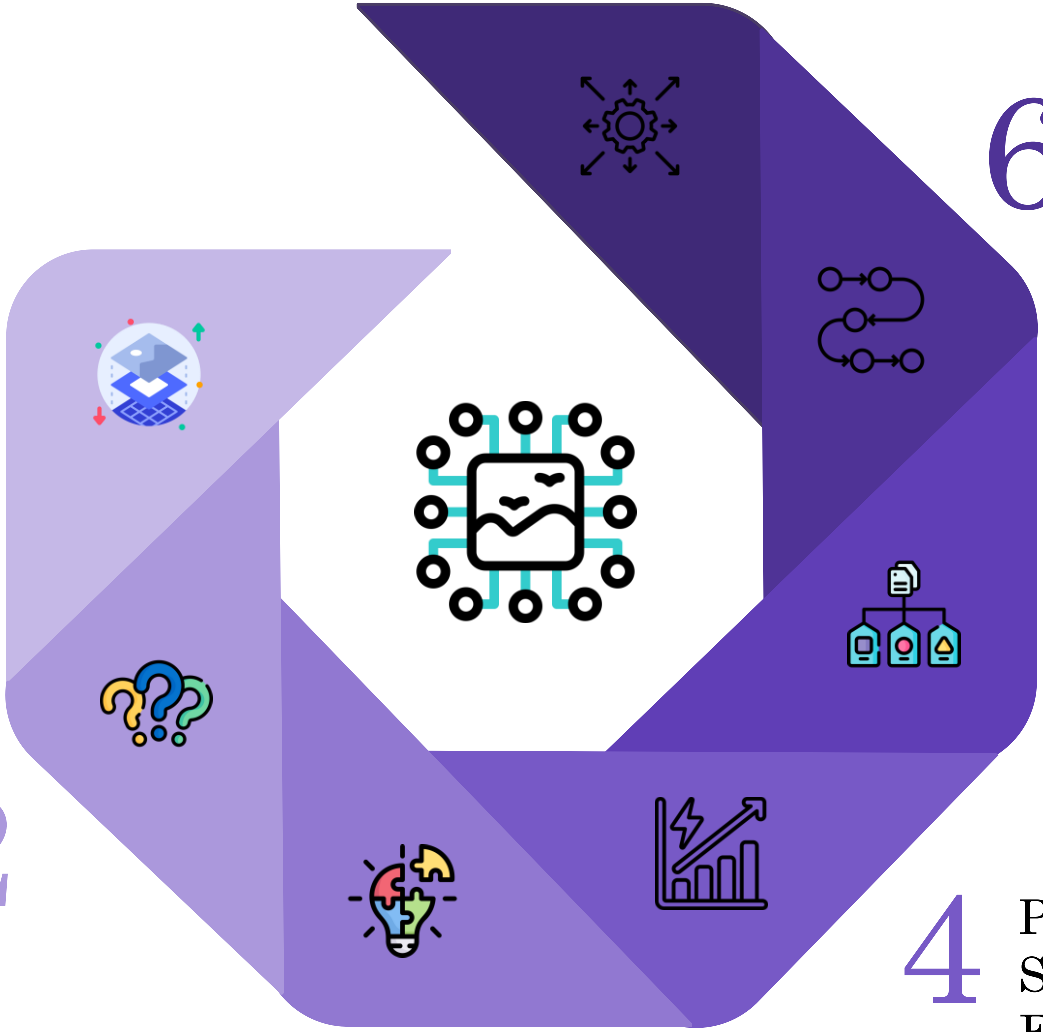
Digital Image Processing

Lecture 04

Local Enhancement via Spatial Filtering

Dr. Muhammad Sajjad
R.A: Asad Ullah

Overview

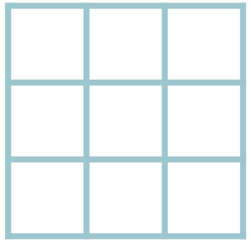


1. Introduction to Spatial Filtering
2. Spatial Filtering (Convolution)
3. Example of Convolution Operation
4. Padding in Spatial Filtering
5. Convolution vs Correlation
6. Smoothing Spatial Filters
7. Sharpening Spatial Filters

Introduction to Spatial Filtering

Spatial Filtering

- A technique used in image processing
- Each pixel's value is modified based on the pixel itself and the values of its neighboring pixels.
- Used for image enhancement, noise reduction, and edge detection.



Median Filter

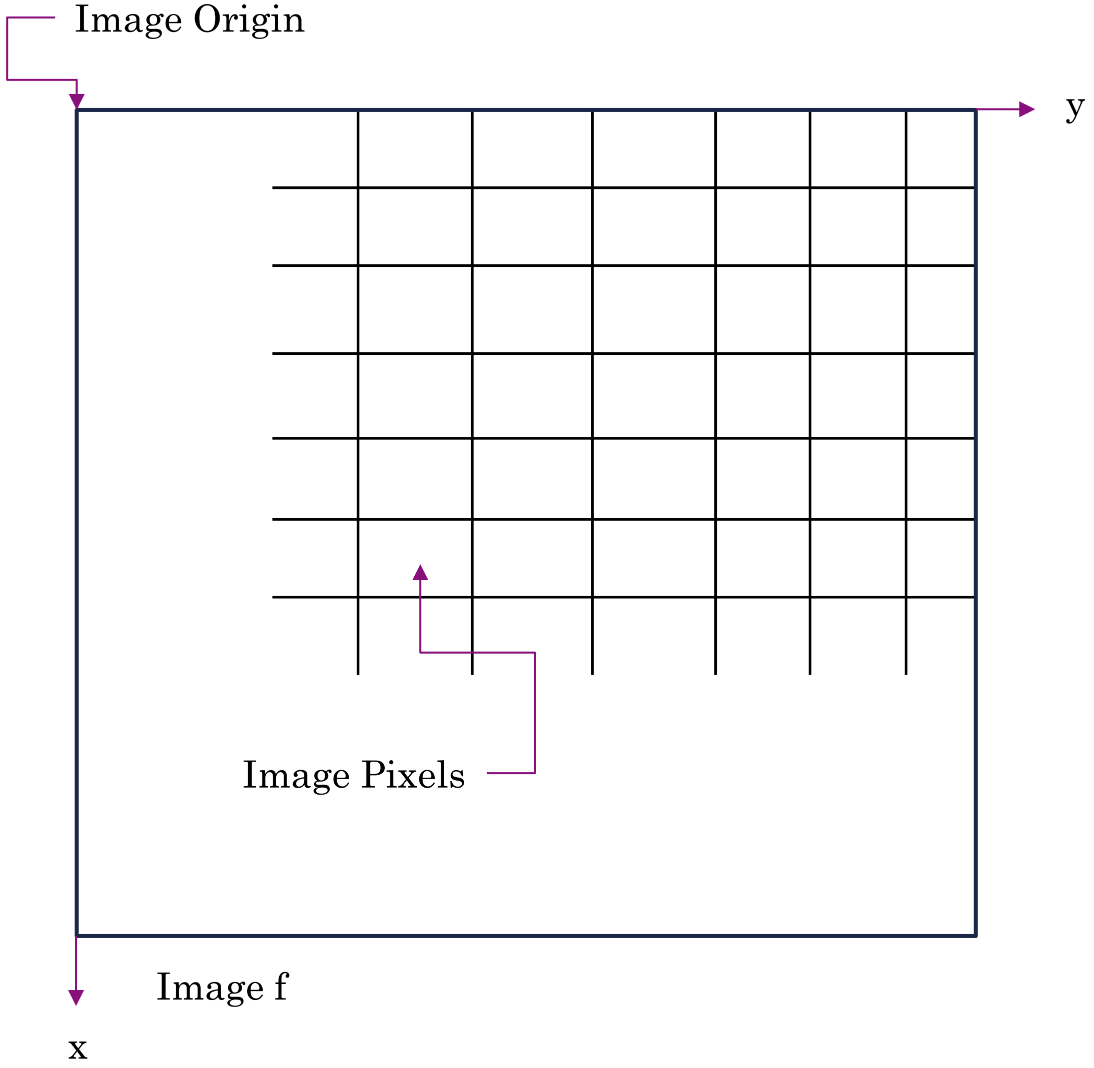
Filter (or Kernel, Mask, Template, Window):

- The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter.
- We prefer odd-sized kernels (e.g., 3x3, 5x5, etc.).
- Kernal size can be represented as (m × n), where m is number of rows of the kernel/filter and n is the number of columns.

Noisy image

Result of Median Filter

Cont.



W (-1,-1)	W (-1,0)	W (-1,1)
W (0,-1)	W (0,0)	W (0,1)
W (1,-1)	W (1,0)	W (1,1)

Filter Kernel, w(s, t)

Kernel coefficients

f (x-1, y-1)	f (x-1, y)	f (x-1, y+1)
f (x, y-1)	f (x, y)	f (x, y+1)
f (x+1, y-1)	f (x+1, y)	f (x+1, y+1)

Pixels values under kernel when it is centered on (x, y)

Spatial Filtering (Convolution)

Linear Spatial Filtering

- Move the kernel across the image, pixel by pixel.
- Perform a sum-of-products operation between the local region of the image f and the filter kernel K.
- Place the computed value into the corresponding pixel in the output image.
- Simple and fast.
- Can blur the image and lose details
- Examples: Smoothing(Box filter, Gaussian filter), Edge Detection (Sobel Filter)

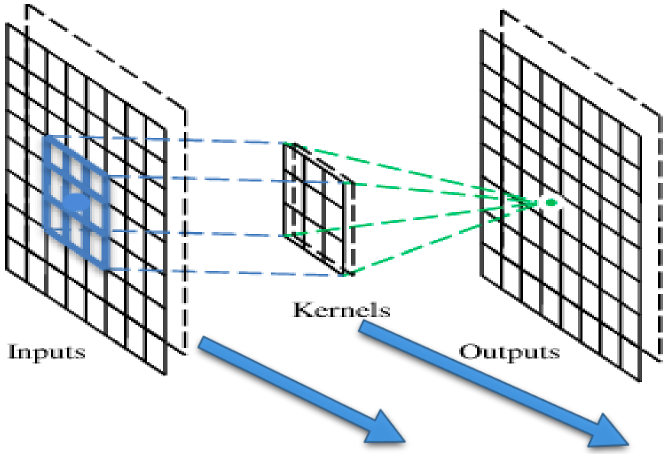
Element-wise Product

Note: In convolution we rotate the kernel by 180 degree before product operation

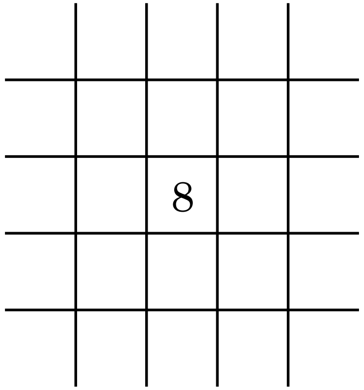
$$\begin{bmatrix} 4 & 0 & 0 \\ 4 & 1 & 2 \\ 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Local region

Sum after element wise product



f is input Image, **w** is kernel, **s, t** are kernel indices



Output Image (g)

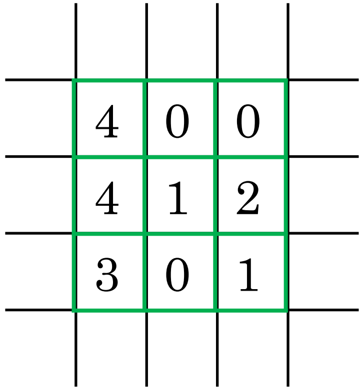
$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) * f(x-s, y-t)$$

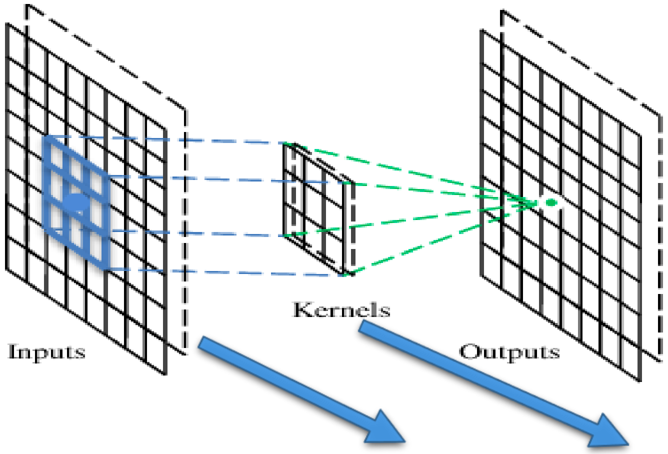
Where a = (m - 1) / 2 and b = (n - 1) / 2

Cont.

Non-Linear Filtering:

- Use operations other than the sum-of-products
- The output is based on operations that don't simply add up the pixel values.
- Example: selecting the median value in a neighborhood (e.g., Median filter) or selecting Min or Max value i.e. Max/Min filter.
- Also called Order-Statistic Filters
- Better at preserving details and removing noise such as salt-and-pepper noise.
- Slower and more complex.





f is input Image, **w** is kernel, **s, t** are kernel indices

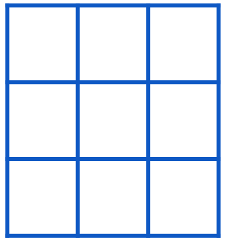
$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \xi\,(f(x-s, y-t))$$

Where a = (m - 1) / 2, b = (n - 1) / 2 and ξ is the median operator.

Note: This filter uses only the neighboring values and updates as the kernel moves through the pixels.

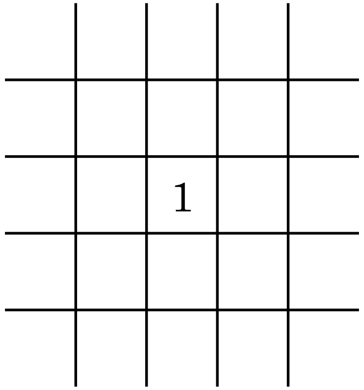
This symbol represent median operation

ξ



Local region

Selecting the median value after sorting the neighborhood

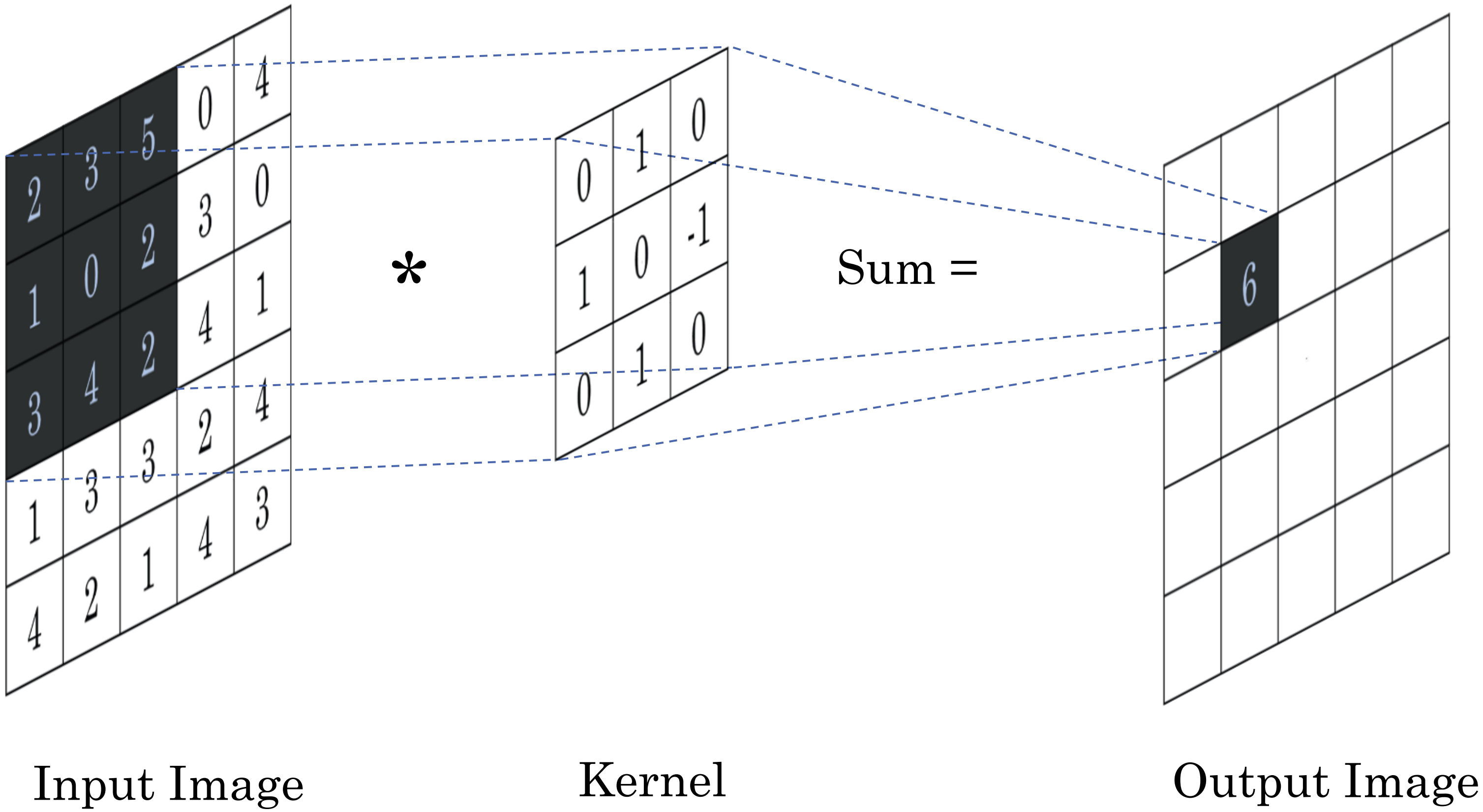


Output Image (g)

Example of Convolution Operation

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1},\mathbf{1}) = (2*0) + (3*1) + (5*0) + (1*1) + (0*0) + (2*-1) + (3*0) + (4*1) + (2*0)$$

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1},\mathbf{1}) = 6$$

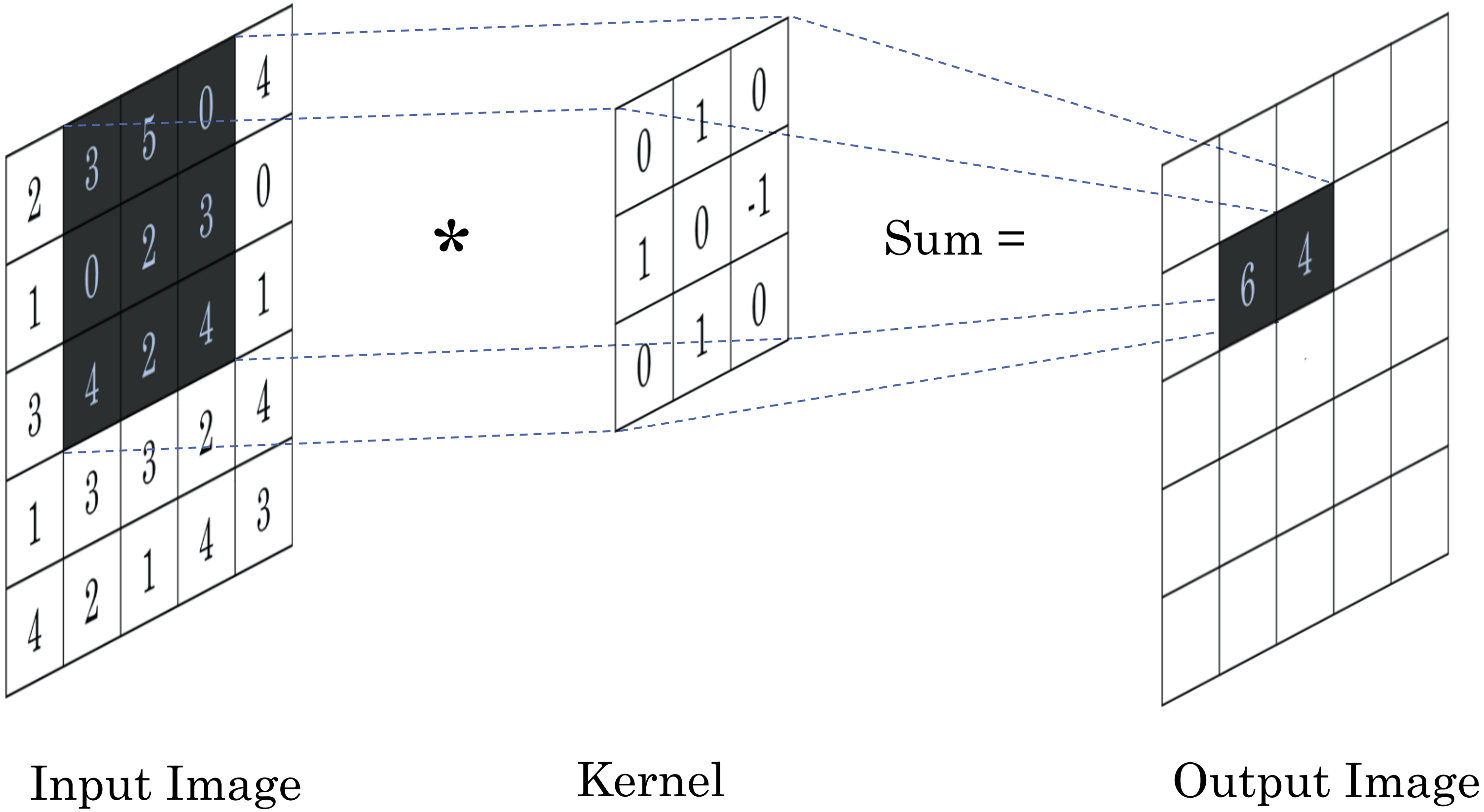


Note: Here the kernel is pre-rotated by 180 degree

Example of Convolution Operation

$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1}, \mathbf{2}) = (3 * 0) + (5 * 1) + (0 * 0) + (0 * 1) + (2 * 0) + (3 * -1) + (4 * 0) + (2 * 1) + (4 * 0)$

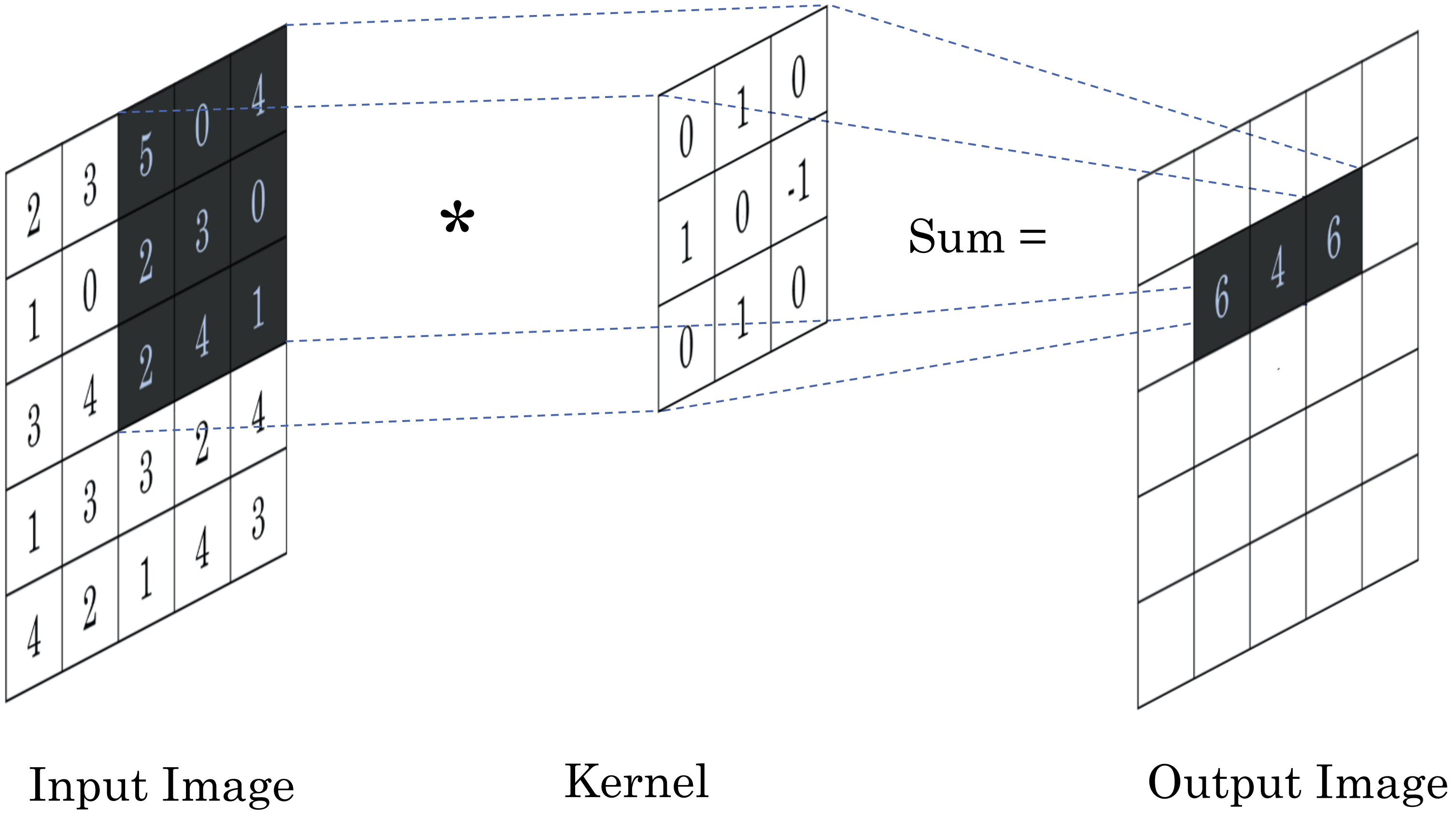
$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1}, \mathbf{2}) = 4$



Example of Convolution Operation

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1},\mathbf{3}) = (5 * 0) + (0 * 1) + (4 * 0) + (2 * 1) + (3 * 0) + (0 * -1) + (2 * 0) + (4 * 1) + (1 * 0)$$

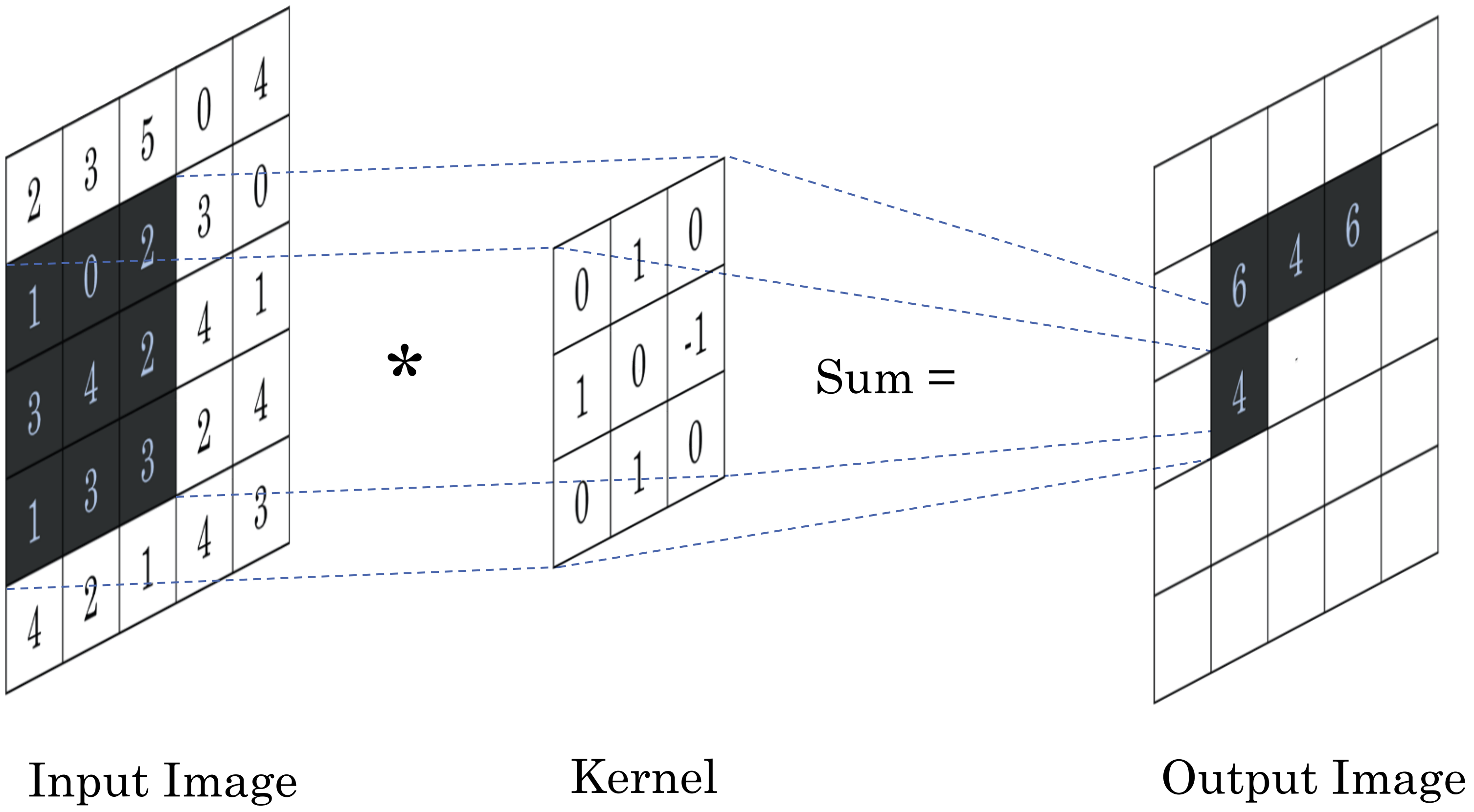
$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{1},\mathbf{3}) = 6$$



Example of Convolution Operation

$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2}, \mathbf{1}) = (1 * 0) + (0 * 1) + (2 * 0) + (3 * 1) + (4 * 0) + (2 * -1) + (1 * 0) + (3 * 1) + (3 * 0)$

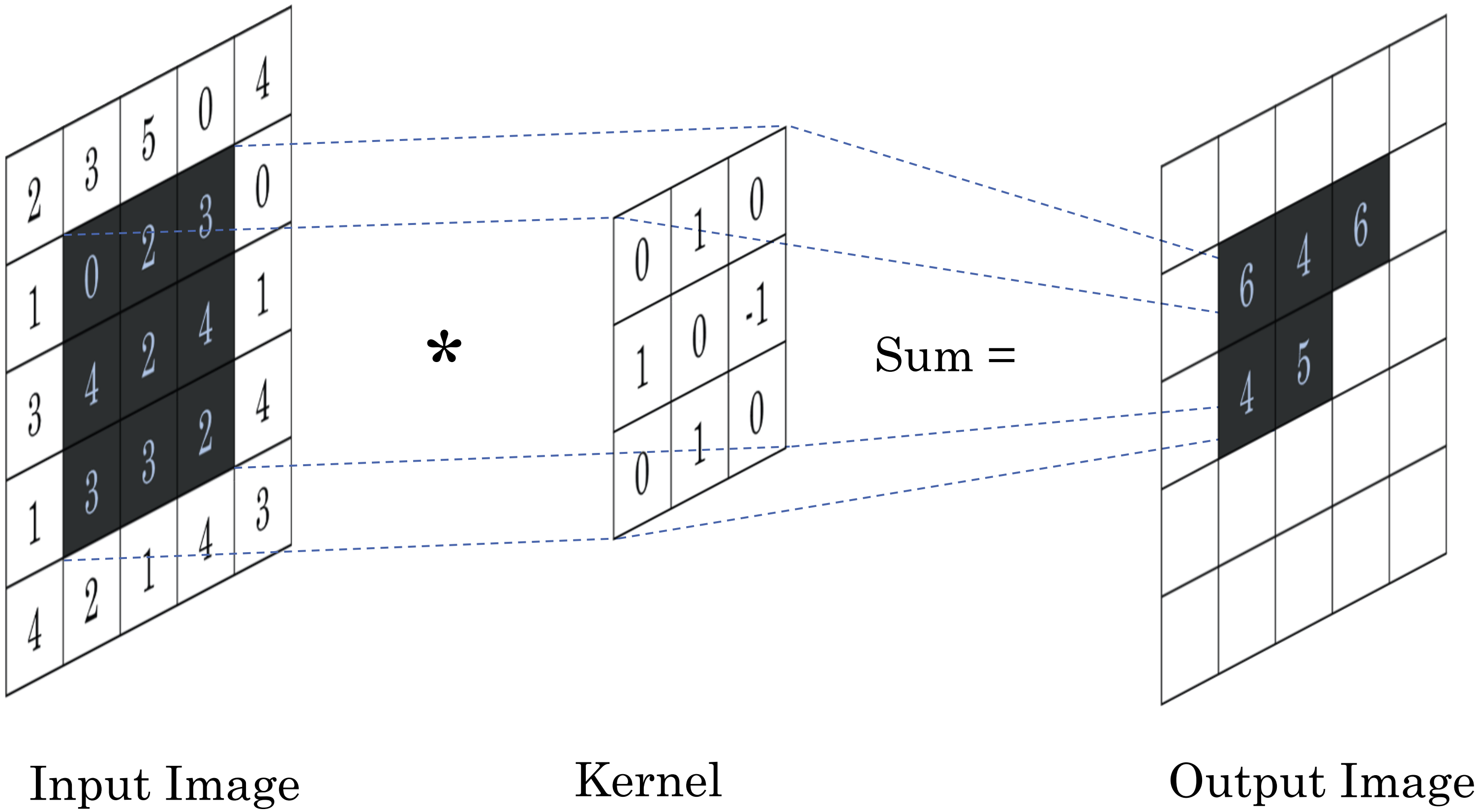
$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2}, \mathbf{1}) = 4$



Example of Convolution Operation

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2},\mathbf{2}) = (0*0) + (2*1) + (3*0) + (4*1) + (2*0) + (4*-1) + (3*0) + (3*1) + (2*0)$$

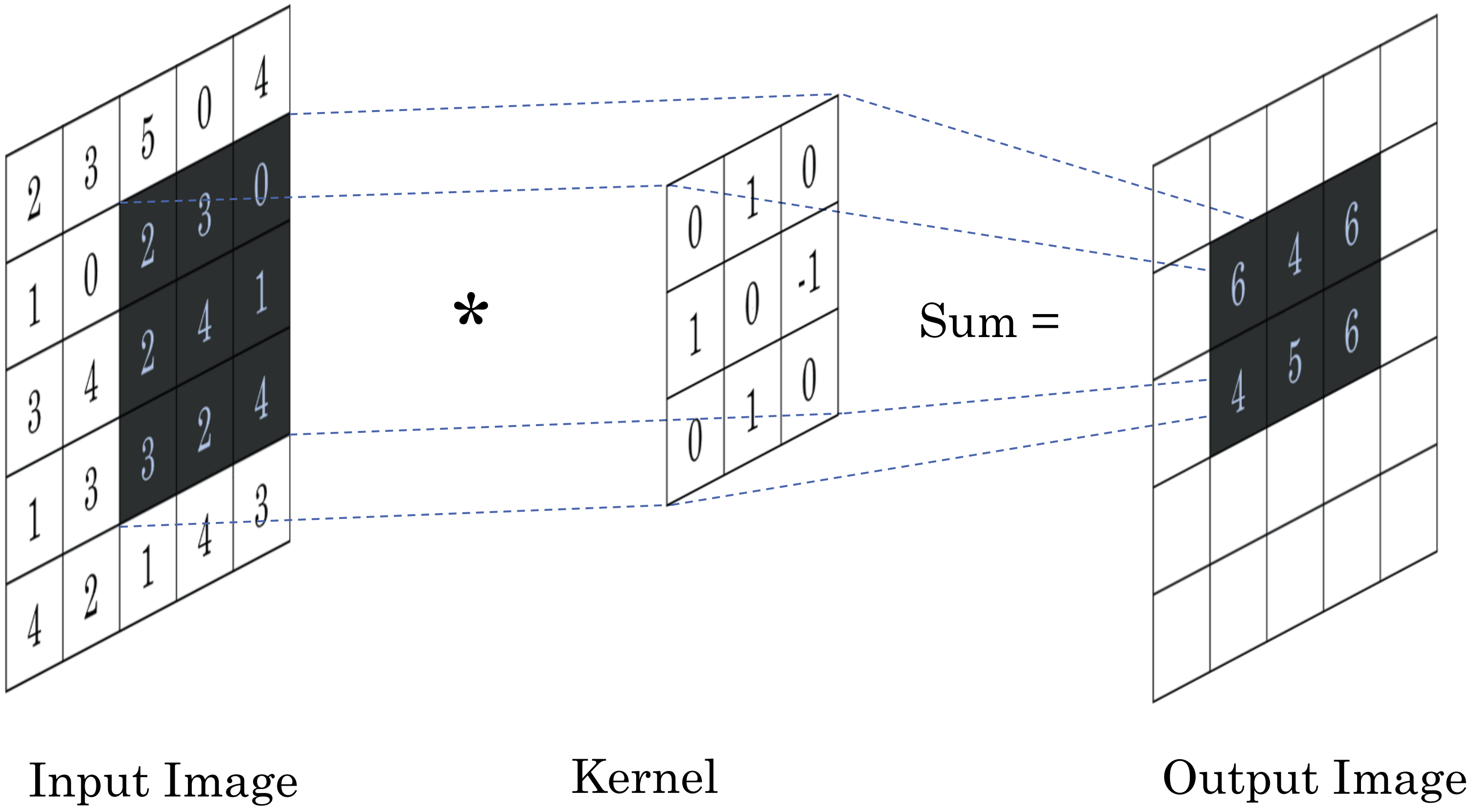
$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2},\mathbf{2}) = 5$$



Example of Convolution Operation

$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2}, \mathbf{3}) = (2 * 0) + (3 * 1) + (0 * 0) + (2 * 1) + (4 * 0) + (1 * -1) + (3 * 0) + (2 * 1) + (4 * 0)$

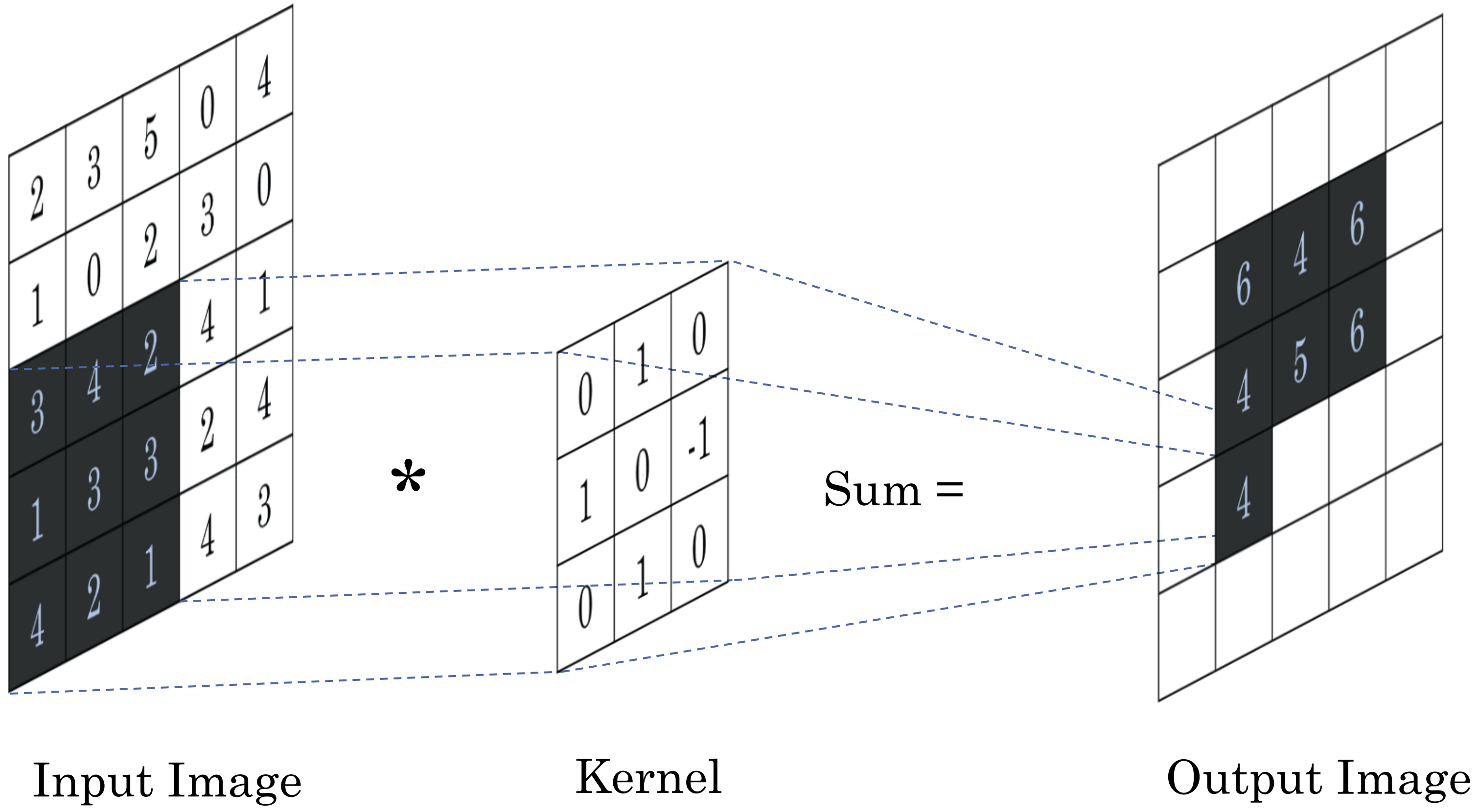
$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{2}, \mathbf{3}) = 6$



Example of Convolution Operation

$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3}, \mathbf{1}) = (3 * 0) + (4 * 1) + (2 * 0) + (1 * 1) + (3 * 0) + (3 * -1) + (4 * 0) + (2 * 1) + (1 * 0)$

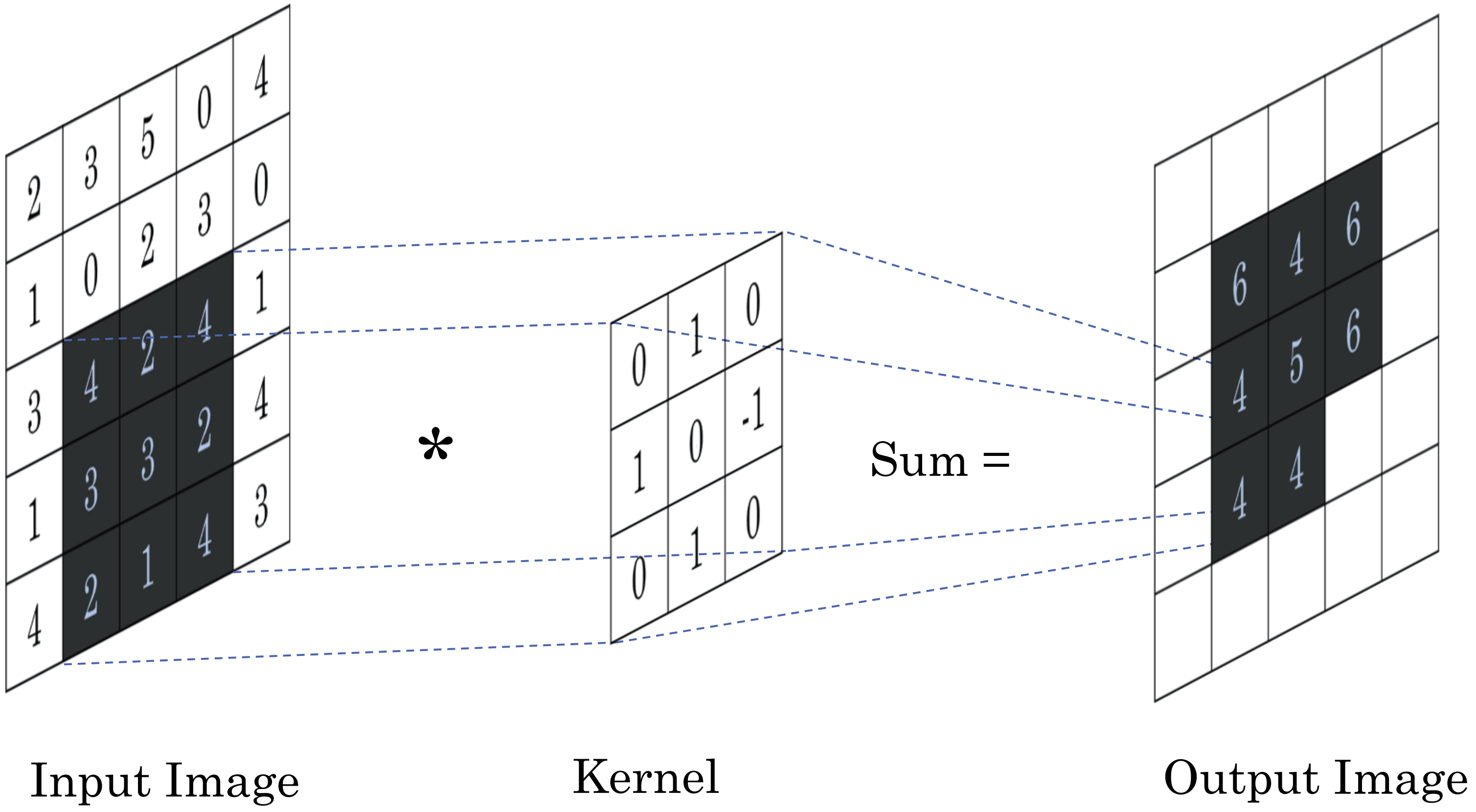
$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3}, \mathbf{1}) = 4$



Example of Convolution Operation

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3},\mathbf{2}) = (4 * 0) + (2 * 1) + (4 * 0) + (3 * 1) + (3 * 0) + (2 * -1) + (2 * 0) + (1 * 1) + (4 * 0)$$

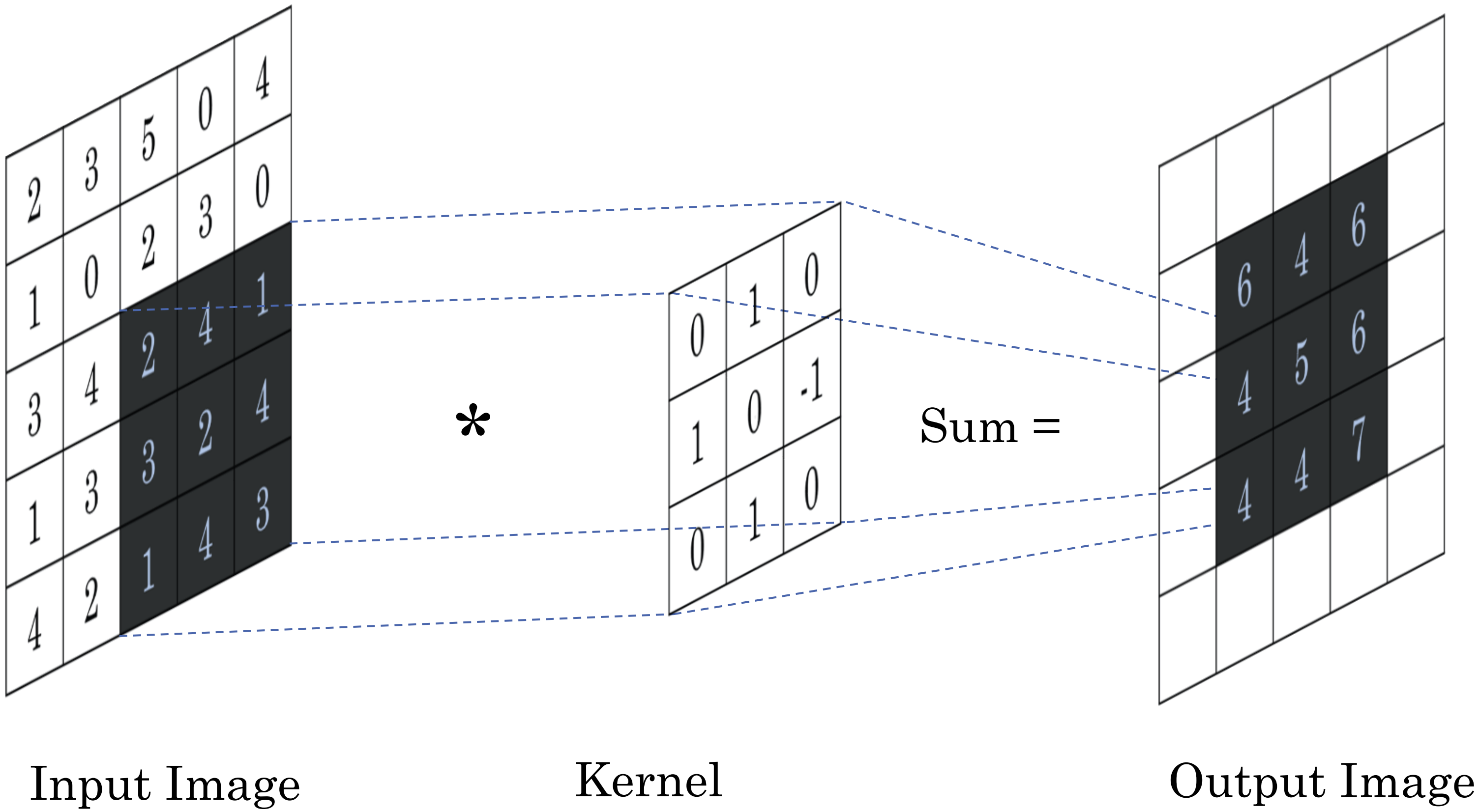
$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3},\mathbf{2}) = 4$$



Example of Convolution Operation

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3},\mathbf{3}) = (2*0) + (4*1) + (1*0) + (3*1) + (2*0) + (4*-1) + (1*0) + (4*1) + (3*0)$$

$$(\boldsymbol{f} * \boldsymbol{w})(\mathbf{3},\mathbf{3}) = 7$$



Padding in Spatial Filtering

- What is padding?
- Why we use padding in Spatial Filtering?
- In the figure shown, part of the kernel 'w' lies outside the image 'f' when processing the edge pixels. As a result, the sum of the products is undefined in that area.
- In the figure shown, the size of the output decreased when we didn’t process the edge pixels.
- Padding ensures that the filter can be applied evenly across all pixels.

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

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

w

0	4	1	3	0
1	3	1	0	2
4	0	2	4	3
2	0	0	1	4
0	3	1	2	0

f (no padding)

0	4	1	3	0
1	3	1	0	2
4	0	2	4	3
2	0	0	1	4
0	3	1	2	0

⇨

1	0	-1
4	-2	-6
3	-4	-4

Padding in Spatial Filtering

Zero Padding

- Zero padding adds zeros around the image border to maintain the same size after filtering.
- **Use cases**: Commonly used in deep learning to ensure output feature maps match input dimensions, especially in multi-layer networks.

Replicate Padding

- Replicate padding copies border pixel values as padding to preserve edge information.
- **Use cases**: Useful in smoothing filters (e.g., Gaussian blur) to avoid artificial edges from zero padding.

For a kernel of size m × n:

- Pad with **(m−1)/2,** rows of zeros (top and bottom).
- Pad with **(n−1)/2**, columns of zeros (left and right).

1	2	3
4	5	6
7	8	9

f (no padding)

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

w

Here kernel (w) size is 3 x 3:
3-1/2 = 1 row above and below
3-1/2 = 1 col left and right

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

Zero Padding

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

Replicate Padding

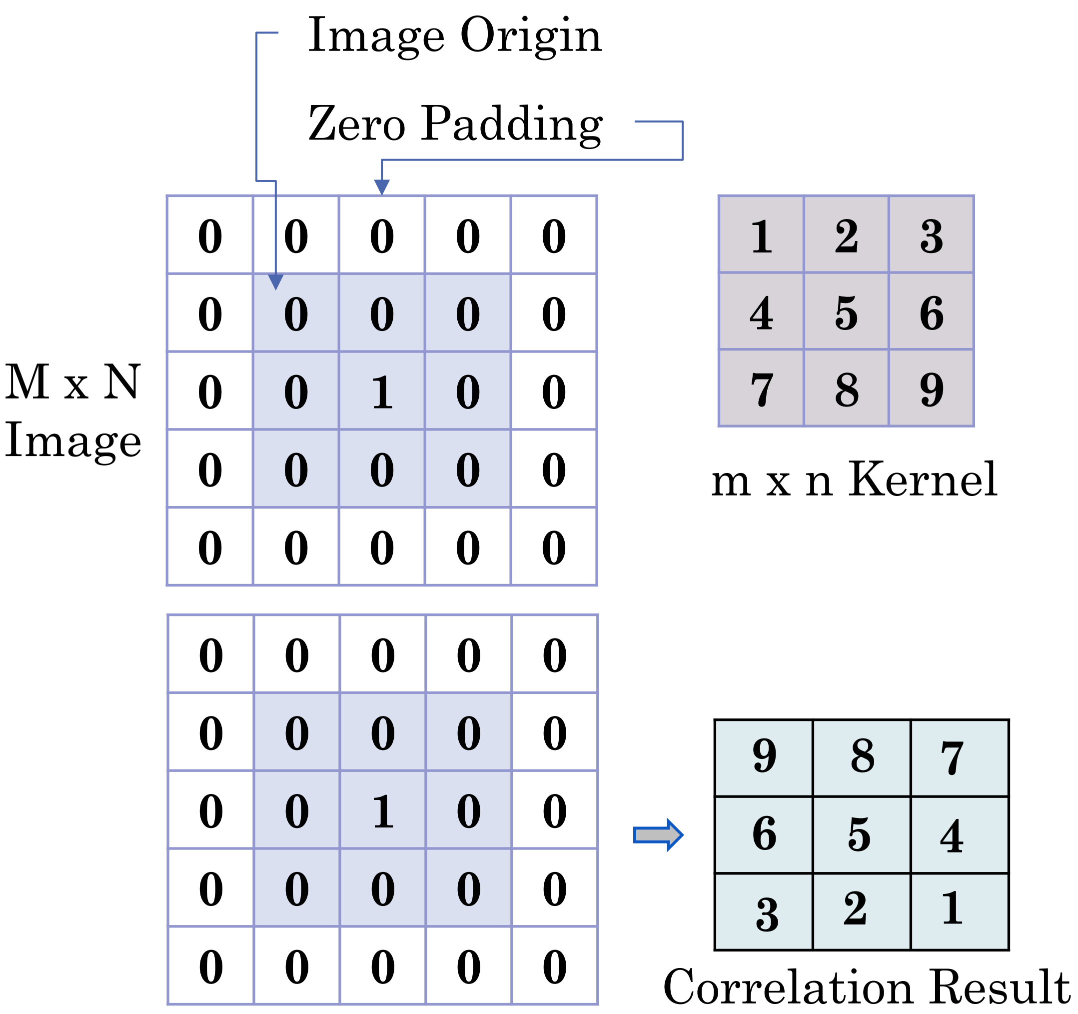
Convolution vs Correlation

Correlation

- Moving the center of a kernel over an image, and computing the sum of products at each location

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

$$a = (m-1)/2 \qquad b = (n-1)/2$$



Cont.

Convolution

- Moving 180" rotated kernel over an image, and computing the sum of products at each location.

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

$$a = (m - 1)/2 \qquad b = (n - 1)/2$$

Here the formula is slightly different. The minus “ - ” sign rotates the kernel by 180 degree.

Image Origin

Zero Padding

M x N Image

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

9	8	7
6	5	4
3	2	1

m x n Kernel Pre-rotated

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

⇨

1	2	3
4	5	6
7	8	9

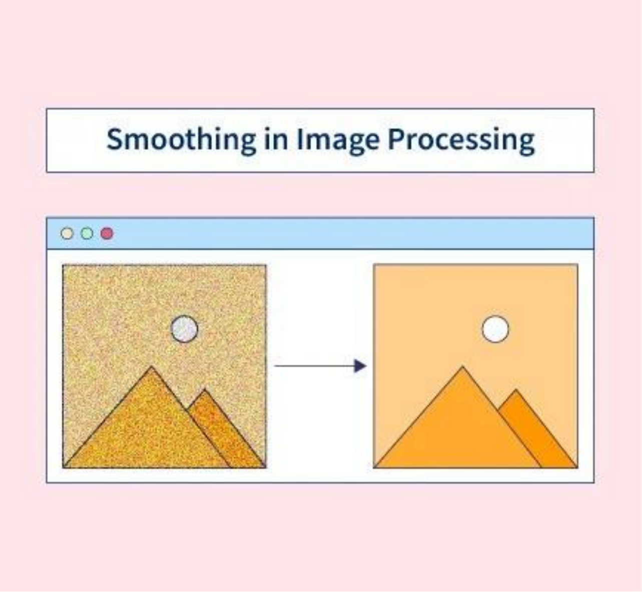
Convolution Result

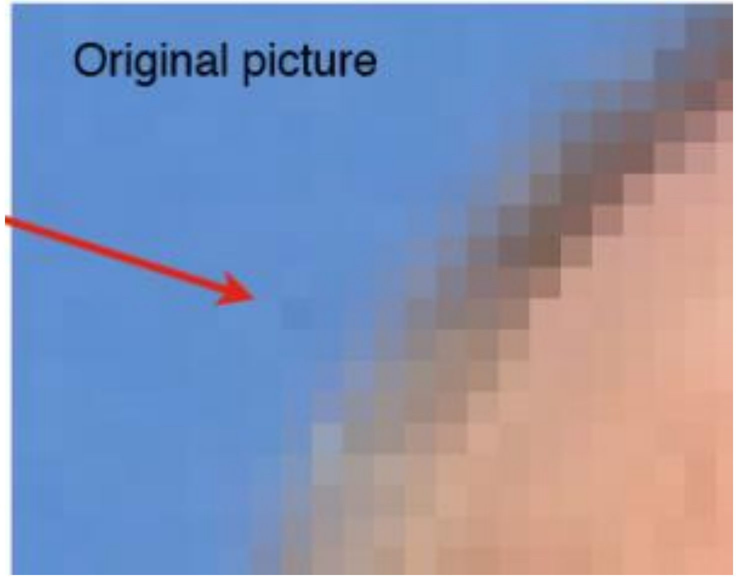
Smoothing Spatial Filters

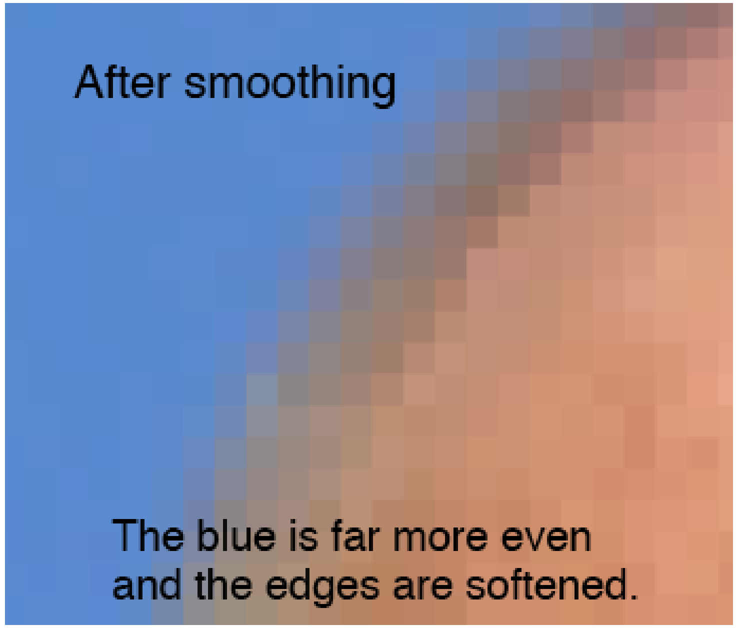
Used for: Blurring and noise reduction.

Blurring helps to:

- Blurring is usually used in preprocessing steps.
- Remove small details and noise before object extraction.
- Bridge small gaps in lines or curves.







Cont.

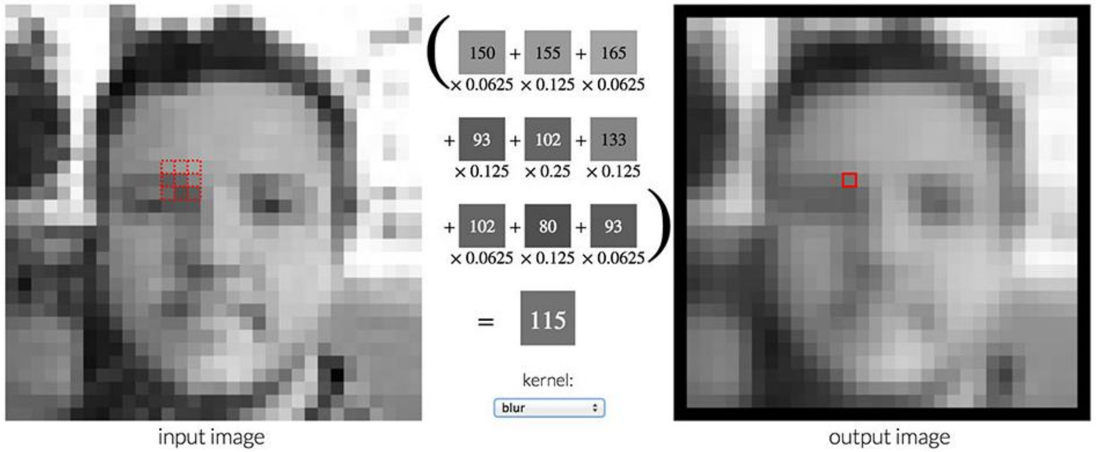
Averaging Linear Filters

- Replaces each pixel with the average of its neighboring pixels.
- Image M x N,   Filter m x n

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

$$\frac{1}{9} * \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Box Filter
All coefficients are equal



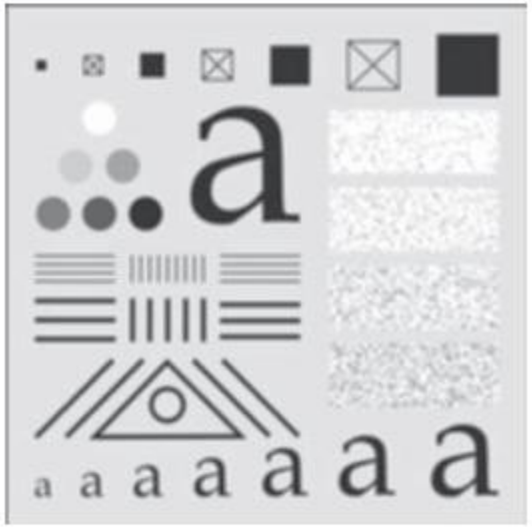
Cont.

Example of box kernel

Here the dark edges show zero padding

Original Image

3 x 3 box
kernel result



11 x 11 box
kernel result

21 x 21 box
kernel result

Cont.

Gaussian Filter (Weighted average linear filter)

- The filter where nearby pixels have higher weight, and distant pixels have lower weight.
- Image M × N,   Filter m × n

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

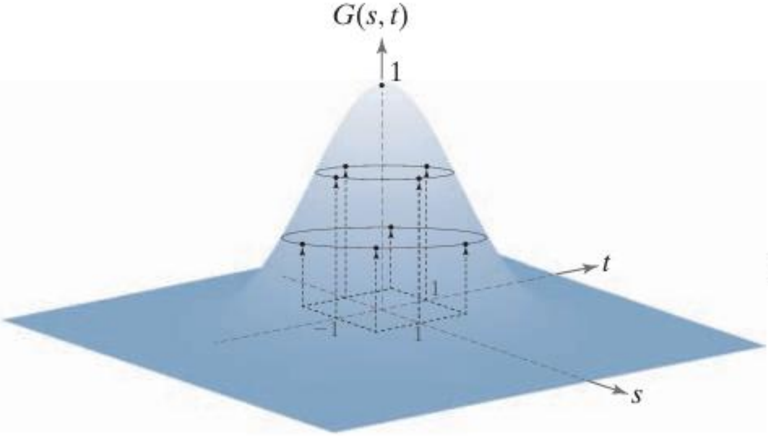
- Mathematically, Gaussian filter is;

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

Here, K is the scale (i.e., 1 in the Weighted Average, as shown in the upper right figure), and σ is the standard deviation.

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

Weighted Average
Nearby pixels have higher weight, and distant pixels have lower weight



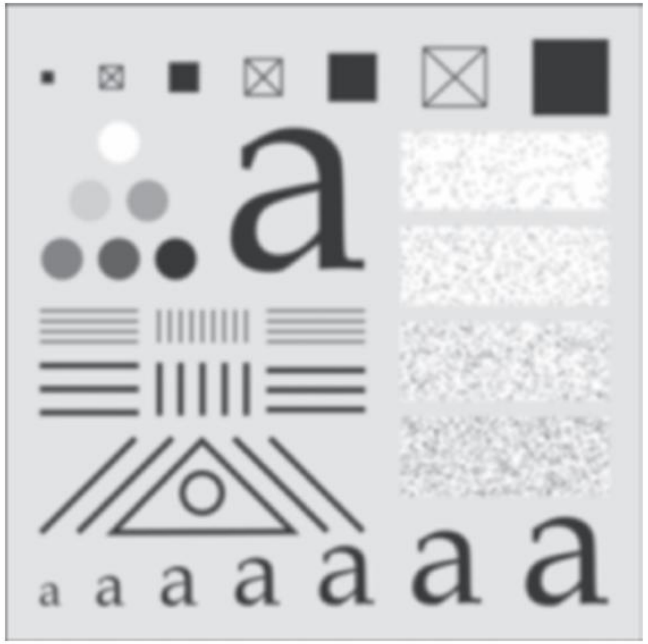
$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

$\lceil 6\sigma \rceil$ x $\lceil 6\sigma \rceil$ can be used for the size of Gaussian kernel

Cont.

Example of Weighted Average Filters

Original Image



Result of 21 x 21 Gaussian kernel with σ = 3.5

Result of 43 x 43 Gaussian kernel with σ = 7

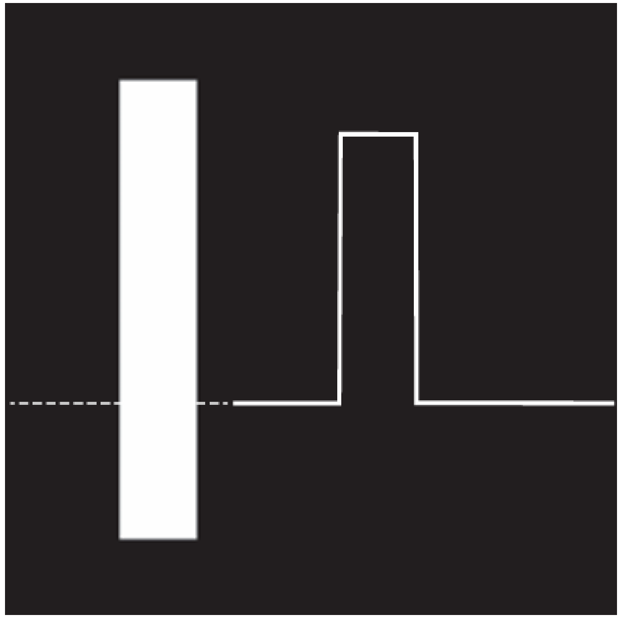
Add 1 t make it odd

$\lceil 6 \times 3.5 \rceil$ x $\lceil 6 \times 3.5 \rceil$ = 21 x 21

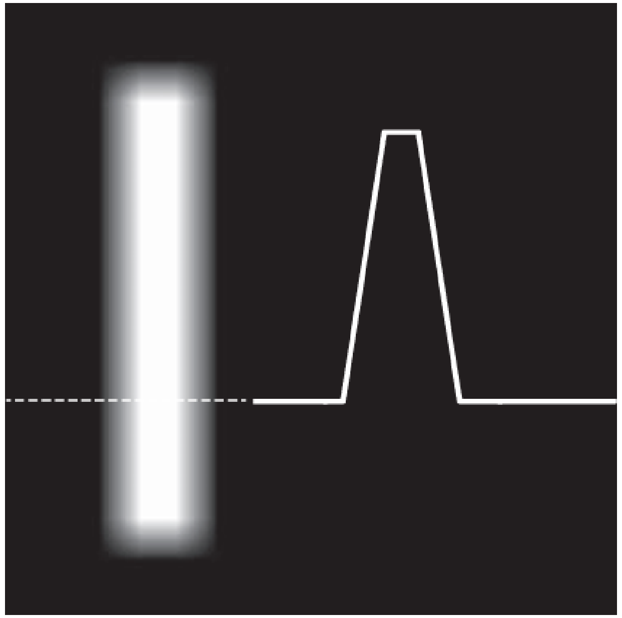
$\lceil 6 \times 7 \rceil$ x $\lceil 6 \times 7 \rceil$ = 42+1 x 42+1

Cont.

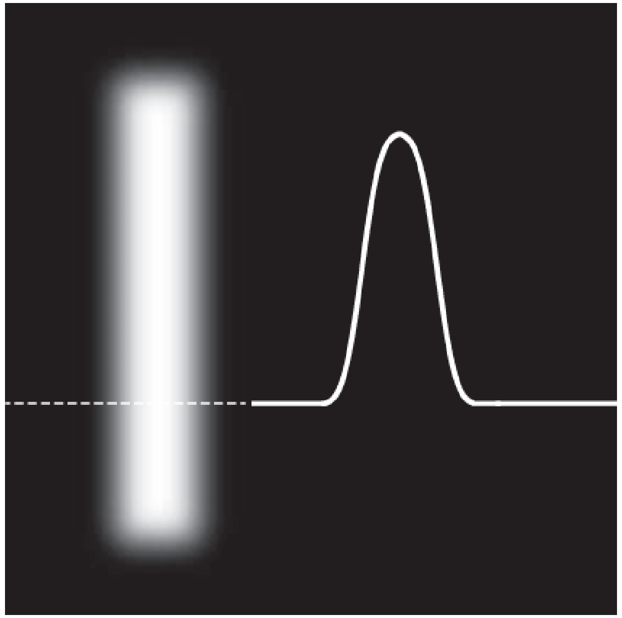
Example Comparison of Box and Gaussian Filter



Original Image



Result of 71x 71 box kernel



Result of 151 x 151 Gaussian kernel with $\sigma = 25$

Cont.

Order-statistic (Nonlinear) Filters

- Nonlinear
- Based on ordering (ranking) the pixels contained in the filter mask
- Replacing the value of the center pixel with the value determined by the ranking result
- E.g., median filter, max filter, min filter

Median Filtering

- Assigns the mid value of all the gray levels in the mask to the center of mask
- Useful in removing impulse noise (also known as salt-and-pepper-noise).

Median Filter 3x3

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

1
2
3
3
5
6
7
8
8

Sorting Values in ascending order

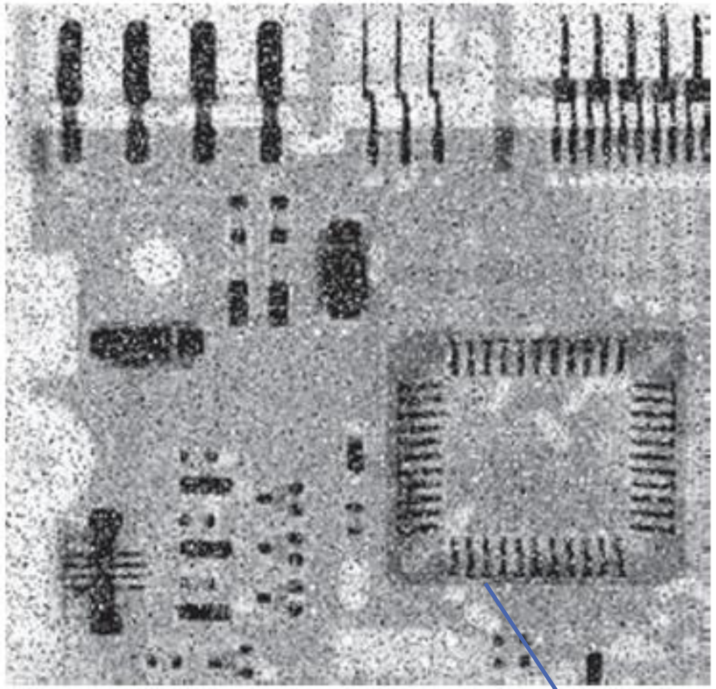
Median Value

5	6	5
3	6	5
4	6	5

Convolution Result

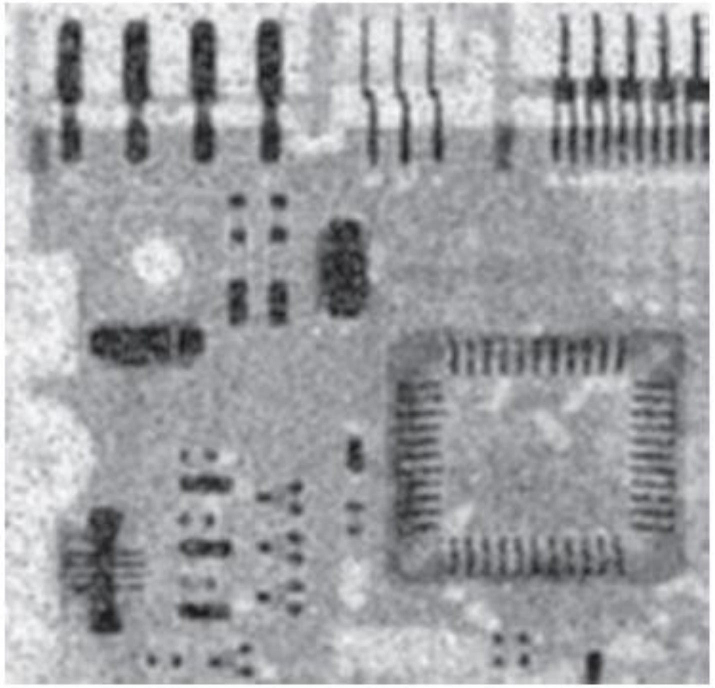
Cont.

Median Filtering

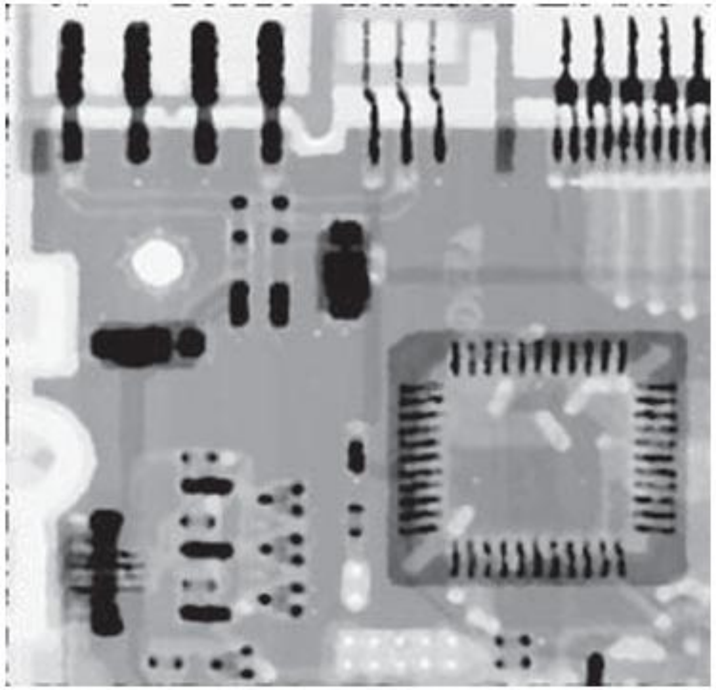


X-ray image of a circuit board

Corrupted by salt-and-pepper noise



Result 3x3 averaging filer



Using 3x3 median filter

Sharpening Spatial Filters

Definition:

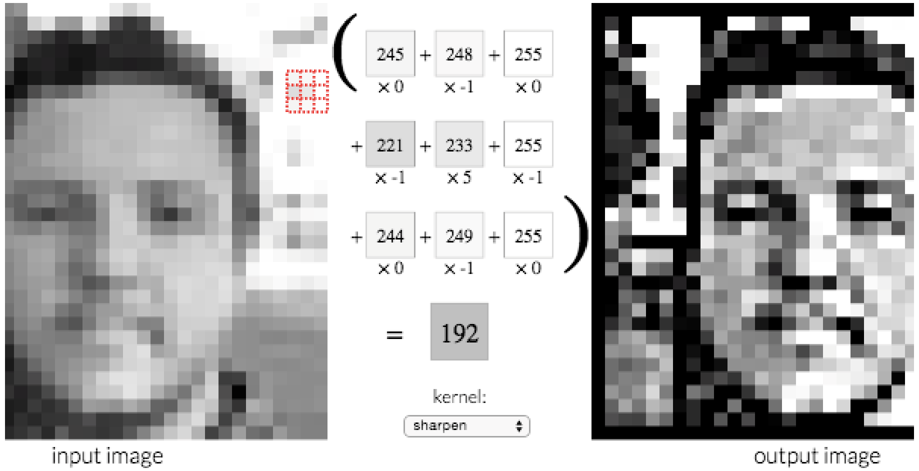
- Sharpening filters enhance the edges and fine details of an image, making objects and boundaries more distinct.
- They work by emphasizing the high-frequency components of the image, such as edges.

Foundation:

- Blurring/smoothing is performed by spatial averaging (equivalent to integration).
- Sharpening is performed by noting only the gray level changes in the image that is the **differentiation.**

Applications:

- Edge Detection: Used in image segmentation, object recognition, and feature extraction.
- Image Enhancement: Improves clarity and sharpness, making details more visible.
- Computer Vision: Sharpening helps in recognizing finer details in objects, such as texture and structure.

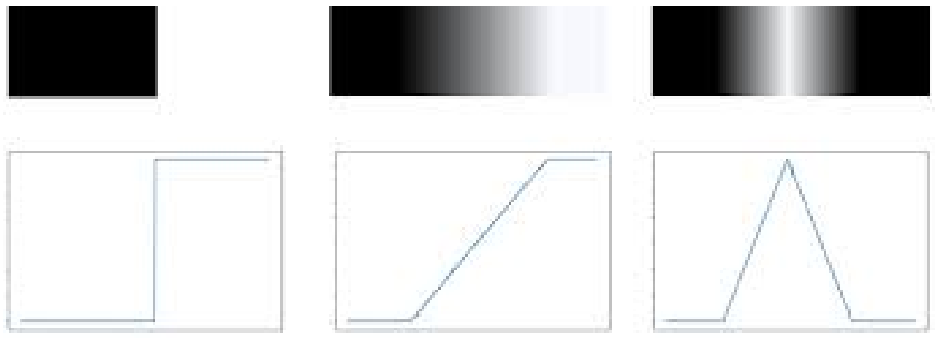


Sharpening via Spatial Differentiation

First-Order Derivatives (Gradient-based sharpening):

- Measures the rate of change of pixel intensity.
- A basic definition of the first-order derivative of a one-dimensional function f (x) is the difference,

$$\frac{\partial f}{\partial x} = f(x+1,) - f(x)$$



Step Edges: Abrupt change

Ramp Edges: Gradual change

Roof Edges: Peak change.

The first derivative must be:

1. Zero in areas of constant intensity, (along flat segments)
2. Nonzero at the onset of an intensity step or ramp.
3. Nonzero along intensity ramps.

1	4	0	5	*	-1	1	⇨	3	-4	5

First-order derivative kernel used for edge detection

Cont.

Second order derivatives of digital functions

We define the second-order derivative of f (x) as the difference,

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

f(x-1)	f(x)	f(x+1)

➡

1	-2	1

The second derivative must be:

- Zero in areas of constant intensity.
- Nonzero at the onset and end of an intensity step or ramp.
- Zero along intensity ramps.

1	4	0	5	3	2

*

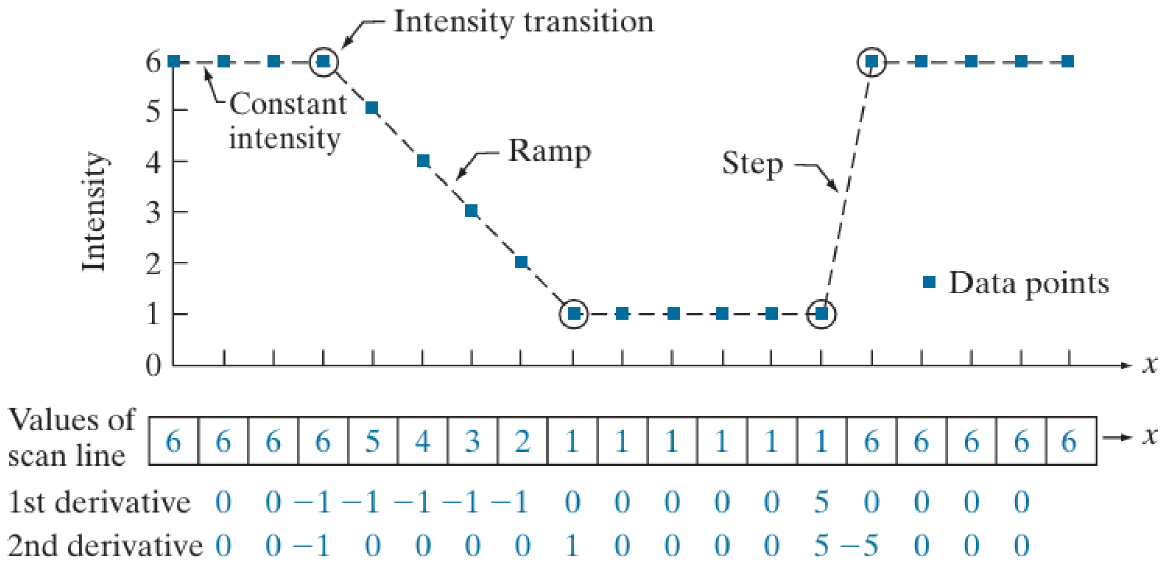
1	-2	1

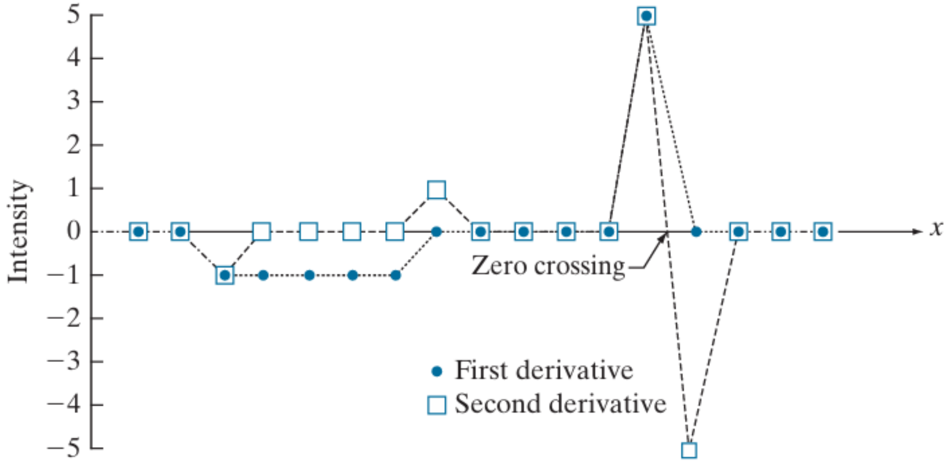
➡

-7	9	-7	1

Example of derivatives

- 1st derivative detect thick edges while 2nd derivative detect thin edges.
- 2nd derivative has mush stronger response at gray-level step than 1st derivative.





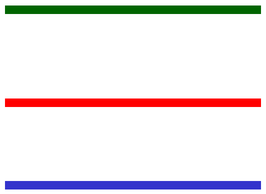
Various situations encountered for derivatives

$$f' = \frac{\partial f}{\partial x} \qquad f'' = \frac{\partial^2 f}{\partial x^2}$$

- Ramps or steps in the 1D profile normally characterize the edges in an image
- f″ is nonzero at the onset and end of the ramp: produce thin (double) edges
- f ′ is nonzero along the entire ramp produce thick edges

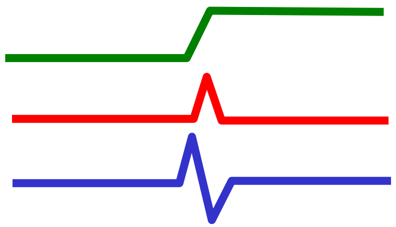
•Flat segment→ (f')=0; (f'')=0

f	0	0	0	0	0	
f'		0	0	0	0	
f''		0	0	0		



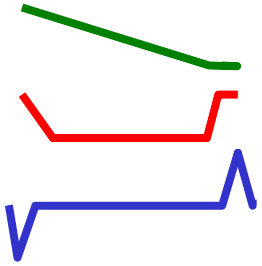
•Step→ (f'):{0,+,0}; (f''):{0,+,–,0}

f	0	0	0	7	7	7	7	
f'		0	0	7	0	0	0	0
f''		0	7	–7	0	0	0	



•Ramp→ (f')≈constant; (f'')=0

f	5	4	3	2	1	0	0	
f'	0	–1	–1	–1	–1	–1	0	0
f''	–1	0	0	0	0	1	0	



The Laplacian Filter

- 2D second-order derivative operator used for image sharpening.
- Highlights sharp intensity transitions and de-emphasizes regions with slow intensity changes.
- Produces grayish edge lines and discontinuities on a dark background.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\therefore \frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\therefore \frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

- We can apply these kernels using convolution operations.

	f(x-1, y)	
f(x, y-1)	f(x, y)	f(x,y+1)
	f(x+1,y)	

Rotation invariant at 90° ++

0	1	0
1	-4	1
0	1	0

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

Laplacian kernel, Equation

Rotation invariant at 45° ++

Includes the diagonal terms

1	1	1
1	-8	1
1	1	1

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian for Image Enhancement

To obtain the enhance image

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y), & w_5 < 0 \\ f(x,y) + \nabla^2 f(x,y), & w_5 > 0 \end{cases}$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

- f(x, y) → Original Image
- $\nabla^2 f(x,y)$ → Laplacian of original image
- In this way, background tonality can be perfectly preserved while details are enhanced.

Laplacian Kernel, Highlight areas of sharp intensity

-1	-1	-1
-1	8	-1
-1	-1	-1

\+

0	0	0
0	1	0
0	0	0

Identity kernel, doesn't modify the pixel values

=

Resultant Kernel, produce a sharper image by preserving the original intensity values

-1	-1	-1
-1	9	-1
-1	-1	-1

Laplacian for Image Enhancement (Example)

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y), & w_5 < 0 \\ f(x,y) + \nabla^2 f(x,y), & w_5 > 0 \end{cases}$$

Blurred image of the North Pole of the moon.

Laplacian image obtained using the 90° isotropic kernel

Image sharpened using equation above

Image sharpened using the same procedure, but with 45° isotropic kernel.

The Gradient Filter

- 2D first derivative (∇f) in image processing are implemented using the magnitude of the gradient
- The gradient is generally given by

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- The magnitude is given by

$$M(x,y) = mag(\nabla f) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

Magnitude is approximated as

$$\nabla f \approx |G_x| + |G_y|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_6	z_7	z_8

For a sub-image given in the figure: The gradient is approximated by

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

-1	0
0	1

0	-1
1	0

This equation can be represented in masks (**Robert cross gradient operators**)

Cont.

Prewitt Operator

Normally the smallest mask used is of size 3 x 3
Based on the concept of approximating the gradient several spatial masks have been proposed:

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

Extract horizontal edges

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

Extract vertical edges

z_1	z_2	z_3
z_4	z_5	z_6
z_6	z_7	z_8

Pixels arrangements

Prewitt Operator's Equation

$$\nabla f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)|$$

Cont.

Sobel Operator

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

Extract horizontal edges

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

Extract vertical edges

z_1	z_2	z_3
z_4	z_5	z_6
z_6	z_7	z_8

Pixels arrangements

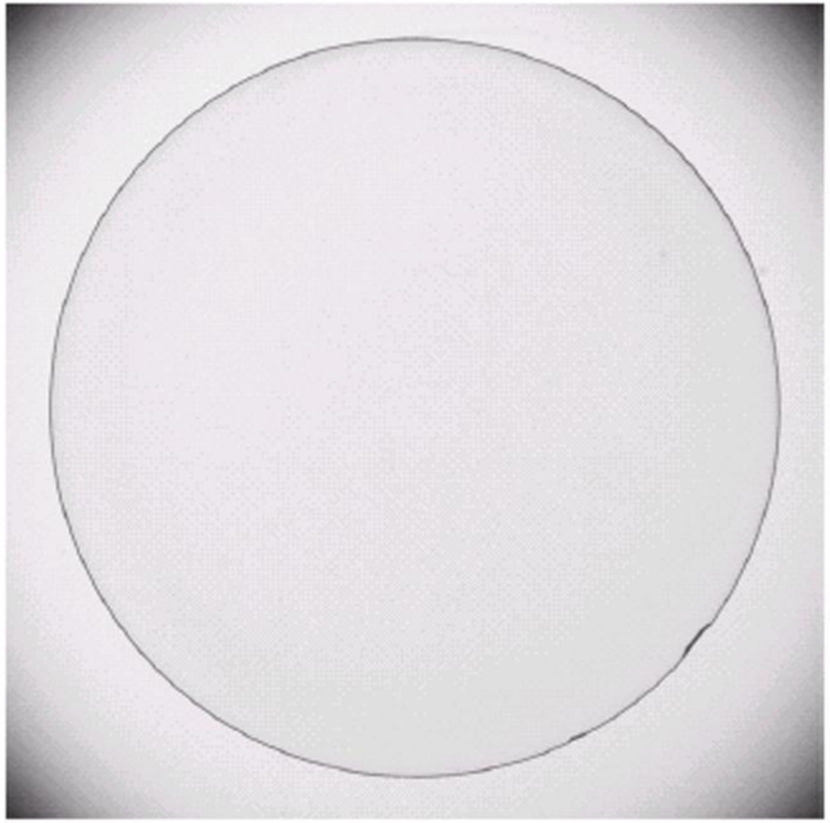
Prewitt Operator's Equation

$$\nabla f \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

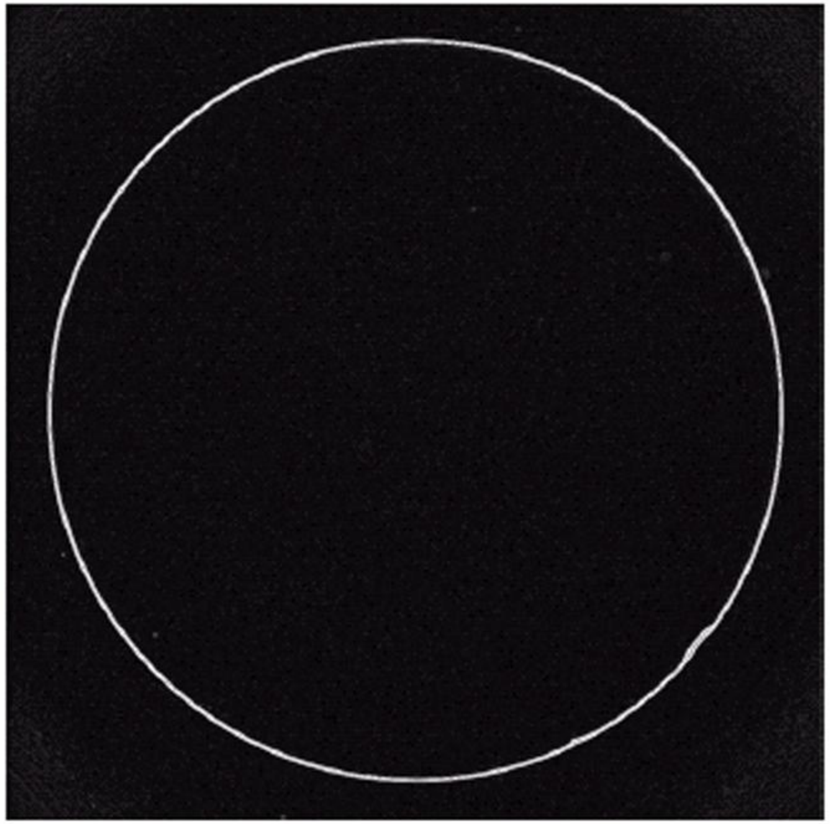
Emphasize more the current position (y)

Emphasize more the current position (x)

Gradient Processing (example)

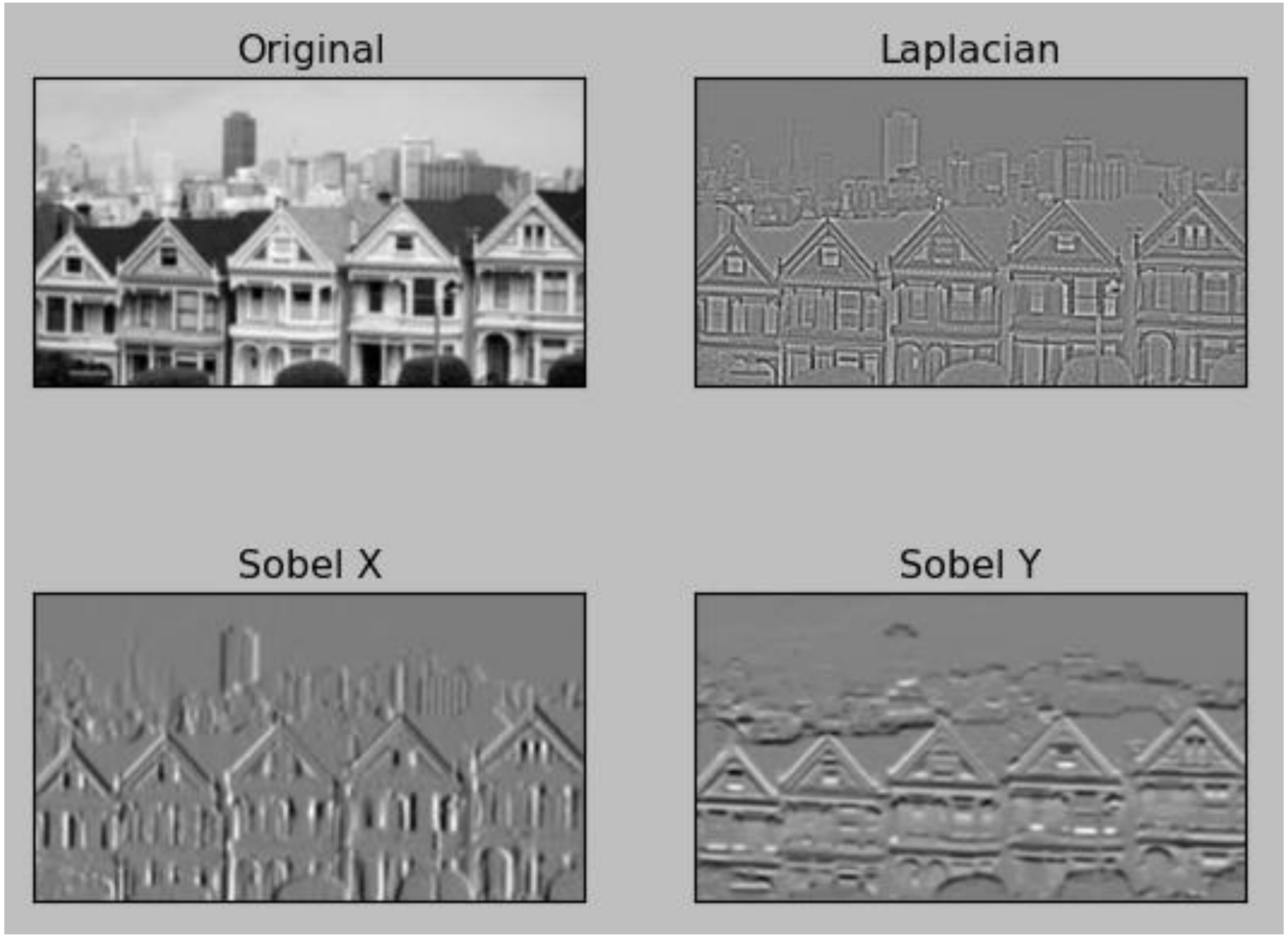


Optical image of contact lens
(note defects at 4 and 5 o'
clock



Sobel Gradient

Sobel vs Laplacian



Thank You

For Your Attention