IIT (BHU) VARANASI

Exploratory Project

CSM-291 I Report

Submitted By

Vaishnavi Gupta
Roll Number: 20124049
Email Id: vaishnavi.gupta.mat20@itbhu.ac.in

Hitansh Shah
Roll Number: 20124044
Email Id: shah.hitanshsanjay.mat20@itbhu.ac.in

Guided by

Dr. R.K. Pandey

Contents

Objective

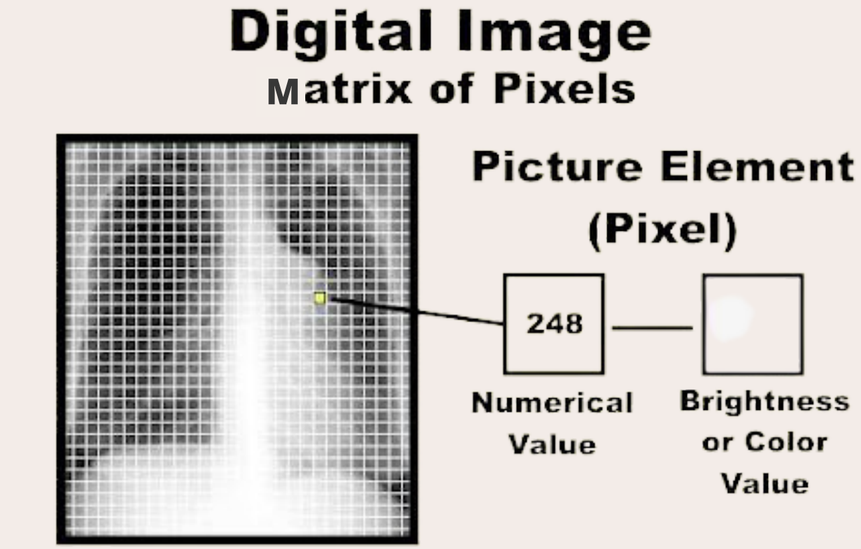
The main objective is to understand and implement some spatial domain methods of image denoising like Gaussian filter, Median filter, Average filter and also to understand and implement the Fractional Filter (k-operator based) for image denoising and to compare the Fractional filter with some of the other methods.

Let us first take a look at some of the important terms related to digital image processing.

Introduction

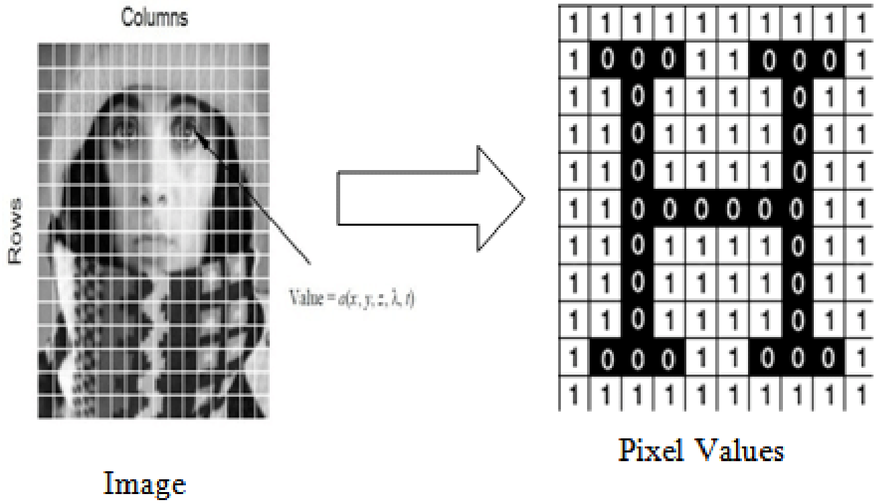
Digital Image

A digital image is a numeric representation of a two-dimensional image. It comprises picture elements, also known as pixels, each with finite, discrete quantities of numeric representation for its intensity or gray level.



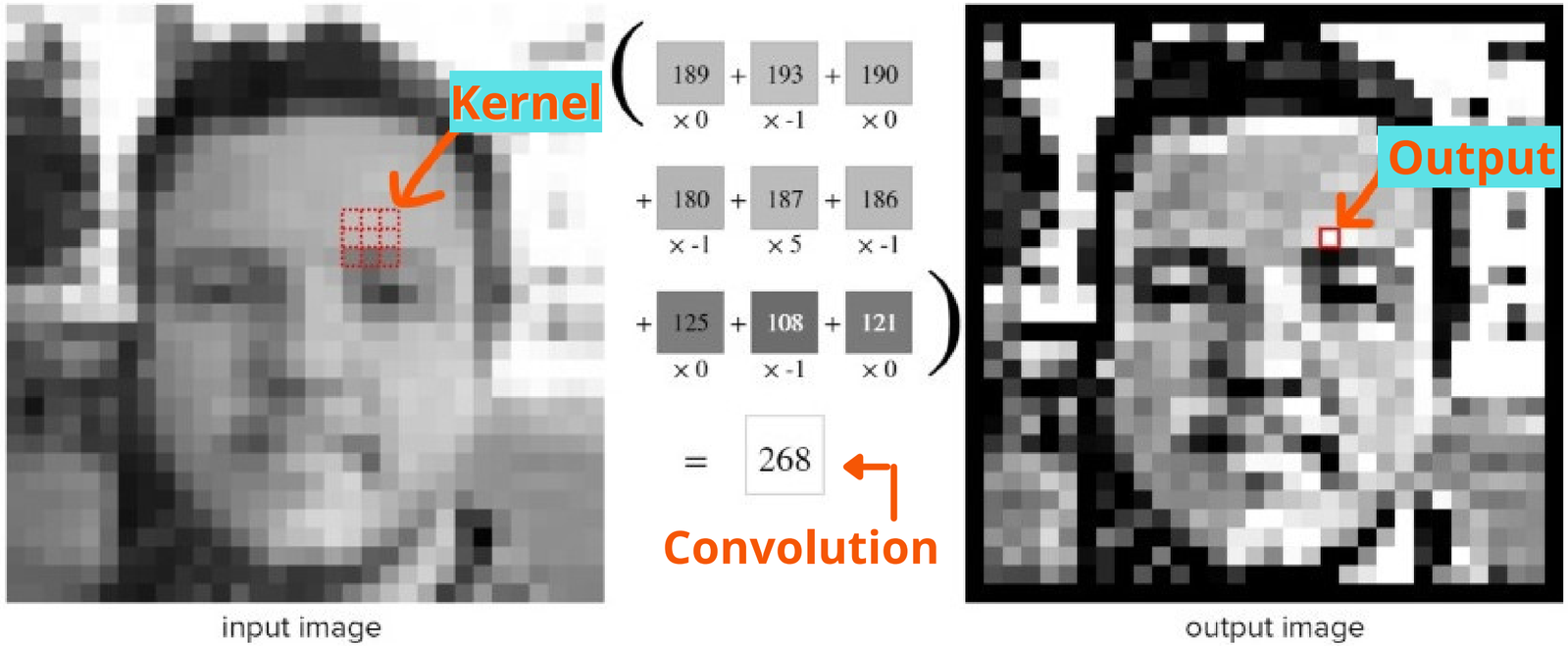
Pixel

A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. It is the basic logical unit in digital graphics i.e. it is the smallest controllable element of a picture represented on the screen.



Kernel

A kernel is a small matrix. The convolution of kernel and image produces different effects such as blurring, sharpening, embossing, and so on. A kernel works by operating on pixel values using straightforward mathematics to construct a new image.



Convolution

Convolution is the process of transforming an image by applying a kernel over each pixel and its local neighbors across the entire image.

There are 3 steps in convolution:

(1) Kernel Matrix is placed over each pixel of the image, each value of the Kernel is multiplied with the corresponding pixel it is over.
(2) The resulting multiplied values are summed and the resulting value is returned as the new value of the center pixel.
(3) This process is repeated across the entire image.

Noise is a random variation of image density, visible as grain in film and pixel level variations in digital images. Noise deforms the image and creates a problem in further processing such as object detection etc.

Generally, noise is introduced into the image during image transmission, acquisition, coding or processing steps.

What is image noise?

What is image denoising? Why do we need image denoising?

- Due to the environment, transmission channel, and other factors, images are inevitably contaminated by noise during acquisition, compression, and transmission, leading to distortion and loss of image information.

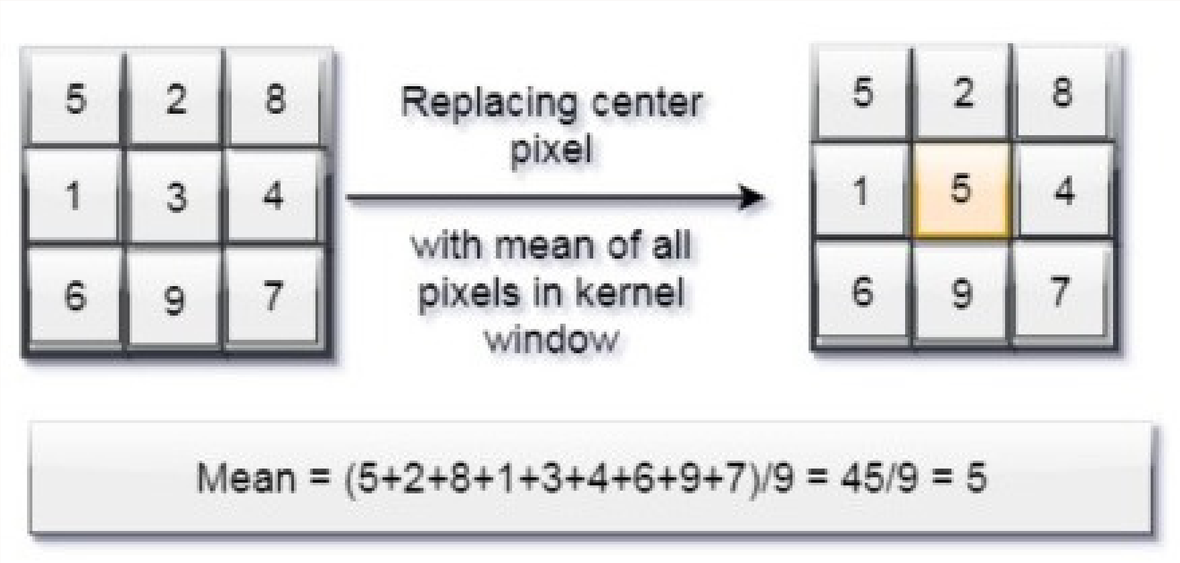
- Image denoising is a process to remove the noise from a noisy image, to restore the actual image.

- It helps reconstruct lost images from known noisy images and makes subsequent image processing tasks, such as video processing, image analysis, and tracking, easier.

Methods of Denoising

There are various methods for denoising. Here, we take a look at some of the most prominently used ones.

- *Mean Filter Method*
- *Median Filter Method*
- *Gaussian Filter Method*
- *Fractional Filter Method*

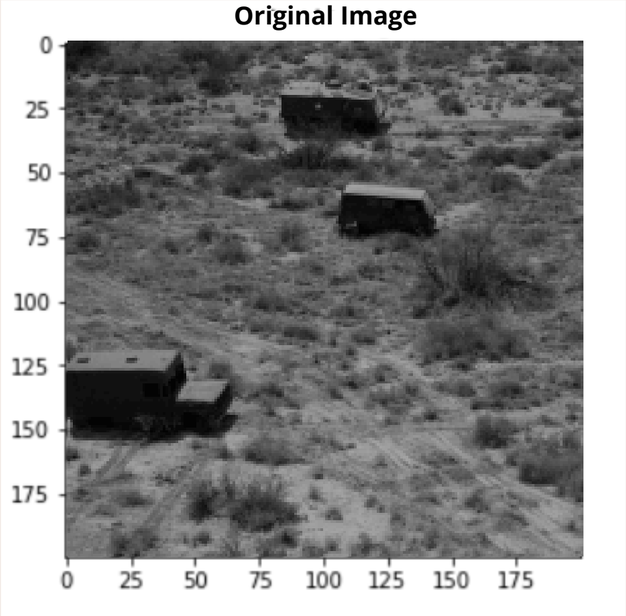


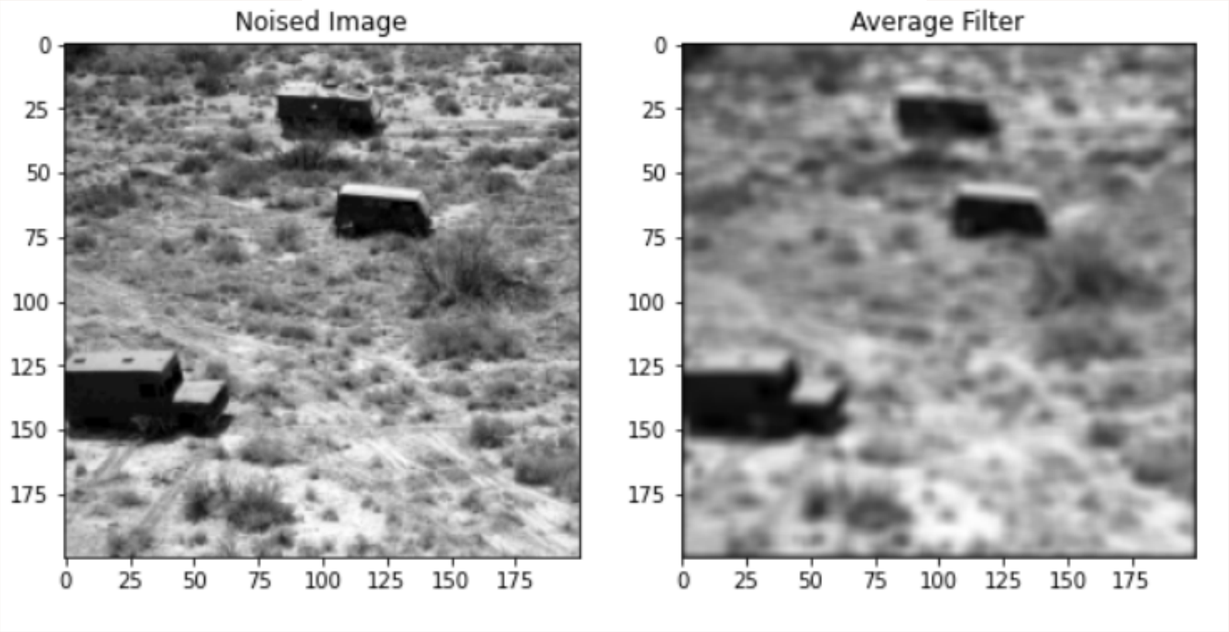
The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. The window, or kernel, is usually square but can be any shape. An example of mean filtering of a single 3x3 window of values is shown here.

Mean Filter Method

Python code for implementation of Mean Filter (3x3) Method

```
def MeanFilter(image,new_image, filter_size):
    row=new_image.shape[0]
    col=new_image.shape[1]
    # create an empty array with same size as input
image
    output = np.zeros(image.shape,float)
    # creat an empty variable
    result = 0
    # deal with filter size = 3x3
    for i in range(1, row+1):
        for j in range(1,col+1):
            for x in range(-1, 2):
                for y in range(-1, 2):
                    result = result + new_image[i+x, j+y]
            #print(result)
            output[i][j] =result / filter_size
            result = 0
    return output
```



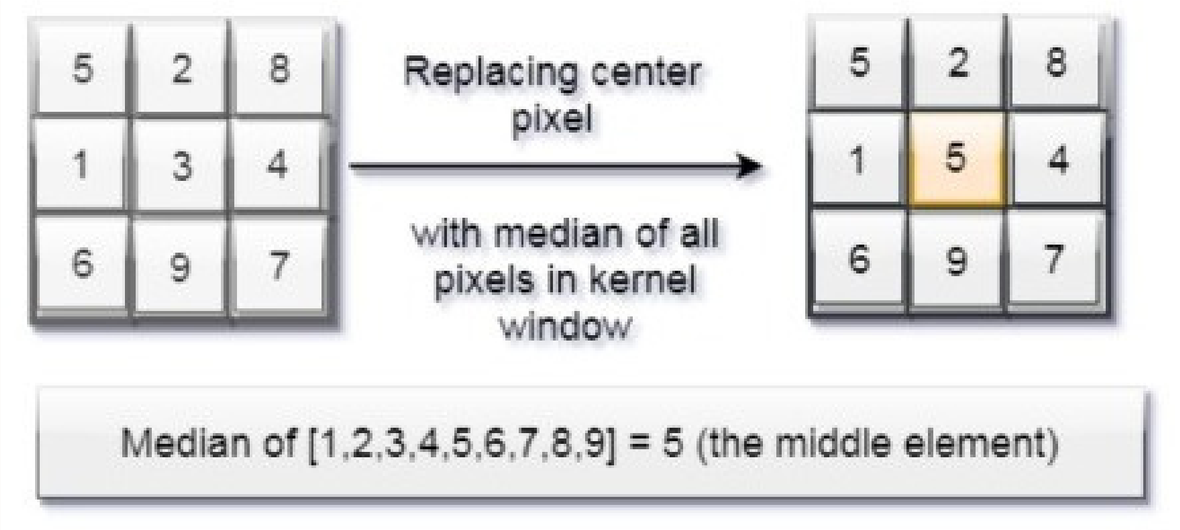


Advantage of mean filter:

It is easy to implement and used to remove the impulse noise.

Disadvantage of mean filter:

Some details of the image are not preserved on using the mean filter.



The median filter is a simple sliding-window spatial filter that replaces the center pixel value in the kernel window with the median of all the pixel values in that kernel window. The window, or kernel, is usually square but can be any shape. An example of mean filtering of a single 3x3 window of values is shown here.

Median Filter Method

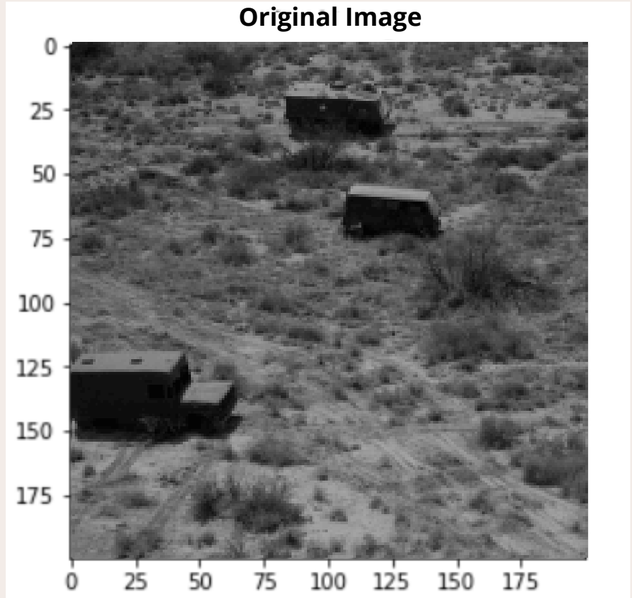
Python code for implementation of Median Filter (3x3) Method

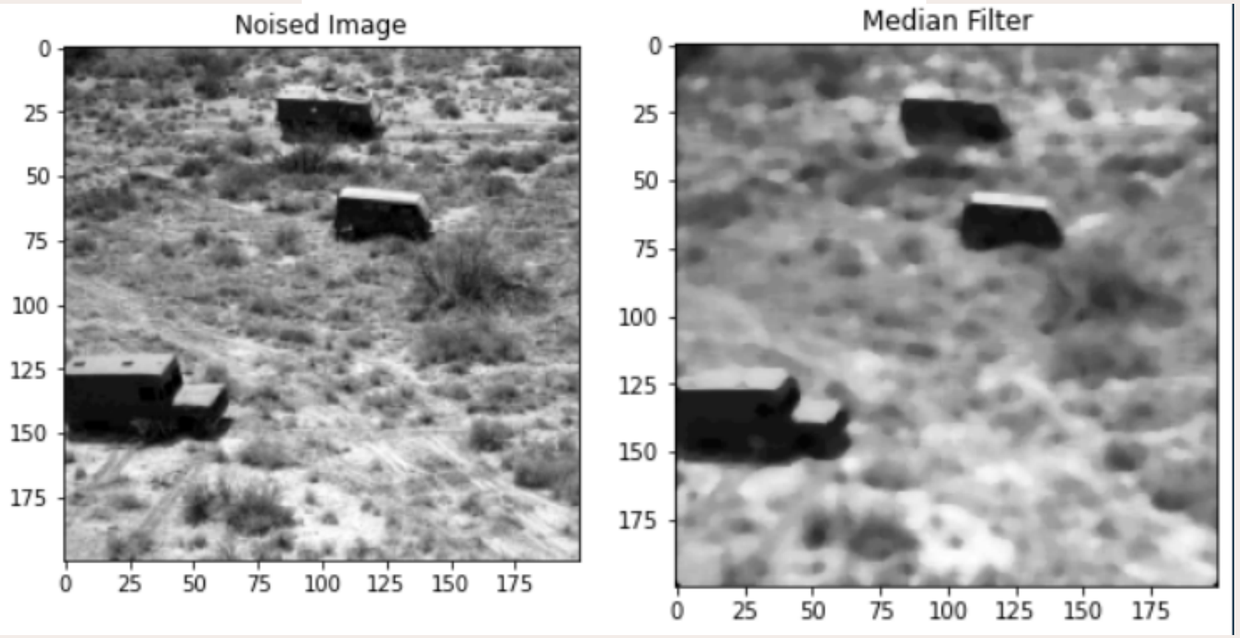
```
def MedianFilter(img,image,filter_size):
    # create the kernel array of filter as same size as filter_size
    filter_array = [image[0][0]] * filter_size
    row=image.shape[0]
    col=image.shape[1]
    # create an empty array with same size as input image
    output = np.zeros(img.shape,float)
    for j in range(1, image.shape[0]-1):
        for i in range(1, image.shape[1]-1):
            filter_array[0] = image[j-1, i-1]
            filter_array[1] = image[j, i-1]
            filter_array[2] = image[j+1, i-1]
            filter_array[3] = image[j-1, i]
            filter_array[4] = image[j, i]
            filter_array[5] = image[j+1, i]
            filter_array[6] = image[j-1, i+1]
            filter_array[7] = image[j, i+1]
            filter_array[8] = image[j+1, i+1]

            # sort the array
            filter_array.sort()

            # put the median number into output array
            output[j][i] = filter_array[4]

    return output
```





Advantage of median filter:

Median is a better representation of adjacent intensity values of pixels because a single unrepresentative pixel in the neighbourhood will not affect the median value significantly.

The median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason, the median filter is much better at preserving sharp edges.

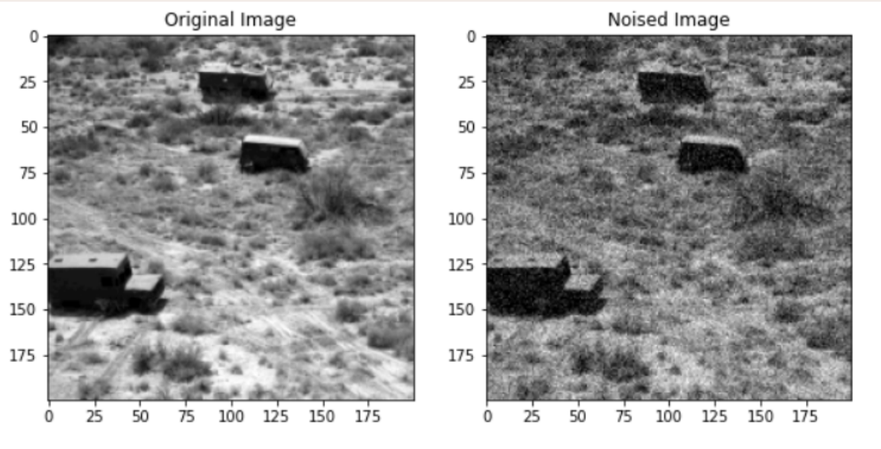
Gaussian Filter Method

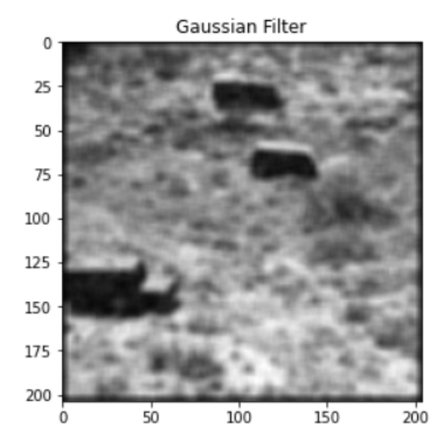
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

A 3×3 Gaussian Kernel Approximation(two-dimensional) with Standard Deviation = 1

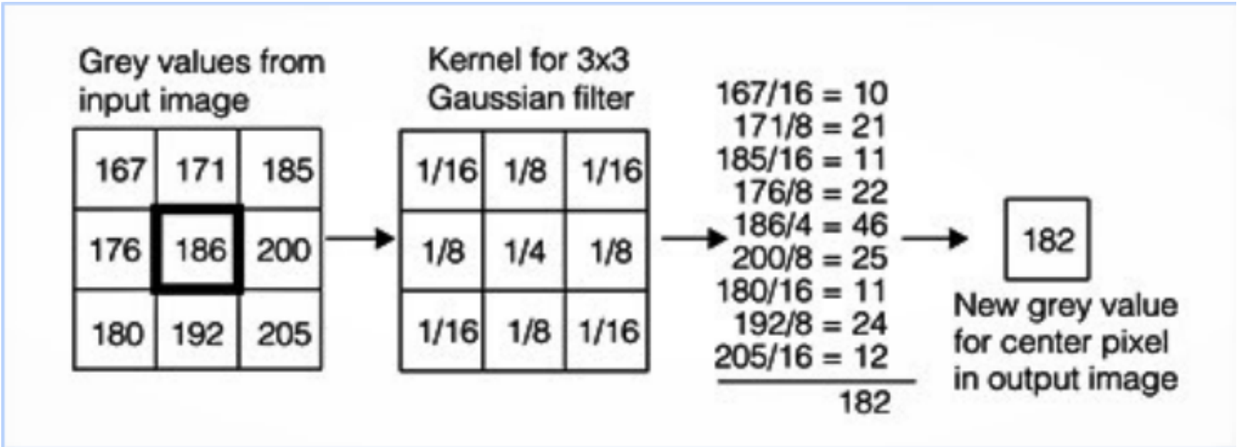
A Gaussian Filter is a low pass filter used for reducing noise (high-frequency components) and blurring regions of an image. The filter is implemented as an Odd sized Symmetric Kernel, which is passed through each pixel of the Region of Interest to get the desired effect. It helps blur and remove noise. It is very good for eliminating noise drawn from normal distribution.

Gaussian filter is attracting attention from computational biologists because it has been attributed with some amount of biological plausibility, e.g. some cells in the visual pathways of the brain often have an approximately Gaussian response.





The idea of gaussian smoothing is to use the 2-D distribution as point-spread function, as need to produce a discrete approximation to gaussian function.
Once the kernel is calculated, Gaussian smoothing can be performed using standard convolution methods. It is applied repeatedly for better denoising the image.



Fractional Filter Method

In recent years, many researchers have recognized the use of fractional calculus in image denoising problems.
The mask of the fractional filter is developed on the basis of a fractional operator named K-operator using the Riesz kernel.
Riemann-Liouville, Riesz Caputo operators are particular cases of K-operator.

The K-operator has two parameters p and q, with the help of which we can adjust the weight of the fractional mask according to the necessity to improve the performance of image denoising.
This is one of the advantages of the K-operator over the other known operators.
In the following examples, fractional-order for the proposed integral mask is chosen manually and varies with the noise variance.

Mathematical Analysis of the K-operator

- The K-operator for regular function ƒ for order α is defined as -

$$(K_P^{\alpha} f)(x) = p \int_a^x k_\alpha(x, \xi) f(\xi) d\xi + q \int_x^b k_\alpha(\xi, x) f(\xi) d\xi$$

where a < x < b, α is a positive real number and P = { a,x,b,p,q } is a parameter set. The kernel kα(x,ξ) may be dependent on α.

- The kernel is defined as -

$$k_\alpha(x, \xi) = \frac{1}{\gamma_n(\alpha)} \begin{cases} |x - \xi|^{(\alpha - n)} & \text{for } \alpha - n \neq 0, 2, 4, \ldots \\ |x - \xi|^{(\alpha - n)} \log\left(\frac{1}{|x - \xi|}\right) & \text{for } \alpha - n = 0, 2, 4, \ldots \end{cases}.$$

Here, |x - ξ| denotes Euclidean distance between x and ξ. The kernel mentioned above is the Riesz Kernel.

- Parameter $\gamma_n(\alpha)$ is defined as -

$$\gamma_n(\alpha) = \begin{cases} 2^{\alpha} \pi^{\frac{n}{2}} \left(\Gamma \frac{(n-\alpha)}{2}\right)^{-1} & \text{for } \alpha - n \neq 0, 2, 4, \ldots \\ (-1)^{\frac{(n-\alpha)}{2}} 2^{(\alpha-1)} \pi^{\frac{n}{2}} \Gamma\left(1 + \frac{\alpha - n}{2}\right) \Gamma \frac{\alpha}{2} & \text{for } \alpha - n = 0, 2, 4, \ldots \end{cases},$$

Design of Fractional Integral Mask

We apply the following steps to the K-operator equation -

- For $\alpha \in (0,2)$, it follows without loss of generality, taking n=1, a=0, using the symmetric property of convolution in the first part and replacing ξ by ξ+x in the second part of integral.

$$(K_P^{\alpha} f)(x) = \frac{p}{\gamma_1(\alpha)} \int_0^x |\xi|^{(\alpha-1)} f(x-\xi) d\xi + \frac{q}{\gamma_1(\alpha)} \int_0^{b-x} |\xi|^{(\alpha-1)} f(x+\xi) d\xi.$$

- Replace the integration by summation of products. For, sufficiently large N and M, divide the interval [0,x] and [0,b-x] into N and M equal parts.
- Approximating ƒ by average value on each sub-interval and replacing |ξ| by as ξ is non-negative in the given interval.
- In digital images, the number of pixels are finite and minimum length of step size is 1 pixel, calculating the integral parts,obtain 2-D image signal in x-axis.

The coefficients of the fractional mask are defined as:

$$c_{f_{-m}} = \frac{q}{2\alpha\gamma_1(\alpha)}\left(m^{\alpha} - (m-1)^{\alpha}\right)$$

$$c_{f_{-(m-1)}} = \frac{q}{2\alpha\gamma_1(\alpha)}\left(m^{\alpha} - (m-2)^{\alpha}\right)$$

$$c_{f_{-L}} = \frac{q}{2\alpha\gamma_1(\alpha)}\left((L+1)^{\alpha} - (L-1)^{\alpha}\right), \qquad 2 \le L \le (m-2)$$

$$c_{f_{-1}} = \frac{2^{\alpha} q}{2\alpha\gamma_1(\alpha)}$$

$$c_{f_0} = \frac{p+q}{2\alpha\gamma_1(\alpha)}$$

$$c_{f_1} = \frac{2^{\alpha} p}{2\alpha\gamma_1(\alpha)}$$

$$c_{f_k} = \frac{p}{2\alpha\gamma_1(\alpha)}\left((k+1)^{\alpha} - (k-1)^{\alpha}\right), \qquad 2 \le k \le (n-2)$$

$$c_{f_{(n-1)}} = \frac{p}{2\alpha\gamma_1(\alpha)}\left(n^{\alpha} - (n-2)^{\alpha}\right)$$

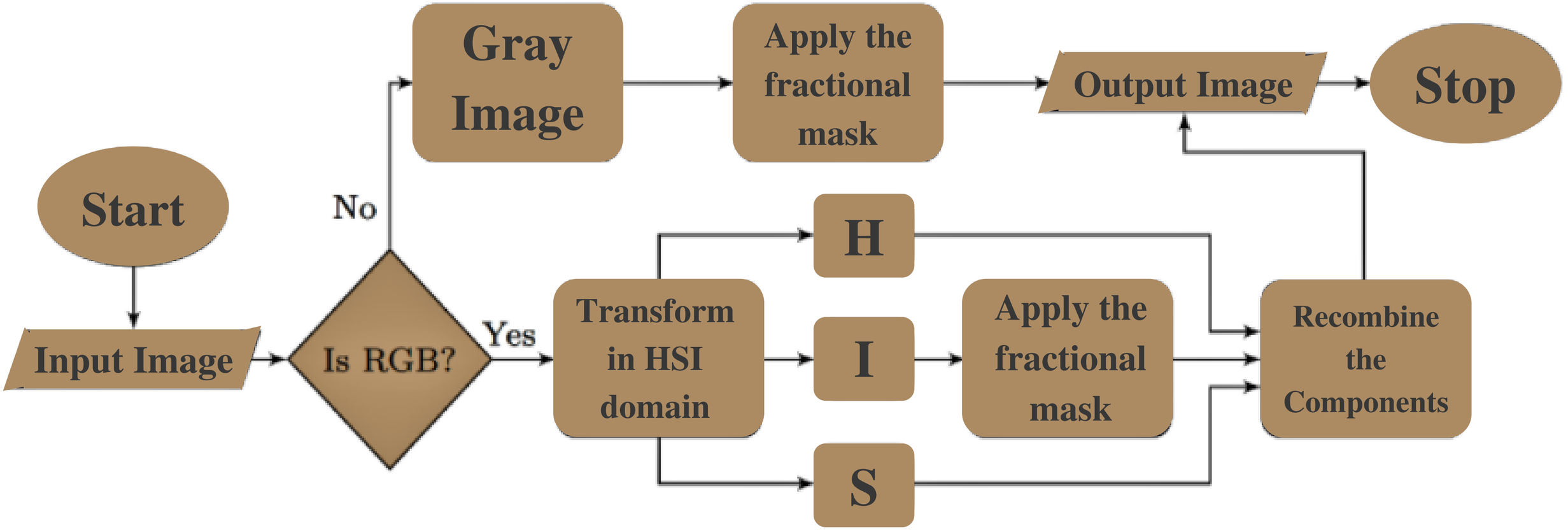
$$c_{f_n} = \frac{p}{2\alpha\gamma_1(\alpha)}\left(n^{\alpha} - (n-1)^{\alpha}\right).$$

The superposition of integral masks in all possible eight directions for q=0:

$$
\begin{array}{ccccccccc}
C_{fn} & 0 & \cdots & 0 & C_{fn} & 0 & \cdots & 0 & C_{fn} \\
0 & \ddots & \ddots & \vdots & \vdots & \vdots & \iddots & \iddots & 0 \\
\vdots & \ddots & C_{f2} & 0 & C_{f2} & 0 & C_{f2} & \iddots & \vdots \\
0 & \cdots & 0 & C_{f1} & C_{f1} & C_{f1} & 0 & \cdots & 0 \\
C_{fn} & \cdots & C_{f2} & C_{f1} & 8C_{f0} & C_{f1} & C_{f2} & \cdots & C_{fn} \\
0 & \cdots & 0 & C_{f1} & C_{f1} & C_{f1} & 0 & \cdots & 0 \\
\vdots & \iddots & C_{f2} & 0 & C_{f2} & 0 & C_{f2} & \ddots & \vdots \\
0 & \iddots & \iddots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\
C_{fn} & 0 & \cdots & 0 & C_{fn} & 0 & \cdots & 0 & C_{fn}
\end{array}
$$

By taking q=0, making all the mask coefficients of non-causal pixel to zero. Taking 5x5 mask. After normalizing and applying Convolution operation to gray-scale image. The RGB color image is denoised by transforming it in hue saturation intensity (HSI) color model and then filtering the I-component of the image.

Flowchart of the proposed algorithm:



Experimental Results and Analysis

- All the denoising methods including fractional filter method are applied on digital images to demonstrate the performance of proposed method.

- After application, the results of the proposed method are compared with the other spatial domain method.

- Experimental results are compared by visual perception and evaluation metrics such as Mean-Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR).

Mean-Square Error (MSE)

X’(x,y) is sampled image without noise, and X(x,y) is the denoised image.

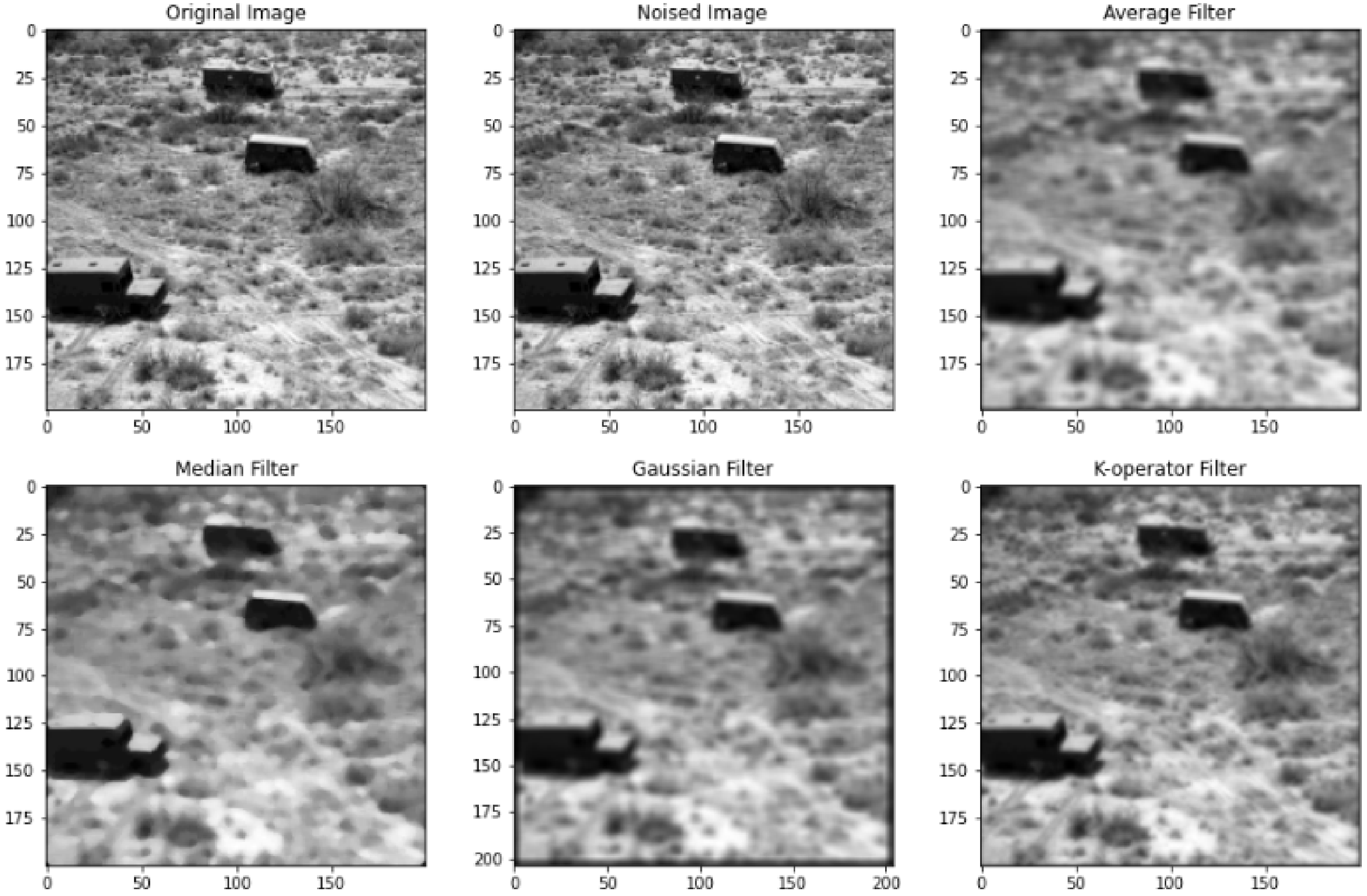
$$\mathrm{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [X'(x, y) - X(x, y)]^2$$

Peak Signal-to-Noise Ratio (PSNR)

X’(x,y) is sampled image without noise, and X(x,y) is the denoised image.

$$\mathrm{PSNR} = 20\log_{10}\left(\frac{2^8 - 1}{\sqrt{\mathrm{MSE}}}\right)$$

Parameters to compare image filter efficiency

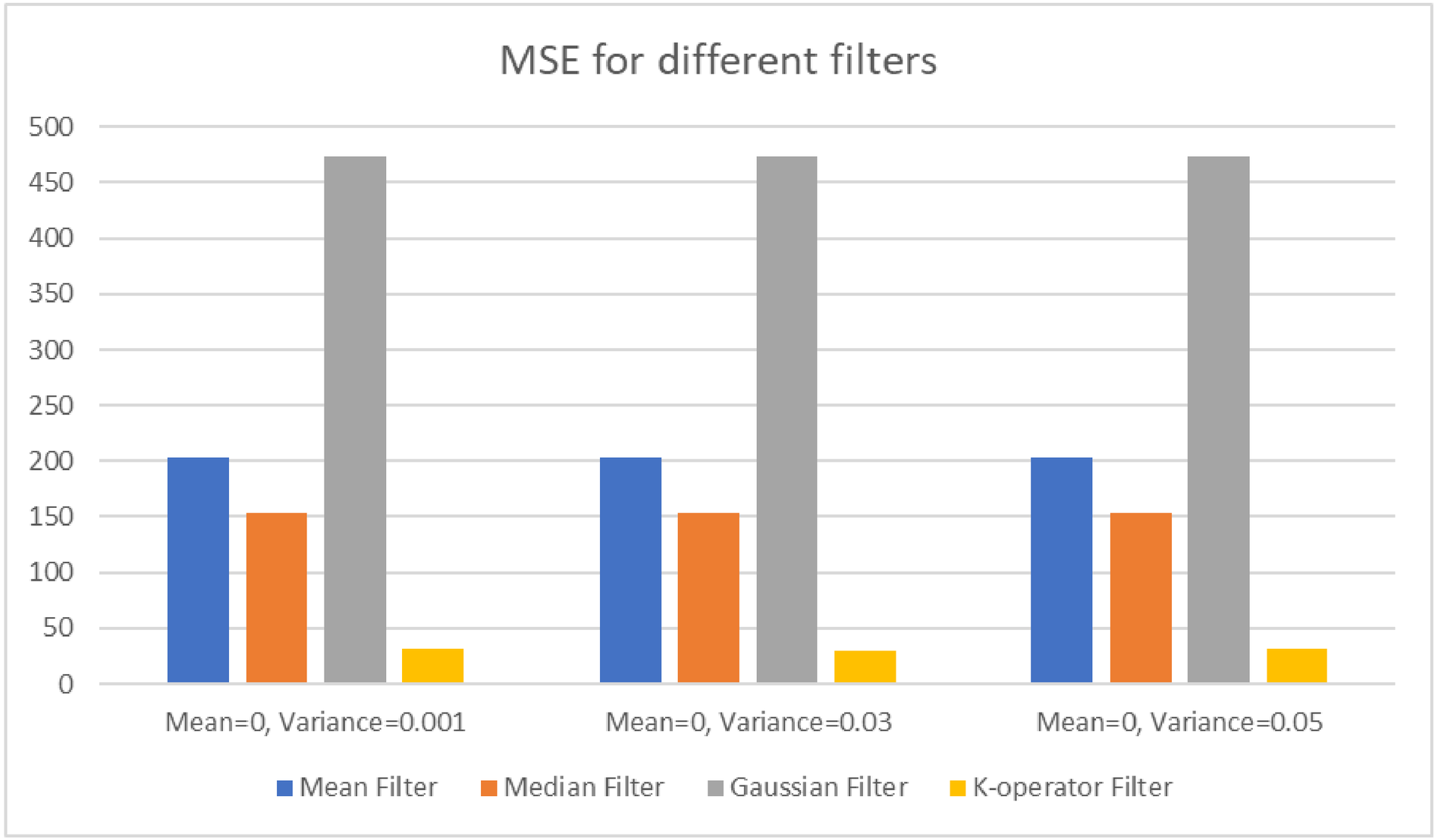


Gaussian noise of mean = 0 and var=0.03 has been added to the original image to obtain the noised image

MSE values for various filters (for previous example)

Variance	Alpha (f-order)	Mean Filter	Median Filter	Gaussian Filter	K- operator Filter (p=1.05, q=0)
0.001	**0.26**	**203.867**	**153.99**	**473.56**	**46.24**
0.03	**0.61**	**203.894**	**153.83**	**473.53**	**67.68**
0.05	**0.42**	**203.91**	**153.74**	**473.55**	**56.10**

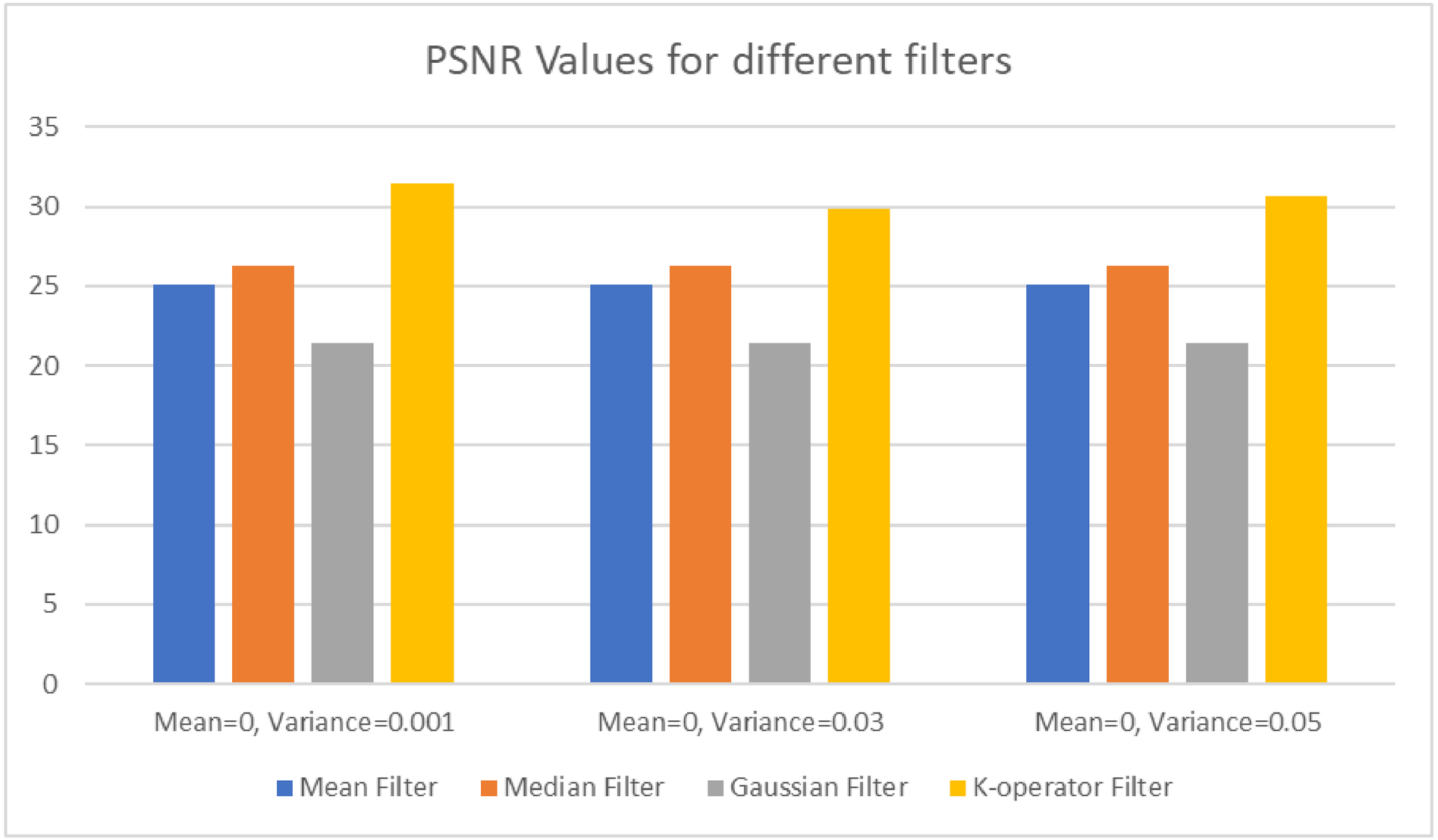
We can see that the MSE (mean square error) value is minimum in the case of the k-operator filter among all these filters i.e. K-operator filter yields a better image as compared to the other filters

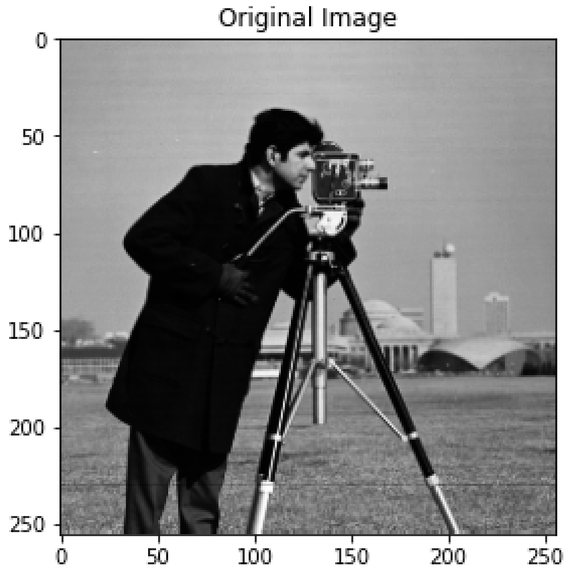

MSE for different filters
500
450
400
350
300
250
200
150
100
50
0
Mean=0, Variance=0.001
Mean=0, Variance=0.03
Mean=0, Variance=0.05
Mean Filter
Median Filter
Gaussian Filter
K-operator Filter

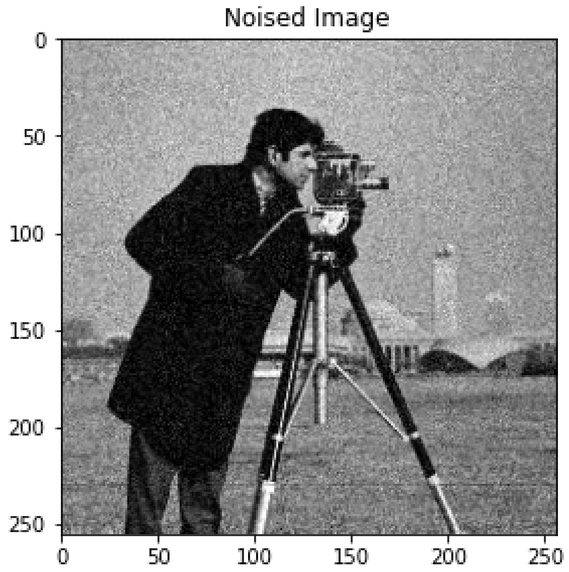
PSNR values for various filters (for previous example)

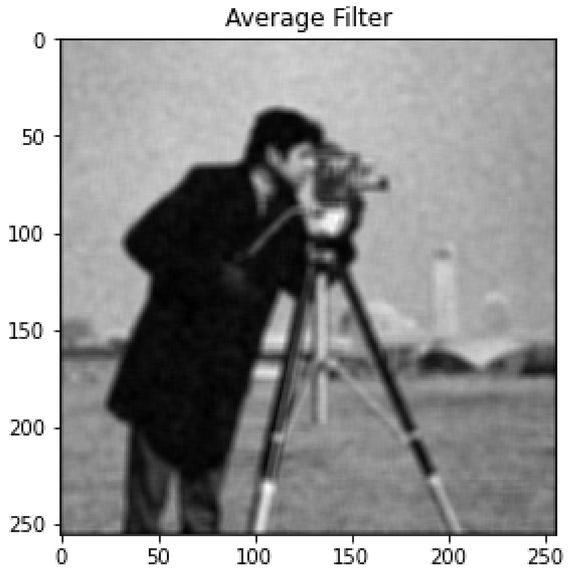
Variance	Alpha (f-order)	Mean Filter	Median Filter	Gaussian Filter	K- operator Filter (p=1.05, q=0)
0.001	**0.26**	**25.04**	**26.256**	**21.38**	**31.48**
0.03	**0.61**	**25.03**	**26.26**	**21.38**	**29.83**
0.05	**0.42**	**25.04**	**26.26**	**21.38**	**30.64**

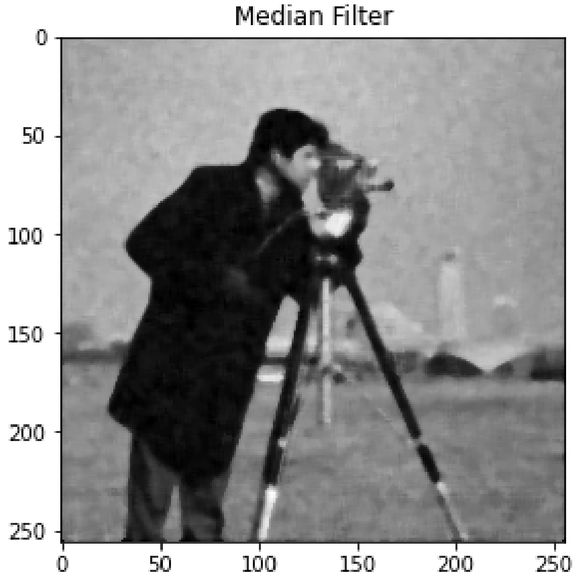
We can see that the PSNR (peak signal-to-noise ratio) value is maximum in the case of the k-operator filter among all these filters i.e. K-operator filter yields a better image as compared to the other filters

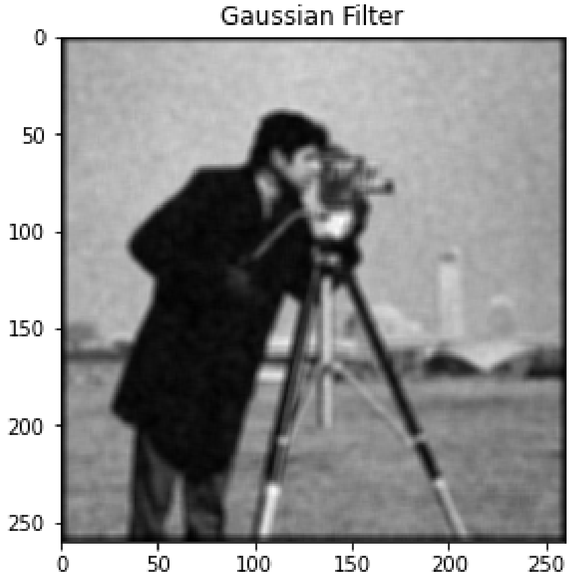

PSNR Values for different filters
35
30
25
20
15
10
5
0
Mean=0, Variance=0.001
Mean=0, Variance=0.03
Mean=0, Variance=0.05
Mean Filter
Median Filter
Gaussian Filter
K-operator Filter

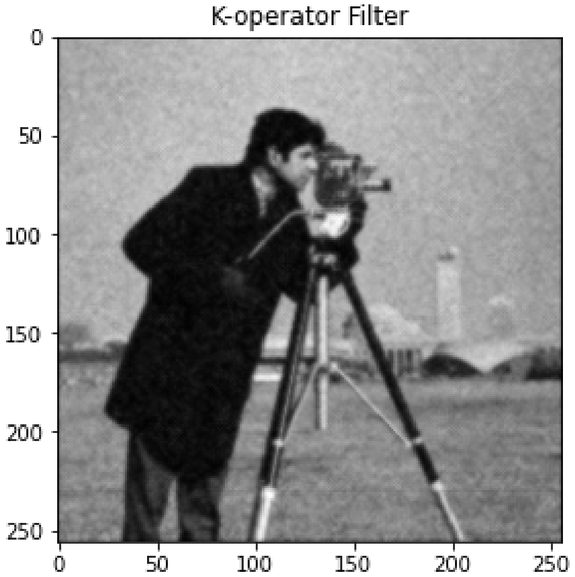










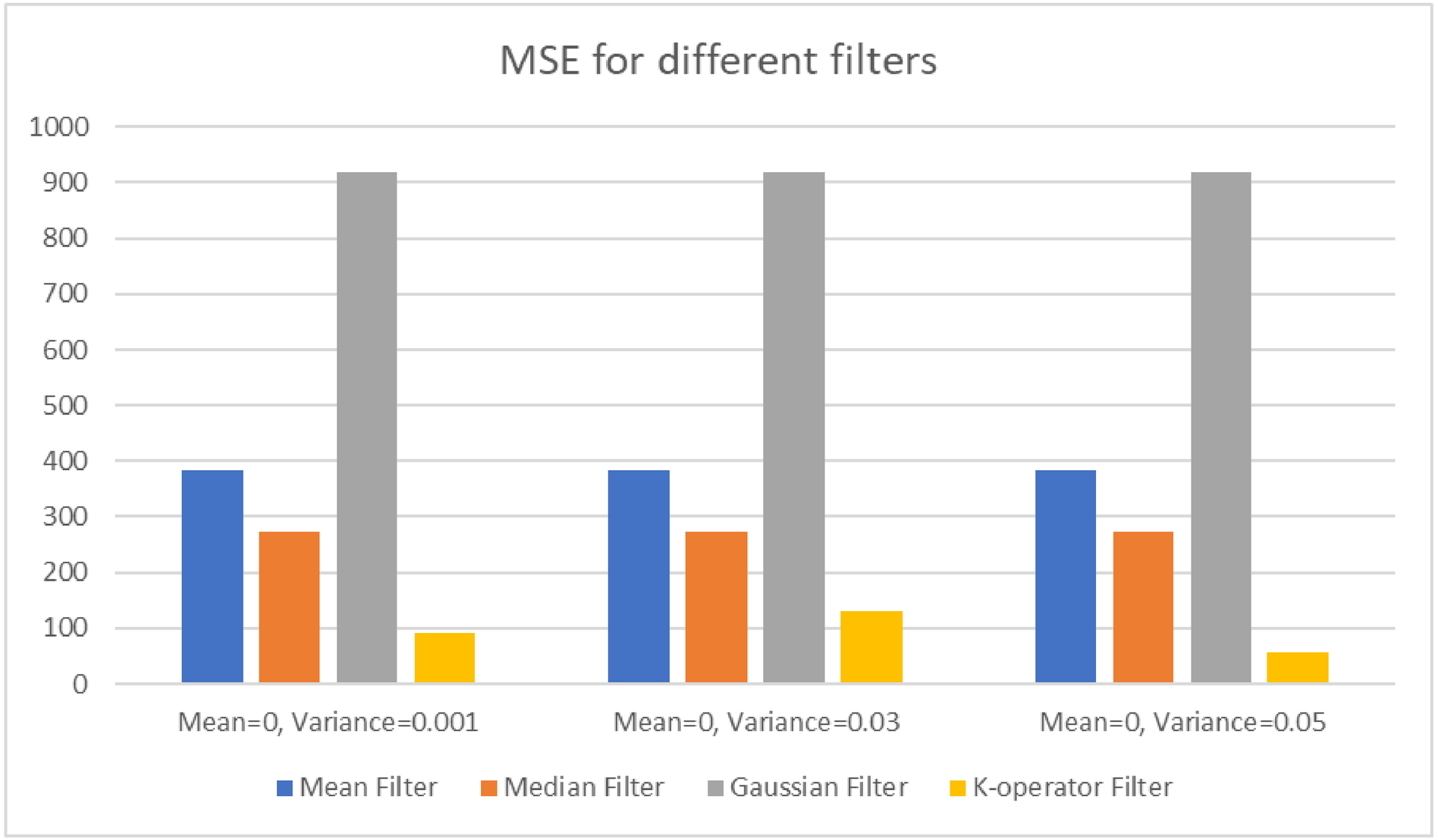


Gaussian noise of mean = 0 and var=0.02 has been added to the original image to obtain the noised image

MSE values for various filters (for above image)

Variance	Alpha (f-order)	Mean Filter	Median Filter	Gaussian Filter	K- operator Filter (p=1.05, q=0)
0.001	**0.26**	**383.65**	**271.93**	**918.33**	**90.81**
0.03	**0.61**	**383.71**	**271.86**	**918.33**	**131.21**
0.05	**0.42**	**383.73**	**271.85**	**918.27**	**109.42**

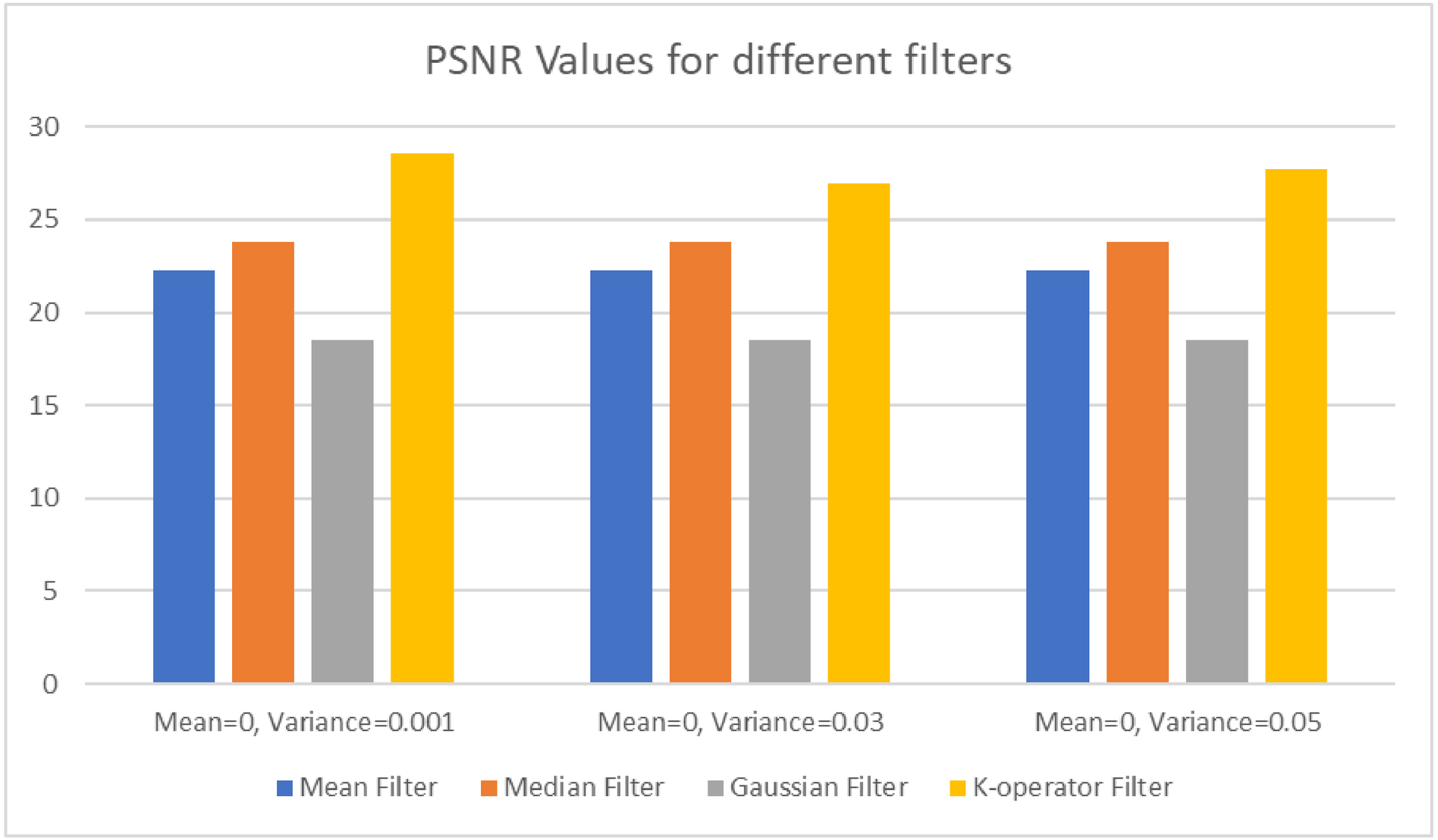
We can see that the MSE (mean square error) value is minimum in the case of the k-operator filter among all these filters i.e. K-operator filter yields a better image as compared to the other filters


MSE for different filters
1000
900
800
700
600
500
400
300
200
100
0
Mean=0, Variance=0.001
Mean=0, Variance=0.03
Mean=0, Variance=0.05
Mean Filter
Median Filter
Gaussian Filter
K-operator Filter

PSNR values for various filters (for previous example)

Variance	Alpha (f-order)	Mean Filter	Median Filter	Gaussian Filter	K- operator Filter (p=1.05, q=0)
0.001	**0.26**	**22.291**	**28.78**	**18.50**	**28.55**
0.03	**0.61**	**22.29**	**23.79**	**18.50**	**26.95**
0.05	**0.42**	**22.29**	**23.78**	**18.50**	**27.74**

We can see that the PSNR (peak signal-to-noise ratio) value is maximum in the case of the k-operator filter among all these filters i.e. K-operator filter yields a better image as compared to the other filters


PSNR Values for different filters
30
25
20
15
10
5
0
Mean=0, Variance=0.001
Mean=0, Variance=0.03
Mean=0, Variance=0.05
Mean Filter
Median Filter
Gaussian Filter
K-operator Filter

Conclusion

The introduced Fractional mask is designed using the K-operator with Riesz kernel. The proposed algorithm is demonstrated using various parameters like MSE, PSNR, and visual perception. It is compared with other traditional image processing filters like the Mean filter, Median filter, Gaussian filter.

Based on numerical results and visual perception, we conclude that the average performance of the proposed algorithm is better than the other methods. It can be implemented in real-time, and it removes noise and preserves texture in the images.

Future Scope

We are thinking of an algorithm that chooses the best value of alpha for better results. This can be done by making a machine learning model.

The algorithm can be applied for real-life applications like signal processing, biological implementation, etc.

Further studies on fractional calculus will help improve the performance of the method.