IMAGE PROCESSING REPORT 2

Image Enhancement uses Retinex Algorithms

1. Ghuwalewala Karan Rakesh(22BLC1201)
2. Piyush Arora(22BLC1217)
3. Ayush Arora(22BLC1218)
4. Ayush Negi(22BLC1259) (**Group coordinator**)
5. Adarsh Singh(22BLC1372)

FUNCTIONS -

1. get_ksize(sigma): This function calculates the kernel size based on the sigma value. The formula used is int(((sigma - 0.8)/0.15) + 2.0)

2. get_gaussian_blur(image, ksize=0, sigma=5): This function applies a Gaussian blur to the input image. If the kernel size is not provided, it calls the get_ksize(sigma) function to calculate it. It then creates a Gaussian kernel using OpenCV's getGaussianKernel(ksize, sigma) function and applies it to the image using OpenCV's filter2D(image, -1, np.outer(sep_k, sep_k))function.

3. ssr(image, sigma): This function performs single scale retinex on the input image. It calculates the logarithm of the input image and subtracts the logarithm of the blurred image (obtained from get_gaussian_blur(image, ksize=0, sigma=sigma)) from it.

4. msr(image, sigma_scales=[15, 80, 250]): This function performs multi-scale retinex on the input image by averaging the results of single scale retinex for different sigma values. It then normalizes the result to a range of 0 to 255(histogram equalization).

5. colour_balance(image, low_per, high_per): This function adjusts the colour balance of the input image by scaling the histogram of each colour channel such that most of the pixel intensities lie within a specified percentile range.

6. msrcr(image, sigma_scales=[15, 80, 250], alpha=125, beta=46, G=192, b=-30, low_per=1, high_per=1): This function performs multi-scale retinex with colour restoration on the input image by applying a gain-controlled amplification to the multi-scale retinex output.

7. msrcp(image, sigma_scales=[15, 80, 250], low_per=1, high_per=1): This function performs multi-scale retinex with colour preservation on the input image by applying a gain-controlled amplification to each colour channel of the multi-scale retinex output.

STEPS -

1. Load and display the input image: The algorithm reads an image from a file and displays it using OpenCV’s imshow function.

2. Resize: The algorithm resizes the image to a fraction of it’s size using resize function of opencv library.

3. Shape: The algorithm uses image.shape which returns a tuple of the number of rows, columns, and channels (if the image is colour).

4. Apply SSR: The algorithm applies single scale retinex algorithm to the input image which uses logarithmic domain to reduce computational complexity.

5. Apply MSR: The algorithm extends ssr to an array of numbers and applies ssr to all.

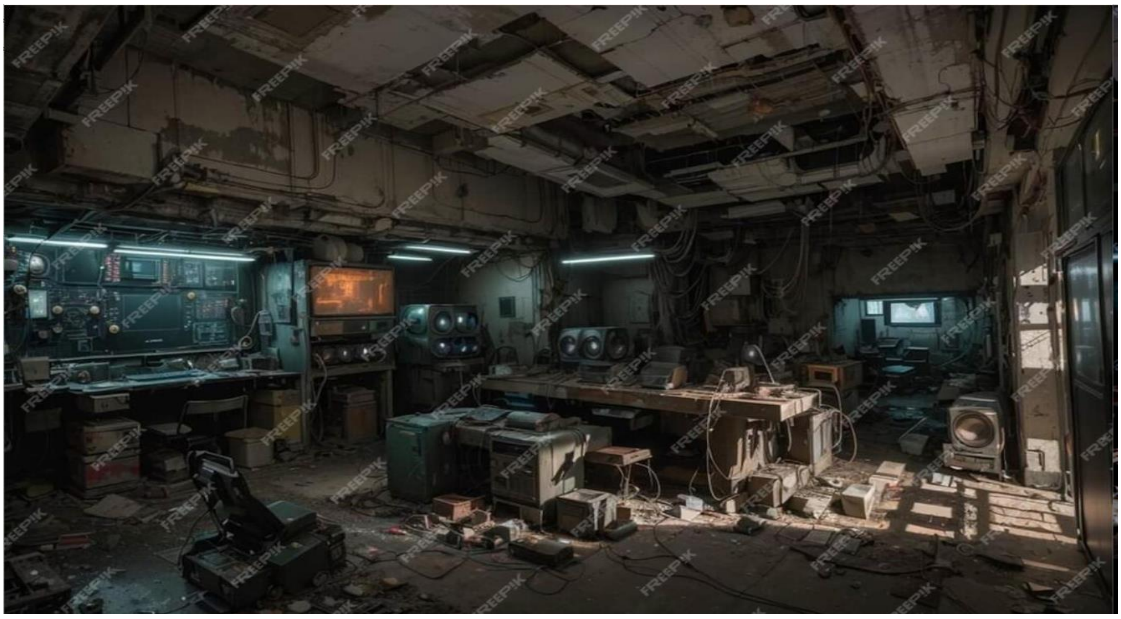
6. Apply MSRCR: The algorithm applies multi-scale retinex with colour restoration (MSRCR) to the input image.

7. Apply MSRCR with Colour Preservation (MSRCP): The algorithm applies multi-scale retinex with colour preservation (MSRCP) to the input image.

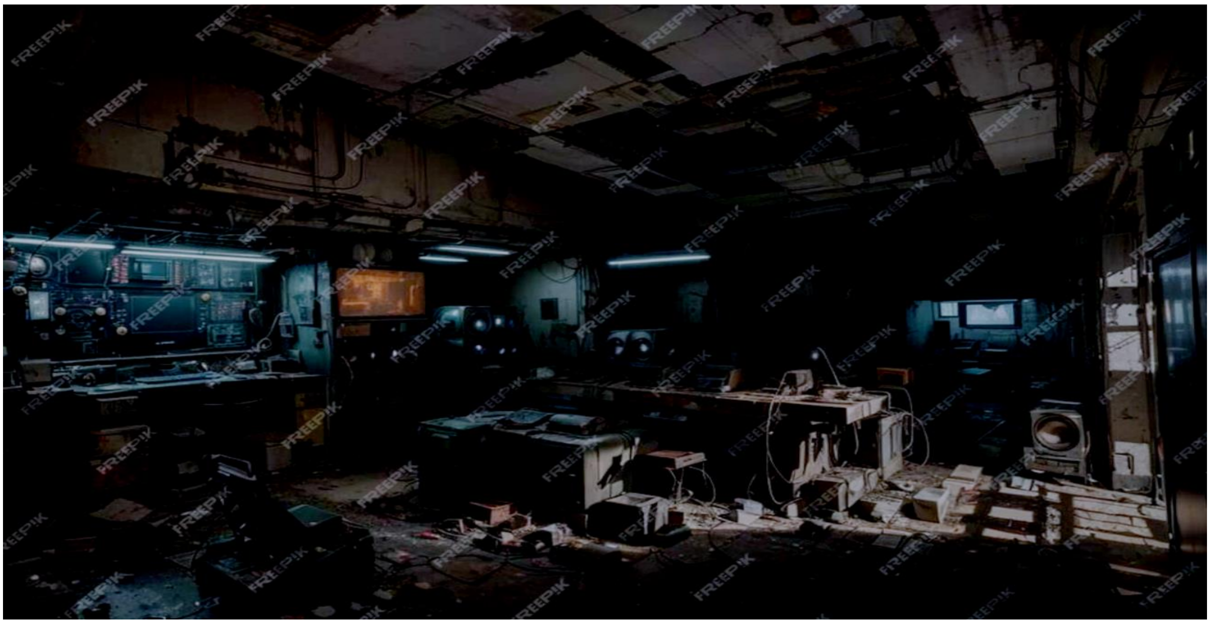
8. Resize image to half, bigger and interpolate: The algorithm displays the msrcp image, resizes it to half of it’s size and tries to create an approximate copy of it using interpolation.

9. Display the output image: The algorithm displays the output image using OpenCV’s imshow function.

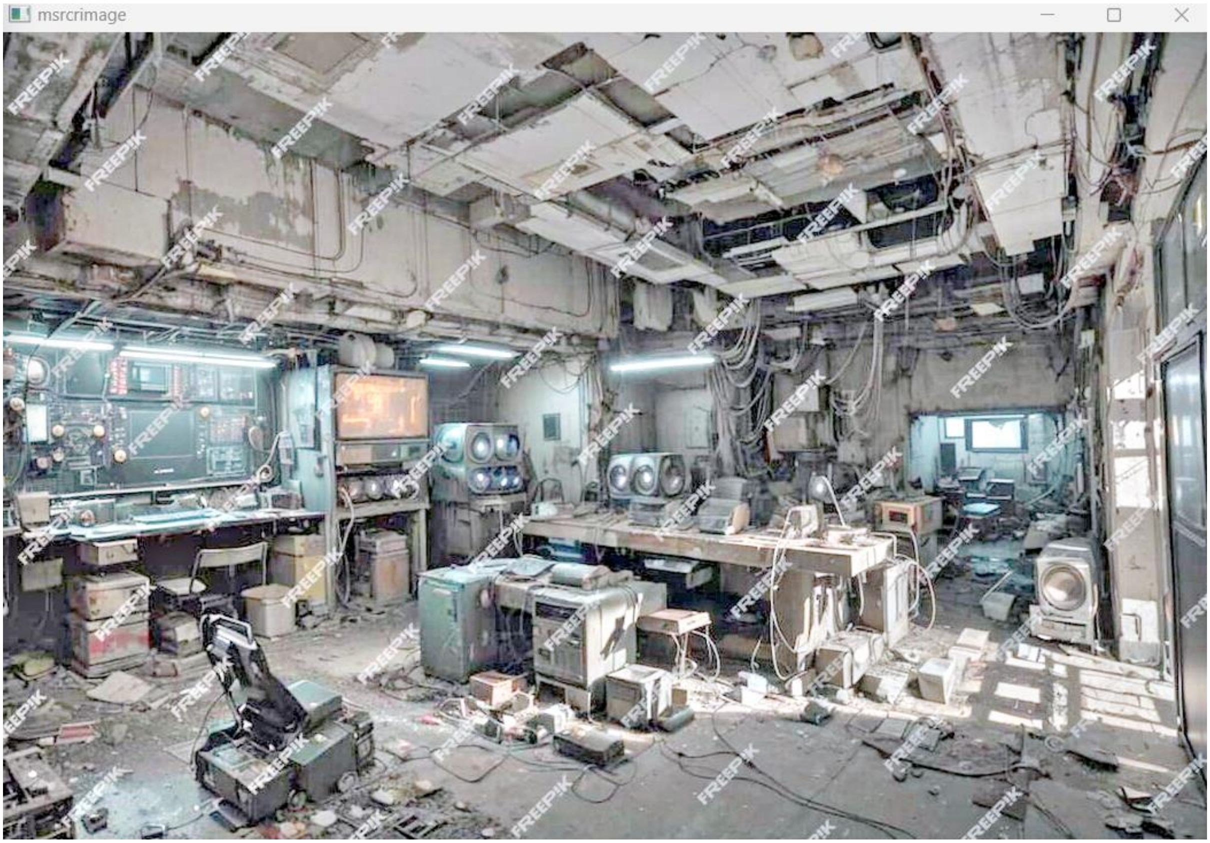
10. Wait for user interaction and close windows: The algorithm waits for a key press event and then closes all OpenCV windows.

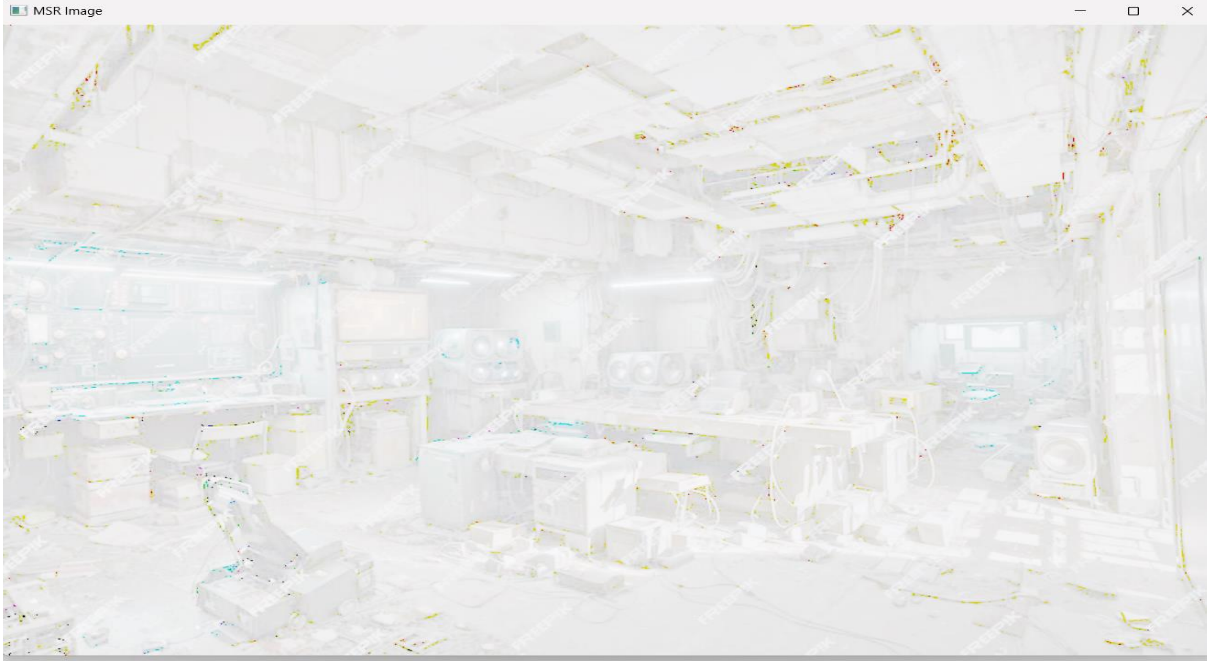


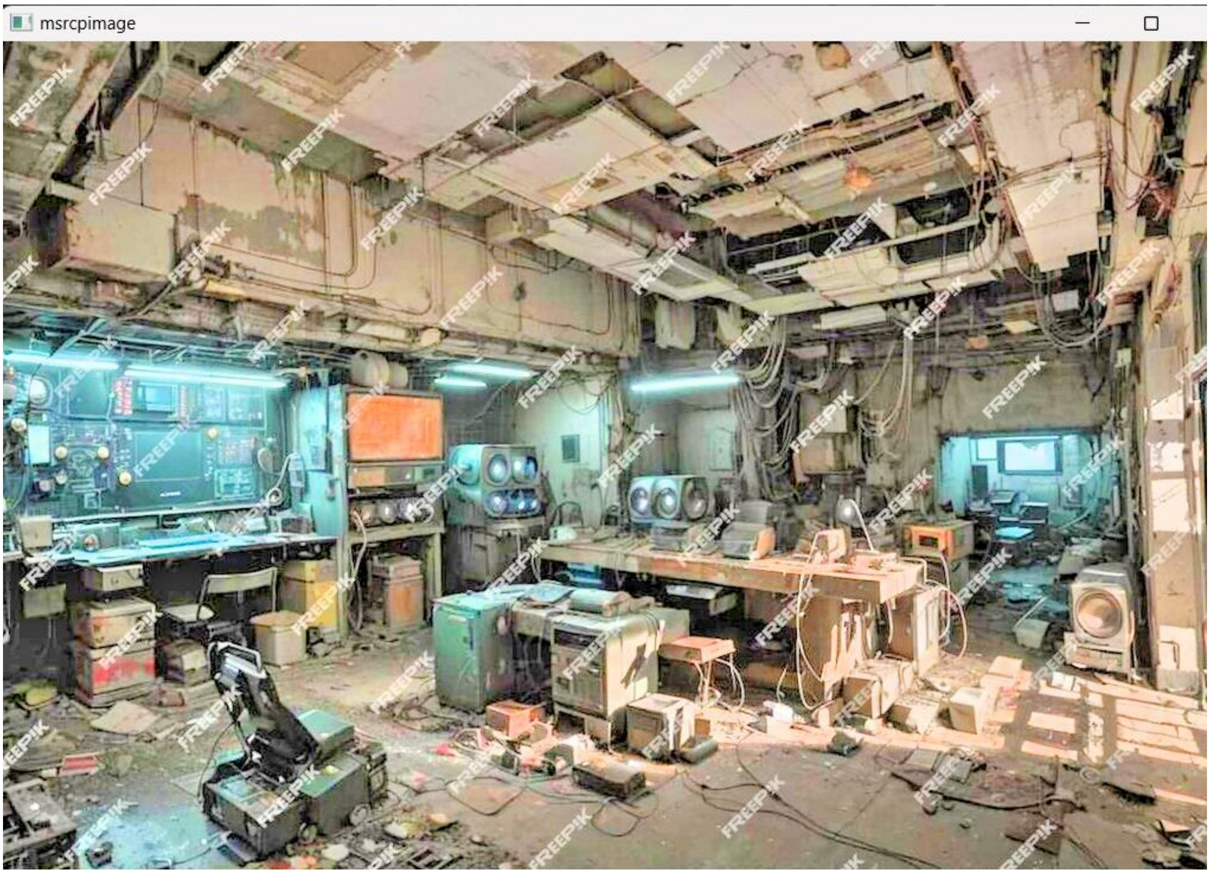
INPUT IMAGE

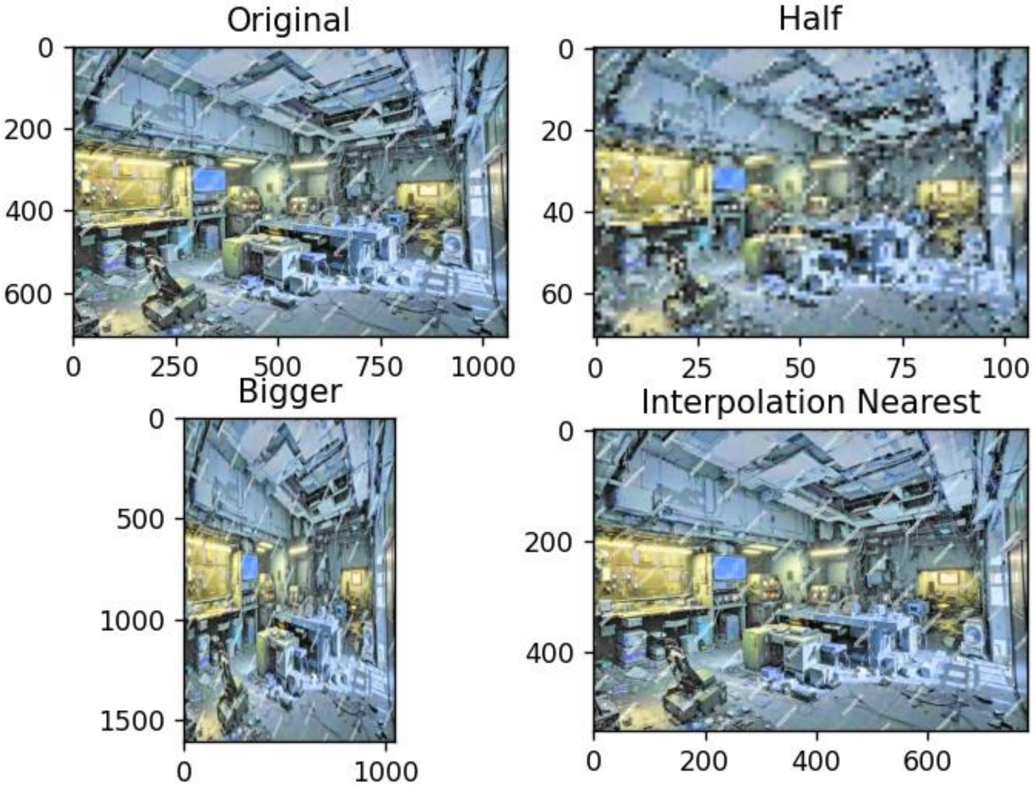


SSR IMAGE


msrcrimage
FREEPIK


MSR Image


msrcpimage
FREEPIK


Original
0
200
400
600
0
250
500
750
1000
Half
0
20
40
60
0
25
50
75
100
Bigger
0
500
1000
1500
0
1000
Interpolation Nearest
0
200
400
0
200
400
600