant color cast channel constant. However, Inspired by the Gray-World (GW) assumption theory that the average value of color object in a perfect image is gray, we correct the G- B channel following the assumptions shown below.

$$\frac{((Ravg+Gavg+Bavg))}{3} = 0.5$$

Where Ravg, Gavg, Bavg are the normalized average values of the recovered red channel, green channel, blue channel, respectively. We correct the G and B channels following the GW assumption theory. Here, R channel is not considered because the red light in the water is hard to compensate by simple color equalization, which may bring about red over- saturation. Equations of Gavg, Bavg , θ_b, θg is used to calculate the G- B channel of color equalization coefficient θb, θg, where M and N are the spatial resolution of an image. Based on the coefficient θb, θg, the intensity values of G-B channel can be corrected by multiplying

θb, θg, respectively. Next, we carry out dynamic histogram stretching for the image channels with the RGHS strategy.

It should be noted that when the distribution of a channel shows a normal distribution, its mode and its mid-point are almost the same. We take the mode value as a dividing line to separately decide the minimum (left) and maximum (right) intensity level values of the input image in the histogram stretching. Because the underwater images are influenced by various factors, to reduce the impact of some extreme pixels on the relative global histogram stretching, it usually takes the stretching range between 0.1% and 99.9% of the histogram. However, if the histogram is not normally distributed, this method that removes an equal number of pixels from two tails of the histogram may not be reasonable. We split the upper and lower proportion of the

intensity values to calculate the $Imax\ and\ Imin$ for each R-G-B channel, which is showed in the equation below.

$$Imin = S.sort[S.sort.index(a) * 0.1\%]$$

$$Imax = S.sort[-(S.length - S.sort.index(a)) * 0.1\%]$$

where S is the set of image pixel values for each R-G-B channel, $S.\ sort$ is the sorted data set in an ascending order, $S.\ sort.\ (a)$ is the index number of the mode in the histogram distribution, and $S.\ [x]$ represents the value in the index x of positive sorted data set. Equation (8) means that from the peak division line, we separate the pixels which values arein the smallest 0.1% of the total number on the left side and the biggest 0.1% of the total number on the right side from the histogram distribution to perform the special method. For different images and RGB channels of the Rayleigh distributions, the $Imax\ and\ Imin$ are both image and channel sensitive.

5.2.6. Adaptive parameter acquisition for the stretching range : For the underwater images, global histogram stretching, range [0, 255], often brings excessive blue-green illumination. In order to achieve an optimal desired range of stretching, we dynamically determine the minimum (Omin) and maximum (Omax) intensity level values for each R-G-B channels.

We first calculate the standard deviation values of the Rayleigh distribution σ_λ, shown in (10)

$$\sigma = \sqrt{\frac{4-}{2}}a = 0.655 * a \quad , \quad \lambda \in \{R,G,B\}$$

Where $\lambda \in \{R,G,B\}$ } indicates the R, G, B channel, a is the mode in a channel. Then, we define the minimum value of the desired range O_{min} and O_{max} as below.

$$O_{min} = a_\lambda - \beta_\lambda * \sigma_\lambda \ , \quad 0 \le O_{\lambda min} \le I_{hmin}$$

$$O_{max} = \frac{a_\lambda + \mu_\lambda * \sigma_\lambda}{k * t_\lambda} \quad I_{hmax} \le O_{min} \le 255 \qquad ---(11)$$

Here, β_λ can be derived from (11) and substitute σ with (10).

$$\beta_\lambda = \frac{a - O_{min}}{\sigma} \cdot \frac{a - I_{min}}{\sigma} \le \beta_\lambda \le \frac{a}{\sigma} \qquad ---(12)$$

On the right of equation (12), as $a \ge I_{min}$, we can get $0 \le \beta_\lambda$. Substitute (10) into the right of equation (12), we can get $\beta_\lambda \le 1.526$. Define $\beta_\lambda \in Z$, then β_λ has a unique solution: $\beta_\lambda = 1$. Therefore, (11) can be simplified as (13).

$$O_{min} = a_\lambda - \sigma_\lambda \qquad ---(13)$$

For the maximum parameters of the desired range, because of different degrees of attenuation of the different light bands in the water, we must take separate analysis of RGB channels to calculate. According to the simplified fuzzy imaging model (1), the haze-free image $J_\lambda(x)$ can be recovered by (14).

$$J(x) = \frac{I_\lambda(x) - (1 - t_\lambda(x))B_\lambda}{kt(x)} \qquad ---(14)$$

5.2.7. ADAPTIVE-STRETCHING IN CIE-LAB COLOR MODEL

After the contrast correction in RGB color model, the image will undergo the color correction process. In this process, the underwater image is converted into CIE-Lab color model to improve color performance. In the CIE-Lab color model, „L‟ component, which is equivalent to the image lightness, represents the brightest value at L = 100 and the darkest value at L= 0. When a = 0 and b =0, the color channel will present true neutral gray values. Hence, the output color gradations of „a‟ and „b‟ components are modified to acquire color correction accurately, meanwhile, „L‟ component is used to adjust the brightness of the entire image. Firstly, the shallow-water image in the CIE-Lab color model is decomposed into respective channels.

The „L‟ component is applied with linear slide stretching, given by (8), which range between 0.1% and 99.9% is stretched to range [0, 100]. The 0.1% of the lower and upper values in the image are setto 0 and 100 respectively.

The „a‟ and „b‟ components are in the range of [−128, 127], of which 0 is the median value. The stretching of „a‟ and „b‟ is defined as an S-model curve.

$$p_{\chi} = I_{\chi} * (\varphi \qquad ^{128}) \, , \; \mathrm{X} \in \{a, b\}$$

Where I_{χ} and p_{χ} represents the input and output pixels, respectively, $\chi \in \{a, b\}$ indicates the „a‟ and b‟ components, φ is optimally-experimental value, set to 1.3 in the method. Equation (18) uses an exponential function as the stretching coefficient, whereby the closer the values to 0,the further theywill be stretched.

Color and brightness in an image are the important parameters of clearness and visibility. Therefore, the objects in the image can be clearly differentiated from the back- ground. After the adaptive stretching process of „L‟, „a‟ and „b‟ components in the CIE- Lab color model, the channels are then composed and converted back into the RGB color model. A contrast- enhanced and color-corrected output image can be generated as the perceivable and visible final output image.

5.2.8. CIELAB COLOR SPACE

The CIELAB color space also referred to as L*a*b* is a color space defined by the International Commission on Illumination. Referring to CIELAB as "Lab" without asterisks should be avoided to prevent confusion with Hunter Lab. It expresses color as three values: L* for perceptual lightness, and a* and b* for the four unique colors of human vision: red, green, blue, and yellow. CIELAB was intended as a perceptually uniform space, where a given numerical change corresponds to similar perceived change in color. While the LAB space is not truly perceptually uniform, it nevertheless is useful in industry for detecting smalldifferences in color.

The CIELAB space is three-dimensional, and covers the entire range of human color perception, or gamut. It is based on the opponent color model of human vision, where red/green forms an opponent pair, and blue/yellow forms an opponent pair. The lightness value, L*, also referred to as "Lstar," defines black at 0 and white at 100. The a* axis is relative to the green-red opponent colors, with negative values toward green and positive values toward red. The b* axis represents the blue- yellow opponents, with negative numbers toward blue and positive toward yellow.

The a* and b* axes are unbounded, and depending on the reference white they can easily exceed ±150 to cover the human gamut. Nevertheless, software implementations often clamp these values for practical reasons. For instance, if integer math is being used it is common to clamp a* and b* in the range of −128 to 127.

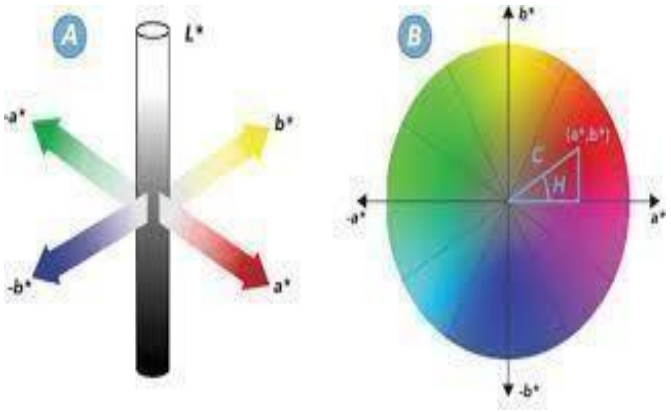


Fig 5.1 RGB color model

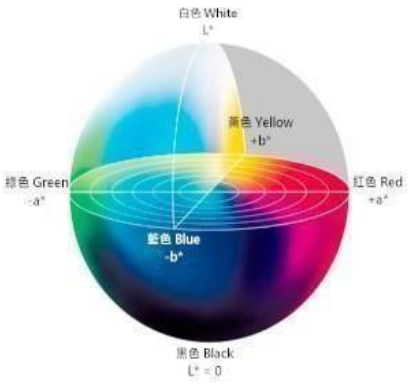


Fig 5.2 CIElab color model

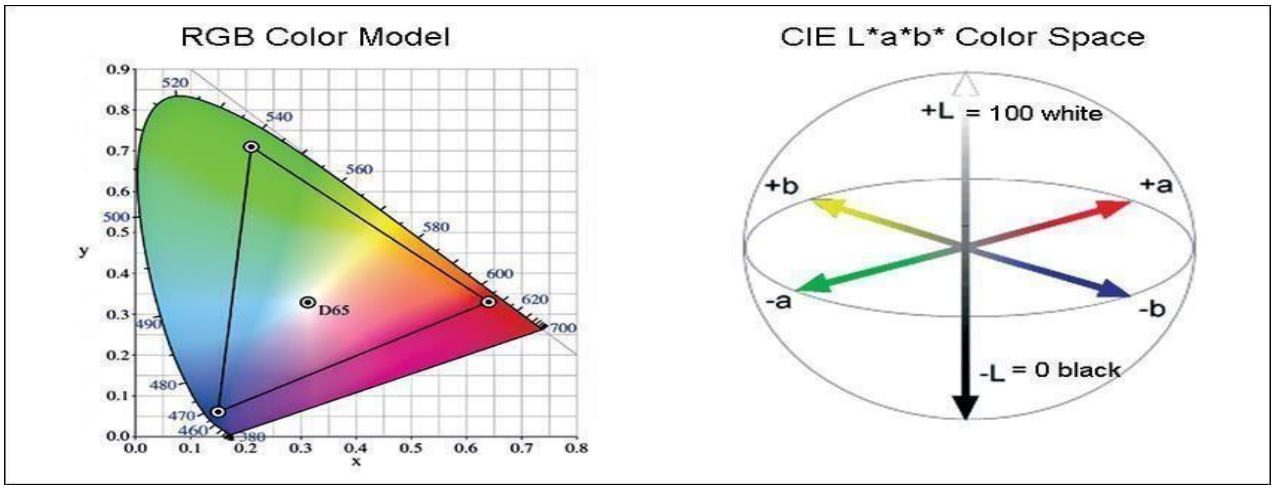


fig 5.3 (A) & (B) CIE LAB Model And RGB Model

5.2.9. CONVERSION BETWEEN RGB AND CIELAB

5.2.9.1. CONVERSION FROM RGB TO LAB

i) From RGB to CIEXYZ

Mathematically we will not convert the RGB into LAB. Initially, we will convert RGB into standardCIEXYZ model. Then, from CIEXYZ we can convert into CIELAB color space.

The numbers in the conversion matrix below are exact, with the number of digits specified inCIEstandards.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{k_{21}} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{11} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49000 & 0.31000 & 0.20000 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00000 & 0.01000 & 0.99000 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

ii) From CIEXYZ to CIELAB

$$L^* = 116\, f\left(\frac{F}{F_n}\right) - 16$$

$$a^* = 500\ \left(f\left(\frac{K}{K_n}\right) - f\left(\frac{F}{F_n}\right)\right)$$

$$b^* = 200\ \left(f\left(\frac{F}{F_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)$$

Where, being $t = \frac{K}{K_n}, \frac{F}{F_n}$ or $\frac{Z}{Z_n}$:

$$f(t) = \begin{cases} \sqrt[3]{t} & if\ t > \blacksquare \\ \frac{t}{3\blacksquare} + \frac{4}{29} & ot\square erwise \end{cases}$$

X, Y, Z describe the color stimulus considered and X_n, Y_n, Z_n describe a specified white achromatic reference illuminant. for the CIE 1931 (2°) standard colorimetric observer and assuming normalization where reference white = *Y* = 100, the values are:

For Standard Illuminant D65:

X_n = 95.0489,
Y_n,= 100,
Z_n = 108.8840

For illuminant D50, which is used in the printing industry:

X_n = 96.4212,
Y_n,= 100,
Z_n = 82.5188

The division of the domain of the *f* function into two parts was done to prevent an infinite slope at $t = 0$. The function *f* was assumed to be linear below some $t = t_0$, and was assumed to match the $\sqrt[3]{t}$ part of the function at t_0 in both value and slope. In other word

$$\sqrt[3]{t_0} = mt_0 + c \quad \text{(match in value)}$$

$$\frac{1}{3}(t_0)^{-\frac{2}{3}} = m \quad \text{(match in slope)}$$

The above two equations can be solved for m and t_0

$$m = \frac{1}{3}\delta^{-2} = 7.787037\ldots$$

$$t_0 = \delta^3 = 0.008856\ldots$$

5.2.9.2. CONVERSION FROM CIELAB TO RGB

i) From CIELAB to CIEXYZ

The reverse transformation is most easily expressed using the inverse of the function f above
:

$$\mathrm{X} = \mathrm{X}_n f^{-1}\left(\frac{L^* + 16}{116} + \frac{a^*}{500}\right)$$

$$Y = Y_n f^{-1}\left(\frac{L^* + 16}{116}\right)$$

$$Z = Z_n f^{-1}\left(\frac{L^* + 16}{116} - \frac{b^*}{200}\right)$$

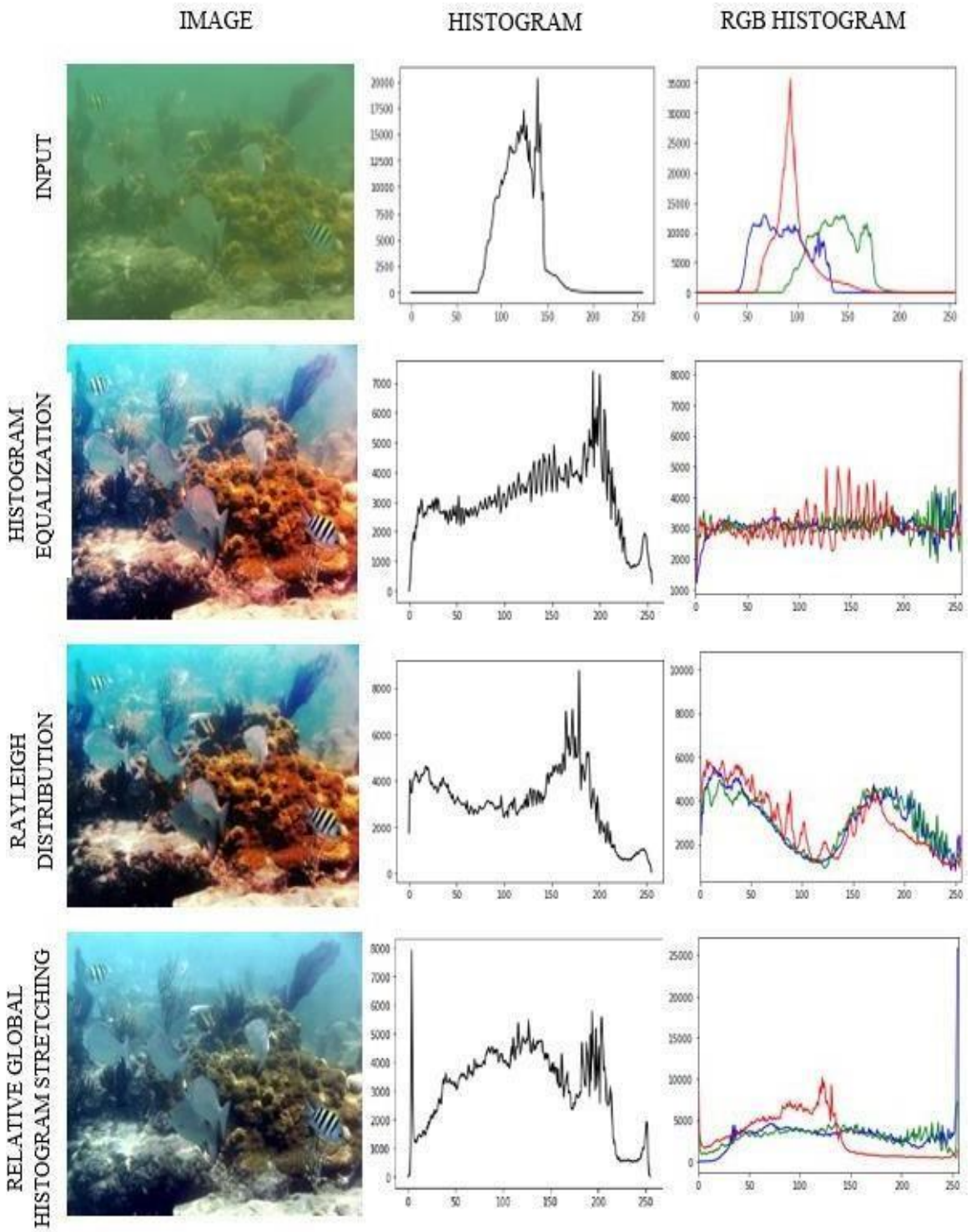
$$\text{Where } f^{-1}(t) = \begin{cases} t^3 \\ 3\delta^2\left(t - \frac{4}{29}\right) \end{cases}$$

ii) From CIEXYZ TO RGB

While the above matrix is exactly specified in standards, going the other direction uses aninversethat is not exactly specified, but is approximately:

$$\begin{matrix} R \\ [G] \\ B \end{matrix} = \begin{bmatrix} 0.41847 & -0.15866 & -0.082835 \\ -0.091169 & 0.25243 & 0.015708 \\ 0.00092090 & 0.0025498 & 0.178000 \end{bmatrix} \cdot \begin{matrix} \mathrm{X} \\ [Y] \\ Z \end{matrix}$$

CHAPTER 6
RESULTS AND DISCUSSION



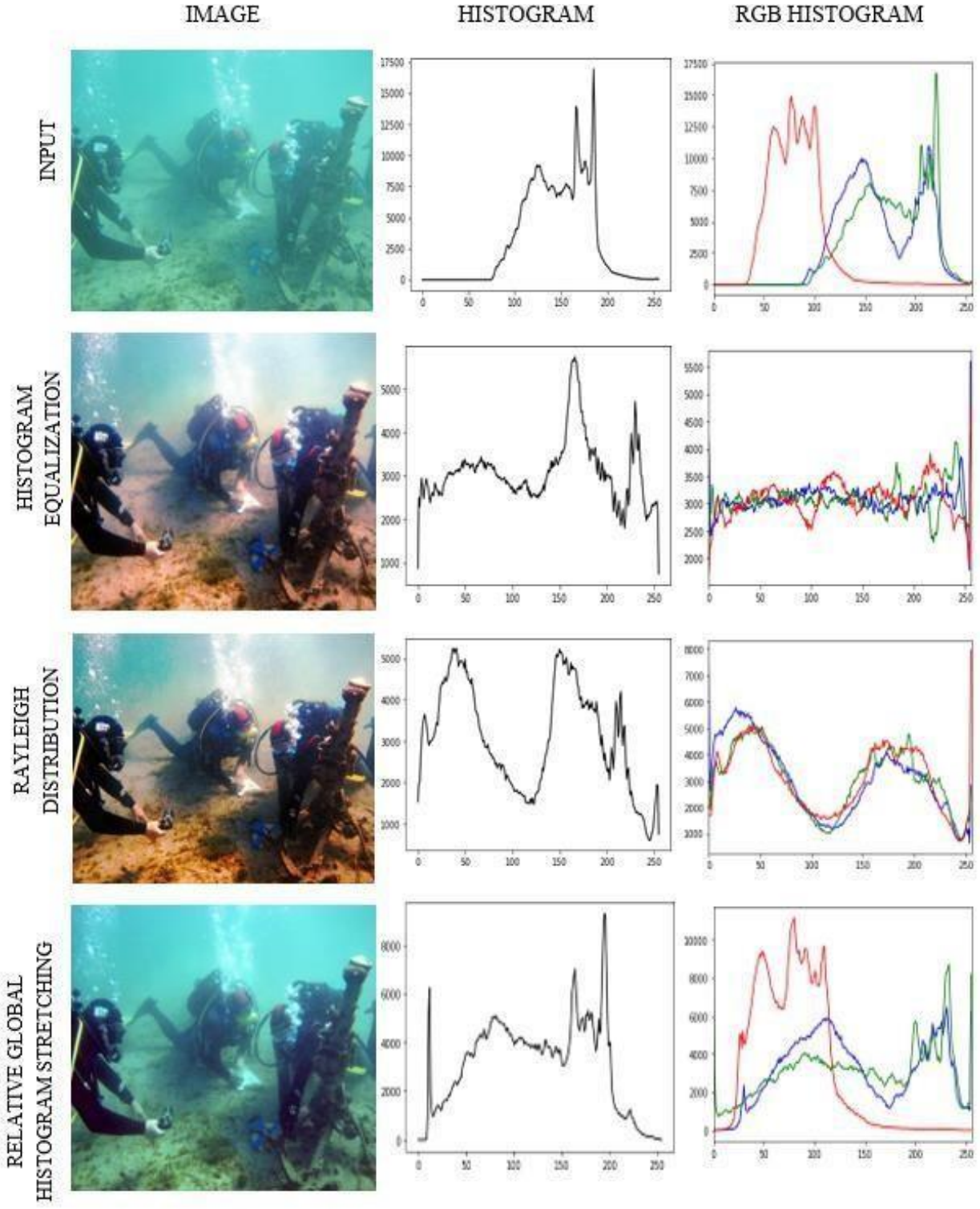


IMAGE
HISTOGRAM
RGB HISTOGRAM
INPUT
HISTOGRAM EQUALIZATION
RAYLEIGH DISTRIBUTION
RELATIVE GLOBAL HISTOGRAM STRETCHING

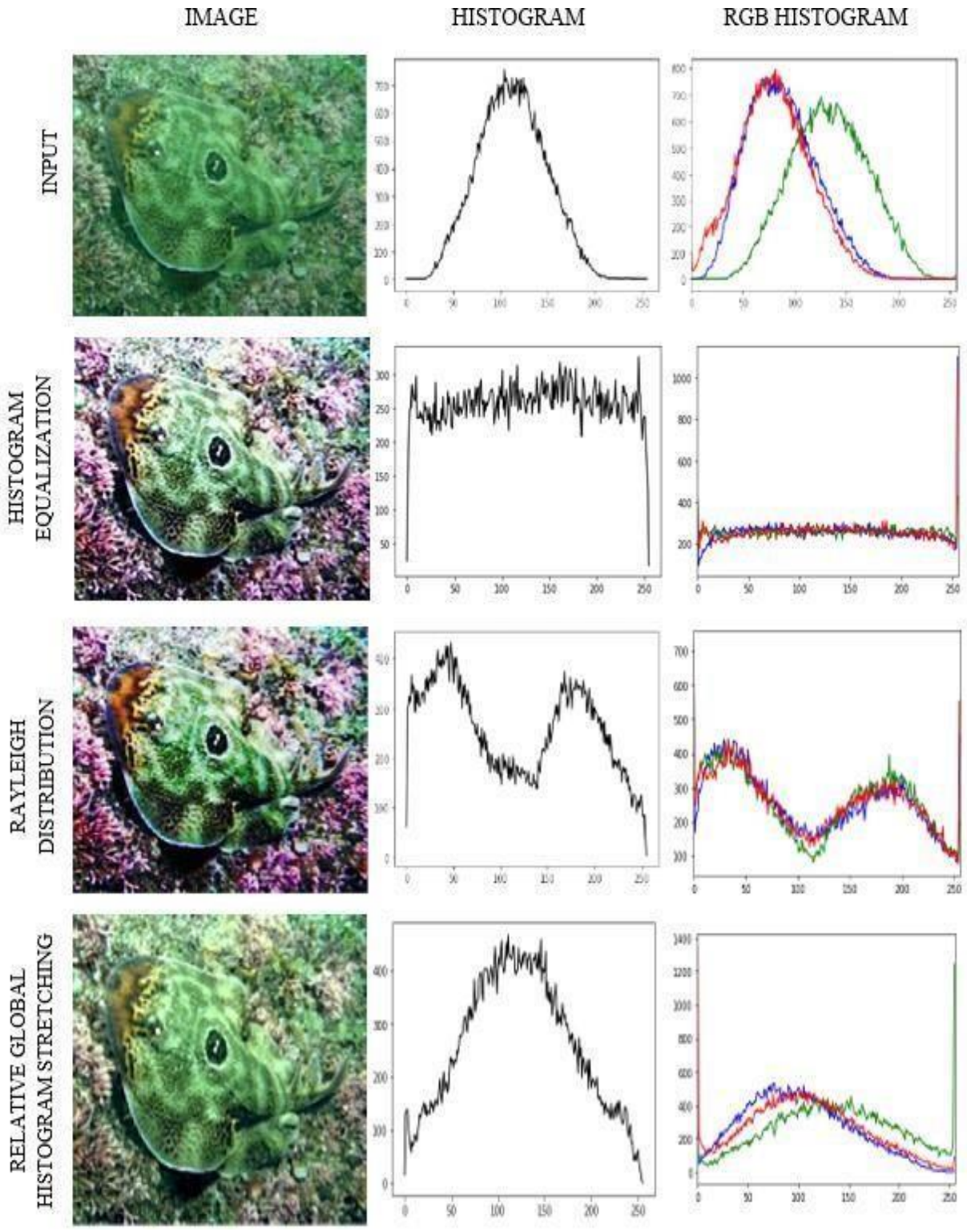


IMAGE
HISTOGRAM
RGB HISTOGRAM
INPUT
HISTOGRAM EQUALIZATION
RAYLEIGH DISTRIBUTION
RELATIVE GLOBAL HISTOGRAM STRETCHING

- Our proposed method first performs $_4c_8$ontrast correction based on simple

histogram stretching with dynamic parameters acquisition in the RGB color model, which takes into account both the histogram distribution feature of the raw image and underwater transmission properties of different light channels. Then, adaptive stretching in the CIE-Lab color model is conducted for color correction. Our proposed method was compared with the typical haze removal model, which is based on histogram stretching in the RGB and HSV/HIS color models. We compare the input with three methods Rayleigh stretching, Histogram Equalization, and global Histogram stretching, and produce the better output image.

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

In Rayleigh stretching, It is a better algorithm than Histogram Equalization. We are just applying the modified Von Kreis hypothesis, global Histogram stretching, divide and composing the lower and upper side of histogram by means of average or mid point. In color correction, we use HSV model for correction purpose.

we have explored the issues related to shallow-water images and existing underwater image enhancement methods, and successfully proposed a new image enhancement method RGHS for different shallow-water images. Our proposed method firstly performs contrast correction based on simple histogram stretching with dynamic parameters acquisition in RGB color model, which takes into account both the histogram distribution feature of the raw image and underwater transmission properties of different light channels. Then, adaptive-stretching in CIE-Lab color model is conducted for color correction. Our proposed method was compared with the typical haze removal model, whichare based on histogram stretching in the RGB and HSV/HSI color models. Qualitative and quantitative results proof that our method is more effective to enhance the visibility, improve details and not to increase artifacts and noise of shallow-water image. The integration of histogram redistribution in the RGB and CIE-Lab color model can be used in other underwater image research. However, our proposed method is limited in the shallow water image enhancement, which ignores theenergy attenuation along propagation path between the surface and the scene and failing to compensate the serious-distorted channel in deeper-water image. In fact, many underwater image enhancement methods are not tested with the images from deep ocean below 1000m. With the reasonable combination of the properties and the degraded physical model of image, we will improve the algorithm to be suitable for different kinds of underwater images.

Finally, We conclude that Histogram equalization is used for contrast correction

purpose itself. But main drawback of this method is pixel losses i.e. color pixel count reduces. In Rayleigh stretching, is used for both color correction and contrast correction. But drawback of it is red color is in saturation. So, we may appear red color shades on image. So, In order to overcome, RGHS is the method, used for both color correction and contrast correction by eliminating R component. So, Red component will not undergo saturation. After Analysis, we conclude that Relative Global Histogram Stretching is more better than Histogram Equalization and Rayleigh Distribution.

7.2 FUTURE WORK

The are so many techniques which will improve the proposed methods. With the development of underwater robotics, Under water exploration has become much easier and less onerous. But, theserobots are expensive mainly because of the high end cameras that are mounted on them. With moreresearch in this field, we can come up with better and robust algorithms which are camera independent.

These algorithms, when incorporated into these robots will not only aid in procuring better enhancedunderwater exploration images, but also in bring down the cost of these robots significantly.

7.3 APPLICATIONS OF UNDERWATER PROCESSING

In recent decay, the applications of underwater image processing have increased in various domains, ever since the development in the intelligence underwater image processing has been increased. Someof the domains are the protection of the underwater environment, navigation of underwater, exploration of underwater resources etc.

1. **Underwater Navigation -** Throughout the navigation, the action of underwater vehiclesdepends on the clarity of the underwater image/video to find the formerly

constructed map.

2. **Underwater Environment Monitoring -**This field is based on the collection system of ocean geographic data. The tools obtain information from the surroundings of underwater then throwthis information to the on-shore station through underwater cable or satellite.

3. **Tracking of Underwater Objects -** In many applications, the detection of underwater objects is difficult to perform. In the underwater environment, the visibility is very poor because ofthe optic camera, there may be sunlight or turbidity in the environment of underwater.

4. **3D Reconstruction of Underwater Objects -** 3D reconstruction of scenes from imagery is well applicable only to pictures taken in air, but because of the fact that the majority of cameras are designed to control in the air, it requires underwater image processing.

5. **Underwater Archaeology -** In Underwater archaeology, historical records are extensively used. The archaeologist also studies everlasting changes in geology to trace submerged sites.

6. **Underwater Biological Research**

7. **Sea Floor Exploration -** Examination of the seafloor gives us data about global geomorphology, volcanic processes, hydrothermal venting, and habitats of deep-sea fauna. This research on physical, chemical, and biological surroundings on ocean bed for scientific purposes.

8. **Marine Archaeology -** Within archaeology, it is a branch that concerns with human communication with the lake, river and ocean by the research of physical remains and shore- side facilities.

9. **Marine Geology -** The study of the history and structure of the ocean floor is called Marine Geology or geological oceanography. It focuses on region wounded by oceans along with the deep seafloor, coastal areas, etc. Geophysical, geochemical, sedimentological and paleontological investigations of the ocean ground and coastal zone are included in this branch.

REFERENCES

[1] Zhang, Weidong, et al. "Underwater image enhancement via minimal color loss and locally adaptive contrast enhancement." *IEEE Transactions on Image Processing* 31 (2022): 3997-4010.

[2] Liu, Risheng, et al. "Twin adversarial contrastive learning for underwater image enhancement and beyond." IEEE Transactions on Image Processing 31 (2022): 4922-4936.

[3] Zhou, Jingchun, Dehuan Zhang, and Weishi Zhang. "Underwater image enhancement method via multi-feature prior fusion." Applied Intelligence (2022): 1-23.

[4] Ding, Xueyan, et al. "A unified total variation method for underwater image enhancement." Knowledge-Based Systems 255 (2022): 109751.

[5] Jiang, Qun, et al. "Two-step domain adaptation for underwater image enhancement." Pattern Recognition 122 (2022): 108324

[6] Zhuang, Peixian, et al. "Underwater Image Enhancement With Hyper-Laplacian Reflectance Priors." *IEEE Transactions on Image Processing* 31 (2022): 5442-5455.

[7] Sun, Kaichuan, Fei Meng, and Yubo Tian. "Progressive multi-branch embedding fusion network for underwater image enhancement." *Journal of Visual Communication and Image Representation* 87 (2022): 103587.

[8] Kang, Yaozu, et al. "A Perception-Aware Decomposition and Fusion Framework for Underwater Image Enhancement." IEEE Transactions on Circuits and Systems for Video Technology (2022).

[9] Sharma, Shobha, and Tarun Varma. "Graph signal processing based underwater image enhancement techniques." Engineering Science and Technology, an International Journal 32 (2022):

[10] Li, Nan, et al. "Single underwater image enhancement using integrated variationalmodel." Digital Signal Processing 129 (2022): 103660.

APPENDIX

a. SOURCE CODE

METHOD-1 HISTOGRAM EQUALIZATION

File : sceneRadianceHE.py

```
import cv2
def RecoverHE(sceneRadiance):
    for i in range(3):
        sceneRadiance[:, :, i] = cv2.equalizeHist(sceneRadiance[:, :, i])
    return sceneRadiance
```

File : sceneRadianceCLAHE.py

```
import cv2
def RecoverCLAHE(sceneRadiance):
    clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(4, 4))
    for i in range(3):
        sceneRadiance[:, :, i] = clahe.apply((sceneRadiance[:, :, i]))
    return sceneRadiance
```

File : main.py

```
import os
import numpy as np
import cv2
import natsort
import xlwt
from skimage import exposure
from sceneRadianceCLAHE import RecoverCLAHE
from sceneRadianceHE import RecoverHE

np.seterr(over='ignore')
if__name__== '_main_':
    pass
folder = "C:/Users/user/Desktop/pytho"
path = folder + "/InputImages"
files = os.listdir(path)
files = natsort.natsorted(files)
for i in range(len(files)):
    file = files[i]
    filepath = path + "/" + file
    prefix = file.split('.')[0]
    if  os.path.isfile(filepath):
        print('********    file  ********',file)
        img = cv2.imread(folder + '/InputImages/' + file)
```

METHOD-2 RAYLEIGH DISTRIBUTION

File : global_stretching_RGB.py

```
import numpy as np

def stretching(img):
    height = len(img)
    width = len(img[0])
    for k in range(0, 3):
        Max_channel = np.max(img[:,:,k])
        Min_channel = np.min(img[:,:,k])
        for i in range(height):
            for j in range(width):
                img[i,j,k] = (img[i,j,k] - Min_channel) * (255 - 0) / (Max_channel - Min_channel)+ 0
    return img
```

File : color_equalisation.py

```
import numpy as np
import cv2

def RGB_equalisation(img,height,width):
    img = np.float32(img)
    b, g, r = cv2.split(img)
    r_avg = np.mean(r)
    g_avg = np.mean(g)
    b_avg = np.mean(b)

    All_avg = np.array((r_avg,g_avg,b_avg))
    All_max = np.max(All_avg)
    All_min = np.min(All_avg)
    All_median = np.median(All_avg)
    A = All_median/All_min
    B = All_median/All_max

    if (All_min == r_avg):
        r = r * A
    if (All_min == g_avg):
        g = g * A
    if (All_min == b_avg):
        b = b * A

    if (All_max == r_avg):
        r = r * B
    if (All_max == g_avg):
        g = g * B
    if (All_max == b_avg):
        b = b * B
```

```
    sceneRadiance = np.zeros((height, width, 3), 'float64')
    sceneRadiance[:, :, 0] = b
    sceneRadiance[:, :, 1] = g
    sceneRadiance[:, :, 2] = r
    sceneRadiance = np.clip(sceneRadiance, 0, 255)
    return sceneRadiance
```

File : sceneRadiance.py

```
import numpy as np

def sceneRadianceRGB(sceneRadiance):
    sceneRadiance = np.clip(sceneRadiance, 0, 255)
    sceneRadiance = np.uint8(sceneRadiance)
    return sceneRadiance
```

File : rayleighDistribution.py

```
import numpy as np
import math
e = np.e
esp = 2.2204e-16

class NodeLower(object):
 def__init_(self,x,y,value):
        self.x = x
        self.y = y
        self.value = value
 def printInfo(self):
        print(self.x,self.y,self.value)

class Node(object):
 def__init_(self,x,y,value):
        self.x = x
        self.y = y
        self.value = value
 def printInfo(self):
        print(self.x,self.y,self.value)

def rayleighStrLower(nodes, height, width,lower_Position):
    alpha = 0.4
    selectedRange = [0, 255]
    NumPixel = np.zeros(256)
    temp = np.zeros(256)
    for i in range(0, lower_Position):
        NumPixel[nodes[i].value] = NumPixel[nodes[i].value] + 1
    ProbPixel = NumPixel / lower_Position
```

```
    CumuPixel = np.cumsum(ProbPixel)

    valSpread = selectedRange[1] - selectedRange[0]
    hconst = 2 * alpha ** 2
    vmax = 1 - e ** (-1 / hconst)
    val = vmax * (CumuPixel)
    val = np.array(val)

    for i in range(256):
        if (val[i] >= 1):
            val[i] = val[i] - esp
    for i in range(256):
        temp[i] = np.sqrt(-hconst * math.log((1 - val[i]), e))
        normalization = temp[i] * valSpread
        if(normalization > 255):
            CumuPixel[i] = 255
        else:
            CumuPixel[i] = normalization
    for i in range(0, lower_Position):
        nodes[i].value = CumuPixel[nodes[i].value]
    return nodes

def rayleighStrUpper(nodes, height, width,lower_Position):
    allSize = height*width
    alpha = 0.4
    selectedRange = [0, 255]
    NumPixel = np.zeros(256)
    temp = np.zeros(256)
    for i in range(lower_Position, allSize):
            NumPixel[nodes[i].value] = NumPixel[nodes[i].value] + 1
    ProbPixel = NumPixel / (allSize-lower_Position)
    CumuPixel = np.cumsum(ProbPixel)
    valSpread = selectedRange[1] - selectedRange[0]
    hconst = 2 * alpha ** 2
    vmax = 1 - e ** (-1 / hconst)
    val = vmax * (CumuPixel)
    val = np.array(val)

    for i in range(256):
        if (val[i] >= 1):
            val[i] = val[i] - esp
    for i in range(256):
        temp[i] = np.sqrt(-hconst * math.log((1 - val[i]), e))
        normalization = temp[i] * valSpread
        if(normalization > 255):
```

```
            CumuPixel[i] = 255
        else:
            CumuPixel[i]  = normalization
    for i in range(lower_Position, allSize):
        nodes[i].value = CumuPixel[nodes[i].value]
    return nodes

def uperLower(r, height, width):
    allSize = height * width
    R_max = np.max(r)
    R_min = np.min(r)
    R_middle = (R_max - R_min) / 2 + R_min
    R_middle = np.mean(r)
    node_upper = []
    node_lower = []
    for i in range(0, height):
        for j in range(0, width):
            oneNode = Node(i, j, r[i, j])
            oneNodeLower = NodeLower(i, j, r[i, j])
            node_upper.append(oneNode)
            node_lower.append(oneNodeLower)
    node_upper = sorted(node_upper, key=lambda node: node.value, reverse=False)
    node_lower = sorted(node_lower, key=lambda node: node.value, reverse=False)

    for i in range(allSize):
        if (node_upper[i].value > R_middle):
            # print('nodes[i].value',nodes[i].value)
            middle_Position = i
            break
    lower_Position = middle_Position

    for i in range(allSize):
        node_upper[i].value = np.int(node_upper[i].value)
        node_lower[i].value = np.int(node_lower[i].value)

    nodesLower = rayleighStrLower(node_lower, height, width,lower_Position)
    nodesUpper  = rayleighStrUpper(node_upper, height, width,lower_Position)

    array_lower_histogram_stretching = np.zeros((height, width))
    array_upper_histogram_stretching = np.zeros((height, width))

    for i in range(0, allSize):
        if(i > lower_Position):
            array_upper_histogram_stretching[nodesUpper[i].x, nodesUpper[i].y] =
nodesUpper[i].value
            array_lower_histogram_stretching[nodesUpper[i].x, nodesUpper[i].y] = 255
        else:
```

```
            array_lower_histogram_stretching[nodesLower[i].x, nodesLower[i].y] =
nodesLower[i].value
            array_upper_histogram_stretching[nodesLower[i].x, nodesLower[i].y] = 0
    return array_lower_histogram_stretching,array_upper_histogram_stretching

def rayleighStretching(sceneRadiance, height, width):
    R_array_lower_histogram_stretching, R_array_upper_histogram_stretching =
uperLower(sceneRadiance[:, :, 2], height, width)
    G_array_lower_histogram_stretching, G_array_upper_histogram_stretching =
uperLower(sceneRadiance[:, :, 1], height, width)
    B_array_lower_histogram_stretching, B_array_upper_histogram_stretching =
uperLower(sceneRadiance[:, :, 0], height, width)

    sceneRadiance_Lower = np.zeros((height, width, 3), )
    sceneRadiance_Lower[:, :, 0] = B_array_lower_histogram_stretching
    sceneRadiance_Lower[:, :, 1] = G_array_lower_histogram_stretching
    sceneRadiance_Lower[:, :, 2] = R_array_lower_histogram_stretching
    sceneRadiance_Lower = np.uint8(sceneRadiance_Lower)

    sceneRadiance_Upper = np.zeros((height, width, 3))
    sceneRadiance_Upper[:, :, 0] = B_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 1] = G_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 2] = R_array_upper_histogram_stretching
    sceneRadiance_Upper = np.uint8(sceneRadiance_Upper)

    return sceneRadiance_Lower, sceneRadiance_Upper
```

File : rayleighDistributionLower.py

```
import numpy as np
import math
import cv2
from skimage.color import rgb2hsv,hsv2rgb

e = np.e
esp = 2.2204e-16

class Node(object):
 def__init_(self,x,y,value):
            self.x = x
            self.y = y
            self.value = value
 def printInfo(self):
            print(self.x,self.y,self.value)

def rayleighStrLower(nodes, height, width,lower_Position):
    alpha = 0.5
    selectedRange = [0, 255]
```

```
    NumPixel = np.zeros(256)
    temp = np.zeros(256)
    for i in range(0, lower_Position):
        NumPixel[nodes[i].value] = NumPixel[nodes[i].value] + 1
    ProbPixel = NumPixel / lower_Position
    CumuPixel = np.cumsum(ProbPixel)

    valSpread = selectedRange[1] - selectedRange[0]
    hconst = 2 * alpha ** 2
    vmax = 1 - e ** (-1 / hconst)
    val = vmax * (CumuPixel)
    val = np.array(val)

    for i in range(256):
        if (val[i] >= 1):
            val[i] = val[i] - esp
    for i in range(256):
        temp[i] = np.sqrt(-hconst * math.log((1 - val[i]), e))
        normalization = temp[i] * valSpread
        if(normalization > 255):
            CumuPixel[i] = 255
        elif(normalization<1):
            normalization =1
        else:
            CumuPixel[i] = normalization

    for i in range(0, lower_Position):
        nodes[i].value = CumuPixel[nodes[i].value]
    return nodes

def uperLower(r, height, width):
    allSize = height * width
    R_max = np.max(r)
    R_min = np.min(r)
    R_middle = (R_max - R_min) / 2 + R_min
    nodes = []
    for i in range(0, height):
        for j in range(0, width):
            oneNode = Node(i, j, r[i, j])
            nodes.append(oneNode)
    nodes = sorted(nodes, key=lambda node: node.value, reverse=False)
    for i in range(240000):
        if (nodes[i].value > R_middle):
            middle_Position = i
            break
```

```
    lower_Position = middle_Position

    for i in range(240000):
        nodes[i].value = np.int(nodes[i].value)
    nodesLower = rayleighStrLower(nodes, height, width,lower_Position)

    array_lower_histogram_stretching = np.zeros((height, width))

    for i in range(0, allSize):
        if(i > lower_Position):
            array_lower_histogram_stretching[nodes[i].x, nodes[i].y] = 250
        else:
            array_lower_histogram_stretching[nodes[i].x, nodes[i].y] = nodesLower[i].value
    return array_lower_histogram_stretching

def rayleighStretching_Lower(sceneRadiance, height, width):
    img_hsv = rgb2hsv(sceneRadiance)
    h, s, v = cv2.split(img_hsv)
    s = s * 255
    v = v * 255

    v_array_lower_histogram_stretching = uperLower(v, height, width)/255
    s_array_lower_histogram_stretching = uperLower(s, height, width)/255
    h_array_lower_histogram_stretching = h

    sceneRadiance_Lower = np.zeros((height, width, 3))
    sceneRadiance_Lower[:, :, 0] = h_array_lower_histogram_stretching
    sceneRadiance_Lower[:, :, 1] = s_array_lower_histogram_stretching
    sceneRadiance_Lower[:, :, 2] = v_array_lower_histogram_stretching

    img_rgb = hsv2rgb(sceneRadiance_Lower)*255
      for i in range(0, 3):
        for j in range(0, img_rgb.shape[0]):
            for k in range(0, img_rgb.shape[1]):
                if img_rgb[j, k, i] > 250:
                    img_rgb[j, k, i] = 250
                if img_rgb[j, k, i] < 5:
                    img_rgb[j, k, i] = 5
    sceneRadiance_Lower = np.uint8(img_rgb)
    return sceneRadiance_Lower
```

File :rayleighDistributionUpper.py

```
import numpy as np
import math
import cv2
```

```
from skimage.color import rgb2hsv,hsv2rgb

e = np.e
esp = 2.2204e-16

class Node(object):
 def__init_(self,x,y,value):
            self.x = x
            self.y = y
            self.value = value
 def printInfo(self):
            print(self.x,self.y,self.value)

def rayleighStrUpper(nodes, height, width,lower_Position):
    allSize = height * width
    alpha = 0.5
    selectedRange = [0, 255]
    NumPixel = np.zeros(256)
    temp = np.zeros(256)
    for i in range(lower_Position, allSize):
          NumPixel[nodes[i].value] = NumPixel[nodes[i].value] + 1
    ProbPixel = NumPixel / (allSize-lower_Position)
    CumuPixel = np.cumsum(ProbPixel)
    valSpread = selectedRange[1] - selectedRange[0]
    hconst = 2 * alpha ** 2
    vmax = 1 - e ** (-1 / hconst)
    val = vmax * (CumuPixel)
    val = np.array(val)

    for i in range(256):
       if (val[i] >= 1):
          val[i] = val[i] - esp
    for i in range(256):
       temp[i] = np.sqrt(-hconst * math.log((1 - val[i]), e))
       normalization = temp[i] * valSpread
       if(normalization > 255):
          CumuPixel[i] = 250
       else:
          CumuPixel[i]  = normalization
    for i in range(lower_Position, allSize):
       nodes[i].value = CumuPixel[nodes[i].value]
    return nodes
```

```
def uperLower(r, height, width):
    allSize = height * width
    R_max = np.max(r)
    R_min = np.min(r)
    R_middle = (R_max - R_min) / 2 + R_min
    nodes = []
    for i in range(0, height):
        for j in range(0, width):
            oneNode = Node(i, j, r[i, j])
            nodes.append(oneNode)
    nodes = sorted(nodes, key=lambda node: node.value, reverse=False)

    for i in range(240000):
        if (nodes[i].value > R_middle):
            # print('nodes[i].value',nodes[i].value)
            middle_Position = i
            break
    lower_Position = middle_Position

    for i in range(240000):
        nodes[i].value = np.int(nodes[i].value)

    nodesUpper = rayleighStrUpper(nodes, height, width,lower_Position)
    array_upper_histogram_stretching = np.zeros((height, width))

        for i in range(0, allSize):
        if(i > lower_Position):
            array_upper_histogram_stretching[nodes[i].x, nodes[i].y] = nodesUpper[i].value
        else:
            array_upper_histogram_stretching[nodes[i].x, nodes[i].y] = 5
    return array_upper_histogram_stretching


def rayleighStretching_Upper(sceneRadiance, height, width):
    img_hsv = rgb2hsv(sceneRadiance)
    h, s, v = cv2.split(img_hsv)
    s = s * 255
    v = v * 255

    v_array_upper_histogram_stretching = uperLower(v, height, width)/255
    s_array_upper_histogram_stretching = uperLower(s, height, width)/255
    h_array_upper_histogram_stretching = h
    sceneRadiance_Upper = np.zeros((height, width, 3))
    sceneRadiance_Upper[:, :, 0] = h_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 1] = s_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 2] = v_array_upper_histogram_stretching
```

```
    img_rgb = hsv2rgb(sceneRadiance_Upper) * 255
    for i in range(0, 3):
        for j in range(0, img_rgb.shape[0]):
            for k in range(0, img_rgb.shape[1]):
                if img_rgb[j, k, i] > 250:
                    img_rgb[j, k, i] = 250
                if img_rgb[j, k, i] < 5:
                    img_rgb[j, k, i] = 5
    sceneRadiance_Upper = np.uint8(img_rgb)
    return sceneRadiance_Upper
```

File : histogramDistributionLower.py

```
import numpy as np
import math
import cv2

def histogram_rgbLower(RGB_array,height, width):
    RGB_min = np.min(RGB_array)
    RGB_max = np.max(RGB_array)
    RGB_middle = (RGB_max + RGB_min) / 2
    array_upper_histogram_stretching = np.zeros((height, width))
    R_predicted_min = 255 * 0.05
    for i in range(0, height):
        for j in range(0, width):
            if RGB_array[i][j] < RGB_middle:
                p_out = int((( RGB_array[i][j] - RGB_min) * ((255 - R_predicted_min) /
(RGB_max - RGB_middle)) + R_predicted_min))
                array_upper_histogram_stretching[i][j] = p_out
            else:
                array_upper_histogram_stretching[i][j] = 255
    return array_upper_histogram_stretching

def histogramStretching_Lower(sceneRadiance, height, width):
    sceneRadiance = np.float64(sceneRadiance)
    b, g, r = cv2.split(sceneRadiance)
    R_array_Lower_histogram_stretching = histogram_rgbLower(r, height, width)
    G_array_Lower_histogram_stretching = histogram_rgbLower(g, height, width)
    B_array_Lower_histogram_stretching = histogram_rgbLower(b, height, width)

    sceneRadiance_Lower = np.zeros((height, width, 3))
    sceneRadiance_Lower[:, :, 0] = B_array_Lower_histogram_stretching
    sceneRadiance_Lower[:, :, 1] = G_array_Lower_histogram_stretching
    sceneRadiance_Lower[:, :, 2] = R_array_Lower_histogram_stretching
    sceneRadiance_Lower = np.uint8(sceneRadiance_Lower)
    return sceneRadiance_Lower
```

File : histogramDistributionUpper.py

```
import numpy as np
import math
import cv2

def histogram_rgbUpper(RGB_array,height, width):
    RGB_min = np.min(RGB_array)
    RGB_max = np.max(RGB_array)
    RGB_middle = (RGB_max + RGB_min) / 2
    array_upper_histogram_stretching = np.zeros((height, width))
    R_predicted_max = 255 * 0.95

    for i in range(0, height):
        for j in range(0, width):
            if RGB_array[i][j] < RGB_middle:
                array_upper_histogram_stretching[i][j] = 0
            else:
                p_out = int((RGB_array[i][j] - RGB_middle) * ((R_predicted_max) / (RGB_max - RGB_middle)))
                array_upper_histogram_stretching[i][j] = p_out
    return array_upper_histogram_stretching

def histogramStretching_Upper(sceneRadiance, height, width):
    sceneRadiance = np.float64(sceneRadiance)
    b, g, r = cv2.split(sceneRadiance)
    R_array_upper_histogram_stretching = histogram_rgbUpper(r, height, width)
    G_array_upper_histogram_stretching = histogram_rgbUpper(g, height, width)
    B_array_upper_histogram_stretching = histogram_rgbUpper(b, height, width)
    sceneRadiance_Upper = np.zeros((height, width, 3))
    sceneRadiance_Upper[:, :, 0] = B_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 1] = G_array_upper_histogram_stretching
    sceneRadiance_Upper[:, :, 2] = R_array_upper_histogram_stretching
    sceneRadiance_Upper = np.uint8(sceneRadiance_Upper)

    return sceneRadiance_Upper
```

File : hsv_Stretching.py

```
import cv2
from skimage.color import rgb2hsv,hsv2rgb
import numpy as np
from global_Stretching_SV import global_stretching

def HSVStretching(sceneRadiance):
    sceneRadiance = np.clip(sceneRadiance, 0, 255)
    sceneRadiance = np.uint8(sceneRadiance)
    height = len(sceneRadiance)
```

```
    width    =    len(sceneRadiance[0])
    img_hsv = rgb2hsv(sceneRadiance)
    img_hsv[:, :, 1] = global_stretching(img_hsv[:, :, 1], height, width)
    img_hsv[:, :, 2] = global_stretching(img_hsv[:, :, 2], height, width)
    img_rgb = hsv2rgb(img_hsv) * 255
    return img_rgb
```

File : global_Stretching_SV.py

```
import numpy as np
def global_stretching(img_L,height, width):
    length = height * width
    R_rray = []
    for  i  in  range(height):
        for j in range(width):
            R_rray.append(img_L[i][j])
    R_rray.sort()
    I_min = R_rray[int(length / 100)]
    I_max = R_rray[-int(length / 100)]
    # print('I_min',I_min)
    # print('I_max',I_max)
    array_Global_histogram_stretching_L = np.zeros((height, width))
    for i in range(0, height):
        for j in range(0, width):
            if img_L[i][j] < I_min:
                p_out = img_L[i][j]
                array_Global_histogram_stretching_L[i][j] = p_out
            elif (img_L[i][j] > I_max):
                p_out = img_L[i][j]
                array_Global_histogram_stretching_L[i][j] = p_out
            else:
                p_out = (img_L[i][j] - I_min) * ((1) / (I_max - I_min))
                array_Global_histogram_stretching_L[i][j] = p_out
    return (array_Global_histogram_stretching_L)
```

File : main.py

```
import math
import os
import natsort
import numpy as np
import datetime
import cv2
from skimage.color import rgb2hsv
from color_equalisation import RGB_equalisation
```

```
from global_stretching_RGB import stretching
from hsvStretching import HSVStretching

from histogramDistributionLower import histogramStretching_Lower
from histogramDistributionUpper import histogramStretching_Upper
from rayleighDistribution import rayleighStretching
from rayleighDistributionLower import rayleighStretching_Lower
from rayleighDistributionUpper import rayleighStretching_Upper
from sceneRadiance import sceneRadianceRGB
e = np.e
esp = 2.2204e-16
np.seterr(over='ignore')
if_name_== ' main ':
   pass
starttime = datetime.datetime.now()
folder = "C:/Users/user/Desktop/pytho"
path = folder + "/InputImages"
files = os.listdir(path)
files =  natsort.natsorted(files)

for i in range(len(files)):
   file = files[i]
   filepath = path + "/" + file
   prefix = file.split('.')[0]
   if os.path.isfile(filepath):
      print('********    file ********',file)
      img = cv2.imread(folder + '/InputImages/' + file)
      prefix = file.split('.')[0]
      height = len(img)
      width = len(img[0])

      sceneRadiance = RGB_equalisation(img, height, width)
      sceneRadiance = stretching(sceneRadiance)
      sceneRadiance_Lower, sceneRadiance_Upper = rayleighStretching(sceneRadiance,
height, width)
      sceneRadiance = (np.float64(sceneRadiance_Lower) +
np.float64(sceneRadiance_Upper)) / 2

      sceneRadiance = HSVStretching(sceneRadiance)
      sceneRadiance = sceneRadianceRGB(sceneRadiance)
      cv2.imwrite('OutputImages/' + prefix + '_RayleighDistribution.jpg', sceneRadiance)

endtime = datetime.datetime.now()
time = endtime-starttime
print('time',time)
```

METHOD-3 RELATIVE GLOBAL HISTOGRAM STRETCHING

File : global_stretching_RGB.py

```
import numpy as np
def stretching(img):
    height = len(img)
    width = len(img[0])
    for k in range(0, 3):
        Max_channel = np.max(img[:,:,k])
        Min_channel = np.min(img[:,:,k])
        for i in range(height):
            for j in range(width):
                img[i,j,k] = (img[i,j,k] - Min_channel) * (255 - 0) / (Max_channel - Min_channel)+ 0
    return img
```

File : stretchRange.py

```
import from scipy import stats
import numpy as np

def stretchrange(r_array, height, width):
    length = height * width
    R_rray = r_array.flatten()
    R_rray.sort()
    print('R_rray', R_rray)
    mode = stats.mode(R_rray).mode[0]
    mode_index_before = list(R_rray).index(mode)
    SR_min = R_rray[int(mode_index_before * 0.005)]
    SR_max = R_rray[int(-(length - mode_index_before) * 0.005)]
    print('mode',mode)
    print('SR_min',SR_min)
    print('SR_max',SR_max)
    return SR_min, SR_max, mode
```

File : S_model.py

```
import numpy as np
import  pylab  as  pl
x = []
y = []
for i in range(-128,127):
    x.append(i)

for j in range(-128,127):
    temp  = j *(2**(1 - abs((j/128))))
```

```
        y.append(temp)

    pl.axis([-128, 127,-128, 127])
    pl.title('S-model Curve Function ',fontsize=20)
    pl.xlabel('Input Value',fontsize=20)
    pl.ylabel('Output  Value',fontsize=20)
    pl.plot(x, y,color='red')
    pl.show()
```

File : desiredRange.py

```
    from scipy import stats
    import numpy as np

    def stretchrange(r_array, height, width):
        length = height * width
        R_rray = r_array.flatten()
        R_rray.sort()
        print('R_rray', R_rray)
        mode = stats.mode(R_rray).mode[0]
        mode_index_before = list(R_rray).index(mode)
        DR_min = (1-0.655) * mode
        SR_max = R_rray[int(-(length - mode_index_before) * 0.005)]
        print('mode', mode)
        print('DR_min', DR_min)
        print('SR_max', SR_max)
        return DR_min, SR_max, mode
```

File : color_equalisation.py

```
    import numpy as np

    def cal_equalisation(img,ratio):
        Array = img * ratio
        Array = np.clip(Array, 0, 255)
        return Array

    def RGB_equalisation(img):
        img = np.float32(img)
        avg_RGB = []
        for i in range(3):
            avg = np.mean(img[:,:,i])
            avg_RGB.append(avg)
        avg_RGB = 128/np.array(avg_RGB)
        ratio = avg_RGB
        for i in range(0,2):
            img[:,:,i] = cal_equalisation(img[:,:,i],ratio[i])
        return img
```

File : relativeglobalhistogramstretching.py

```
import math
import numpy as np
from stretchRange import stretchrange
pi = math.pi
e = math.e
from scipy import stats

def global_stretching(r_array, height, width, lamda, k):

    length = height * width
    R_rray = []
    for i in range(height):
        for j in range(width):
            R_rray.append(r_array[i][j])
    R_rray.sort()
    I_min = R_rray[int(length / 200)]
    I_max = R_rray[-int(length / 200)]

    array_Global_histogram_stretching = np.zeros((height, width))
    d = 4
    length = height * width
    R_rray = []

    SR_min, SR_max, mode = stretchrange(r_array, height, width)
    DR_min = (1 - 0.655) * mode
    t_n = lamda ** d
    O_max_left = SR_max * t_n * k/ mode
    O_max_right = 255 * t_n * k/ mode
    Dif = O_max_right -O_max_left
    if(Dif >= 1):
        sum = 0
        for i in range(1, int(Dif+1)):
            sum = sum + (1.526+ i) * mode / (t_n * k)
        DR_max = sum/int(Dif)


        for i in range(0, height):
            for j in range(0, width):
                if r_array[i][j] < I_min:
                    p_out = (r_array[i][j] - I_min) * ( DR_min /I_min ) + I_min
                    array_Global_histogram_stretching[i][j] = p_out
                elif (r_array[i][j] > I_max):
                    p_out = (r_array[i][j] - DR_max) * (DR_max / I_max) + I_max
                    array_Global_histogram_stretching[i][j] = p_out
                else:
                    p_out = int((r_array[i][j] - I_min) * ((255 - I_min) / (I_max - I_min))) + I_min
```

```
                array_Global_histogram_stretching[i][j] = p_out
    else:

        if r_array[i][j] < I_min:
            p_out = (r_array[i][j] - np.min(r_array)) * (DR_min / np.min(r_array)) +
np.min(r_array)
            array_Global_histogram_stretching[i][j] = p_out
        else:
            p_out = int((r_array[i][j] - I_min) * ((255 - DR_min) / (I_max - I_min))) + DR_min
            array_Global_histogram_stretching[i][j] = p_out
    return (array_Global_histogram_stretching)

def RelativeGHstretching(sceneRadiance, height, width):
    sceneRadiance[:, :, 0] = global_stretching(sceneRadiance[:, :, 0], height, width, 0.97, 1.25)
    sceneRadiance[:, :, 1] = global_stretching(sceneRadiance[:, :, 1], height, width, 0.95, 1.25)
    sceneRadiance[:, :, 2] = global_stretching(sceneRadiance[:, :, 2], height, width, 0.83, 0.85)
    return sceneRadiance
```

File : LabStretching.py

```
import cv2
from skimage.color import rgb2hsv,hsv2rgb
import numpy as np
from skimage.color import rgb2lab, lab2rgb
from global_StretchingL import global_stretching
from global_stretching_ab import global_Stretching_ab

def LABStretching(sceneRadiance):
    sceneRadiance = np.clip(sceneRadiance, 0, 255)
    sceneRadiance = np.uint8(sceneRadiance)
    height = len(sceneRadiance)
    width = len(sceneRadiance[0])
    img_lab = rgb2lab(sceneRadiance)
    L, a, b = cv2.split(img_lab)

    img_L_stretching = global_stretching(L, height, width)
    img_a_stretching = global_Stretching_ab(a, height, width)
    img_b_stretching = global_Stretching_ab(b, height, width)

    labArray = np.zeros((height, width, 3), 'float64')
    labArray[:, :, 0] = img_L_stretching
    labArray[:, :, 1] = img_a_stretching
    labArray[:, :, 2] = img_b_stretching
    img_rgb = lab2rgb(labArray) * 255

    return img_rgb
```

```
    sceneRadiance = np.clip(sceneRadiance, 0, 255)
    sceneRadiance = np.uint8(sceneRadiance) height
    = len(sceneRadiance)
    width = len(sceneRadiance[0])
    img_hsv = rgb2hsv(sceneRadiance)
    img_hsv[:, :, 1] = global_stretching(img_hsv[:, :, 1], height, width)
    img_hsv[:, :, 2] = global_stretching(img_hsv[:, :, 2], height, width)
    img_rgb = hsv2rgb(img_hsv) * 255

    return img_rgb
```

File : global_stretching_ab.py

```
import numpy as np
import math
e = math.e

def global_Stretching_ab(a,height, width):
    array_Global_histogram_stretching_L = np.zeros((height, width), 'float64')
    for i in range(0, height):
        for j in range(0, width):
            p_out = a[i][j] * (1.3 ** (1 - math.fabs(a[i][j] / 128)))
            array_Global_histogram_stretching_L[i][j] = p_out
    return (array_Global_histogram_stretching_L)
```

File : global_stretchingL.py

```
import numpy as np

def global_stretching(img_L,height, width):
    length = height * width
    R_rray = (np.copy(img_L)).flatten()
    R_rray.sort()
    print('R_rray',R_rray)
    I_min = int(R_rray[int(length / 100)])
    I_max = int(R_rray[-int(length / 100)])
    print('I_min',I_min)
    print('I_max',I_max)
    array_Global_histogram_stretching_L = np.zeros((height, width))
    for i in range(0, height):
        for j in range(0, width): if
            img_L[i][j] < I_min:
                p_out = img_L[i][j]
                array_Global_histogram_stretching_L[i][j] = 0
            elif (img_L[i][j] >
                I_max):p_out =
                img_L[i][j]
            array_Global_histogram_stretching_L[i][j] = else:
                p_out = int((img_L[i][j] - I_min) * ((100) / (I_max - I_min)))
                array_Global_histogram_stretching_L[i][j] = p_out
```

```
    return (array_Global_histogram_stretching_L)
```

File : main.py

```
import os
import numpy as np
import cv2
import natsort
import datetime
from LabStretching import LABStretching
from color_equalisation import RGB_equalisation
from global_stretching_RGB import stretching
from relativeglobalhistogramstretching import RelativeGHstretching

np.seterr(over='ignore')
if__name__== '_main_':pass
starttime = datetime.datetime.now()

folder = "C:/Users/user/Desktop/pytho"
path = folder + "/InputImages"
files = os.listdir(path)
files =  natsort.natsorted(files)

for i in range(len(files)):
   file = files[i]
   filepath = path + "/" + file
   prefix = file.split('.')[0]
   if  os.path.isfile(filepath): print('********
                                file  ********',file)
      img = cv2.imread(folder +'/InputImages/' + file)

      height = len(img) width
      = len(img[0])

      sceneRadiance = img
      sceneRadiance = stretching(sceneRadiance)
      sceneRadiance = LABStretching(sceneRadiance)

      cv2.imwrite('OutputImages/' + prefix + '_RGHS.jpg', sceneRadiance)
```

b. SCREENSHOTS

Input images

Output images

C.RESEARCH PAPER

UNDERWATER IMAGE PROCESSING AND ENHANCEMENT

Pennama Hemasagarreddy
Dept.of Computer Science Engineering(B.E.)
Guide - Ms.Kabitha
Mail id-kabitha.cse@sathyabama.ac.in
Sathyabama Institite Of Science and Technology
Chennai, India
hemasagarreddy2002@gmail.com

S.Keerthivas
D*ept.of Computer Science Engineering(B.E.)*
Guide – Ms.Kabitha
Sathyabama Institite Of Science and Technology
Chennai, India
s.keerthivas02@gmail.com

Abstract*— Restoring a high-quality video clip of aquatic environments is useful for a wide variety of disciplines, including marine ecology, aquatic archaeology, aquatic natural recognition, & aquatic biology. Photograph shots in water are more complicated and less clear than those taken on dry land. In order to create an underneath picture, light has to be absorbed by the water as well as the particulates found in water have to modify the stream of reflected light by the aquatic environment before it reach the lens. To fix these common problems in underwater photography, we suggest a simple yet efficient technique called relative histogram stretching.

Keywords*—: *Iteration, Image enhancement, Color contrast, Hybrid filter, Noise reduction

I. INTRODUCTION

The oceans is vital to the survival of Earth's life because it contains an incredible variety of strange, nameless organisms as well as large quantities of energy. As a result, a lot of people all across the world are working hard to do some serious high-tech marine research. Experts strive to collect high-resolution undersea images for use in fields as diverse as mobile robots, search and rescue the analysis of man-made objects, environmental surveillance, including real-time guidance. The chemical & physical features of underwater environments, though, have a profound effect onunderwater photography quality, creating challenges that aresimpler to overcome in surface photography. Underwater photographs, for example, always have a colored tint, such as a greenish-blue tint due to the different absorption proportions of blue, green and red photons.

The importance of restoring high-quality undersea pictures for fields like coastal environment and underwater archaeology cannot be overstated. Underwater imaging and sensing for robots. Light is continually being reflected and deflected by air particles before it reaches the camera. There is hardly much water. In water, the dissipation rate of light varies with its wavelength. Images shot underwater seem blue because red is more affected than green or blue. Artificial light is often used to supplement natural light for underwater

imaging. Lighting situations underwater with artificial light isa common way to improve photographs taken there. Artificialillumination, however, may be affected by absorption and dispersion. Since non-uniform illumination is being introduced simultaneously, the underwater picture has a bright spot in the centre. When compared to images taken on land or in the sky, the predominant hues in underwater photography tend to be blue and green. However, vision is drastically reduced dueto the substantial absorption coefficient in water in comparison to air and the increased incoming light dispersion. So, things that arevery far away from the acquisition equipment or the observer, in addition to those that are somewhat near in certain conditions, have very poor visibility and little contrast with their environment.

II. LITERATURE SURVEY

A innovative approach to improving undersea photos that cangradually adjust colours, focus, and brightness. This is because the suggested solution incorporates both multi- channel colour compensation and colour correction. A Gaussdispersion pyramids as well as the brightness constrained adaptable normalisation both help with the problem of blurred details and poor contrast [1]. The suggested technique centreson four fundamental tenets: colour compensation, color management, details enhancement, & contrasts improvement.Both statistical & perceptual evaluations show that the suggested technique successfully eliminates picture blur, achieves retouching, and vastly improves image quality.

Underwater photos often have distorted color & blurry features because of absorption coefficient, diffraction, & scatter that depends on the light's wavelength that is being used to capture the image. [2,3]We suggest an innovative system for enhancing visual feature data, named SGUIE- Net,and show how to use semantics as rising instruction bytraining improvement features for different regions. With this in mind, we present a 8s2emantics region-wise improvement modules to best learn local enrichment characteristics for semantics areas with multi-scale perceptions. Having fedthem as supplementary characteristics into the study in the

deep levels are suggested to be integrated using a UFPN. Our technique has been shown to achieve 1.4%mAP & 1.0%mAP in thorough testing and thorough evaluations upon that URPC2020 datasets.

In order to get superior marine photographs, the authors of this research advocated the use of an ADMNNet. We suggest ADMB to enhance the variety of image features by trying to merge the characteristics of various RFs into a solitary framework, as opposed to established CNN-based UIE channels, which participant consent a remedied RF dimensions of neurones in one set of feature. For instance, ADMB is comprised of a MCAM & DFSM. [5,6] Since embedded particulates absorb and dispersion of light, it's common for captured submerged photos to experience from significant quality deterioration, including such color changes & degradation of clarity. To address these degenerative concerns, we present a DJCNET that uses two independent branches to maintain the diverse and distinctivequalities of undersea pictures. [7] The selected absorbing of lights in waters as well as the elimination of distortions are the goals of the development of the light absorbing correction branches, while the adjustment of the blur generated by dispersion is the focus of the layout of a light dispersion corrections subsidiary.

Because of the enormity of the aquatic ecosystem, the data included in a single submerged picture is inadequate, making it difficult to match the objectives of marine studies.
[8] In this work, the authors offer a MFPF approach to improving the graphical fidelity of undersea photos by identifying & combining several characteristic prior record of the pictures. This layout takes full benefit from the advancements in white balancing, assisted filtration, & various exposure series technologies. [9] Additionally, it improves marine photography by fusing together characteristics from different scales. From theseexperiments, we learn that this design effectively addressesa wide range of degeneration issues, eliminates excessive augmentation, and enhances shadowy areas by using a multi-feature previous fusion technique.

III. EXISTING METHODOLOGY

As a method for modifying intensity in image analysis, proposed method takes use of the picture's distribution. This method often improves the general contrast of several photographs when the useful information they contain is expressed by near contrasting ratios. This modification may be used to more uniformly distribute the intensities over the histogram. The result is that areas with little local contrast may gain it. Histogram equalisation is used to effectively distribute the most common pixel intensity. This effect is particularly effective when the foreground and backdrop of a photograph are the same tone. In instance, this method may increase the amount of clarity in underexposed or overexposed photographs, as well as enrich x-ray images of skeletal anatomy.

The system's invertibility and simple method are two major strengths. For this reason, in principle, the initial histogram may be restored by knowing the histogram-based algorithm.

This does not need a lot of processing power. The technique's lack of selectivity is one of its drawbacks. However, the amount of useable signal may decrease as a result of the viewpoints of ambient noise.
Histogram Equalization Calculation:

$$(v) = round\left(\frac{cdf(v) - cdf_{min}}{(M*N) - cdf_{min}} * (L-1)\right)$$

cdf(v) – Cumulative Distribution Function of the histogram table.

cdfmin – Minimum Cumulative Distribution Frequency.

M – Width of the gray scale Image.

N – Length of the gray scale Image.

L – Number of grey levels used (Usually 256)

Note : The cdf must be normalized to [0, 255]

Data augmentation may also be used on color photos by applying the same method separately to the blue, green, and red components of the image's Rgb data. Applying the same method on the blue, green, and red components of an RGB picture, nevertheless, may result in observable modifications to the image's colour balancing when the operation adjusts the respective proportions of the hue regions. If the picture is transformed to some other colour palette, like Lab colors space & HSV/HSL colors language in specific, the approach may be applied to the luminance or values channels without affecting the tone or chroma of the picture. Equalization histograms may be performed in a number of ways in three-dimensional space. Histogram in 3D colour space was used by Venetsanopoulos & Trahanias.

On the downside, it causes whitening, where brighter pixels are more likely to be selected than darker ones. To achieve this, Han advocated using a novel functional form determined by the iso-luminance surface.

IV. PROPOSED METHODOLOGY

There are two main components of the proposed technique for enhancing images: contrast restoration & color management. Once the channels have been divided up, the improved Van Schick concept is implemented. The average (Ravg, Gavg, and Bavg) of the three color channels are determined.

The sample mean, determined by averaging those three parameters, is then used to set the standard and target levels, instead of the greatest value. The two extra streams are divided by the integrators A & B, that are in turn obtained using equation(1.1) as well as (1.2).

$$A = \frac{median(R_{avg}, G_{avg}, B_{avg})}{\min(R_{avg}, G_{avg}, B_{avg})} \quad \text{----(1.1)}$$

$$B = \frac{median(R_{avg}, G_{avg}, B_{avg})}{\max(R_{avg}, G_{avg}, B_{avg})} \quad \text{---(1.2)}$$

The spectrum of the picture is then extended evenly. The distribution is then divided into a lower & a higher half, determined by the median value. The histogram's minimum and maximum values are each expanded to cover the whole span of the Distribution function. The distribution of the resulting picture is then extended worldwide. Once an average has been determined, the histogram is split into a lower & upper section. Using the Raleigh distributions as a baseline, the minimum and maximum ranges of the graph are expanded to include the whole contrast ratio. A Rayleigh distributed was stretched using the formula(1.1). Integrating the stretch & Rayleigh formulae yields this expression. Pixels "pin" represents the source pixels, while "imin" & "imax" represent the lowest and highest intensities in the given picture channels. The variable an of the distributed.

V. STRUCTURED FLOW DIAGRAM AND APPROACH

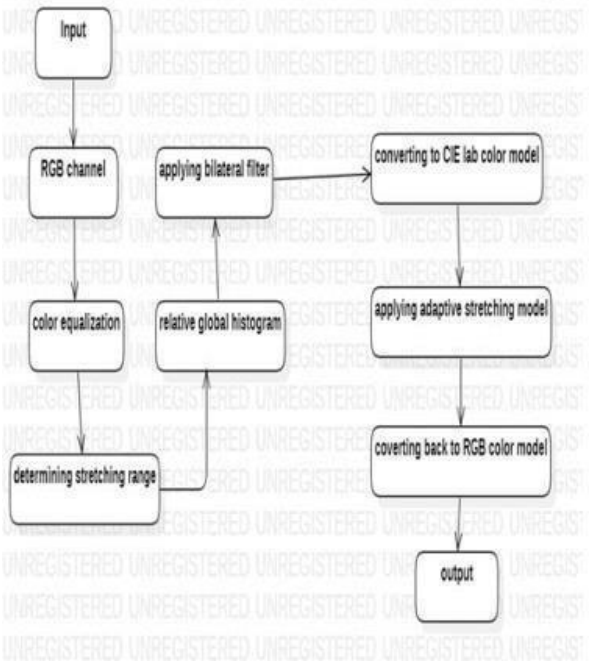


Fig.1. Flow Diagram of our system

Color restoration & RGHS are applied to the picture after B, G, R channels have been decomposed. Following the abovementioned conversion, the bidirectional filtering is used to eliminate the noise while maintaining the intended bright undersea picture's finer features. That way, the hue cast & contrast adjustment that come from light absorbing and scattering will be gone. To adjust for such "a" & "b" elements in CIE-Lab colour spaces, the "L" element of the picture is extended worldwide throughout in the color grading step. The brilliance & intensity of the colours in the picture will be improved by the adaptable extending of "L", "a", & "b." In the last phase, we use five different amounts. There are three primary stages in our suggested procedure: Adjusting the contrast, adjusting the colours, and evaluating the overall quality are the three main steps.

A) Extension & Contraction in the CIE-Lab color model -

The picture will undergo hue restoration when the RGB brightness modification has been made. The procedure involves converting the submerged picture to the Cie space to improve colour accuracy. The Cie-lab space defines image brightness as the "L" element, with L = 100 indicating the highest possible image and L = 0 indicating the lowest possible image. If both a and b equal 0, the colourchannels will show completely neutral grayscale. It is necessary to alter the "a" and "b" constituent hue gradients at outputting to get the specific color restoration. The "L" factor may also be used to adjust the overall contrast of the picture. Initially, the Cie scheme breaks down the relatively shallow picture into its constituent components. Ranges between 0.1% to 99.9% are expanded to ranges [0, 100] when the "L" factor is added using quadratic sliding extending mentioned in (8). The numbers at the bottom and top.1% of the picture are both fixed at 100. The interval [- 128, 127] represents the possiblevalues for the "a" & "b" elements, with middle value meaning 0. Extending "a" with "b" results in the S-shaped curves of the theory (2).

$$p_{\chi} = I_{\chi}^{*}\left(\varphi^{\,1-|\overline{\ }|}_{128}\right), X \in \{a, b\} \quad \text{---(2)}$$

The numbers are exponentially extended in this formula, thus the nearer they are to zero, the more they are extended. Color & contrast are what really make or break an image's legibility. There is enough contrast between foreground and backdrop that objects can be distinguished. Its "L", "a", & "b" bits of the CIE-Lab color image are dynamically extended before being merged and translated down to the Rgb image. The ultimate, perceptible outgoing picture may be made by adjusting the intensity and colours of the image pixels.

B) The Color Space of the CIE-Lab -

The Cielab system, often referred as the L*a*b* hue orbit, was established by the Intergovernmental Panel on Lighting. CIELAB must always be referenced to by adding an asterisk (*) after the word "Lab" to prevent any potential misunderstanding with Hunter Lab. There are three numbers involved in representing colour there: L* for how bright something is, a* for how reddish that was, b* for how bluish that was, & y* for how yellowish that was. The CIELAB is

developed to be a homogeneous sensory field, thus a given quantitative increase would map to a matching observed color changes. Despite its lack of complete sensory uniformity, the LAB space has found practical use in business for detecting small change in color.

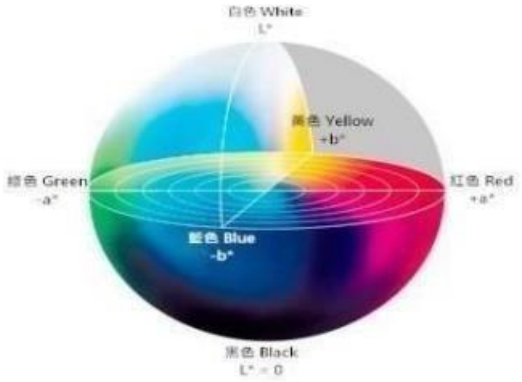


Fig.2. CIE-Lab color space

In its three dimensions, the CIELAB region encompasses the whole spectrum of colours visible to the human eye. An adversary duo is made up of red with green, while a competitor pair is made up of blue & yellow, according to the adversary colour model in human sight. The brightness factor L*, or "Lstar," sets 0 as black & 100 as white. Neg numbers on the a* axis are towards the green adversary colour, while values are towards the red adversary colour. Blue & yellow are opposites on the b* axis, having negative sign moving away from blue & positive values moving towards yellow. Visual system is sensitive to a wide range of colours, hence the a* & b* vectors have no limits and therefore can readily go above 150, dependent on the baseline white. Unfortunately, for the sake of efficiency, software packages often limit these numbers. Clamping a* & b* towards the spectrum 128 to 127 is typical in situations when integer mathematics is being applied, for illustration.

Fig.3. Comparison of before and after use of our model images

VI. CONCLUSION

The method of Relative Global Histogram Stretching is an improvement over the standard Histogram Equalization. The revised Von Kreis theory is easily implemented by extending the global histogram, splitting it, and reassembling the bottom & top halves of the graph to use the mean or midway. When making adjustments to colours, we use the HSV paradigm to get good results. We analysed the issues plaguing shallow-water photos and the methods currently used to enhance depth map, then we satisfactorily proposed a new method dubbed RGHS for enhancing a widerange of shallow-water pictures. Our proposed procedure, which considers both the spectral allocation feature of the picture as well as the aquatic transfer properties ofnumerous light streams, begins with a juxtaposition adjustments based on a simple histogram stretch with vibrant specifications acquirement in the Rgb model. After that, adjustments are made to the hue, saturation, lightness, and brightness according to the CIE-Lab colors space.

We compared our proposed method to the standard techniques for dehazing, which rely on distribution bending in the RGB & HSI/HSV colour models. Both quantitative & qualitative analysis shows how much better our method is at bringing out features & minimising noise in photographs captured in intertidal zone. The incorporation of gaussian dispersion into the RGB or CIELab colour is useful for studies involving other types of image data. However, our proposed technique has shortcomings whenever it refers to boosting submerged photographs since it does not account for the significant damage canal in high groundwater imagesthat ignores the energy retardation along the line of dispersion between scene as well as the top. As a matter of fact, deep sea images obtained at depths of more than a thousand metres are not used to test many of the methods meant to improve underwater photography.

In summary, we establish that proposed method serves the goal of contrast correction. The main problem with this technique, however, is that it reduces the amount of colour pixels. is used in Rayleigh stretching to adjust colour and contrast. On the other hand, the saturation of the red hue is far too high. As a result, the photo may contain shades of red. Consequently, RGHS is the method, used for both color calibration & contrasting restoration by eliminating the R component, to get over this problem. As a result, there will be no over-saturation of the Red channel. We conclude that relative global histogram stretching is preferable than histogram equalization.

REFERENCES

[1] Zhang, Weidong, et al. "Underwater image enhancement via minimal color loss and locally adaptive contrast enhancement." IEEE Transactions on Image Processing 31 (2022): 3997-4010.

[2] Liu, Risheng, et al. "Twin adversarial contrastive learning for underwater image enhancement and beyond." IEEE Transactions on Image Processing 31 (2022): 4922-4936.

[3] Zhou, Jingchun, Dehuan Zhang, and Weishi Zhang. "Underwater image enhancement method via multi-feature 8p5rior fusion." Applied Intelligence (2022): 1-23.

[4] Ding, Xueyan, et al. "A unified total variation method for underwater image enhancement." Knowledge-Based Systems 255 (2022): 109751.

[5] Jiang, Qun, et al. "Two-step domain adaptation for underwater image enhancement." Pattern Recognition 122 (2022): 108324.

[6] Zhuang, Peixian, et al. "Underwater Image EnhancementWith Hyper-Laplacian Reflectance Priors." IEEE Transactions on Image Processing 31 (2022): 5442-5455.

[7] Sun, Kaichuan, Fei Meng, and Yubo Tian. "Progressive multi-branch embedding fusion network for underwater image enhancement." Journal of Visual Communication and Image Representation 87 (2022): 103587.

[8] Kang, Yaozu, et al. "A Perception-Aware Decomposition and Fusion Framework for Underwater Image Enhancement." IEEE Transactions on Circuits and Systems for Video Technology (2022).

[9] Sharma, Shobha, and Tarun Varma. "Graph signal processing based underwater image enhancement techniques." Engineering Science and Technology, an International Journal 32 (2022): 101059.

underwater image processing and enhancement

ORIGINALITY REPORT

1%	0%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Visvesvaraya Technological University Student Paper	1%

Exclude matches	Off	Exclude bibliography	On
Exclude quotes	Off		