Image Filters (Week 3)

CMP020L014S

Image

- An image is a visual representation of something.
- It can be 2D/3D, which can be fed into the visual system to convey information.
- In the context of signal processing, an image is a distributed amplitude of color(s).
- The smallest element of image is called pixel.
  - Pixel is a point on the image that takes on a specific shade or color. In data science, 2D/3D images usually represented in the following way:
    - Grayscale - A pixel is an integer with a value between 0 to 255 (0 is completely black and 255 is completely white).
    - RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).

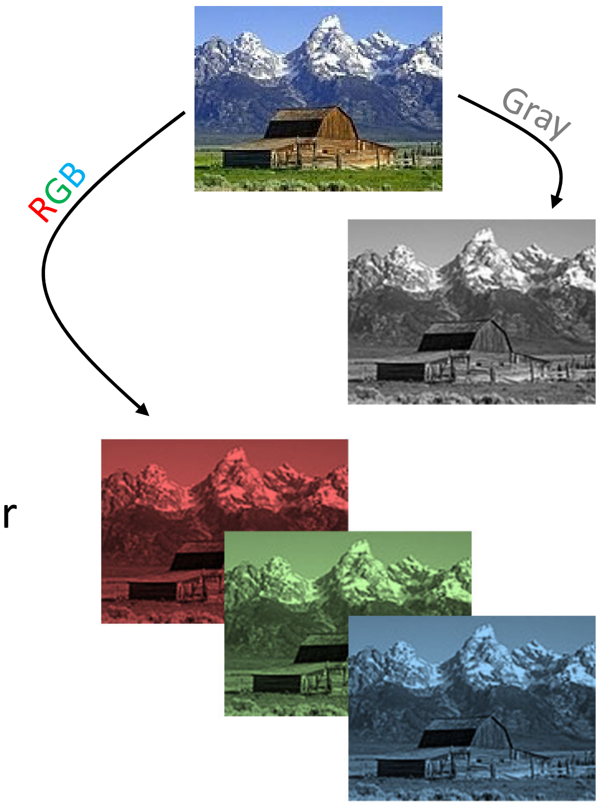
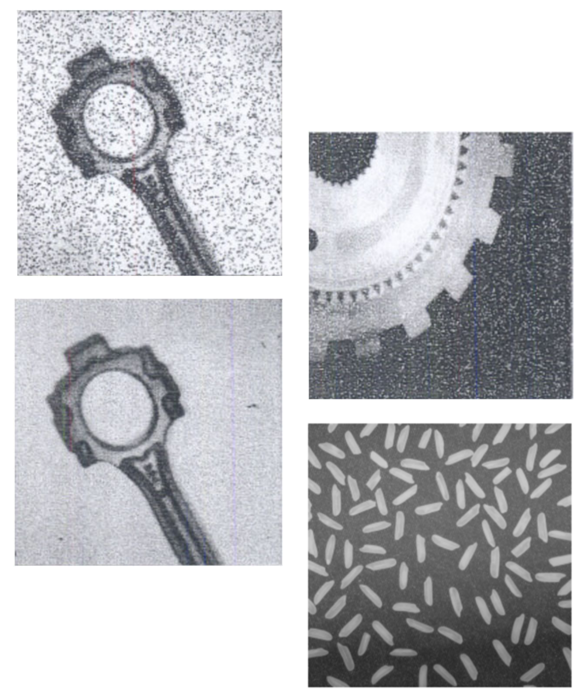


Image Filters

- The common image related artifacts during image acquisition are noise caused due to external interference and imbalance in illumination.
  - Salt and pepper noise contains random occurrences of both black and white intensity values.
  - Impulse noise contains only random occurrences of white intensity values.
  - Gaussian noise contains variations in intensity that are drawn from a Gaussian or normal distribution and is a very good model for many kinds of camera sensor noise.
  - Uneven illumination is one of the most unavoidable issues that make images look imperfect.



Mean Filter

- This filter is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood:

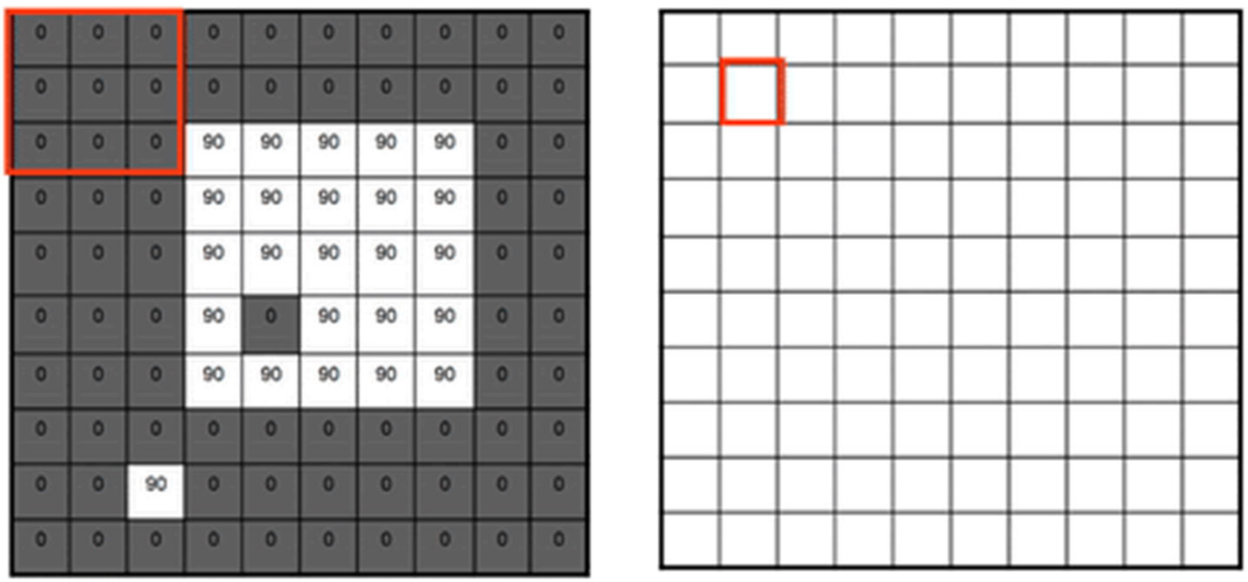
$$h[i,j] = \frac{1}{M} \sum_{(k,l)\in N} f[k,l]$$

where, M is the total number of pixels in the neighborhood N. For example, taking a 3x3 neighborhood about $[i,j]$ yields:

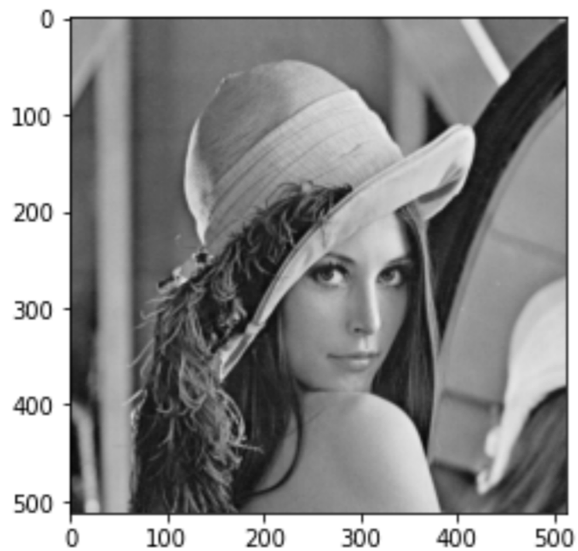
$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} f[k,l]$$

Mean Filter

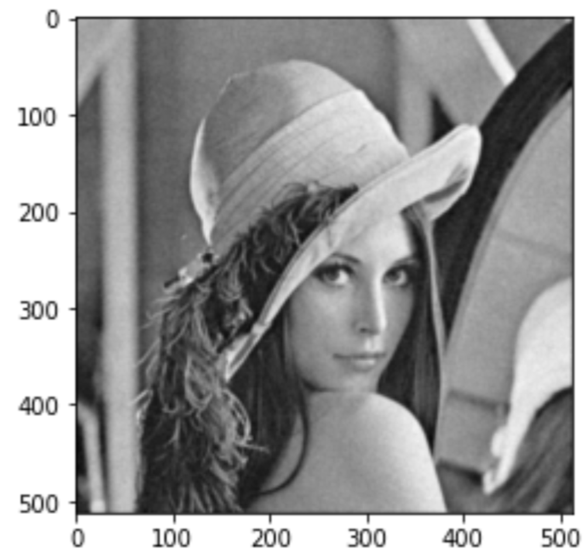
- Mean filter (F) is a 3x3 matrix.



Mean Filter



Input image



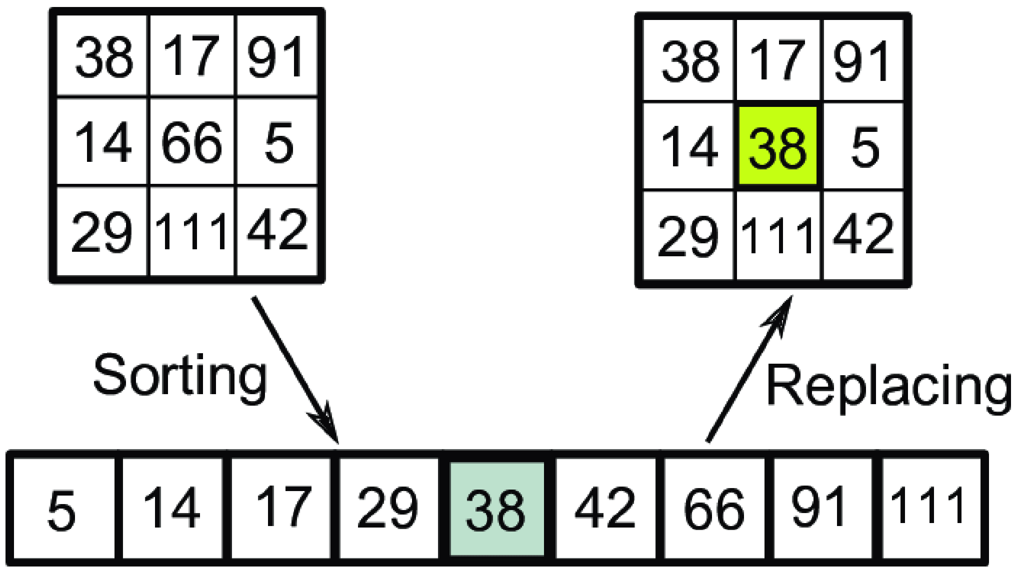
Processed image

Median Filter

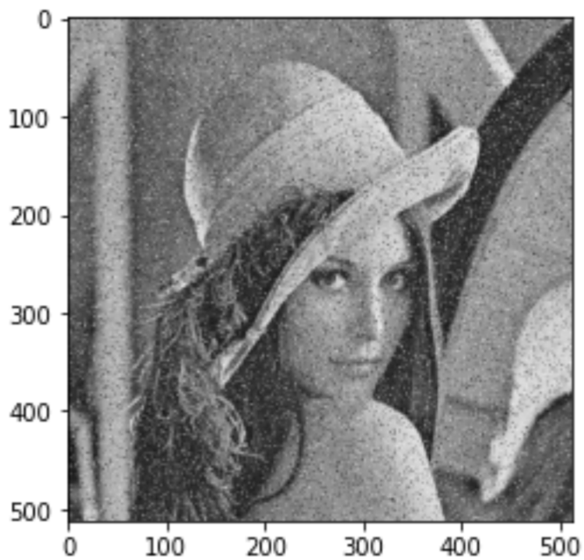
- The main problem with local averaging operations is that they tend to blur sharp discontinuities in intensity values in an image.
- An alternative approach is to replace each pixel value with the median of the gray values in the local neighborhood. Filters using this technique are called median filters.
- Median filters work in successive image windows in a fashion similar to linear filters, i.e.
  - Sort the pixels into ascending order by gray level.
  - Select the value of the middle pixel as the new value for pixel $[i, j]$.

Median Filter

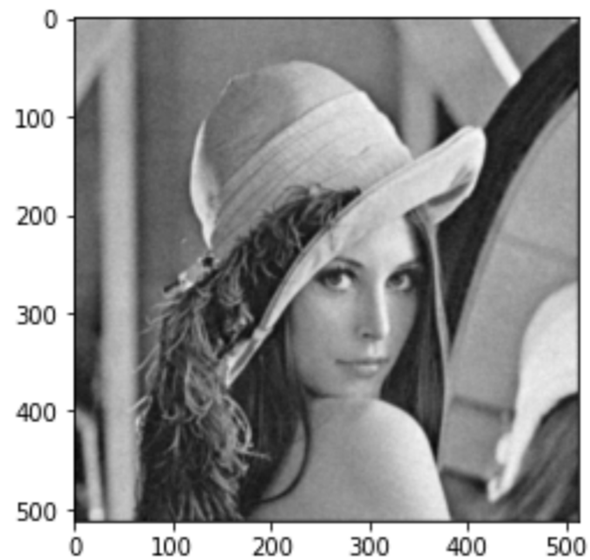
- Median filter (F) is a 3x3 matrix of ones.



Median Filter



Input image



Processed image

Mean/Median Filter: DIY

- Consider matrix I and mean filter F.

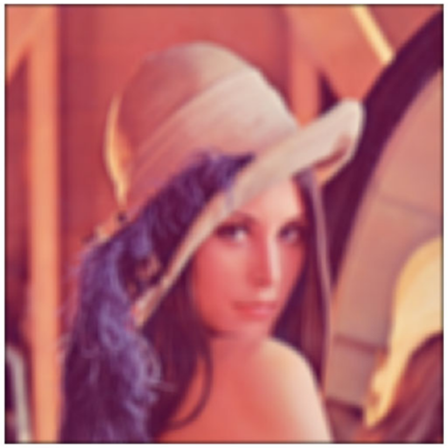
10	20	85	97	55
40	60	70	66	52
9	70	90	87	12
15	54	33	60	11
6	26	73	59	9

1	1	1
1	1	1
1	1	1

- Multiply both matrices element-by-element centered at $F_{2,2}$ by averaging the value of the transformed matrix, slide F all over I and repeat the same task.
- Produce new transformed comprising of mean/median tranformation.

Gaussian Filter

- The Gaussian filter is a modified version of the Mean filter where the weights of the impulse function are distributed normally around the origin.
  - Hence, the intensity falls in a Gaussian fashion away from the origin.
- Gaussian filters help to reduce noise by suppressing the high-frequency components which come at the cost of a final image being blurred, called Gaussian blur.



Gaussian blur

Gaussian Filter

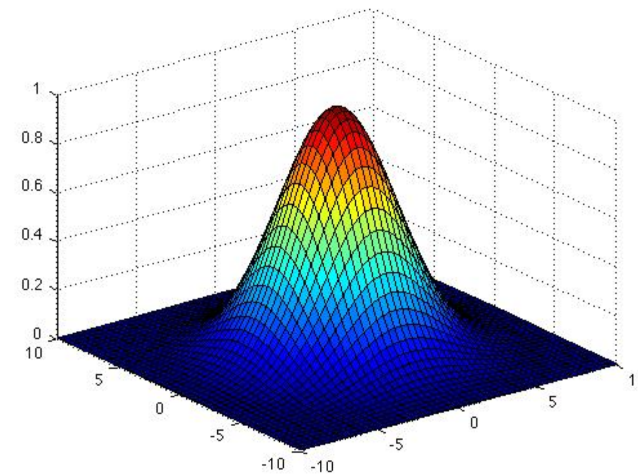
- Designing Gaussian filters is to compute the mask weights directly from the discrete Gaussian distribution.

$$g[i,j] = ce^{-\frac{i^2+j^2}{2\sigma^2}}$$

where, c is a normalizing constant. By rewriting this as,

$$\frac{g[i,j]}{c} = e^{-\frac{i^2+j^2}{2\sigma^2}}$$

and choosing a value for σ^2, we can evaluate it over an $n \times n$ window to obtain a kernel, or mask, for which the value at $[0,0]$ equals 1.



Gaussian Filter: Example

- Choosing σ=$\sqrt{2}$ and n=7, the expression yields the array:

$[i,j]$	-3	-2	-1	0	1	2	3
-3	.011	.039	.082	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105
1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

- However, we desire the filter weights to be integer values for ease in computations. Therefore, we take the value at one of the corners in the array, and choose k such that this value becomes 1.

$$g[3,3] = e^{-\frac{3^2+3^2}{2(\sqrt{2})^2}} = 0.011$$

- Hence, $g[0,0]$ can be modified as: $g[0,0]/g[3,3]$ = 1.000/0.011 = 91

Gaussian Filter: Example

- Now, by divide the rest of the weights by $g[3,3]$, we will obtain following filter values:

$[i,j]$	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

- When performing the convolution, the output pixel values must be normalized by the sum of the mask weights to ensure that regions of uniform intensity are not affected.
  - Average = $\sum_{i=-3}^{+3} \sum_{j=-3}^{+3} g[i,j] = 1115$
- Hence, convolution with gaussian filter should be performed as follows:
  - $h[i,j] = \frac{1}{1115} (f[i,j] * g[i,j])$

Laplacian Filter

- Laplacian filters are also called second derivative filters used to find areas of rapid change (edges) in images.
- Since second derivative filters are very sensitive to noise, it is common to smooth the image (e.g., using a Gaussian filter) before applying the Laplacian.
- Recall Taylor series expansion (for x-direction):
  - $f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \ O(h^4)$
  - $f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + O(h^4)$
- Adding, $f(x-h) + f(x+h) = 2f(x) + h^2 f''(x) + O(h^4)$
  - $\Rightarrow \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} = f''(x) + \boxed{O(h^2)}$ Neglect H.O.T.

How x-directional filter will look like?

How y-directional filter will look like?

1	**-2**	**1**

Central difference approx to second derivative

Laplacian Filter

- To find the Laplacian filter, both x- and y-directional filters should be combined together.

$$I_{xx} + I_{yy} = \left(\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \right) * I$$

Perform matrix addition

$$= \boxed{\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I} \Rightarrow \nabla^2 I(x, y)$$

Laplacian filter

- However, it can be observed that, it tends to amplify the noise in the image. Like salt and pepper noise surrounding the center of the filter.
- Hence, first, we use a Gaussian filter on the noisy image data to smoothen it and then subsequently use the Laplacian filter for edge detection.

Laplacian Filter: Alternative Expression

- Let's recall how the partial derivative is calculated in 2D function f that represents a matrix.
- In continuous setting, partial derivative of f with respect to x is defined as follows:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

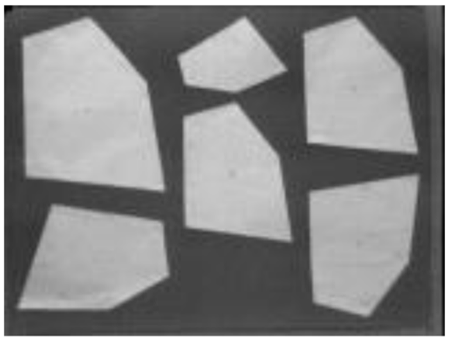
- However, in computer vision, we are dealing with matrix which is a discrete data. Thus, we approximate it by using finite differences.

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+1, y) - f(x,y)}{1} \Rightarrow \boxed{\begin{array}{|c|c|} 1 & -1 \end{array}}$$

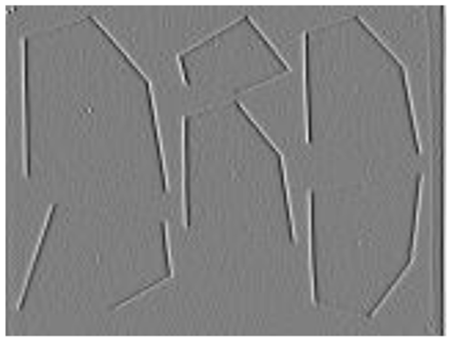
- Second derivative by repeated convolution: $\begin{array}{|c|c|} \hline 1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 1 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 1 & -2 & 1 \\ \hline \end{array}$

What we will get after convolving two vectors?

Laplacian Filter: Application

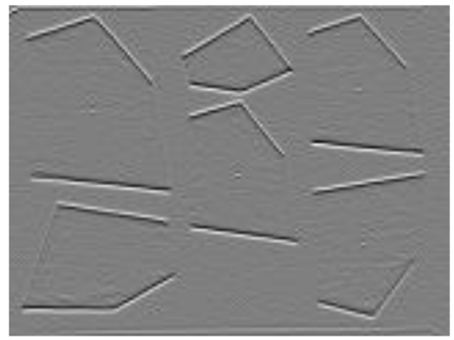


$$I(x, y)$$



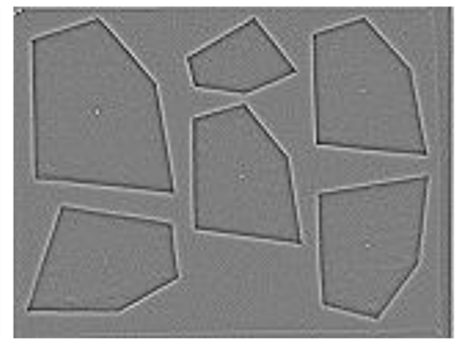
0	0	0
1	-2	1
0	0	0

$$\frac{\partial^2 I(x, y)}{\partial x^2}$$



0	1	0
0	-2	0
0	1	0

$$\frac{\partial^2 I(x, y)}{\partial x^2}$$



0	1	0
1	-4	1
0	1	0

$$\frac{\partial^2 I(x, y)}{\partial x^2} + \frac{\partial^2 I(x, y)}{\partial x^2}$$