Image Denoising Using Python

By

Rohan Sapre

11BIT025

Biyanta Shah

11BIT028

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

NOVEMBER 2014

Abstract

Image Denoising is one of the important aspects of image processing. Image denoising is the process of removing noise from an image. All image capturing and recording devices are susceptible to noise. The noise can be random, coherent with white noise or non- coherent due to the mechanism of the device. Such disturbances in the image signal while transporting the signal from one place to another may cause distortion in the image, rendering the image unclear. The noise from the image needs to be removed for artistic purposes or practical purposes like computer vision. Image denoising for gray scale images is implemented in this project. Usually for image processing MATLAB is the most standard tool, but we have used Python in order to denoise the image. In this paper we have included various types of filters used in image denoising, along with some comparison parameters and ratios. We conclude showing the difference between image denoising and image enhancement.

Contents

Chapter 1

Introduction

1.1 Introduction to Image Noise

To a computer an image is a collection of numbers that constitute different light intensities in different areas of the image. This numeric representation forms a grid and the individual points are referred to as pixels. Most images consist of a rectangular map of image pixels where each pixel is located and its colour. These pixels are represented horizontally row by row.

All images will not be pure. Images consist of noise, 'Image Noise' is the extra spurious and unwanted information in the image, which obstructs from giving the actual image and distorts the image. Image noise is random variation of brightness or colour information in images and usually an aspect of electronic noise. Noise gets introduced into the data via any electrical system used for storage, transmission, or processing.

There are six main types of noise [1], which are induced into images

1. **Gaussian Noise:**

- It is caused by poor illumination or by high temperature.
- Gaussian Noise is independent at each pixel and is independent of signal intensity. Gaussian Noise is removed by using convolution spatial filters like mean and median filters.

2. **Salt-Pepper Noise:**

- Salt – pepper noise will have dark pixels in bright regions and bright pixels in dark regions. It is caused due to analog to digital converter errors or bit errors in transmission.
- It can be removed by median filter, which is a type of spatial filters, by dark frame subtraction and by interpolating around dark/bright pixels.

Dark frame is an image captured in dark by sensors, just an image of noise. A dark frame or several frames can be subtracted from the subsequent images to correct the fixed pattern noise caused by dark current. These fixed pattern noise is a particular pattern often noticeable during longer exposure shots where certain pixels susceptible to giving brighter intensities above general background noise. Dark current is the small electric current that flows through even when no photons are entering the device.

3. **Shot Noise:**

- Dominant noise in lighter parts of the image due to an image sensor is caused by variation in number of photons sensed at given exposure level. Hence it is also known as Photon Shot Noise.
- Shot noise has a root mean square value proportional to square root of image intensity. The noises at different pixels are independent of each other.

4. **Quantization Noise:**

- In lossy image compression schemes utilizing the discrete cosine transform (DCT), quantization of the DCT coefficients introduces error in the image representation and a loss of signal information. At high compression ratios, this introduced error produces visually undesirable compression artefacts that can dramatically lower the perceived quality of a particular image. This distorted image obtained is due to quantization noise.

5. **Anisotropic Noise:**

- Noise sources that show up significant orientation in images. For example, image sensors are sometimes subject to row noise or column noise.
- Anisotropic noise can be removed by anisotropic diffusion where noise is reduced without removing the significant parts of the image like edges, lines or other details that are important to interpret the image.

6. **Film Grain Noise:**

- Film grain or granularity is the random optical texture of processed photographic film due to the presence of small particles of a metallic silver, or dye clouds, developed from silver halide that have received enough photons. While film grain is a function of such particles (or dye clouds) it is not the same thing as such. It is an optical effect, the magnitude of which (amount of grain) depends on both the film stock and the definition at which it is observed. It can be objectionably noticeable in an over-enlarged photographic film photograph.

Image denoising is the use of different filters to remove such types of noises. Such various denoising filters are discussed in the following sections.

1.2 Objective

The main objective of this research and analysis project is to implement different image denoising algorithms and filters and compare them for efficiency purposes.

1.3 Hardware and Software

- Machine: HP Envy 4
- Microprocessor: Intel core i5 3rd gen processor
- RAM: 4 GB
- Processing Speed: 1.7 GHz
- Operating System: Ubuntu 14.04 LTS
- Programming Platform: Python (version 2.7.6)

Chapter 2

MATLAB v/s Python

2.1 MATLAB

MATLAB is a popular scripting language for numerical computing. It is popular and powerful due to its toolbox. Since MATLAB is developed commercially, dedicated programmers are working in adding new features and enhancing the existing ones all the time. Large image data sets created using latest imaging modalities typically take a long time to process. The typical approach is to run the processing in parallel on many cores either on a desktop, a cluster or a supercomputer. Commercial software like MATLAB require dedicated license such as Distributed Computing Server for running such parallel jobs, which has an added cost.

2.2 Python

Python on the other hand is a free and open source scripting language. It can be scaled to large number of processors. It has been shown that there is not a significant difference in computational time for a program written in MATLAB v/s the one written in python using numpy [2]. Thus Python working with MATLAB will provide the advantages of both.

2.3 Comparison of MATLAB and Python

- In Python the base projects on the top of numpy and matplotlib have been stable for some years, while the more applied packages at the bottom have exploded in recent years. The Python projects are also newer and tend, therefore, to be programmed in a more modern way [3].
- MATLAB is not a real programming language, it is a linear algebraic package, and thus adding non-numerical capabilities becomes complicated. MATLAB would not work for all types of applications such as data mining, web designing; but Python is suitable for these purposes as well as for linear algebraic or statistical operations.
- With Python, you can have a full open-source stack, which means it is free of cost for its users while MATLAB licensing is expensive. Besides the basic MATLAB licenses, you will often need to buy a few more licenses for specific toolboxes. If you need to run the code on a cluster, often that will

mean more licenses [3].

- MATLAB is undoubtedly the tool, which is used for image processing worldwide but keeping in mind the advantages and flexibility of programming in Python, we chose to do our project in Python.

Chapter 3

Spatial Filters

3.1 Mean Filter

- Mean filter reduces the amount of intensity variation between one pixel value and the next, which reduces the noise in images. Mean filter simply replaces the values in the decided kernel size by the mean values of the pixels in the kernel grid. The concerned pixel is replaced mean values of its neighbouring pixels, which fit the kernel grid including itself. Thus it eliminates the pixel values, which are unrepresentative of their surroundings. A default kernel size is a 3x3 matrix. For severe smoothing we use a higher order kernel matrix [4]. The kernel dimensions are always odd numbered.
- Mean Filter is also known as Gaussian Smoothing used for removing Gaussian Noise.
- The problem in mean filter is:
  A single pixel in the grid with a very unrepresentative value can significantly affect the mean value. When filter encounters an edge, filter will interpolate new values for pixels on edge and so it will blur the edge. This is a problem if sharp edges are required.

The algorithm for **Mean Filter in Python** is as follows:

1. Store the noisy image in a numpy array and find its dimensions.
2. The mean matrix is also to be stored in an array.
3. Create a temporary array to store the pixel values of image array; size of the temporary array should be equal to size of the mean matrix (kernel size). Also create the result array, which will store the final denoised image with its dimensions equal to the image array.
4. Now for the size of the row and column of the image, store each 3x3 dimension of image in the temporary array.
5. Once the first 3x3 matrix of image is stored in the temporary array then its dot product is to be performed with the mean matrix.
6. This done, this dot product has to be appended to the final result matrix.
7. Steps 4-6 continues till all the rows and columns of image array are appended into the result array with the dot product of the temporary array and the mean matrix.

8. The result matrix obtained will contained the denoised image of the original noisy image.

Results of the **Mean Filter** are as follows:

Image:

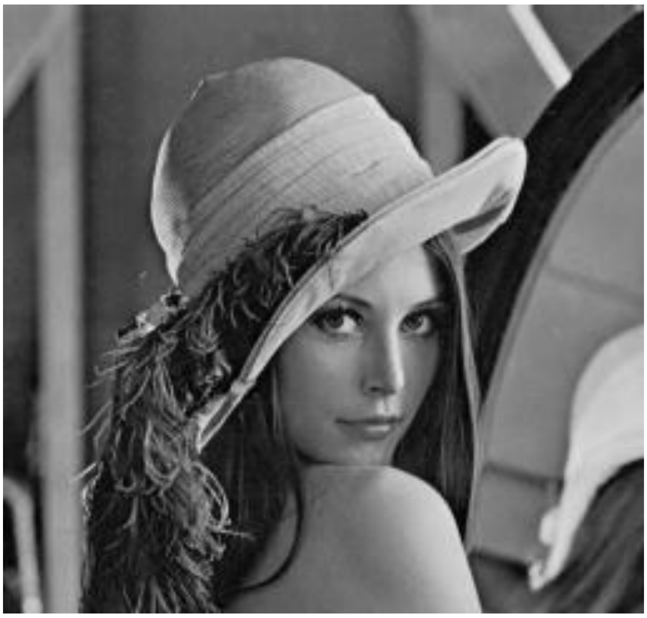


Figure 1: Original Image

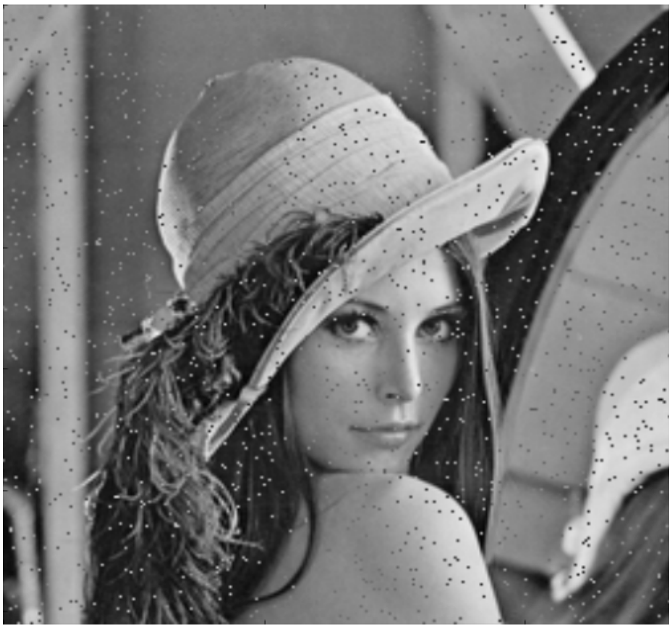


Figure 2: Noisy Image, before applying the mean filter

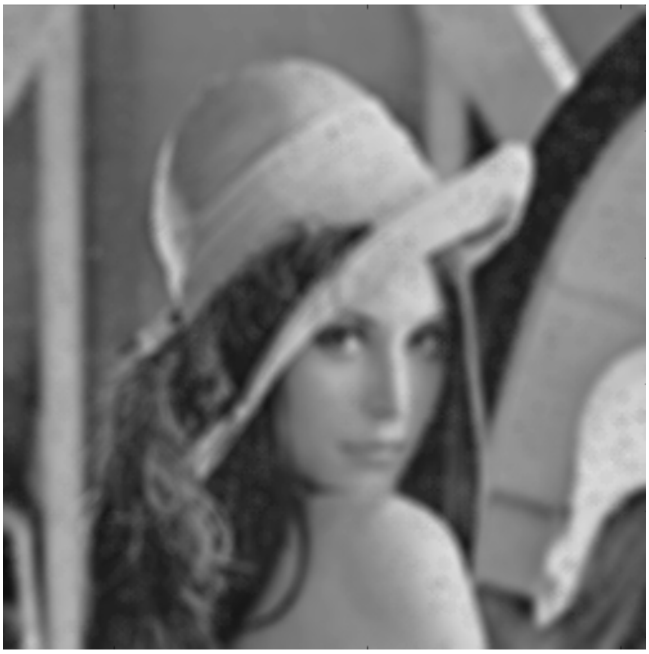


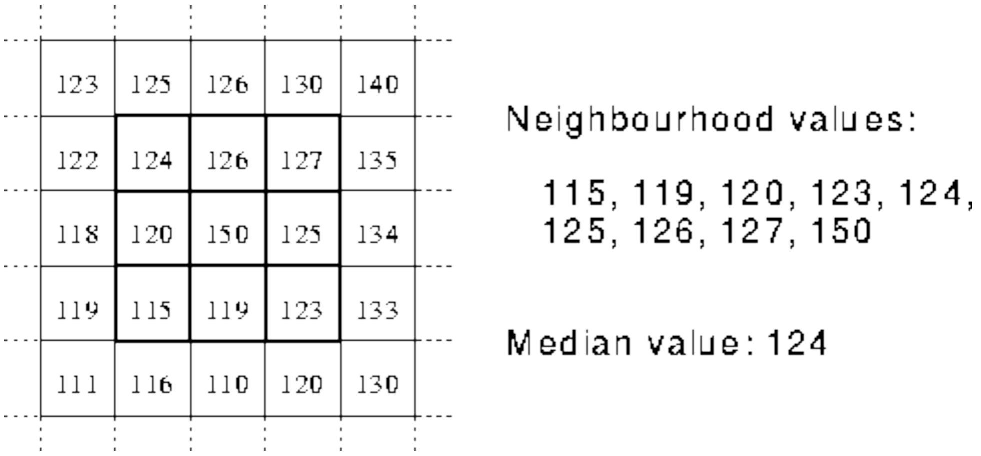
Figure 3: Image after the application of Mean filter

3.2 Median Filter

- Median filter replaces the pixel values with the median of the neighbouring pixel values including it rather than replacing with the mean value. A median filter sort the pixel values in numerical order and replaces the respective pixel with the median pixel value.

- Median filters have two main advantages over mean filter [4]
  1. Median is more robust average than mean and a single very unrepresentative pixel in the kernel matrix will not affect the median value.
  2. Median is one of the values of the pixels in the kernel grid so unrealistic new pixels will not be created, hence median when encounters an edge will not straddle it. Thus it is better at preserving sharp edges of an image.
- Though Median filter is better than mean filter, it takes a longer time to compute. Thus increases complexity.

If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used. Figure illustrates an example calculation of how median is better than mean.



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Figure 4 [5]: Calculating the median value of a pixel neighborhood. As can be seen, the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value: 124. A 3×3 square neighborhood is used here. Larger neighborhoods i.e. higher kernel value will produce more severe smoothing.

The algorithm for **Median Filter in Python** is as follows:

1. Store the noisy image in a numpy array and find its dimensions.
2. The median matrix is also to be stored in an array.
3. The image array will be stored in an array and sorted in ascending order.
4. Size of the temporary array should be equal to size of the median matrix (kernel size). Also create the result array, which will store the final denoised image with its dimensions equal to the image array.
5. Now for the size of the row and column of the image, store each 3x3

dimension of image in the temporary array.

6. Once the first 3x3 matrix of image is stored in the temporary array then its dot product is to be performed with the median matrix.
7. The median matrix is the average of the fourth and fifth element of each 3x3 matrix copied.
8. This done, this dot product has to be appended to the final result matrix.
9. Steps 4-6 continues till all the rows and columns of image array are appended into the result array with the dot product of the temporary array and the median matrix.
10. The result matrix obtained will contained the denoised image of the original noisy image.

Results of **Median Filter** are as follows:

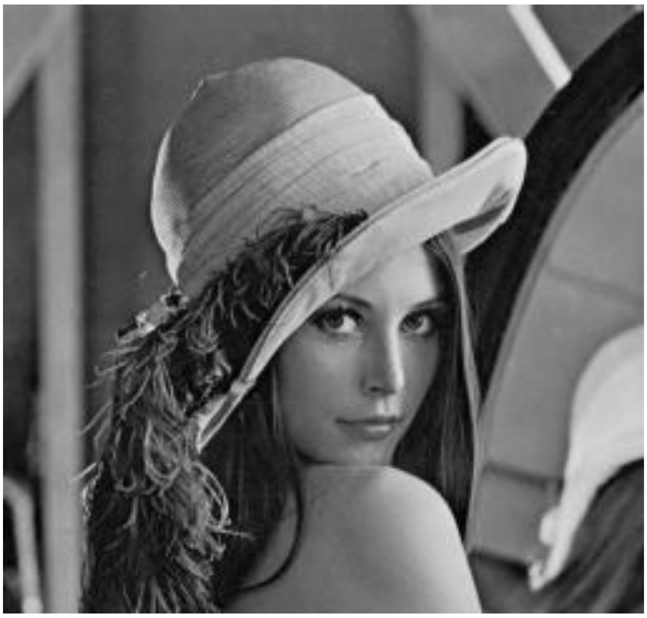


Figure 5: Original Image, before adding the noise

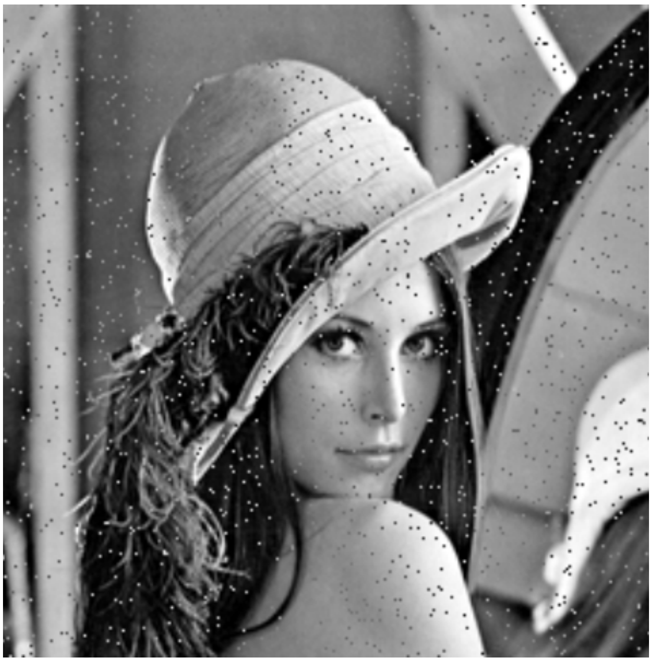


Figure 6: Image after adding noise, ready to be passed onto the median filter

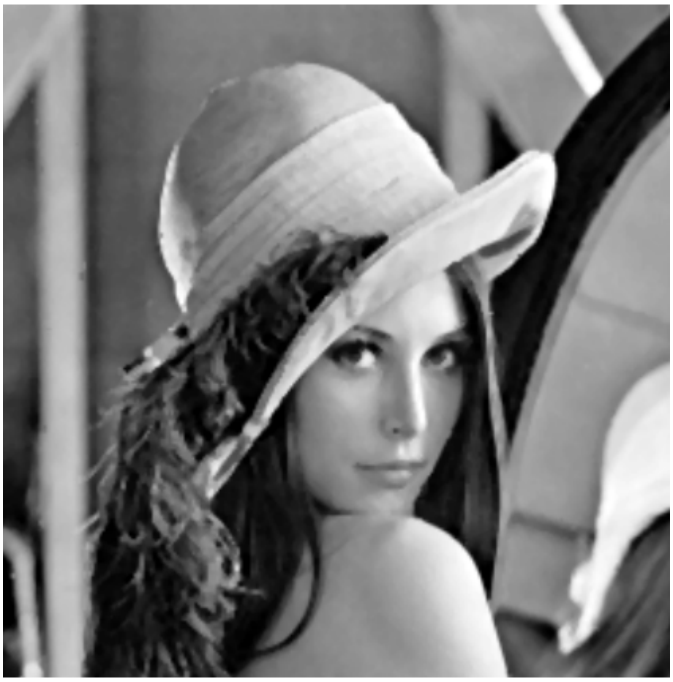


Figure 7: Image after the application of median filter

We can compare from Figure 3 and Figure 7, that median filter is better than mean filter and gives a better smoothening effect.

Although, Median filter is a useful non-linear image smoothing and enhancement technique. It also has some disadvantages. The median filter removes both the noise and the fine detail since it can't tell the difference between the two. Anything relatively small in size compared to the size of the neighborhood will have minimal affect on the value of the median, and will be filtered out [5]. In other words, the median filter can't distinguish fine detail from noise.

3.3 Adaptive Median Filter

- The adaptive median filtering has been applied widely as an advanced method compared with standard median filtering. The Adaptive Median Filter performs spatial processing to determine which pixels in an image have been affected by impulse noise.
- The Adaptive Median Filter classifies pixels as noise by comparing each pixel in the image to its surrounding neighbor pixels. The size of the kernel is adjustable, as well as the threshold for the comparison. A pixel that is different from a majority of its neighbors, as well as being not structurally aligned with those pixels to which it is similar, is labeled as impulse noise [4].
- These noise pixels are then replaced by the median pixel value of the pixels in the neighborhood that have passed the noise-labeling test.

Adaptive median filter does the following things

- Removes impulse noise
- Smoothing of other noise
- Reduces excessive distortion like thinning or thickening of object boundaries.

Chapter 4

Logarithmic Transformation

A logarithmic transformation can be briefly described as the dynamic range of each pixel value in the image being compressed by its logarithmic value. By applying this transform, the low intensity pixel values will be enhanced. The logarithmic operator is a simple point processor where the mapping function is a logarithmic curve. In other words, each pixel value is replaced with its logarithm. Most implementations take either the natural logarithm or the base 10 logarithm. However, the basis does not influence the shape of the logarithmic curve, only the scale of the output values, which are scaled for display on an 8-bit system. Hence, the basis does not influence the degree of compression of the dynamic range. The logarithmic mapping function is given by [6]:

$$Q(i,j) = c \, \log(|P(i,j)|) \qquad (1)$$

Since the logarithm is not defined for 0, many implementations of this operator add the value 1 to the image before taking the logarithm. The operator is then defined as [6]:

$$Q(i,j) = c\ \log(1 + |P(i,j)|) \quad (2)$$

The scaling constant *c* is chosen so that the maximum output value is 255 (For an 8-bit image). That means if *R* is the value with the maximum magnitude in the input image, *c* is given by [6]:

$$c = \frac{255}{\log(1 + |R|)} \quad (3)$$

Algorithm for **Logarithmic transformation** is as follows:

1. The image on which this transform is to be performed is stored in a numpy array and its dimensions are found out using the shape () function in python.
2. The size of the image is found out i.e. no of bits per sample is calculated.
3. Once the size is found (2^ (size of image) – 1) becomes the numerator in equation (3).
4. Using the dimensions of the image, the pixel having the maximum magnitude i.e. the highest frequency is found out and that value is given to R. Since all values are found, we can now find the value of constant c using equation (3).
5. Once value of c is found, using equation (2), value of the final image can be found which will be the logarithmic transform of the original image.
6. Equation (2) will have to be performed as many times as the dimensions of the original image. In the equation, i and j are the rows and columns of the image respectively.

Results of **Logarithmic Transform** are shown below:

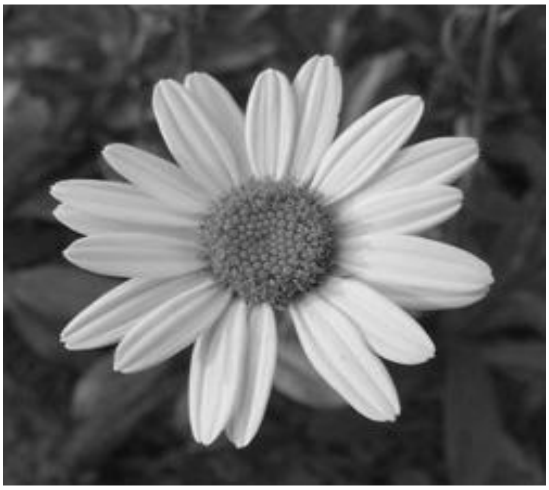


Figure 8: Original Image, before applying the logarithmic transforms

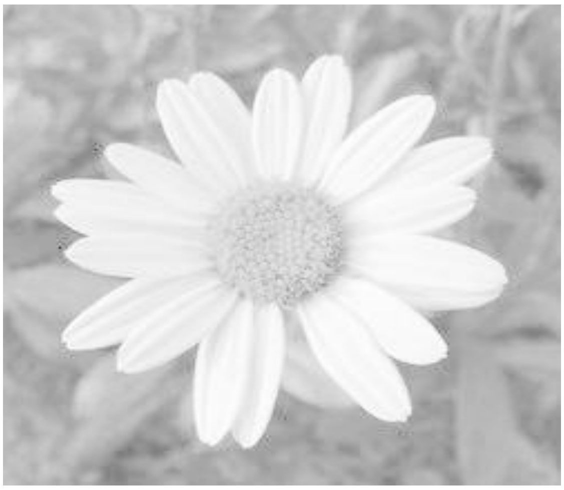


Figure 9: Image after applying Logarithmic Transform. The low intensity (lighter pixels) become darker while the high intensity pixels become lighter.

Chapter 5

PSNR Value

5.1 Introduction to PSNR

Improving the visual quality of an image can be subjective. The conclusion of a method being superior over another method may vary from person to person.

Hence a proper metric is required to test the quality of an image. PSNR is one such image quality metric. The term PSNR stands for **Peak Signal to Noise Ratio,** which is the ratio of the maximum possible signal in the image to the noise in the image. If the noise is less, then PSNR value will be greater, which proves to be a better image quality. Thus images with a higher PSNR value are considered to be superior in quality.

Using the same set of images, different filters will be applied and their PSNR values calculated. The filter, which gives us the highest PSNR value, is the superior filter.

To calculate PSNR, we require the original image and the degraded image to have the same dimensions. Then we are required to calculate the MSE of each image i.e. the original and noisy image. Mean squared error (MSE) of an original image measures the average of the squares of the "errors", that is, the difference between the original image and noisy image. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss [7]. The difference occurs because of randomness or because the original image doesn't account for information that is present in the noisy image. The MSE of two identical images will be zero because there will be no differences between the two images [7].

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{4}$$

Because many signals have a very wide dynamic range, (ratio between the largest and smallest possible values of a changeable quantity) the **PSNR** is usually expressed in terms of the logarithmic decibel scale [8].

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned} \tag{6}$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. More generally, when samples are represented with B bits per sample, MAX_I is 2^B-1. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three [8].

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better. For 16-bit data typical values for the PSNR are between 60 and 80 dB. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB [8].

5.2 Comparison of PSNR values of normal and filtered images

The procedure that we followed for comparison of different filters is:

1. Take an image, store it in a numpy array. Now add noise to it and store that to in a numpy array.
2. Since the dimensions of the images will be known by the shape () function, we can find out the MSE of the original image, by using Equation (5). 'm' will be the rows of the image and 'n' will be the columns of the image.
3. Using mathematical operations available in the python libraries, the MSE will be found out.
4. The next step is to find the maximum magnitude in the image i.e. the pixel with the maximum value. This is found by comparing each pixel with another to find the greatest among them. This pixel will be found from the original image.
5. Now we have MAX_I and MSE, so using the math.log () function, we can find out the PSNR.
6. Now we filter the noisy image, using the in-built filter median, and then we repeat step 2 to step 5, but now in place of the original image we will use the filtered image.

When this algorithm was carried out, the PSNR for the original image came out to be **43.677**, while that for the median filtered image came out to be **33.333**. This proves that though the values are in the range of accepted PSNR values, the PSNR value of the original image is higher than the median filtered image and thus the noise is not fully removed by the median filter. Thus original image is superior.

Chapter 6

Histogram Equalization

6.1 Introduction to Histograms

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.

The horizontal axis of the graph represents the tonal variations, while the vertical axis represents the number of pixels in that particular tone. The left side of the horizontal axis represents the black and dark areas, the middle represents medium grey and the right hand side represents light and pure white areas. The vertical axis represents the size of the area that is captured in each one of these zones. Thus, the histogram for a very dark image will have the majority of its data points on the left side and center of the graph [9]. Conversely, the histogram for a very bright image with few dark areas or shadows will have most of its data points on the right side and center of the graph.

Image histograms are present on many modern digital cameras. Photographers can use them as an aid to show the distribution of tones captured, and whether image detail has been lost to blown-out highlights or blacked-out shadows.

Image editors typically have provisions to create a histogram of the image being edited. The histogram plots the number of pixels in the image (vertical axis) with a particular brightness value (horizontal axis). Algorithms in the digital editor allow the user to visually adjust the brightness value of each pixel and to dynamically display the results as adjustments are made [9]. Improvements in picture brightness and contrast can thus be obtained.

In the field of computer vision, image histograms can be useful tools for thresholding. Because the information contained in the graph is a representation of pixel distribution as a function of tonal variation, image histograms can be analyzed for peaks [9]. This threshold value can then be used for edge detection and image segmentation.

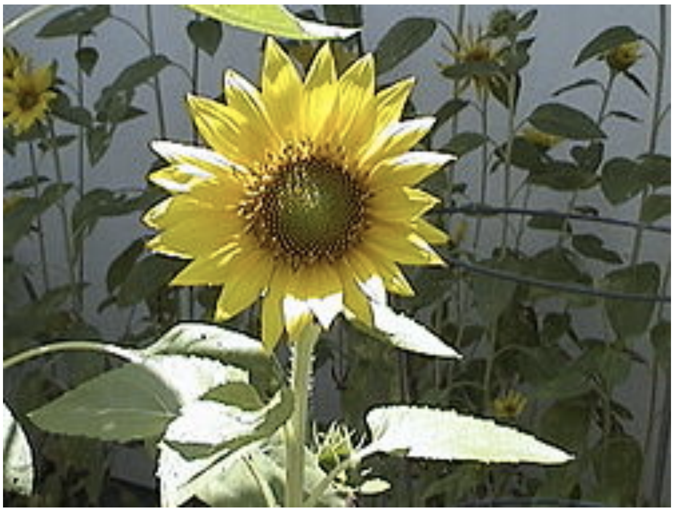


Figure 10: Image whose histogram is displayed below [11].

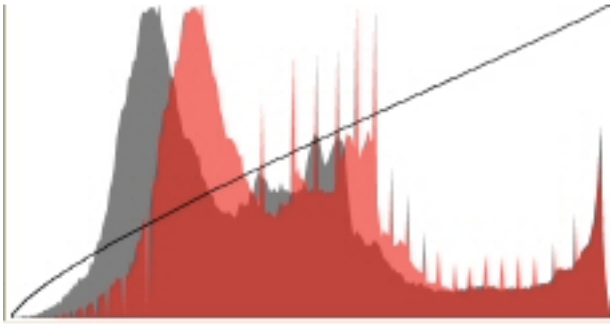


Figure 11: The Histogram, where the x-axis represents variations in tone while y-axis represents the number of pixels in the tone. The tonal variations go from dark → light [11].

6.2 Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator [10]. So in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive.

A disadvantage of the method is that it does not differentiate. It may increase the contrast of background noise, while decreasing the usable signal. In scientific imaging where spatial correlation is more important than intensity of signal, the small signal to noise ratio usually hampers visual detection. For example separating the DNA fragments of quantized length [10].

Histogram equalization often produces unrealistic effects in photographs; however it is very useful for scientific images like thermal, satellite or x-ray images, often the same class of images that user would apply false-color to. Also histogram equalization can produce undesirable effects when applied to images with low color depth. For example, if applied to 8-bit image displayed with 8-bit gray-scale palette it will further reduce color depth i.e. number of unique shades of gray of the image. Histogram equalization will work the best when applied to images with much higher color depth than palette size, like continuous data or 16-bit gray-scale images.

Histogram equalization is implemented in two ways : image change or palette change. If we define a new image, keeping the palette same, then it is known as image change. If the palette is changed, keeping image constant, then it is known as palette change. Palette change is better than image change as it preserves the original data of the image.

To implement the histogram equalization, we first need to find the **PMF** and **CDF** values.

PMF is **Probability Mass Function** [10], which gives the probability of each pixel in the image or the frequency of the count of a pixel in the image. All the pixel values appearing in image are displayed along with the total count of pixels.

Then the first pixel is taken and checked as to how many times it has been repeated. The number of times the pixel has occurred in the image/ total count is the probability of that pixel. Similarly this is carried out for all the pixels. We see that when we plot these values of the pixels with their probabilities, they will not increase monotonically. In order to make it increase monotonically, we will calculate the CDF.

CDF is **Cumulative Distributive Function** [10], which is a function that calculates the cumulative sum of all the values calculated by PMF. As we add the values obtained in PMF, we will get a monotonically increasing function, which is a necessary condition for histogram equalization.

Now we multiply the CDF values that are obtained with the gray level -1, where gray level is 2^(bits per pixel). The values that we get after multiplying with CDF, will map them to the old gray level values, and the histogram will be equalized.

Algorithm for implementing **Histogram Equalization** is as follows:

1. The image to be equalized is stored in an array and its dimensions are found out.
2. The values of the pixels from the image array are extracted and a loop is run according to the dimensions of the image, to calculate the occurrence of the single value of pixel in the entire image. This is then divided by the total number of pixels, thus giving us the probability of occurrence of that pixel value.
3. Step 2 is repeated for all the pixels in the image, which will give us the PMF.
4. CDF is found out for all these pixels. The value of the first pixel probability will remain as it is and all the rest will be added with the value of the pixel coming before it, which at last gives us the total number of pixels.
5. This CDF found, we multiply each value with the value of gray level – 1. For example if the bits per pixel is 3, then number of levels is 8, thus we multiply the CDF by 7.
6. These new values of CDF are then mapped onto the old values, with its frequencies.
7. The new values are stored in a separate array, which is the equalized image.

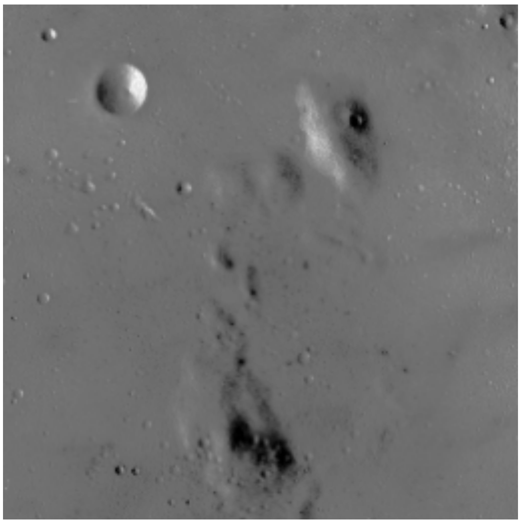


Figure 12: Original low contrast image.

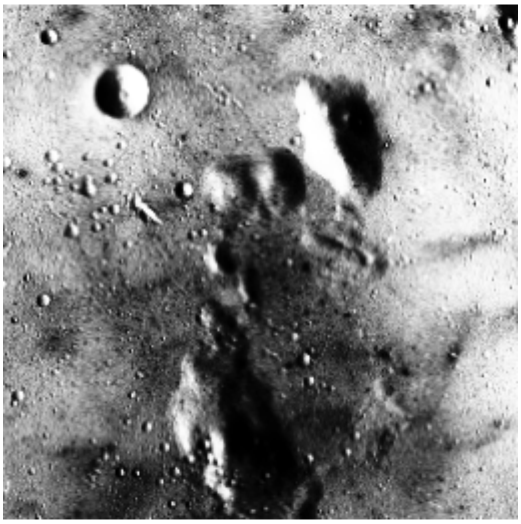


Figure 13: Histogram Equalization Image (High Contrast)

As shown in the figure 12 and 13, the equalized image will have its low intensity areas enhanced and the image will have a higher contrast. During histogram equalization, the shape of the histogram changes which is different from histogram stretching, because there its overall shape does not change.

Chapter 7

Comparison of Denoising and Enhancement

7.1 Image Enhancement

Image enhancement is basically improving the interpretability or perception of information in images for human viewers and providing `better' input for other automated image processing techniques. The principal objective of image enhancement is to modify attributes of an image to make it more suitable for a given task and a specific observer. During this process, one or more attributes of the image are modified. The choice of attributes and the way they are modified are specific to a given task. Moreover, observer-specific factors, such as the human visual system and the observer's experience, will introduce a great deal of subjectivity into the choice of image enhancement methods s[14].

One type of image enhancement is point processing, also known as contrast enhancement [12]. It is the process to produce a higher contrast for an image than the original by darkening a particular level. Enhancement at any point in an image depends only on the gray level at that point.

Image enhancement is broadly categorized into two categories:

1. Spatial Domain Methods
2. Frequency Domain Methods

In spatial domain techniques, we directly deal with the image pixels. The pixel values are manipulated to achieve desired enhancement.
In frequency domain methods, the image is first transferred in to frequency domain. It means that, the Fourier Transform of the image is computed first. All the enhancement operations are performed on the Fourier transform of the image and then the Inverse Fourier transform is performed to get the resultant image. These enhancement operations are performed in order to modify the image brightness, contrast or the distribution of the grey levels. As a consequence the pixel value (intensities) of the output image will be modified according to the transformation function applied on the input values [13].

A few examples of image enhancement are: negative of an image, logarithmic transformations and histogram equalizations. An example of image enhancement is shown in figure 13.

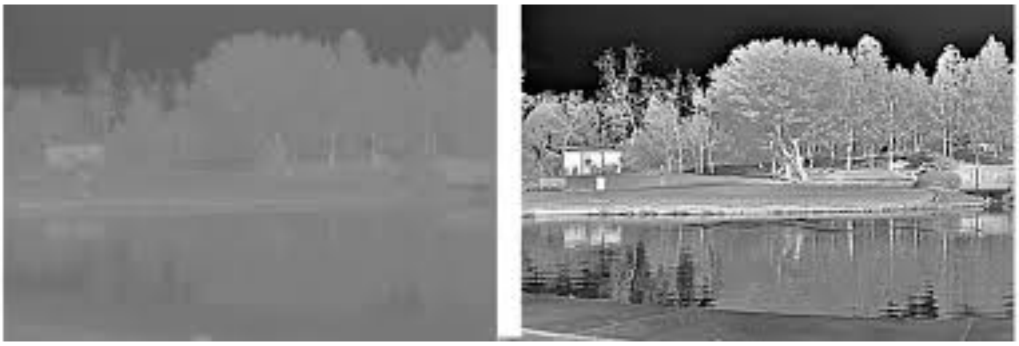


Figure 13: By image enhancement, enhancing the contrast, the image is transformed [14].

7.2 Image Denoising v/s Image Enhancement

IMAGE DENOISING	IMAGE ENHANCEMENT
1. Image Denoising is considered with removing the noise and distortion from the image.	1. The main aim of Image Enhancement is to make the image better for visual appearance and make it sharper.
2. Filters for denoising are used to remove the noise, by replacing the values of the pixels with similar values depending upon the algorithms. This leads to smoothening of the edges.	2. Image enhancement algorithms do not attempt to smoothen the edges. Their main purpose is to make the edges sharper and the colors more contrasting [13].
3. When denoising an image, the color of the denoised image remains the same.	3. The color of the image might change and noise might also be induced in the enhanced image.
4. Denoising is helpful for satellite images to determine the crops or weather. With a distorted image such determination is not possible.	4.Enhancement is helpful for medical images, facial recognition; where a sharp image is necessary rather than an image with blurred lines [12].
5. Filters for image denoising are: mean, median, anisotropic diffusion, wavelet thresholding etc.	5. Image enhancement techniques are: logarithmic transform, histogram equalization, negative of image etc.

Chapter 8

Conclusion

Image denoising is usually implemented in MATLAB, but we chose to implement it in Python owing to the aforementioned advantages. After observing the results of both Mean and Median filters, we found that the image gets blurred when passed through a mean filter, while median filter does retain the sharpness of the image. Considering the pixel values over a large range, the mean varies more than the median, therefore, we observed that median filter would be more efficient in image denoising than mean filter. Logarithmic transformation filters are generally applied to dark images in order to enhance the low intensity pixels resulting in a brighter image. The PSNR value of two images help us compare the effects of different filters on the noisy image. It determines the efficiency of the filters to remove the noise from an image. Histogram Equalization distributes the intensity over the histogram by spreading the most frequent intensity values, which eventually increases the contrast of the image. At last we differentiate the two most important domains of image processing, which are image denoising and image enhancement.

References

[1] R. Gonzalez and R. Woods *Digital Image Processing*, Addison-Wesley Publishing Company, Edition 3, 1992.

[2] Python, *www.stat.washington.edu/~hoytak/blog/whypython.html*; last viewed on July 16, 2014.

[3] Matlab v/s Python, *www.pyzo.org/python_vs_matlab.html*; last viewed on July 16, 2014.

[4] A. Jain *Fundamentals of Digital Processing*, Prentice Hall, 1989.

[5] Mean and median filter, *http://www.markschulze.net/java/meanmed.html*; last viewed on July 20, 2014.

[6] Logarithmic Transform, *http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixlog.htm;* last viewed on August 15, 2014.

[7] Hore A , Ziou D, *Image quality Metrics: PSNR vs. SSIM* in Pattern recognition (ICPR), 2010 20th International Conference from August 23-26, 2010.

[8] *Peak Signal to Noise Ratio as an Image quality Metric* in National Instruments, published on September 11, 2013.

[9] Introduction to Histograms, *http://www.illustratedphotography.com/basic-photography/zone-system-histograms;* last viewed on November 12, 2014.

[10] Histogram Equalization, *http://www.tutorialspoint.com/dip/Histogram_Equalization.htm*; last viewed on November 12, 2014.

[11] Histograms, *http://what-when-how.com/wp-content/uploads/2011/09/tmp646.jpg*; last viewed on November 12, 2014.

[12] M Unser, *Splines: A Perfect fit for signal and image processing,* in Signal Processing magazine, IEEE, Volume 16, Issue 6.

[13] Bhabatosh Chanda and Dwijesh Dutta Majumder, *Digital Image Processing and Analysis*, PHI Learning Pvt. Ltd., 2004.

[14] Raman Maini and Himanshu Aggarwal, *A Comprehensive Review of Image Enhancement Techniques*, Journal Of Computing, Volume 2, Issue 3, March 2010, ISSN 2151-9617.