INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Electronics & Eletrical Communication Engineering

Vision and Intelligent Systems

EC69211 – Image and Video Processing Laboratory

Experiment – 4

Spatial Filtering

Submitted by:

Bbiswabasu Roy (19EC39007)

Jothi Prakash (19EC39023)

CONTENTS

Sl No	Content	Pg No
1	Cover Page	1
2	Contents	2
3	Objective	3
4	Theory	3-4
5	Algorithms	5
6	Results	5-13
7	Discussion	13-14

Objective:

1. Applying various spatial filters to noisy images as well as images without noise and understanding their effect
2. Implementing function to unblur an image which was previously blurred using Gaussian filter

Theory:

Mean Spatial Filter

The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter.

1	1	1
1	1	1
1	1	1

x 1/9

Weight w

Median Spatial Filter

the median filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighboring pixel values, it replaces it with the median of those values

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

Prewitt Filter

The Prewitt operator is one of the first algorithms for edge detection by gradient transform. The Prewitt operator detects image edges by convolution with two filter masks. One for horizontal and one for vertical direction.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * I; \quad Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * I$$

Prewitt operator with 3×3 masks

Laplacian Filter

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection (see zero crossing edge detectors).

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Sobel Kernels

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

Horizontal

-1	0	+1
-2	0	+2
-1	0	+1

Gx

Vertical

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Gaussian Blur

The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Laplacian of Gaussian

Laplacian filters are derivative filters used to find areas of rapid change (edges) in images. Since derivative filters are very sensitive to noise, it is common to smooth the image (e.g., using a Gaussian filter) before applying the Laplacian. This two-step process is called the Laplacian of Gaussian (LoG) operation.

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Algorithm:

Following algorithm is used to apply the spatial filters

1. Take the input path of the images from the user
2. Take as input the filter that needs to be applied
3. Based on the chosen filters, take other required inputs such as variance or orientation of filter from the user.
4. Generate the kernel based on the chosen filter
5. Perform the required convolution of the kernels with the images by flipping the kernel, taking elementwise product of image pixels with kernel elements and summing them up
6. Store the output images in the desired folder location

Following algorithm is used to apply Gaussian Unblur filter

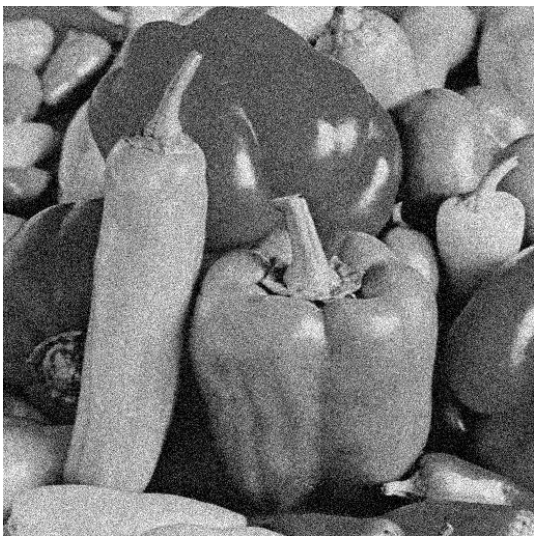
1. Take the input image from the user and let it I_o
2. Apply the gaussian filter convolution on it $A_k = I_k * G\sigma$
3. Perform pixel by pixel division on the resultant Image $B_k = I_o / A_k$
4. Again perform convolution as $C_k = B_k * G\sigma$
5. Finally perform pixel by pixel multiplication as $I_{k+1} = I_k * C_k$
6. Repeat the above process till I_k and I_{k+1} converge with an small difference among them

Results:

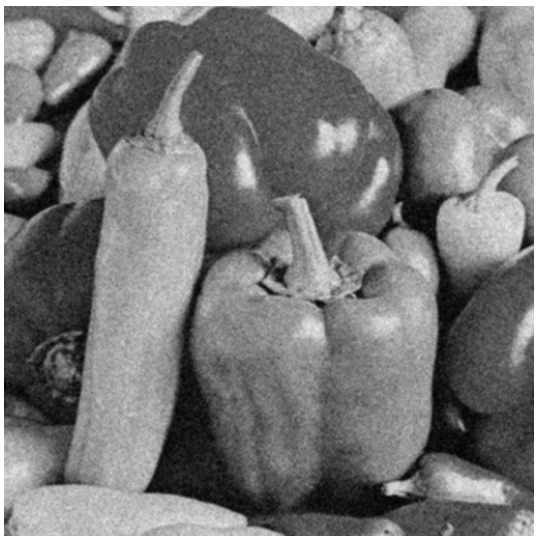
Mean filter was found to have smoothing effect on all given input images. It could eliminate Gaussian noise in the images with small amount of noise but for the images with high amount of Gaussian noise (ex – *Camerman_Gaussian_0.05.jpg*), it could not perform appreciable denoising. Also, mean filter didn't give good result on images with salt & pepper noise.

Camerman_Gaussian_0.005.jpg

Camerman_Gaussian_0.005.jpg after passing through mean filter



Pepper_Gaussian_0.01.jpg



Pepper_Gaussian_0.01.jpg after passing through mean filter

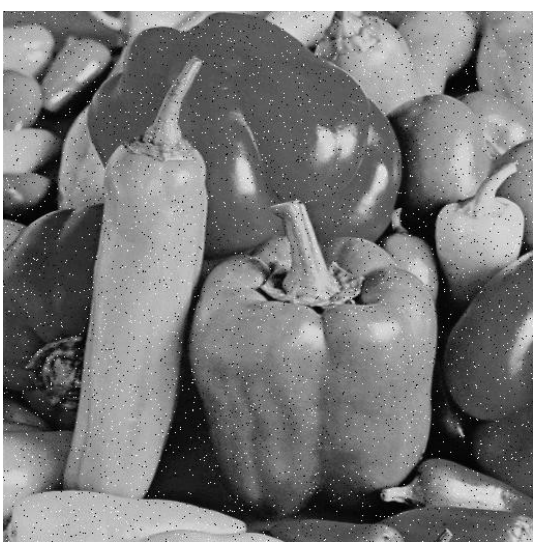
Median filter has the property that if it encounters a very high intensity pixel (white) or very low intensity pixel (black) in the neighborhood, then it will reject those pixels as it takes pixel with median value. This makes it very appropriate for filtering images contaminated with salt & pepper kind of noise. In such cases, the output image neither contains the *salt* not does it contain the *pepper* as they generally do not occur as the median value in the surrounding. However if the neighborhood contains too many white (or black) corrupted pixels, then median filter might fail to denoise them.

Cameraman_Salt&Pepper_0.02.jpg

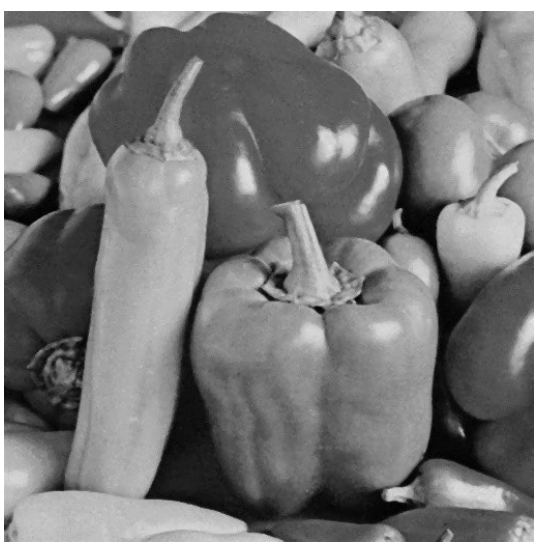
Cameraman_Salt&Pepper_0.02.jpg after passing through median filter

Cameraman_Salt&Pepper_0.08.jpg

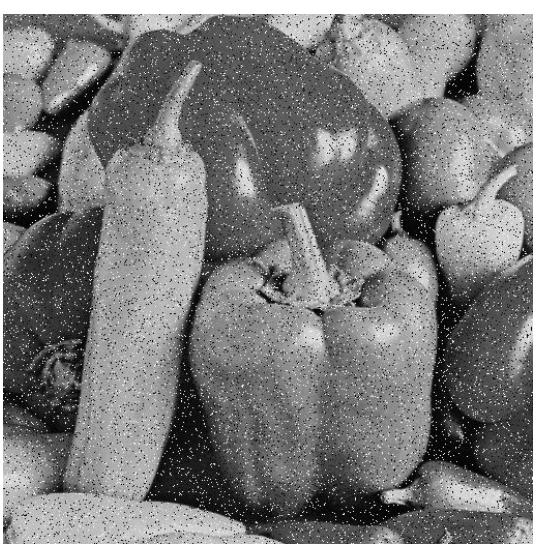
Cameraman_Salt&Pepper_0.08.jpg after passing through median filter



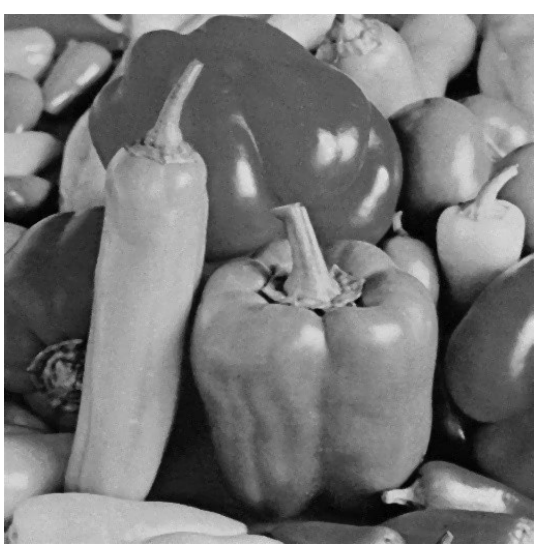
Pepper_Salt&Pepper_0.02.jpg



Pepper_Salt&Pepper_0.02.jpg after passing through median filter



Pepper_Salt&Pepper_0.08.jpg



Pepper_Salt&Pepper_0.08.jpg after passing through median filter

Prewitt filter was found to detect edges in the input images. In the generated output, the edge pixels had high intensity while rest of them had very low intensity. When passed through horizontal prewitt filter, the horizontal edgdes were more prominent while when passed through vertical prewitt filter, vertical edges were prominent.

pirate.jpg after passing through prewitt-H filter

mandril_gray.jpg after passing through prewitt-H filter

pirate.jpg after passing through prewitt-V filter

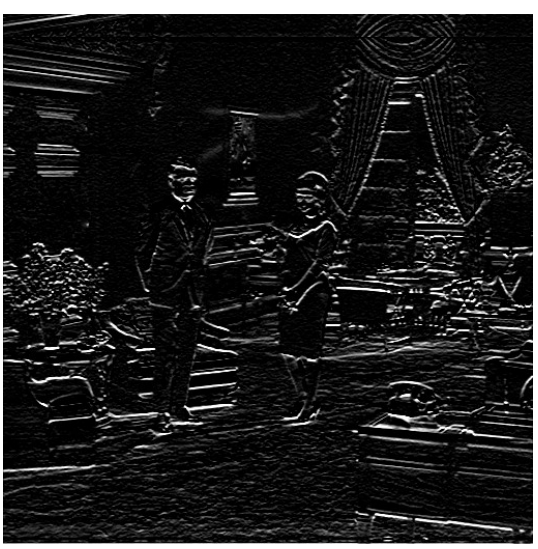
mandril_gray.jpg after passing through prewitt-V filter

Unlike the prewitt filter, Laplacian filter used only a single kernel which could detect edges along multiple directions.

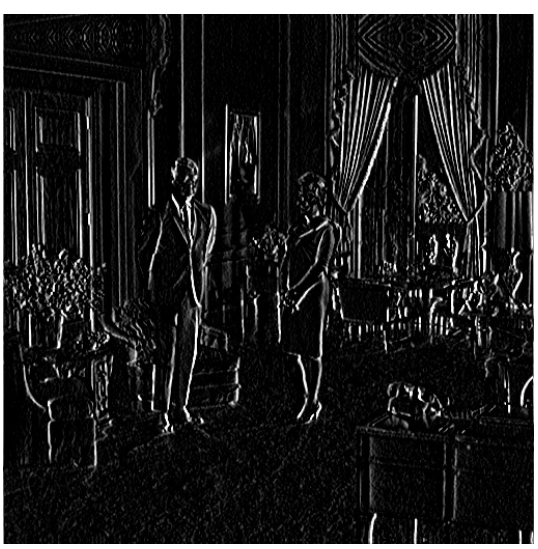
lena_gray_512.jpg after passing through laplacian filter

jetplane.jpg after passing through laplacian filter

Sobel filter was applied along horizontal, vertical and diagonal directions to obtain edges along respective directions. Unlike prewitt filter, it gives higher weightage to central pixel. The following output images for *livingroom.jpg* shows how each orientation of sobel filter extracts edges in respective orientations.

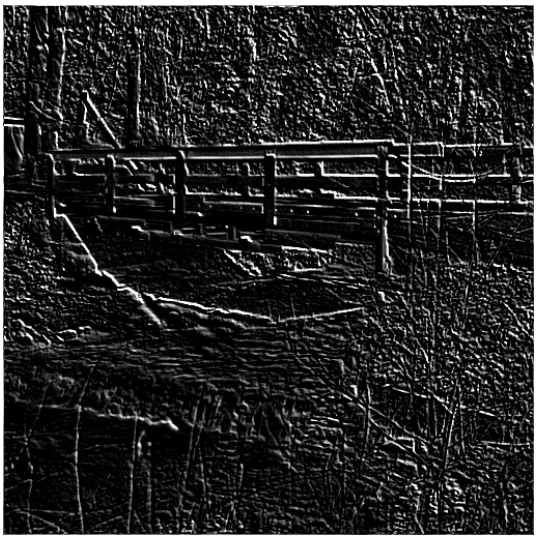


livingroom.jpg after passing through sobel-H filter



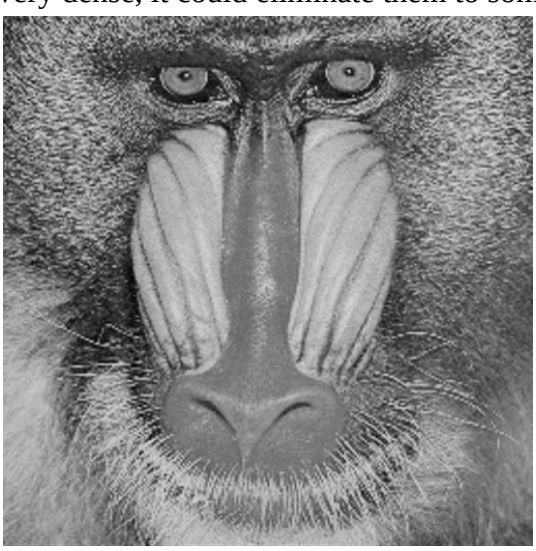
livingroom.jpg after passing through sobel-V filter

livingroom.jpg after passing through sobel-D filter

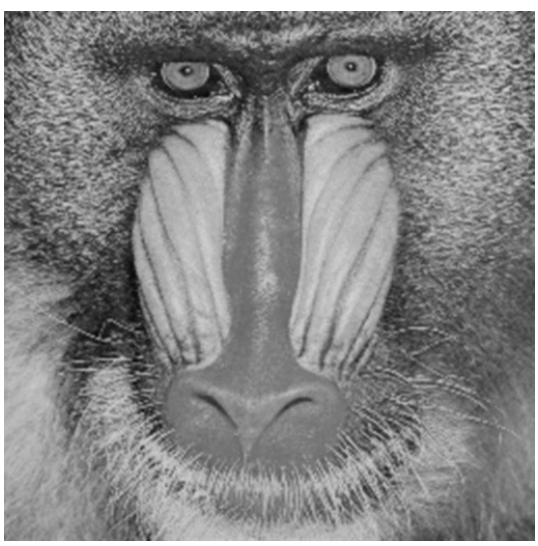


walkbridge.jpg after passing through sobel-D filter

Gaussian filter gave smoothing/blurring effect on given input images. However, it was not very effective in eliminating noise from images. In fact mean filter was found to perform better in eliminating noise. The main reason behind this is that Gaussian filter applies higher weightage to central pixel and has a tendency to maintain its properties. On the other hand, mean filter applies same weightage to all its surrounding pixels and if noisy pixels are not very dense, it could eliminate them to some extent.



original mandril_gray.jpg

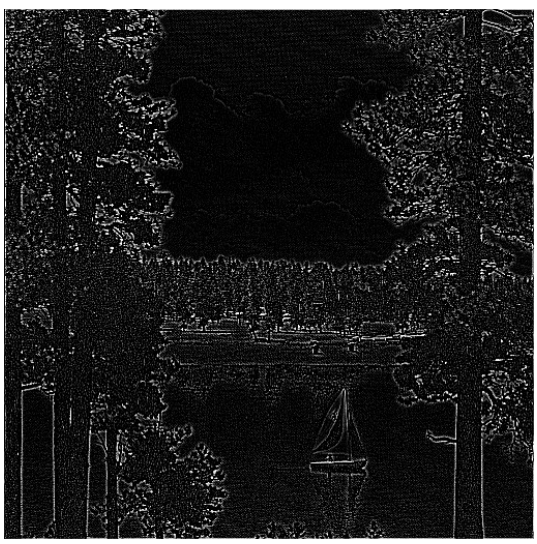


mandril_gray.jpg after passing through Gaussian filter

original lena_gray_512.jpg

lena_gray_512.jpg after passing through Gaussian filter

Laplacian of Gaussian filter essentially consisted of simultaneous smoothing and edge detection so that edges detected by the filter becomes somewhat tolerant to noise. The differences are clearly depicted below.



lake.jpg after passing through laplacian filter

lake.jpg after passing through laplacian of Gaussian filter

walkbridge.jpg after passing through laplacian filter

walkbridge.jpg after passing through laplacian of Gaussian filter

c*amerman_Gaussian_0.05*.jpg was passed through Gaussian blur filter and then through Gaussian unblur function c*amerman_Gaussian_0.05.jpg* for several iterations as long as the mean absolute change in intensity levels was >0.1. It went for 60 iterations and the output images were found to become gradually sharper with each iteration.

Blurred camerman_gaussian_0.05.jpg

unblurred camerman_gaussian_0.05.jpg after 12 iterations

unblurred camerman_gaussian_0.05.jpg after 36 iterations

unblurred camerman_gaussian_0.05.jpg after 48 iterations

unblurred camerman_gaussian_0.05.jpg after 60 iterations

original camerman_gaussian_0.05.jpg

Discussion:

- The blur or smoothing filters can also be seen as Lowpass filters as they reject abrupt changes in intensity level of images. Similarly, the edge detector filters can be seen as Highpass filters as they allow only abrupt changes (or high frequency components) to pass through them.
- If the size of the kernel is very small, it will not capture sufficient information about surroundings and hence the effect of filter will not be very prominent. On the other hand, if the size of the kernel is very large, the output image pixels will lose local properties and might include effects from far away pixels. Hence, we must choose some optimal size of kernel.
- At the edges and at the corner of the image, the kernel will not be able to totally fit on the image and these pixels need to be handled properly. There can be various ways to tackle this

such as by padding the original image with some constant value, paddin with same values as those of edge pixels, cropping the kernel at the edges and applying appropraite normalization after cropping (used by us), getting an edge-cropped output image by not extending kernel beyond edges.

- If we want to detect edges in an image having significant background noise, we must apply Laplacian of Gaussian as it was found to be more tolerant to noise as compared to other filters.
- The standard deviation in the Gaussian kernel determines the amount of smoothing to be applied. For example, if its value is set to zero, we won't get any smoothing while if its values was very high, we would get a filter very close to mean filter.
- While unblurring the blurred image, we must iterate as long as the change in the pixel data is sufficiently small. Different metrics could be used to compute the change including root mean square change, mean absolute change, max absolute change and so on. Depending on the metrics chosen and the limit set for the stopping criteria, number of iterations will vary as a consequence of which the unblurring effect will also vary.