Lab 3: Restoration of Noisy Images

Kamala Rai Danuwar

13 [20070291]

Bachelors of Engineering in Computer

Everest Engineering College

16/02/2024

Summary

In this practical, we are supposed to restore the corrupted images. We inspected the noise types in the images provided to us visually and applied mean and median filters. I applied both the mean and median filters in the 3 corrupted images and compared between them to see which filter is better in each of the images.

Similarly, also I calculated the Mean Squared Error for each image pair (noisy-noiseless).

I. PROBLEM DESCRIPTION

In this practical we learnt to observe and handle the noisy images and what methods are good for the restoration. In this lab I applied mean and median filter to the three noisy images given to us (‘messi_N.jpg’, ‘ronaldo_N.jpg’, ‘ronaldo_de_lima_N.jpg’) and them compared all the four outcomes of each images i.e. original image, corrupted image, restoration using mean filter and restoration using median filter. Then for each image pair (noisy-noiseless) and each filter (mean and median) calculated the mean square signal-to-noise ratio by asking input from user for the filter window.

II. THEORETICAL BACKGROUND

If an image is corrupted by an additive noise but there is no degradation, then we have

$$g(x,y) = f(x,y) + \eta(x,y)$$

where g(x,y) is the corrupted image, f(x,y) is the original image and $\eta(x,y)$ is the noise.

For the purpose of this lab, we require following definitions. Let S_{xy} denote the pixels in a box of size m*n, centered around (x,y) and $\hat{f}(x,y)$ denote the estimate of the actual image.

$$\text{Mean Filter: } \hat{f}(x,y) = \frac{1}{mn} \sum_{(r,c)\epsilon \mathrm{Sxy}} \{g(r,c)\}$$

$$\text{Median Filter: } \hat{f}(x,y) = \text{median}_{(r,c)\epsilon \mathrm{Sxy}} \{g(r,c)\}$$

The mean-squared signal-to-noise ratio is defined as

$$\mathrm{SNR}_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]^2}$$

The main methods used in this practical are:

- getFilterWindowFromUser() : this method prompts the user to input the filter window used for filtering.
- meanFilter(int[][] f, int filterSize): this method applies mean filter to the input image f using the filtersize taken from the user.
- medianFilter(int[][] f, int filtersize): this method applies median filter to the input mage f using the filtersize taken from the user.
- calculateMSE(int[][] original, int[][] restored): this method calculates the Mean Squared Error(MSE) between two images: the original image and the restored image. It iterates over each pixel in the two images and calculates the squared difference between the corresponding pixels and sums up all the squared differences and divides by the total number of pixels to compute MSE. The MSE is computer using both the mean and median filter.

III. RESULTS AND LESSON LEARNT

To test my problems, I compared the original image, corrupted image and the restored images using both mean and median filter. By this I learnt visually which type of noise is present in the corrupted image and similarly learnt which type of filtering is better for reducing different types of noise present in the corrupted image.

Furthermore, I learnt that the corrupted image can be restored using both the mean and median filter but the amount of noise reduction is visually differentiable. Some kind of filtering is best for one kind of corrupted image but it may not hold true for another corrupted image.

PROGRAM LISTING

```
public class Image {

        public int getFilterWindowFromUser() {
                    Scanner scanner = new Scanner(System.in);
                    System.out.print("Enter the filter window (odd
        integer): ");
                    return scanner.nextInt();
                }

                public static int [][] meanFilter (int [][] r, int
        filterSize) {
                    int [][] R = new int [r.length][r[0].length];

                    for (int x = 0; x<r.length; x++) {
                        for (int y = 0; y<r[0].length; y++) {
                            int sum = 0;
                            int count = 0;

                            //sum of pixels within the filter window
                            for (int i = -filterSize / 2; i <=
        filterSize / 2; i++) {
                                for (int j = -filterSize / 2; j <=
        filterSize / 2; j++) {
                                    int newX = Math.max(0,
        Math.min(r.length - 1, x + i));
                                    int newY = Math.max(0,
        Math.min(r[0].length - 1, y + j));
                                    sum += getPixelValue(newX, newY,
        r);
                                    count++;
                                }
                            }
                            R[x][y] =  sum / count;    //finding mean
                        }
                    }
                    return R;
                }



                private static int getPixelValue(int x, int y, int[][]
        r) {
                    return r[x][y];
        }



        public static int [][] medianFilter (int [][] r, int
filterSize) {
            int [][] R = new int [r.length][r[0].length];

            for (int x = 0; x<r.length; x++) {
               for (int y = 0; y<r[0].length; y++) {

                   //create a neighbourhood array
                   int[] neighbourhood = new int[filterSize *
filterSize];
                   int index = 0;

                   //populate the neighbourhood array with
pixel values
                   for (int i = -filterSize / 2; i <=
filterSize / 2; i++) {
                        for (int j = -filterSize / 2; j <=
filterSize / 2; j++) {
                            int newX = Math.max(0,
Math.min(r.length - 1, x + i));
                            int newY = Math.max(0,
Math.min(r[0].length - 1, y + j));
                            neighbourhood[index++] =
getPixelValue(newX, newY, r);
                        }
                    }



                   //finding median
                   Arrays.sort(neighbourhood);
                   R[x][y] =
neighbourhood[neighbourhood.length / 2];
               }
            }
            return R;
        }
```

```
                //////////////////////////////////mse

                public static double calculateMSE(int[][] original,
        int[][] restored) {
                    if (original.length != restored.length ||
        original[0].length != restored[0].length) {
                     throw new IllegalArgumentException("Image
        dimensions do not match.");
                     }

                     int height = original.length;
                     int width = original[0].length;
                     double sumSquaredError = 0.0;

                     for (int x = 0; x < height; x++) {
                         for (int y = 0; y < width; y++) {
                             int error = original[x][y] -
        restored[x][y];
                             sumSquaredError += error * error;
                         }
                     }

                     return sumSquaredError / (height * width);
                 }




}


public static void main(String[] args) {


     //////////////////////messi

           /*
                    Image messi0 = new Image("C:\\Users\\Kamala Rai
        Danuwar\\Documents\\NetBeansProjects\\IPPR\\messi.jpg");
                    int [][] r0 = messi0.getPixelArray();
                    Image messi1 = new Image("C:\\Users\\Kamala Rai
        Danuwar\\Documents\\NetBeansProjects\\IPPR\\messi_N.jpg");
                    int [][] r1 = messi1.getPixelArray();
```

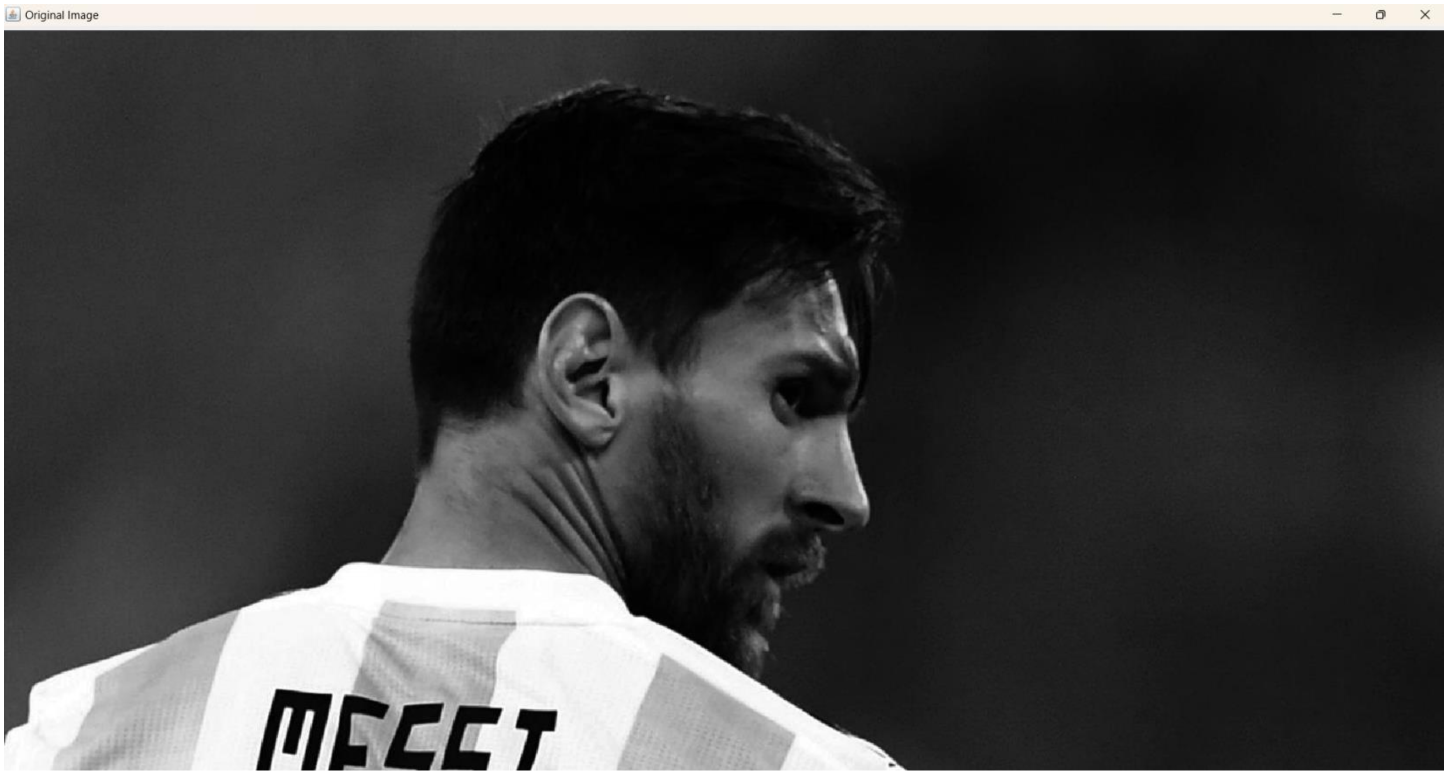
```
            int filterWindow =
    messi1.getFilterWindowFromUser();
            */



      ///////////////////////////Ronaldo


    /*
            Image ronaldo0 = new Image("C:\\Users\\Kamala Rai
    Danuwar\\Documents\\NetBeansProjects\\IPPR\\ronaldo.jpg");
            int[][] r0 = ronaldo0.getPixelArray();
            Image ronaldo1 = new Image("C:\\Users\\Kamala Rai
    Danuwar\\Documents\\NetBeansProjects\\IPPR\\ronaldo_N.jpg");
            int[][] r1 = ronaldo1.getPixelArray();
            int filterWindow =
    ronaldo1.getFilterWindowFromUser();
            */

/////////////////////////////Ronaldo de lima


            /*
            Image ronaldo_de_lima0 = new
    Image("C:\\Users\\Kamala Rai
    Danuwar\\Documents\\NetBeansProjects\\IPPR\\ronaldo_de_lima.jp
    g");
            int[][] r0 = ronaldo_de_lima0.getPixelArray();
            Image ronaldo_de_lima1 = new
    Image("C:\\Users\\Kamala Rai
    Danuwar\\Documents\\NetBeansProjects\\IPPR\\ronaldo_de_lima_N.
    jpg");
            int[][] r1 = ronaldo_de_lima1.getPixelArray();

            int filterWindow =
    ronaldo_de_lima1.getFilterWindowFromUser();
            */
```

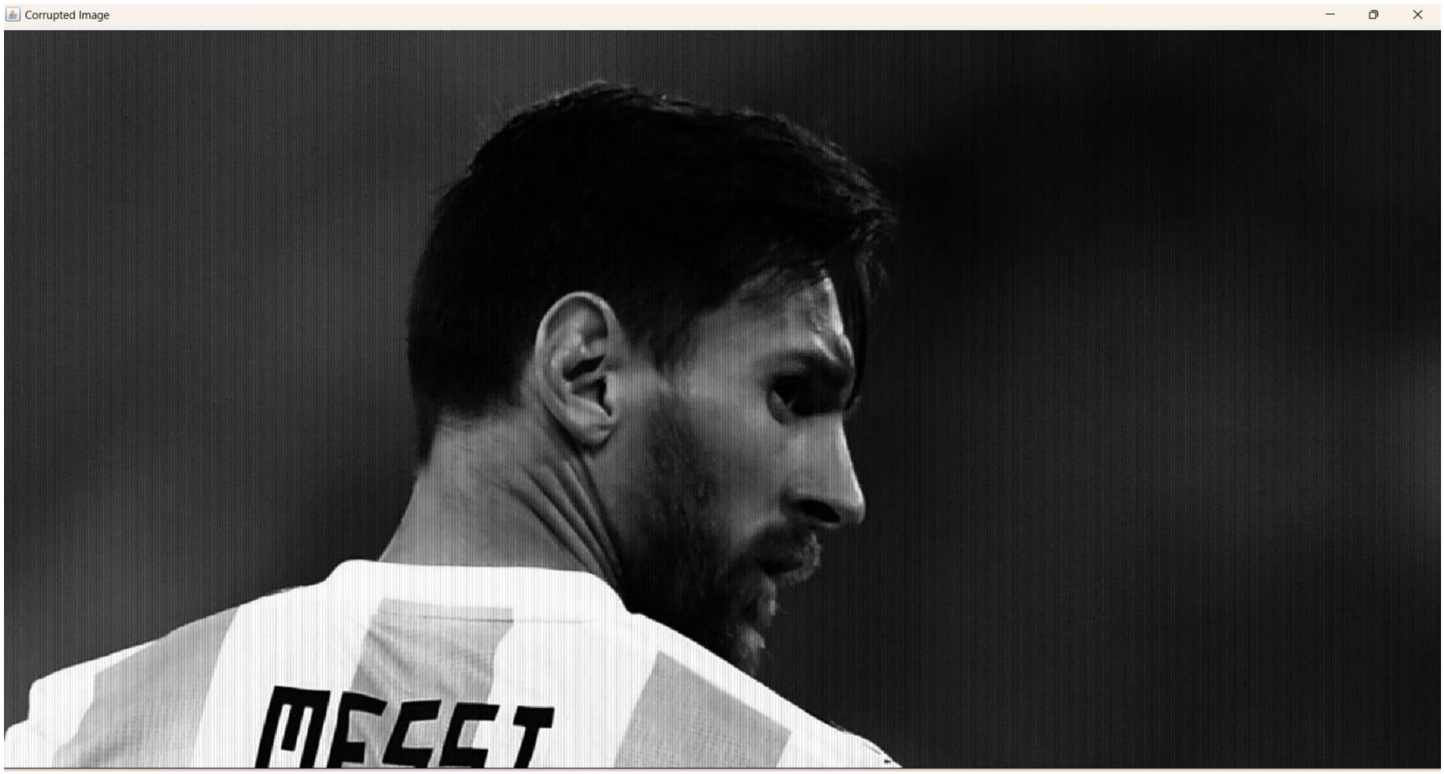
```
   ////////////the below code snippets are same for all three
images
///it is for finding mean square error and also for applying
mean and median filter for all the three images




  /*
           int [][] R0 = meanFilter(r1, filterWindow);

           int [][] R1 = medianFilter (r1,filterWindow);

           Image.display(r0, "Original Image");
           Image.display(r1, "Corrupted Image");
           Image.display(R0, "Restored Image using Mean
Filter");
           Image.display(R1, "Restored Image using Median
Filter");

           //MSE provides quantitative measure of diff betn
original and restored images
           //lower MSE values indicate better image
restoration


           double mseMeanFilter = calculateMSE(r0, R0 );
//passing original image and image after mean filter
           System.out.println("Mean Squared Error (Mean
Filter): " + mseMeanFilter);

           double mseMedianFilter = calculateMSE(r0, R1 );
//passing original image and image after median filter
           System.out.println("Mean Squared Error (Median
Filter): " + mseMedianFilter);

*/
        }
```

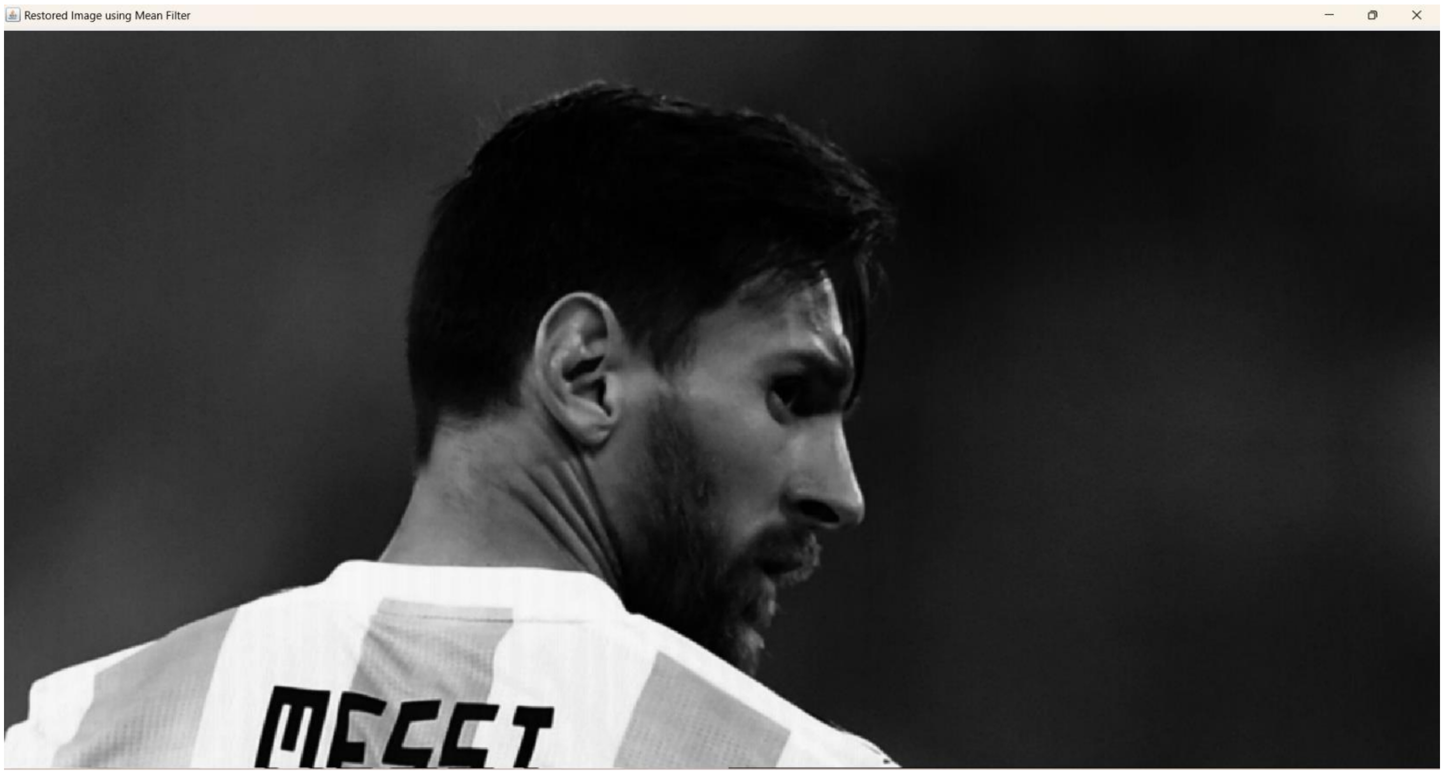
OUTPUTS

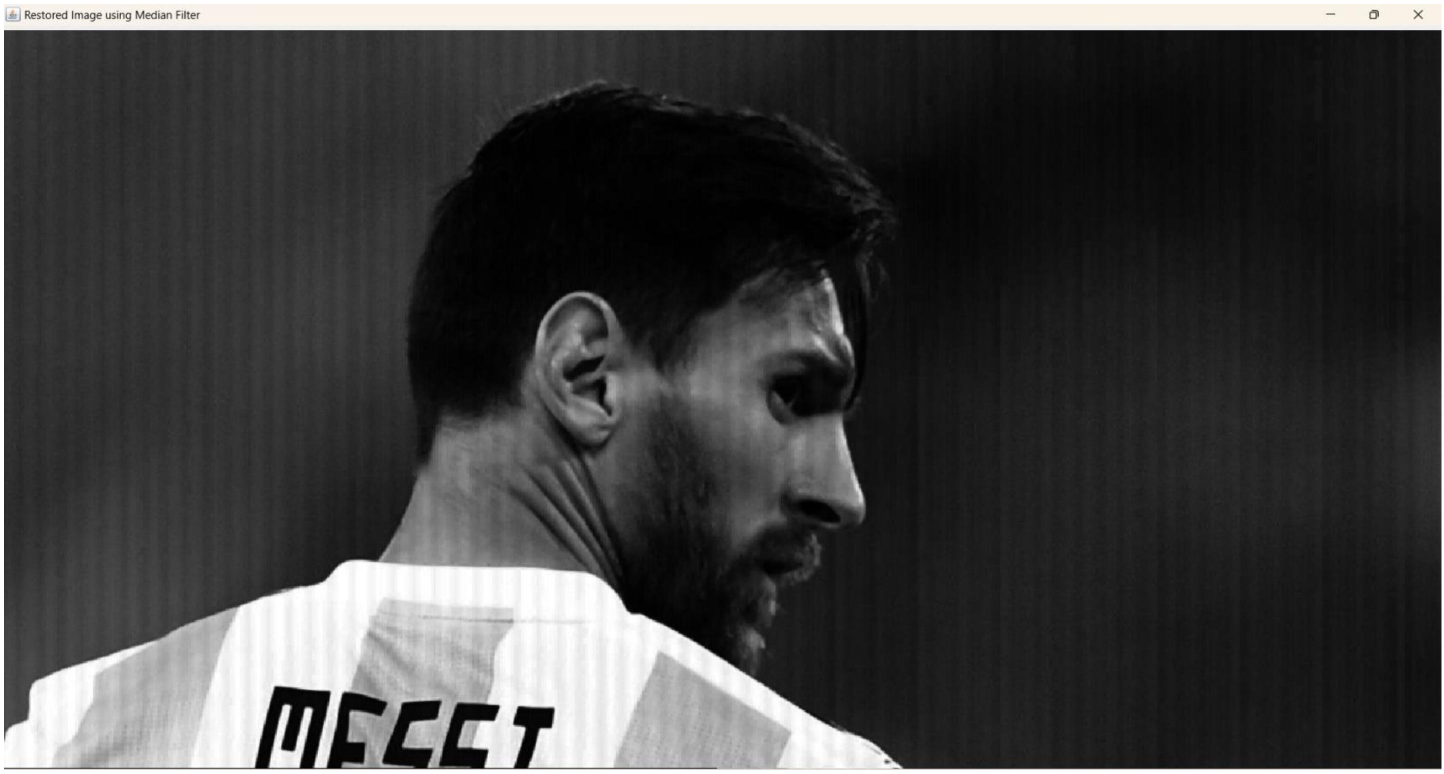
1) For Messi.jpg

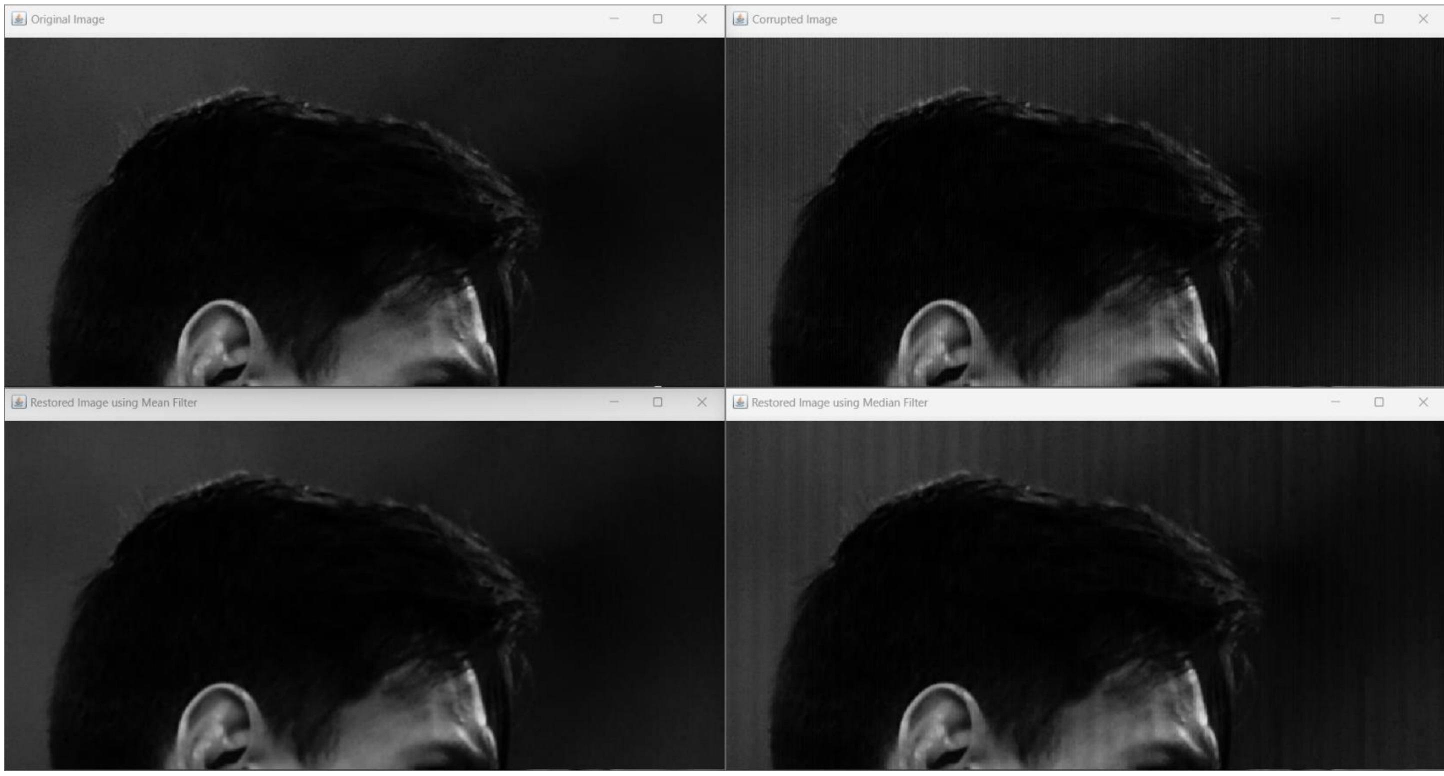
```
Enter the filter window (odd integer): 3
Mean Squared Error (Mean Filter): 9.259820726182726
Mean Squared Error (Median Filter): 14.691960228814018
```




Corrupted Image

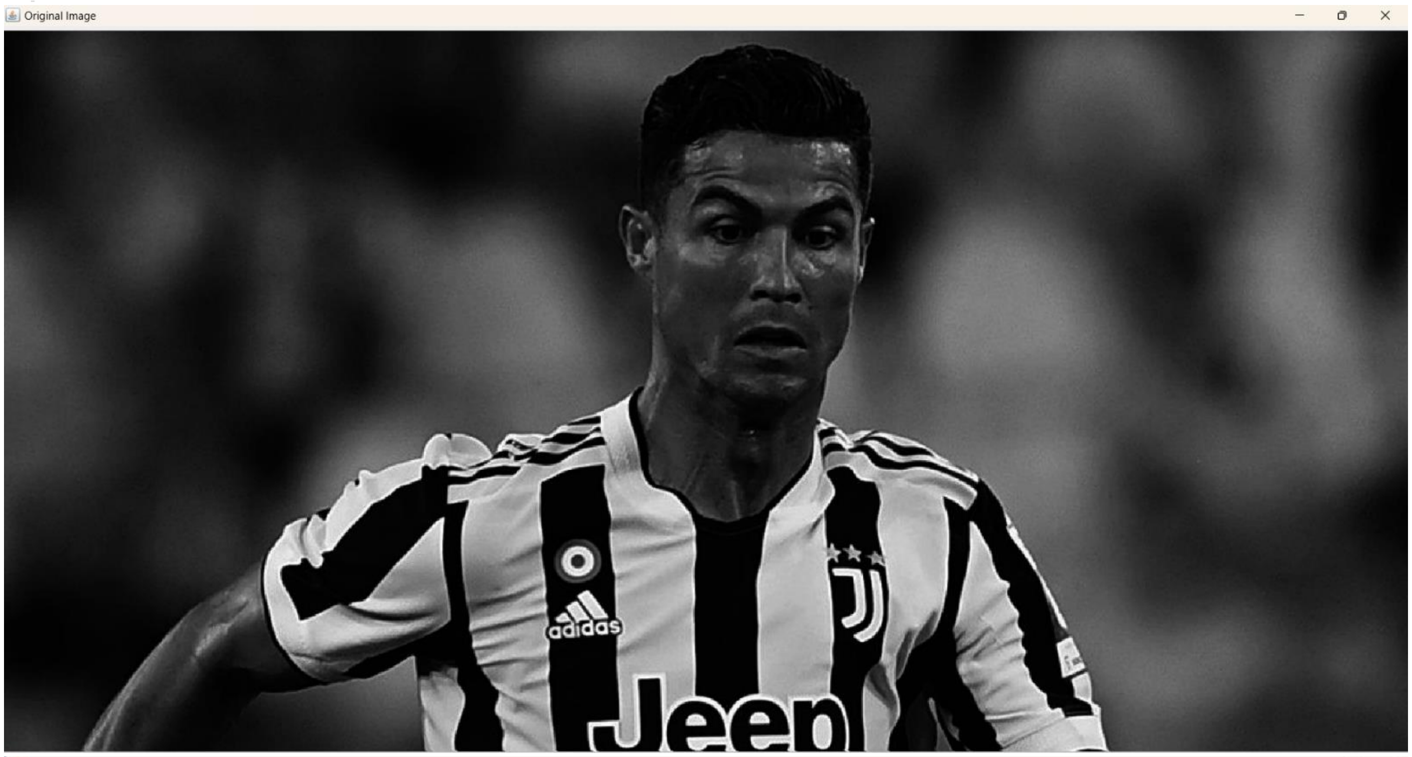

Restored Image using Mean Filter

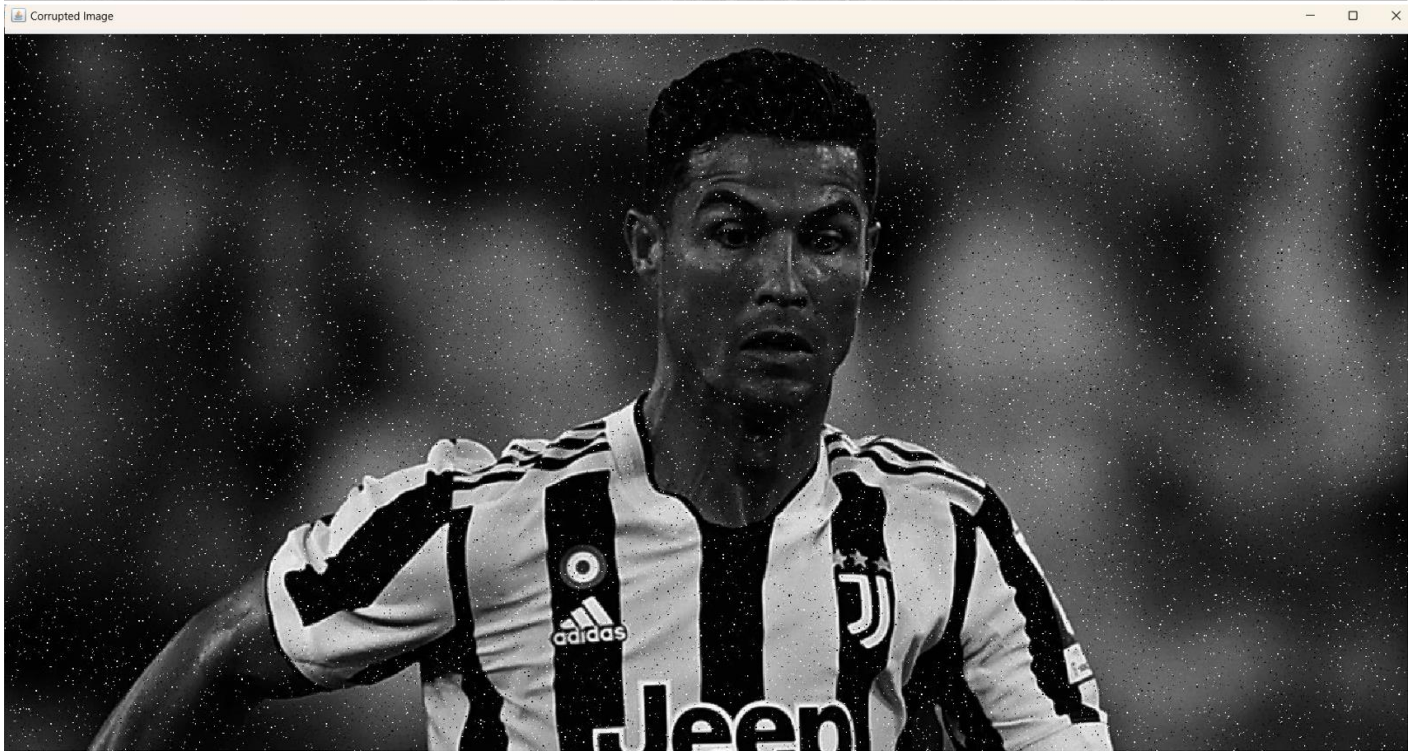

Restored Image using Median Filter
MESSI

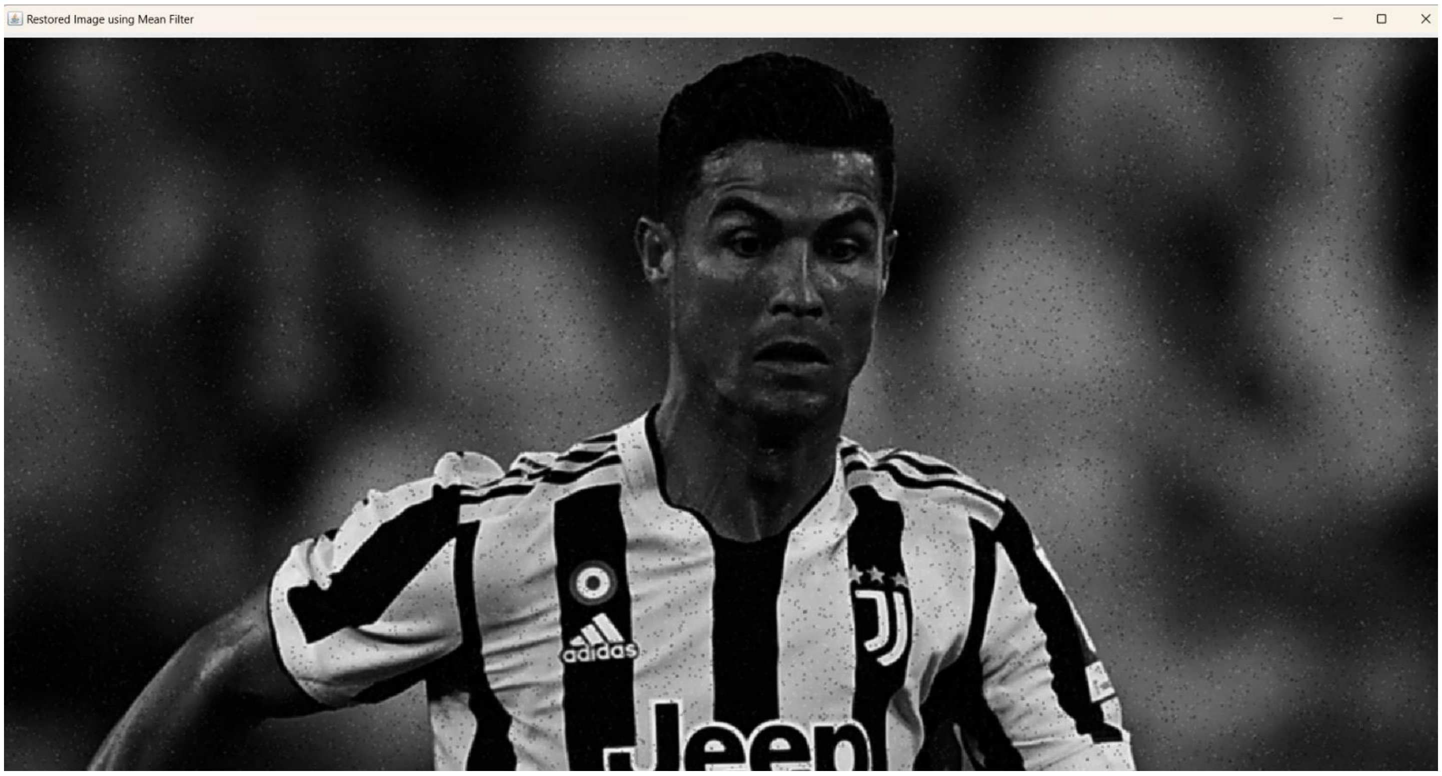
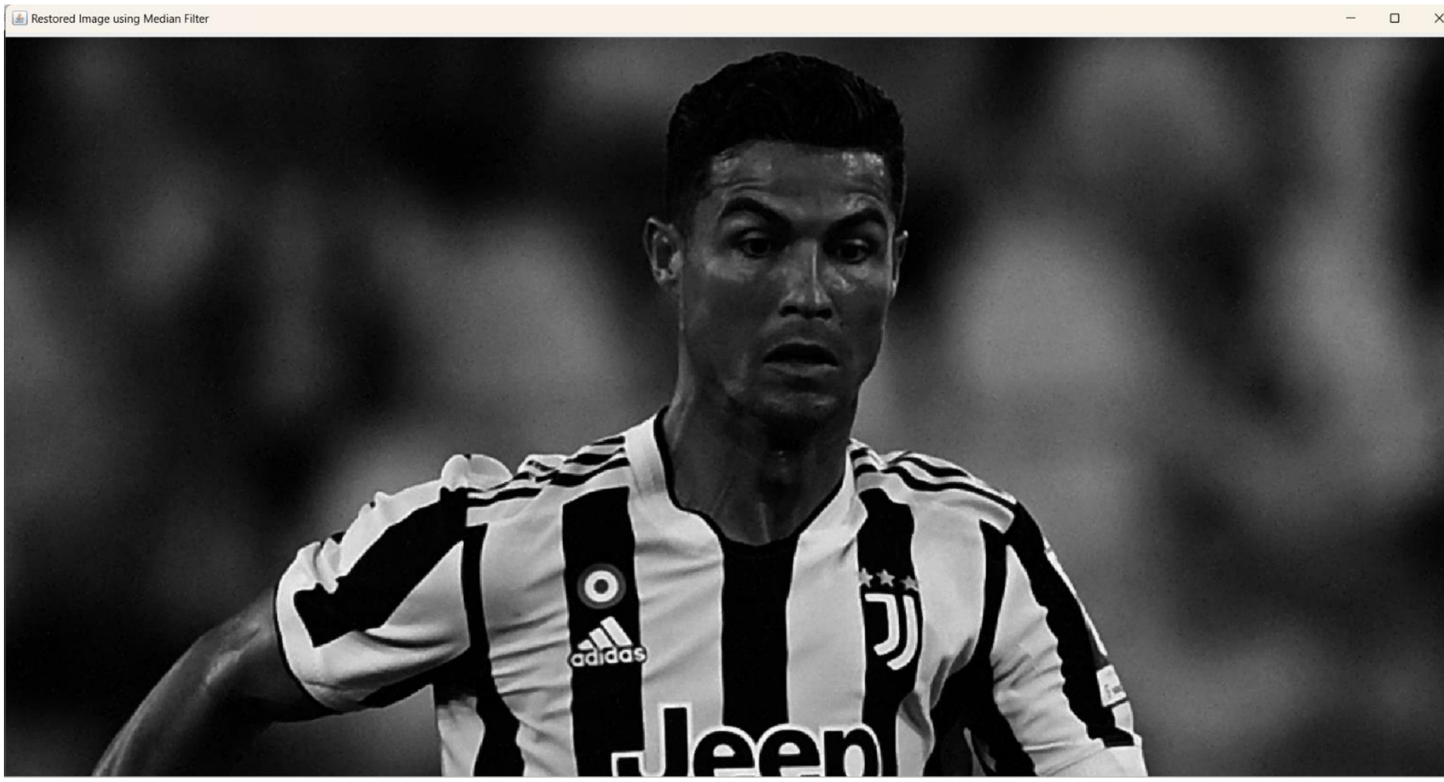

Original Image
Corrupted Image
Restored Image using Mean Filter
Restored Image using Median Filter

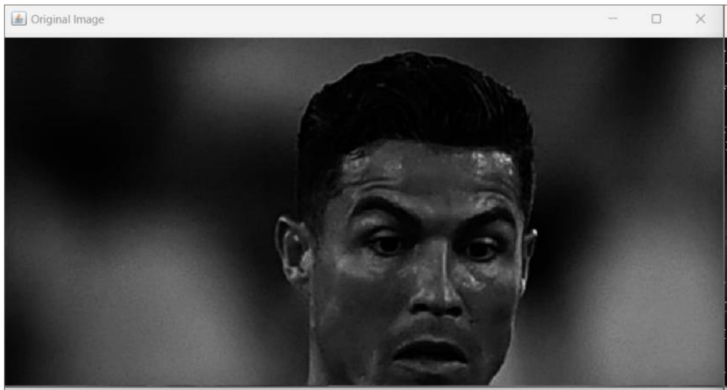
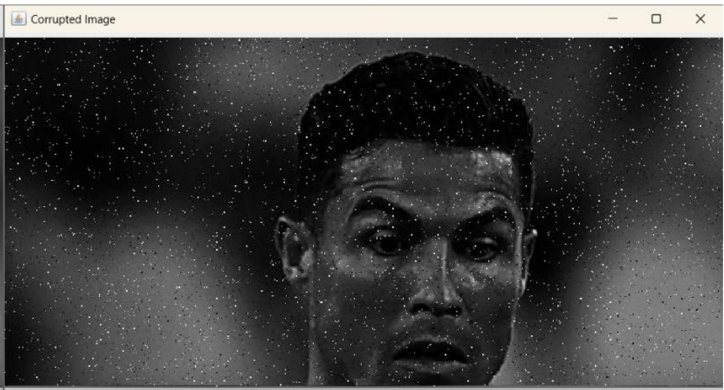
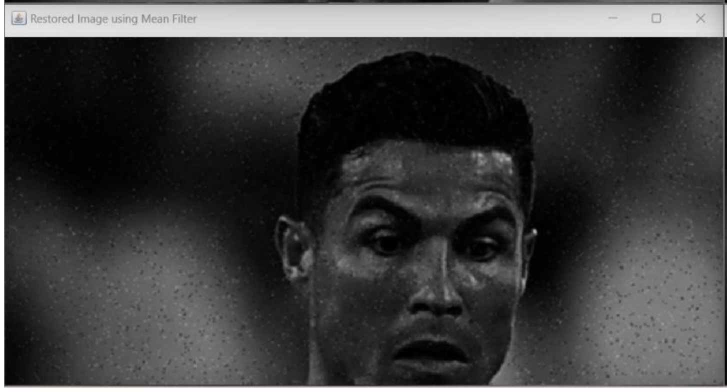
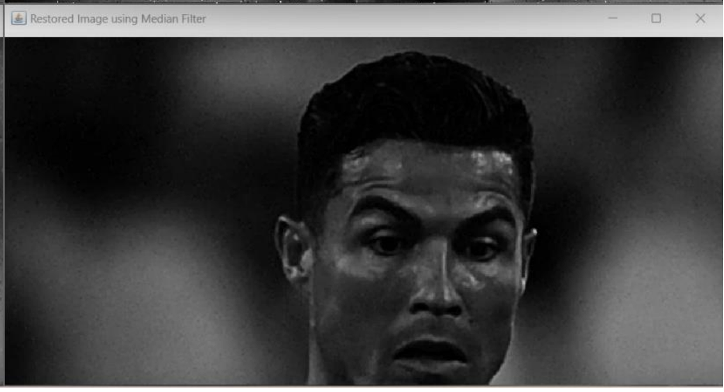
2) For Ronaldo.jpg

```
Enter the filter window (odd integer): 3
Mean Squared Error (Mean Filter): 67.66251274883844
Mean Squared Error (Median Filter): 37.74501378763268
```



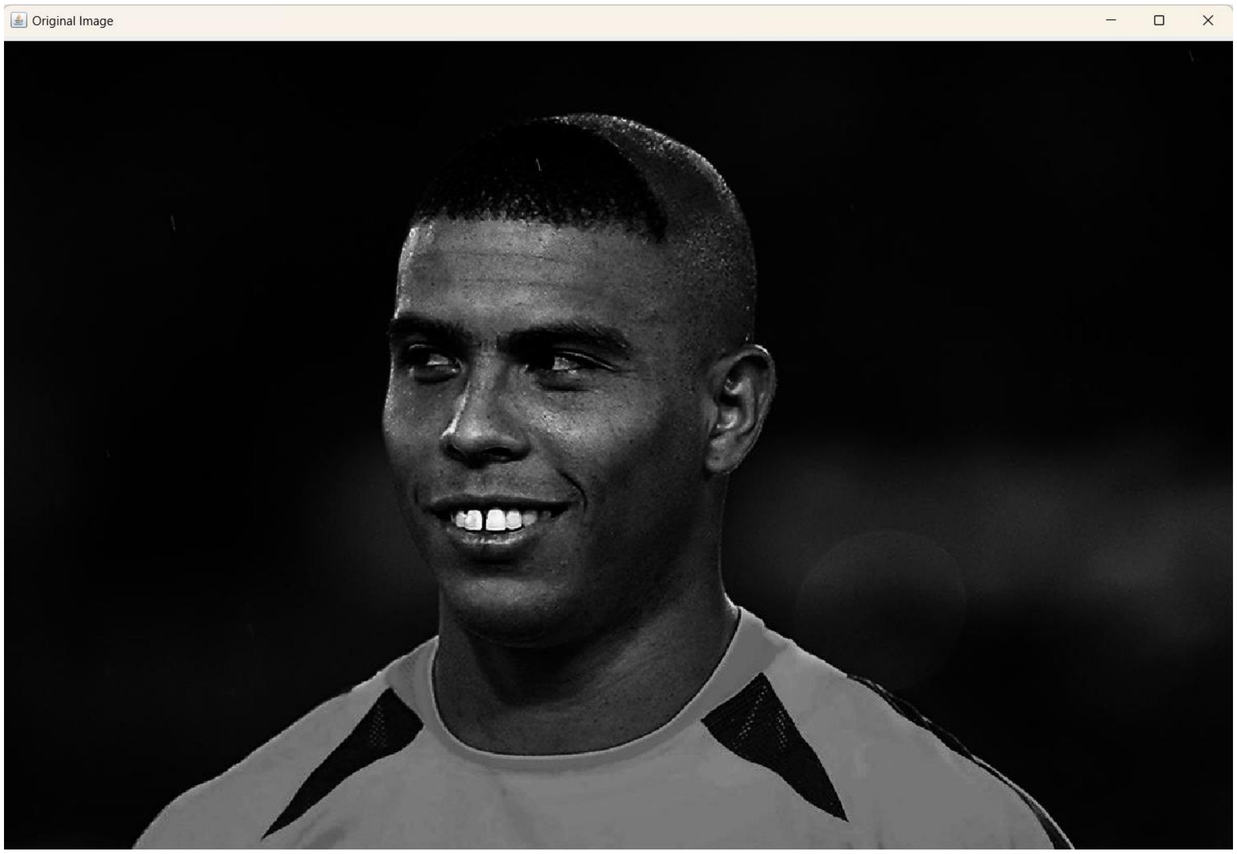


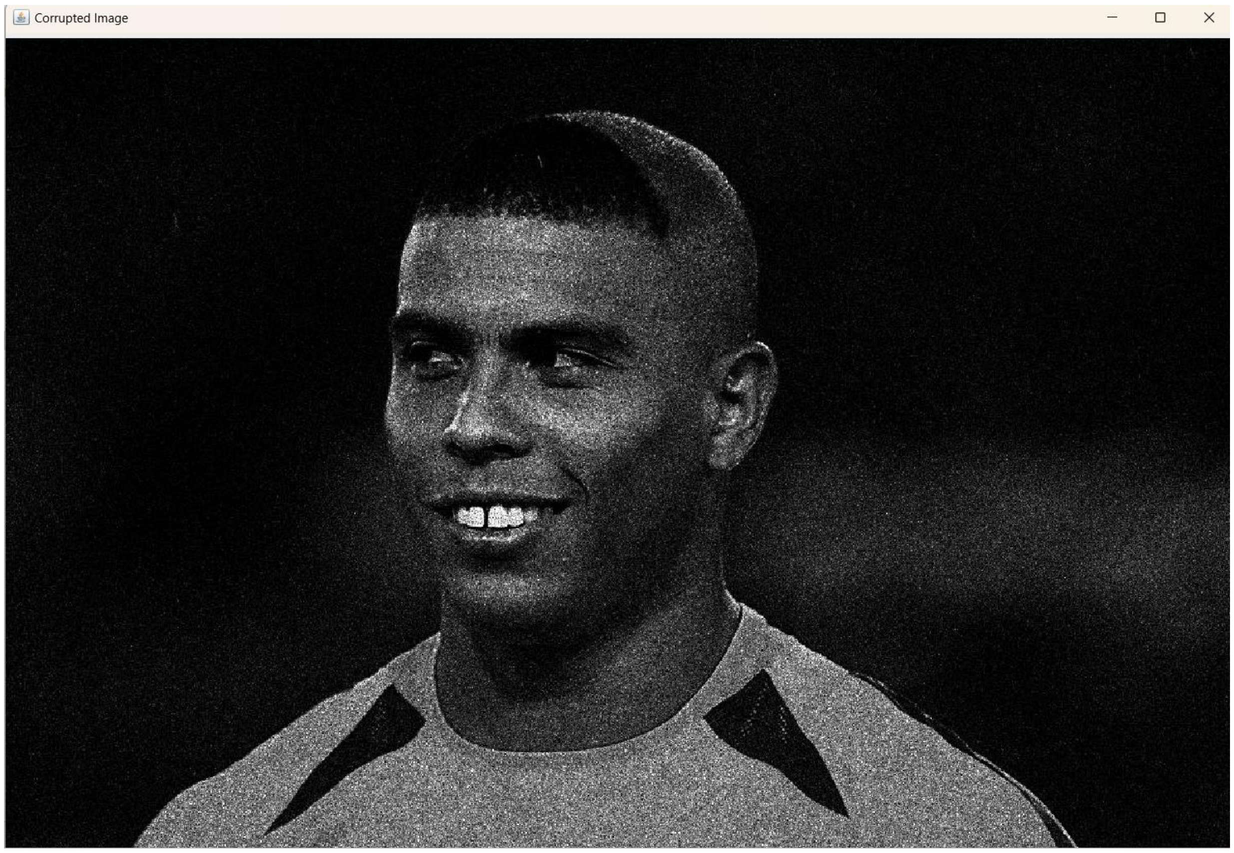

Restored Image using Mean Filter
adidas
Jeep

Restored Image using Median Filter
adidas
Jeep

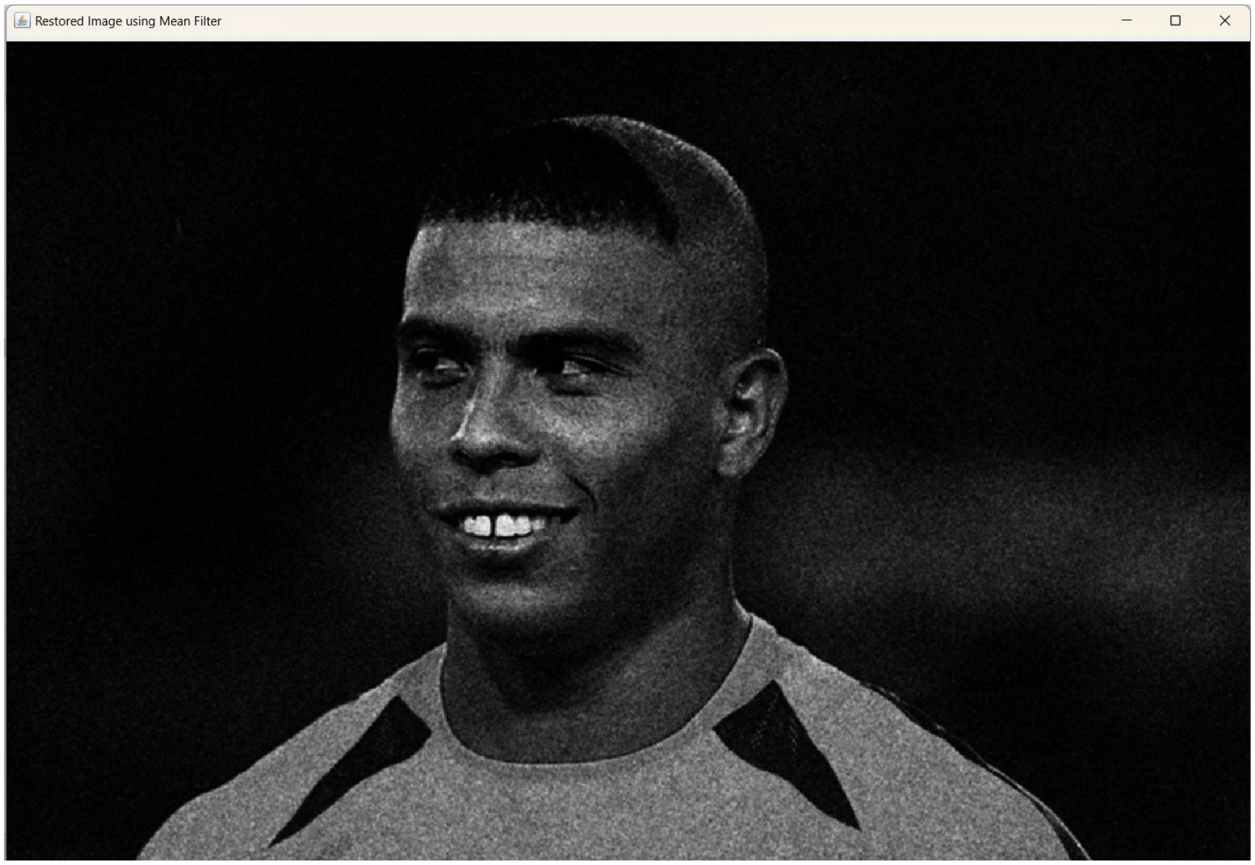

Original Image

Corrupted Image

Restored Image using Mean Filter

Restored Image using Median Filter

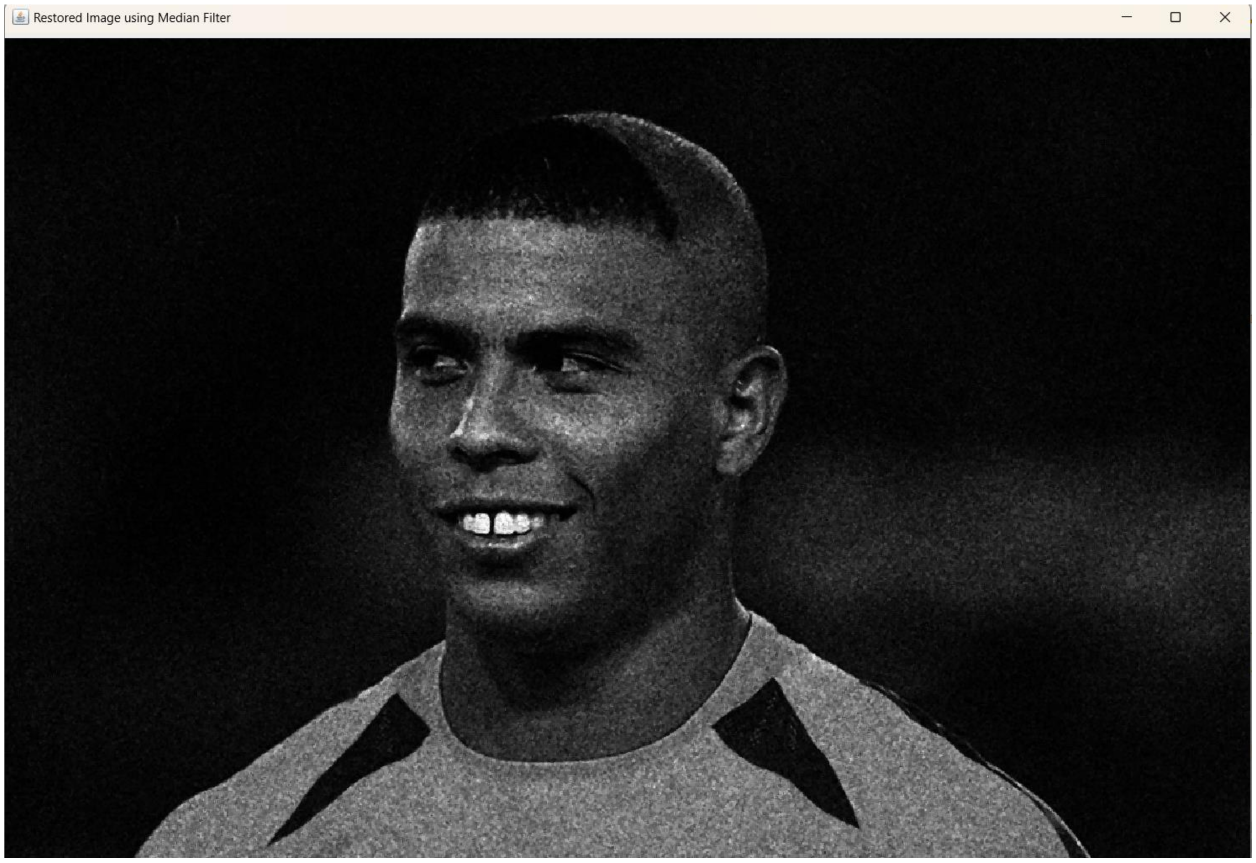
3) Ronaldo de lima.jpg

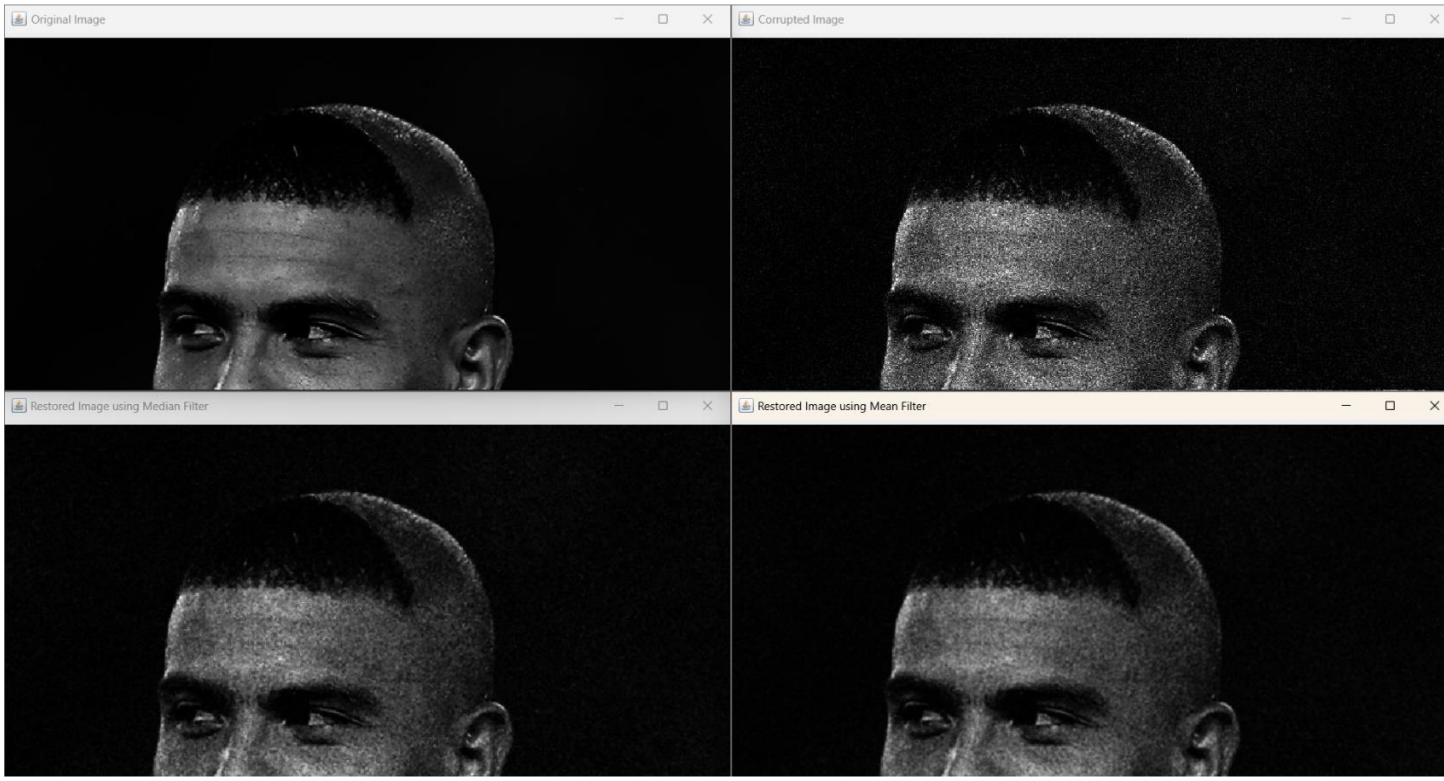
```
Enter the filter window (odd integer): 3
Mean Squared Error (Mean Filter): 161.40599333333333
Mean Squared Error (Median Filter): 226.3770025
```




Corrupted Image


Restored Image using Mean Filter


Restored Image using Median Filter


Original Image
Corrupted Image
Restored Image using Median Filter
Restored Image using Mean Filter