



Fall 2018

COSC 6360: Digital Image Processing
Final Report

Spatial Filtering



Black Panther

Amruta Ghodke - 1540554

Ashna Shah - 1638044

Kishansingh Rajput - 1784635

Rahul Sawant - 1632656

Shiyi Liu - 1617482

Index

1.	Abstract.....	3
2.	UI Design Implementation.....	3
3.	Image Smoothing.....	7
3.1.	Linear filter.....	7
3.1.1.	Mean Filter (Average Filter)	
3.2.	Non-linear filter.....	9
3.2.1.	Median filter	
4.	Image Sharpening.....	10
4.1.	First order derivative.....	
10		
4.1.1.	Prewitt operator	
4.1.2.	Roberts operator	
4.1.3.	Sobel operator	
4.1.4.	Custom Sobel operator	
4.2.	Laplacian filter.....	
16		
4.3.	Unsharp masking and high boosting.....	
20		
5.	References.....	
24		

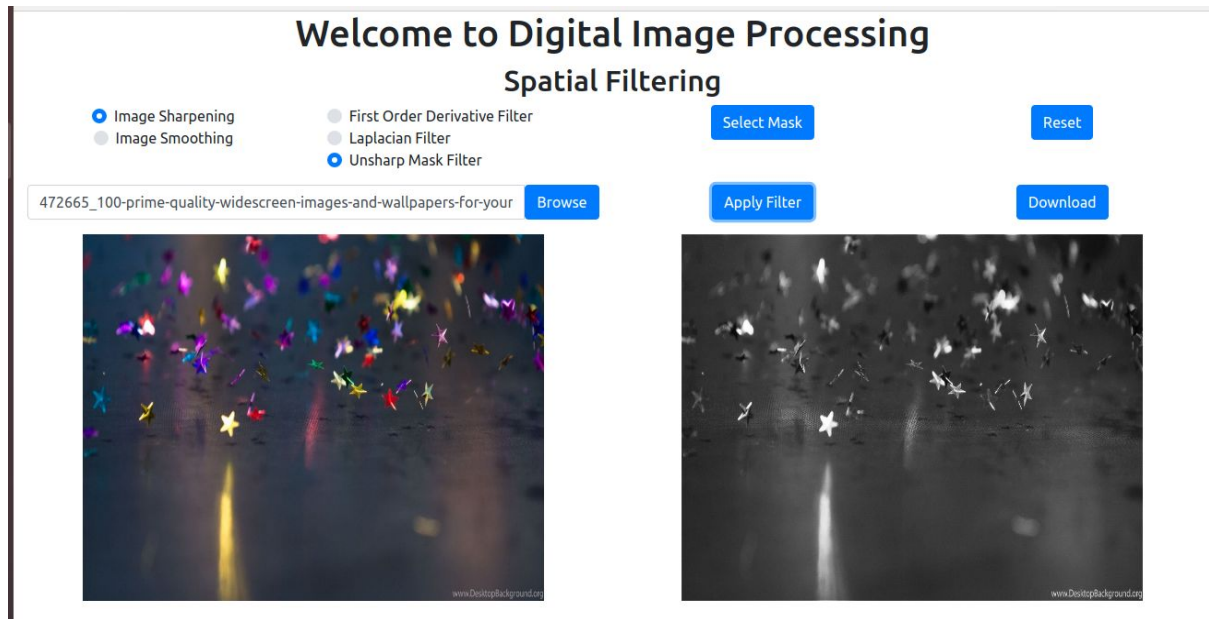
1. Abstract

By studying “spatial filtering”, we tried to apply different methods to grayscale image to perform operation likes - smoothing, sharpening. We implemented spatial filtering APIs and integrated with front end developed as web interface. This allows user to select type of operation user wants to perform and type of filters (popularly used filters or custom filters based on type of operation). We provided flexibility to the user to specify parameters required for filtering. At the end, user can see original and filtered image and compare them. It helps user to analyze and visualize changes. We explored many areas while working on this project. Observation and Results provided detailed understanding of the topics and helped to explore topics in the detail.

2. UI Design Implementation

- UI Design is implemented using Flask and React js.
- It is a user friendly UI and provides option to upload, download and compare images.
- User can also input mask for various filter
- Most of the filters gives output within **10 seconds** for **512 x 512** images. After that, as the resolution changes it takes more time to get output.
- On the UI screen, there are steps to follow to get the desired filtered image:
 - Step 1: Select Image Smoothing / Image Sharpening
 - Step 2: Accordingly select filters
 - Smoothing - Linear, Non-Linear
 - Sharpening - First order derivative, Laplacian, Unsharp masking and boost
 - Step 3: Select mask for the filter. There are some suggested masks for each filter and there is provision to enter custom filter for user in various sizes as 3*3, 5*5 and 7*7
 - Step 4: Browse the image and upload it
 - Step 5: Click Apply filter to see desired output image
 - Step 6: You can also download the filtered(output) image
 - Also, you can redo the selection by clicking the reset button

Screenshots of UI :



Welcome to Digital Image Processing

Spatial Filtering

- ☒ Image Sharpening
- ☐ Image Smoothing

- ☒ First Order Derivative Filter
- ☐ Laplacian Filter
- ☐ Unsharp Mask Filter

Select Mask

Reset

fall-autumn-red-season.jpg

Browse

Apply Filter

Download



Welcome to Digital Image Processing

Spatial Filtering

- ☒ Image Sharpening
- ☐ Image Smoothing

- ☐ First Order Derivative Filter
- ☒ Laplacian Filter
- ☐ Unsharp Mask Filter

Select Mask

Reset

283c7de64b104fd81fe2175cc9bdc8c0-cellphone-wallpapers-iphone-b-

Browse

Apply Filter

Download



Welcome to Digital Image Processing

Spatial Filtering

☐ Image Sharpening

☒ Image Smoothing

☒ Linear Filter

☐ Non-Linear Median Filter


Select Mask


Reset

Browse

Apply Filter

Download





Welcome to Digital Image Processing

Spatial Filtering

☐ Image Sharpening

☒ Image Smoothing

☐ Linear Filter

☒ Non-Linear Median Filter


Select Mask


Reset

Browse

Apply Filter

Download





3. Image Smoothing

Image smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing tasks, such as removal of small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with a linear filter and also by non-linear filtering.

3.1 Image Smoothing Using Linear filter

Linear filtering is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood. For example, smoothing by averaging: form the average of pixels in a neighbourhood; smoothing with a Gaussian: form a weighted average of pixels in a neighbourhood.

3.1.1 Mean Filter (Average Filter)

Mean filtering is a method of 'smoothing' images by reducing the amount of intensity variation between neighbouring pixels. The output of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. The mean filter, which is also called averaging filter, works by moving through the image pixel by pixel, replacing each value with the mean value of neighbouring pixels, including itself.

Formula to get smoothing image using mean filter

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

where $m = 2a + 1$, $n = 2b + 1$.

Application:

- To remove noise, reduce salt and pepper, and blur images.

Filter used to implement this feature :

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

Fig: Commonly used 3x3 average filter

In addition to above, we have provided **custom mask** option to the user where he can choose mask who is having dimension

- 5*5
- 7*7
- 9*9

Algorithm to get smoothed image using mean filter

- Take path of the image from user
- Take type of mask(eg: 3X3, 11X11)
- Compute input image dimensions
- padded image dimensions by adding $\text{int}(\text{size}/2)$ to both height and width
- Use four for loops to convolution between padded image and mask
 - Store results to the 2D array
- Return output image by using results from the 2D array

Difficulties/roadblocks

- Compute the row and column for padded image from input image by using the mask

Results:

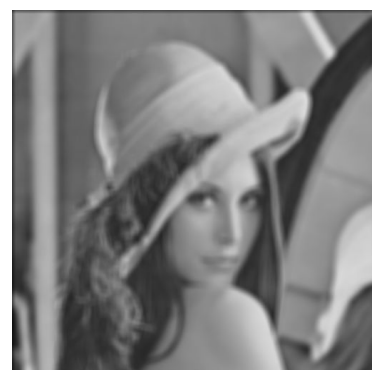
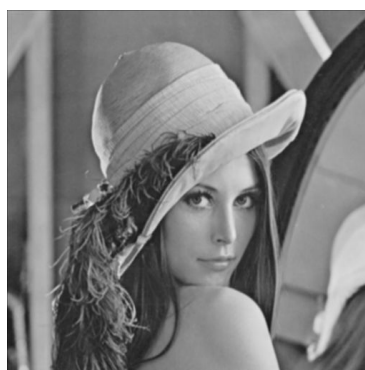
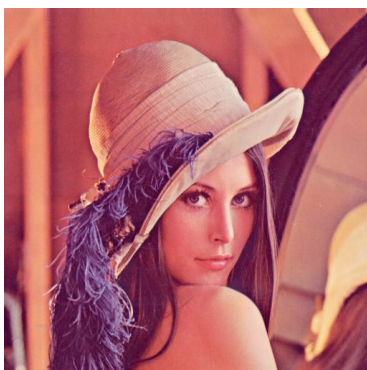


Fig: Original image

Fig : Result of smoothed image
when 3X3 mean filter applied

Fig : Result of smoothed image
when 11X11 mean filter applied

Observation:

- As what we can tell from results in above, image after processes 3X3 mean filter more blurred than image after processes 11X11 mean filter
- When we use smaller value of mask, we get more blurred image compare to when we use larger mask value
- Mean filter useful when need to blur an image or part of image. It is easier to use because averaging operation just involves addition and division

3.2 Image Smoothing Using Non-linear filter

Non linear filters like median filter, maximum filter and minimum filters are more effective in noise removal. Impulsive noises like salt and pepper are removed using these filters. Computation cost for these filters are less compare to the averaging filters.

3.2.1 Median Filter

Median filter replaces central value of window with the median of all the intensity values covered under the window. It is used in removal salt and pepper type noise.

Algorithm :

- Take input image and size of filter
- Compute input_image dimensions and padded image dimensions by adding $\text{int}(\text{size}/2)$ to both height and width
- Create padded_image by filling zeros in added rows and columns
- Move window of given filter size over image and replace the central pixel of window with Median of intensity values covered under window to obtain smooth_image
- Return smooth_image

Results :

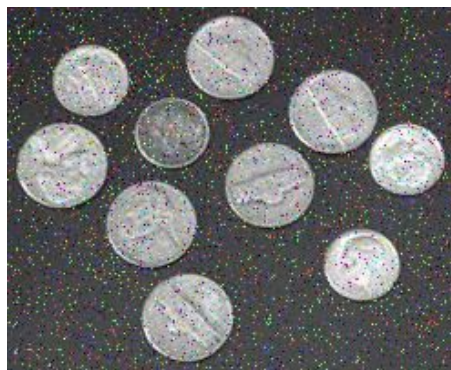


Fig : Original Image



Fig : Smooth image with 3x3 median filter

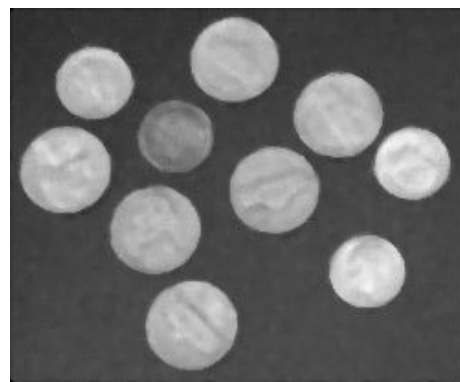


Fig : Smooth image with 5x5 median filter

4. Image Sharpening

The main aim in image sharpening is to highlight fine detail in the image, or to enhance detail that has been blurred (perhaps due to noise or other effects, such as motion). To achieve the goal, we can deal with the image in spatial domain as well as frequency domain. In this project, we will look at the 3 mainly used methods in spatial domain to achieve the sharpened image.

4.1 Image Sharpening Using First Order Derivative

First order derivative filters in image processing are implemented using the magnitude of the gradient. To detect both the edges, horizontal and vertical, we convolve the original image with respective filter mask. The magnitude is computed to find the edges and added in the original image to achieve sharpened image. Edges in the images are detected using differentiation similar to signals. Hence, this operation is called derivative operators. In this section, we have implemented 4 different gradient filters.

4.1.1 Prewitt operator

Prewitt operator is use to find edges in the image. It was developed by M.S. Prewitt. There are two filters available to find vertical and horizontal edges. Below are the masks of Prewitt operator. In Prewitt operators, the weight at the middle of first row/column and last row/column is 1.

Masks used to implement this feature

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Fig: Prewitt mask

Results



Fig: Images using Prewitt operator

4.1.2 Roberts operator

Roberts cross operator is more suitable to find cross/diagonal edges in the image. It was proposed by Lawrence Roberts. There are two filters available to find diagonal and

off-diagonal edges. This filter has lesser computation compare to other operators as the mask is 2x2. Below are the masks of Roberts operator.

Masks used to implement this feature

-1	0	0	-1
0	1	1	0

Fig: Roberts mask

Results

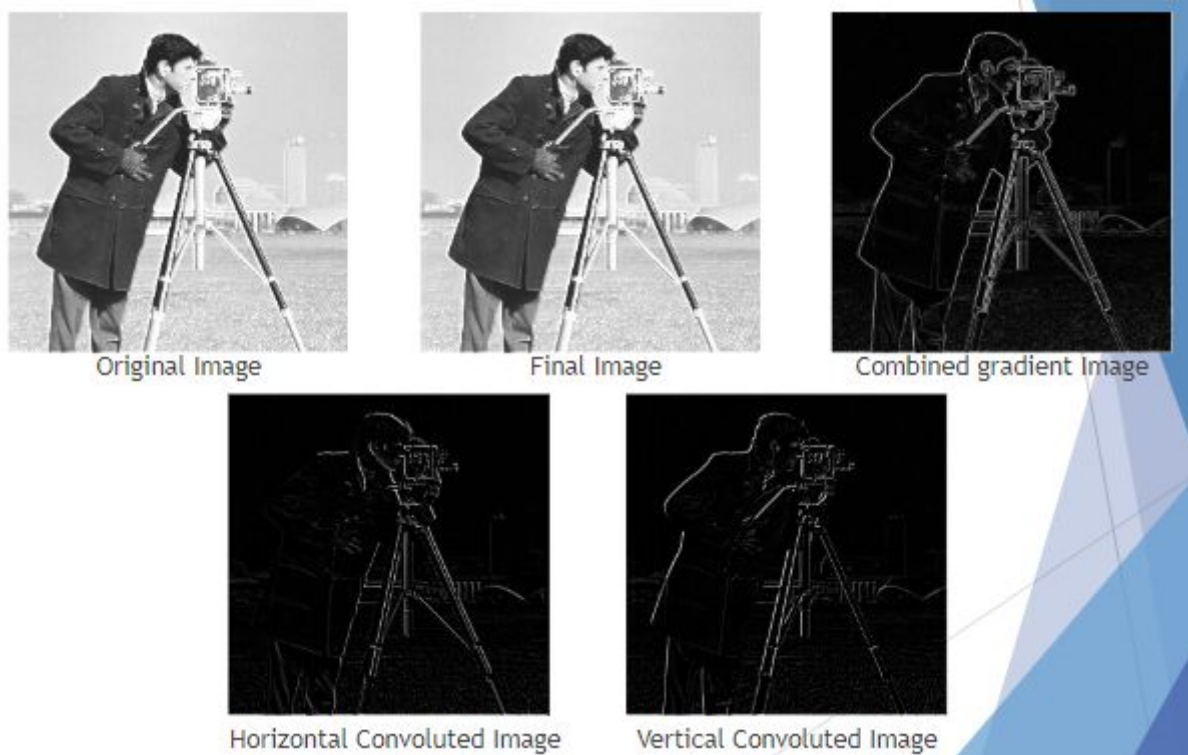


Fig: Images using Sobel operator

4.1.3 Sobel operator

Sobel operator, sometimes call Sobel-Feldman, is use to find edges in the image. It is named after Irwin Sobel and Gary Feldman. There are two filters available to find vertical and horizontal edges. Below are the masks of Sobel operator. The main point behind successful results of Sobel operator in image processing is weight. Although it looks similar to Prewitt operator, the weight is 2 which detects edges easily. Computation for Sobel operators are more than Prewitt.

Masks used to implement this feature

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

Fig: Sobel mask

Results



Fig: Images using Sobel operator

4.1.4 Custom Sobel operator

The purpose behind creating this operator is to detect the edges more accurately. We have provided option of changing the weight of the Sobel operator which results into better edges detection.

Masks used to implement this feature

-1	-w	-1	-1	0	1
0	0	0	-w	0	w
1	w	1	-1	0	1

Fig: Custom Sobel mask

Results



Fig: Images using Custom Sobel operator

Here is the algorithm used to compute edges using all mentioned filters.

Algorithm

- Take input image and name of the mask from the user
- Create an empty image of the same size with padding of (n-1), where n is size of the filter
- Convolve the filter over the original image
- Call the convolution function twice, for horizontal and vertical filter
- Compute the magnitude value obtained using both filters and store it in empty image
- Add the image with computed values in the original image to achieve sharpened image

Difficulties/roadblocks

- Resulting noisy image after convolving the self-created convolution function with original image
 - Converted the original image to int32 and performed the same operation which resulted into edges in the output image
- Getting weight for the Custom Sobel operator
 - To create a mask with user inputted weight, added an optional input parameter (**weight) in main function
- Convolving the Roberts operator with the original image
 - Since, the Roberts operator is 2x2 mask, it is difficult to convolve it with input image. Hence, we padded the mask first before convolution to get the edges

Observation

- Roberts operator is use to detect ± 45 degree edges sharper than horizontal or vertical edges
- The edges detected by Prewitt are less sharper than Sobel Operator
- Adding the feature of weight in the Custom Sobel helped to recognize edges sharper than Sobel operator
- Best way of comparing the results is using PSNR and MSE values and results show that Sobel operator is better than other two
- All the above operators are simple to implement and compute as well. But they are sensitive to noisy images and provide inaccurate results

4.2 Image Sharpening Using Laplacian

The Laplacian is an example of a second order derivative method which is good at finding the fine detail in an image. Any feature with a sharp discontinuity will be enhanced by a Laplacian operator.

Formula to get sharpen image using laplacian

$$g(x, y) = f(x, y) + c \left[\nabla^2 f(x, y) \right]$$

Here $f(x, y)$ - Input image

C - based on filter, its value will be 1 or -1

$\nabla^2 f(x, y)$ can be calculated using algorithm described later in this section

Application:

- To restore fine details to an image which has been smoothed to remove noise.[1]
- This is commonly used as an enhancement technique in remote sensing applications.[2]

Masks used to implement this feature

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

Fig: Commonly used masks

C= -1 for top 2 filters and c = 1 for bottom 2 filter.

In addition to above, we have provided **custom mask** option to the user where he can choose mask who is having dimension

- 5*5
- 7*7
- 9*9

Algorithm to get sharpen image using laplacian

- Take image and mask (3*3 or custom mask) from user
- Retrieve image and mask dimensions
- Compute padded image dimensions (take integer of filter dimension/2, we have to add that much rows above and below, and columns on left and right)
- Store image in padded image
- Using two for loops with condition till original image dimension apply convolution between padded image and mask
 - Based on mask type, multiply resultant value by 1 or -1
 - Store resultant value in new 2D matrix
- Computer filtered image by adding result of colvolution to the input image

Difficulties/roadblocks

- Implementation of calculating rows and columns to be padded to the original image for custom filters

Results:



Fig: Original image



Fig : Result of convolution and final image when 1st filter applied. (-4 in the middle)



Fig : Result of convolution and final image when 2nd filter applied. (-8 in the middle)



Fig : Result of convolution and final image when 3rd filter applied. (4 in the middle)



Fig : Result of convolution and final image when 4th filter applied. (8 in the middle)



Fig : Result of convolution and final image when custom 5*5 filter applied.
Following Custom Filter used

$$\begin{bmatrix} 4 & 3 & 2 & 3 & 4 \\ 3 & 1 & 1 & 1 & 3 \\ 2 & 1 & -5 & 1 & 3 \\ 3 & 1 & 1 & 1 & 3 \\ 4 & 3 & 1 & 3 & 4 \end{bmatrix}$$

Observation:

- When we include diagonal terms in mask, we get more sharpen image compare to when we exclude diagonal terms.
- Laplacian either take out outward edges (eg . -4 in the middle) or take out inward edges (eg 4 in the middle)based on mask used.
- Laplacian makes any edges in the original image much sharper and give them more contrast. [2]

4.3 Image Sharpening Using Unsharp Masking and High Boosting

Unsharp masking and high boosting are used to sharpen and enhance the details like edges in an image. Any feature with a sharp discontinuity will be enhanced. It is one of the simple techniques in spatial filtering.

Applications :

To enhance fine details in an image, for example in forensic department when biometric images are required to analysed their fine details are very important and hence to highlight those unsharp masking is one of the techniques used .

Filters used in implementation:

The first part of this technique is smoothing the original image, for smoothing following filters can be used. Average filters or gaussian filters can be used for smoothing.

$\frac{1}{9} \times$	1	1	1
	1	1	1
	1	1	1
Average filter			
$\frac{1}{16} \times$	1	2	1
	2	4	2
	1	2	1
Gaussian filter			

Similarly filters with larger size can also be used.

Algorithm :

- Take original_image and filter from user
- Retrieve original_image and filter dimensions
- Compute padded_image dimensions (take integer of (filter dimension)/2, we have to add that much rows above and below, and columns on left and right)
- Store image in padded_image
- Using two loops with condition till original_image-dimension apply convolution between padded_image and smoothing filter to obtain smooth_image
- Remove padded zeros from smooth_image
- Subtract smooth_image from original_image to obtain Mask
- Get the K-value entered by user (For high boosting)
- Multiply K-value with the mask and add to original_image to obtain sharp_image
- Return sharp_image

Difficulties/Roadblocks :

Sometimes the intensity values in the sharpened image may get out of range, i.e. it can get below 0 or above 255. There are two methods to deal with this situations as follow

1. Threshold the intensity values to 0 and 255, i.e. the values below 0 are set to 0 and the values above 255 are set to 255.
2. Perform full contrast stretch on the raw sharpened image. The disadvantage of this method is that sometimes image become darker with large K-value and details may get lost. So it's better to use first method.

Results :



Fig : Original Image

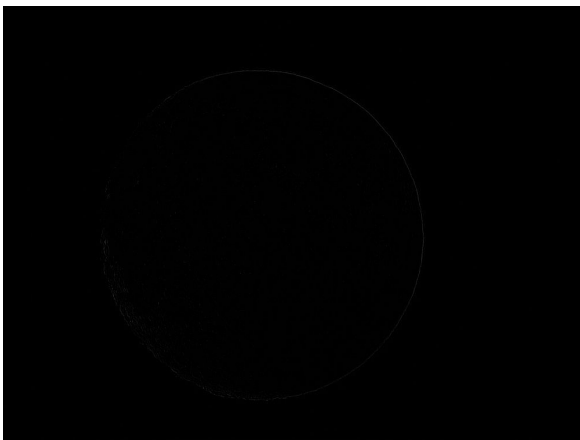


Fig : Mask with 3x3 average filter

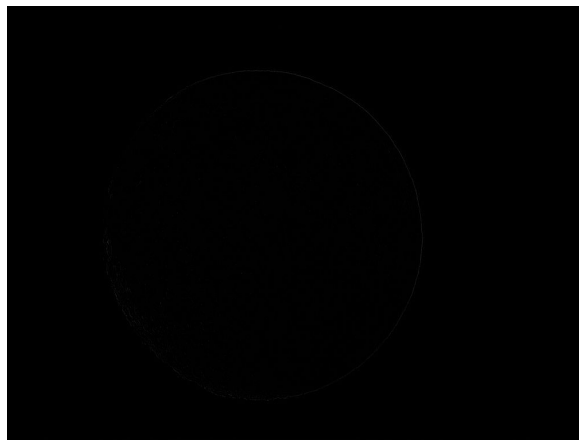


Fig : Mask with 3x3 gaussian filter

Results of Unsharp Masking :



Fig : Sharp image with 3x3 Average Filter

Fig : Sharp image with 3x3 Gaussian Filter

Result of High Boosting :

Note - For all the results shown below average filter of dimension 3x3 are used for smoothing.



Fig : Sharp Image ($K = 1$)



Fig : Sharp Image ($K = 2$)



Fig : Sharp Image ($K = 3$)



Fig : Sharp Image ($K = 5$)



Fig : Sharp Image (K = 7)



Fig : Sharp Image (K = 9)



Fig : Sharp Image (K = 10)

Fig : Sharp Image (K = 15)

Observations :

- As the K-value is increased the image becomes more and more sharper.
- Also with increase of K-value the noise in the image also get highlighted.

5. References

- [1] Image sharpening with a Laplacian Kernel, http://www.idlcoyote.com/ip_tips/sharpen.html
- [2] Laplacian/Laplacian of Gaussian, <https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>
- [3] Digital Image Processing, *Third Edition*, by Rafael Gonzalez and Richard Woods
- [4] Sobel, Prewitt, Roberts operator, https://www.tutorialspoint.com/dip/concept_of_masks.htm
- [5] Spatial Domain filtering, Part 1, <https://sisu.ut.ee/imageprocessing/book/8>

- [6] Digital Image Processing, <http://qil.uh.edu/dip/>