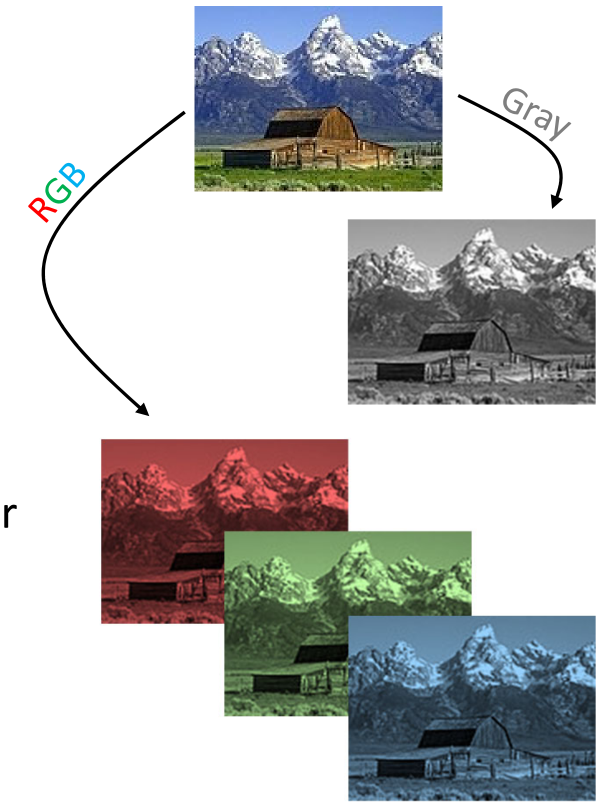
# Image Filters
# (Week 3)

# Image

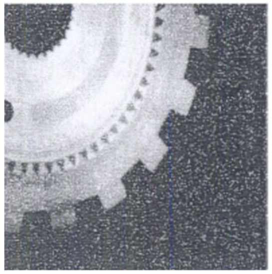
- An image is a visual representation of something.
- It can be 2D/3D, which can be fed into the visual system to convey information.
- In the context of signal processing, an image is a distributed amplitude of color(s).
- The smallest element of image is called pixel.
  - Pixel is a point on the image 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

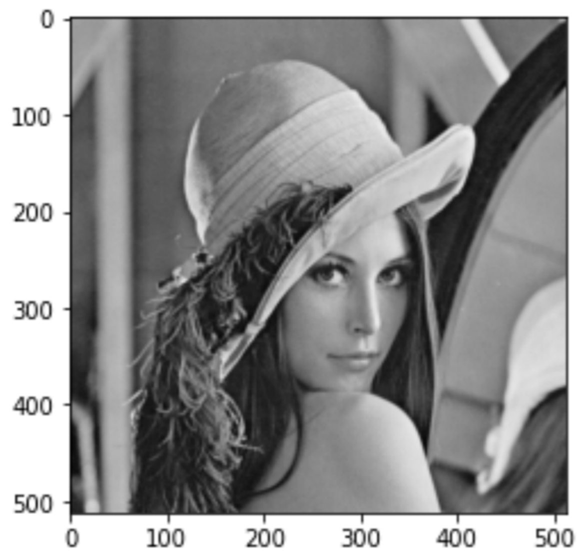
- This filter is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood:

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

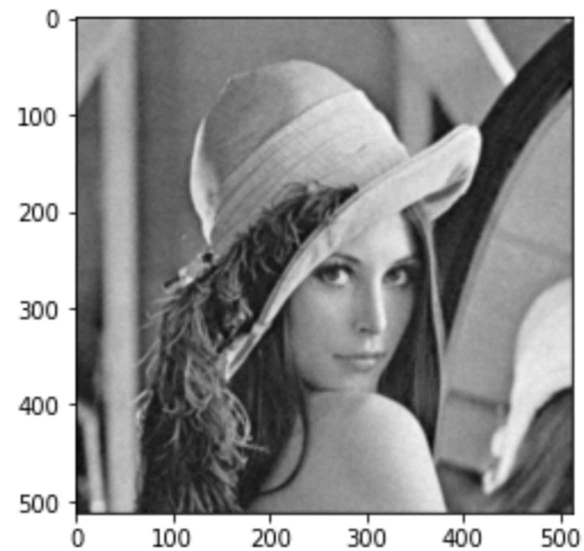
where, M is the total number of pixels in the neighborhood N. For example, taking a 3x3 neighborhood about $[i,j]$ yields:

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

# Mean Filter




Input image

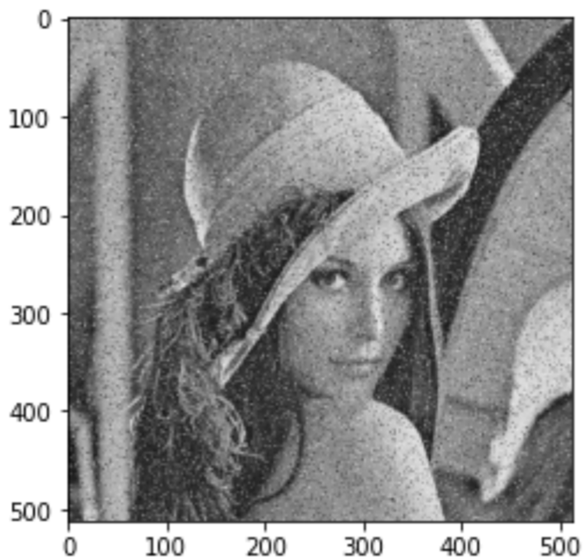



Processed image

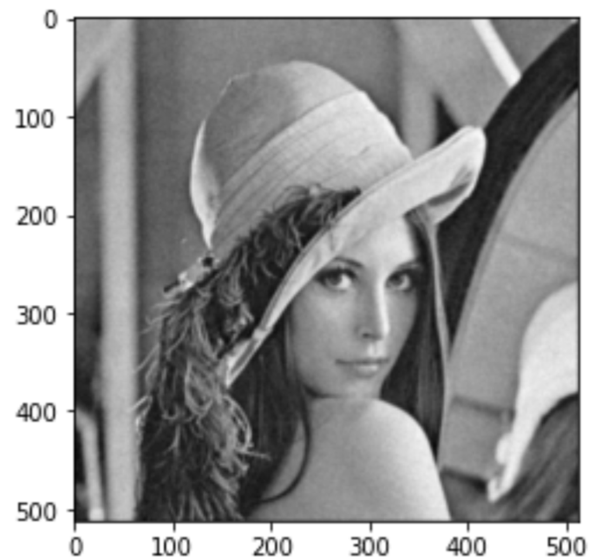
# Median Filter

- 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



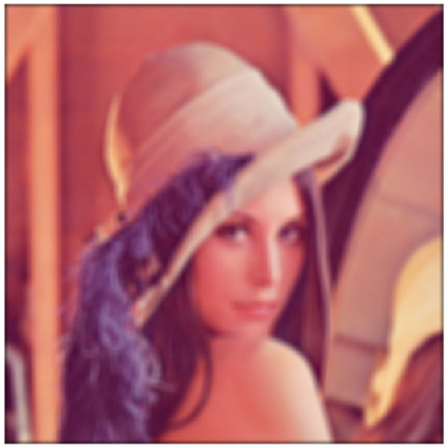

Input image




Processed image

# 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

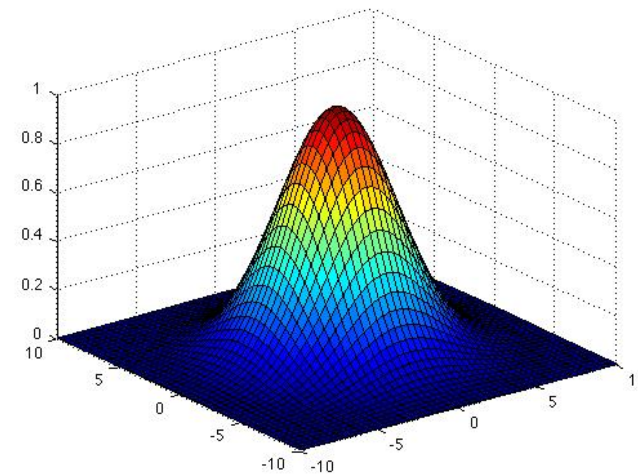
- Designing Gaussian filters is to compute the mask weights directly from the discrete Gaussian distribution.

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

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 $\sigma^2$, 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.



# 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, $f(x-h) + f(x+h) = 2f(x) + h^2 f''(x) + O(h^4)$
  - $\Rightarrow \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} = f''(x) + O(h^2)$

# Laplacian Filter

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

$$I_{xx} + I_{yy} =$$

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

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:

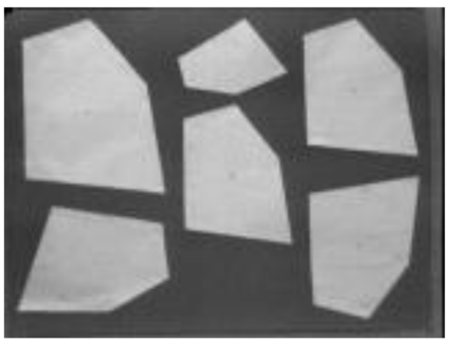
$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

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

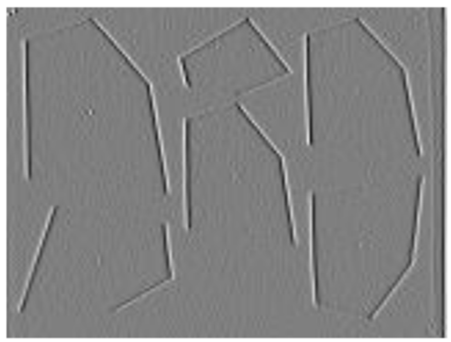
$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+1, y) - f(x,y)}{1} \Rightarrow \boxed{\begin{array}{c|c} 1 & -1 \end{array}}$$

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

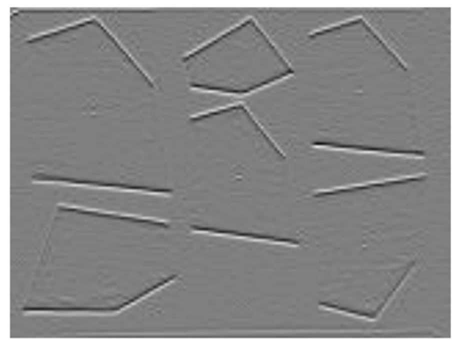
# Laplacian Filter: Application



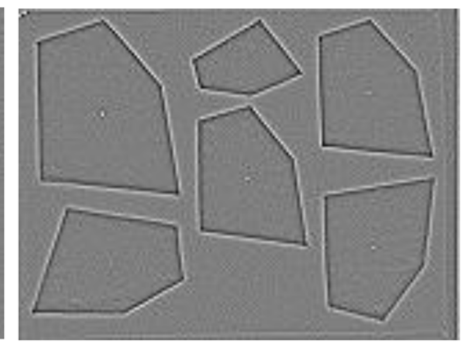
$I(x, y)$



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



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



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