High Impact Skills Development Program

in Artificial Intelligence, Data Science, and Blockchain

Module: [Computer Vision]

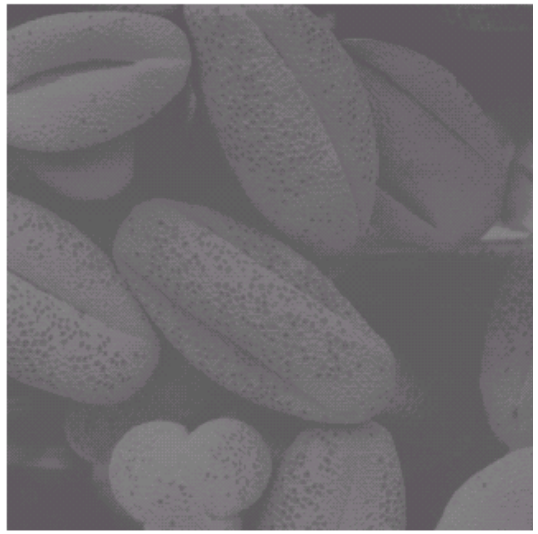
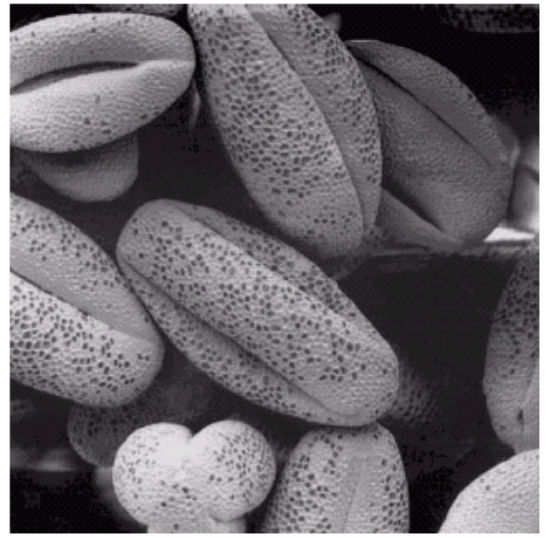
Lecture 2: [Image Preprocessing]

Instructor: [Dr. Rabia Irfan]

[Assistant Professor], SEECS, NUST

Image Enhancements



Process of improving the visual quality of an image by modifying its features to make it more suitable for specific applications or more visually appealing for human observation.

Two types:

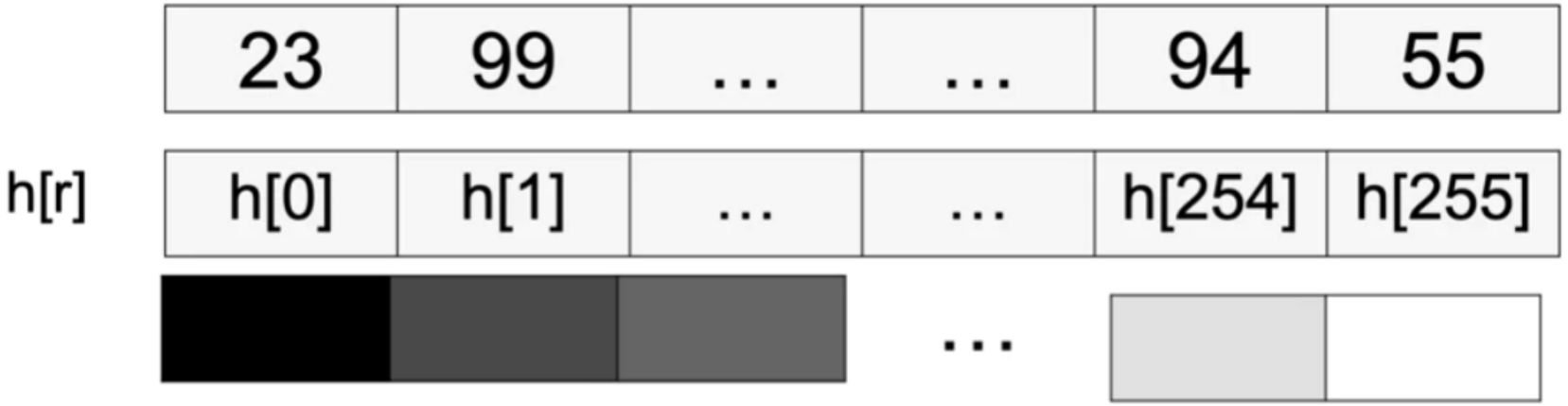
- ➔ Spatial Transformation (**Geometric** + Intensity) – Individual Pixel based
- ➔ Spatial Filtering – Pixel Neighborhood based

Agenda

- Following topics will be discussed under the topic of Image Enhancement:
  - Histograms
  - Intensity Transforms
  - Spatial Filtering

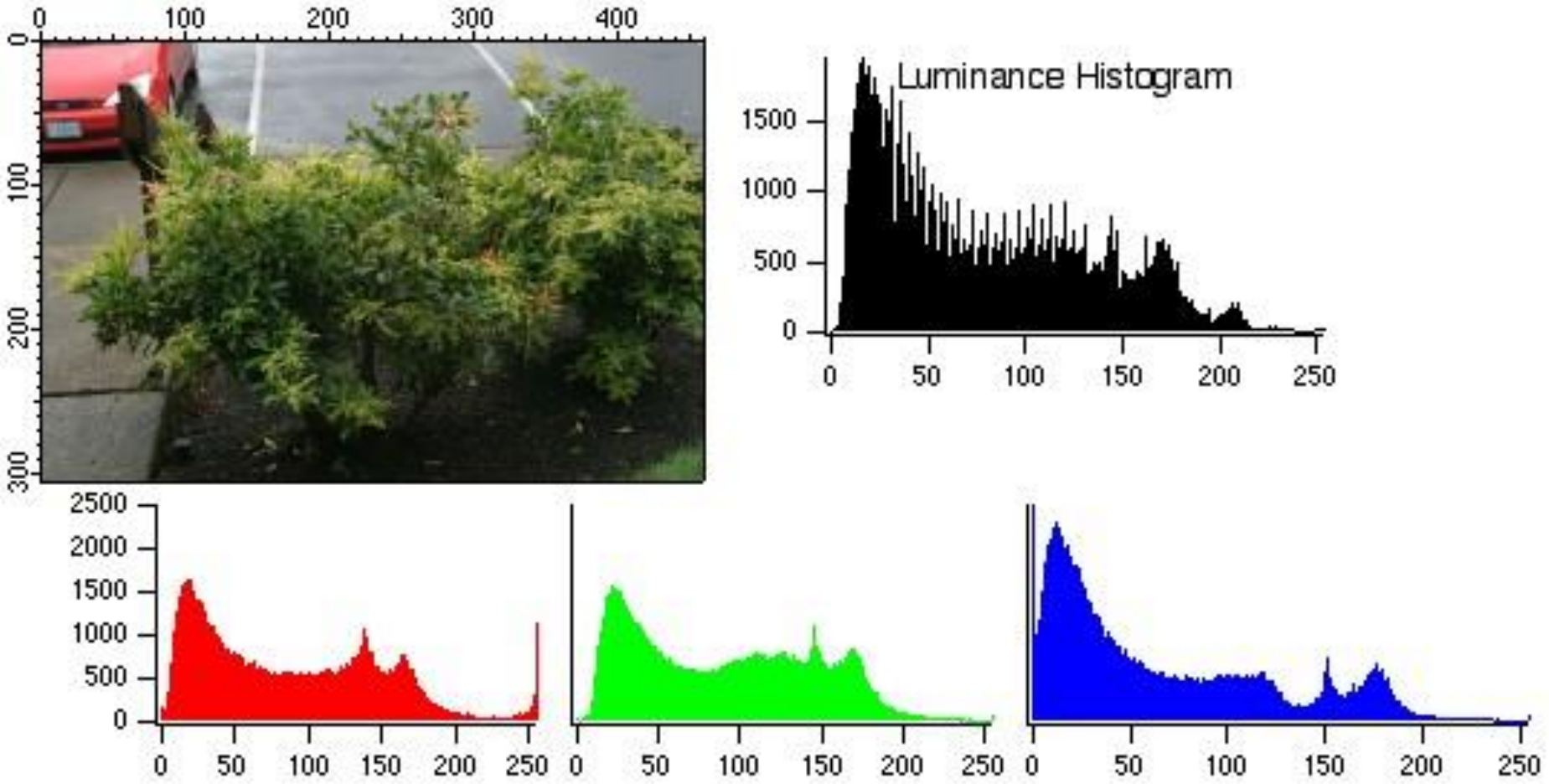
Histograms

- Counts the number of occurrences or frequencies of pixel intensities in an image

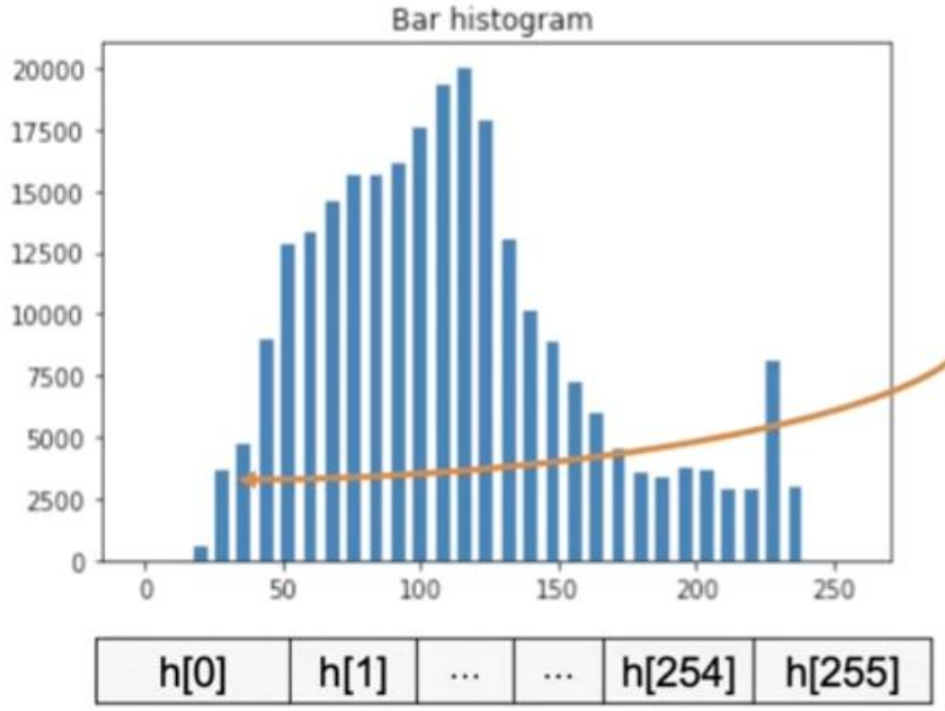


- It plots the number of pixels vs. each intensity value.

Histograms


0
100
200
300
400
Luminance Histogram
1500
1000
500
0
50
150
250
2500
2000

Histograms


Bar histogram
20000
17500
15000
12500
10000
7500
5000
2500
0
0
50
100
150
200
250
h[0]
h[1]
...
...
h[254]
h[255]

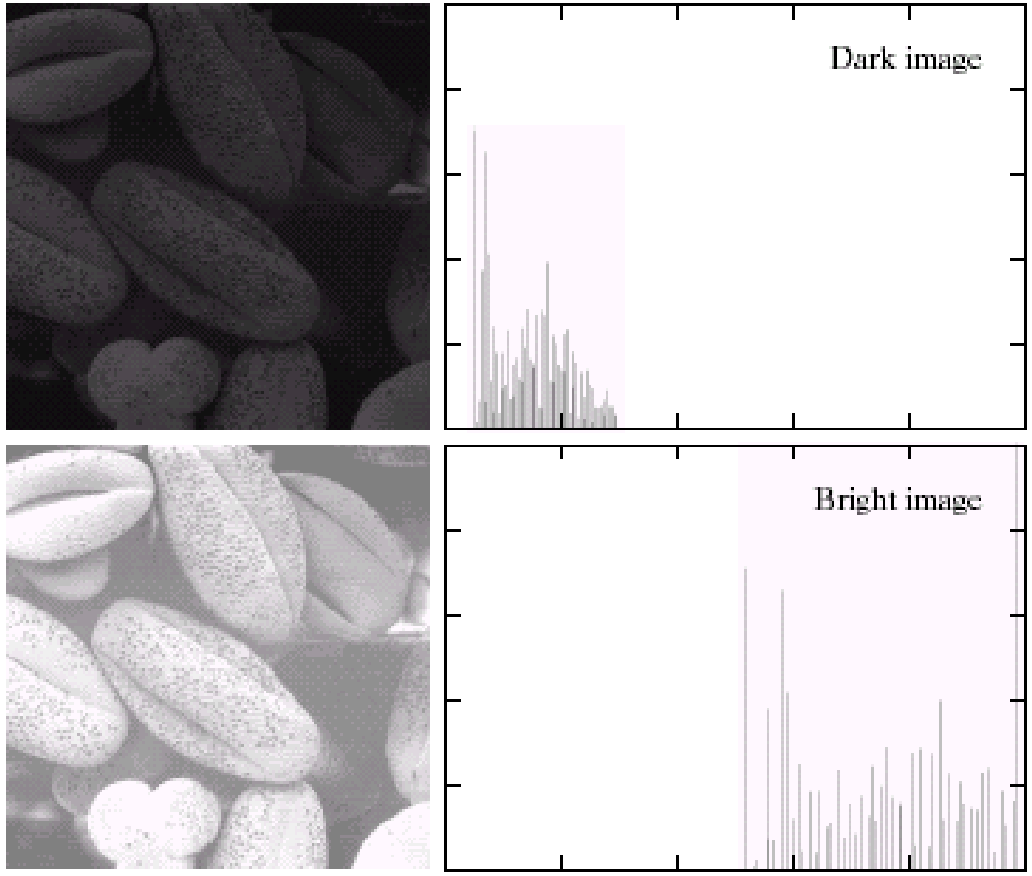

0
100
200
300
400
500
0
100
200
300
400
500

Histograms

```
import cv2

goldhill = cv2.imread("goldhill.bmp")

hist = cv2.calcHist([goldhill],[0],None,[256],[0,255])
```

34	1	…	…	94	0
0	1	2	…	254	255

Histograms

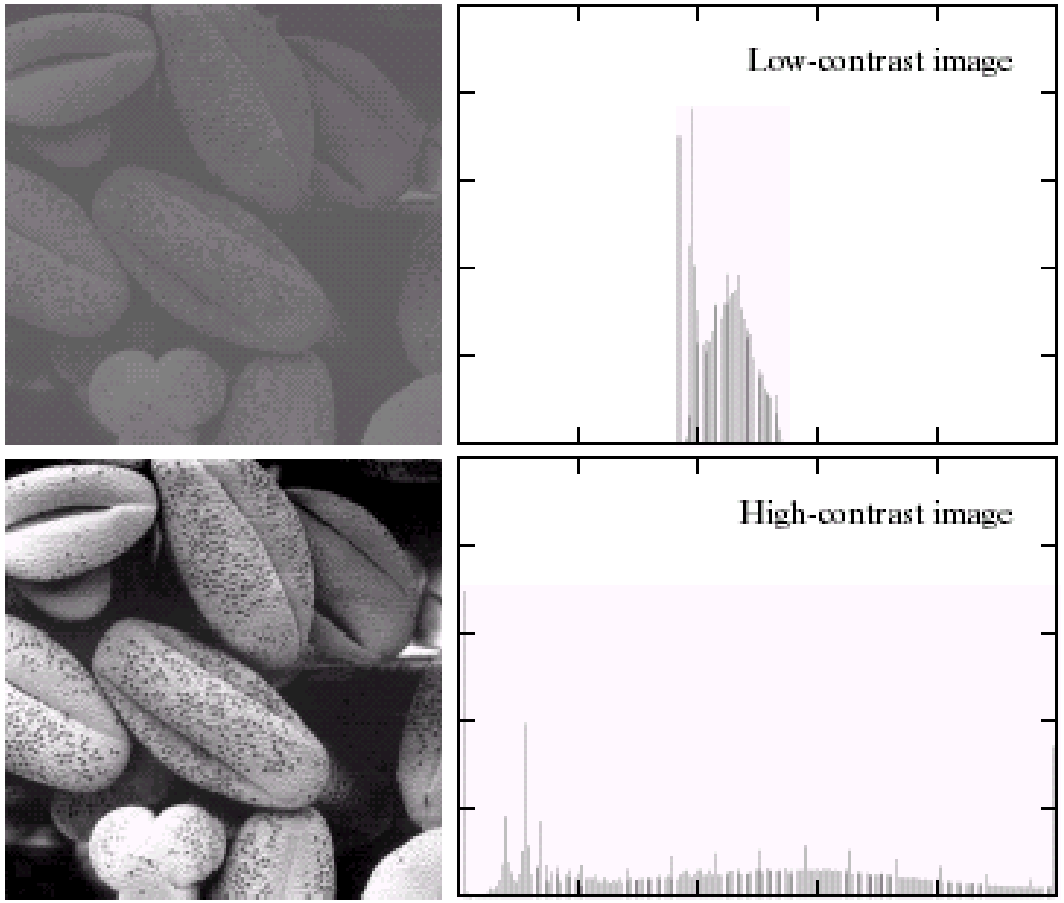


Dark Image

How would the histograms of these images look like?

Bright Image

Histograms



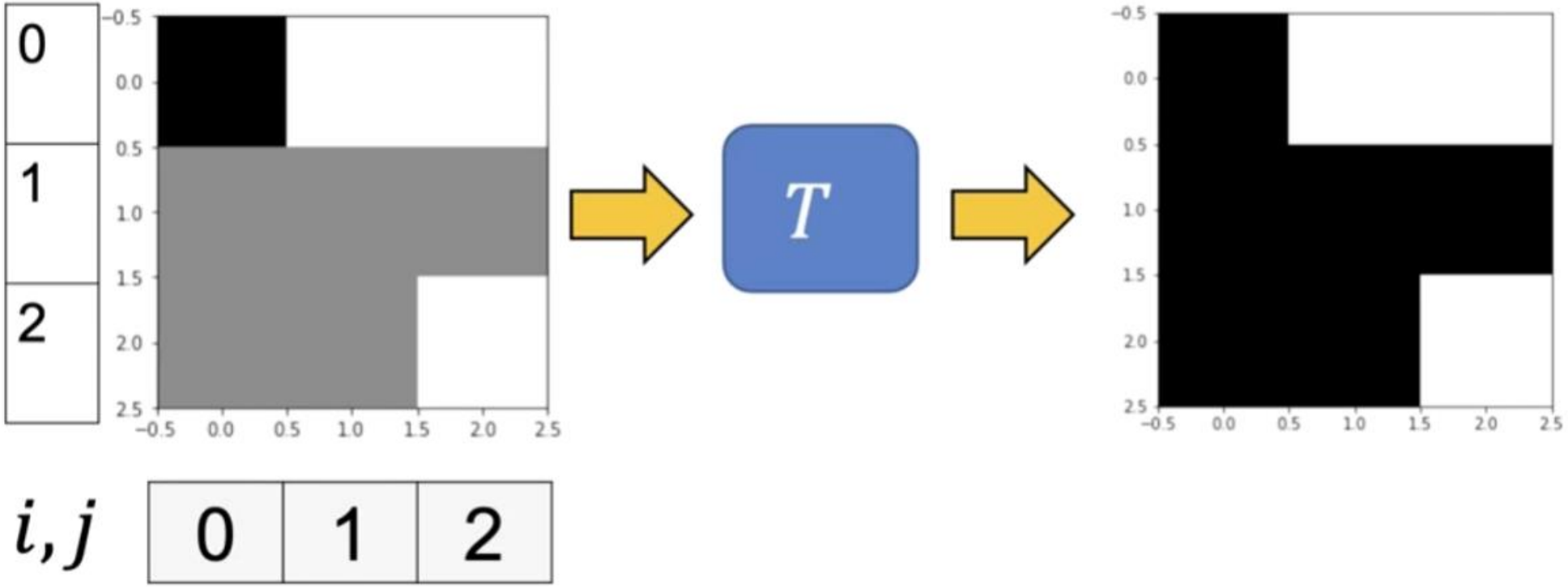
Low Contrast Image

How would the histograms of these images look like?

High Contrast Image

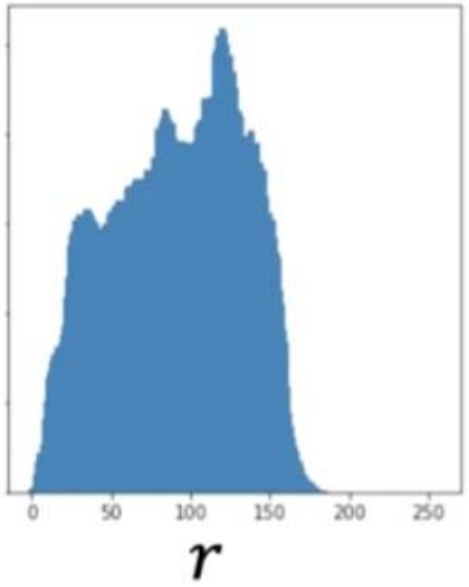
Intensity Transformations

$f[i,j]$ $g[i,j] = T\{f[i,j]\}$ $g[i,j]$



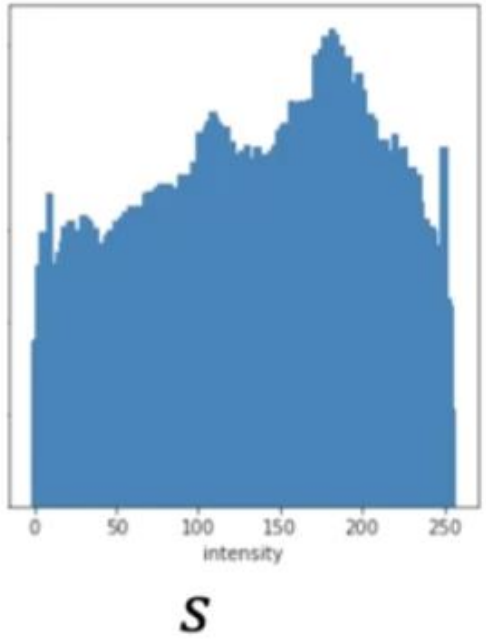
Intensity Transformations

h_r



$s = T\{r\}$

h_s



Intensity Transformations

$f[i,j]$ $\qquad g[i,j] = 2f[i,j] + 1 \qquad$ $g[i,j]$

0	2	2
1	1	1
1	1	2

1	5	5
3	3	3
3	3	5

Intensity Transformations

r	h_r
0	1
1	5
2	3
3	0
4	0
5	0
6	0

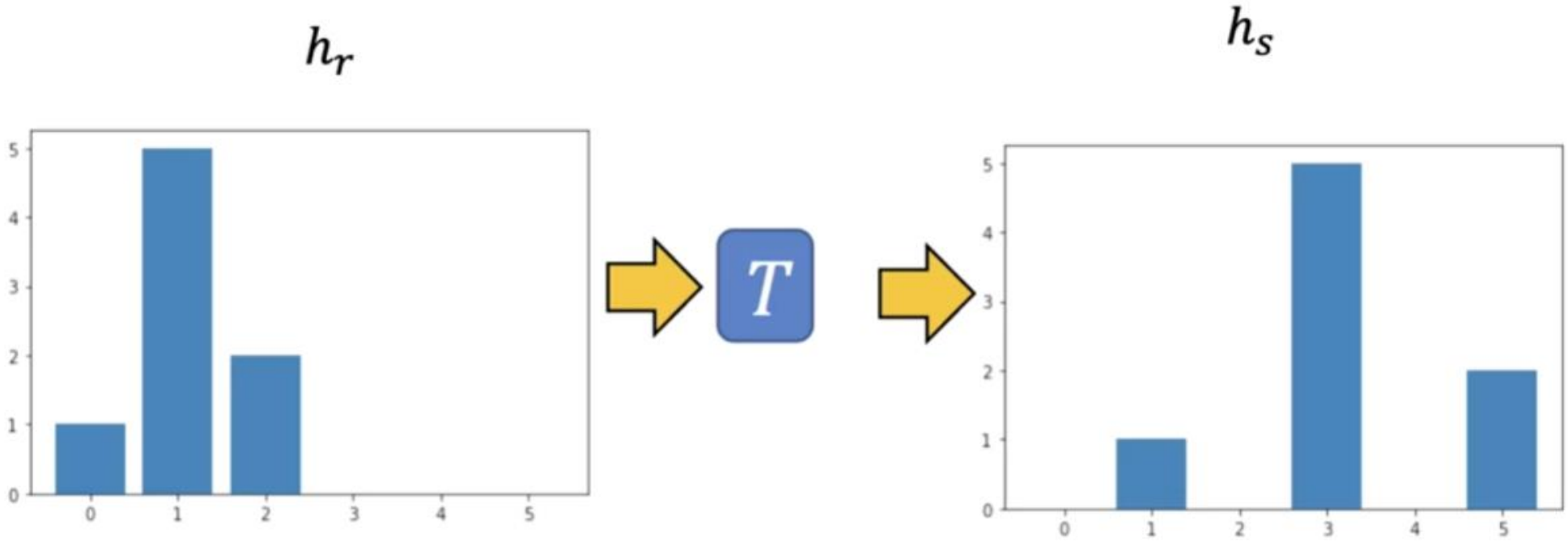
$$s = 2r + 1$$

$$= 2(1) + 1$$

$$= 3$$

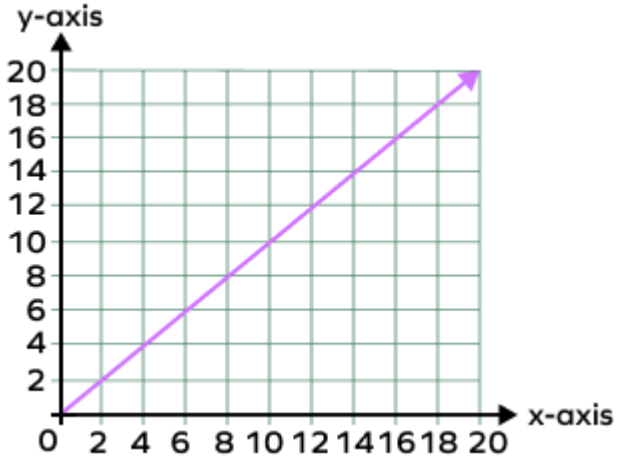
s	h_s
0	0
1	1
2	0
3	5
4	0
5	2
6	0

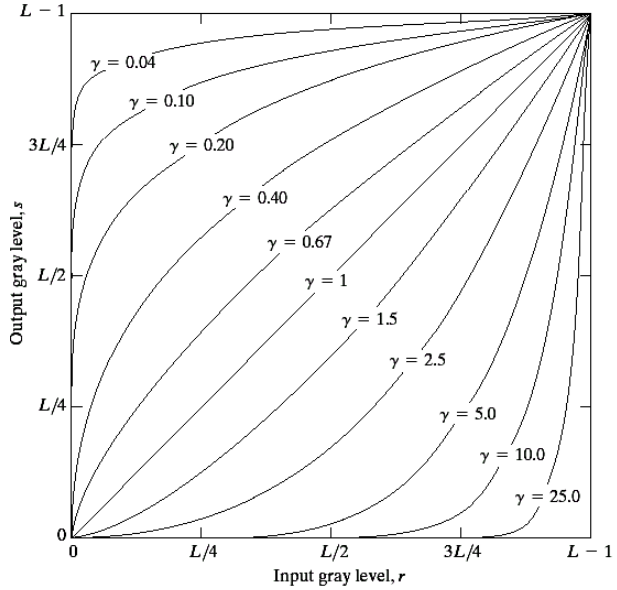
Intensity Transformations

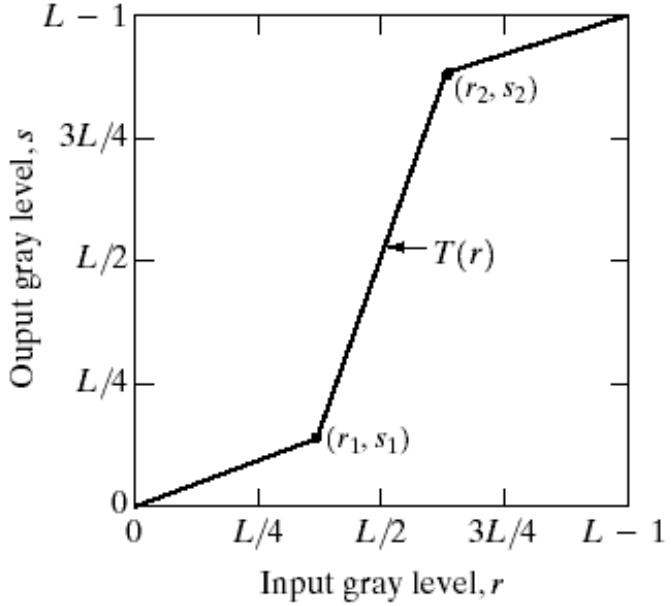


Intensity Transformations

- Linear Transformations (Image Negative)
- Non-Linear Transformations (Log, Inverse Log, Power Law)
- Piecewise-Linear Transformations







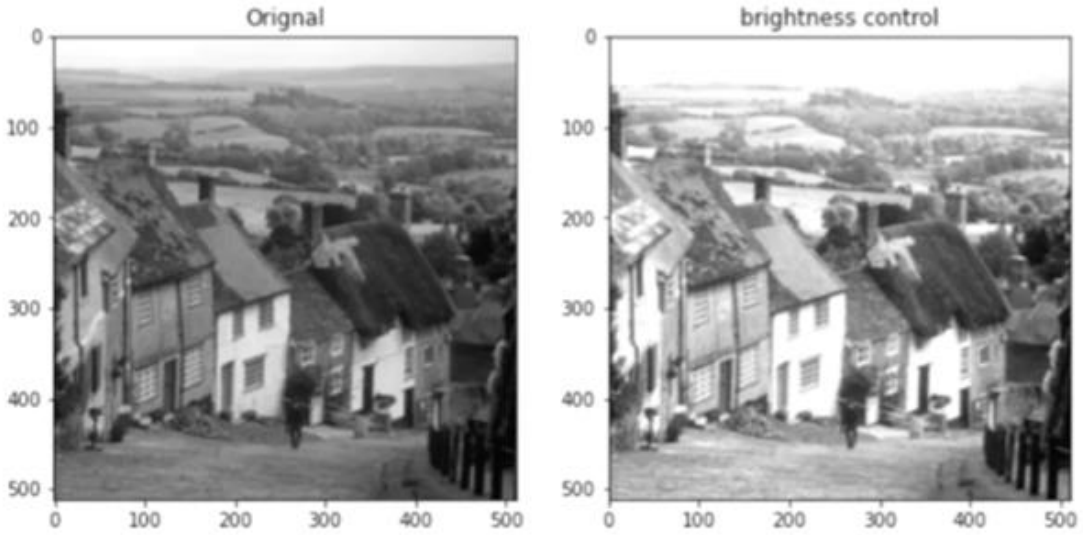
Linear Transformations - Positives

$$g[i,j] = \alpha f[i,j] + \beta$$

```
alpha = 1 # Simple contrast control
beta = 100   # Simple brightness control
```

$$g[i,j] = 1f[i,j] + 100$$

```
new_image = cv2.convertScaleAbs(goldhill, alpha=alpha, beta=beta)
```

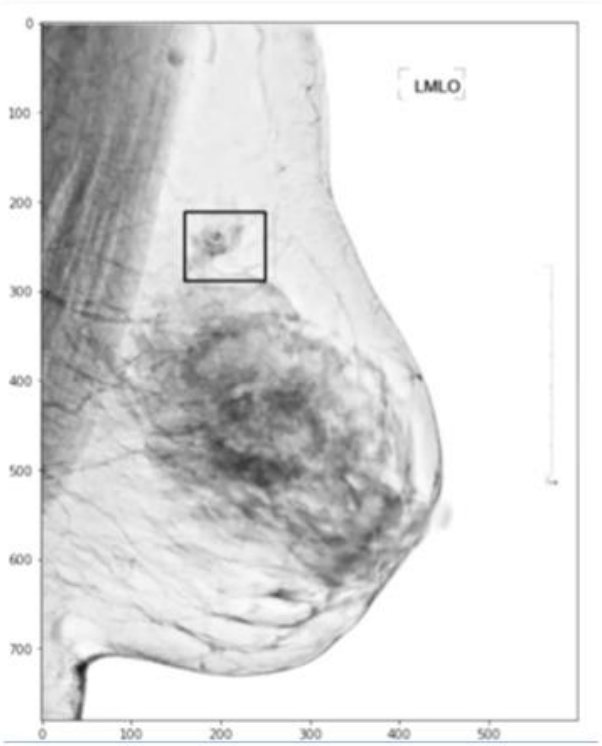


Linear Transformations - Negatives

```
image= cv2.imread("mammogram.png",cv2.IMREAD_GRAYSCALE)
```

$$g[i,j] = -1f[i,j] + 255$$

```
img_neg=-1* image+255
```



Jian, Wushuai, Xueyan Sun, and Shuqian Luo. "Computer-aided diagnosis of breast microcalcifications based on dual-tree complex wavelet transform." Biomedical engineering online 11.1 (2012): 1-12.

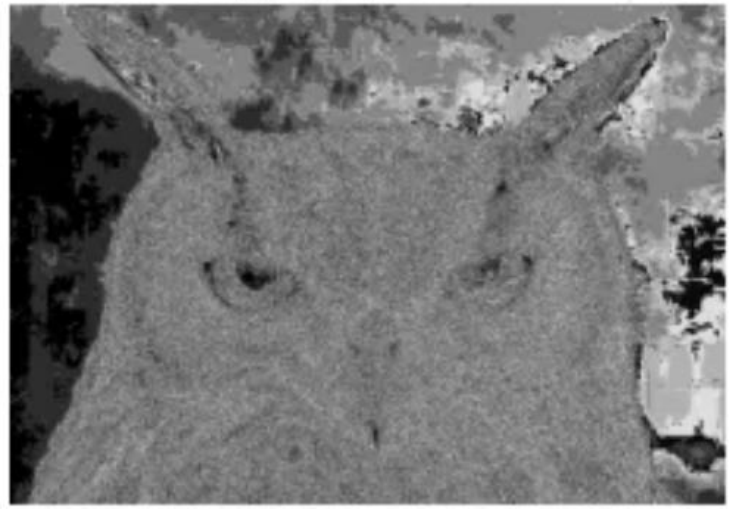
Non-Linear Transformations – Log & Inverse Log

- **Enhances Dark Regions**: The log transform makes dim areas brighter, helping to reveal details that would otherwise be difficult to see.
- **Compresses Bright Pixels**: Very bright pixels have smaller increments after transformation, preventing them from overwhelming the image.
- Opposite is done by Inverse Log.

Original Image

log_transformed

inverse_log_transformed



Non-Linear Transformations – Power Law

MR image of fractured human spine

Result after Power law transformation c = 1, $\gamma = 0.6$

Result after Power law transformation c = 1, $\gamma = 0.4$

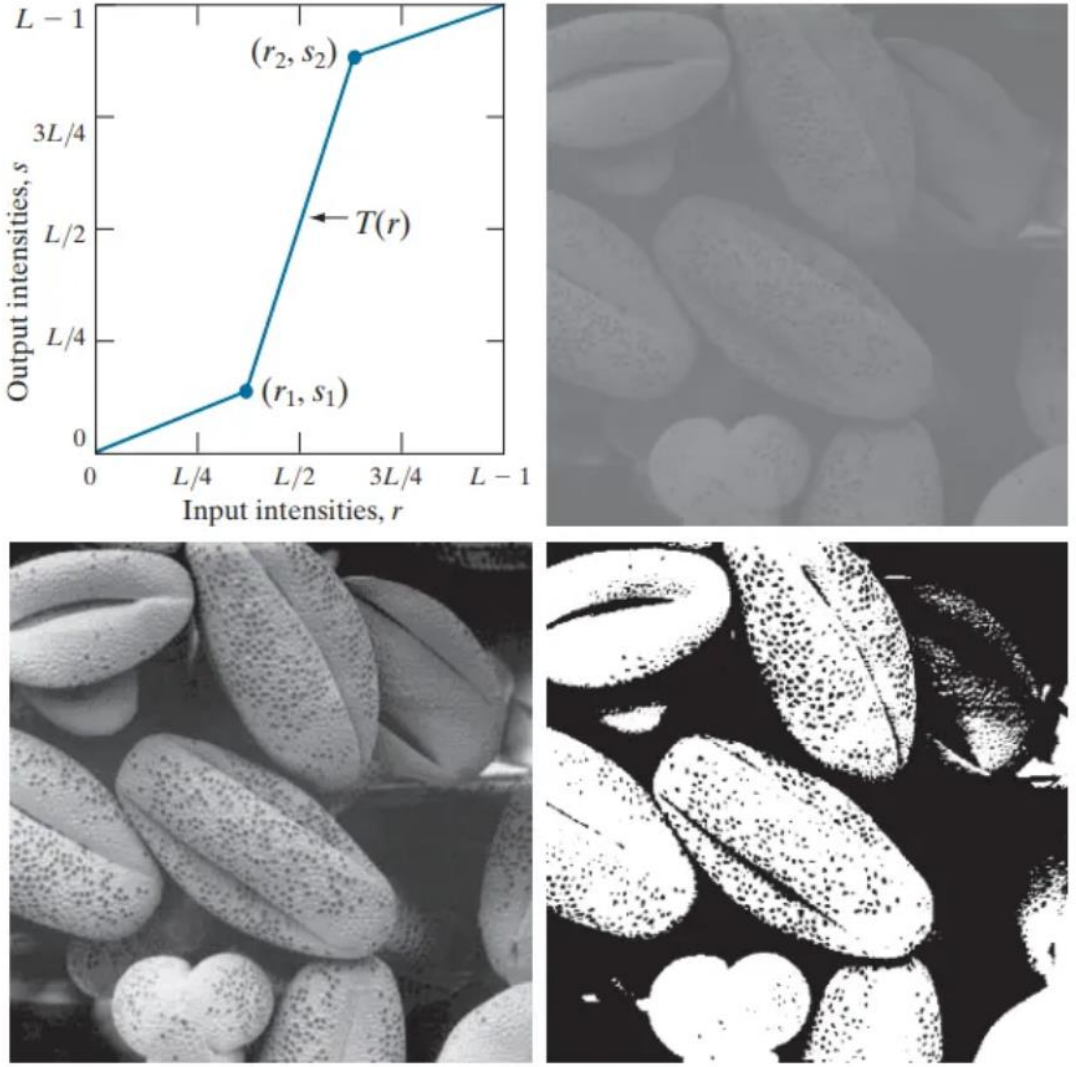
Result after Power law transformation c = 1, $\gamma = 0.3$

Piece-wise Linear Transformation

- Types:
  - Contrast Stretching
  - Histogram Equivalization
  - Thresholding
  - Intensity Level Slicing
  - Bit-plane Slicing

a b
c d

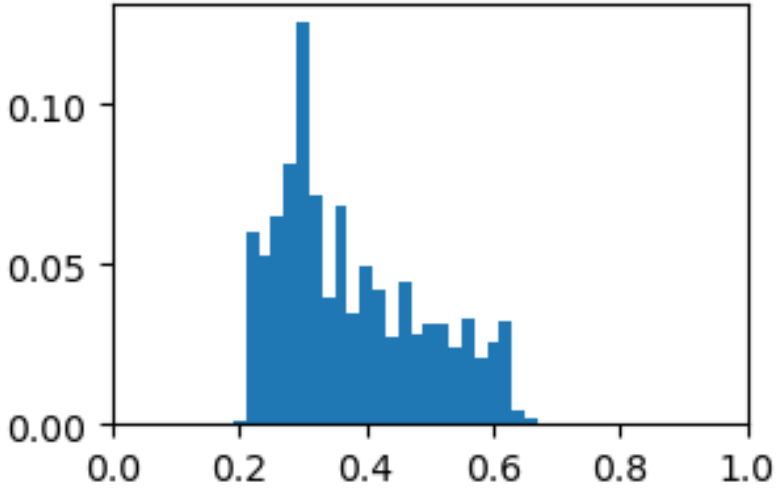
FIGURE Contrast stretching. (a) Piecewise linear transformation function. (b) A low-contrast electron microscope image of pollen, magnified 700 times. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)



Contrast Stretching

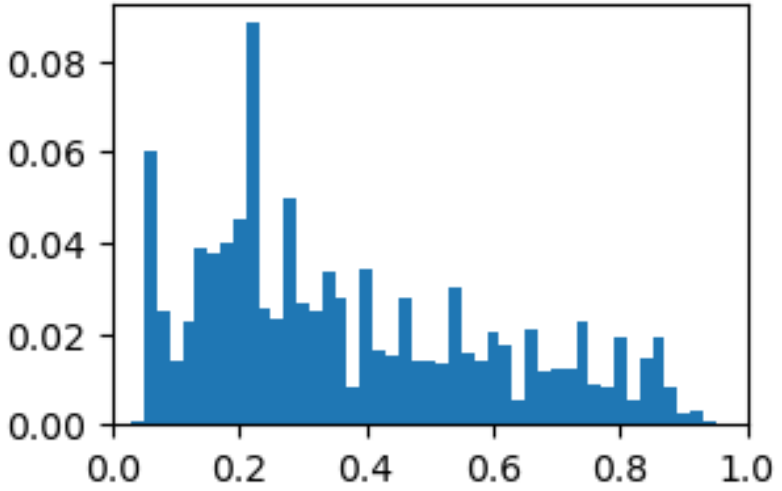
Low contrast orginal

Histogram of low contrast image



Contrast Stretched

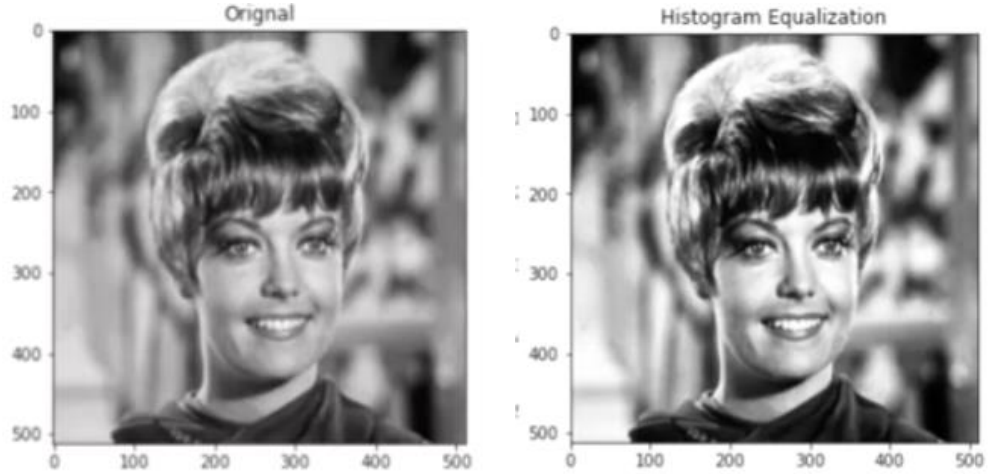
Histogram of contrast stretched image

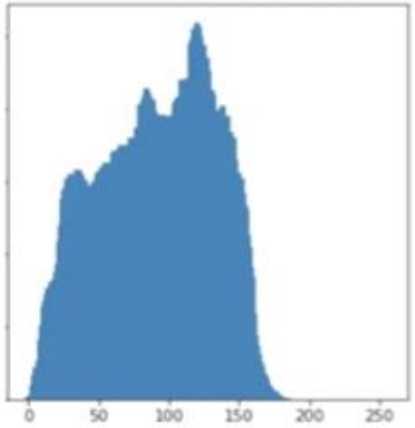


Piecewise-Linear Transformations

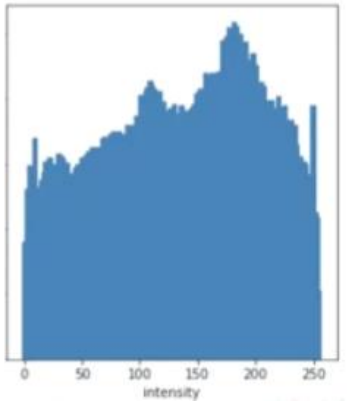
- Histogram Equalization: An algorithm that uses the histogram of images to adjust contrast

```
zelda= cv2.imread("zelda.png",cv2.IMREAD_GRAYSCALE)

new_image= cv2.equalizeHist(zelda)
```

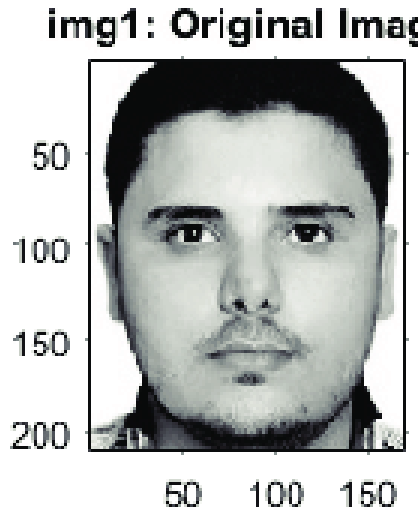


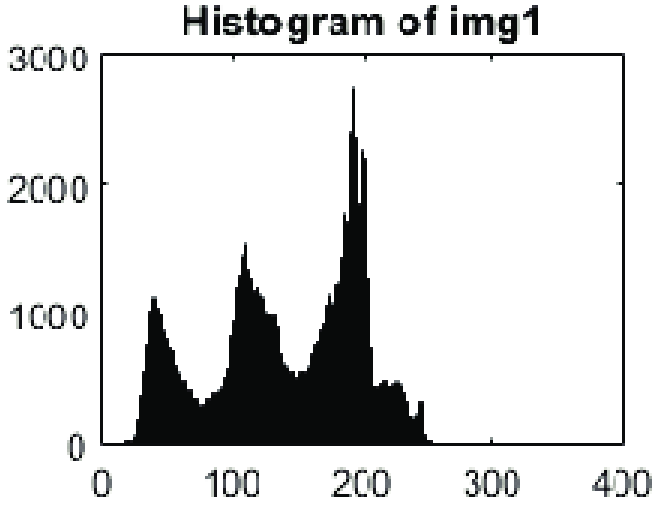


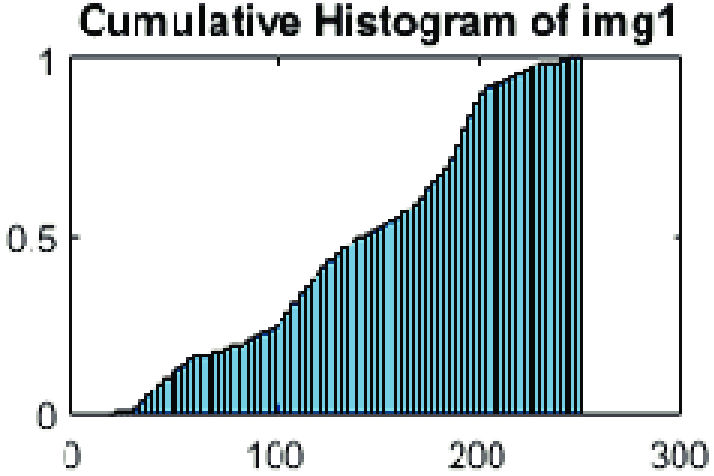
Histogram Equalization

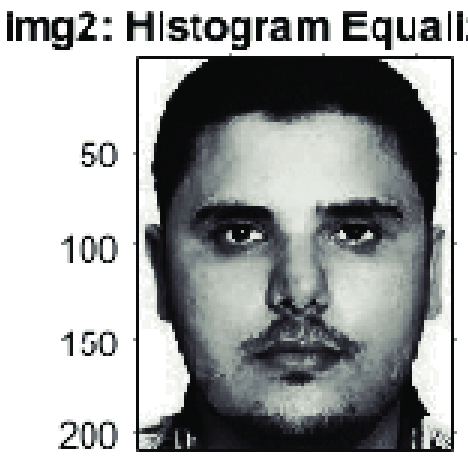


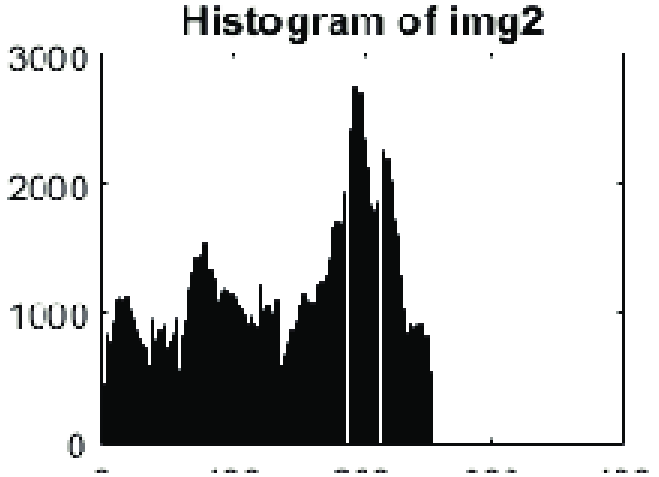
Histogram Equalization

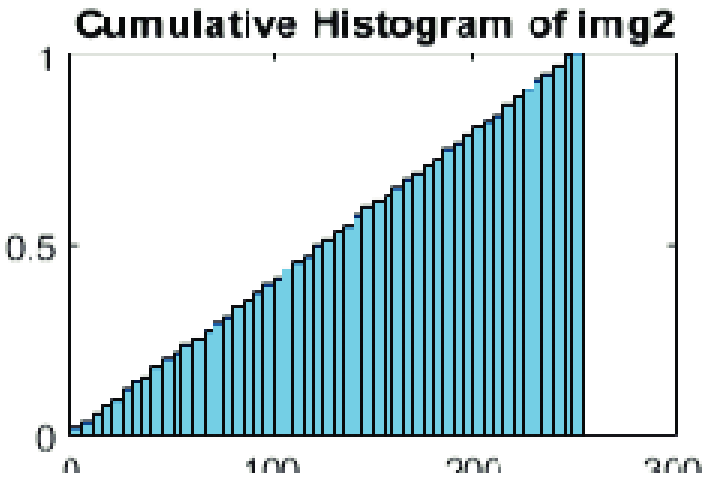






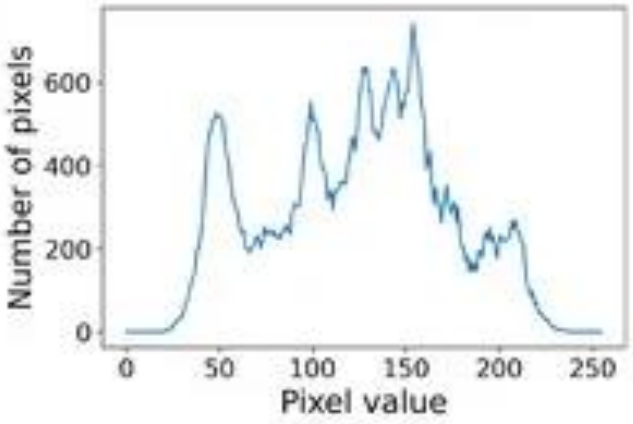




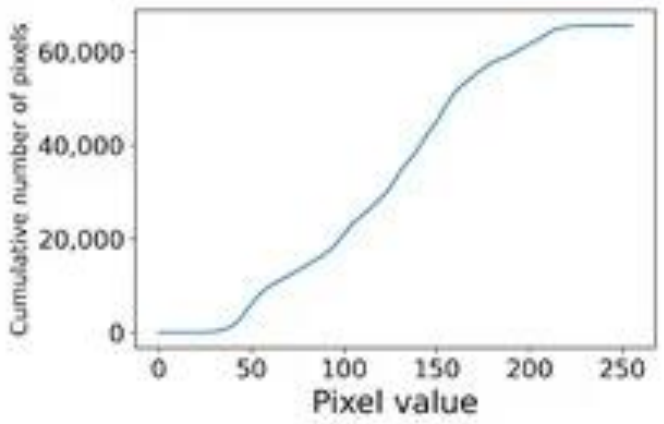


Histogram Equalization

(a) Original image

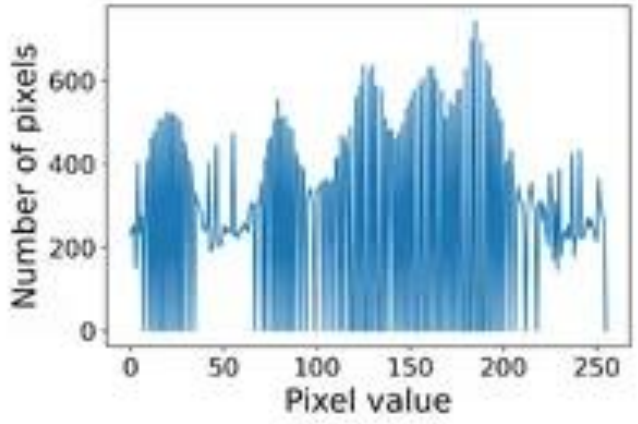


(b) Histogram

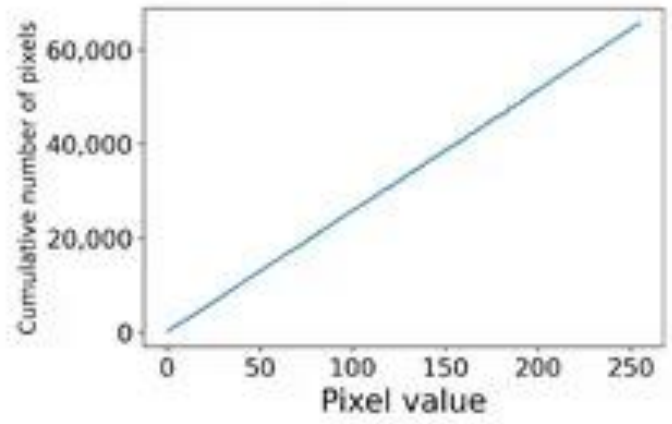


(c) Cumulative histogram

(d) Histogram-equalized image



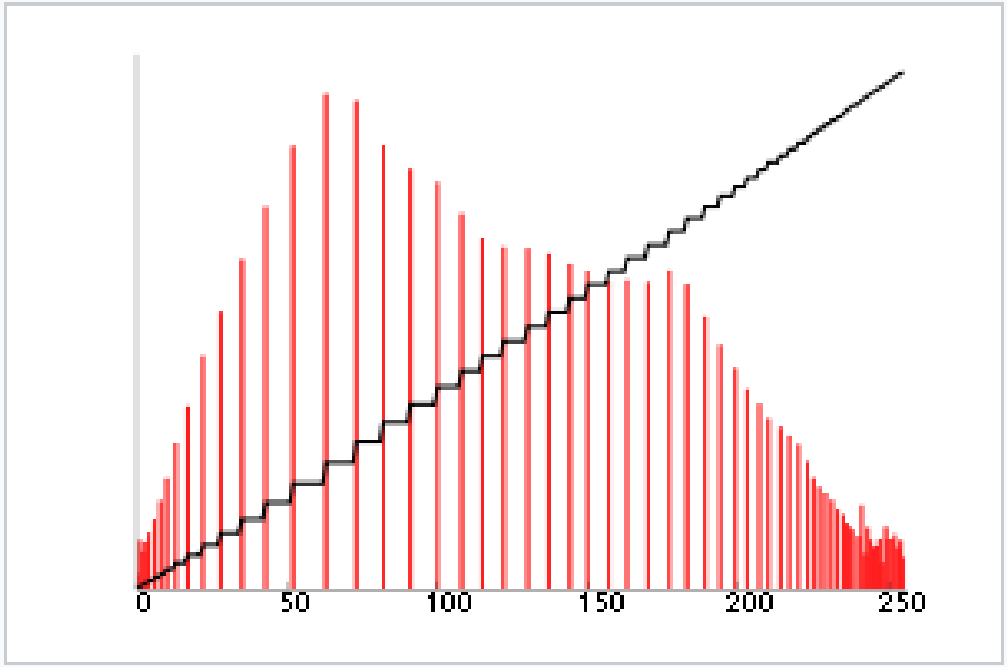
(e) Histogram



(f) Cumulative histogram

Histogram Equalization

After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Thresholding and Simple Segmentation

input_img

i			
0	**0**	**2**	**2**
1	**1**	**1**	**1**
2	**1**	**1**	**2**
j	0	1	2

```
for i  in range(N):
    for j in range(M):
        if input_img[i,j]> 1:
            image_out[i,j]=255
        else:
            image_out[i,j]=0
```

image_out

Thresholding and Simple Segmentation

input_img

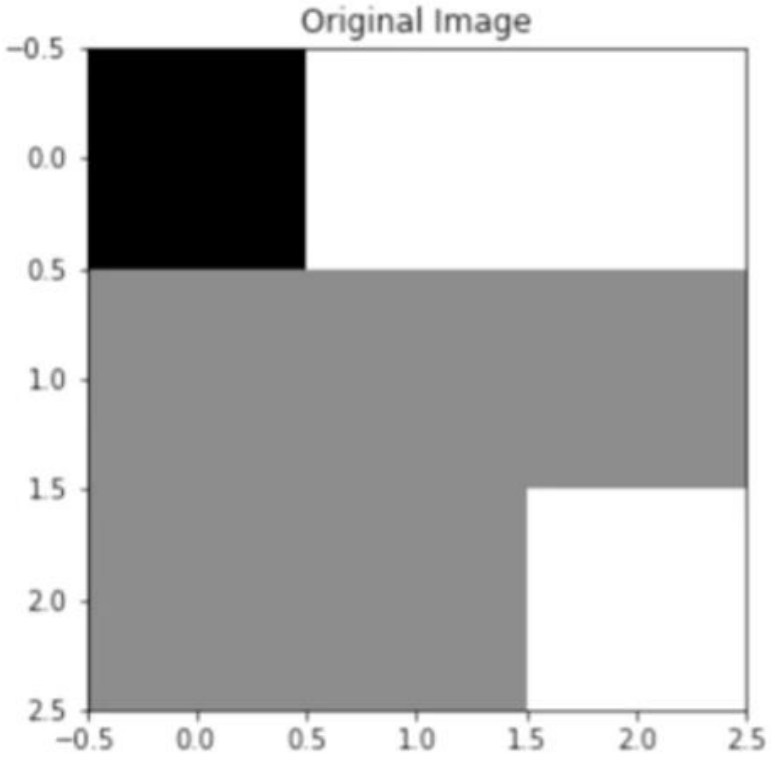
i			
0	**0**	**2**	**2**
1	**1**	**1**	**1**
2	**1**	**1**	**2**
j	0	1	2

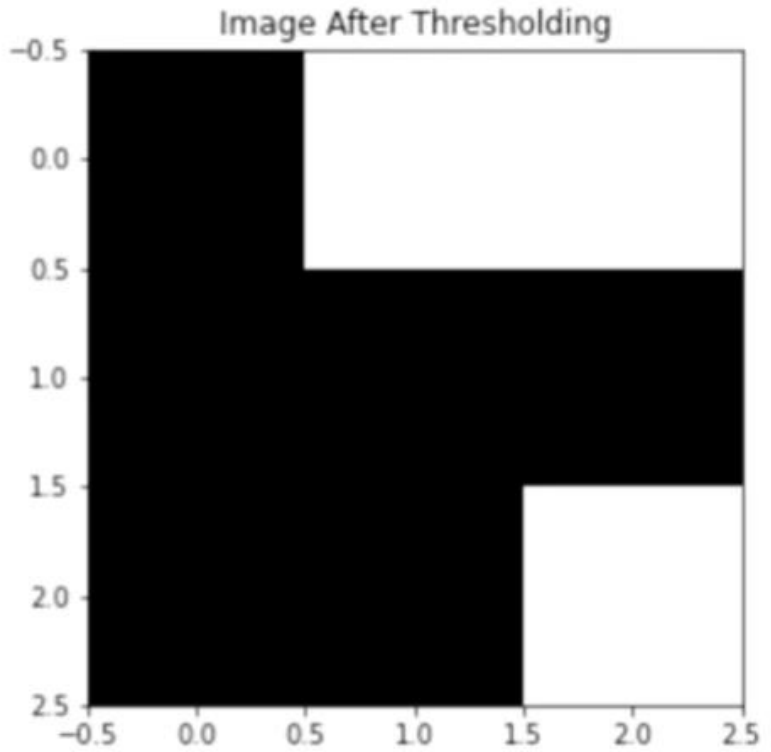
```
for i  in range(N):
    for j in range(M):
        if input_img[i,j]> 1:
            image_out[i,j]=2
        else:
            image_out[i,j]=0
```

image_out

0	**255**	**255**
0	**0**	**0**

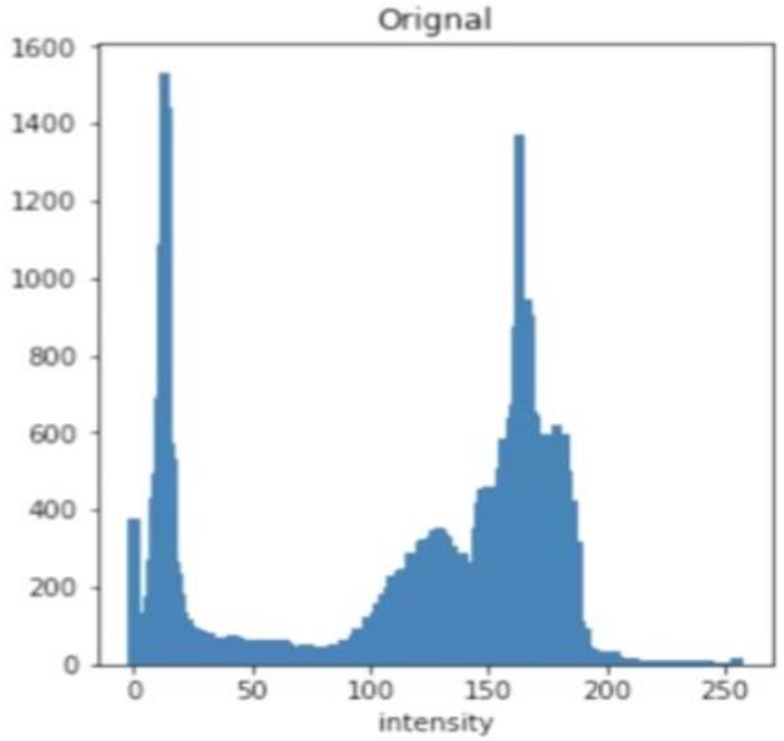
Thresholding and Simple Segmentation





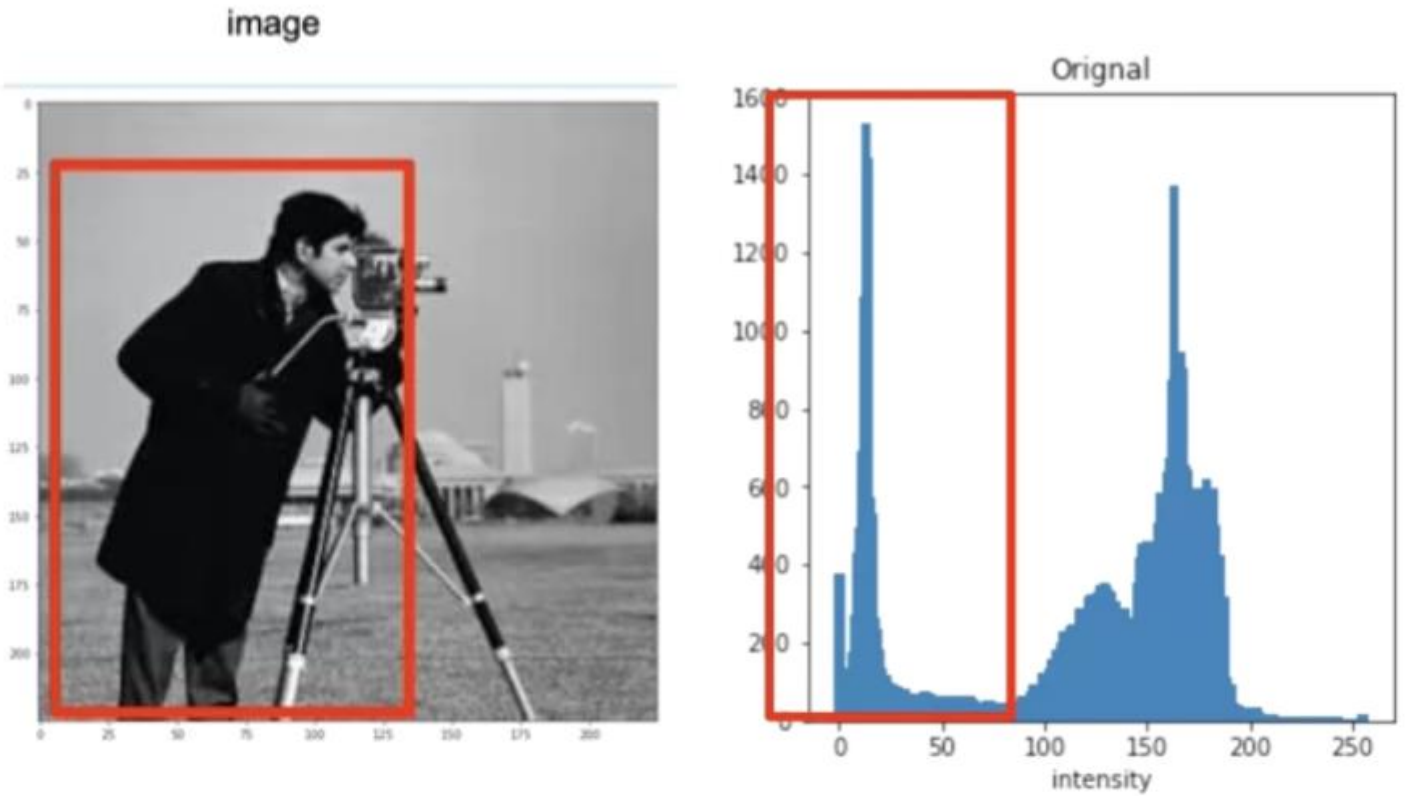
Thresholding and Simple Segmentation

```
image= cv2.imread("cameraman.jpeg",cv2.IMREAD_GRAYSCALE)
```



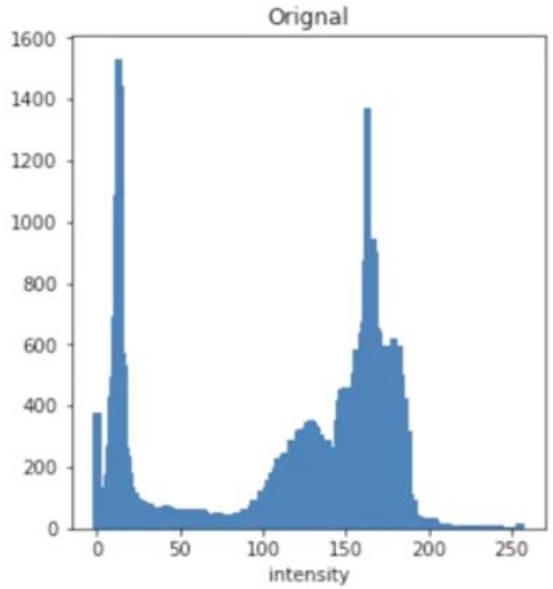
Thresholding and Simple Segmentation

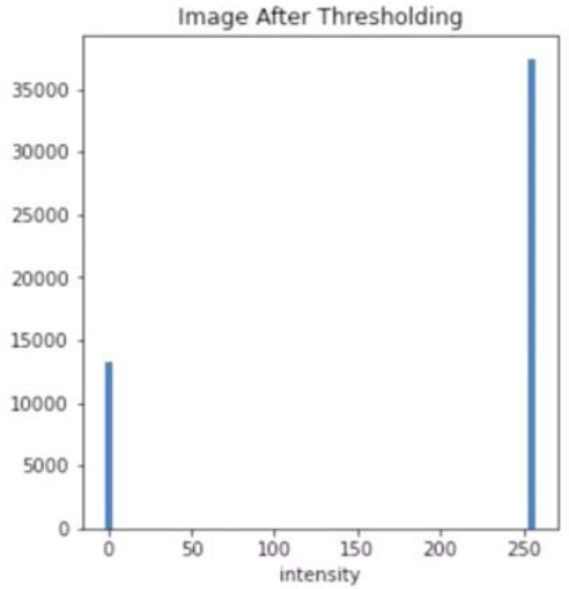
```
image= cv2.imread("cameraman.jpeg",cv2.IMREAD_GRAYSCALE)

max_value=255
threshold=87
```

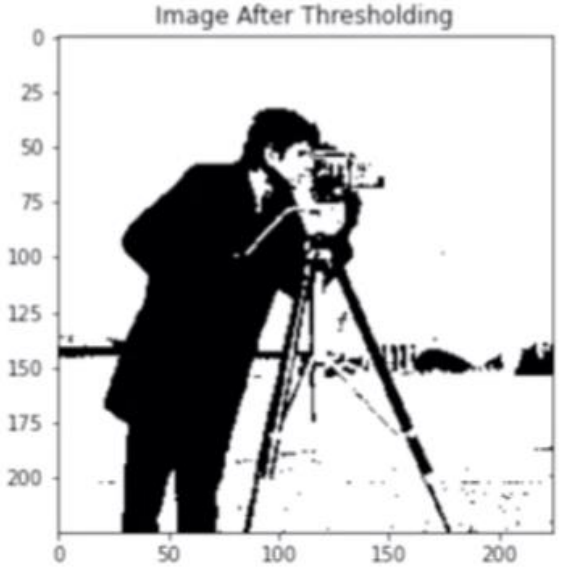


Thresholding and Simple Segmentation

```
image= cv2.imread("cameraman.jpeg",cv2.IMREAD_GRAYSCALE)

max_value=255
threshold=87

ret, new_image = cv2.threshold(image,threshold,max_value,cv2.THRESH_BINARY)
```



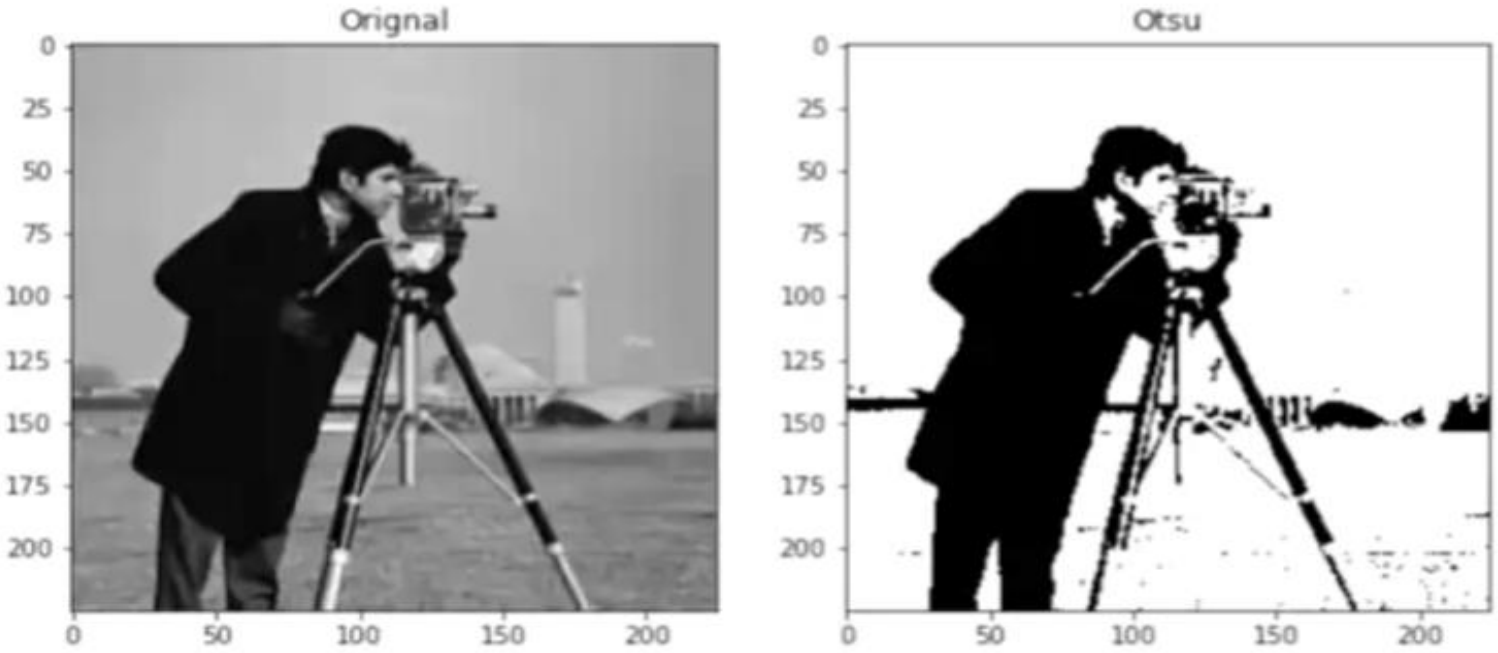




Thresholding and Simple Segmentation

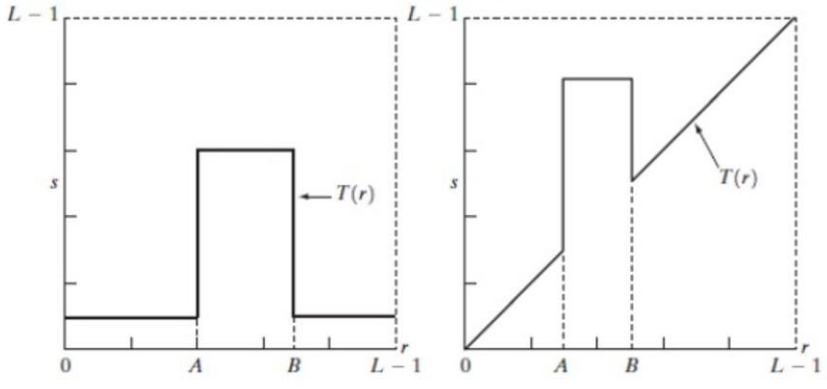
```
ret, otsu = cv2.threshold(image,0,255,cv2.THRESH_OTSU)
```

ret:88.0



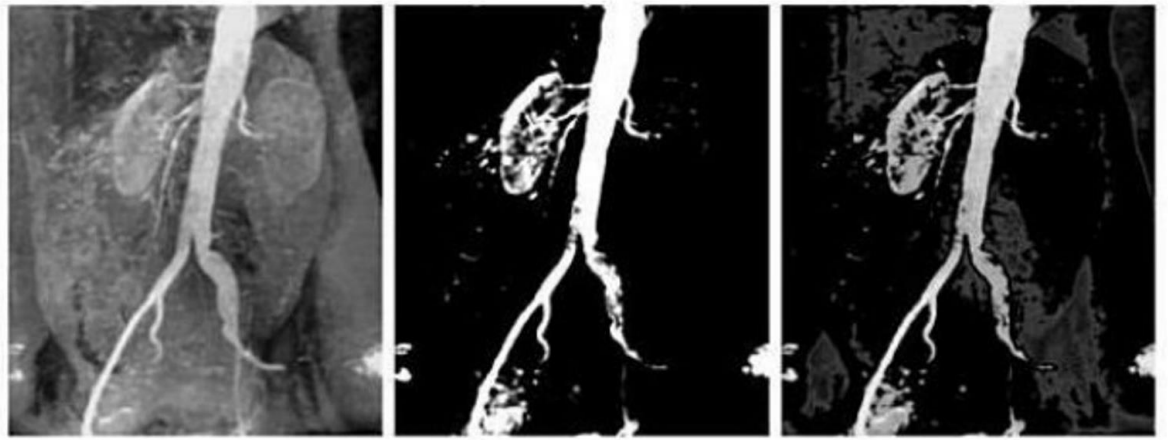
Intensity Level Slicing

Highlighting a specific range of gray levels in an image



a b
c d

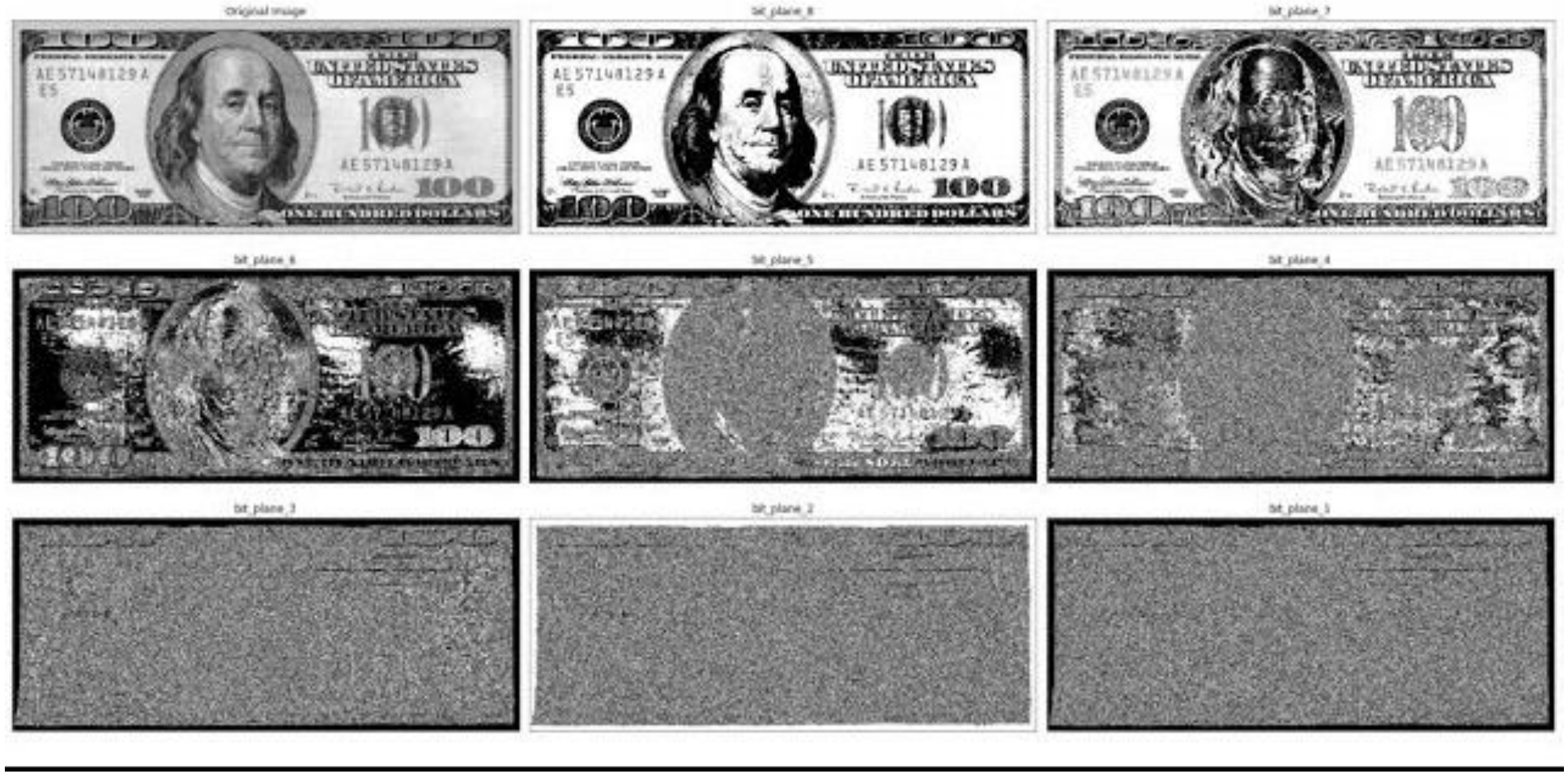
FIGURE 3.11 (a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level (b) This transformatio highlights ran $[A, B]$ but preserves all



a b c

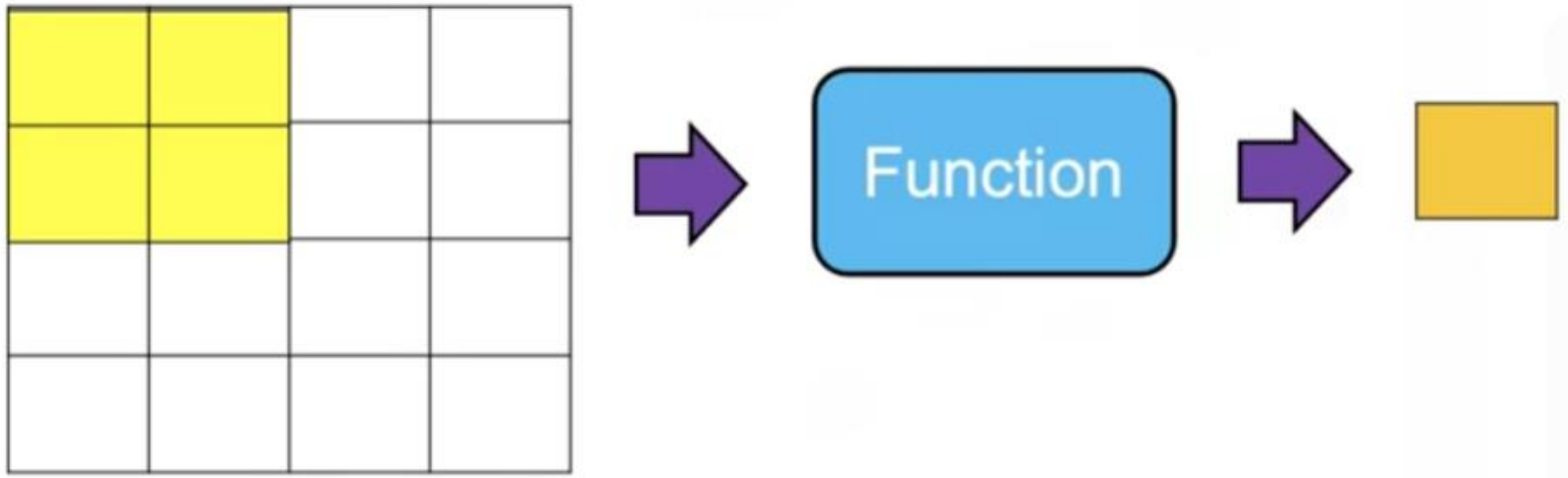
FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

Bit-plane Slicing



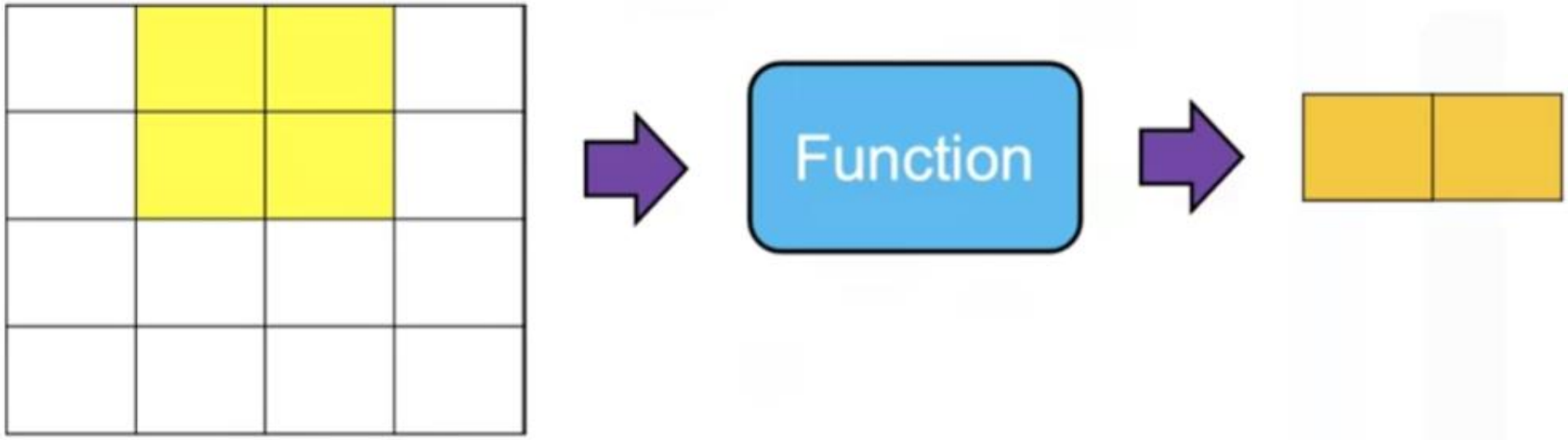
Spatial Operations

- Spatial operations consist of a neighborhood, we apply a function on the neighborhood and obtain the result.



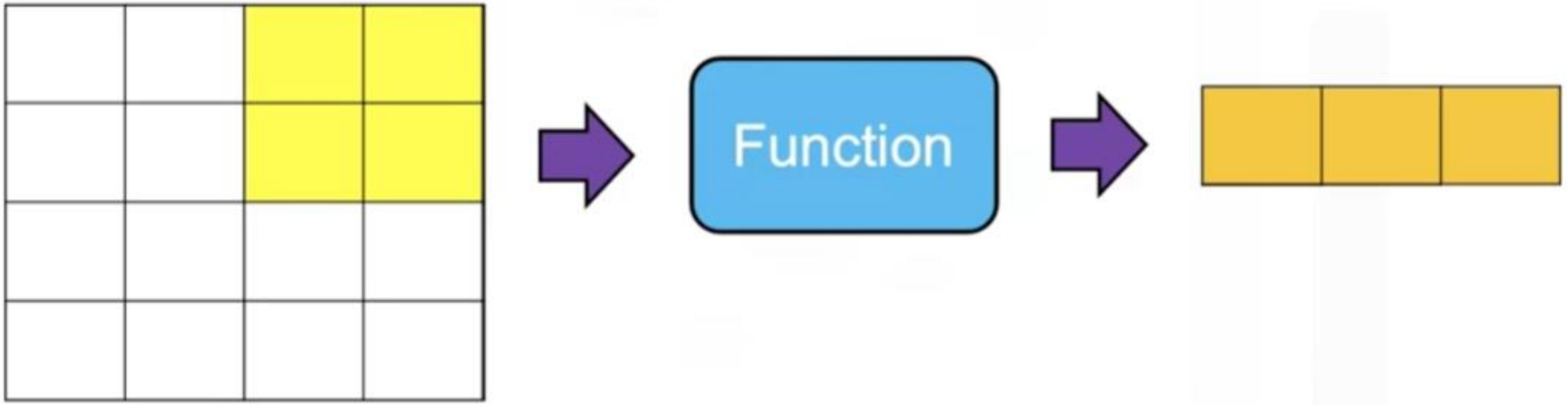
Spatial Operations

- We then shift the neighborhood using the sliding window approach and obtain the result for the next pixel in the resultant image.



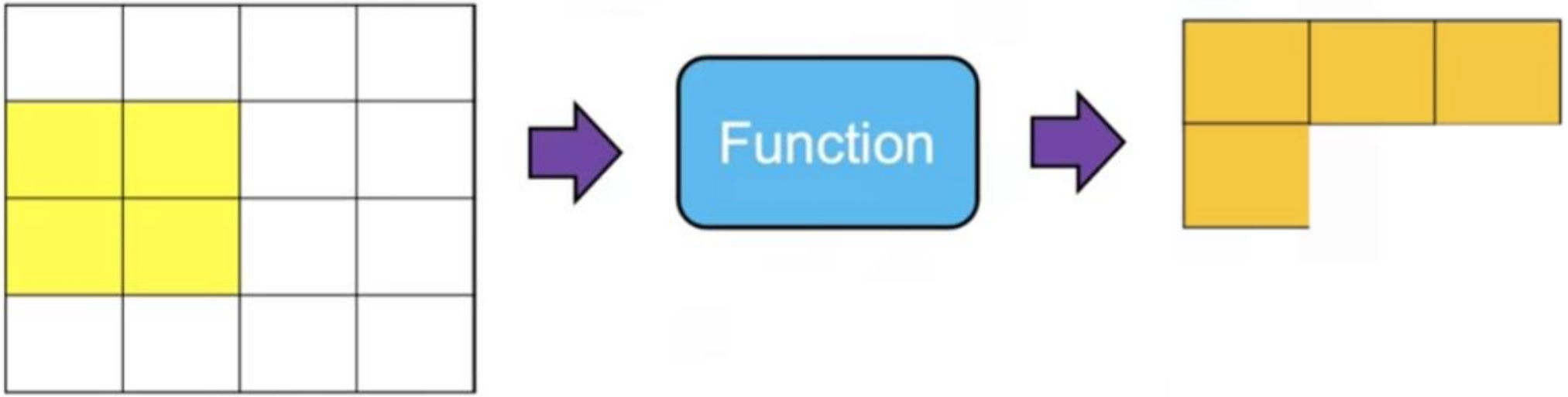
Spatial Operations

- We then shift the neighborhood using the sliding window approach and obtain the result for the next pixel in the resultant image.



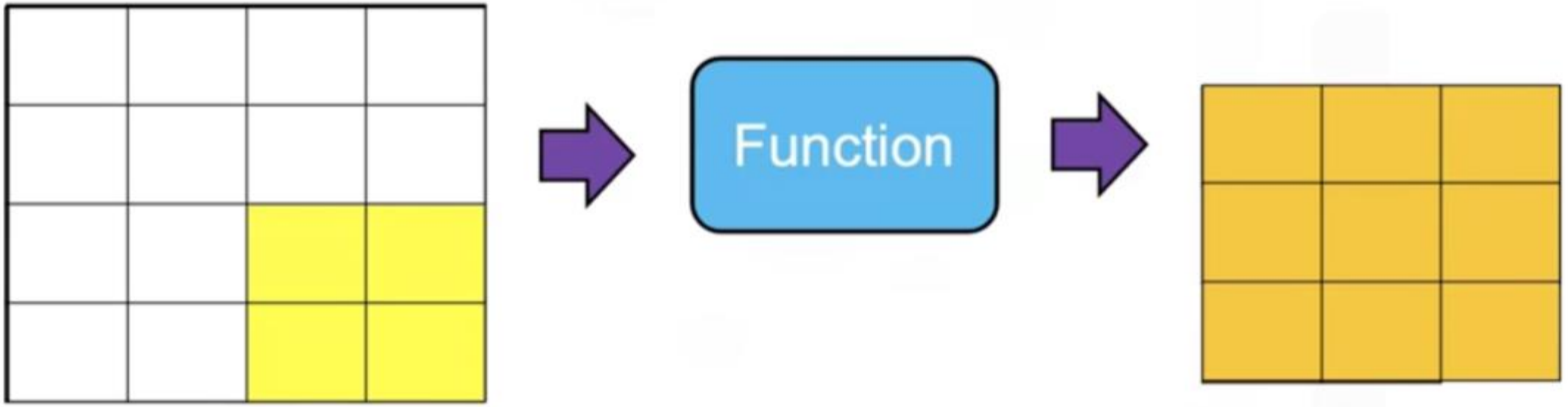
Spatial Operations

- We then shift the neighborhood using the sliding window approach and obtain the result for the next pixel in the resultant image.



Spatial Operations

- The result is a new image that has enhanced characteristics. These operations take advantage of the spatial patters in the image.



Kernels

A kernel is a filter that performs a specific task on the neighboring pixels determined by the selected window size.

image

x

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

kernel

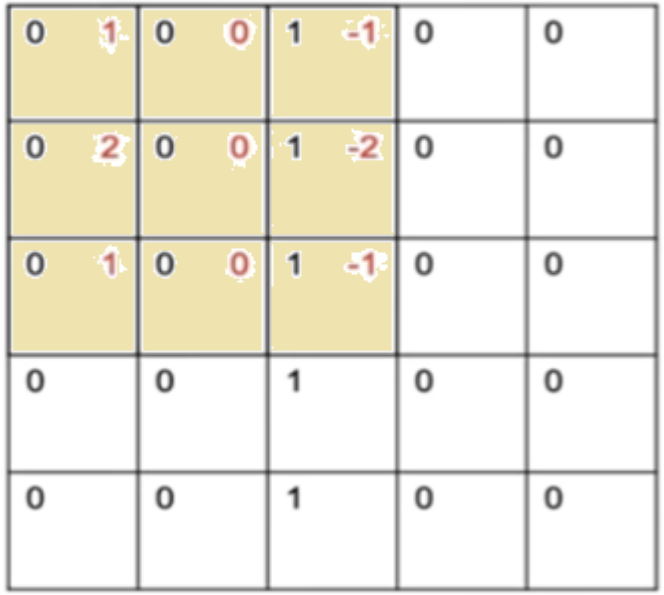
w

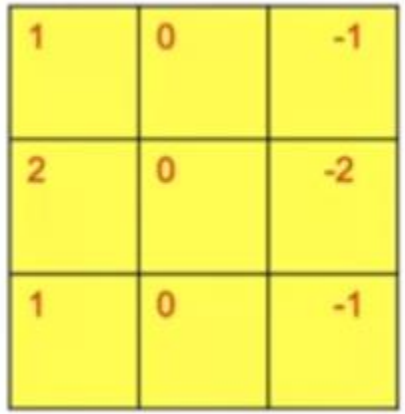
1	0	-1
2	0	-2
1	0	-1

Convolution – Linear Filtering

Convolution is a linear operation that multiplies the corresponding values of the filter with the pixels in the image and sums the prod **X**

$$Z = W * \mathbf{X}$$

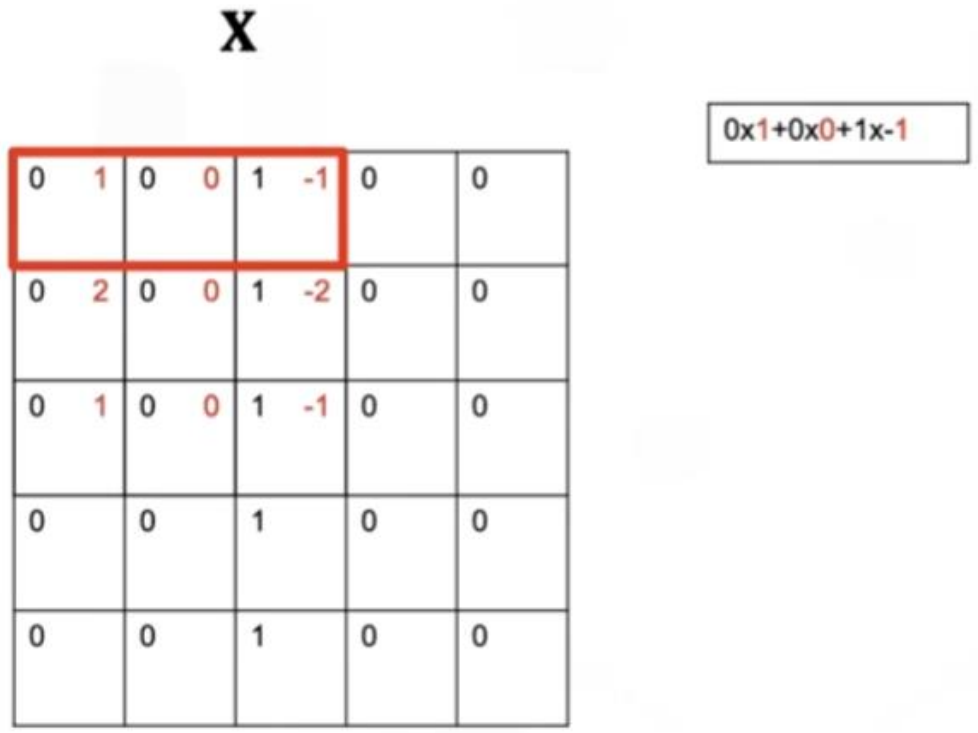




Convolution

Convolution is a linear filter that multiplies the corresponding values of the filter with the pixels in the image and sums the products.

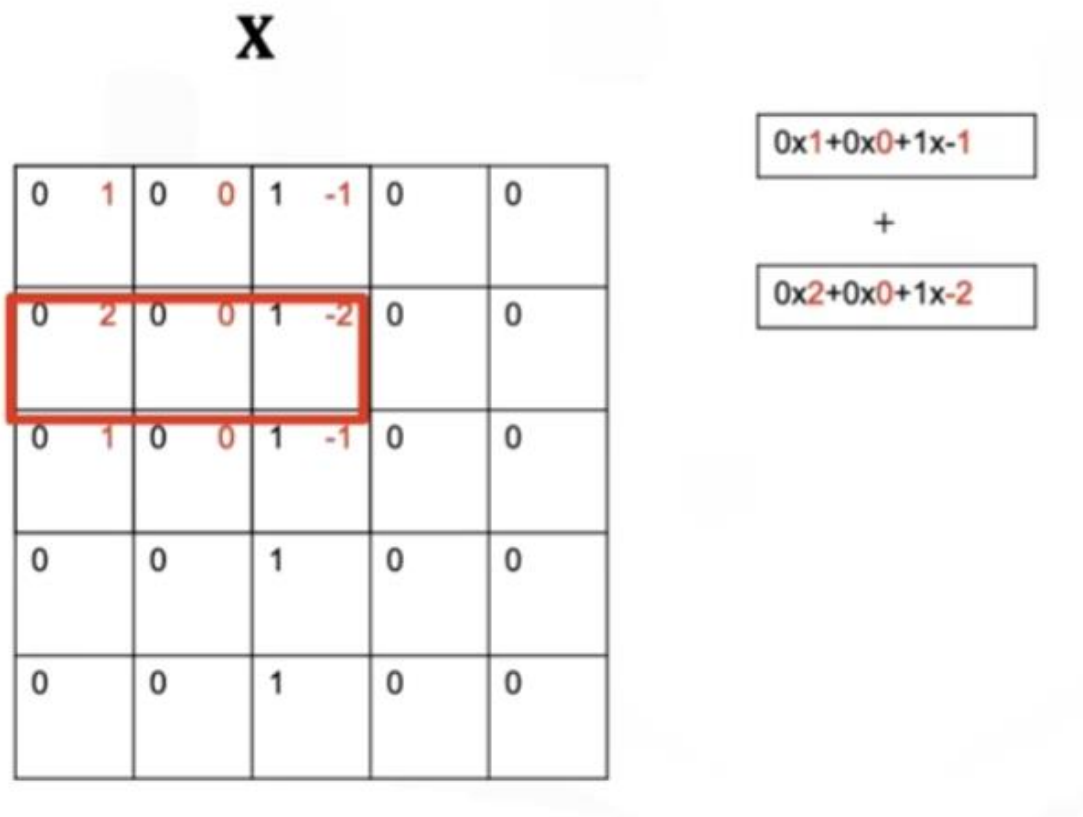
$$Z = W * X$$



Convolution

Convolution is a linear filter that multiplies the corresponding values of the filter with the pixels in the image and sums the products.

$$Z = W * X$$



Convolution

Convolution is a linear filter that multiplies the corresponding values of the filter with the pixels in the image and sums the products.

$$Z = W * X$$

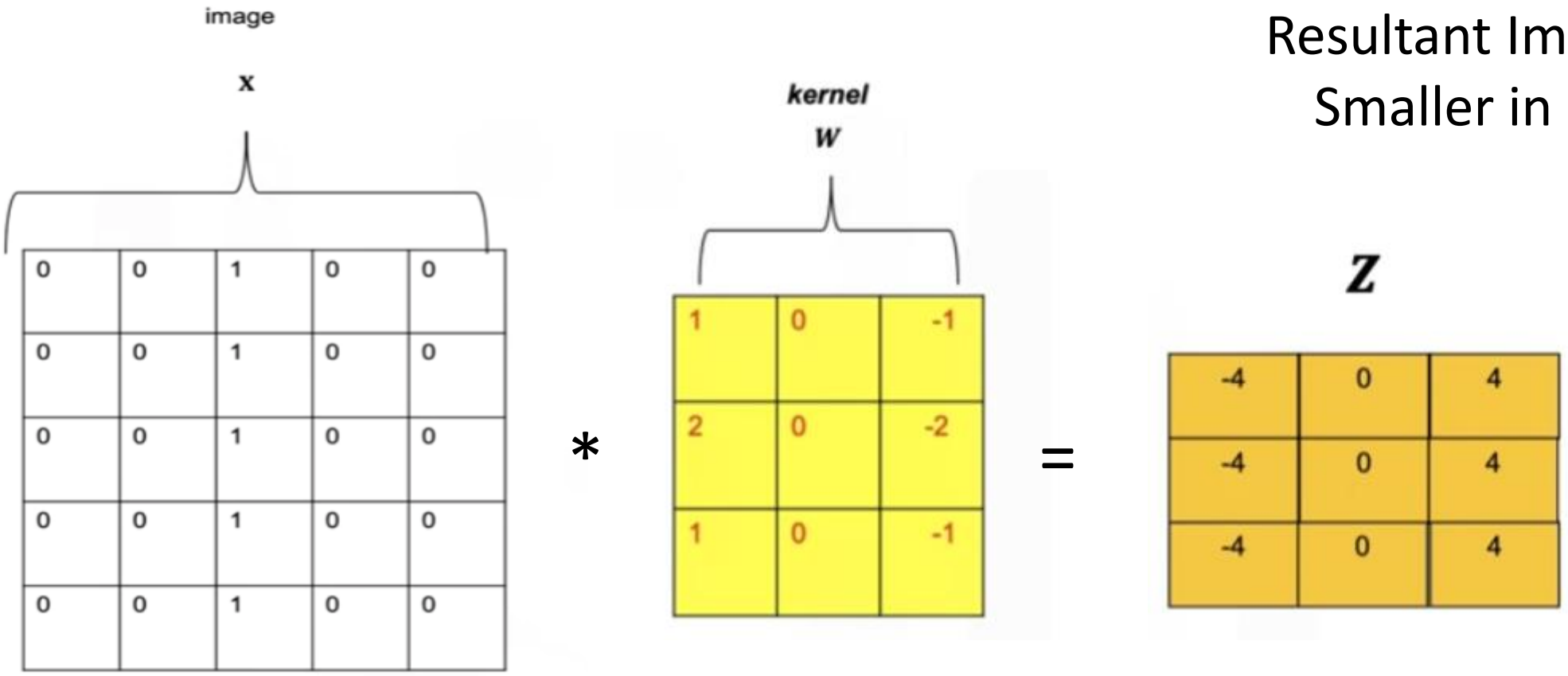
Convolution

Convolution is a linear filter that multiplies the corresponding values of the filter with the pixels in the image and sums the products.

$$Z = W * X$$

Convolution

$$Z = W * X$$



Resultant Image is Smaller in size

Different Types of Kernels/Filters

- Using the kernel operations different types of kernels can create different effect when applied on the image.
- In general, some of them provide smoothing/noise reduction and some of them are for edge detection/enhancement.
- They apply on the original image using convolution operation.

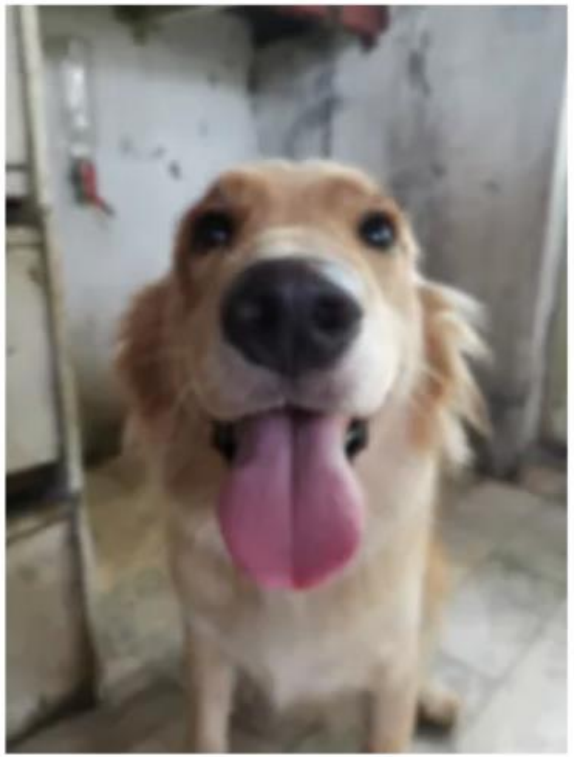
Kernel Type	Matrix Example	Application
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	Original image reproduction
Sharpening	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	Detail and edge enhancement

Different Types of Kernels/Filter

Kernel Type	Matrix Example	Application
Box Blur	$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$	Basic blur and noise reduction
Gaussian Blur	$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$	Smoother blur effect for noise reduction
Sobel (Edge)	$\begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}$	Edge detection (horizontal/vertical)
Laplacian (Edge)	$\begin{bmatrix}0 & -1 & 0\\-1 & 4 & -1\\0 & -1 & 0\end{bmatrix}$	Second derivative edge detection
Emboss	$\begin{bmatrix}-2 & -1 & 0\\-1 & 1 & 1\\0 & 1 & 2\end{bmatrix}$	3D or embossed appearance
Edge Enhancement	$\begin{bmatrix}0 & -1 & 0\\-1 & 5 & -1\\0 & -1 & 0\end{bmatrix}$	Intense edge detail highlighting

Different Types of Kernels/Filters

- Smoothing/Noise Reduction Filters are also called as Low-pass filter. They can be linear or non-linear.



- Sharpening/Edge Detection filters are also called as High-pass filter. They are usually linear.

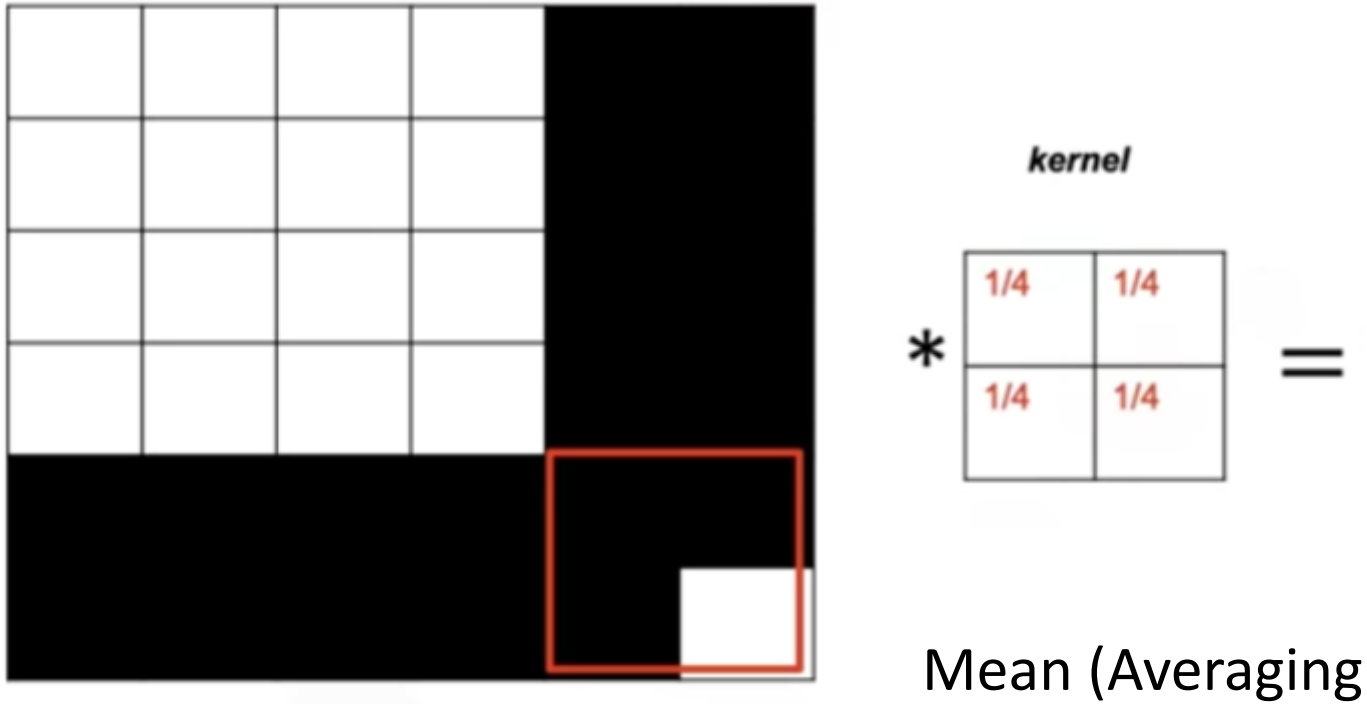
Different Types of Kernels/Filters

- **Low-Pass Linear Filter:** Blurring an image to reduce noise using a Gaussian filter.
- **High-Pass Linear Filter:** Detecting edges using a Laplacian filter.
- **Low-Pass Non-Linear Filter:** Removing salt-and-pepper noise with a median filter while keeping edges intact.

- For understanding the effect of different types of kernels/filters, follow this link: https://setosa.io/ev/image-kernels/
- For implementation of different types of kernels/filters in simple Pythons, follow this link: https://medium.com/swlh/image-processing-with-python-convolutional-filters-and-kernels-b9884d91a8fd

Low Pass Filters

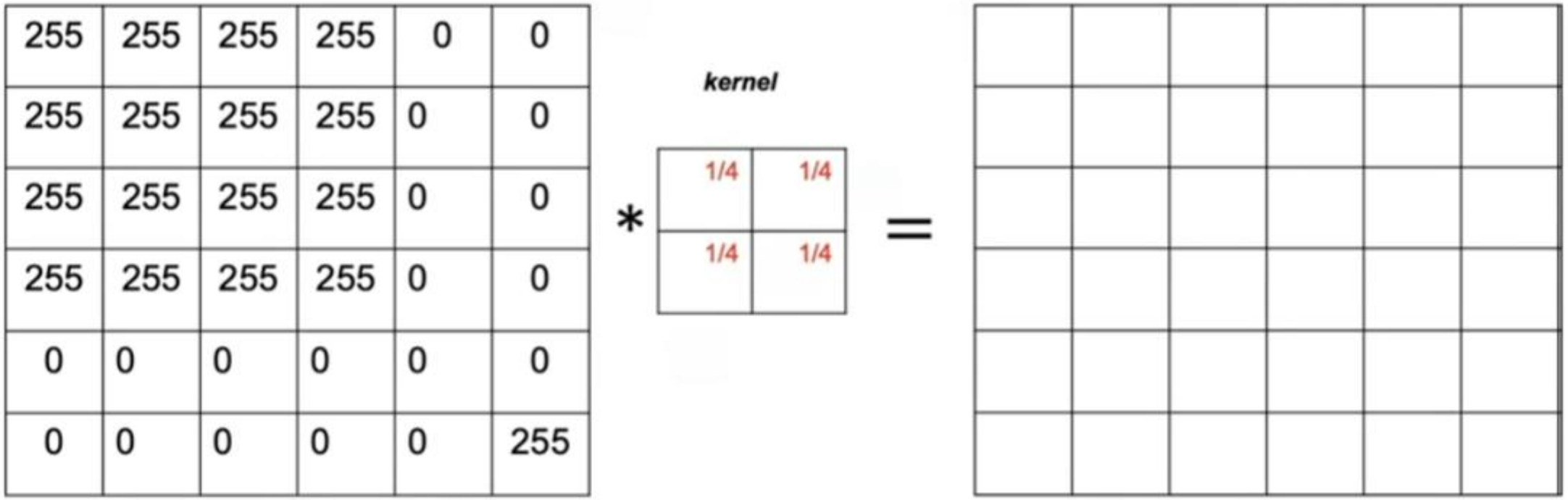
Low pass filters are used to smooth (blur) the image to remove noise.



Mean (Averaging) Filter

Low Pass Filters

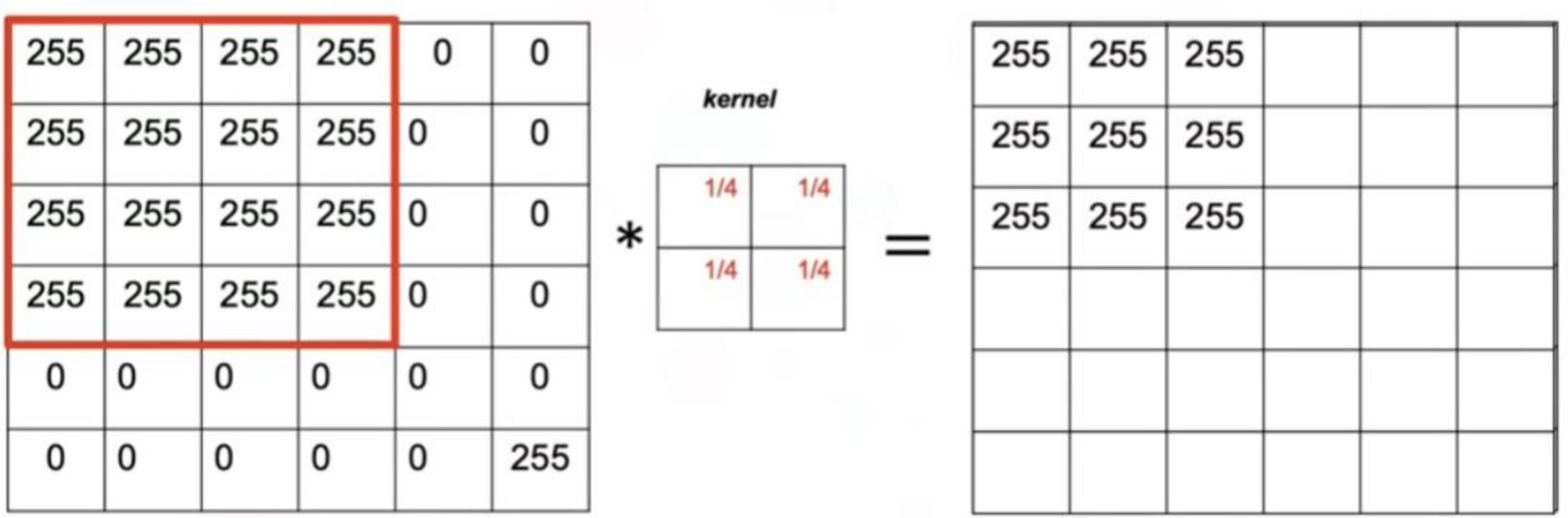
Low pass filters are used to smooth (blur) the image to remove noise.



Mean (Averaging) Filter

Low Pass Filters

Low pass filters are used to smooth (blur) the image to remove noise.



Mean (Averaging) Filter

Low Pass Filters

Low pass filters are used to smooth (blur) the image to remove noise.

255	255	255	255	0	0
255	255	255	255	0	0
255	255	255	255	0	0
255	255	255	255	0	0
0	0	0	0	0	0
0	0	0	0	0	255

$*$

kernel

1/4	1/4
1/4	1/4

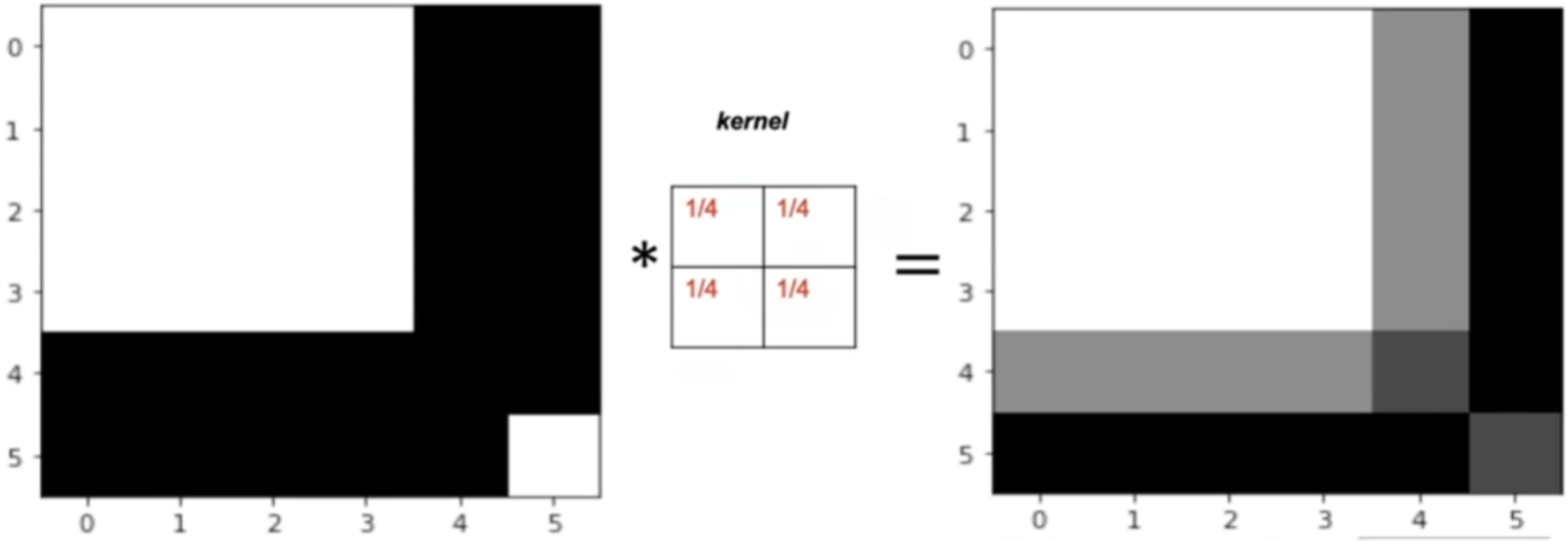
$=$

255	255	255	255	128	0
255	255	255	255	128	0
255	255	255	255	128	0
255	255	255	255	128	0
128	128	128	128	64	0
0	0	0	0	0	64

Mean (Averaging) Filter

Low Pass Filters

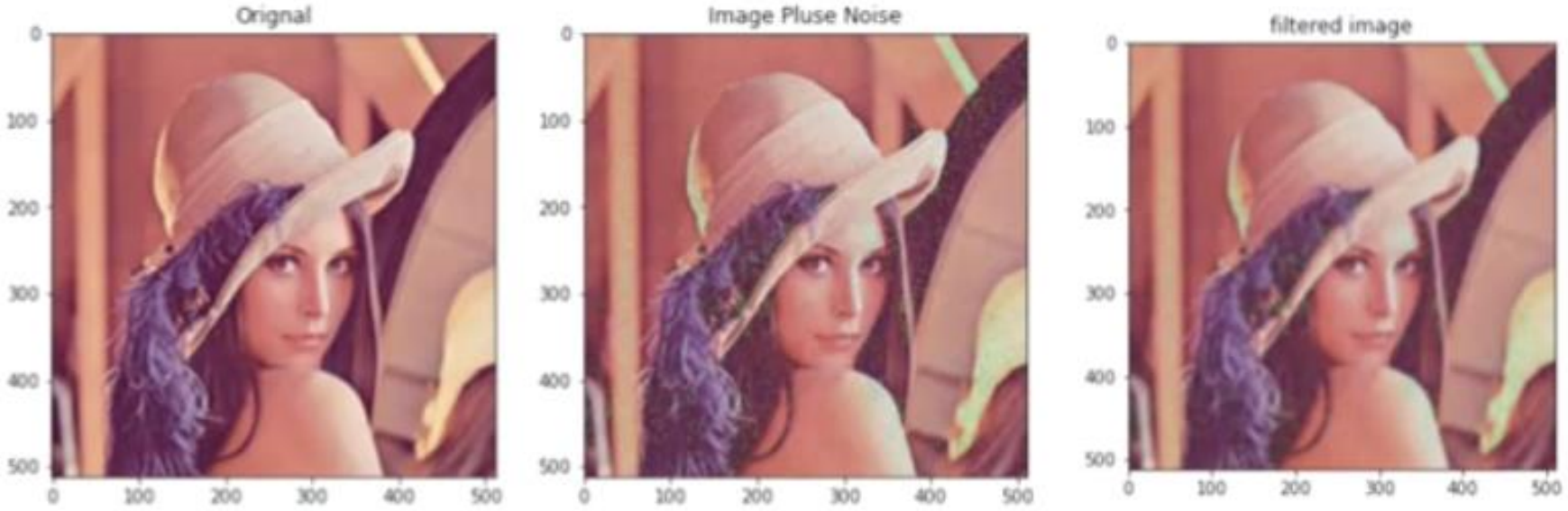
Low pass filters are used to smooth (blur) the image to remove noise.



Mean (Averaging) Filter

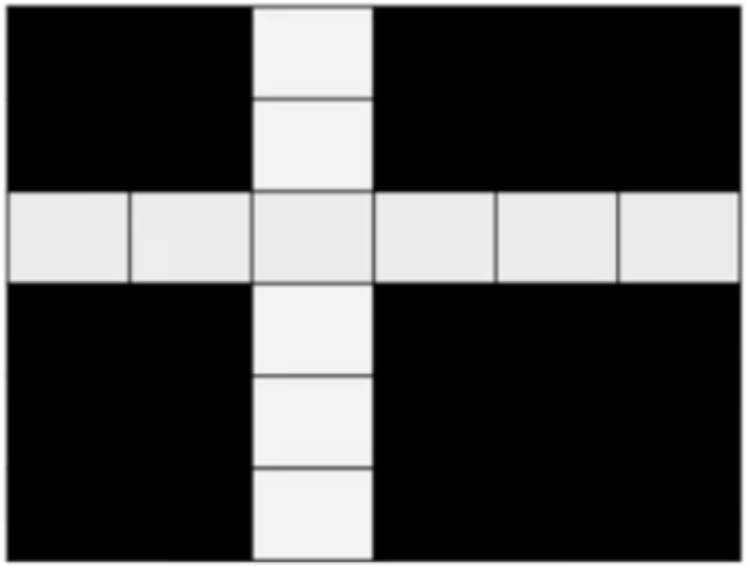
Implementation – Mean Filtering

```
new_image= image+Noise

kernel = np.ones((6,6))/36

image_filtered=cv2.filter2D(src=new_image ,ddepth=-1,kernel=kernel)
```



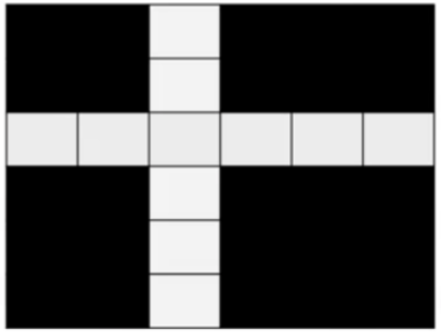
Edge Detection Filters

Edges are the points in an image where the intensity levels (image brightness) changes sharply. Edge detection uses methods to approximate derivatives and gradients to identify these points.

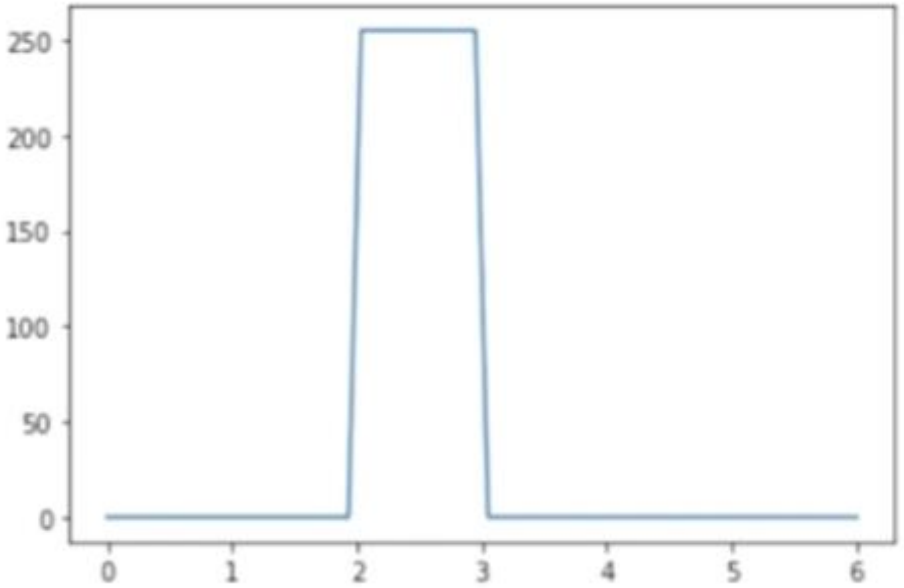


Edge Detection Filters

Edges are the points in an image where the intensity levels (image brightness) changes sharply. Edge detection uses methods to approximate derivatives and gradients to identify these points.



0	0	255	0	0	0
0	0	255	0	0	0
255	255	255	255	255	255
0	0	255	0	0	0
0	0	255	0	0	0
0	0	255	0	0	0



Edge Detection Filters

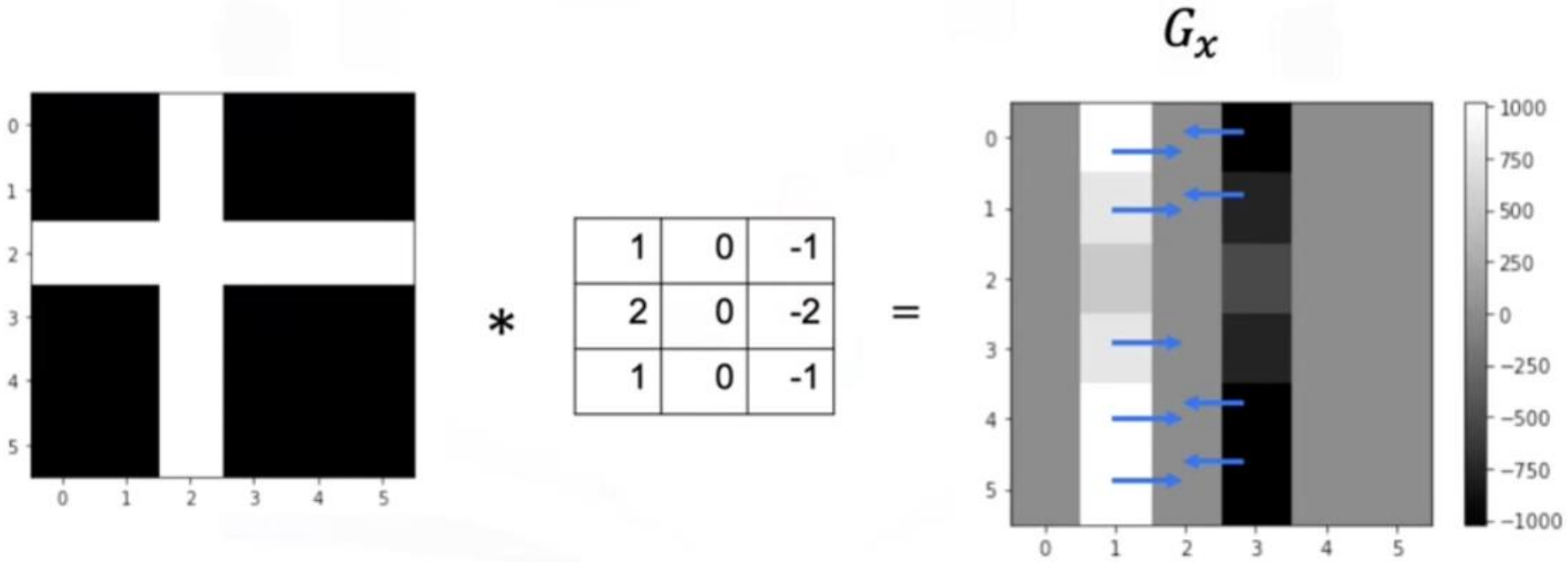
0	1	2	3	4	5
0	0	255	0	0	0
0	0	255	0	0	0
255	255	255	255	255	255
0	0	255	0	0	0
0	0	255	0	0	0
0	0	255	0	0	0

$$I[i,j+1] - I[i,j]$$

$$\frac{\partial I}{\partial x}$$

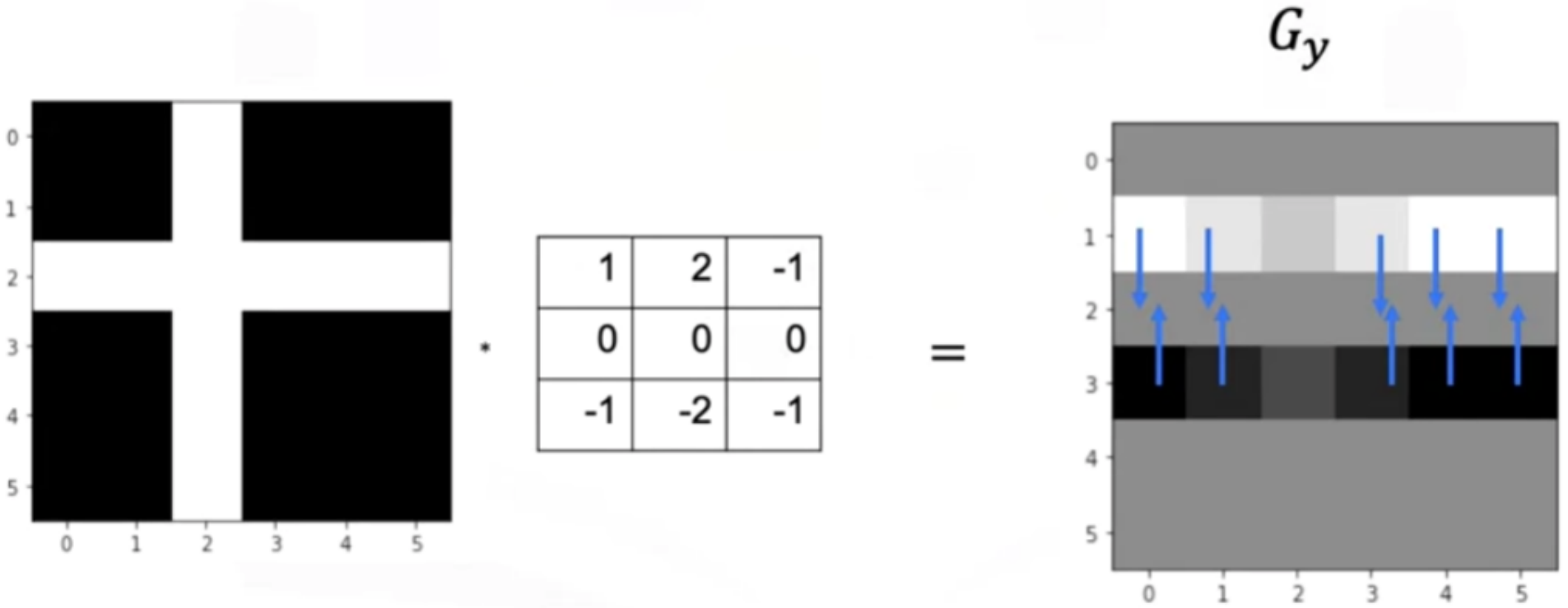
i	$I[i,2]$	$I[i,1]$	$I[i,j+1] - I[i,j]$
0	255	0	255
1	255	0	255
2	255	255	0
3	255	0	255
4	255	0	255
6	255	0	255

Edge Detection Filters



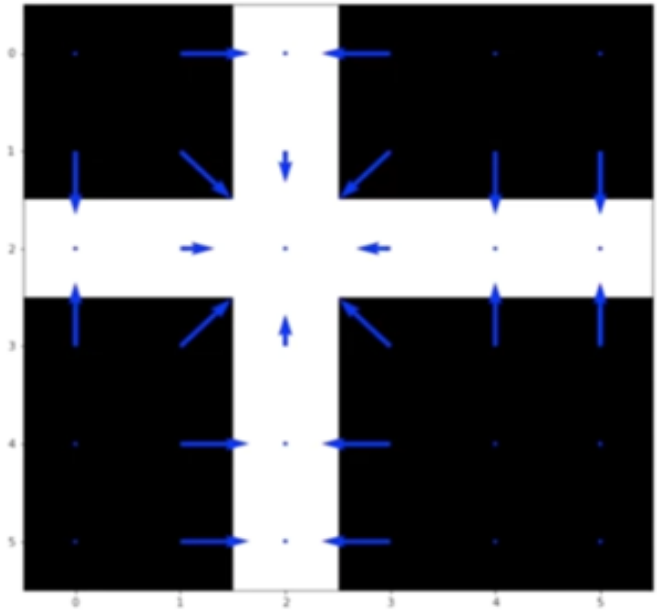
Horizontal Edge Detection

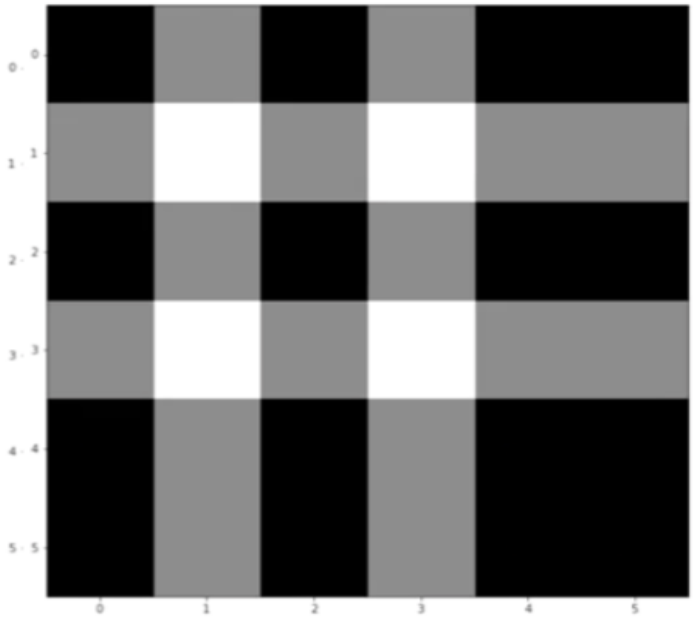
Edge Detection Filters



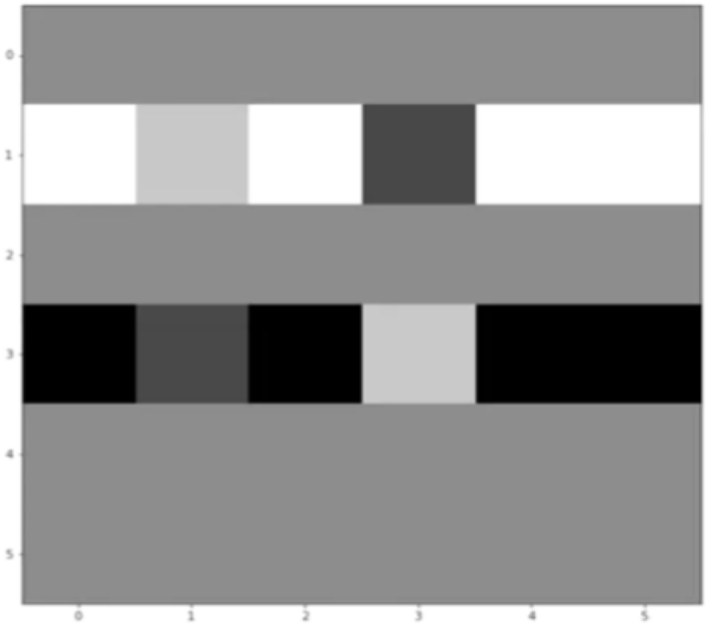
Vertical Edge Detection

Edge Detection Filters





$\|G\|$



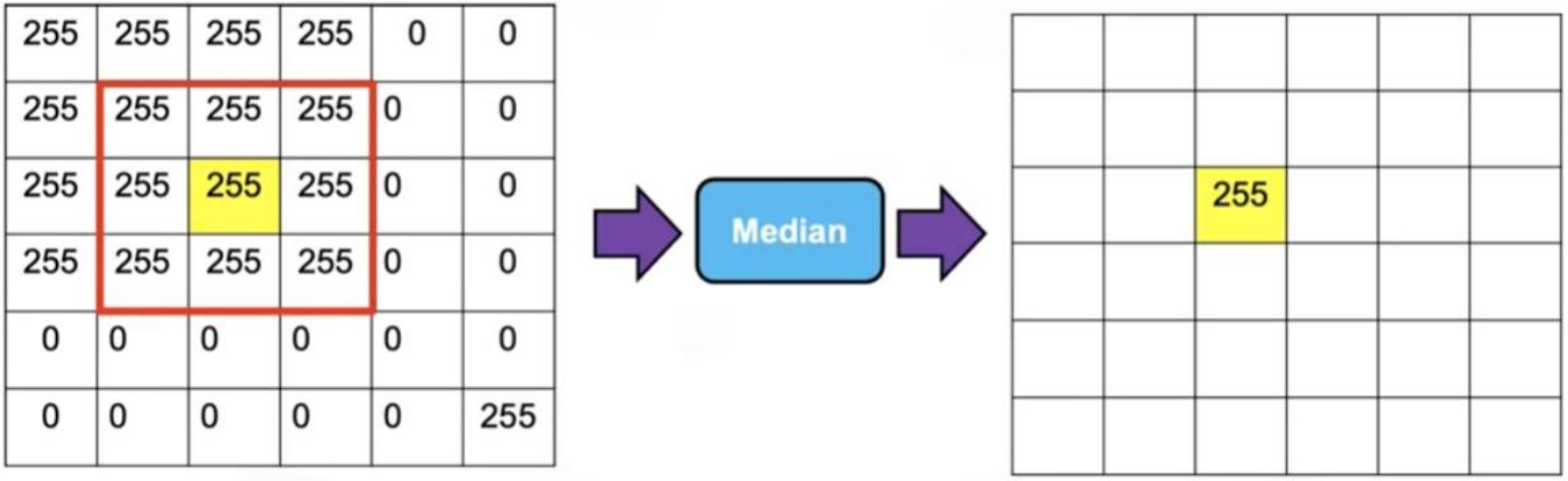
$$G = [G_x, G_y]$$

$$\|G\| = \sqrt{{G_x}^2 + {G_y}^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

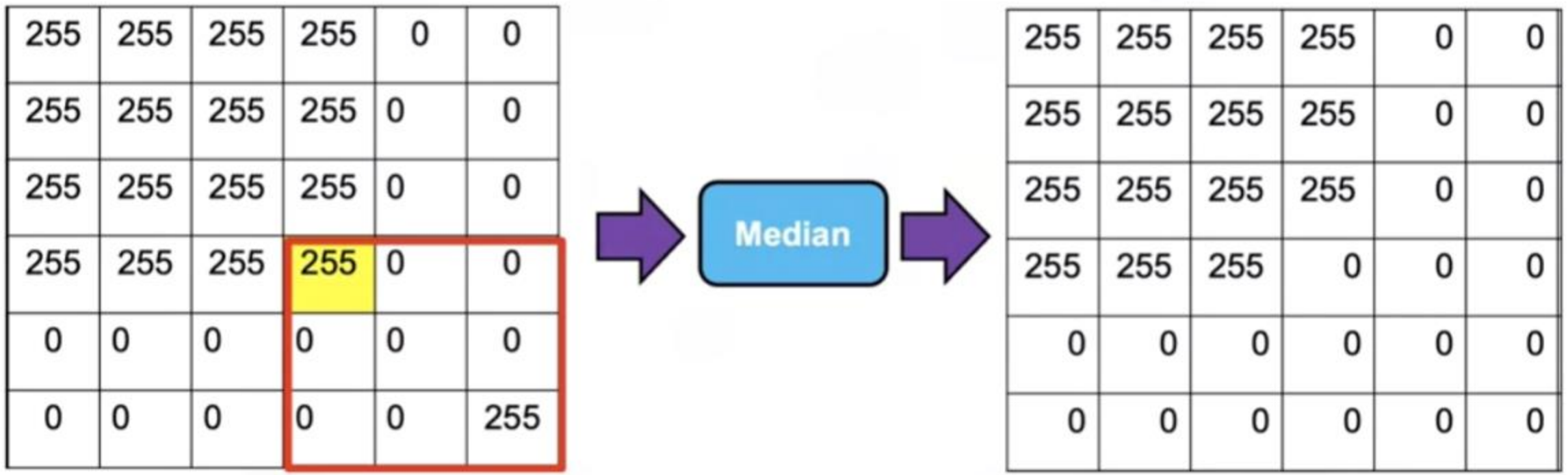
Median Filters – Non-Linear Filtering

Median filters are best for Impulse or Salt-and-Pepper noise.



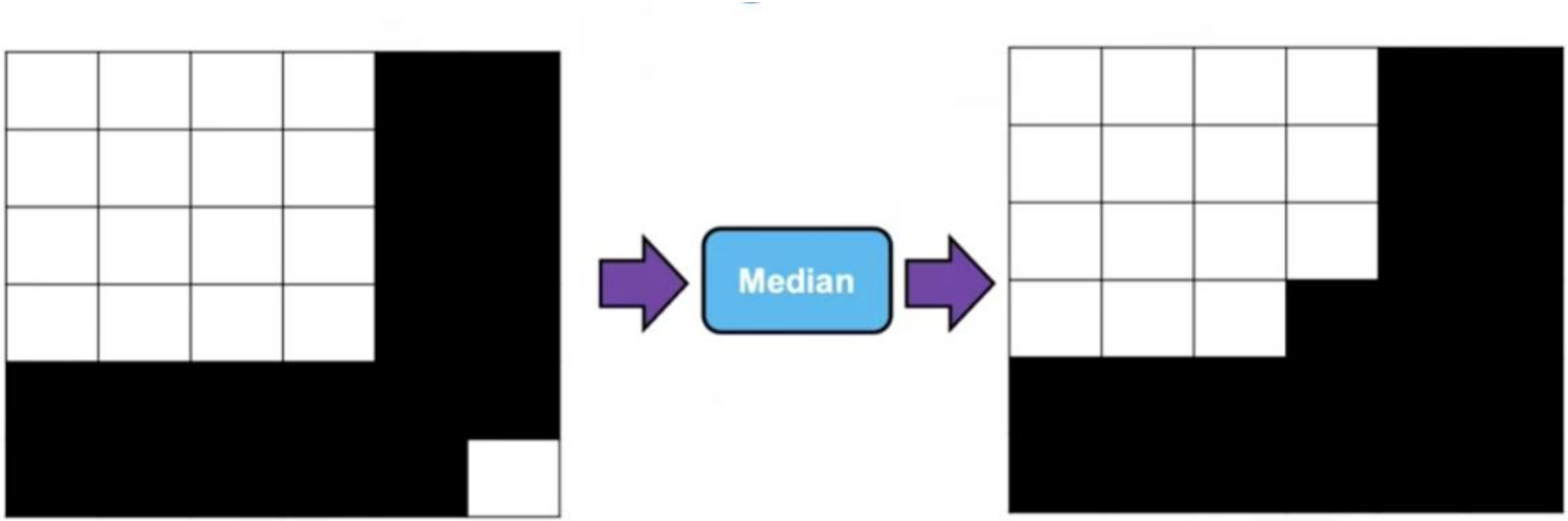
Median Filters – Non-Linear Filtering

Median filters are best for Impulse or Salt-and-Pepper noise.



Median Filters – Non-Linear Filtering

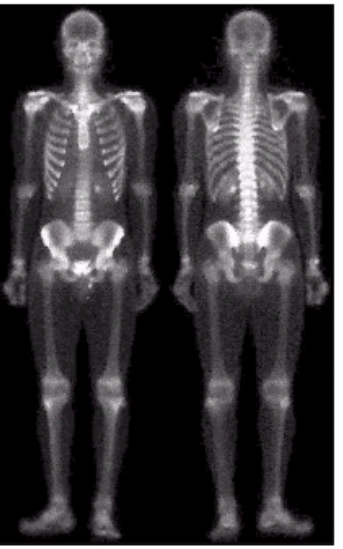
Median filters are best for Impulse or Salt-and-Pepper noise. However, these may distort the shape.



Using Multiple Filters

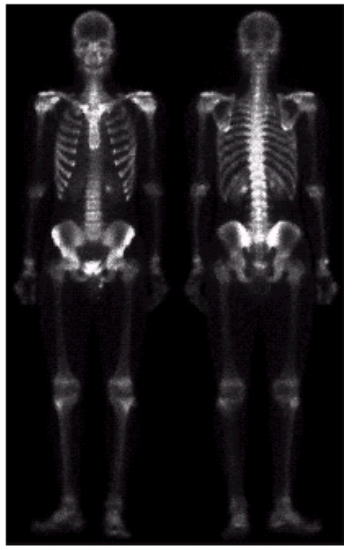
Result of applying a power-law trans. to (g)

(h)



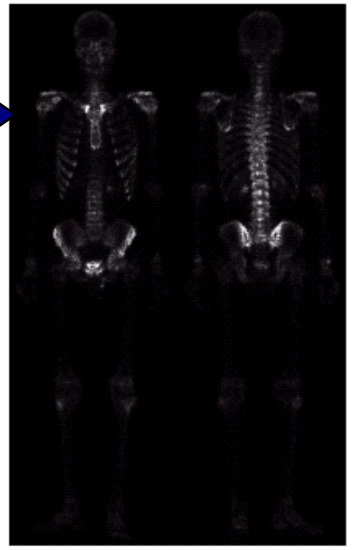
Sharpened image which is sum of (a) and (f)

(g)



The product of (c) and (e) which will be used as a mask

(f)



(e)

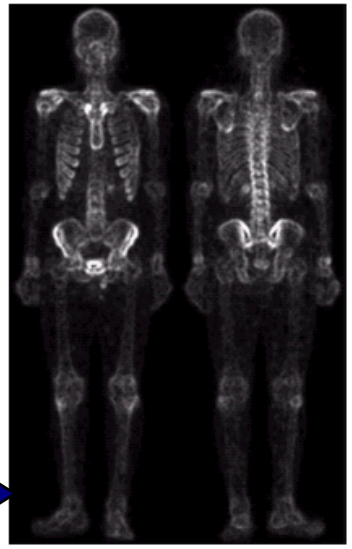


Image (d) smoothed with a 5*5 averaging filter

Combining Spatial Enhancement Methods

Acknowledgements

Some of the content of these slides is taken from:

- www.coursera.com
- Digital Image Processing by Gonzalez