Experiment no: 3

1. **Aim:** TO study various histogram equalization.

2. **Software Tool Used:** MATLAB

3. **Theory:**

Histogram equalization is a widely used image processing technique that improves the contrast of an image. By redistributing the intensity values of pixels, this method makes features in an image more visible, especially in cases where the contrast is low.

Image Histogram: A histogram in the context of an image represents the distribution of pixel intensity values. For a grayscale image, the histogram plots the number of pixels for each intensity level, ranging from 0 (black) to 255 (white) in an 8-bit image.

Low-Contrast Image: In a low-contrast image, the intensity values are clustered around a narrow range, resulting in a histogram that is concentrated in a small area of the intensity scale. This often leads to images that look washed out or too dark.

3. The Process of Histogram Equalization:

Histogram equalization works by spreading out the most frequent intensity values over a wider range. The key idea is to transform the pixel intensities so that the resulting histogram is approximately uniform, meaning that each intensity level is equally likely.

1.Compute the Histogram of the Image: Determine the frequency of each intensity level in the image.

- **Method:**
  - For a grayscale image, scan through all the pixels and count how many times each intensity value (ranging from 0 to 255) occurs.
  - This count is plotted as a histogram where the x-axis represents the intensity levels, and the y-axis represents the number of pixels.

$$\text{Histogram}(r_k) = n_k$$

where (r_k) is an intensity level, and (n_k) is the number of pixels with that intensity.

2. Calculate the Probability Distribution Function (PDF): Find the probability of occurrence of each intensity level.

- **Method:**
  - Divide each histogram value by the total number of pixels in the image.
  - This gives the Probability Distribution Function (PDF) for each intensity level.

$$p(r_k) = \frac{n_k}{N}$$

where N is the total number of pixels in the image.

3. Calculate the Cumulative Distribution Function (CDF): Accumulate the probability values to form the CDF.

- **Method:** For each intensity level, sum up the PDF values from the lowest intensity level to the current one.

$$CDF(r_k) = \sum_{j=0}^{k} p(r_j)$$

4. Normalize the CDF: Scale the CDF so that the new intensity values range from 0 to 255 (for an 8-bit image).

- **Method:**
  - Multiply each CDF value by (L−1), where L is the number of possible intensity levels (256 for an 8-bit image).
  - Round the result to the nearest integer to get the new intensity level.

$$s_k = \text{round}\left[(L-1) \times CDF(r_k)\right]$$

5. Map the Original Intensities to the Equalized Values: Replace the intensity values in the original image with the new intensity values obtained from the normalized CDF.

- **Method:**
  - After normalizing the CDF, create a lookup table where each original intensity r_k is mapped to a new intensity s_k.
  - For each pixel in the original image, find its intensity r_k and replace it with the corresponding new intensity s_k from the lookup table.
  - After all pixels are mapped to their new values, the result is the equalized image with enhanced contrast.

6. Generate the Equalized Histogram: Verify the equalization by generating and comparing the histogram of the processed image.

- **Method:**
  - Plot the histogram of the equalized image.
  - This histogram should ideally be more uniformly distributed compared to the original histogram.

- **Contrast Enhancement:** By redistributing the pixel intensities, histogram equalization enhances the contrast, making features in the image more distinguishable.
- **Global Operation:** This method adjusts the intensity of all pixels in the image uniformly, which is effective for images where the overall contrast needs improvement.

4. Result:

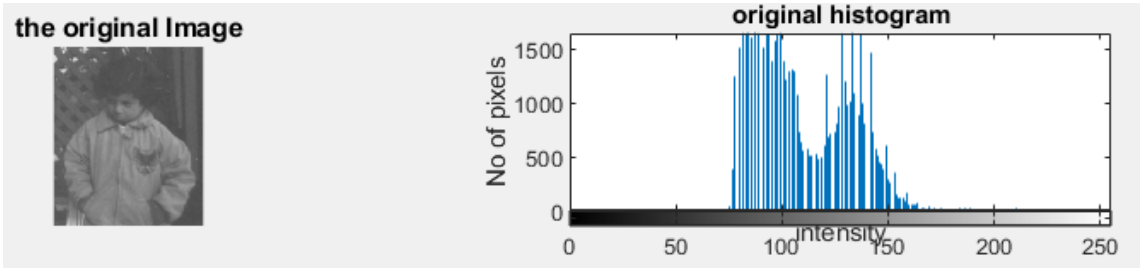


Fig (1): Original Image with its Histogram

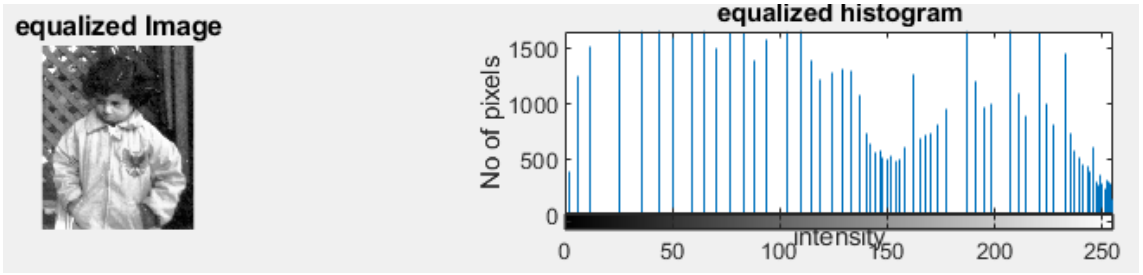
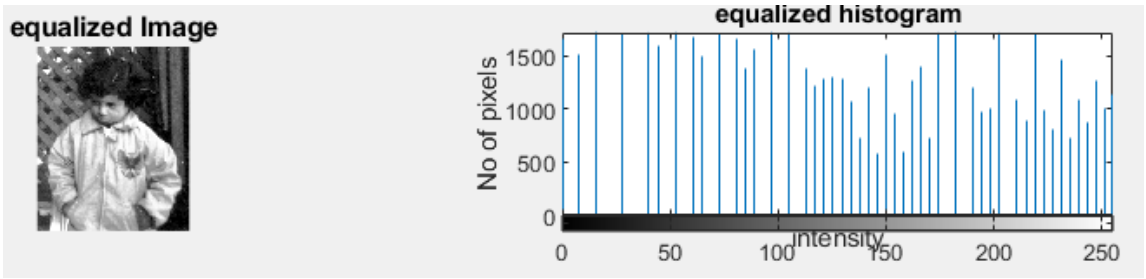
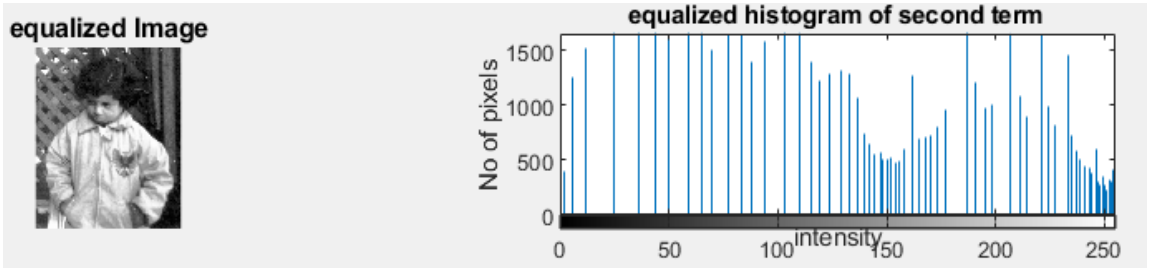


Fig (2): Generated Equalized Image and histogram using manual algorithm



Fig(3) : Generated Equalized Image and histogram using Prebuild function



Fig(4): Generated Equalized Image and histogram using manual algorithm in second term

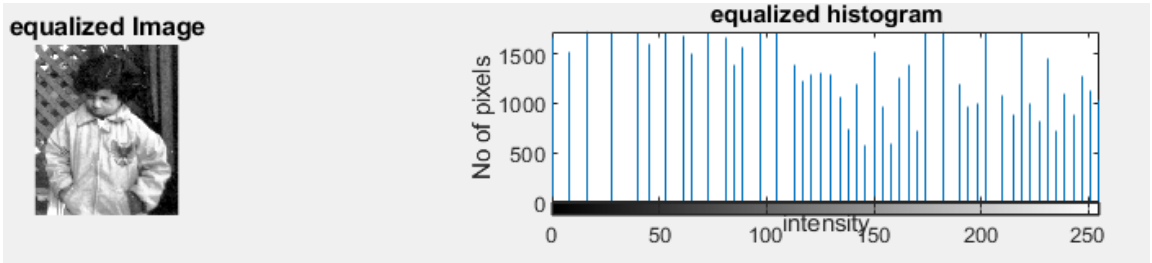


Fig (5): Generated Equalized Image and histogram using manual algorithm in second term

5. Discussion:

we performed histogram equalization on the image `pout.tif` to improve its contrast and visualized the results through a series of subplots. Each step was analyzed to understand the impact of the histogram equalization process on the image and its corresponding histogram.

1. Original Image and Histogram (Subplots 1 & 2(Fig (1)):

- The original image displayed low contrast, where details were not clearly visible. The histogram confirms this by showing a narrow range of pixel intensities clustered around the middle of the intensity scale.
- This limited range of intensity values results in an image that lacks visual depth and differentiation between light and dark regions.

2. First Equalization by design algorithm (Subplots 3 & 4(Fig (2)):

- **Equalized Image:** The first application of histogram equalization significantly improved the contrast of the image. The previously muted details became more pronounced, and the overall image appeared clearer.
- **Equalized Histogram:** The histogram after equalization shows a more uniform distribution across the full intensity range (0-255). This even distribution indicates that the pixel intensities were effectively spread out, which explains the improved contrast in the image.

3. Equalized Image using inbuild function (Subplots 5 & 6(Fig (3)):

- **Built-in Function Equalization:** The image equalized using MATLAB's histeq function shows a contrast improvement like the one achieved with the custom algorithm(fig(2)). The image is clear and well-defined, with enhanced detail visibility.
- **Equalized Histogram Again:** The histogram after using the built-in function is also well-distributed, confirming that the built-in function effectively enhances image contrast in a better manner comparable to the custom algorithm

4. Second-Term Equalization by designed algorithm (Subplots 7 & 8(Fig (4)):

- **Re-Equalized Image with Second Term:** When applying the histogram equalization with the second term, the image shows little visual change compared to the first re-equalized image. The process confirms that additional equalization steps make minimal changes when the histogram is already well-distributed.
- **Second-Term Histogram:** The histogram remains consistent with previous equalized versions, reinforcing that the majority of contrast enhancement occurs during the first equalization step.

5. Second-Term Equalized Image using inbuild function (Subplots 9 & 10(Fig (5)):

- **Final Equalized Image:** The built-in MATLAB `histeq` function was applied to the re-equalized image. The visual result is nearly identical to the earlier equalized images, confirming that the image's contrast has been maximized.

- **Final Histogram:** The final histogram shows that the pixel intensities are well-distributed across the intensity range, indicating that the image has been fully equalized, and further adjustments are unnecessary.

6. Conclusion:

The histogram equalization process effectively enhances the contrast of a low-contrast image by redistributing pixel intensities across the entire intensity range. The results demonstrate that a single application of histogram equalization is typically sufficient to achieve significant contrast enhancement. Subsequent equalizations, whether manually implemented or using MATLAB's built-in function, offer minimal additional benefits, as the histogram becomes uniformly distributed after the first application. This reinforces the efficiency of histogram equalization as a powerful tool for image processing, particularly in enhancing the visibility of image details.

7. Reference:

1.Gonzalez, R.C., & Woods, R.E. (2018). *Digital Image Processing* (4th ed.). Pearson.
2.Pratt, W.K. (2007). *Digital Image Processing: PIKS Scientific Inside* (4th ed.). Wiley-Interscalene .

8. Code:

```
subplot(5,2,1);  r = imread("pout.tif");     imshow(r);      title("the original Image");
subplot(5,2,2);    imhist(r);    xlabel("intensity");     ylabel("No of pixels");
title("original histogram");

[nk, rk] = imhist(r);
pdf = nk / numel(r);
cdf = cumsum(pdf);
sk = cdf * 255 ;
sk = round(sk);
[rows, cols] = size(r);
equi_img = zeros(rows,cols,'uint8');
for i =1:rows
  for j = 1:cols
    equi_img(i,j) = sk(r(i,j)+1);
  end
end
subplot(5,2,3); imshow(equi_img); title("equalized Image");
[nk2, rk2] = imhist(equi_img);
subplot(5,2,4);    imhist(equi_img);   xlabel("intensity");     ylabel("No of pixels");
title("equalized histogram");

subplot(5,2,5);     n = histeq(r);      imshow(n);     title("equalized Image");
subplot(5,2,6);    imhist(n);    xlabel("intensity");    ylabel("No of pixels");
title("equalized histogram");

[nk3, rk3] = imhist(equi_img);
pdf3 = nk3 / numel(equi_img);
cdf3 = cumsum(pdf3);
sk3 = cdf3 * 255 ;
sk3 = round(sk3);
[rows, cols] = size(equi_img);
equi_img2 = zeros(rows,cols,'uint8');
for i =1:rows
  for j = 1:cols
    equi_img2(i,j) = sk3(equi_img(i,j)+1);
  end
end
subplot(5,2,7);        imshow(equi_img2);        title("equalized Image");
[nk4, rk4] = imhist(equi_img2);
subplot(5,2,8); imhist(equi_img2); xlabel("intensity"); ylabel("No of pixels");
title("equalized histogram of second term ");

subplot(5,2,9); n = histeq(equi_img); imshow(n); title("equalized Image");
subplot(5,2,10);     imhist(n); xlabel("intensity");     ylabel("No of pixels");
title("equalized histogram");
```