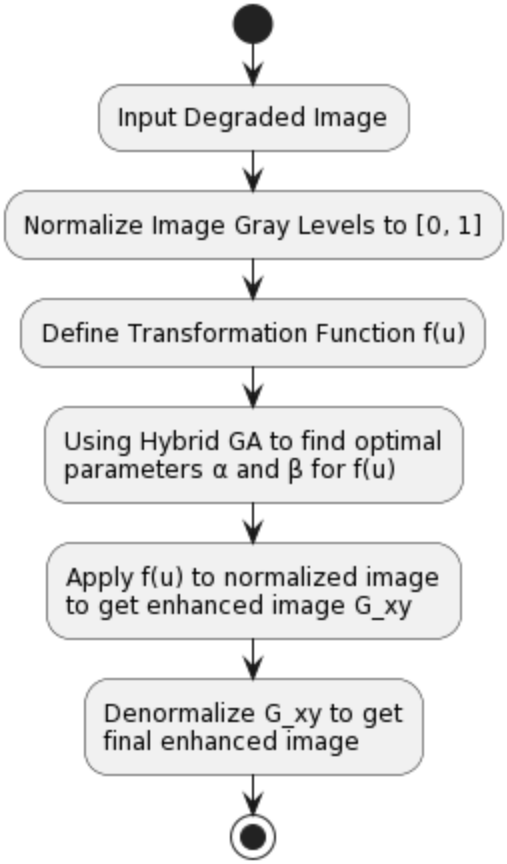
Name	Adwait Purao
UID no.	2021300101

Experiment 6	
AIM :	Image Enhancement using recently published techniques based on any one of the following operations. 1. Enhancement using Point Operation 2. Enhancement using Histogram Processing
OBJECTIVE:	● To Investigate Existing Histogram Processing Techniques ● To Analyze Limitations of Current Image Enhancement Techniques ● To Implement and Evaluate the Proposed Methodology ● To Quantitatively Assess Image Enhancement Performance
INTRODUCTION:	In image enhancement, Tubbs proposed a normalized incomplete Beta function to represent several kinds of commonly used non-linear transform functions to do the research on image enhancement. But how to define the coefficients of the Beta function is still a problem. We proposed a Hybrid Genetic Algorithm which combines the Differential Evolution to the Genetic Algorithm in the image enhancement process and utilize the quickly searching ability of the algorithm to carry out the adaptive mutation and searches. Finally we use the Simulation experiment to prove the effectiveness of the method.

BLOCK DIAGRAM:



IMPLEMENTATION:

```
import numpy as np
import scipy
from skimage import io, color, img_as_float
from skimage.transform import resize
from scipy.optimize import differential_evolution
from skimage.metrics import structural_similarity as ssim
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import peak_signal_noise_ratio
import matplotlib.pyplot as plt

def beta_function(u, alpha, beta):
    """
    Normalized incomplete Beta function for image enhancement.
    """
    x = u.copy()
    out = np.zeros_like(x)
    mask = (x >= 0) & (x <= 1)  # Create a mask to handle
values outside [0, 1]
```

```
    x = x[mask]  # Apply the mask
    out[mask] = x ** alpha * (1 - x) ** beta /
scipy.special.beta(alpha, beta)
    return out

def enhance_image(image, bounds):
    """
    Enhance the image using the Hybrid Genetic Algorithm.
    """
    def objective_func(params):
        alpha, beta = params
        enhanced = beta_function(image, alpha, beta)
        # Specify the data range for SSIM
        return -np.mean(ssim(image, enhanced,
data_range=image.max() - image.min()))

    result = differential_evolution(objective_func, bounds,
maxiter=600, popsize=30, disp=False, workers=1)
    alpha, beta = result.x
    enhanced = beta_function(image, alpha, beta)
    return enhanced

def safe_psnr(img1, img2, data_range=None):
    """
    Compute the peak signal-to-noise ratio (PSNR) between two
images, avoiding division by zero.
    """
    if data_range is None:
        data_range = np.max(img1) - np.min(img1)
    err = np.mean((img1 - img2) ** 2)
    return 10 * np.log10((data_range ** 2) / (err + 1e-12))  #
Add a small constant to avoid division by zero

def main():
    # Load the image
    original_image =
io.imread('C:\\Users\\aspur\\OneDrive\\FOSIP\\EXPERIMENTS\\06.
Image Enhancement using point processing\\input_image.png')

    # Check if image has four channels (RGBA)
    if original_image.shape[2] == 4:
        # Use only RGB channels
```

```
        original_image = original_image[:, :, :3]

    # Convert to grayscale
    original_image_gray = color.rgb2gray(original_image)

    # Enhance the image
    bounds = [(1, 20), (1, 20)]
    enhanced = enhance_image(original_image_gray, bounds)

    # Convert enhanced image mode to a supported mode for PNG
    enhanced = color.gray2rgb(enhanced)

    # Ensure images have the same dimensions
    if original_image.shape != enhanced.shape:
        # If dimensions don't match, pad or crop the enhanced
image
        if original_image.shape[0] < enhanced.shape[0]:
            enhanced = enhanced[:original_image.shape[0],
:original_image.shape[1], :]
        elif original_image.shape[0] > enhanced.shape[0]:
            pad_width = ((0, original_image.shape[0] -
enhanced.shape[0]), (0, original_image.shape[1] -
enhanced.shape[1]), (0, 0))
            enhanced = np.pad(enhanced, pad_width,
mode='constant')

    # Convert the images to the same data type
    original_image = img_as_float(original_image)
    enhanced = img_as_float(enhanced)

    # Check if the original and enhanced images are identical
    if np.array_equal(original_image, enhanced):
        print("Original and enhanced images are identical.")
        original_psnr = 0
        enhanced_psnr = 0
    else:
        # Evaluate the performance
        original_psnr = safe_psnr(original_image,
original_image)
        enhanced_psnr = safe_psnr(original_image, enhanced)

    print(f"Original PSNR: {original_psnr:.2f}")
```

```
    print(f"Enhanced PSNR: {enhanced_psnr:.2f}")

    # Save the enhanced image
    enhanced_uint8 = (enhanced * 255).astype(np.uint8)
    enhanced_uint8 = np.squeeze(enhanced_uint8)  # Remove
single-dimensional entries

io.imsave('C:\\Users\\aspur\\OneDrive\\FOSIP\\EXPERIMENTS\\06.
Image Enhancement using point processing\\enhanced_image.png',
enhanced_uint8)

    # Plot histograms
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.hist(original_image.ravel(), bins=256, color='blue',
alpha=0.7)
    plt.title('Original Image Histogram')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.subplot(1, 2, 2)
    plt.hist(enhanced.ravel(), bins=256, color='red',
alpha=0.7)
    plt.title('Enhanced Image Histogram')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.show()

if __name__ == "__main__":
    main()
```

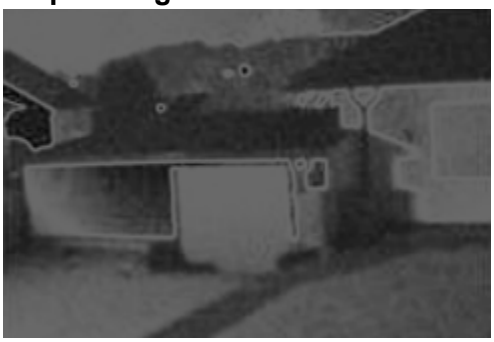
OUTPUT:

Terminal:

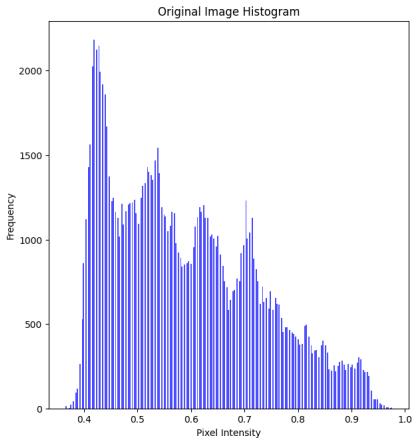
```
aspur@LAPTOP-LG4IQEFB MINGW64 ~/OneDrive/FOSIP/EXPERIMENTS/06. Image Enhancement using poin
t processing
$ python ImgEnhance.py
Original PSNR: 115.73
Enhanced PSNR: 5.64
```

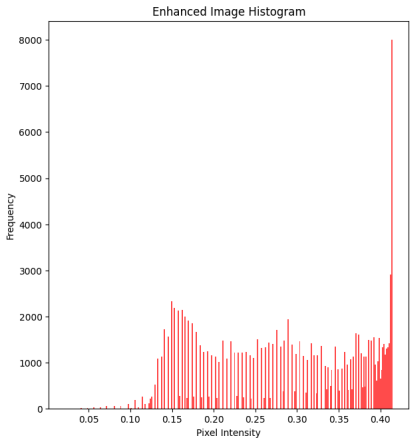
Input Image:

Output Image:



Histogram of Input and Output Image:





REFERENCE:

D. Mu, C. Xu and H. Ge, "Hybrid Genetic Algorithm Based Image Enhancement Technology," 2011 International Conference on Internet Technology and Applications, Wuhan, China, 2011, pp. 1-4, doi: 10.1109/ITAP.2011.6006336. keywords: {Image enhancement;Genetic algorithms;Wavelet transforms;Indexes;Filtering;Image edge detection},

https://ieeexplore.ieee.org/document/6006336

CONCLUSION:

The paper discusses the use of a Hybrid genetic algorithm for image enhancement, focusing on maintaining the integrity of perspective image information. Experimental results demonstrate the effectiveness of this approach, showcasing significant improvements in image quality. Compared to other evolutionary algorithms, the hybrid genetic algorithm stands out for its simplicity, robustness, and rapid convergence towards optimal solutions. It requires only a few parameters to be set, which can be applied across various problems. The algorithm's quick search capability facilitates adaptive mutation and search for optimal parameter values, reducing computational complexity compared to exhaustive methods. Overall, the proposed image enhancement method offers practical value due to its efficiency and effectiveness.