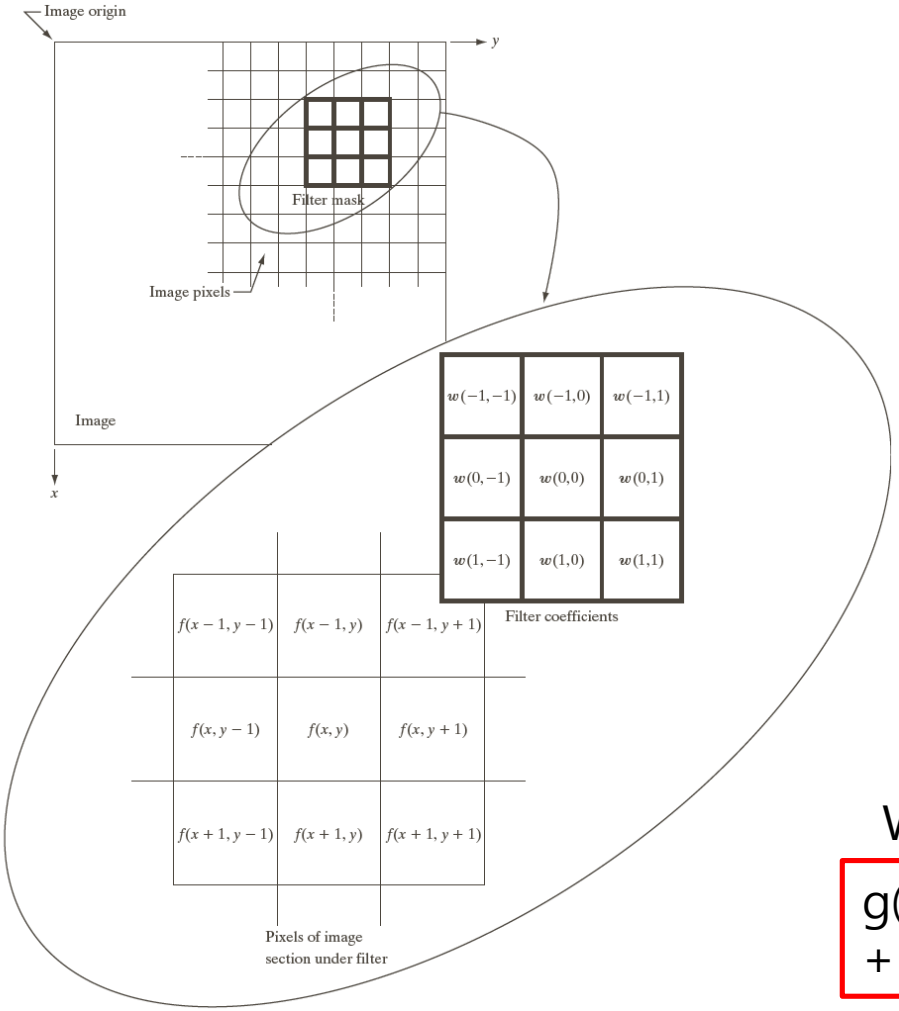
Spatial Filtering

Sung Soo Hwang

Introduction

- Spatial filtering
  - Spatial filters = spatial masks, kernels, templates, windows



When using 3X3 spatial filters,

$$g(x,y) = w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y) +\ldots w(0,0)f(x,y)+\ldots w(1,1)f(x+1,y+1)$$

Spatial filtering

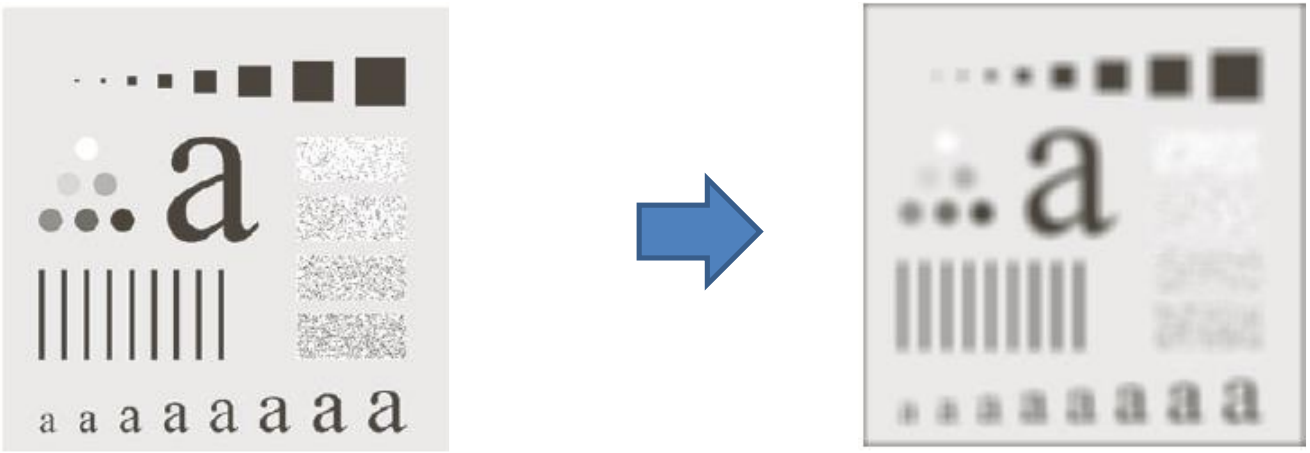
- Averaging filter
  - The average of the pixels contained in the neighborhood of the filter mask
  - Sometimes called averaging filters (or low pass filters)
  - For every pixel, replace the value of the pixel by the average of the intensity levels in the neighborhood
  - Advantage and disadvantage
    - It reduces random noises
    - It blurs an image

Spatial filtering

- Averaging filter

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

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



Spatial filtering

- Gaussian filter

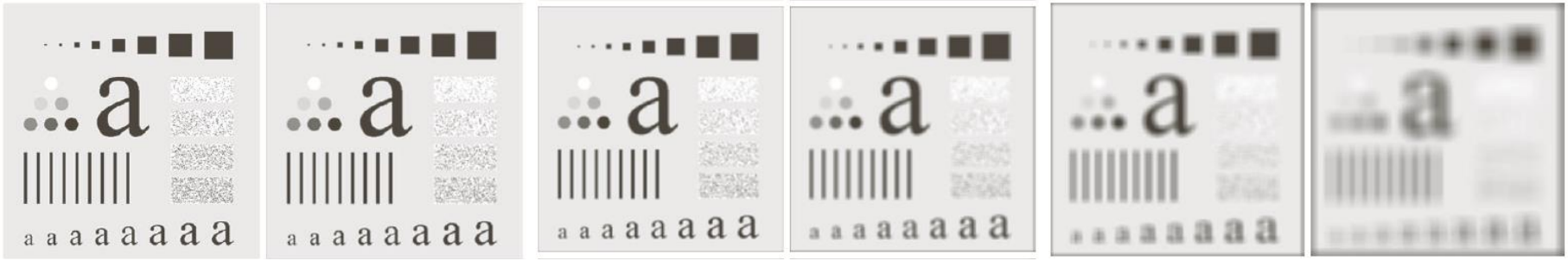
$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian Function

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

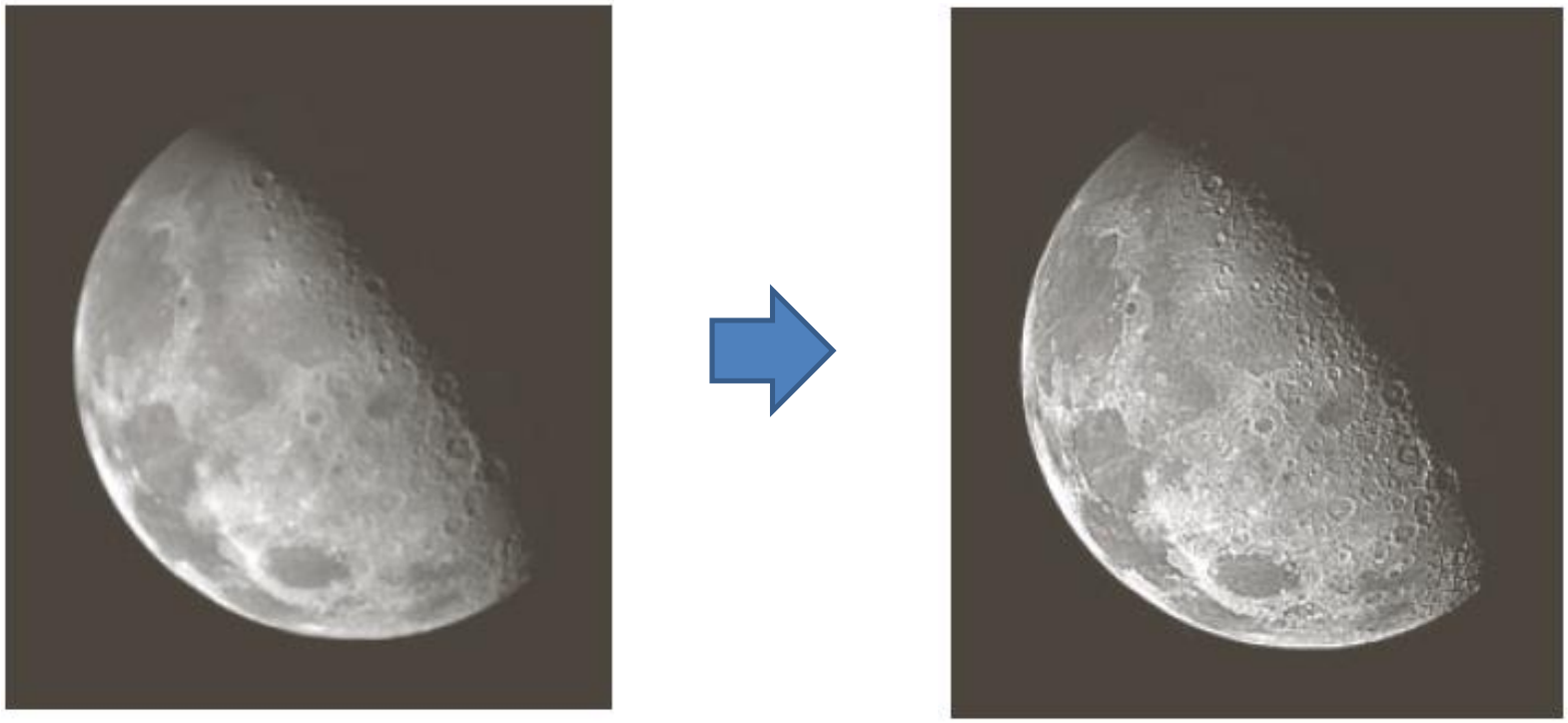
3x3 filter mask

- Mask size
  - Mask size matters
  - If you want to blur small objects, use a small size mask
  - Using a large mask is computationally expensive



Spatial filtering

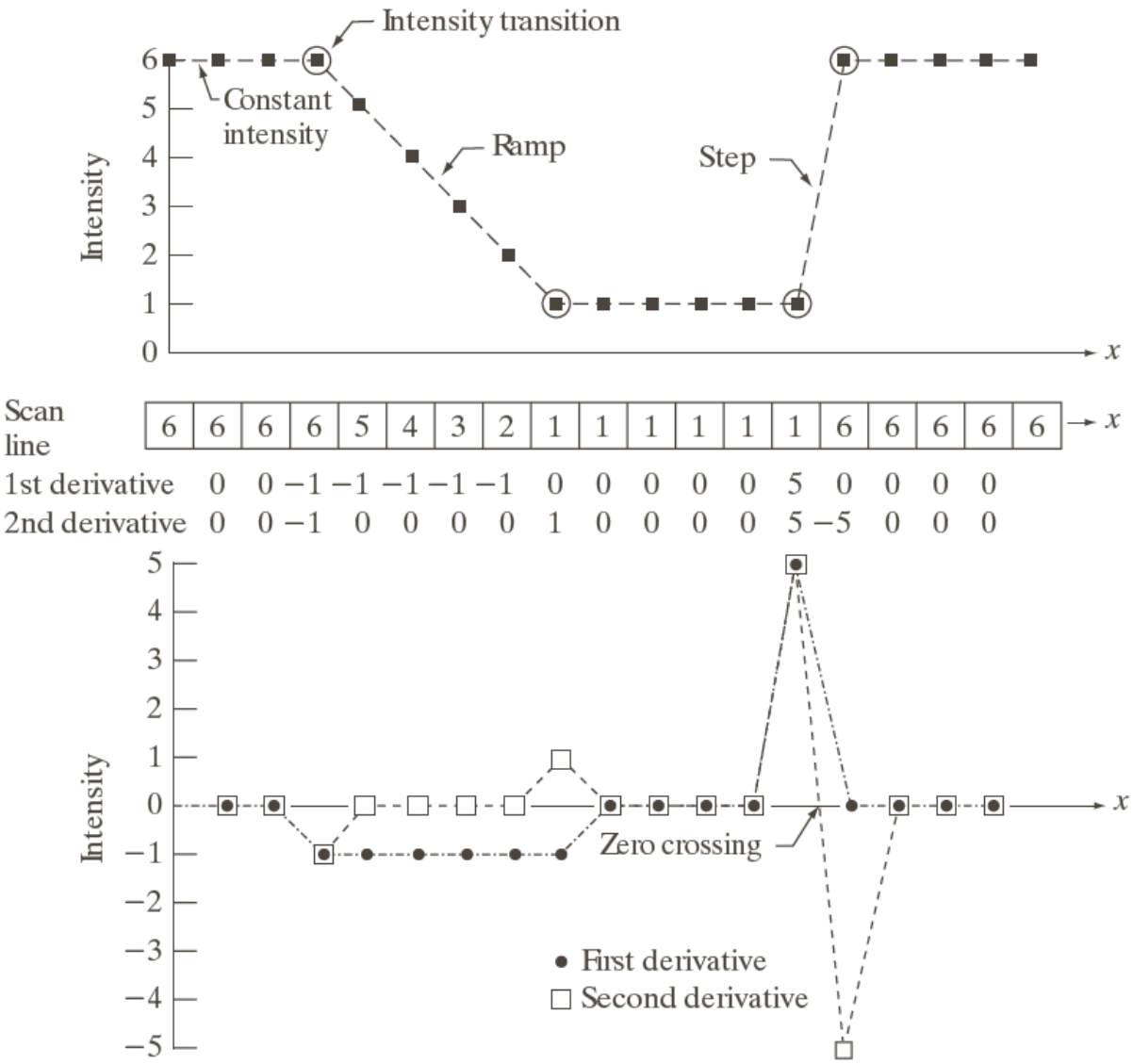
- Sharpening
  - The principal objective of sharpening is to highlight transitions in intensity



  - Averaging is analogous to integration, it is logical to conclude that sharpening can be accomplished by spatial differentiation

Spatial filtering

- Sharpening using second derivative



Spatial filtering

- Sharpening using second derivative
  - Mask for applying second derivative

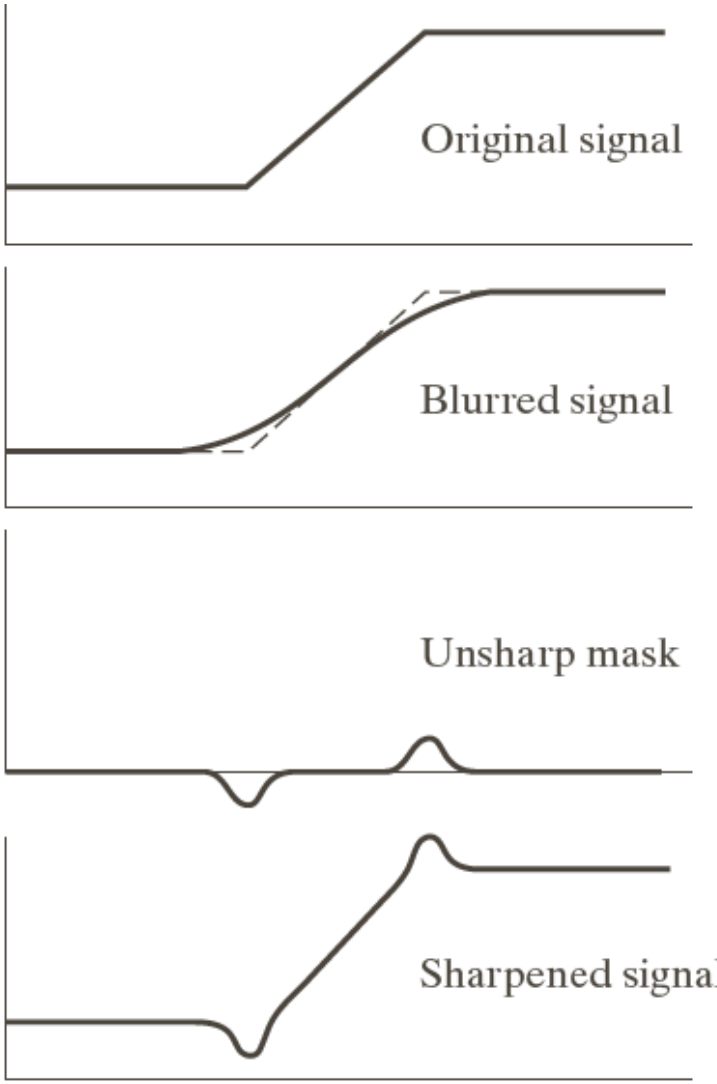
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

  - Algorithm
    1. obtain second derivative of the input image
    2. Add the second derivative with the input image

Spatial filtering

- Sharpening using unsharp masking
  - Unsharp masking

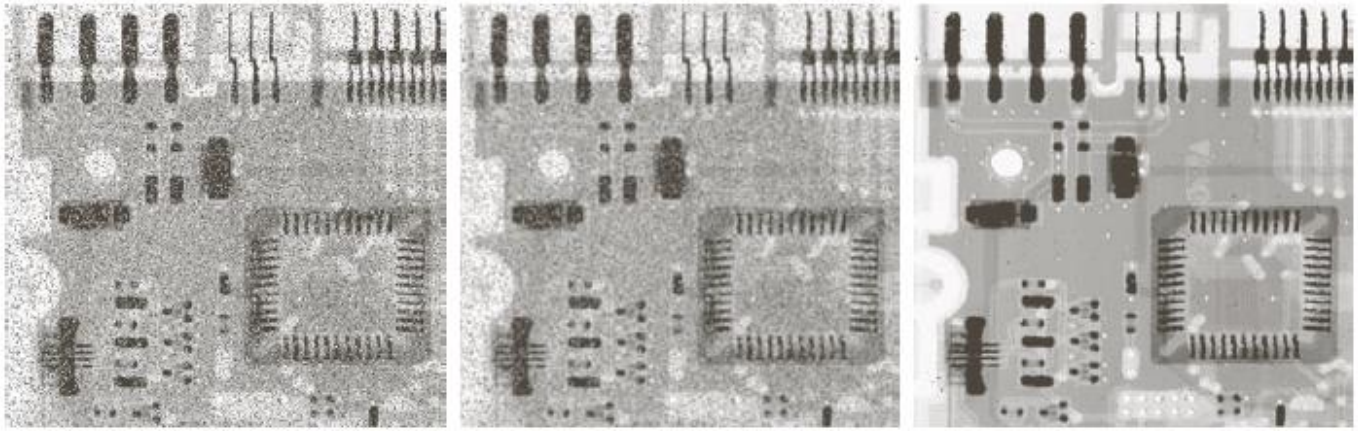


$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

$$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

Other filter - Median filter

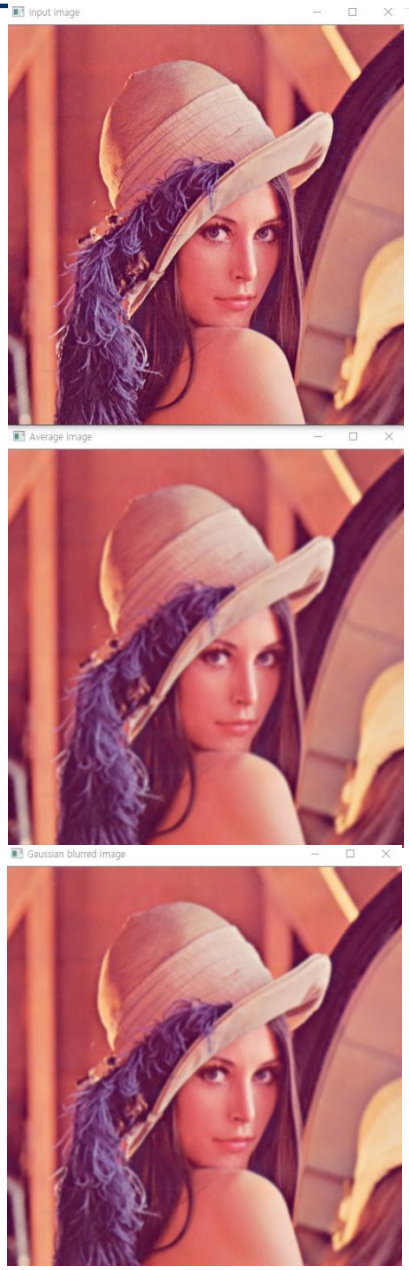
- Median value
  - For 3X3 neighborhood, the median is the 5^{th} largest
  - For 5X5 neighborhood, the median is the 13^{th} largest
- Median filter
  - Find the median value of a mask, and replace the values of pixels in the mask with the median value
  - Isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $m^2/2$ are eliminated by an mXm median filter
  - It is effective in the presence of impulse noise (or salt-and pepper noise)



Spatial filtering

- Averaging filter
  - Example code

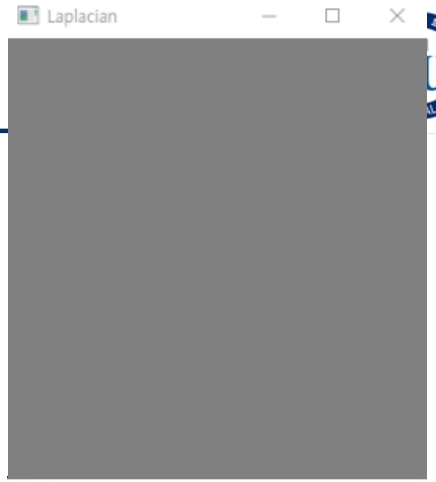
```
int main() {
    Mat image, AvgImg, GaussianImg;
    image = imread("lena.png");

    // Blurs an image using the normalized box filter
    // image: input image, AvgImg: output image, Size(5, 5): blurring kernel size
    blur(image, AvgImg, Size(5, 5));

    // Blurs an image using a Gaussian filter
    // image: input image, GaussianImg: output image, Size(5, 5): Gaussian kernel size
    // 1.5: Gaussian kernel standard deviation in X direction
    GaussianBlur(image, GaussianImg, Size(5, 5), 1.5);

    imshow("Input image", image);
    imshow("Average image", AvgImg);
    imshow("Gaussian blurred image", GaussianImg);

    waitKey(0);
    return 0;
}
```

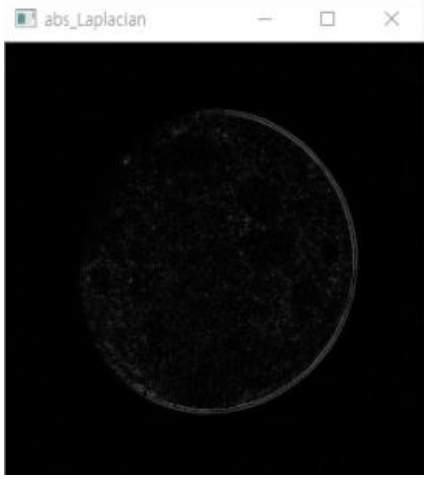


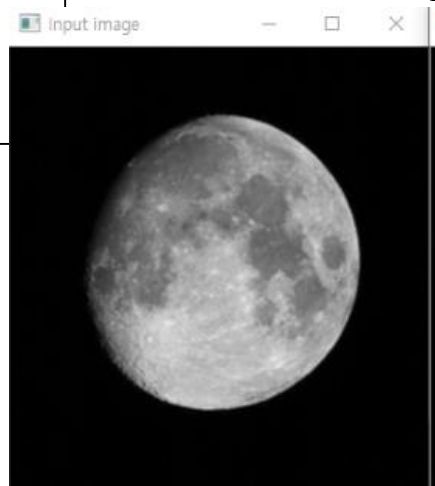
Spatial filtering

- Sharpening using second derivative
  - Example code

```
int main() {
	Mat image, laplacian, abs_laplacian, sharpening;
	image = imread("moon.jpg", 0);

	GaussianBlur(image, image, Size(3, 3), 0, 0, BORDER_DEFAULT);
	// calculates the Laplacian of an image
	// image: src, laplacian: dst, CV_16S: desire depth of dst,
	// 1: aperture size used to compute second-derivative (optional)
	// 1: optional scale factor for the computed Laplacian values
	// 0: optional delta value that is added to the result
	Laplacian(image, laplacian, CV_16S, 1, 1, 0);
	convertScaleAbs(laplacian, abs_laplacian);
	sharpening = abs_laplacian + image;

	imshow("Input image", image);
	imshow("Laplacian", laplacian);
	imshow("abs_Laplacian", abs_laplacian);
	imshow("Sharpening", sharpening);

	waitKey(0);
}
```





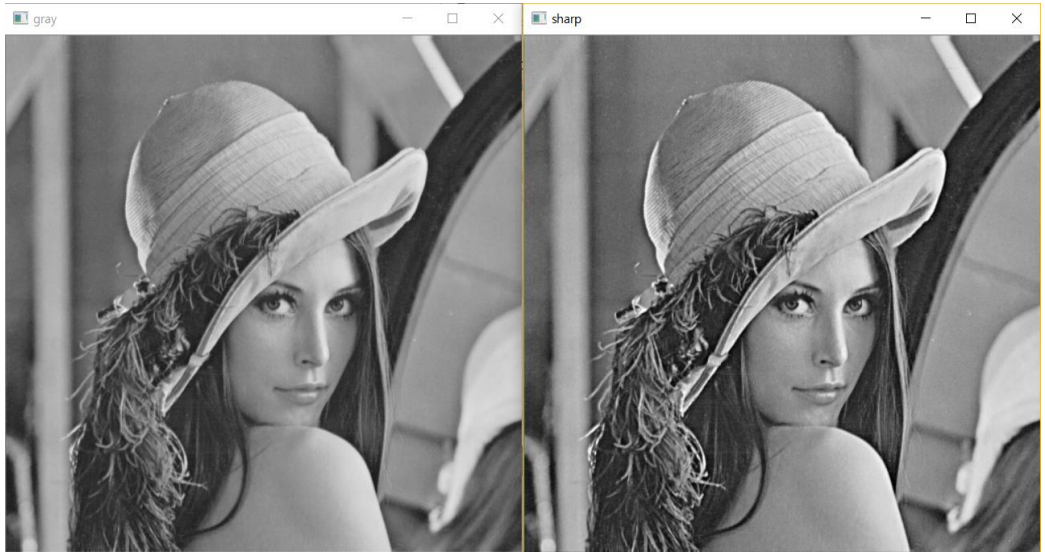


Try putting
Laplacian(image, laplacian, CV_16S, 5, 5, 5);
To visualize the changes of the function

Spatial filtering

- Using unsharp masking
  - Example code

```
int main() {
	Mat input = imread("lena.png");
	Mat gray, blur, sharp;
	cvtColor(input, gray, COLOR_BGR2GRAY);
	GaussianBlur(gray, blur, Size(5, 5), 3);
	addWeighted(gray, 1.5, blur, -0.5, 0, sharp);
	imshow("gray", gray);
	imshow("sharp", sharp);

	waitKey(0);
}
```



Other filter

- Median filter
  - Example code

```
int main() {
	Mat image = imread("saltnpepper.png", 0);
	imshow("SaltAndPepper", image);
	Mat mf1, mf2;
	// Blurs an image using the median filter
	// image: src, mf1: dst, 3: aperture size(must be odd and greater than 1)
	medianBlur(image, mf1, 3);
	imshow("MedianFiltered1", mf1);

	medianBlur(image, mf2, 9);
	imshow("MedianFiltered2", mf2);

	waitKey(0);
	return 0;
}
```

http://www.fit.vutbr.cz/~vasicek/imagedb/img_corrupted/impnoise_055/189003.png