| **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:**    **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. | |