CCAI-322

Image Processing

AIL

Group members:

Hanouf Albarakaty

[redacted]

Saja alzahrani

[redacted]

Razan Tawfiq

[redacted]

Problem description:

Underwater images often suffer from poor visibility, low contrast, and color distortion due to factors like water turbidity and absorption. These challenges can hinder various underwater activities such as marine research, search and rescue operations, and recreational diving. To address this issue, we propose an image processing project aimed to enhance underwater images to improve their visual quality and utility.

Goals:

1: Improve visibility: Enhancing underwater images aims to increase the visibility of objects and details that may be obscured or distorted due to factors such as scattering, absorption, and low light conditions underwater.

2: Enhance color and contrast: Underwater images often suffer from poor color rendition and low contrast. The goal is to enhance the color accuracy and increase the contrast to improve the overall visual quality of the images.

3: Reduce noise and artifacts: Underwater images can be affected by various types of noise and artifacts, including graininess, blur, and distortion. Enhancement techniques aim to reduce these unwanted elements and improve the overall clarity and sharpness of the images.

4: Restore natural appearance: The goal is to enhance underwater images in a way that restores their natural appearance as much as possible. This includes recovering natural colors, reducing the effects of haze and turbidity, and improving the overall visual fidelity.

5: Object detection: Identifying and localizing objects of interest within an image using ROI precisely to enable machines to understand and interact with visual data accurately and efficiently across various domains.

Process identification: Utilize enhancement techniques like filtering, color correction, contrast adjustment, noise reduction, object detection.

Methodology:

1: Pre-processing: Clean and preprocess images to standardize quality.

2: Visibility Enhancement: Experiment with dehazing algorithms to improve visibility.

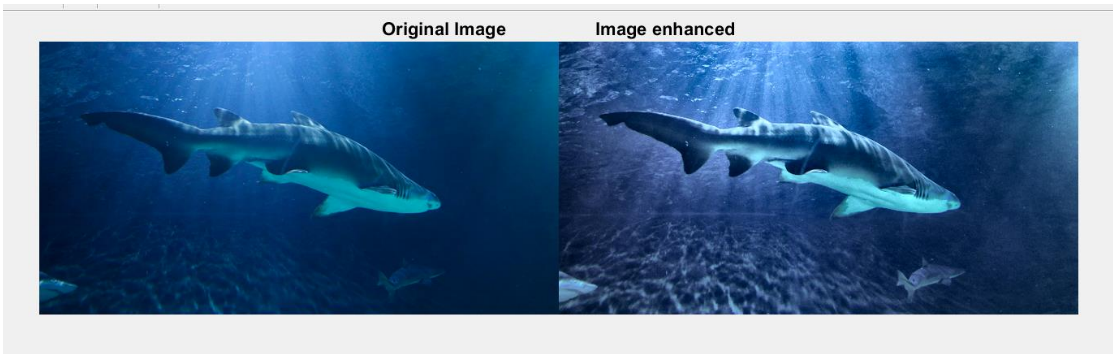
3: Color and Contrast Enhancement: Apply color correction and contrast adjustment techniques.

4: Noise Reduction: Implement filtering methods (e.g., median, Gaussian) to reduce noise.

5: Object Detection: Utilize machine learning models or segmentation algorithms for object identification.

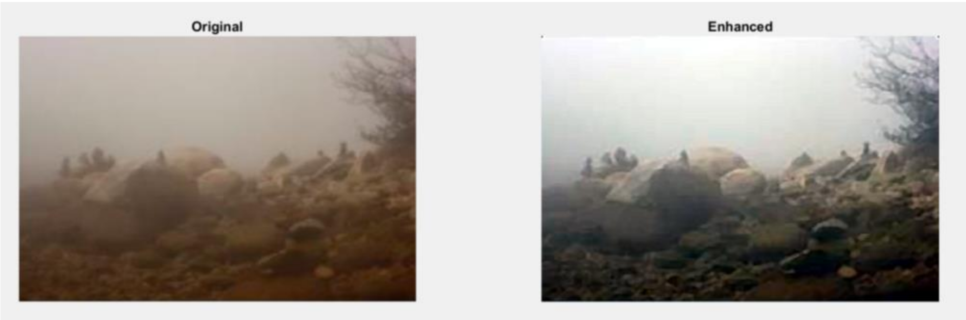
Findings:

```
shadow = imread("underwater4.1.png");
shadow_lab = rgb2lab(shadow);
max_luminosity = 100;
L = shadow_lab(:,:,1)/max_luminosity;


shadow_adapthisteq = shadow_lab;
shadow_adapthisteq(:,:,1) = adapthisteq(L)*max_luminosity;
shadow_adapthisteq = lab2rgb(shadow_adapthisteq);
figure
montage({shadow,shadow_adapthisteq},"Size",[1 2])
title("Original Image                    Image enhanced")
```



The original image details weren't clear, and the lightness of the image was kind of poor, so we improved the visibility and details of the image, especially in regions with low contrast or uneven lighting conditions. We enhanced its contrast using adaptive histogram equalization on the lightness channel of the Lab color space.

```
image = imread('Screenshot (1031).png');

% Convert the image to double precision for processing
image = im2double(image);

% Color correction using histogram stretching
enhanced = image;
for i = 1:3
    enhanced(:,:,i) = imadjust(image(:,:,i));
end

% Denoise the enhanced image using a median filter
denoised = enhanced;
for i = 1:3
    denoised(:,:,i) = medfilt2(enhanced(:,:,i));
end

% Sharpen the denoised image with enhanced edges
amount = 2.5; % Adjust the amount to control the strength of sharpening
radius = 1; % Adjust the radius to control the size of the neighborhood
sharpened = imsharpen(denoised, 'Amount', amount, 'Radius', radius);

% Display the original and enhanced images
figure;
subplot(1, 2, 1), imshow(image), title('Original');
subplot(1, 2, 2), imshow(sharpened), title('Enhanced');
```



The original image needed color correction and noise reduction. We performed color correction using histogram stretching on each color channel, and then denoises the enhanced image using a median filter. Our goal was to improve the image's color balance and reduce noise for further image processing or visualization purposes.

```python
from PIL import Image, ImageEnhance, ImageFilter
import matplotlib.pyplot as plt

# Load the original image
original_image = Image.open("/kaggle/input/firstblurredim/Screenshot (1020).png")

# Load the image
image = original_image.copy()

# Get the width and height of the image
width, height = image.size

# Define the crop region
crop_left = 0  # Crop from the left side
crop_top = 0
crop_right = width // 2
crop_bottom = height

# Crop the image
cropped_image = image.crop((crop_left, crop_top, crop_right, crop_bottom))
```

```python
# Enhance contrast
factor = 1.5
enhancer = ImageEnhance.Contrast(cropped_image)
enhanced_contrast = enhancer.enhance(factor)

# Apply sharpening with increased effect
sharpened_image = enhanced_contrast.filter(ImageFilter.SHARPEN)

# Increase the sharpening effect subtly
sharpened_image = sharpened_image.filter(ImageFilter.UnsharpMask(radius=1, percent=

# Create a new figure and axes
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Display the original image
axes[0].imshow(original_image)
axes[0].axis('off')
axes[0].set_title('Original Image')

# Display the cropped image without modification
axes[1].imshow(cropped_image)
axes[1].axis('off')
```

```python
axes[2].set_title('Sharpened Image')

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```

The original image was suffering from low contrast, but we were interested in the right side of the image.

1: Defining the coordination of cropping the image, we cropped the image using the crop() method of the PIL Image object. It creates a new image cropped_image by specifying the coordinates of the crop region.

2: Enhanced the contrast of the cropped image: by using the ImageEnhance module of PIL

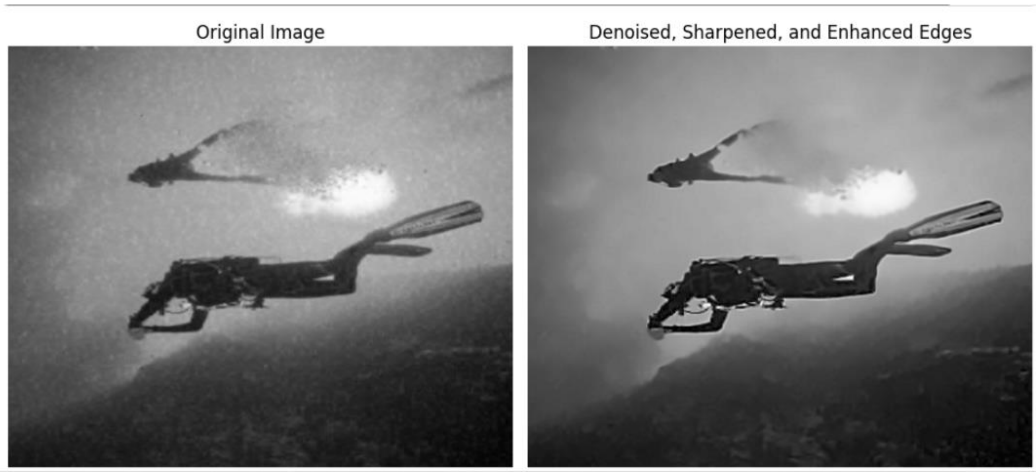
3: Applied the sharpening filter by using filter() method of the PIL Image object. The sharpening filter is used here for enhancing the edges and details in the image. Also, we applied another sharpening effect to the sharpened_image using the ImageFilter.UnsharpMask filter from PIL. The UnsharpMask filter is a specific type of sharpening filter that increases the apparent sharpness of an image.

Overall, we cropped, enhanced the contrast, applied sharpening filters on the image. Now the image is useful for visually comparing the effects of various image processing operations on the input image.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread('/kaggle/input/noisy2/NoiseImage3.jpg')

# Split the image into color channels
b, g, r = cv2.split(image)

# Apply Gaussian blur to each color channel
blurred_b = cv2.GaussianBlur(b, (0, 0), 3)
blurred_g = cv2.GaussianBlur(g, (0, 0), 3)
blurred_r = cv2.GaussianBlur(r, (0, 0), 3)

# Calculate the sharpened image by subtracting the blurred image from the original
sharpened_b = cv2.addWeighted(b, 2.0, blurred_b, -1.0, 0)
sharpened_g = cv2.addWeighted(g, 2.0, blurred_g, -1.0, 0)
sharpened_r = cv2.addWeighted(r, 2.0, blurred_r, -1.0, 0)

# Merge the sharpened color channels back into a single image
sharpened_image = cv2.merge((sharpened_b, sharpened_g, sharpened_r))
```

```python
# Apply non-local means denoising to the sharpened image
denoised = cv2.fastNlMeansDenoisingColored(sharpened_image, None, 10, 10, 7, 21)

# Enhance edges using Laplacian operator
laplacian = cv2.Laplacian(denoised, cv2.CV_64F)
enhanced_edges = np.clip(denoised - 0.3 * laplacian, 0, 255).astype(np.uint8)

#---------------------------------------------------------------
# Create a figure with two subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Display the original image in the first subplot
ax1.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
ax1.set_title('Original Image')
ax1.axis('off')

# Display the denoised and sharpened image with enhanced edges in the second subpl
ax2.imshow(cv2.cvtColor(enhanced_edges, cv2.COLOR_BGR2RGB))
ax2.set_title('Denoised, Sharpened, and Enhanced Edges')
ax2.axis('off')

# Adjust the spacing between subplots
```

```python
# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```



The original image has noise, and its details weren't clear. We performed several image processing operations on the image.

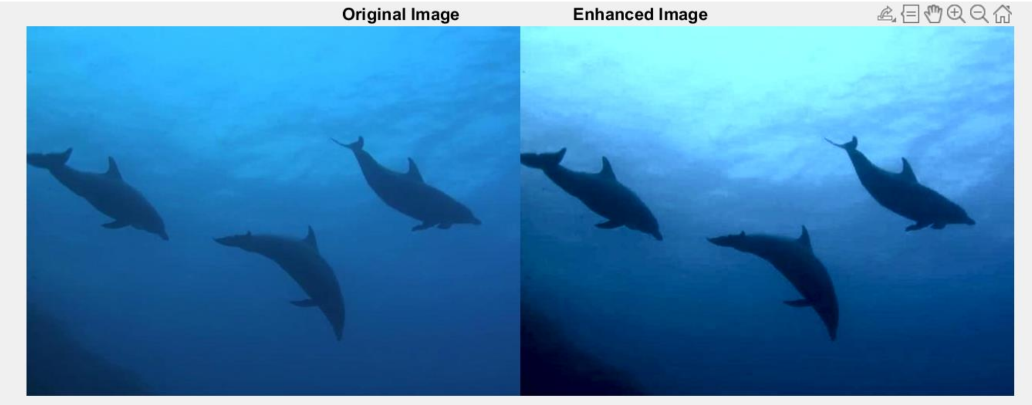
1: We applied Gaussian blur to all three channels green, blue and red using the GaussianBlur() function from OpenCV.

2: Sharpening: We calculated the sharpened blue channel by subtracting the blurred blue channel from the original channel and then multiplying the result by 2.0. We used the addWeighted() function from OpenCV for this calculation.

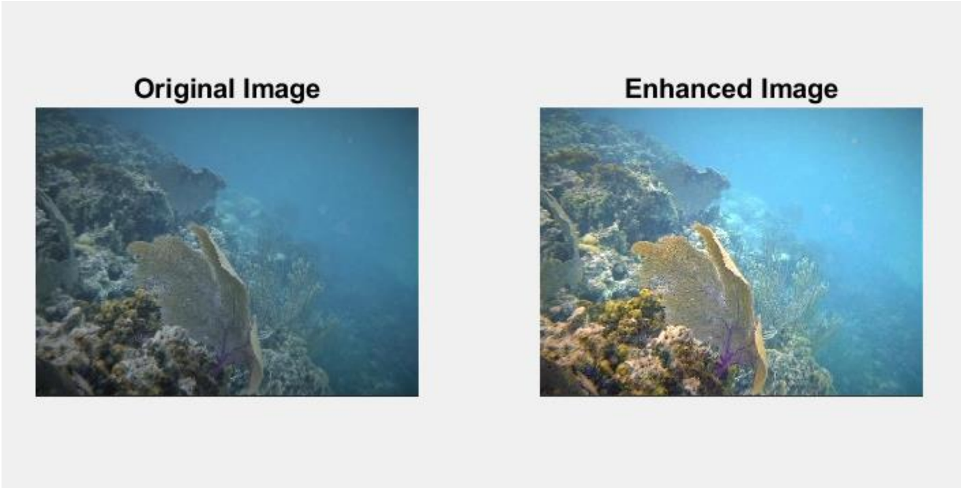
3: Denoise: we applied the Laplacian operator to the denoised image using the Laplacian() function from OpenCV. The Laplacian operator detects edges in an image.

4: Edge enhancement: we enhanced the edges in the denoised image by subtracting a fraction (0.3) of the Laplacian image from the denoised image.

```
shadow = imread("underwater3.png");
shadow_lab = rgb2lab(shadow);
max_luminosity = 100;
L = shadow_lab(:,:,1)/max_luminosity;
shadow_imadjust = shadow_lab;
shadow_imadjust(:,:,1) = imadjust(L)*max_luminosity;
shadow_imadjust = lab2rgb(shadow_imadjust);

figure
montage({shadow,shadow_imadjust},"Size",[1 2])
title("  Original Image               "+"        Enhanced Image")
```

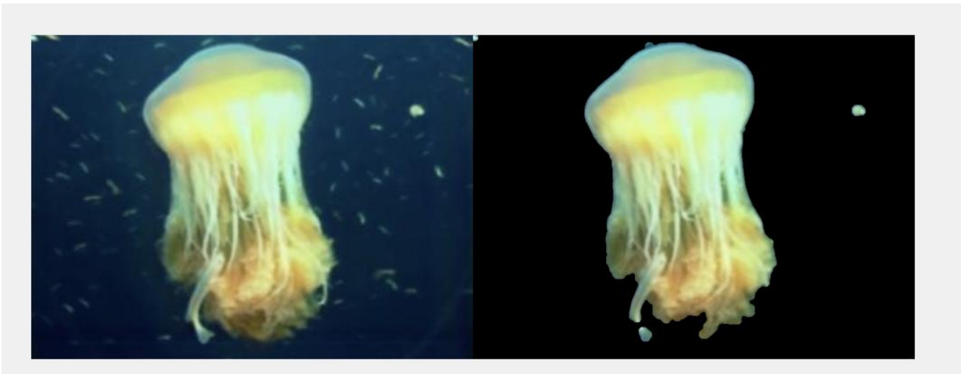


The original image was suffering from low contrast and visibility, so we performed image enhancement on the image. At first, we converted the RGB image to the LAB color space, performed contrast adjustment on the L channel. Then converted it back to the RGB color space. We used the contrast adjustment to enhance the visibility and details in the image by stretching the intensity values of the L channel.




```
img=imread("IMG_3271.jpg")
I=imreducehaze(img)
I=imlocalbrighten(I,0.1)
subplot(1, 2, 1), imshow(img), title('Original Image');
subplot(1, 2, 2), imshow(I), title('Enhanced Image');
```

The original image suffered from obscurity due to dense fog, resulting in reduced visibility and muted colors. To address this challenge, we applied “imreducehaze “to effectively eliminate the underwater haze, accompanied by a brightness increase for enhanced clarity. The result is a vastly improved image with clearer colors, successfully overcoming the challenges posed by the unique atmospheric conditions prevalent beneath the water's surface.

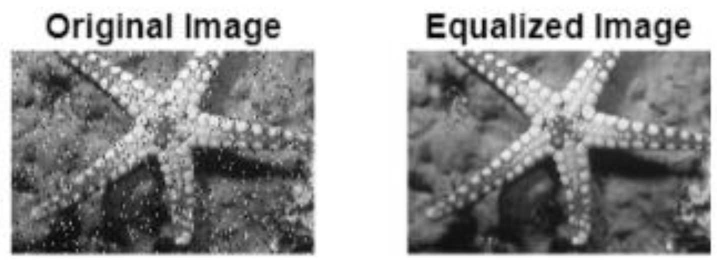


```
img2=imread("IMG_3348.jpg")
img3=imreducehaze(img2)
imwrite(img3,"imgse.jpg")
[BW,maskedImage]=segmentImage(img2)
montage({img2,maskedImage})
function [BW,maskedImage] = segmentImage(RGB)
```

```
% Close mask with default
radius = 3;
decomposition = 0;
se = strel('disk', radius, decomposition);
BW = imclose(BW, se);

% Open mask with default
radius = 3;
decomposition = 0;
se = strel('disk', radius, decomposition);
BW = imopen(BW, se);

% Open mask with default
radius = 9;
decomposition = 4;
se = strel('disk', radius, decomposition);
BW = imopen(BW, se);
```

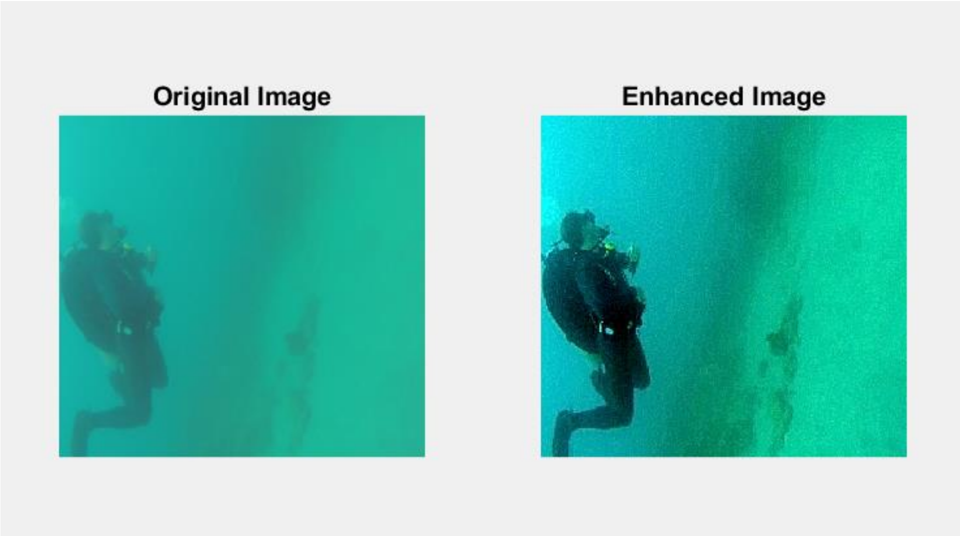
The original image contains noticeable noise, affecting image quality. Our approach involved the application of” imreducehaze” to mitigate the noise, followed by thresholding and morphological operations for further refinement. The processed image demonstrates a significant reduction in noise, providing a cleaner and visually more appealing outcome compared to the original.

This approach is beneficial for scenarios where noise reduction is crucial. The sequential application of techniques ensures a comprehensive improvement in image quality.



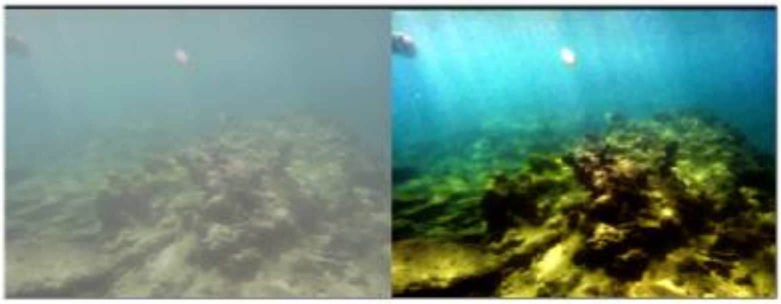

```
img=imread("/MATLAB Drive/IMG_3719.PNG")
img1=medfilt2(img)
subplot(1, 2, 1), imshow(img), title('Original Image')
subplot(1, 2, 2), imshow(img1), title('Equalized Image')
```

The original image contains salt and pepper noise, degrading overall image quality. To address this, a median filter was applied, effectively removing the noise and resulting in a cleaner, smoother image. This approach showcases the tailored effectiveness of median filtering in mitigating specific types of noise.




```
img1=imread("nm_80up.jpg");
img2=imreducehaze(img1);
img2=imsharpen(img2,"Amount",4)
subplot(1, 2, 1), imshow(img1), title('Original Image');
subplot(1, 2, 2), imshow(img2), title('Enhanced Image');
```

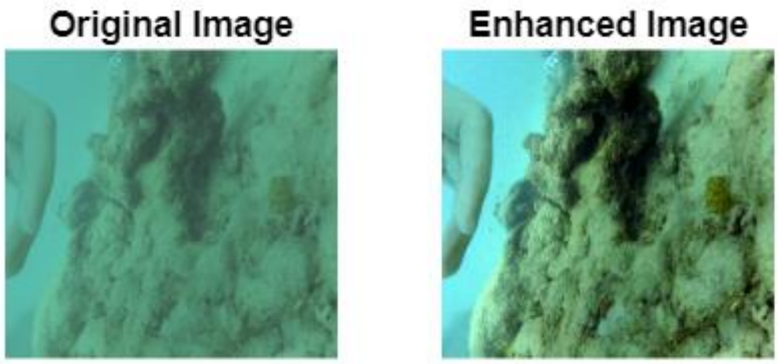
The original image is unclear due to fog, and the details are lacking. Through the use of imreducehaze to eliminate fog and subsequent image sharpening, the processed image achieved enhanced clarity, improved colors, and sharper edges. This combined approach successfully addressed both challenges posed by the unique atmospheric conditions prevalent beneath the water's surface. and detail enhancement.



```
img = imread("IMG_3275.jpg");
fAverage = fspecial("average",[ 5 , 5
imgAveraged = imfilter(img, fAverage);
I=imreducehaze(imgAveraged)
I=histeq(I);
red=I(:,:,1)
green=I(:,:,2)
blue=I(:,:,3)
gamma=1.7
redT=uint8(255*(double(red)/255) .^gamma)
greenT=uint8(255*(double(green)/255) .^gamma)
blueT=uint8(255*(double(blue)/255) .^gamma)
I2=cat(3,redT,greenT,blueT)
montage({img, I2})
```

The original image lacked clarity and exhibited faded colors. Employing a spatial filter (average) for initial enhancement, followed by separating the histogram into RGB components and applying Power-Law (Gamma) Transformation, resulted in a visually appealing image with improved clarity, vibrant colors, and enhanced contrast.

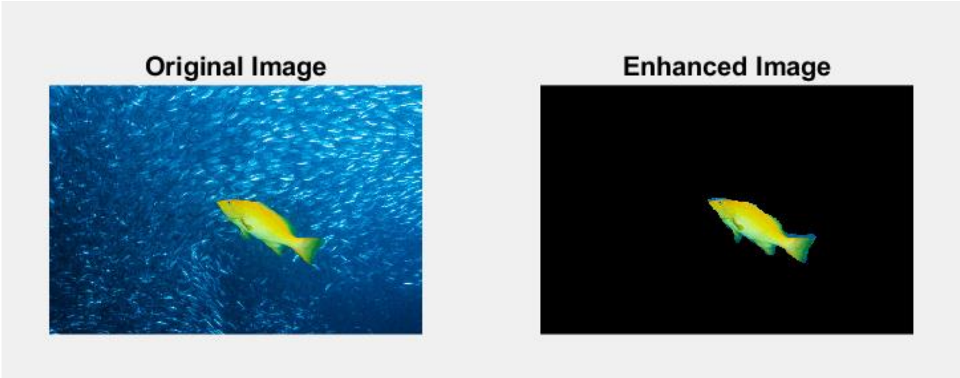
This multi-step approach is beneficial for enhancing overall image quality by addressing both spatial and color-related challenges.



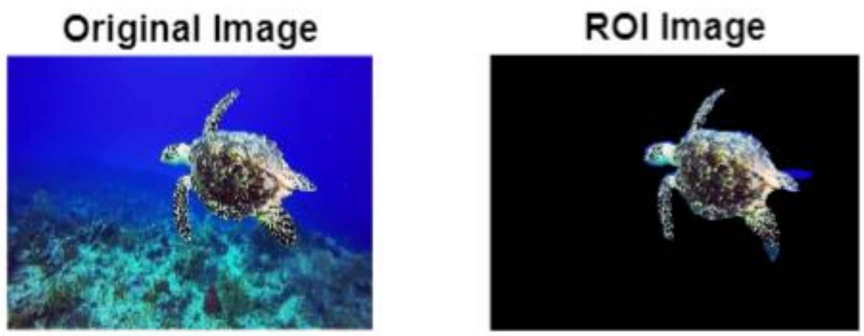

```
img=imread("IMG_3347.jpg")
imshow(img)
imhist(img)
img1=imreducehaze(img)
subplot(1, 2, 1), imshow(img), title('Original Image');
subplot(1, 2, 2), imshow(img1), title('Enhanced Image');
```

The original image is unclear due to fog and exhibits low contrast. Our approach involved using “imreducehaze” to eliminate fog and applying Histogram Equalization to enhance contrast. The result is a visually improved image with clearer colors, successfully addressing challenges related to fog and low contrast.

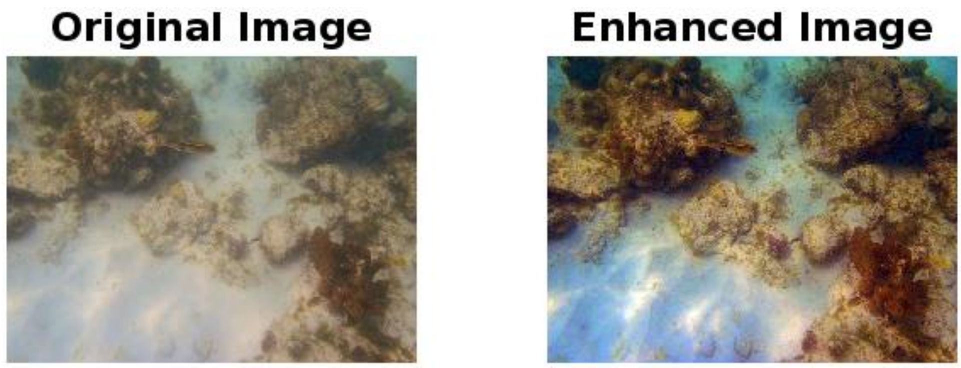
This approach is particularly effective when dealing with foggy conditions and low-contrast scenarios, providing a comprehensive solution for image enhancement.




```
img=imread("img18.jpg")
[BW,maskedImage]= segmentImage(img)
subplot(1, 2, 1), imshow(img), title('Original Image');
subplot(1, 2, 2), imshow(maskedImage), title('Enhanced Image');

function [BW,maskedImage] = segmentImage(RGB)
```



Utilized Region of Interest (ROI) to detect fish and turtle in both images. The processed images highlight the identified regions containing fish or turtle, showcasing the successful application of ROI for automating the identification of specific objects within the images. This technique streamlines the analysis process and enhances efficiency in object detection.

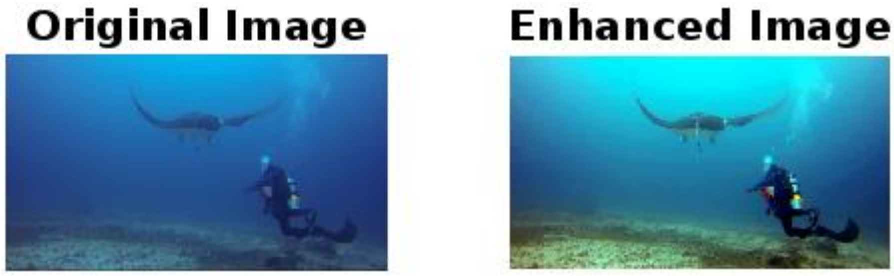



```
img1=imread("Img1.jpg");
img2=imreducehaze(img1);
subplot(1, 2, 1), imshow(img1), title('Original Image');
subplot(1, 2, 2), imshow(img2), title('Enhanced Image');
```

The original image has a low contrast, so we used the imreducehaze function which is designed to reduce haze or atmospheric scattering in an image. The result is an improved image with clearer, more vibrant colors.

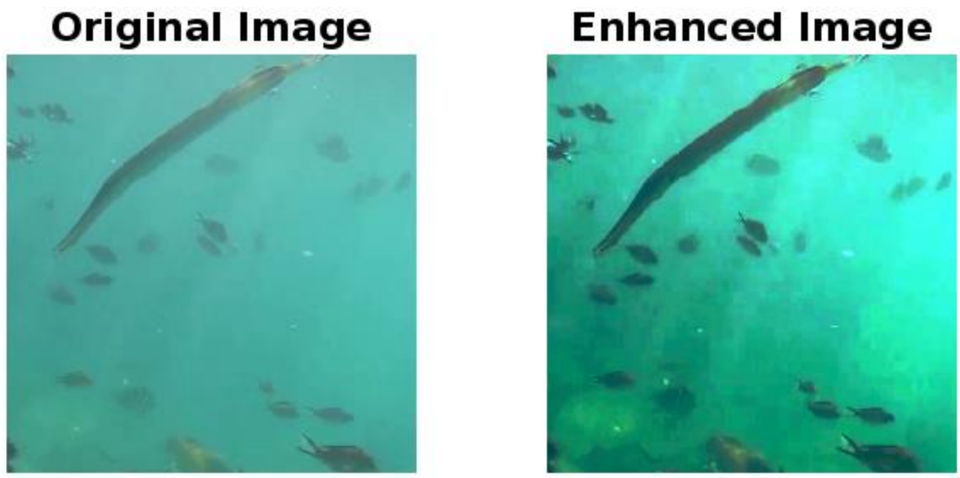

```
img=imread("Img7.jpg")
I=imreducehaze(img)
I=imlocalbrighten(I,0.1)
subplot(1, 2, 1), imshow(img), title('Original Image');
subplot(1, 2, 2), imshow(I), title('Enhanced Image');
```

The problem with the original image is its reduced visibility and degraded quality. So, we used the imreducehaze function to enhance the visibility and improve the overall quality of the hazy image. In addition to that, we brightened the image a little more.




```
img=imread("Img3.jpg")
I=imreducehaze(img)
I=imlocalbrighten(I,0.1)
subplot(1, 2, 1), imshow(img), title('Original Image');
subplot(1, 2, 2), imshow(I), title('Enhanced Image');
```

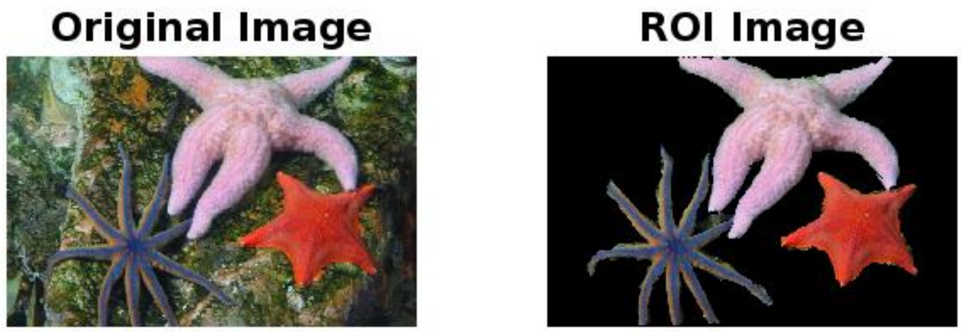
We encountered an issue with the original image, which had reduced visibility and degraded quality. To rectify this, we employed the imreducehaze function, which improved visibility and enhanced the overall quality of the hazy image. Moreover, we further enhanced the image by applying a slight brightness adjustment.



img21-done.m   img5_e.m

```
I = imread('Img5.jpg');

J = histeq(I);

imshowpair(I,J,'montage')
axis off

subplot(1, 2, 1), imshow(I), title('Original Image');
subplot(1, 2, 2), imshow(J), title('Enhanced Image');
```

The original image is foggy and unsharp and the objects in the background are not distinguishable, so we fixed it using perform histogram equalization which is a technique that redistributes the pixel intensities of an image to enhance its contrast and improve the overall appearance.




```
originalImage=imread("img21.png");
[BW,maskedImage] = segmentImage(originalImage);
subplot(1, 2, 1), imshow(originalImage), title('Original Image');
subplot(1, 2, 2), imshow(maskedImage), title('ROI Image');
function [BW,maskedImage] = segmentImage(RGB)
%segmentImage Segment image using auto-generated code from Image Segmenter app
%  [BW,MASKEDIMAGE] = segmentImage(RGB) segments image RGB using
%  auto-generated code from the Image Segmenter app. The final segmentation
%  is returned in BW, and a masked image is returned in MASKEDIMAGE.

% Auto-generated by imageSegmenter app on 03-Dec-2023
%----------------------------------------------------


% Convert RGB image into L*a*b* color space.
X = rgb2lab(RGB);

% Create empty mask
BW = false(size(X,1),size(X,2));

% Draw ROIs

xPos = [458.8784 460.3532 461.0545 462.0000 463.0000 464.0000 465.0000 466.0000 467.0000 467.9455 468.7013 469.2442 469.7558 470.2987 471.0545 472.0000 4
yPos = [359.9231 359.6468 359.0000 358.4077 358.3532 358.6468 358.5923 357.9455 357.0000 356.0000 355.0000 354.0000 353.0000 352.0545 351.2987 350.7013 3
m = size(BW, 1);
n = size(BW, 2);
```

img21-done.m

```
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;

xPos = [209.7973 211.0000 211.0545 211.2987 211.7558 212.2442 212.7558 213.1897 213.4026 213.2987 213.4077 214.0000 214.7013 215.1897 215.4571 215.5429 2
yPos = [334.5385 336.3532 337.0545 338.0000 339.0000 340.0000 341.0000 342.0000 343.0000 344.0000 345.0000 346.0000 347.0000 348.0000 349.0000 350.0000 3
m = size(BW, 1);
n = size(BW, 2);
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;

xPos = [295.1306 296.3532 297.0545 298.0000 299.0000 300.0000 301.0000 302.0000 303.0000 304.0000 305.0000 306.0000 307.0000 308.0000 309.0000 310.0000 3
yPos = [5.6923 6.0545 6.3532 7.0545 8.0000 8.8910 9.4026 9.5429 9.7558 10.0000 10.2442 10.4571 10.4884 10.5116 10.5429 10.8103 11.2987 11.9455 12.4026 12
m = size(BW, 1);
n = size(BW, 2);
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;

% Draw ROIs

xPos = [205.1847 206.3532 207.0545 208.0000 209.0000 210.0000 210.9455 211.7013 212.2987 213.0545 214.0000 215.0000 216.0000 217.0000 218.0000 219.0000 2
yPos = [3.3846 4.2987 4.7013 5.0000 5.3532 6.0545 7.0000 8.0000 9.0000 10.0000 10.9455 11.7013 12.2987 13.0545 14.0000 15.0000 15.9455 16.6468 16.8910 16
m = size(BW, 1);
n = size(BW, 2);
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;
```

```
n = size(BW, 2);
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;

% Draw ROIs

xPos = [514.2297 515.2987 515.7013 516.0000 516.3532 516.9455 517.4026 517.5974 518.0545 518.7013 519.2987 520.0545 520.9455 521.7013 522.2987 523.0545 5
yPos = [184.5385 185.3532 186.0545 187.0000 188.0000 189.0000 190.0000 190.9455 191.7013 192.2987 193.0545 194.0000 195.0000 196.0000 197.0000 198.0000 1
m = size(BW, 1);
n = size(BW, 2);
addedRegion = poly2mask(xPos, yPos, m, n);
BW = BW | addedRegion;

% Create masked image.
maskedImage = RGB;
maskedImage(repmat(~BW,[1 1 3])) = 0;
end
```

To narrow our focus and exclude irrelevant areas, we defined a region of interest (ROI). This allowed us to concentrate on a specific region instead of the entire image. In this case, we highlighted the starfish that appeared in the image.

Conclusion: Our project aims to enhance underwater images by improving visibility, enhancing color and contrast, reducing noise and artifacts, and restoring the natural appearance. We utilize techniques such as dehazing, color correction, contrast adjustment, noise reduction, and object detection using ROI. Successfully implementing this project would significantly improve the visual quality and utility of underwater images, benefiting various applications including marine research, search and rescue operations, and recreational diving. By addressing challenges such as poor visibility, low contrast, and color distortion, we aimed to provide a comprehensive solution for enhancing underwater imagery.