3D Data Processing

Image processing-1

Hyoseok Hwang

contents

- Introduction of image processing
- Pixel point processing
- Geometric transform
- Domain transform
- Spatial filtering

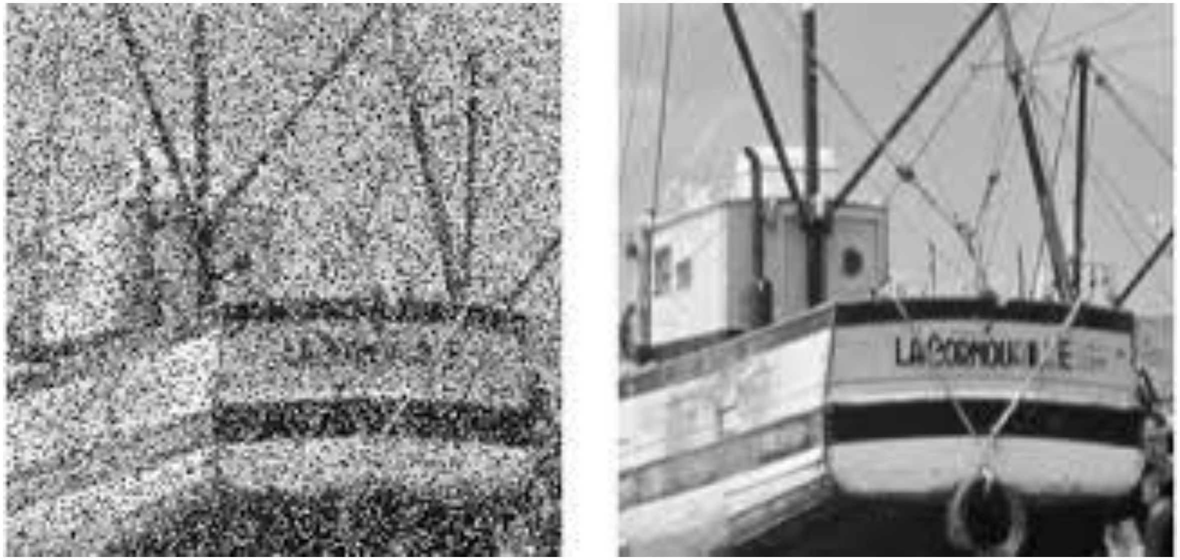
Image processing

- Image Enhancement
  - Brightness, contrast enhancement
  - Denoising
  - Deblurring
- Image Transform
  - Geometric transform
  - Frequency domain
  - Hough transform
- Image Restoration
  - Colorization
  - Inpainting

Image enhancement

- Accentuate certain image features for image display or subsequent analysis
  - brightness/contrast enhancement, histogram processing, denoising, edge sharpening, Smoothing, Detail enhancement

contrast enhancement



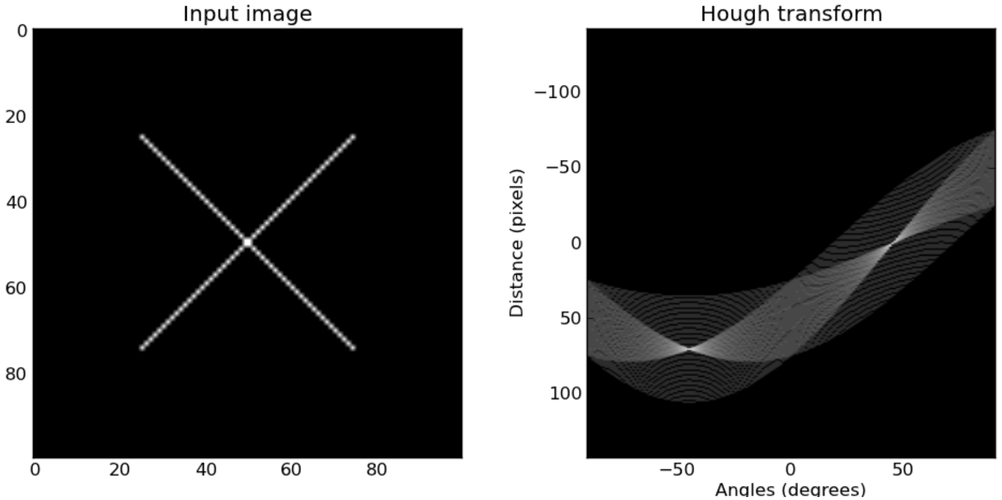
denoising

http://dmmd.net/main_wp/research/denoising/

Image transform

- Transformation is a function. A function that maps one set to another set after performing some operations.
  - Geometric Transform, the shape or geometric domain.
  - Fourier Transform, the spatial frequency composition of the image
  - Hough Transform, the parameters of geometric shapes.

Fourier transform

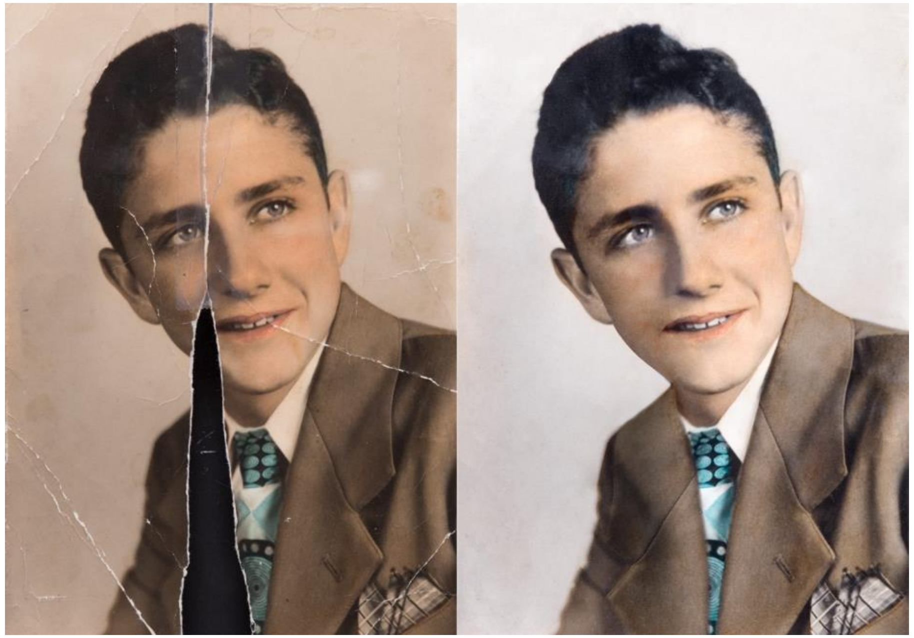


Hough transform

https://scikit-image.org/docs/0.3/auto_examples/plot_hough_transform.html

Image restoration

- Remove or minimize some degradations
  - Deblurring, inpainting, geometric distortion correction, inverse filtering, least mean square (Wiener) filtering

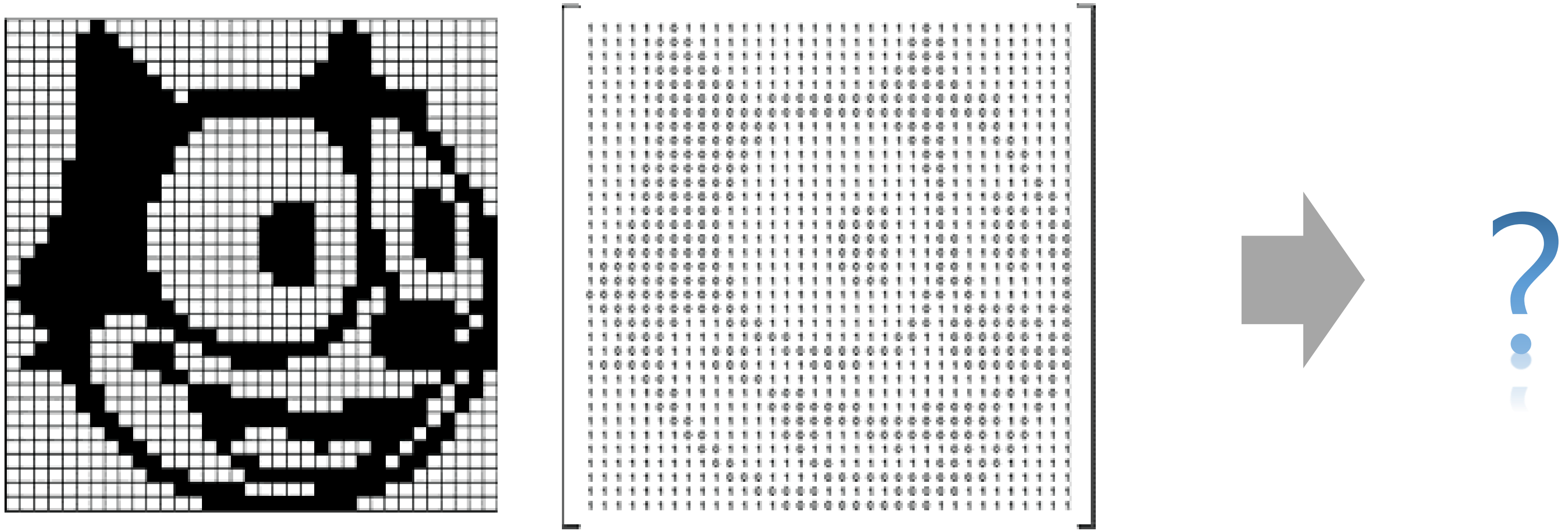


inpainting

colorization

Image processing

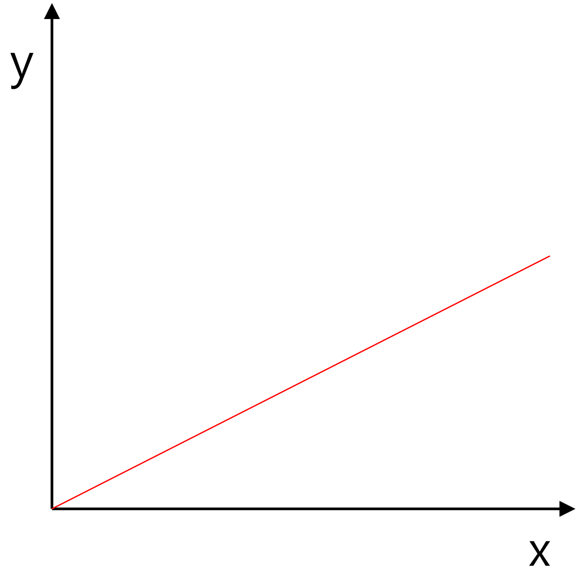
- Summary
  - An image is represented by matrix
  - **In conclusion, image processing is nothing more than changing the element values of the matrix.**



Pixel point processing

- Shape is fixed
- Pixel values are more important than their positions
- Only pixel values relatively changes

- Simple method
  - Compute new value for pixel from its old value
    - y= T(x), T is a linear mapping function
  - In grayscale images, T(x) alters brightness and contrast to compensate for poor exposure, bad lighting, and bring out detail



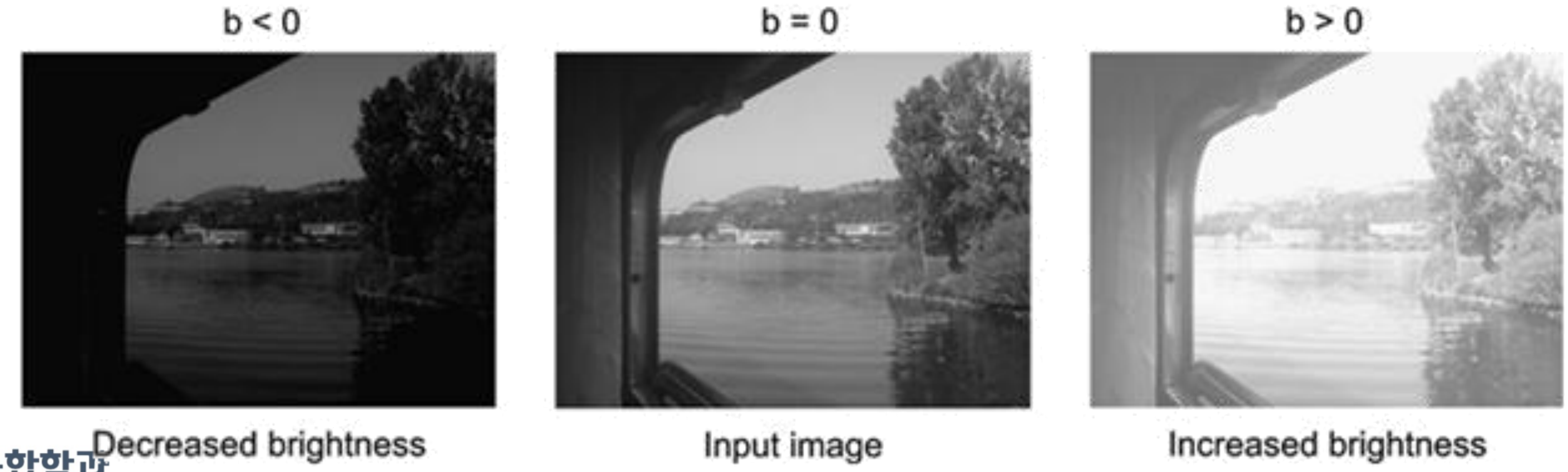
Pixel point processing

- Brightness
  - The brightness is the pixel value itself.
  - We can define the transform function T as adding or subtracting some number from the original value

$$I'(x,y) = I(x,y) + b$$

  - The conversion value must not be outside the range of 0 and 255.

$$I'(x,y) = \mathrm{clip}(0, 255, I(x,y) + b)$$



Pixel point processing

- Contrast
  - an brightness difference between two or more things.
  - Various methods to adjust contrast
    - Linear mapping
    - Exponential
    - Gamma correction
    - Histogram equalization

Low Contrast Image

High Contrast Image

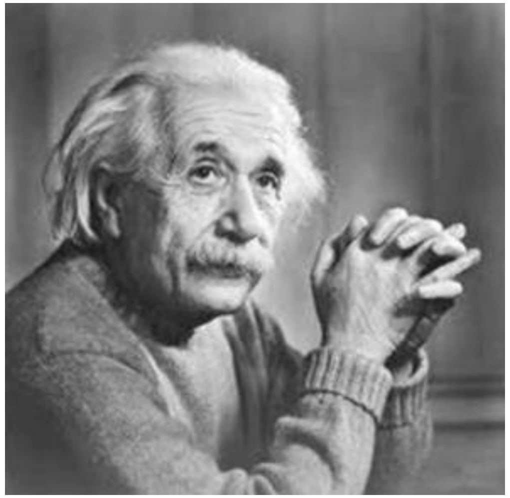
- Linear mapping case
  - We can define the transform function T as multiplying some number from the original value

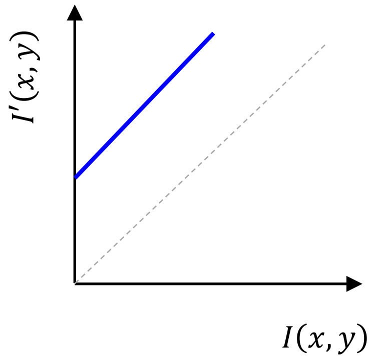
$$I'(x,y) = aI(x,y)$$

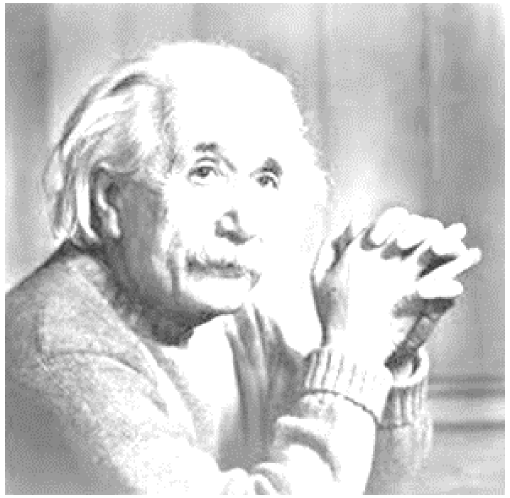
Pixel point processing

- Brightness and Contrast adjustment by linear mapping

$$I'(x, y) = aI(x, y) + b$$







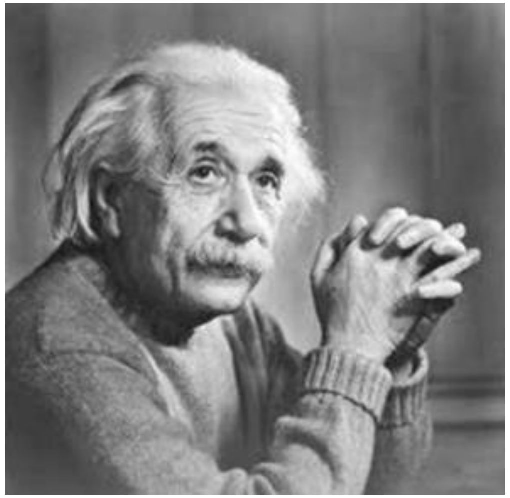
$$b > 0$$

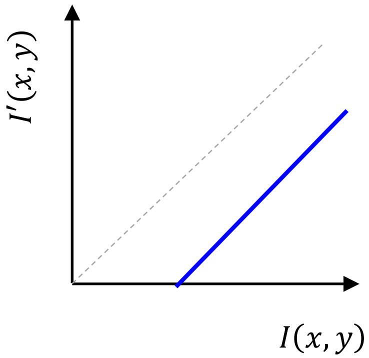
Refer exercise example: 3_image_bright_contrast.py

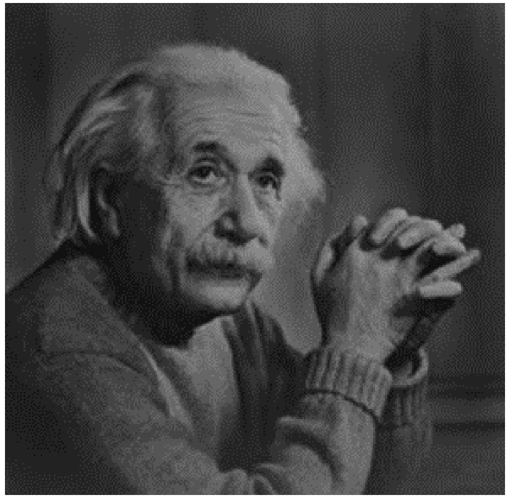
Pixel point processing

- Brightness and Contrast adjustment by linear mapping

$$I'(x,y) = aI(x,y) + b$$





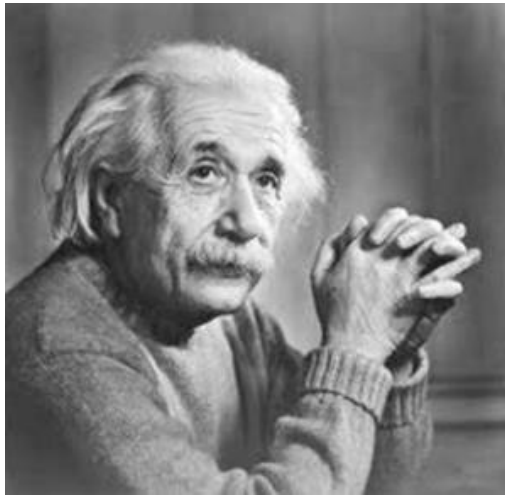


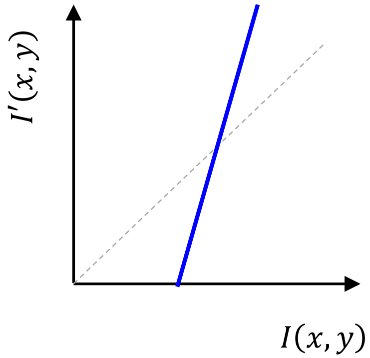
$$b < 0$$

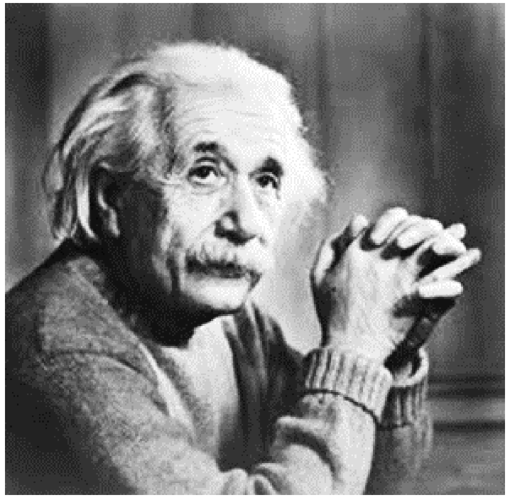
Pixel point processing

- Brightness and Contrast adjustment by linear mapping

$$I'(x,y) = aI(x,y) + b$$





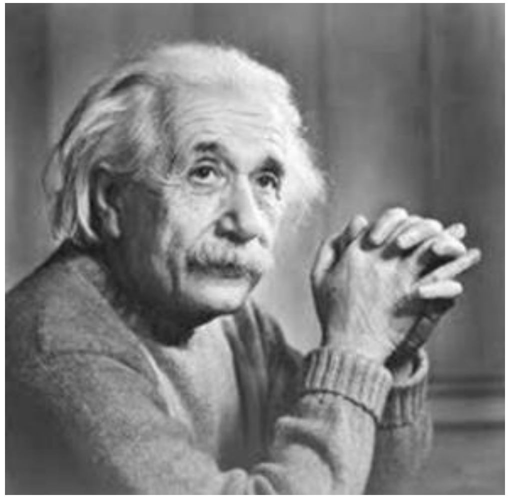


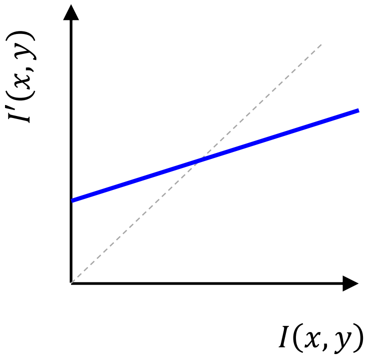
$$a > 1$$

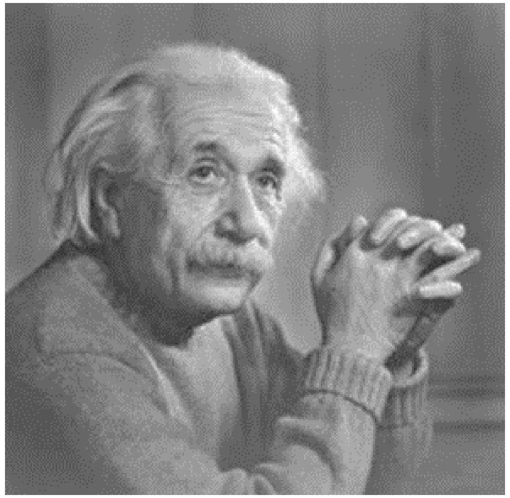
Pixel point processing

- Brightness and Contrast adjustment by linear mapping

$$I'(x,y) = aI(x,y) + b$$





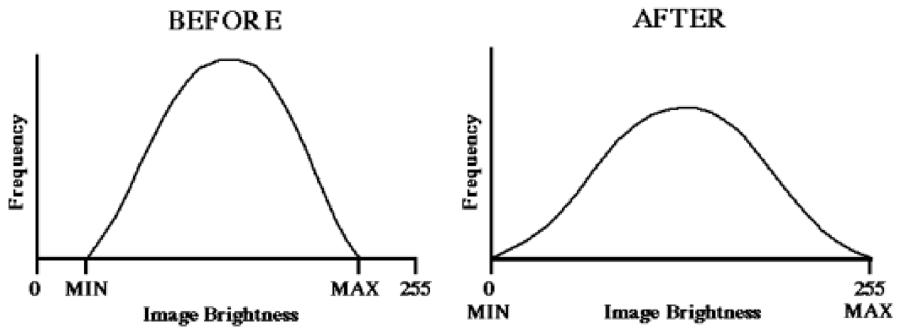


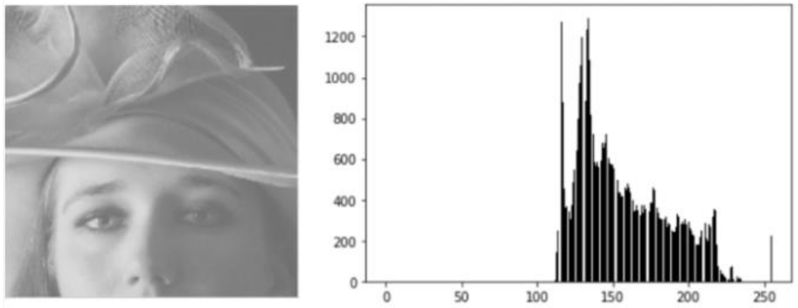
$$a < 0$$

Pixel point processing

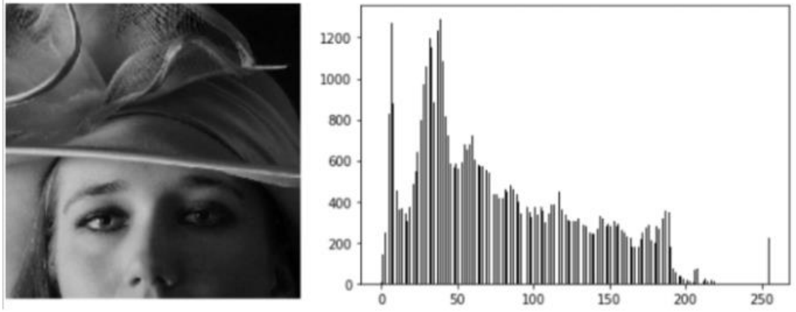
- Brightness and Contrast adjustment by other method
  - Minimum-Maximum linear contrast stretch

$$I'(x,y) = \left(\frac{I_{max} - I_{(x,y)}}{I_{max} - I_{min}}\right) \times 255$$





Input Image before Contrast Stretching with its histogram

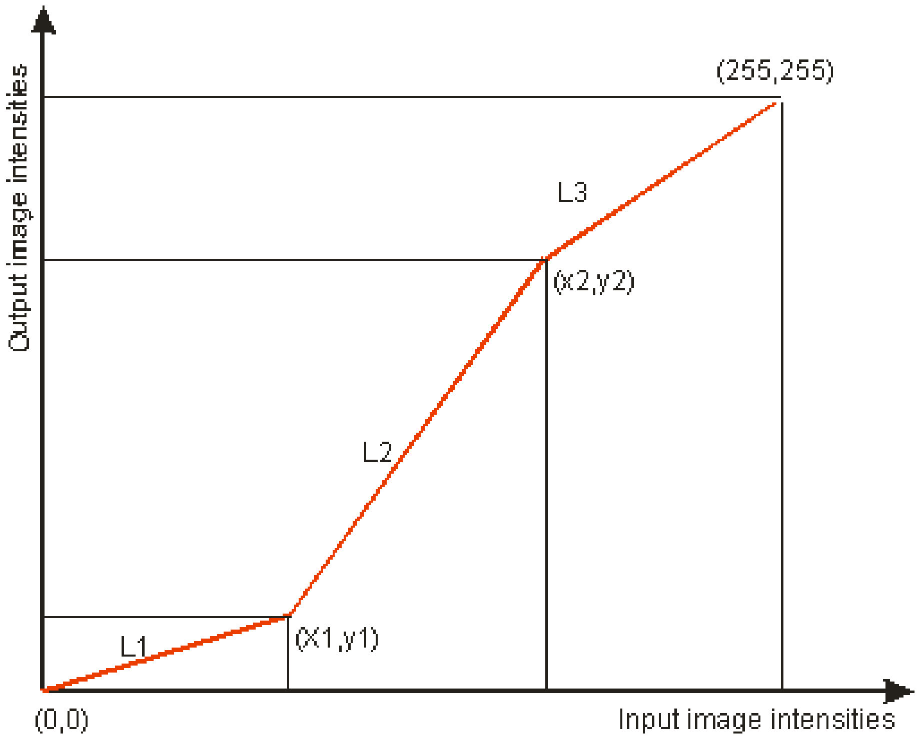


Input Image after Contrast Stretching with its histogram

https://samirkhanal35.medium.com/contrast-stretching-f25e7c4e8e33

Pixel point processing

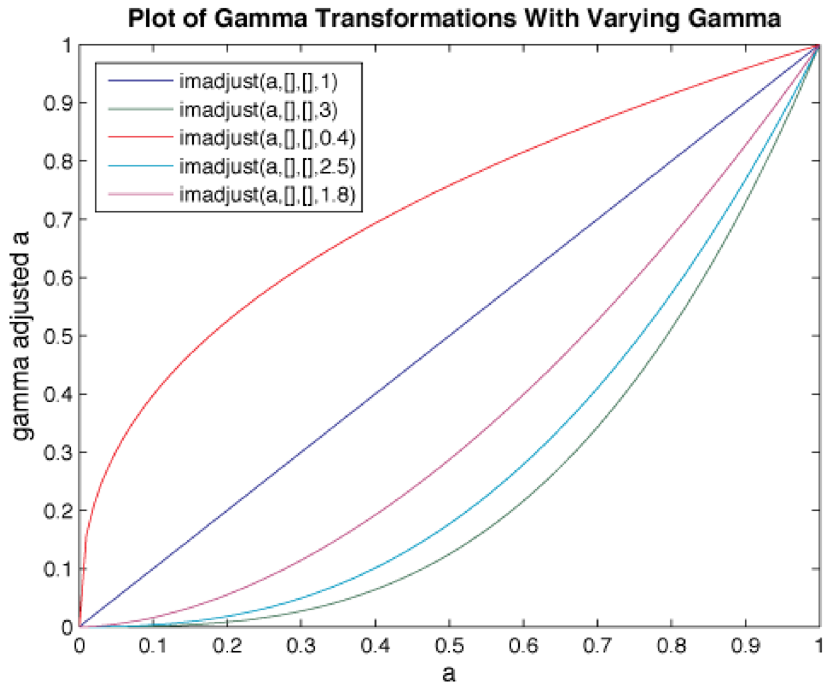
- Brightness and Contrast adjustment by other method
  - Piecewise linear function

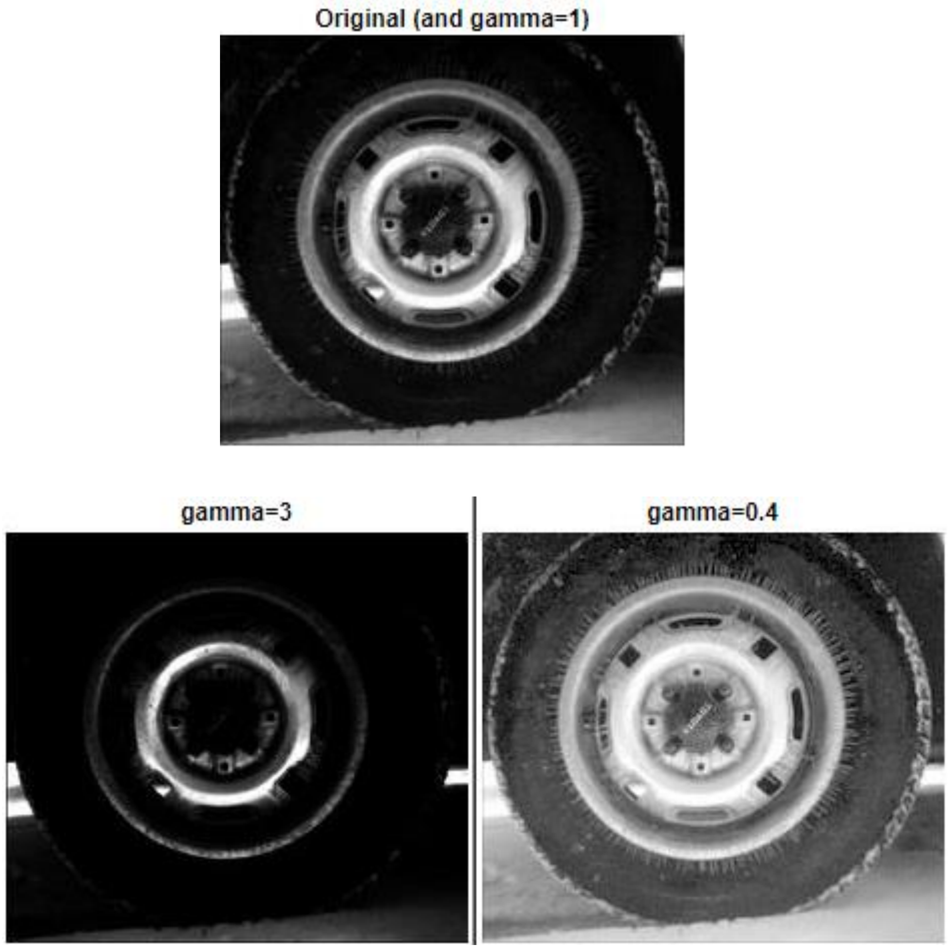


$$y = \begin{cases} \dfrac{y1}{x1}x, & 0 \le x \le x1 \\ \dfrac{y2-y1}{x2-x1}x + y1, & x1 < x < x2 \\ \dfrac{255-y2}{255-x2}x + y2, & x2 < x < 255 \end{cases}$$

Pixel point processing

- Brightness and Contrast adjustment by other method
  - Non-linear mapping function
    - Exponential, Logarithm





Pixel point processing

- Brightness and Contrast adjustment by other method
  - Non-linear mapping function

$$I'^{(x,y)} = \left(\frac{I(x,y)}{255}\right)^{\gamma} \times 255$$

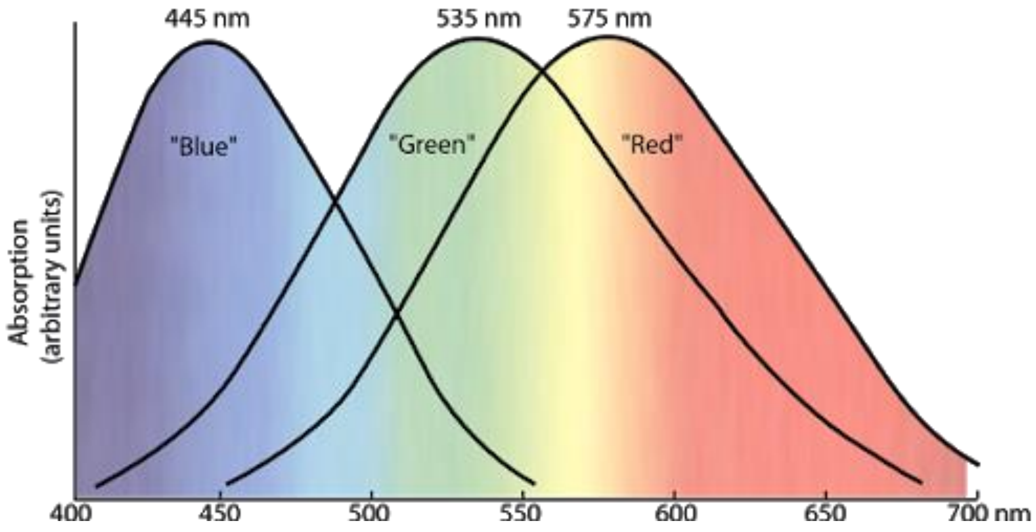
$\gamma = 0.5$

original

$\gamma = 2.5$

Pixel point processing

- Color conversion
  - RGB (3-channels) to Grayscale (1-channel)
  - Intuitive method
    - gray = (R + G + B) / 3
  - Conventional method
    - gray = 0.299R + 0.587G + 0.114B



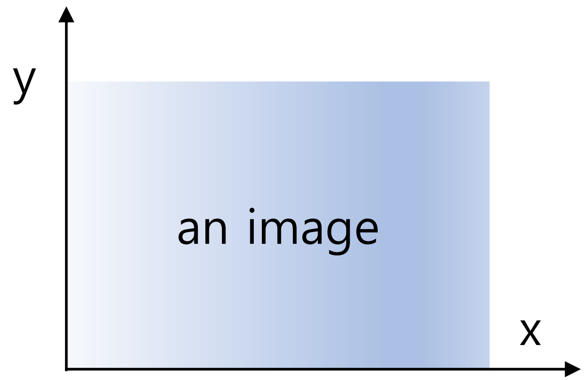
- Inverse conversion is impossible
- [openCV function]
  - gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

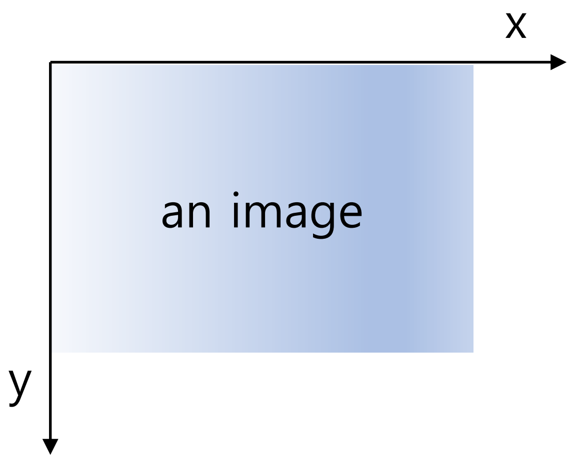
Pratice

- Read Image
- Convert to Grayscale
- Brightness control
- Contrast control : alpha, min-max, gamma

Geometric transform

- Coordinate
  - An image has its independent coordinate system
  - What is the origin and basic axis?





  - Every pixel has their position value (x,y).
  - Geometric transform focus on moving positions of pixels
    → "Where will this pixel be located?"

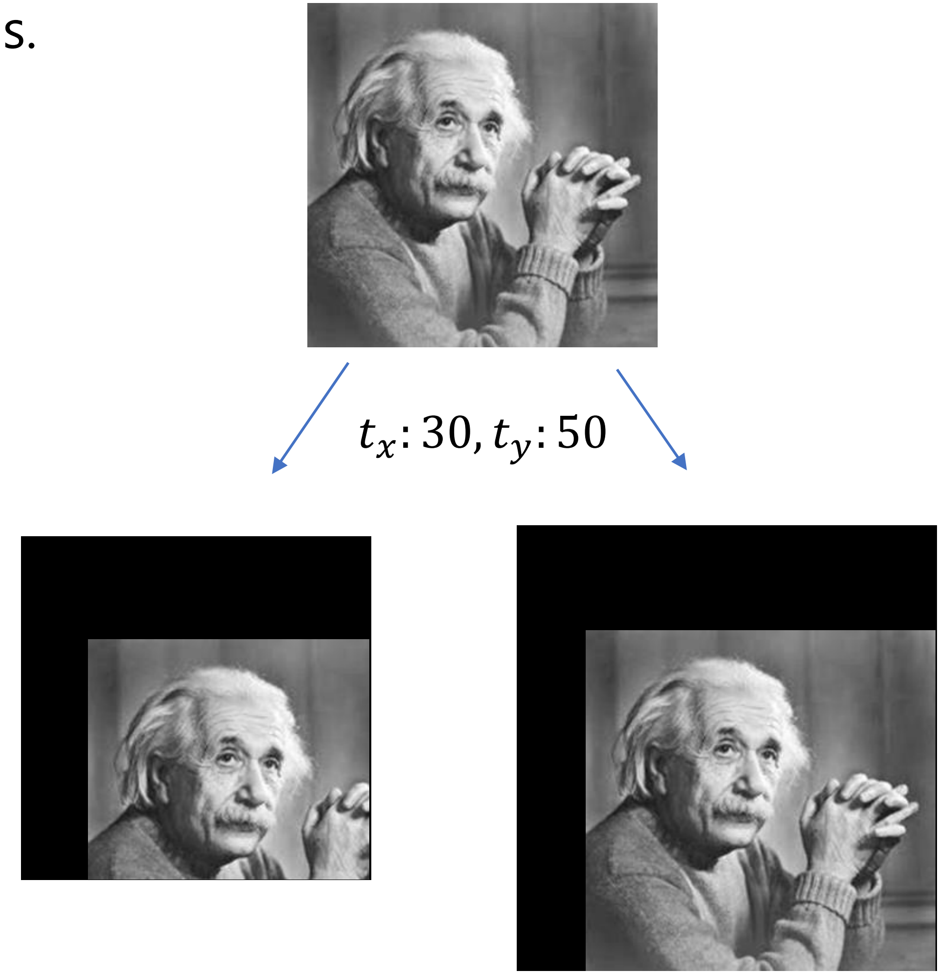
Geometric transform

- Translation
  - Moving positions of pixels along with x, or y-axis.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

    - t_x : amount of translation along the x-axis
    - t_y : amount of translation along the y-axis

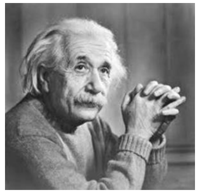
  - Be careful about going out of the image area.
    - Pixels outside the area are ignored.
    - Increasing the image size



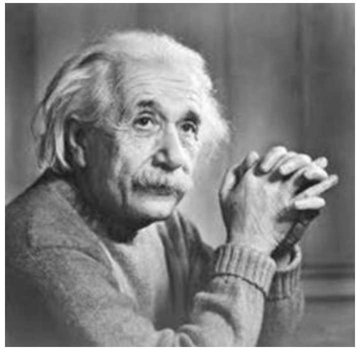
Geometric transform

- Scaling
  - Resizing of a digital image
  - Up-scale, Zoom-in: Increase the dimensions of the original image
  - Down-scale, Zoom-out: Decrease the dimensions of the original image
  - By simply multiplication a scalar value (scale factor) to the original position.

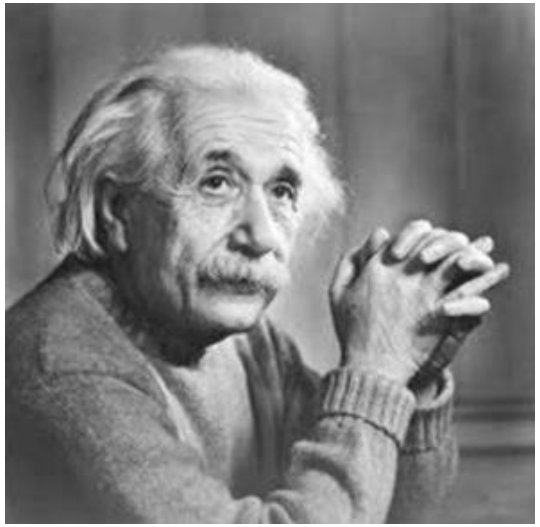
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = s \begin{bmatrix} x \\ y \end{bmatrix}$$



$0< s < 1$



$s > 1$

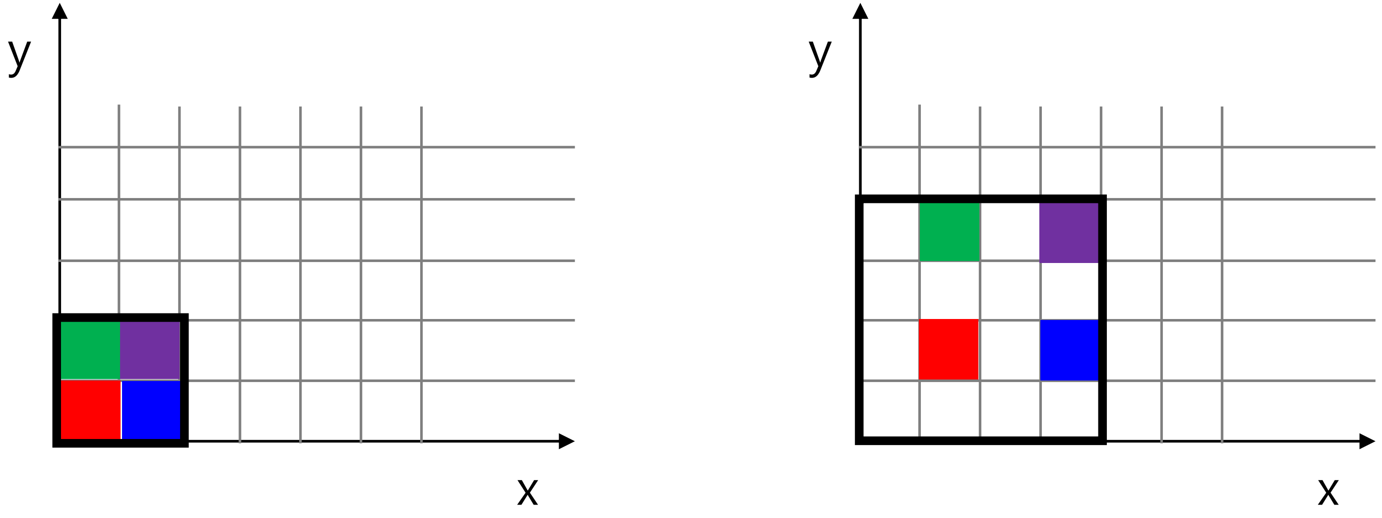


- Is there no problem?

Geometric transform

- Problems of scaling
  - The amount of information is preserved.
  - Dimension reduction is possible
  - But Generating more information is impossible.
  - Example) Upscale to 2x → The value of the empty pixel is unknown.

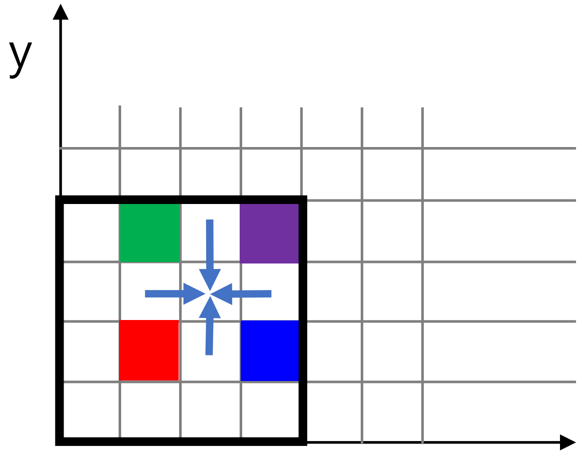
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = 2 \begin{bmatrix} x \\ y \end{bmatrix}$$



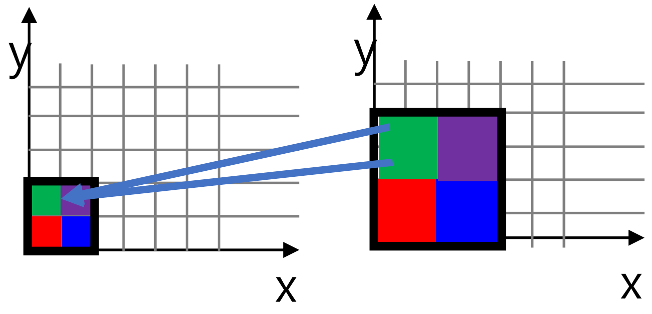
Geometric transform

- Problems of scaling and solutions
  - Interpolation
    - Estimate a value by referencing values of neighborhood.
    - It is necessary to determine which pixels to interpolate.
  - Backward warping
    - For each pixel in the reference image, take a pixel from the source image using an inverse transform.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{s} \begin{bmatrix} x' \\ y' \end{bmatrix}$$



interpolation

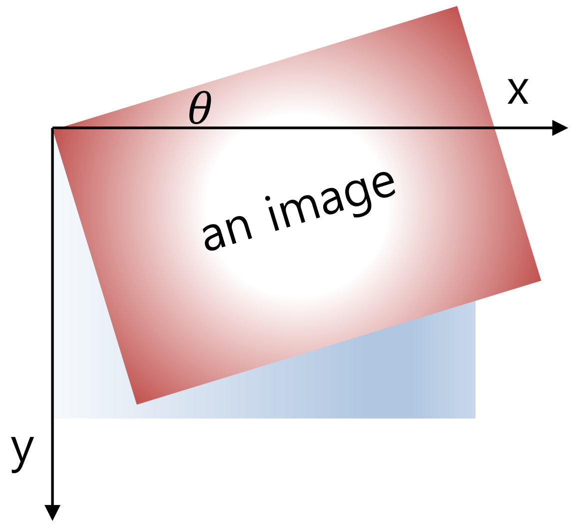


Backward warping

Geometric transform

- Image rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms.
- The input to an image rotation routine is an image, the rotation angle θ, and a point about which rotation is done.

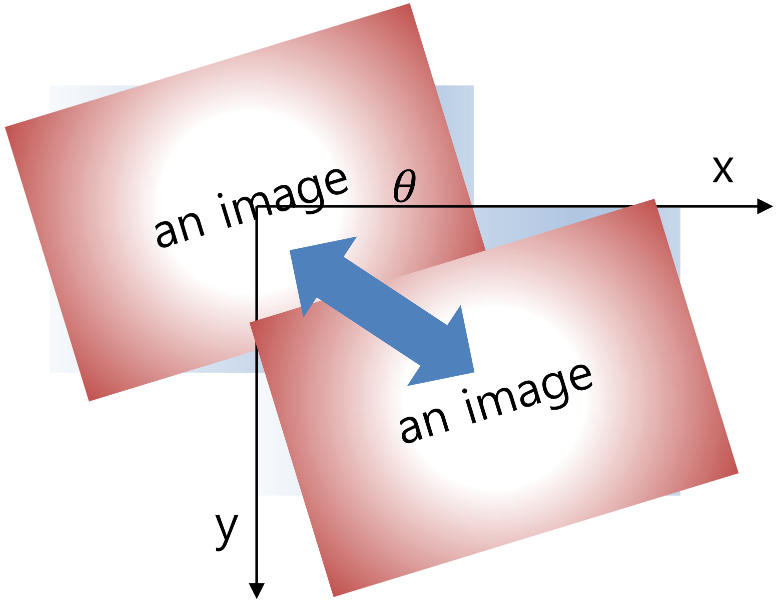
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Geometric transform

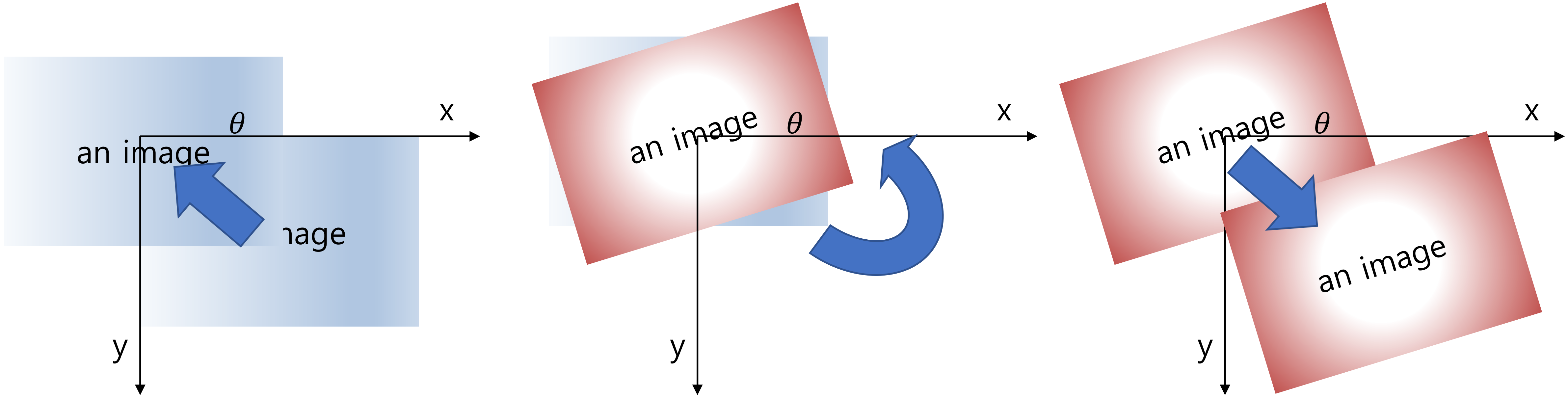
- The rotation equation is based on the origin of the coordinate axis.
- To rotate around the center of the image, use the following equation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - \frac{w}{2} \\ y - \frac{h}{2} \end{bmatrix} + \begin{bmatrix} \frac{w}{2} \\ \frac{h}{2} \end{bmatrix}$$



Geometric transform

- The rotation equation is based on the origin of the coordinate axis.
- To rotate around the center of the image, use the following equation.



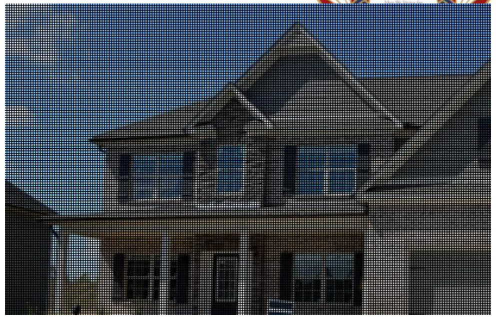
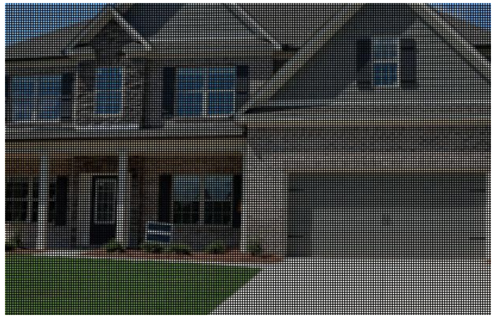
1.translation

2. rotation

3. translation

Practice 2

- Geometric transform
  - Translation
  - Scaling
    - Forward transform based on (0,0)
    - Forward transform based on image center (Mission)
    - Backward transform based on image center (Mission)
  - Rotation (center)
    - Forward
    - Backward (mission)

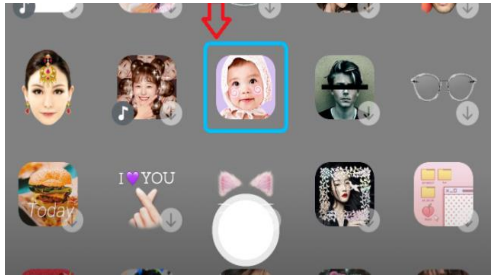




Spatial filter

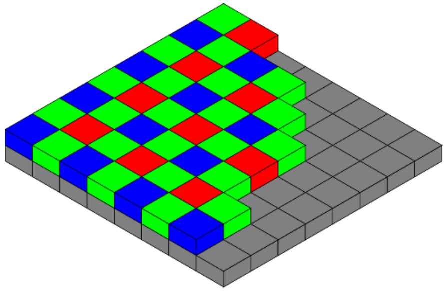
- Filters
- Convolution
- Spatial filters
  - Blur filter (moving average, Gaussian)
  - Edge detection

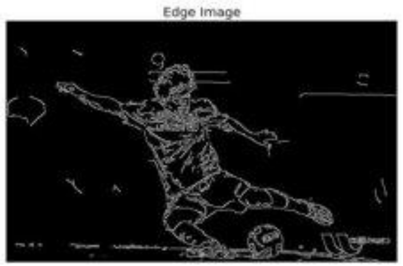
Filters

- Filters around us do
  - Separation
  - Removal
  - Conversion



- Filter in Image Processing
  - Cut specific bandwidth of light
  - Modify image (Photoshop filters)

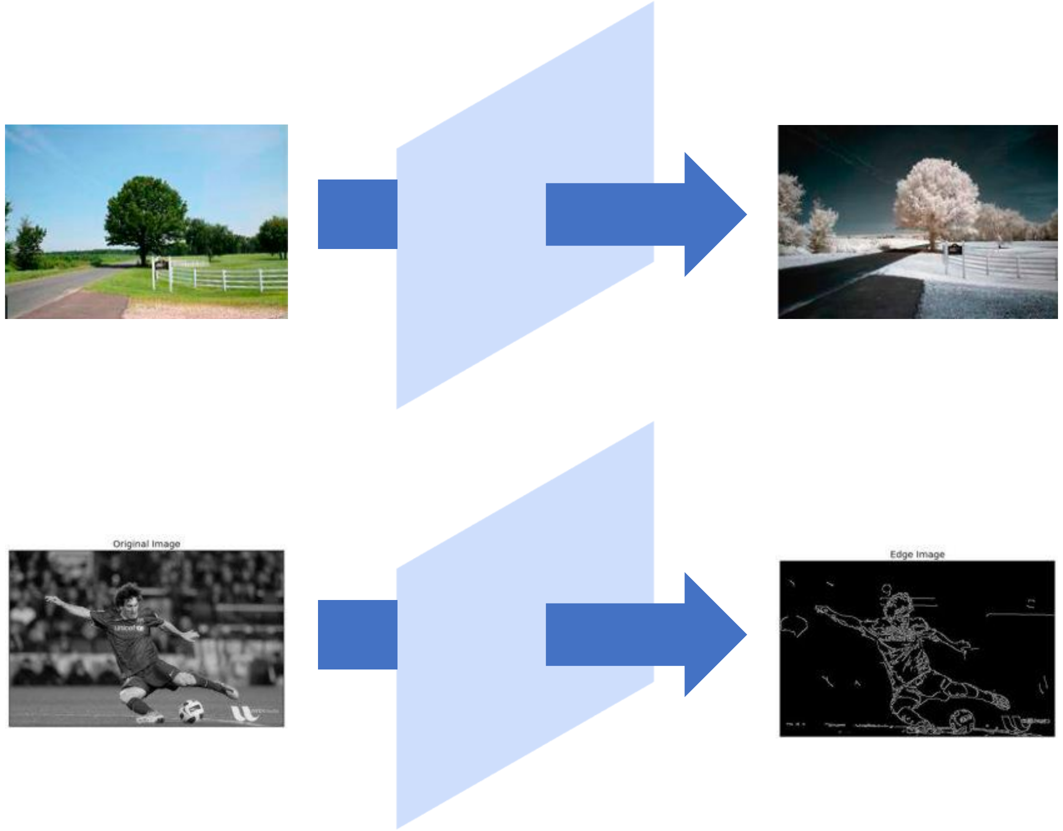




http://biz.newdaily.co.kr/site/data/html/2020/07/24/2020072400010.html

Filters

- A filter might be holistic? or fractional?
  - A holistic filter is actually point-wise operating filter
  - Spatial filter needs a specific operation



Possible?

Possible?

Spatial Filters

- Spatial filters used different **masks** (**kernels**, templates or windows)
- The mechanics of spatial filtering consists of:
  - Neighborhood (small rectangle)
  - Predefined operation that is performed on the image pixel
- Filtering creates new pixel with coordinates equal to the coordinates of the center of the neighborhood
- Spatial filters of image processing
  - The same small size filter is applied through **convolution**.

Convolution

- Definition
  - A mathematical operation on two functions *f* and *g*, producing a third function that is typically viewed as a modified version of one of the original functions, giving the **integral of the pointwise multiplication of the two functions** after one is reversed and shifted.

definition $$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau.$$

discrete convolution $$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Convolution

- Equation to concept.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

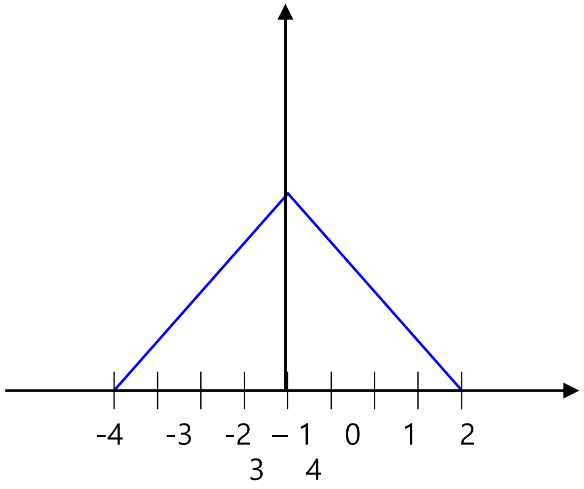
- * : operator
- [n]: result of [n] position of a NEW signal
- ∑: sum of all range
- m: position of an original position of each signal

- Hint: convolution → 합성곱

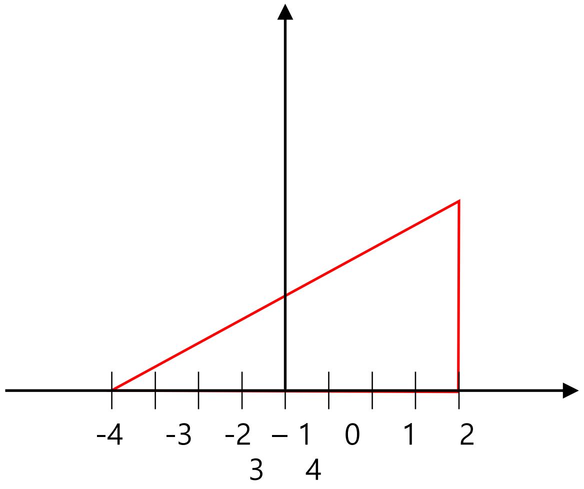
Convolution

- Equation to concept.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$



$f(m)$

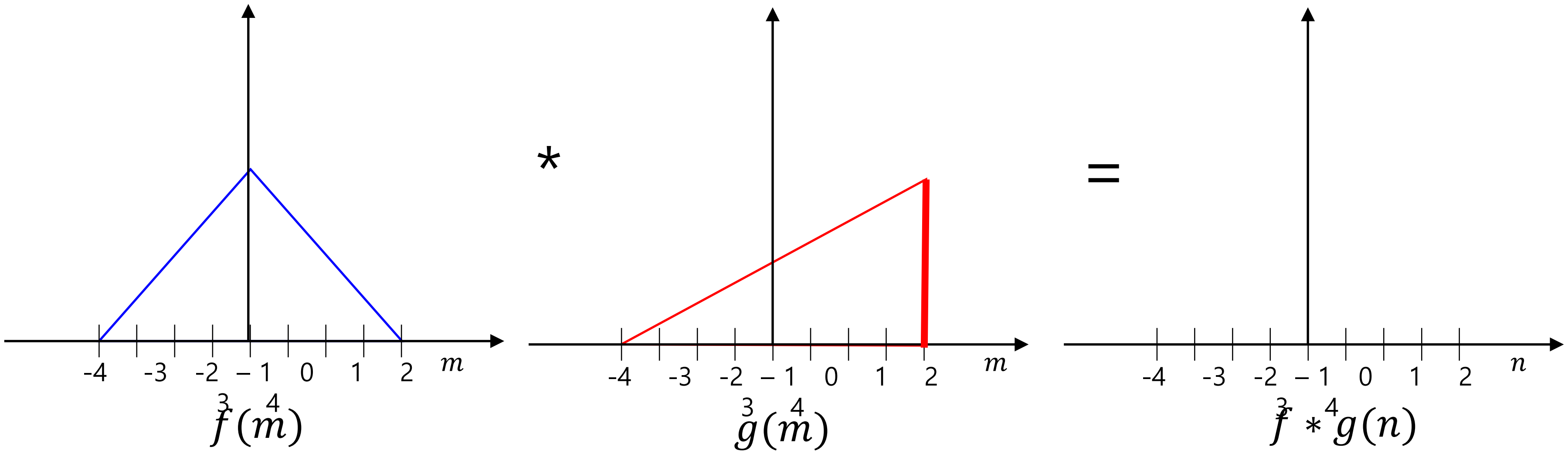


$g(m)$

convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

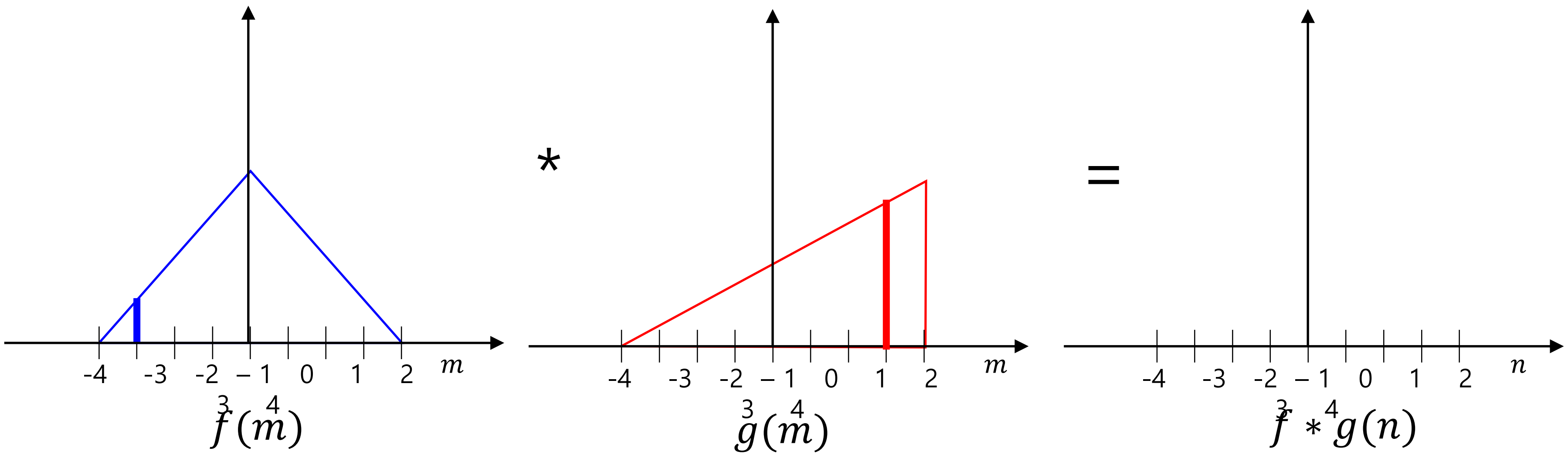
- when n=0
  - m= -4 : f(-4)g(4) = 0



convolution

- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14

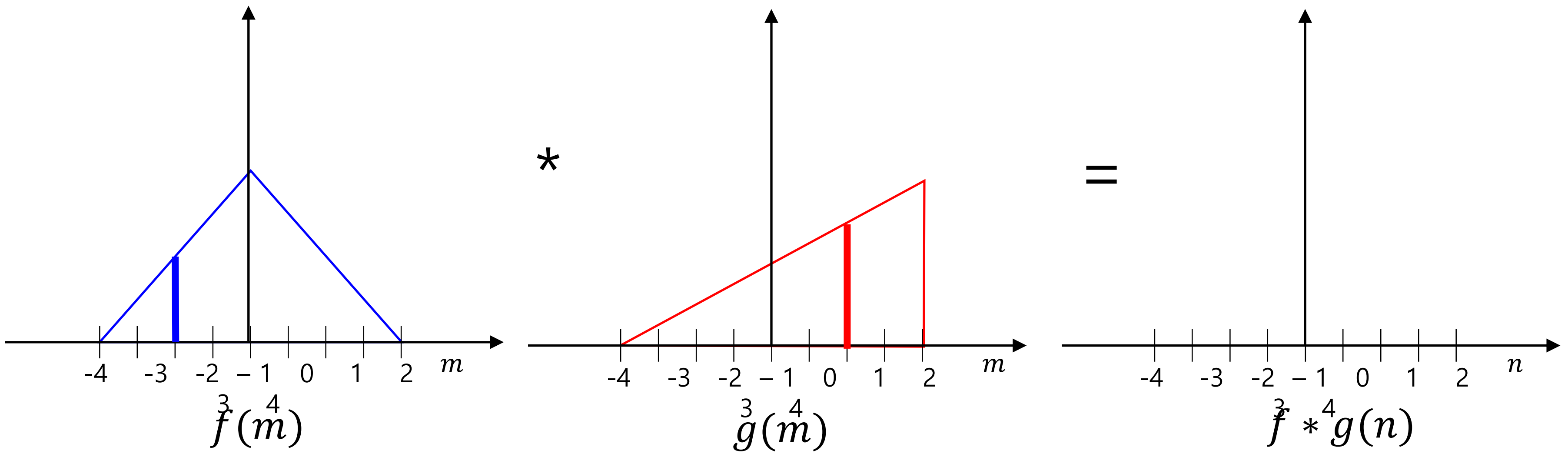
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$



convolution

- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3

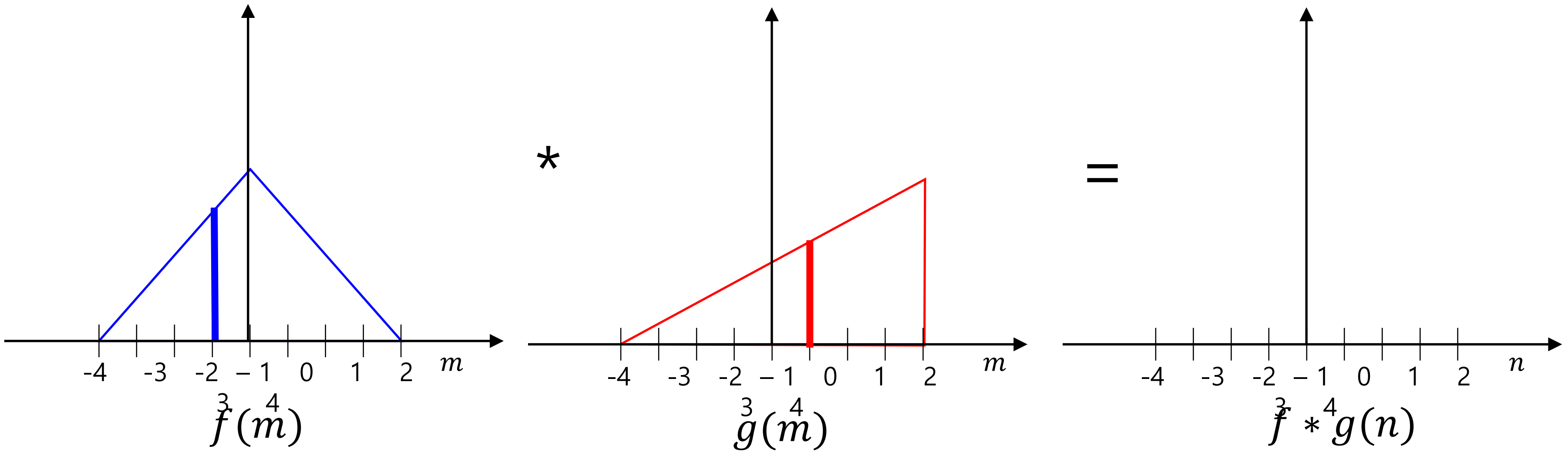
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$



convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

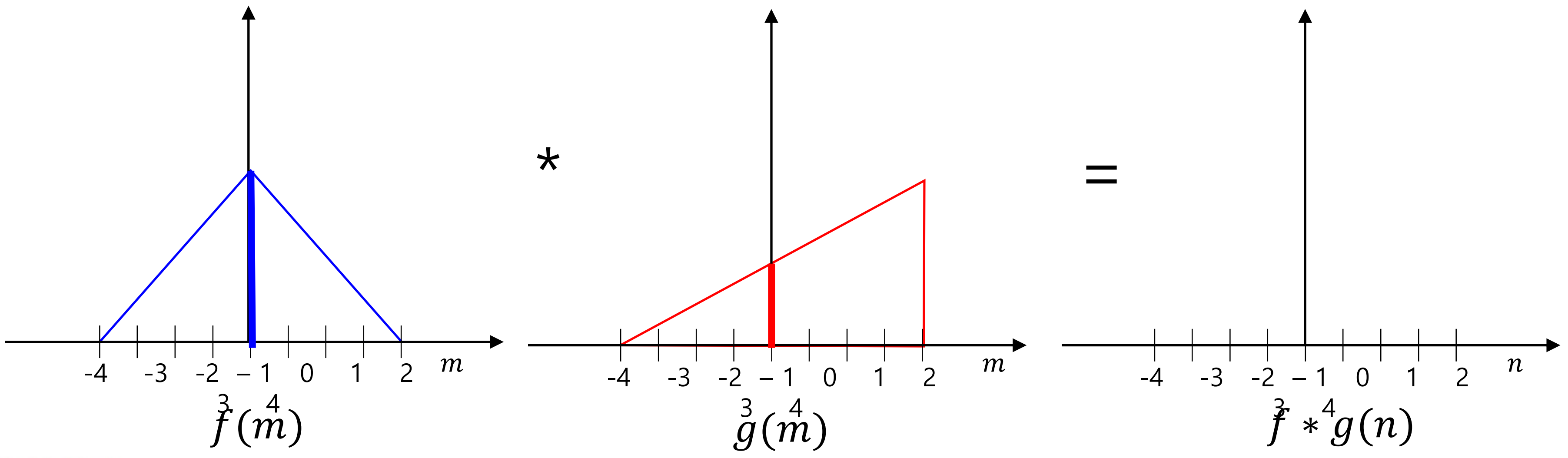
- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42



convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

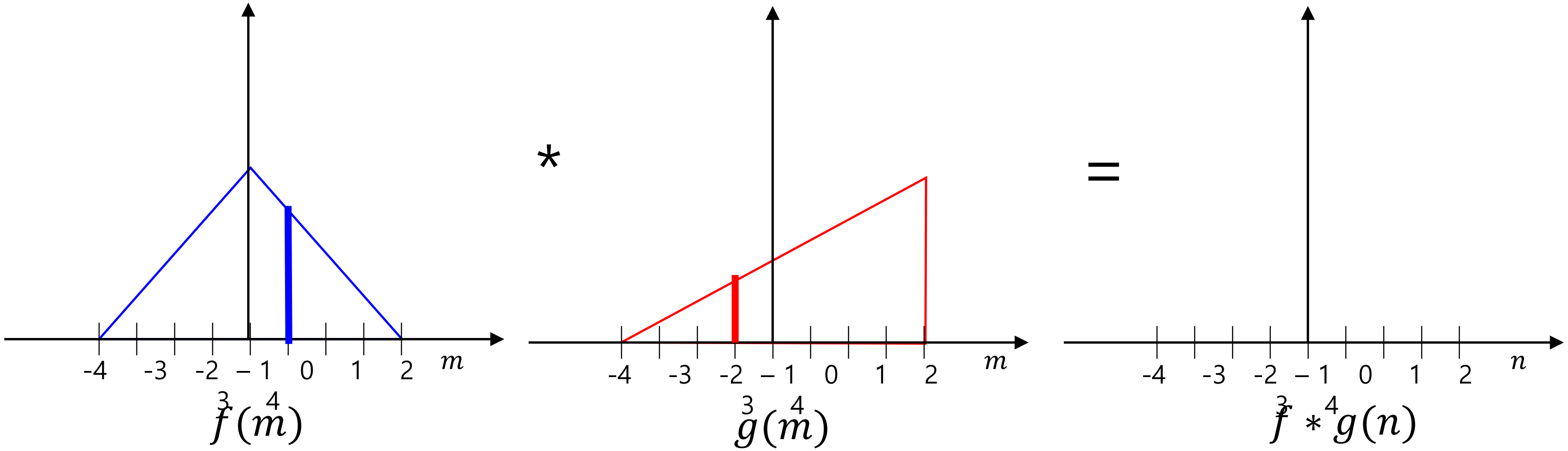
- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42
  - m= 0 : f(0)g(0) = 0.5



convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

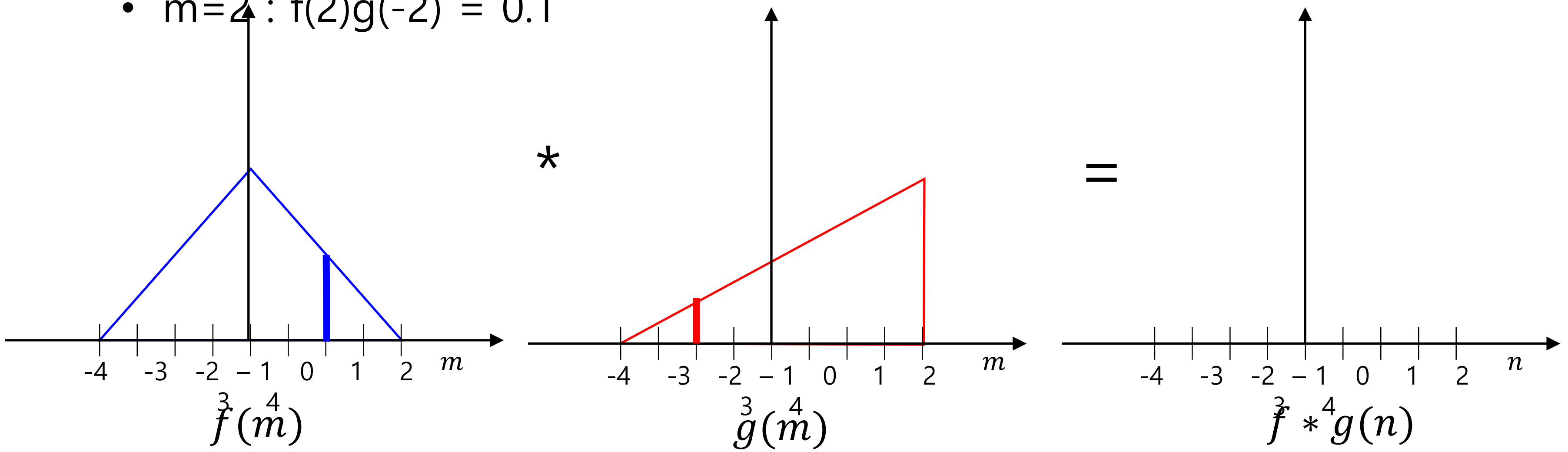
- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42
  - m= 0 : f(0)g(0) = 0.5
  - m= 1 : f(1)g(-1) = 0.32



convolution

- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42
  - m= 0 : f(0)g(0) = 0.5
  - m= 1 : f(1)g(-1) = 0.32
  - m=2 : f(2)g(-2) = 0.1

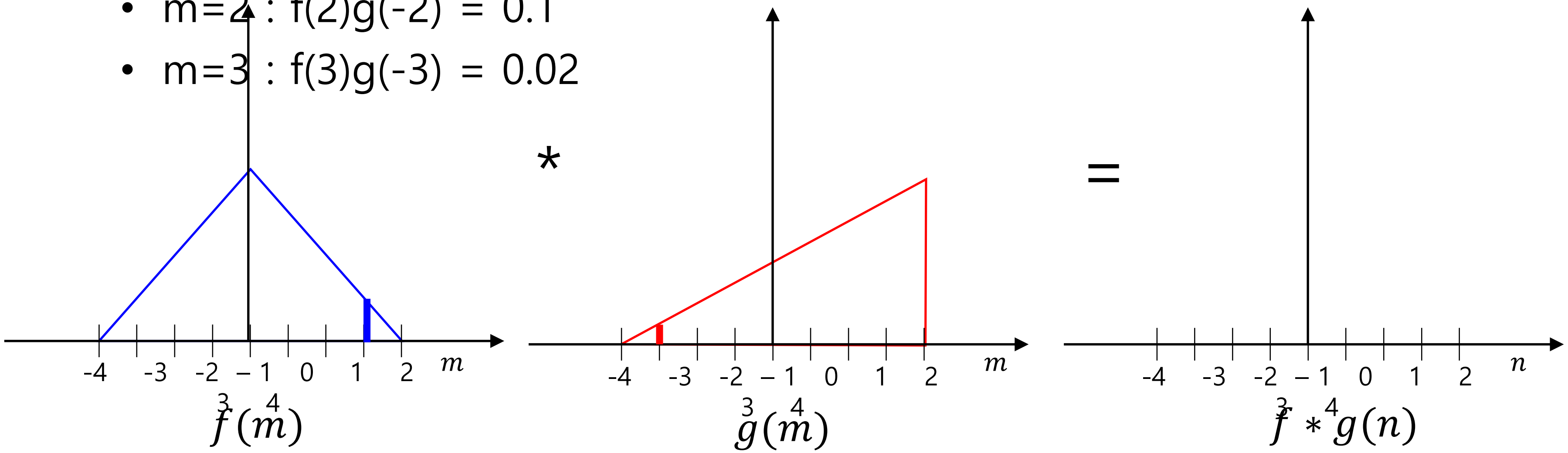
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$



convolution

- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42
  - m= 0 : f(0)g(0) = 0.5
  - m= 1 : f(1)g(-1) = 0.32
  - m=2 : f(2)g(-2) = 0.1
  - m=3 : f(3)g(-3) = 0.02

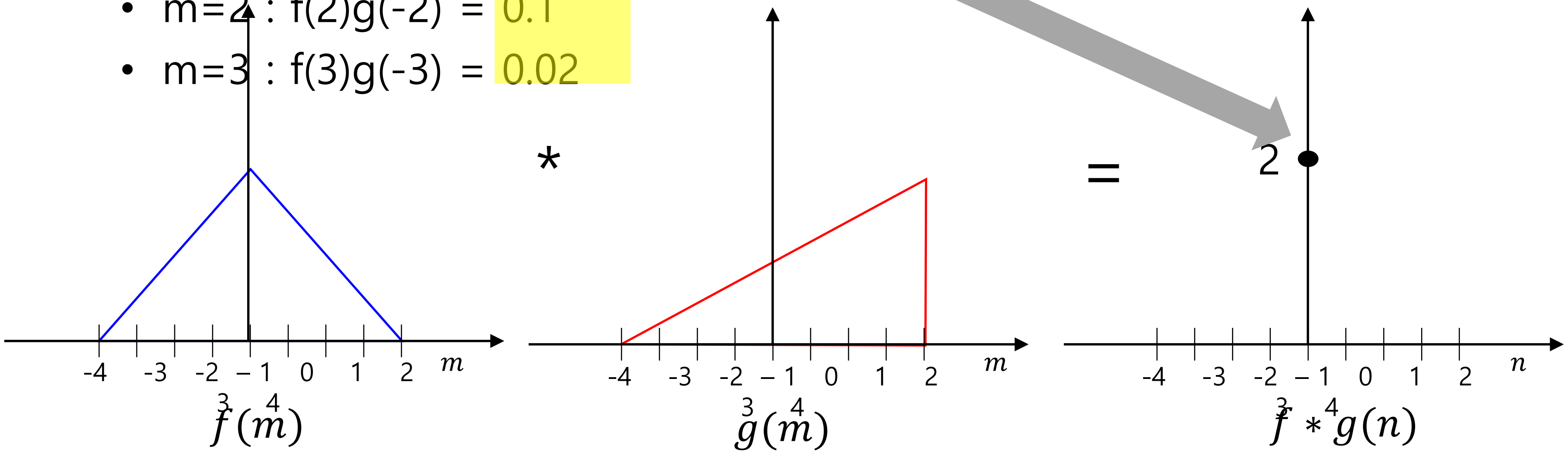
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$



convolution

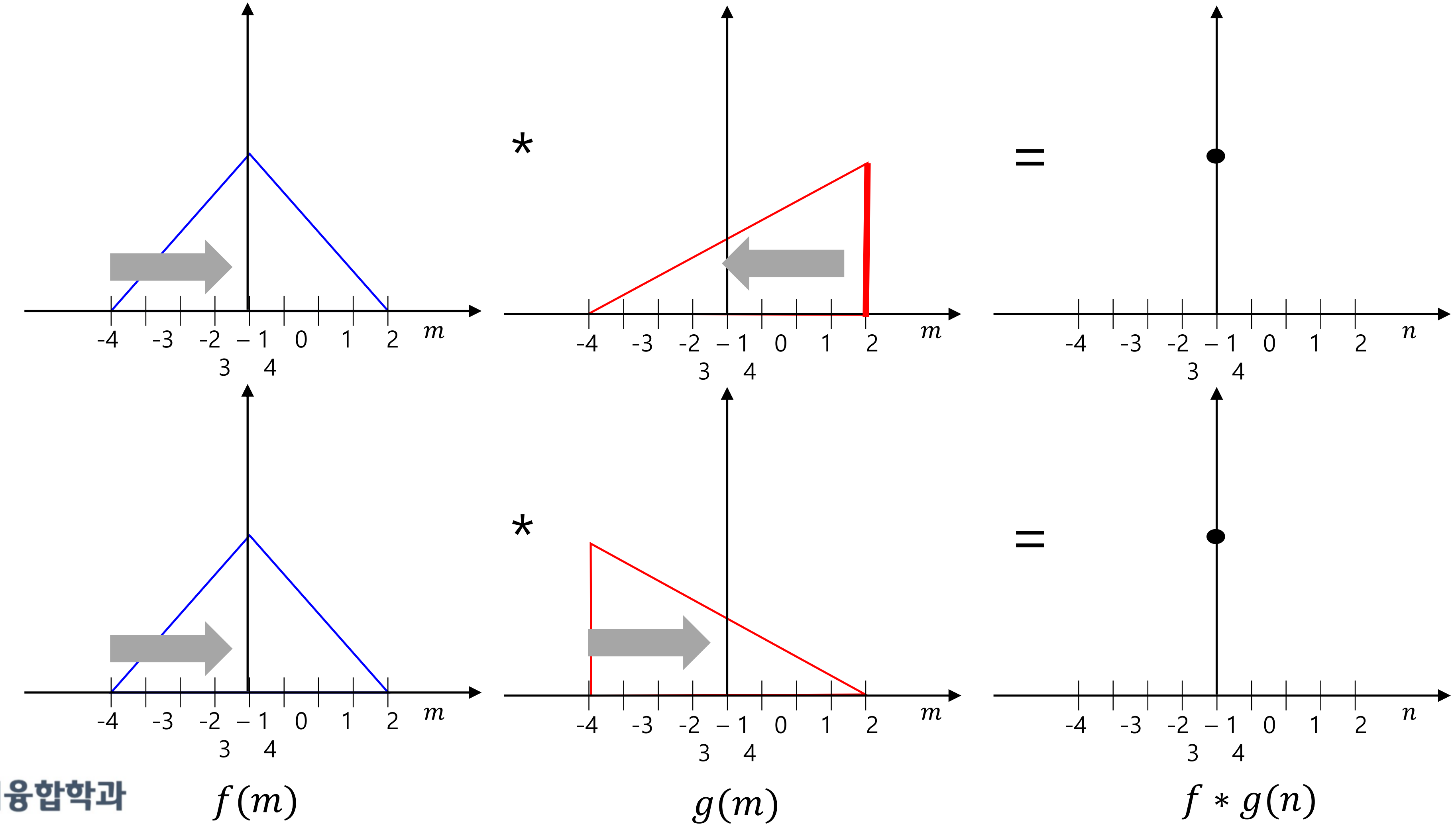
$$(f * g)[n] = \boxed{\sum_{m=-\infty}^{\infty}} f[m]g[n-m]$$

- when n=0
  - m= -4 : f(-4)g(4) = 0
  - m= -3 : f(-3)g(3) = 0.14
  - m= -2 : f(-2)g(2) = 0.3
  - m= -1 : f(-1)g(1) = 0.42
  - m= 0 : f(0)g(0) = 0.5
  - m= 1 : f(1)g(-1) = 0.32
  - m=2 : f(2)g(-2) = 0.1
  - m=3 : f(3)g(-3) = 0.02



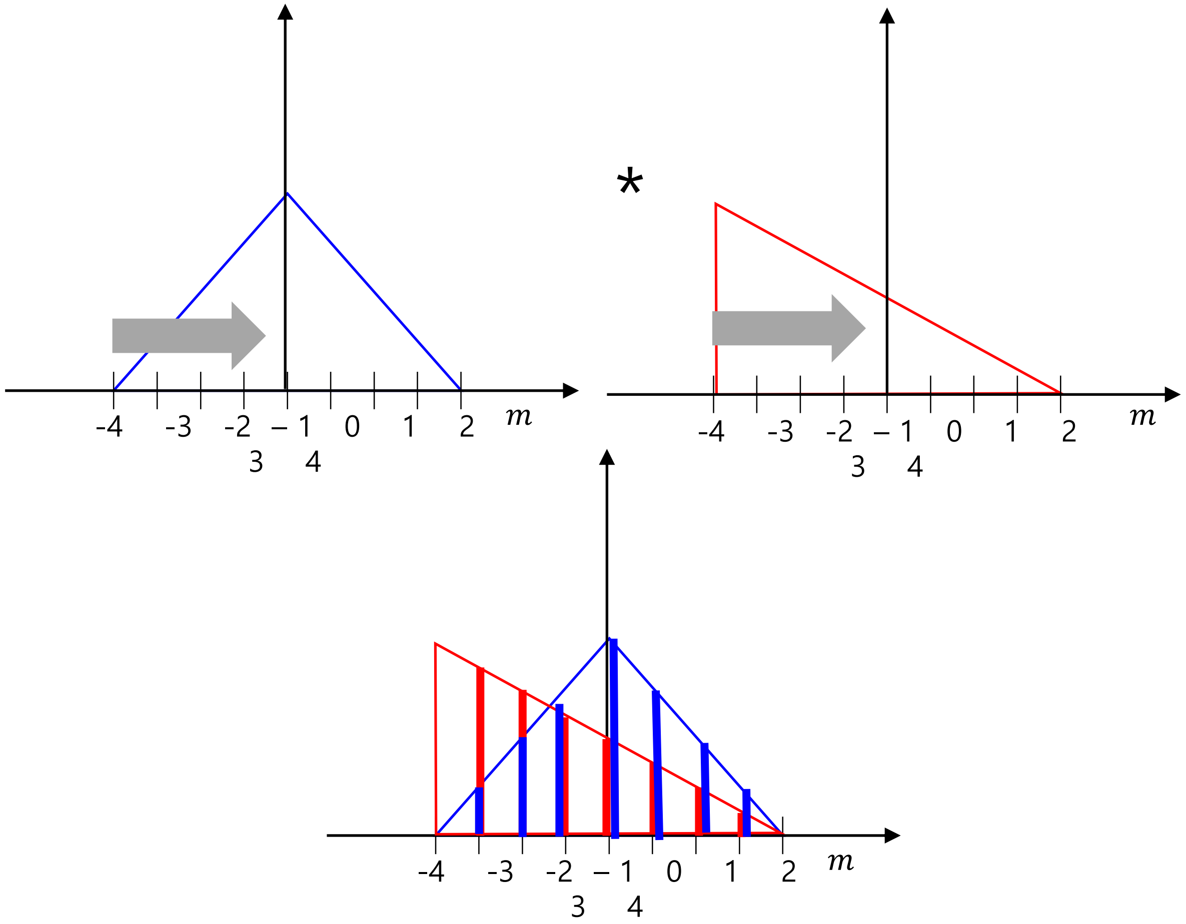
convolution

- when n=0



convolution

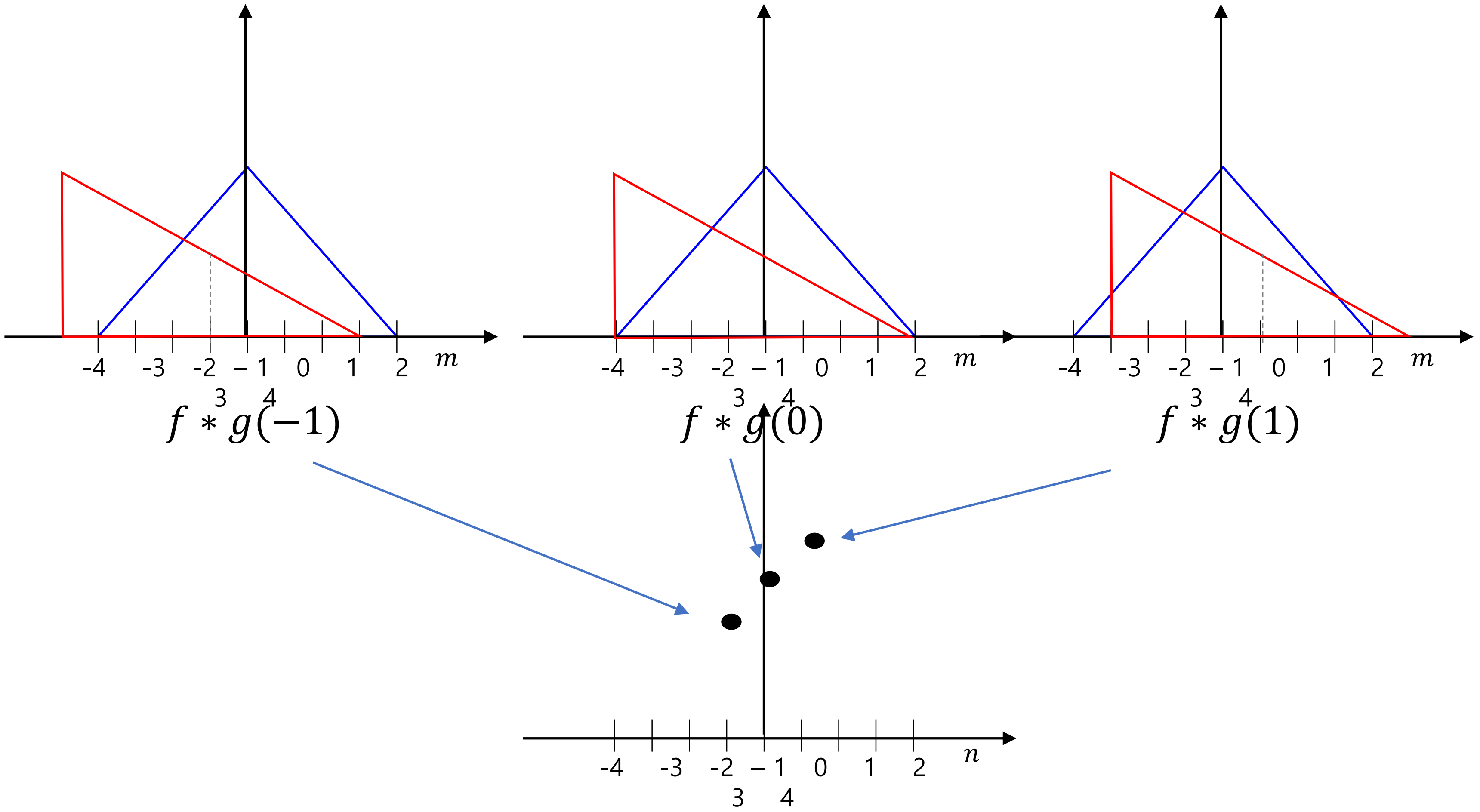
- when n=0



$f * g(0)$

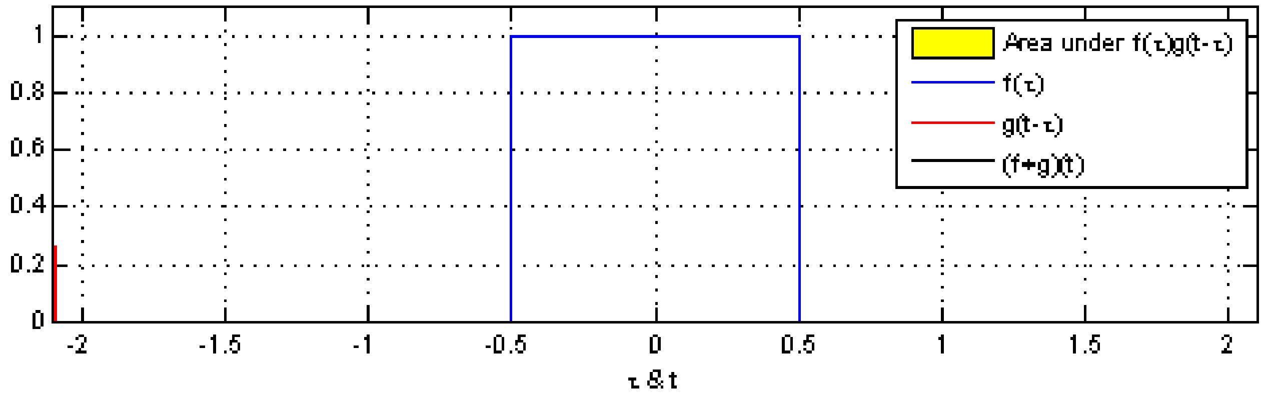
convolution

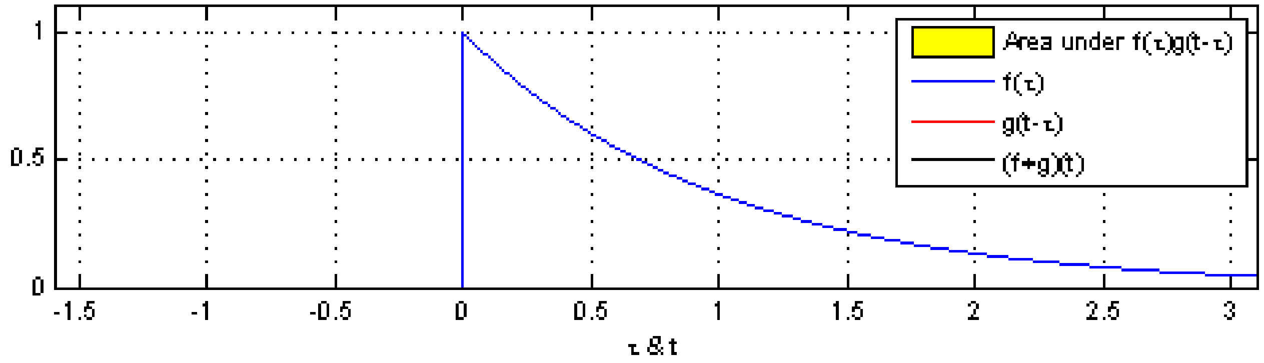
- when n=-1, 0, 1…



convolution

- Example of convolution





https://en.wikipedia.org/wiki/Convolution

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

1	0	1
2	-1	3
2	1	0

$$f * g = ?$$

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

0	1	2
3	-1	2
1	0	1

- Flip the function
- flip around the x-axis, then flip around the y-axis = transpose matrix

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

0	1	2
3	-1	2
1	0	1

Sum the element-wise multiplication
= 1x0 + 2x1 + 3x2 + 2x3 + 3x(-1) + 4x2 + 3x1 + 4x0 + 5x1 = 22

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

0	1	2
3	-1	2
1	0	1

Sum the element-wise multiplication
= 2x0 + 3x1 + 4x2 + 3x3 + 4x(-1) + 5x2 + 4x1 + 5x0 + 6x1 = 30

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

0	1	2
3	-1	2
1	0	1

Sum the element-wise multiplication
= 3x0 + 4x1 + 5x2 + 4x3 + 5x(-1) + 6x2 + 5x1 + 6x0 + 7x1 = 45

2D convolution

- An example of 2D convolution

f

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
1	3	5	7	9
2	4	6	8	1

g

0	1	2
3	-1	2
1	0	1

Sum the element-wise multiplication
= 2x0 + 3x1 + 4x2 + 3x3 + 4x(-1) + 5x2 + 1x1 + 3x0 + 5x1 = 32

Spatial filters

- Moving average in 2D
  - Blur filter
  - Note. The summation of filter values should be 1

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

$$\frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

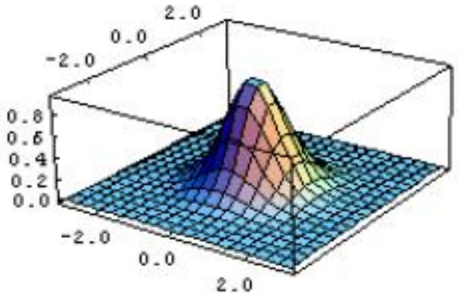
Spatial filters

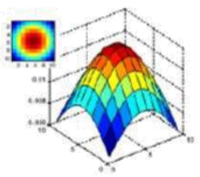
- Gaussian filter
  - Blur filter (Gaussian blur)
  - The kernel is the approximation of a Gaussian function

$$H[u,v]\,\frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{2\sigma^2}}$$

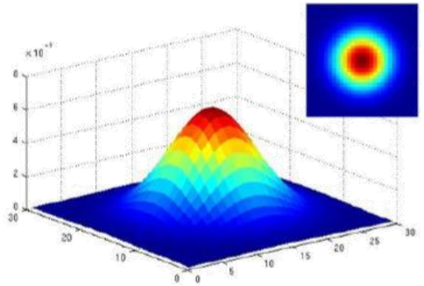
$$\frac{1}{16}\boxed{\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}}$$

$H[u,v]$

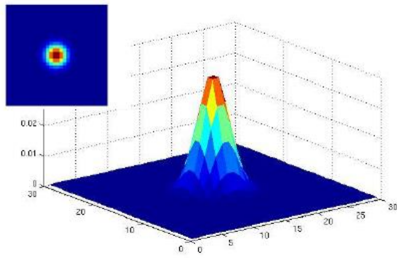




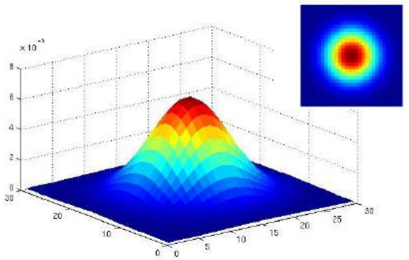
σ = 5 pixels
with 10 × 10 pixel kernel



σ = 5 pixels
with 30 × 30 pixel kernel



σ = 2 pixels
with 30 × 30 pixel kernel



σ = 5 pixels
with 30 × 30 pixel kernel

Edge detection

- Differentiation and convolution
  - Edges on images are high frequency → large gradients

- For a 2D function I(x,y) the partial derivative along x is:

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

- For discrete data, we can approximate using finite differences:

$$\frac{\partial I(x,y)}{\partial x} \approx \frac{I(x+1, y) - I(x,y)}{1}$$

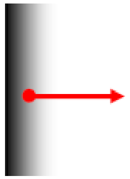
- What would be the respective filters along x and y to implement the partial derivatives as a convolution?

Edge detection

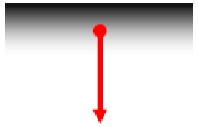
- Direction, magnitude of the gradient
- The gradient of an image:

$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

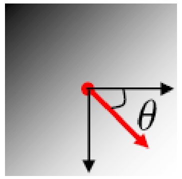
- The gradient points in the direction of fastest intensity change



$$\nabla I = \left[\frac{\partial I}{\partial x}, 0\right]$$



$$\nabla I = \left[0, \frac{\partial I}{\partial y}\right]$$



$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

- The gradient direction is given by:

$$\theta = atan2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

- The edge strength is given by the gradient magnitude

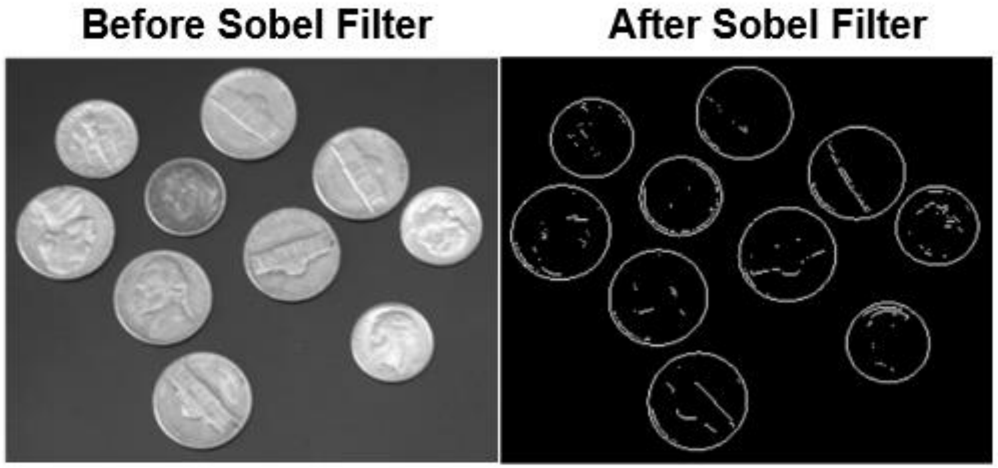
$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Edge detection

- Finite-difference filters

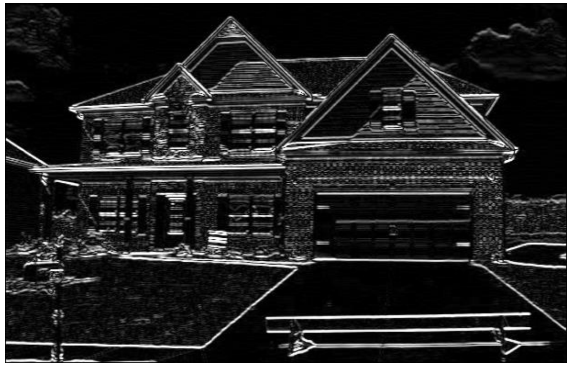
Prewitt filter $G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$ and $G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$

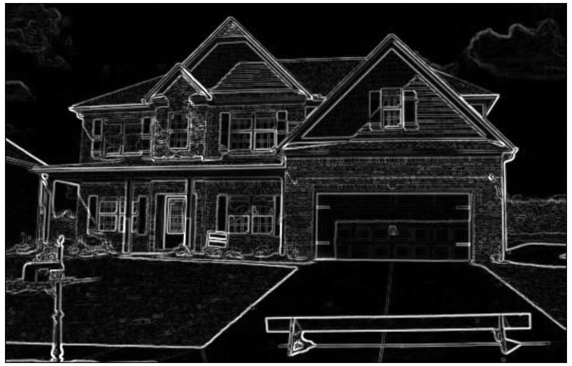
Sobel filter $G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$ and $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$



Practice 3

- Spatial filter
  - Convolution function (mission)
  - Blurring
  - Edge Detection





Thank you