

Image Processing & OpenCV

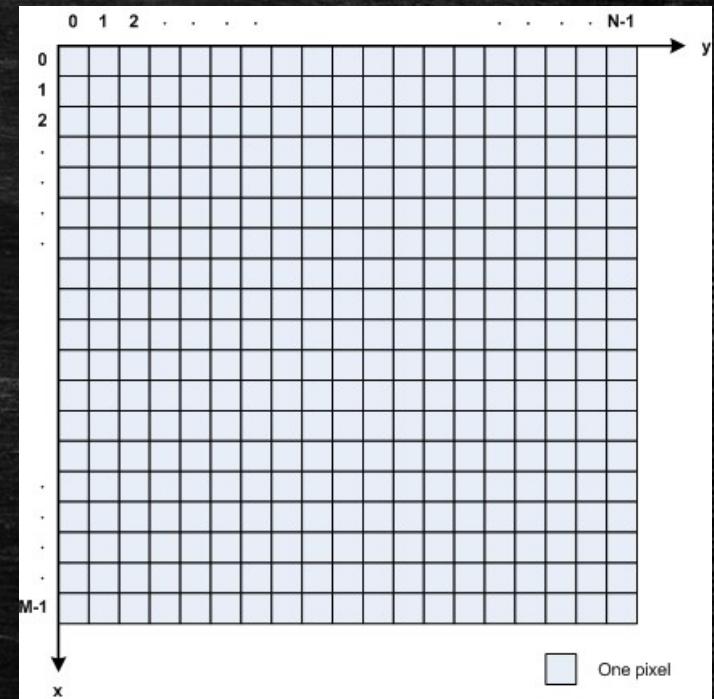
Jirasak Sittigorn

Department of Computer Engineering, School of Engineering
King Mongkut's Institute of Technology Ladkrabang

Digital Image

- ภาพดิจิตอลสามารถอธิบายได้ด้วยฟังก์ชันสองมิติ $f(x,y)$
 - x และ y เป็นตำแหน่งของระนาบ
 - f เป็นค่าที่หรือแอมพลิจูดที่ตำแหน่ง x และ y
- สามารถเขียนสมการให้อยู่ในรูปของ Matrix ได้ดังนี้

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M,0) & f(M,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

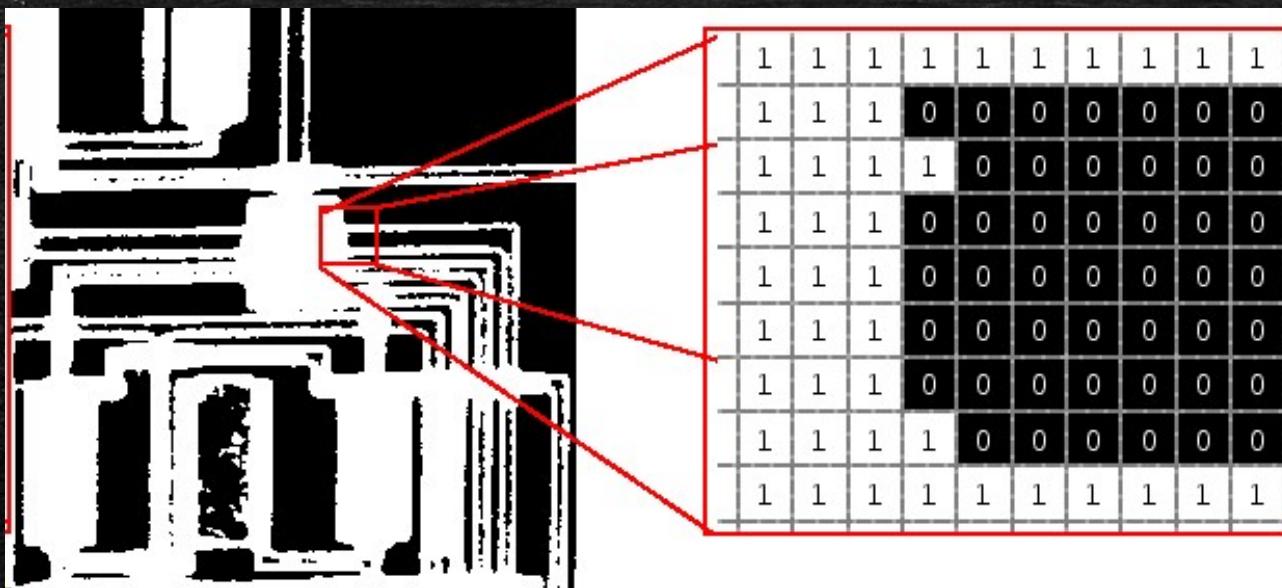


Digital Image

- Binary Image หรือ Black and White Image
- Intensity Image หรือ Monochrome Image หรือ Gray Image
- Indexed Image
- Color Image หรือ RGB Image

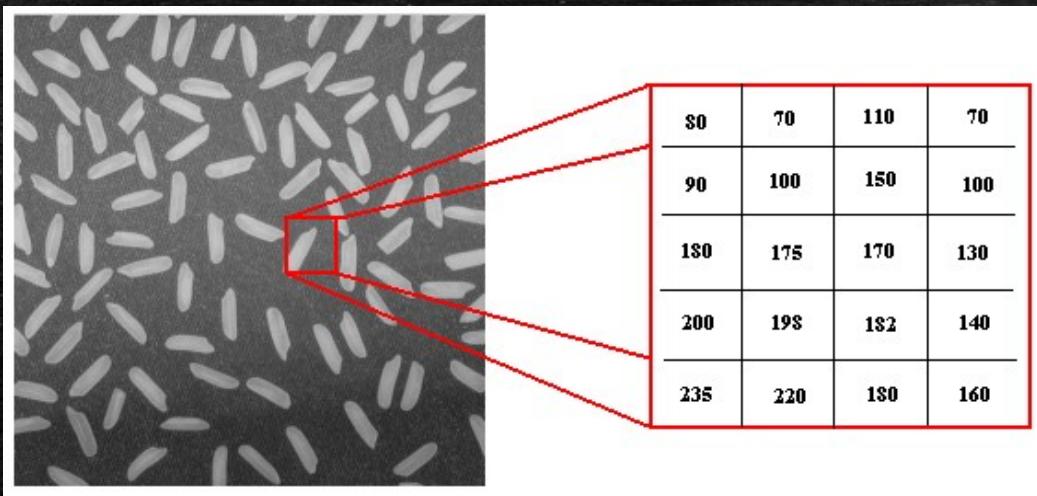
Binary Image หรือ Black and White Image

- เป็นภาพที่มีระดับความแตกต่างของสีในภาพเพียง 2 ระดับ คือสีขาว (แทนค่าด้วย 1) และสีดำ (แทนค่าด้วย 0) เท่านั้น



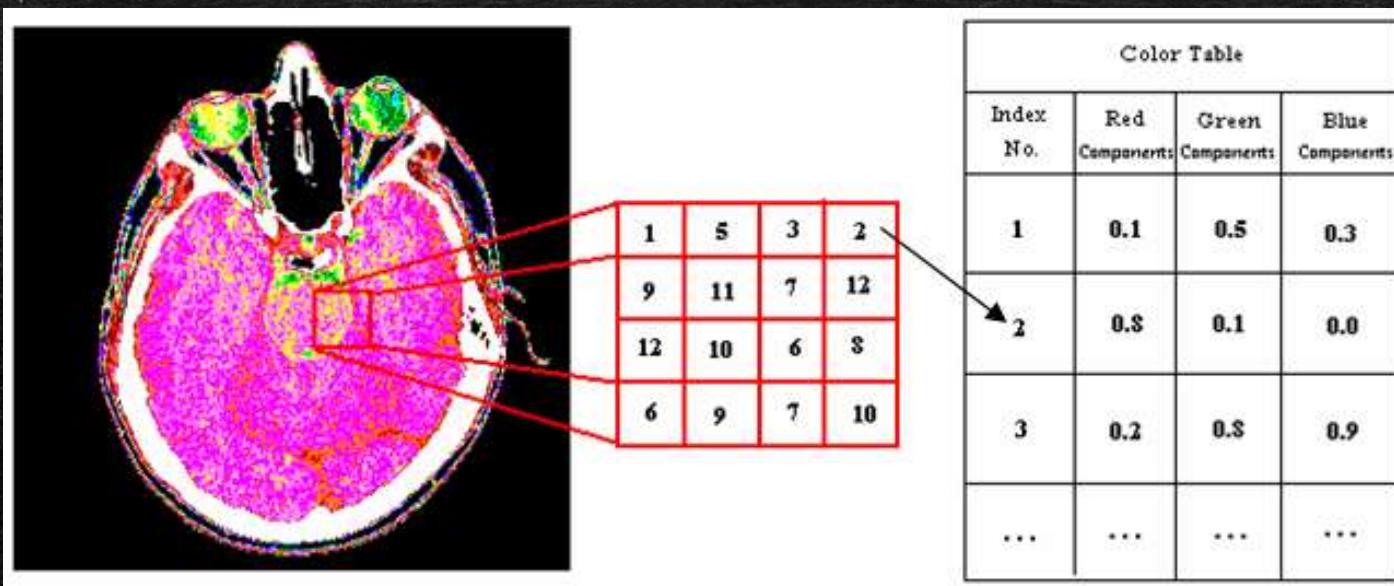
Intensity Image หรือ Monochrome Image หรือ Gray Image

- เป็นภาพที่มีระดับสีของภาพตามความเข้มแสงที่เข้ามา โดยภาพจะมีลักษณะเป็นภาพโทนสีเทา โดยระดับความเข้มแสง ขึ้นอยู่กับจำนวนบิตข้อมูลที่ใช้แทนระดับความเข้มแสง



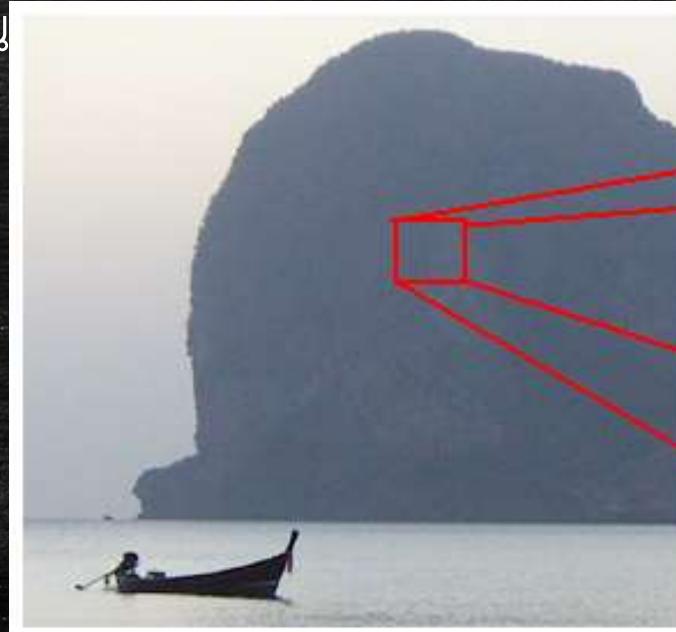
Indexed Image

- เป็นภาพที่แต่ละพิกเซลจะเก็บค่า Index Number ซึ่งเป็นค่าที่มีความสัมพันธ์กับสีในตารางที่ใช้เปรียบเทียบ



Color Image หรือ RGB Image

- เป็นภาพที่แต่ละพิกเซลจะประกอบด้วยค่า 3 ค่า คือ ค่าของสีแดง สีเขียว และสีน้ำเงิน



			10	22	24	25
	80	75	72	70	20	
42	20	40	50	65	15	
35	20	37	44	60	14	
25	10	30	28	55		
10	15	17	25			

Color Image (RGB Color Model)

- ในกล้องถ่ายภาพจะมี Sensor สำหรับแสงสี 3 สี คือ สีแดง สีเขียว และสีน้ำเงิน

- Red Filter



- Green Filter



- Blue Filter

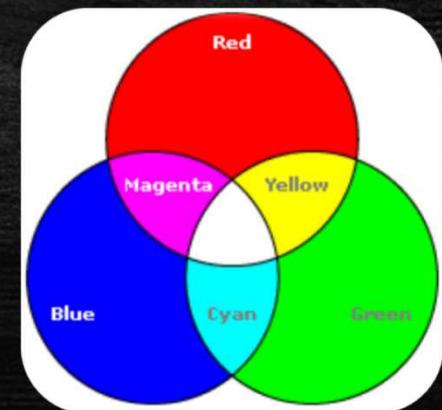


Image Structure

- ในภาพ 1 ภาพประกอบด้วย จุดภาพ (Pixel) หลายจุด
ภาพประกอบกันเพื่อแสดงรายละเอียดของภาพ และในแต่ละ
จุดภาพจะมีค่าที่แสดงสีในจุดภาพ
 - Resolution
 - Image Intensity Level & Image Band

Resolution

- หมายถึง Image Dimension หรือ ความละเอียดของภาพ
สามารถหมายถึงขนาดของภาพได้ ในการใช้งานมักบอกเป็น
จำนวน Dot per Inch (DPI)
 - Real world mapping (Scanning and Printing)
 - Image Resolution from sensor input

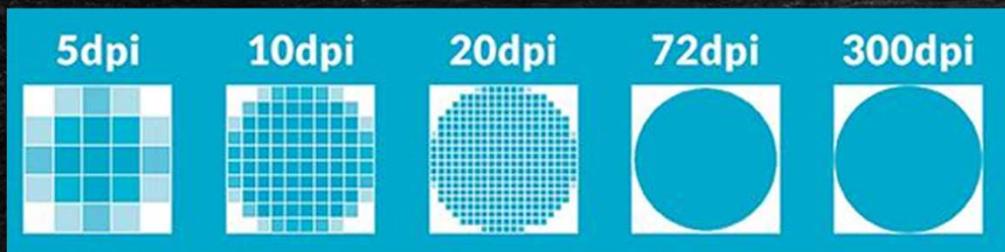


Image Intensity Level & Image Band

- Image Intensity Level เป็นค่าที่แสดงระดับสีของแต่ละจุดภาพ หากจำนวนสีที่สามารถแสดงแต่ละจุดภาพมีจำนวนมากทำให้ภาพ มีหลากหลายระดับสี แต่จะทำให้ขนาดของข้อมูลแต่ละจุดภาพ เพิ่มขึ้น เรียกว่า Bit Depth มีหน่วยเป็น bits/pixel เช่น
 - Binary Image 1 bits/pixel
 - Grayscale image 8 bits/pixel
- Image Band เป็นจำนวนแม่สีที่อยู่ในภาพ

Image Data Types

- Uint8 (unsigned integer 8 bits)
 - Values: 0 & 1 (0 & 255) for Binary Image
 - Values: 0-255 for Grayscale image Color Image
 - No negative / No number greater than 255
 - Truncate outside its range
 - Cannot perform mathematical operation
 - In some developing tools
- Double
 - Safe for math operation
 - May need to rescale back to uint8

Binary Image

Binary Image



```
import numpy as np
from matplotlib import pyplot as plt
N = 16
if True :
    img_bw = np.zeros( (N,N) )
    img_bw[0:int(N/2-1),:] = 1
else :
    # Grayscale image only values 0 and 255
    img_bw = np.zeros( (N,N), dtype=np.uint8)
    img_bw[0:int(N/2-1),:] = 255
plt.imshow(img_bw, cmap=plt.cm.gray)
plt.show()
```

Grayscale image

```
import numpy as np
from matplotlib import pyplot as plt
fig=plt.figure(figsize=(8, 4))
N = 16

raw = np.random.randint(10, size=(N,N))
img_bw = (raw >= 5) * 1
img_gray = np.random.randint(255,
size=(N,N))

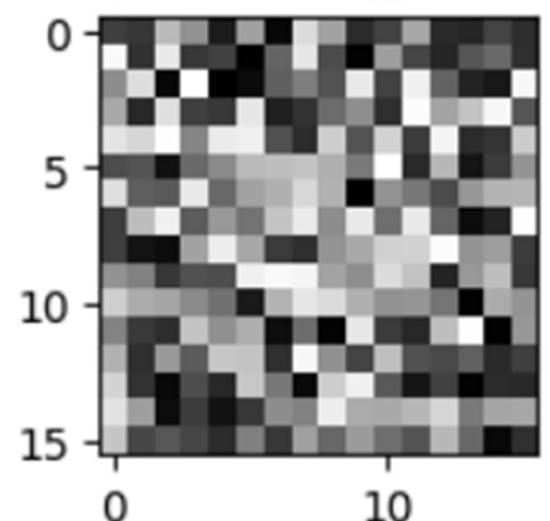
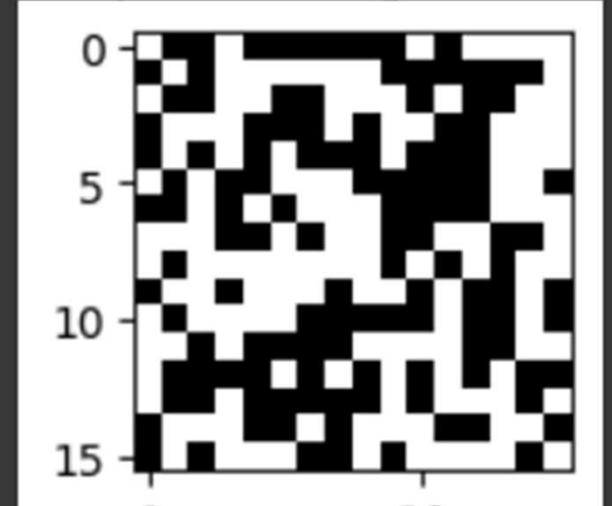
#fig.add_subplot(1, 2, 1)
fig.add_subplot(2, 1, 1)
plt.imshow(img_bw, cmap=plt.cm.gray)

#fig.add_subplot(1, 2, 2)
fig.add_subplot(2, 1, 2)
plt.imshow(img_gray, cmap=plt.cm.gray)
```

NextGen AI Camp, Department of Computer Engineering, KMITL



<matplotlib.image.AxesImage>



Grayscale image

```
import numpy as np
import matplotlib.pyplot as plt    #from matplotlib import pyplot as plt

!wget "https://www.ce.kmitl.ac.th/api/faculty/download/Jirasak"

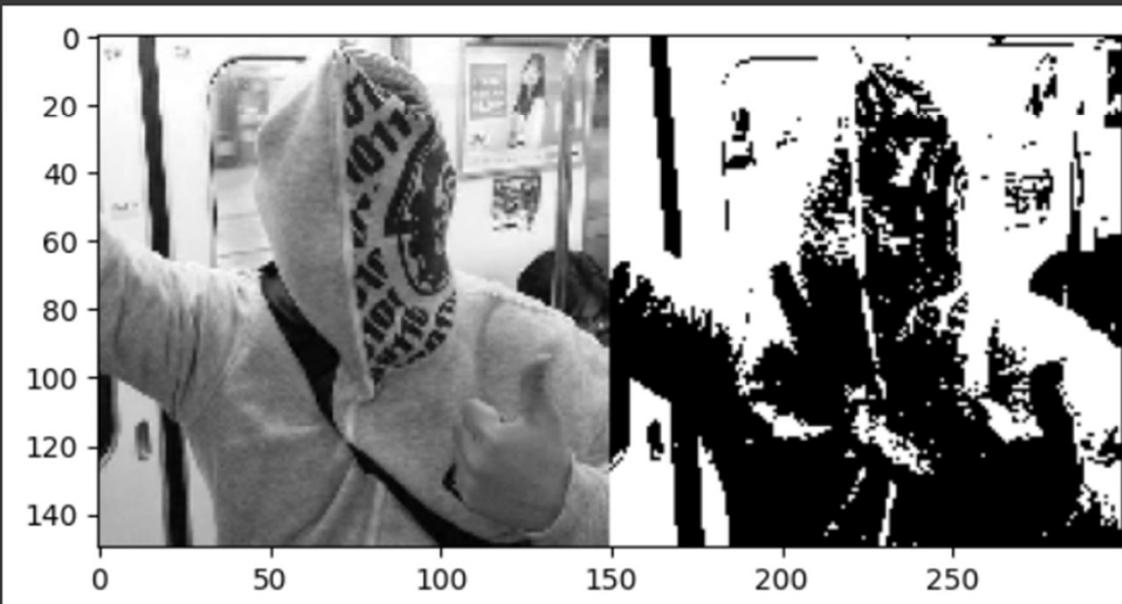
img_gray = plt.imread('Jirasak')[0:150,0:150,1]      #img.shape : Green Band
img_bw = (img_gray >= 127) * 1
img_bw = img_bw * 255

# Merge Grayscale Image & Binary Image only [0, 255]
both = np.hstack((img_gray, img_bw))
plt.imshow(both, cmap='gray')
plt.show()
```

```
→ --2024-07-16 05:29:04-- https://www.ce.kmitl.ac.th/api/faculty/download/Jirasak
Resolving www.ce.kmitl.ac.th (www.ce.kmitl.ac.th)... 161.246.127.223
Connecting to www.ce.kmitl.ac.th (www.ce.kmitl.ac.th)|161.246.127.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26663 (26K) [image/jpeg]
Saving to: 'Jirasak'

Jirasak          100%[=====] 26.04K  124KB/s   in 0.2s

2024-07-16 05:29:06 (124 KB/s) - 'Jirasak' saved [26663/26663]
```



Color Image

```
import numpy as np
import matplotlib.pyplot as plt
#from matplotlib import pyplot as plt

!wget "https://www.ce.kmitl.ac.th/
api/faculty/download/Jirasak"

img_rgb = plt.imread('Jirasak')
plt.imshow(img_rgb)
plt.show()
```

```
→ --2024-07-19 03:46:35-- https://www.ce.kmitl.ac.th/api/faculty/download/Jirasak
Resolving www.ce.kmitl.ac.th (www.ce.kmitl.ac.th)... 161.246.127.223
Connecting to www.ce.kmitl.ac.th (www.ce.kmitl.ac.th)|161.246.127.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26663 (26K) [image/jpeg]
Saving to: 'Jirasak'

Jirasak          100%[=====] 26.04K   122KB/s   in 0.2s
```

2024-07-19 03:46:36 (122 KB/s) - 'Jirasak' saved [26663/26663]



Color Image

- Display Images with
axis off

```
fig, ax = plt.subplots(figsize=(5, 5))
ax.imshow(img_rgb)
ax.axis('off')
plt.show()
```



Color Image

- Display RGB Channels of image

```
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(img_rgb[:, :, 0], cmap='Reds')
axs[1].imshow(img_rgb[:, :, 1], cmap='Greens')
axs[2].imshow(img_rgb[:, :, 2], cmap='Blues')
axs[0].axis('off')
axs[1].axis('off')
axs[2].axis('off')
axs[0].set_title('Red channel')
axs[1].set_title('Green channel')
axs[2].set_title('Blue channel')
plt.show()
```



Image Array Structure and Reshape

- Color Image
 - Ex. RGB Image
 - Resolution 3×4 pixel
 - 3 row
 - 4 column
 - Image Intensity Level : 8 bits/pixel
 - Image Band : Red Green Blue

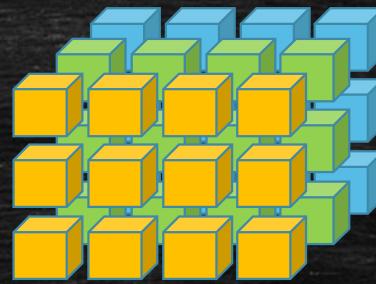
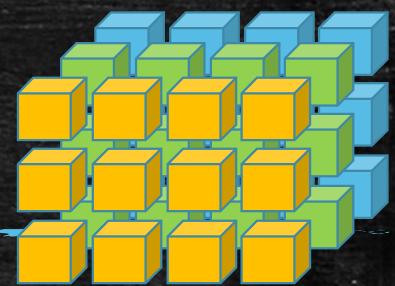


Image Array Structure and Reshape



- Image shape
 - Read from OpenCV
 - Normally color channels of each pixel is in the last dimension
 - Ex
 - Image shape: (1, 3, 2)
 - 1 = No. of Height (rows)
 - 3 = No. of Width (columns)
 - 2 = No of color channels

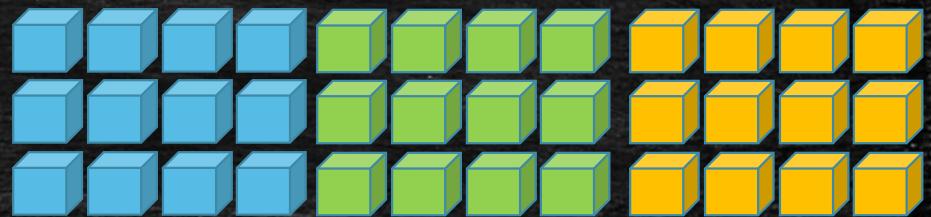


Image Array Structure and Reshape

- matplotlib vs Open CV
 - Display default : R, G, B
 - Channel 0 : R
 - Channel 1 : G
 - Channel 2 : B
 - CV default : B, G, R
 - Channel 0 : B
 - Channel 1 : G
 - Channel 2 : R

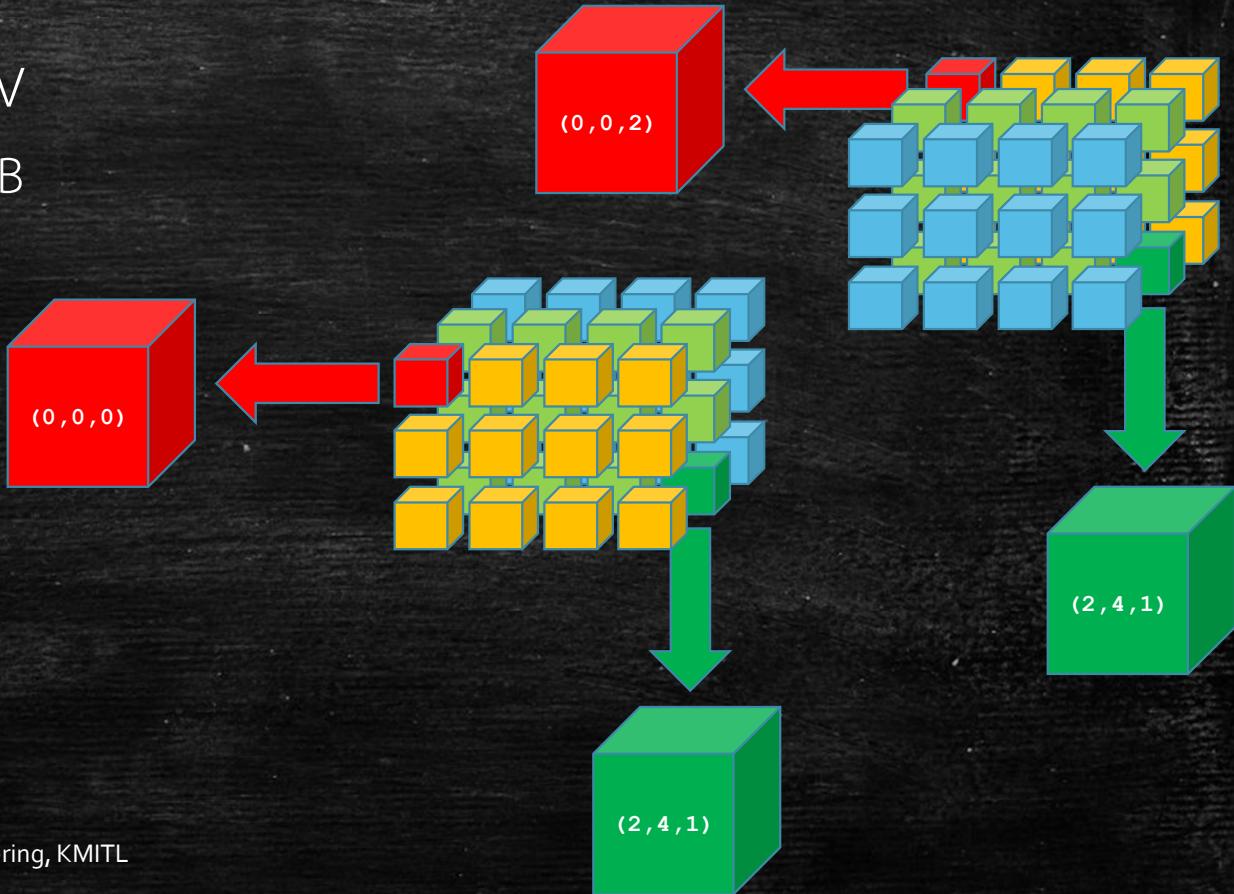
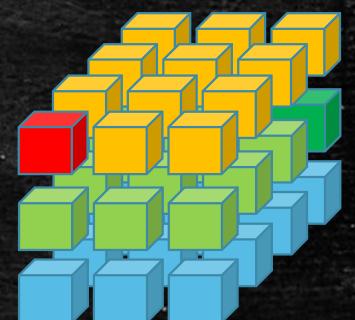
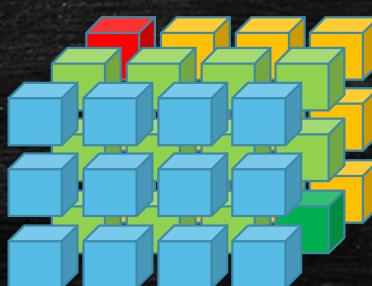


Image Array Structure and Reshape

- Image shape
 - OpenCV vs Pytorch
 - Normally color channels of each pixel is in the last dimension
- Ex
 - Image shape:
 - OpenCV -> (H, W, C)
 - Pytorch -> (C, H, W)

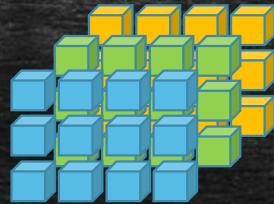
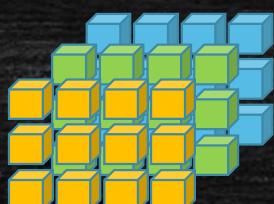


matplotlib vs Open CV

```
import numpy as np
import pandas as pd

import cv2
import matplotlib.pyplot as plt

!wget "https://www.ce.kmitl.ac.th/api/faculty/download/Jirasak"

img_mpl = plt.imread('Jirasak')
img_cv2 = cv2.imread('Jirasak')


#img_mpl.shape, img_cv2.shape

fig, axs = plt.subplots(2, 1, figsize=(5, 10))
axs[0].imshow(img_cv2)
axs[1].imshow(img_mpl)
axs[0].axis('off')
axs[1].axis('off')
axs[0].set_title('CV2 Image')
axs[1].set_title('Matplotlib Image')
plt.show()
```



Read and Write images

- Load the Dependencies
- Reading an Image
 - Reading an Image form Google Drive
 - Reading an Image form website URL
- Image Contours and Histograms
- Writing an image

Load the Dependencies

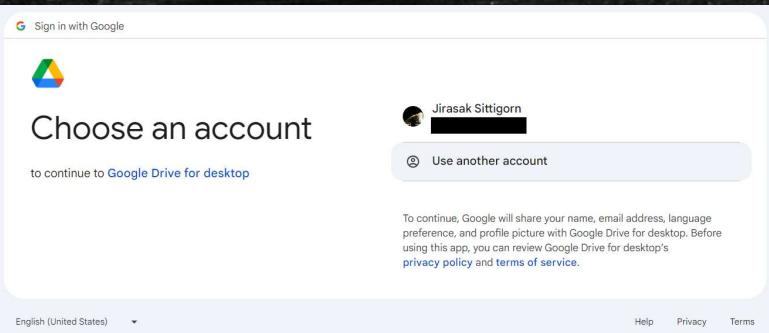
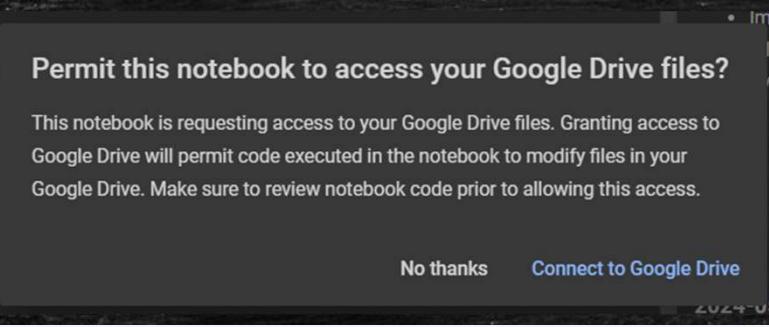
- ก่อนทำการจัดการกับภาพจำเป็นต้องเรียกใช้ Libraries ที่จำเป็นลงใน notebook เช่น numpy, pandas, cv2, skimage, PIL, matplotlib
 - Numpy is an array manipulation library, used for linear algebra, Fourier transform, and random number capabilities.
 - Pandas is a library for data manipulation and data analysis.
 - CV2 is a library for computer vision tasks.
 - Skimage is a library which supports image processing applications on python.
 - Matplotlib is a library which generates figures and provides graphical user interface toolkit.
- Reference : Introduction to Image Processing in Python

Load the Dependencies

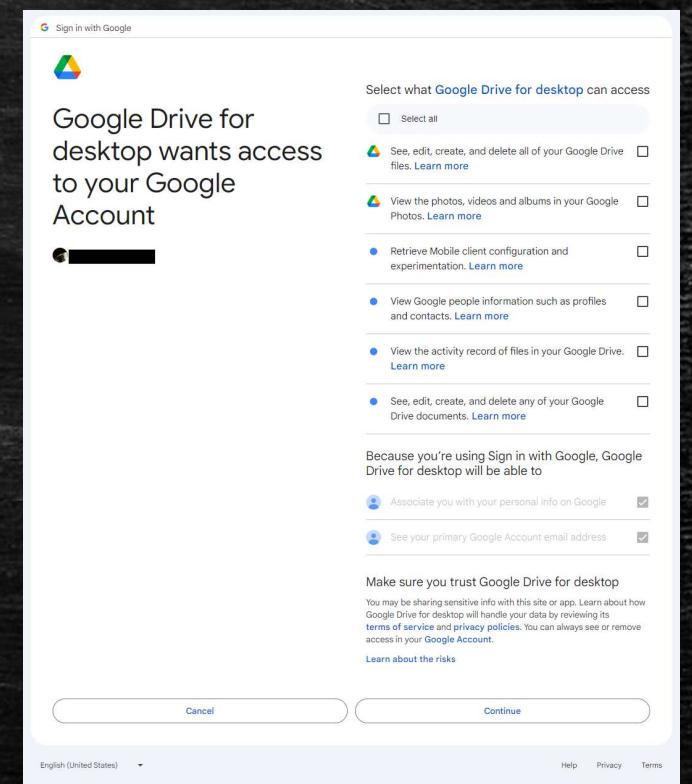
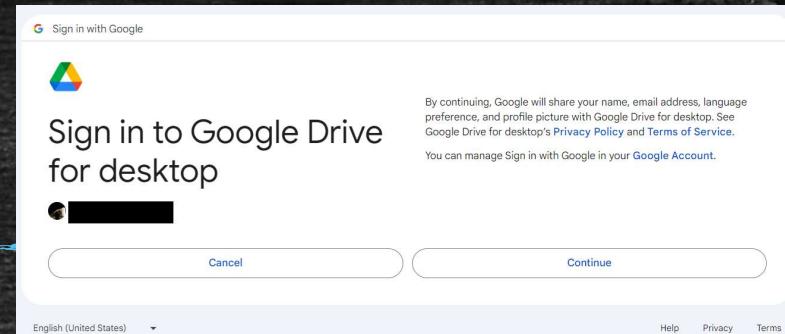
```
import numpy as np
import pandas as pd
import cv2 as cv
from google.colab.patches import cv2_imshow
# for image display
from skimage import io
from PIL import Image
import matplotlib.pyplot as plt
```

Reading an Image (Google Drive)

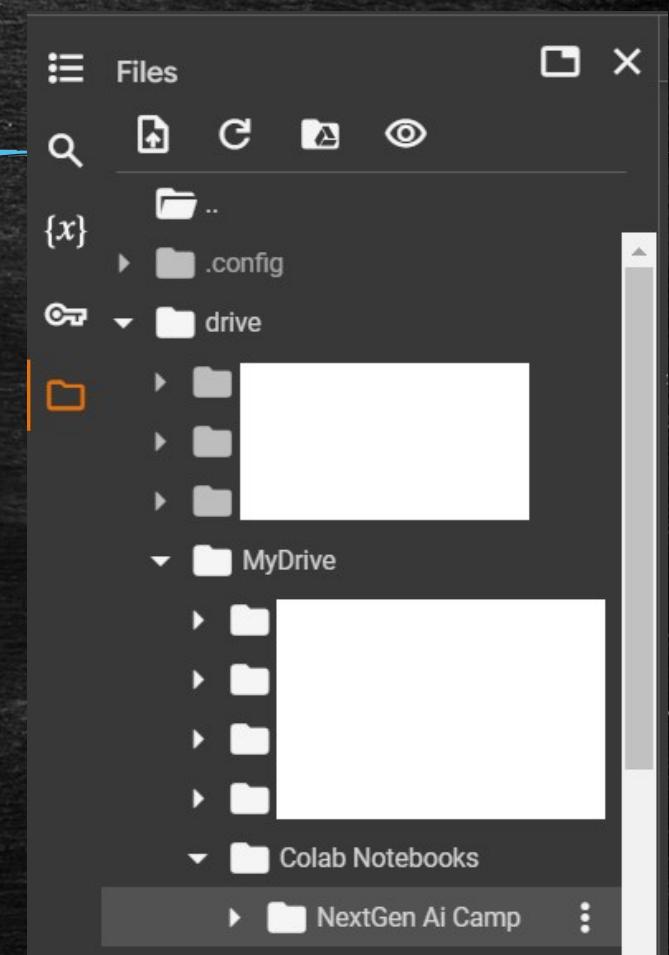
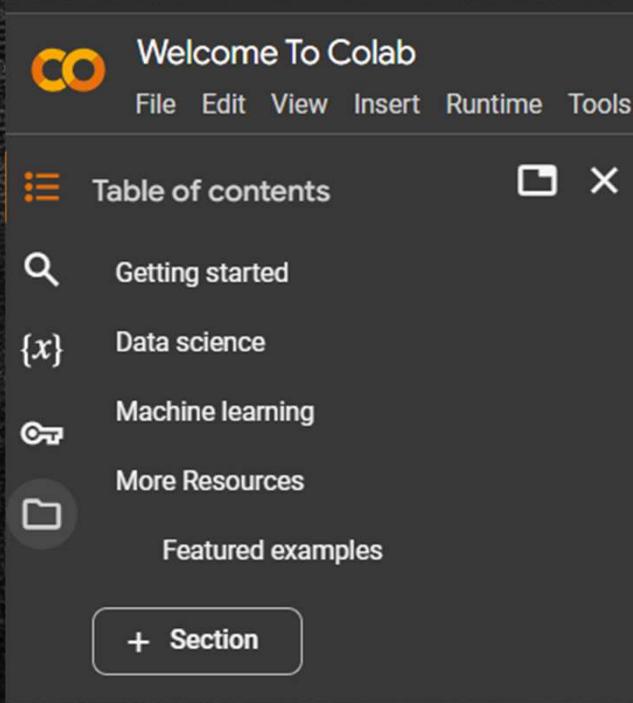
```
from google.colab import drive  
drive.mount('/content/drive')
```



NextGen AI Camp, Department of Computer Engineering, KMITL



Reading an Image (Google Drive)

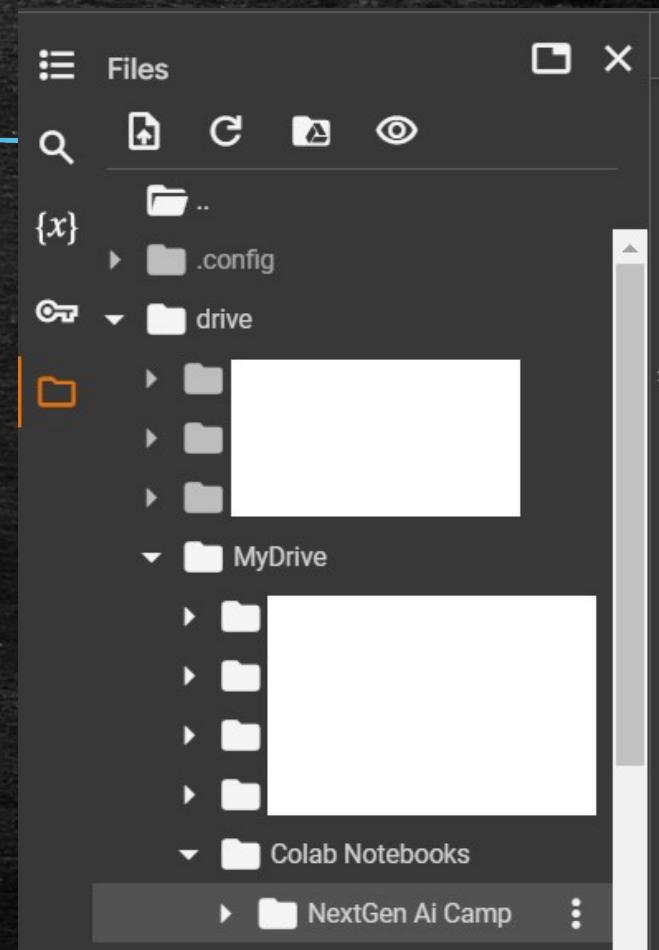


Reading an Image (Google Drive)

```
import numpy as np
import pandas as pd
import cv2 as cv
from google.colab.patches import cv2_imshow
from skimage import io
from PIL import Image
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

drive = '/content/drive/MyDrive/Colab Notebooks/
NextGen Ai Camp/Goldfish jumping.jpg'
img_drive = io.imread(drive)
cv2_imshow(img_drive)
```



Reading an Image (URL)

- ! wget "URL"
 - อ่านได้หลากหลายกว่า io.imread (ไม่ต้องลงถึงชื่อไฟล์)
 - ต้องดูผลชื่อที่ Download มา ก่อน
- img = io.imread("URL.type")
 - url ต้อง access ถึงไฟล์
 - สามารถใช้ตัวแปรเก็บค่าได้

Reading an Image (URL)

```
url = "https://t4.ftcdn.net/jpg/00/02/65/21/  
360_F_2652182_kzshWTrp8SC1KwkMbLnt0NCVtD1XUF.jpg"  
img_url = io.imread(url)  
cv2.imshow(img_url)
```



Reading an Image

```
image_RGB = cv.cvtColor(img_url, cv.COLOR_BGR2RGB)  
cv2.imshow(image_RGB)
```

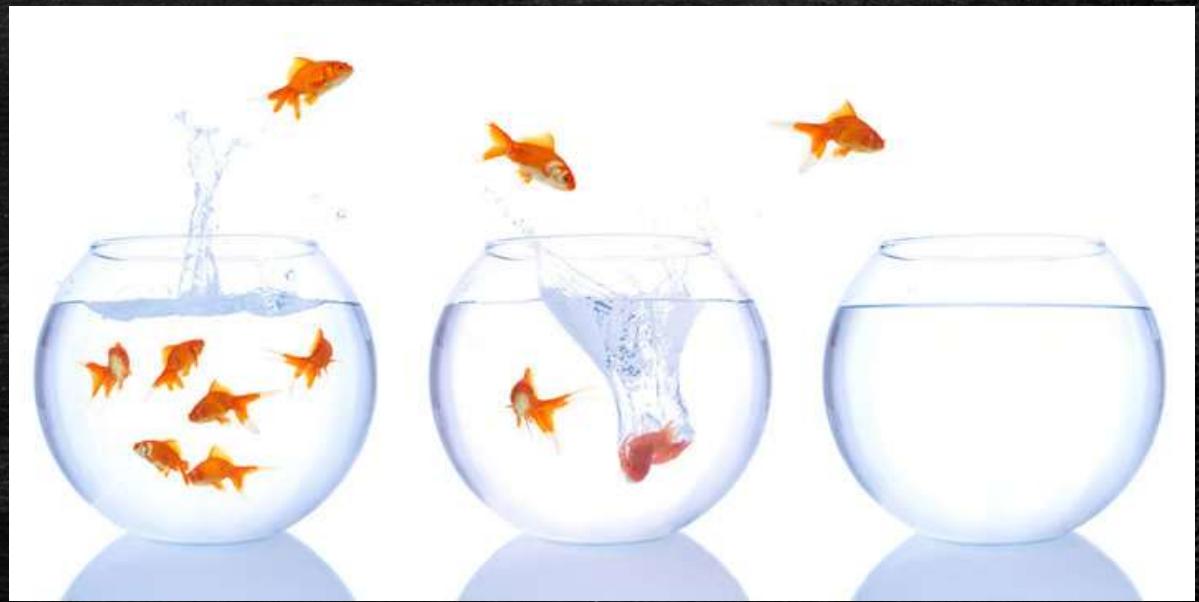


Image Histograms

- แสดงการกระจายทางสถิตของระดับสีในภาพ

- Grayscale Image

- Brightness distribution
 - Single histogram

- Color image

- Color distribution
 - Separated histogram
 - Combined histogram

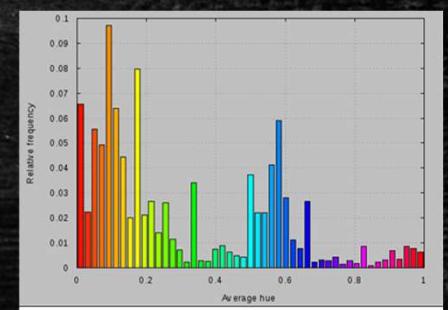
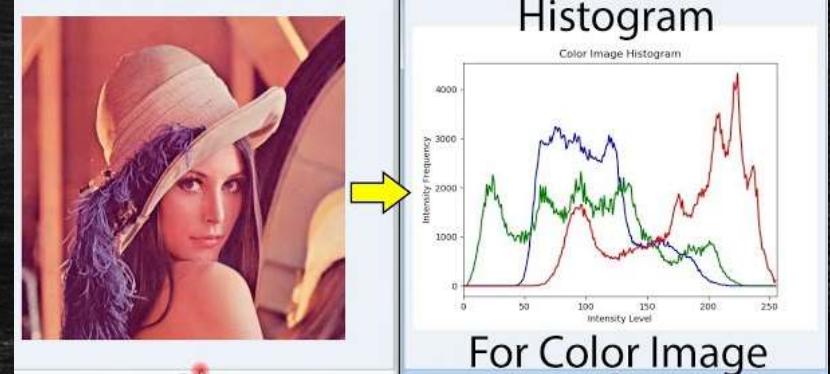
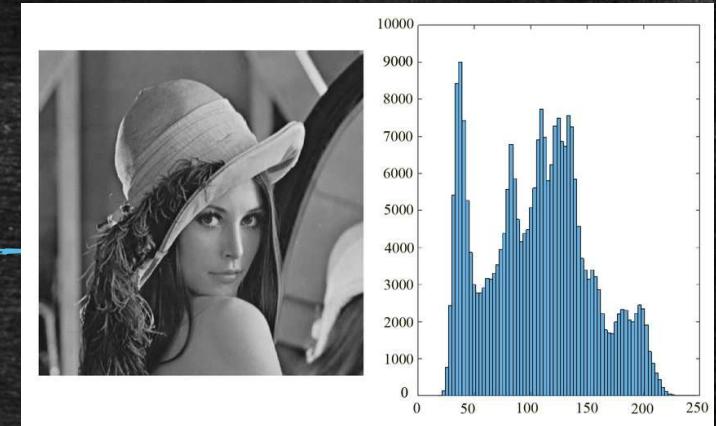


Image Histograms

```
# import numpy as np
# import pandas as pd
# import cv2 as cv
# from google.colab.patches import cv2_imshow
# from skimage import io
# from PIL import Image
# import matplotlib.pyplot as plt

# url =
#"https://t4.ftcdn.net/jpg/00/02/65/21/360_F_2652182_kzshWTrp8SC1KwkMbLnt0NCVtD1XUF.jpg"
# image = io.imread(url)
# cv2_imshow(image)

plt.hist(image.ravel(), bins = 256, range = [0,256])
plt.show()
```

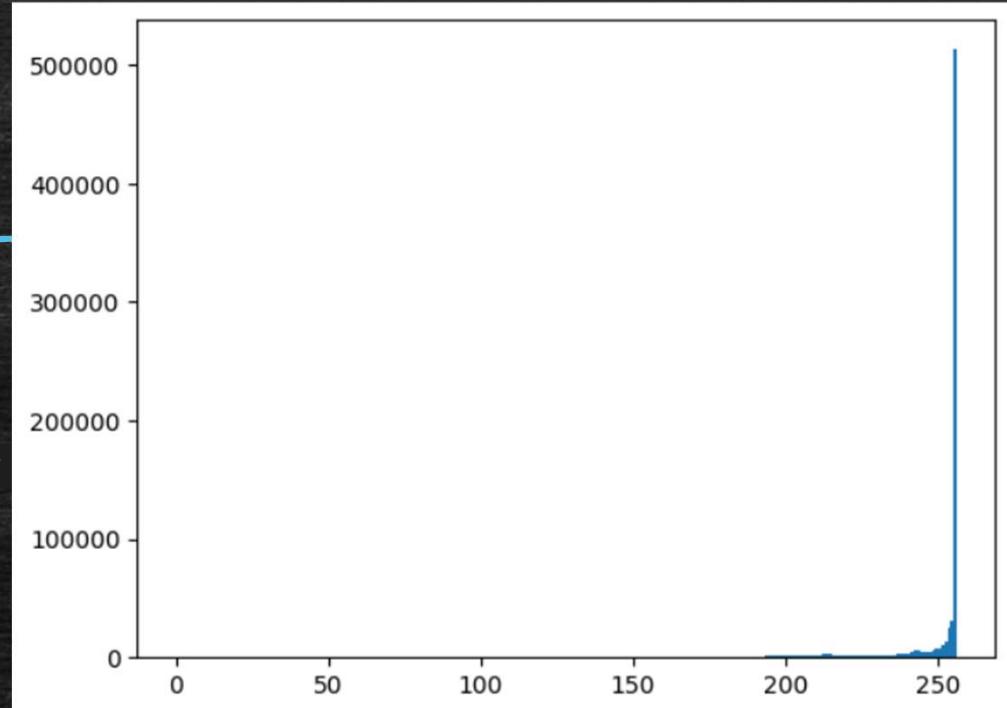


Image Histograms

```
color = ('b','g','r')
for i,col in enumerate(color):
    histr =
cv.calcHist([image],[i],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.show()
```

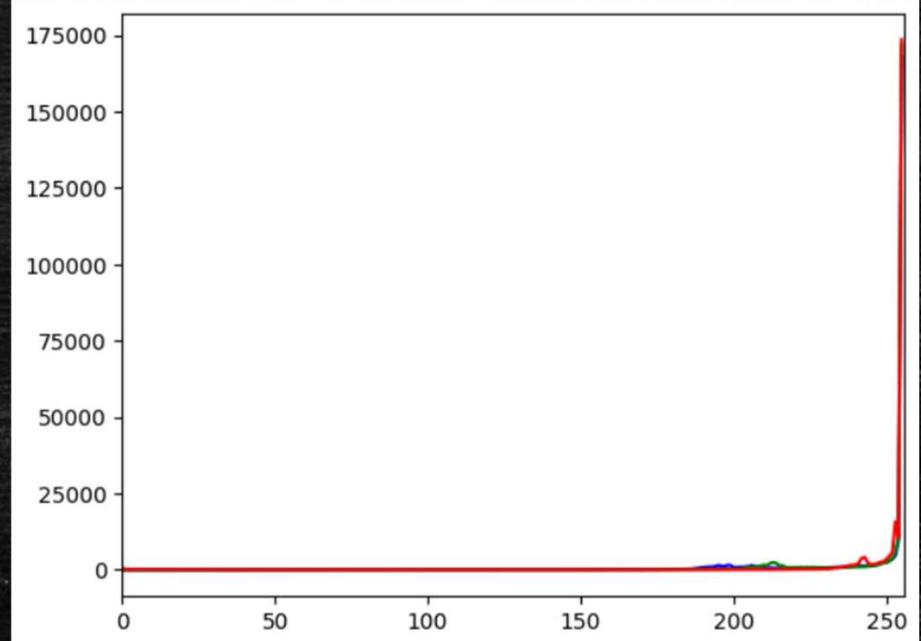


Image Histograms

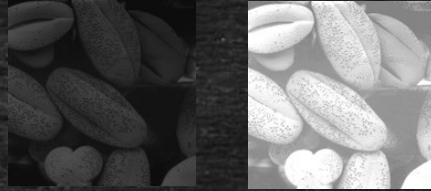
- PDF & CDF
 - The PDF (probability density function) is denoted by $p(x)$

$$p(x) = \frac{d[P(x)]}{dx} = P'(x)$$

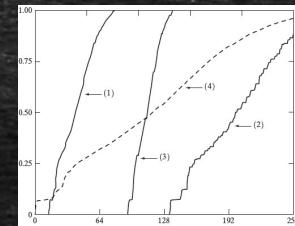
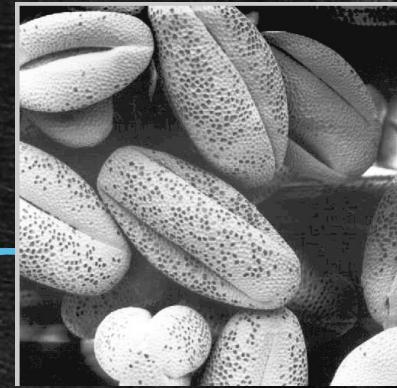
- The CDF (cumulative density function) is denoted by $P(x)$

$$P(x = x_n) = \int_{x=x_0}^{x_n} p(x) dx$$

Image Histograms



Histogram Equalization



- Image Processing

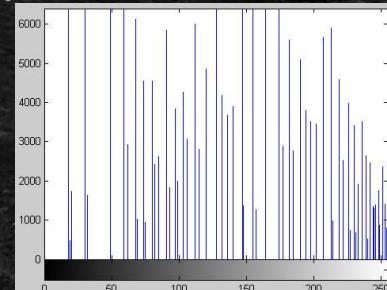
- Histogram Equalization

- ช่วยเพิ่ม Contrast ให้กับภาพโดยเฉพาะบริเวณซึ่งที่มีความหนาแน่นสูงใน Histogram

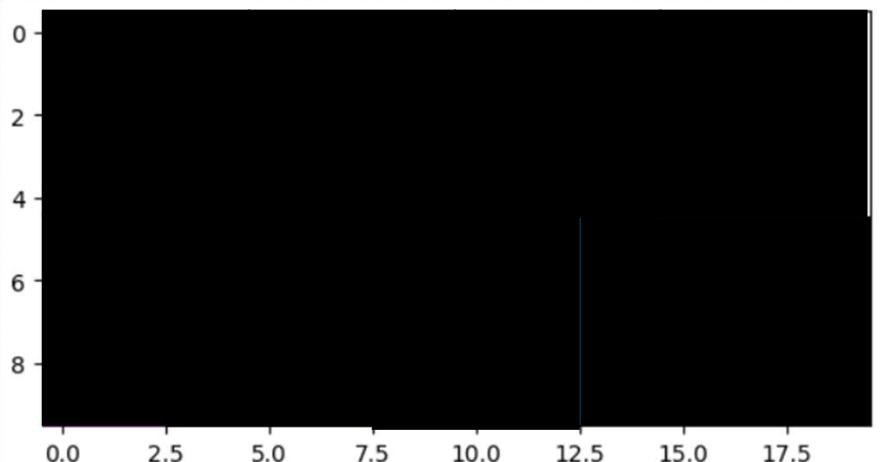
- ทำให้ Histogram กระจายตัวอย่างสม่ำเสมอ (uniform)

- Histogram Matching (Specification)

- เป็นการประมวลผลภาพเพื่อให้ Histogram ของภาพ Output มีรูปแบบตามที่ต้องการ



Writing an image



```
import numpy as np
from matplotlib import pyplot as plt
img_rgb = np.zeros( (10,20,3),
dtype=np.uint8)
img_rgb [ 0:5,  0:5,  0] = 255
img_rgb [ 0:5,  5:10, 1] = 255
img_rgb [ 0:5,  10:15, 2] = 255
img_rgb [ 0:5,  15:20, :] = 255

img_rgb [5:10,  3:8,  0:2] = 255
img_rgb [5:10,  8:13,  1:3] = 255
img_rgb [5:10,  0:3,      0] = 255
img_rgb [5:10,  0:3,      2] = 255

plt.imshow(img_rgb)
plt.show()

cv2.imwrite('/content/drive/MyDrive/Colab
Notebooks/NextGen Ai
Camp/test.png',img_rgb)
```

Image Enhancement in the Spatial Domain

- Basic Gray Level Transformations
- Histogram Processing
- Enhancement Using Arithmetic/Logic Operations
- Basics of Spatial Filtering
- Smoothing Spatial Filters
- Sharpening Spatial Filters
- Combining Spatial Enhancement Methods

Spatial Image Filtering

- ภาพโดยทั่วไป มักจะมีสิ่งรบกวน (Noise) ปะปนเข้ามาเสมอ
- สิ่งรบกวนในภาพอาจมีสาเหตุมาจากการ
 - สภาพแวดล้อมในขณะรับภาพ
 - ความผิดปกติของ Sensor
- เมื่อภาพมีสิ่งรบกวนเกิดขึ้น ทำให้ภาพสูญเสียรายละเอียดไป การกำจัดสิ่งรบกวนนั้นเป็นไปได้ยาก เนื่องจากความไม่แน่นอนของตำแหน่ง และขนาดของสิ่งรบกวน
- การปรับปรุงคุณภาพของภาพอีกประเภท คือการกำจัด หรือลดสิ่งรบกวนที่เกิดขึ้นมาในภาพ

Spatial Image Filtering

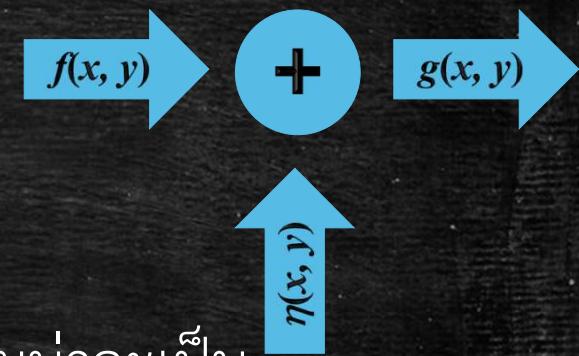
- เทคนิคที่ใช้ในการลดสิ่งรบกวนในภาพ คือ Noise Filtering โดย อาศัยภาพย่อย (Subimage) หรืออาจใช้คำอื่นแทน ได้แก่ ตัวกรอง (Filter), หน้ากาก (Mask), เคอเนล (Kernel), เทมเพลต (Template) หรือ หน้าต่าง (Windows) เคลื่อนไปประมวลผลใน ภาพ
- ซึ่งการลดสิ่งรบกวนในภาพนั้น ไม่สามารถกำจัดสิ่งรบกวนได้ ทั้งหมด แต่สามารถช่วยให้ภาพมีรายละเอียด หรือมีคุณภาพดีขึ้น พอที่จะนำไปประมวลผลต่อได้

Noise Model

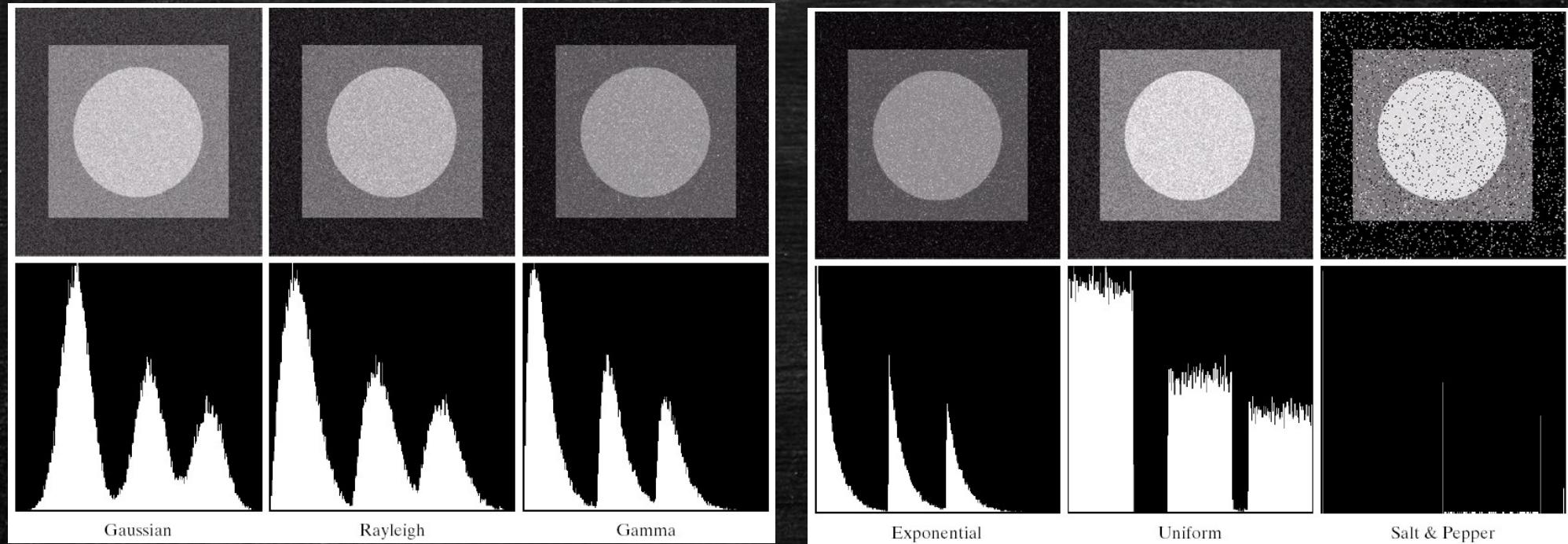
- รูปแบบและลักษณะของสิ่งรบกวนที่เกิดขึ้น

$$g(x, y) = f(x, y) + n(x, y)$$

- ชนิดของสิ่งรบกวนแบ่งตามลักษณะการกระจายความน่าจะเป็น
 - Gaussian noise
 - Rayleigh noise
 - Erlang (Gamma) noise
 - Exponential noise
 - Uniform noise
 - Impulse (salt-and-pepper) noise



Noise Model



Noise Model

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

!wget "https://www.ce.kmitl.ac.th/api/faculty/download/Jirasak"

img = plt.imread('Jirasak')

[w, h, c] = img.shape

no = 1
if no == 1 :
    #Gaussian Noise
    gauss_noise = np.zeros( (w,h,c) , dtype=np.uint8)
    cv2.randn(gauss_noise,128,20)
    gauss_noise = (gauss_noise*0.5).astype(np.uint8)
    noise_img = gauss_noise
```



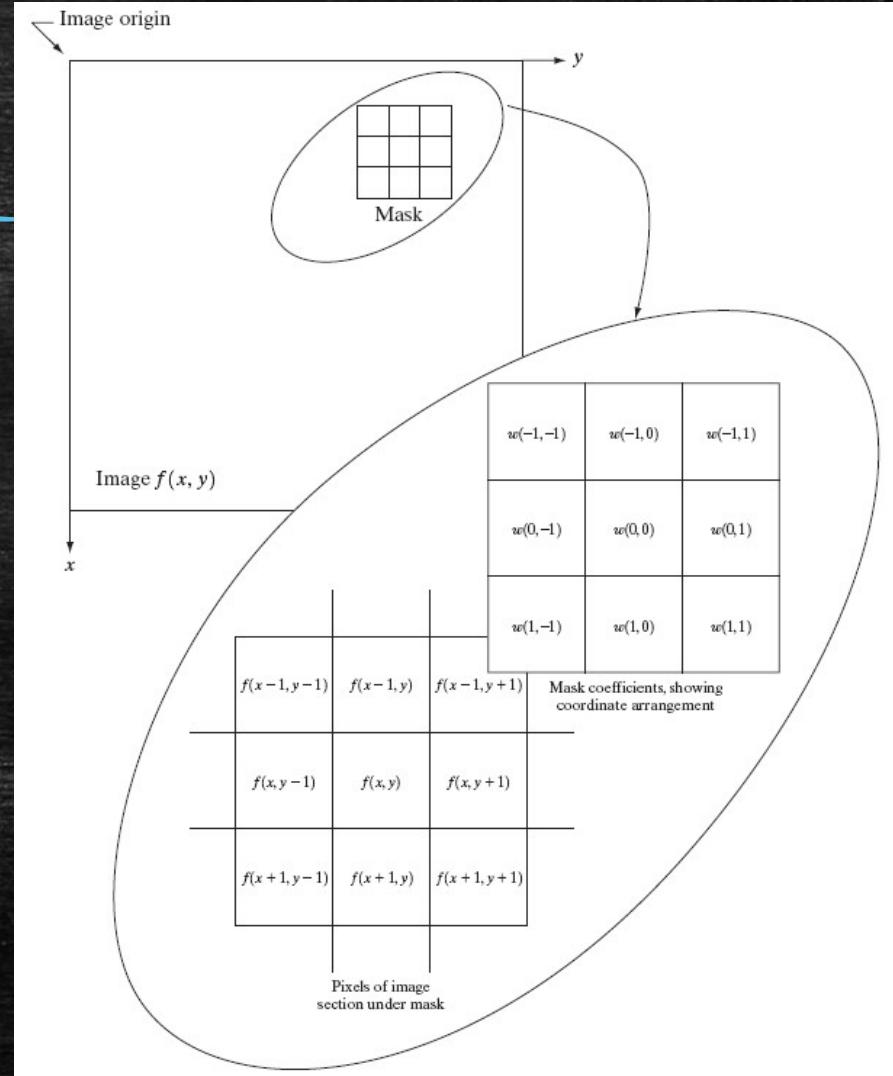
Noise Model

```
elif no == 2 :  
    #Uniform Noise  
    uni_noise = np.zeros( (w,h,c) , dtype=np.uint8)  
    cv2.randu(uni_noise,0,255)  
    uni_noise = (uni_noise*0.5).astype(np.uint8)  
    noise_img = uni_noise  
elif no == 3 :  
    imp_noise = np.zeros( (w,h,c) , dtype=np.uint8)  
    cv2.randu(imp_noise,0,255)  
    imp_noise = cv2.threshold(imp_noise,245,255,cv2.THRESH_BINARY) [1]  
    noise_img = imp_noise  
  
img_n =cv2.add(img, noise_img)  
  
image_RGB = cv.cvtColor(n_img, cv.COLOR_BGR2RGB)  
cv_imshow(image_RGB)
```

Windowing Filtering Technique

- ในการลดสิ่งรบกวนในภาพไม่สามารถทำได้ด้วยการจัดการเฉพาะภายในพิกเซลแต่ละพิกเซล
- จำเป็นต้องอาศัยการพิจารณาระดับความเข้มแสงของพิกเซล ข้างเคียงมาช่วยในการถ่วงน้ำหนักด้วย
- โดยใช้หน้าต่างของจุดภาพขนาดต่างๆ มาช่วยถ่วงน้ำหนัก เช่น หน้าต่างของจุดภาพขนาด 3×3 หรือ ขนาด 5×5 หรือขนาดใหญ่กว่านั้น

- เทคนิคพื้นฐานที่นำมาใช้ในการลดสัญญาณรบกวนด้วยการใช้ภาพย่อๆ ที่นิยมใช้เด็ก
 - Convolution
 - Correlation



Convolution technique

- ใช้เทคนิคในการเลื่อนหน้าต่างอย่างขนาด $m \times n$ ที่มีค่า $w(x, y)$ คือ สัมประสิทธิ์การกรอง (น้ำหนักถ่วงของเทมเพลตตัวกรอง หรือ ค่าตัวถ่วงน้ำหนักการกรอง)
- ผลจากการกระทำที่พิกเซล $g(x, y)$ คือ

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

- เมื่อ $a = \frac{(m-1)}{2}$ $b = \frac{(n-1)}{2}$

Convolution technique

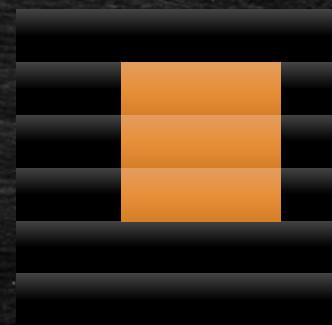
- ตัวอย่าง ในการใช้หน้าต่างย่อขนาด 3×3

$$W = \begin{bmatrix} w(-1, -1) & w(-1, 0) & w(-1, 1) \\ w(0, -1) & w(0, 0) & w(0, 1) \\ w(1, -1) & w(1, 0) & w(1, 1) \end{bmatrix}$$

- ผลจากการกระทำที่พิเศษ $g(x, y)$ คือ

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

$$\begin{aligned} g(x, y) = & w(-1, -1)f(x+1, y+1) + w(-1, 0)f(x+1, y) + w(-1, 1)f(x+1, y-1) \\ & + w(0, -1)f(x, y+1) + w(0, 0)f(x, y) + w(0, 1)f(x, y-1) \\ & + w(1, -1)f(x-1, y+1) + w(1, 0)f(x-1, y) + w(1, 1)f(x-1, y-1) \end{aligned}$$



Pseudo code Convolution technique

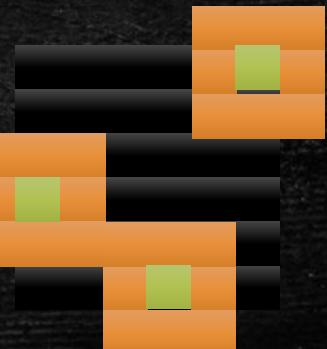
- กำหนดค่า หน้าต่างย่อย w
- คำนวณ หาค่า a, b
$$a = (m-1)/2$$
$$b = (n-1)/2$$
- คำนวณระดับความเข้มแสงภาพผลลัพธ์แต่ละจุดภาพ
Loop i = -a to a
Loop j = -b to b
$$g(x,y) = g(x,y) + w(i,j)f(x-i,y-j)$$
End**End**

Correlation technique

- ค่า น้ำหนักตัวถ่วง โดยทั่วไป ใน theme เพลต จะมีคุณสมบัติสมมาตร จึงทำให้ผลลัพธ์ที่ได้จากการทำ Convolution และ Correlation ไม่แตกต่างกัน
 - ทำให้บางครั้งจะนำเทคนิค Correlation มาใช้เพื่อลดสัญญาณรบกวนเนื่องจากการเขียนโปรแกรมค่อนข้างง่าย
- แต่ถ้าค่า น้ำหนักตัวถ่วง ใน theme เพลต ไม่สมมาตร จะทำให้ผลลัพธ์ที่ได้แตกต่างกัน

ปัญหาในการใช้ Windowing Filtering Technique

- ผลลัพธ์ของ $g(x, y)$ อาจได้เป็นค่าลบ หรือเป็นค่ามากเกินช่วงของข้อมูลที่กำหนดไว้ ซึ่งแก้ไขได้โดย
 - กำหนดชนิดข้อมูล g เป็นแบบมีเครื่องหมาย
 - แปลงค่าที่ต่ำกว่า 0 โดยเปลี่ยนจำนวนบิตที่เก็บข้อมูล
 - ทำการ normalize ผลลัพธ์ $g(x, y)$ ให้อยู่ในช่วงเดียวกับ $f(x, y)$
- ปัญหาที่เกิดขึ้นจากตำแหน่งบริเวณขอบภาพ ไม่สามารถใช้ Windowing Filtering Technique ได้



การแก้ไขปัญหาบริเวณขอบภาพของ Windowing Filtering

- ไม่ทำ Windowing Filtering ที่ขอบภาพ
- ลอกค่าของ $f(x, y)$ ที่ไปแทนขอบภาพใน $g(x, y)$
- ตัดขอบภาพใน $g(x, y)$ ออก
- ใช้หน้าต่างที่เหมาะสมกับบริเวณของภาพ
- การกำหนดดัชนีแบบสะท้อน (Reflected indexing)
- การใช้ดัชนีภาพแบบวนรอบ (Circular indexing)

Linear filtering

- Arithmetic mean filter
- Geometric mean filter
- Harmonic mean filter
- Contra harmonic mean filter

Arithmetic mean filter

- ใช้วิธี Convolution ซึ่งมีค่า น้ำหนักถ่วงของเทมเพลตตัวกรอง เป็นค่าเดียวในทุกตำแหน่ง โดยค่า น้ำหนักถ่วงถูก Normalize ด้วย จำนวนพิกเซลทั้งหมดใน เทมเพลตตัวกรอง เท่ากับ $m \times n$ ค่า arithmetic mean สามารถเขียนเป็นสมการได้ดังนี้

$$g(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} f(s,t)$$

- ข้อดี : เป็นวิธีการทำ mean filter ที่ง่ายที่สุด
- ข้อเสีย : ภาพผลลัพธ์ที่ได้จะเบลอ และความคมชัดลดลง

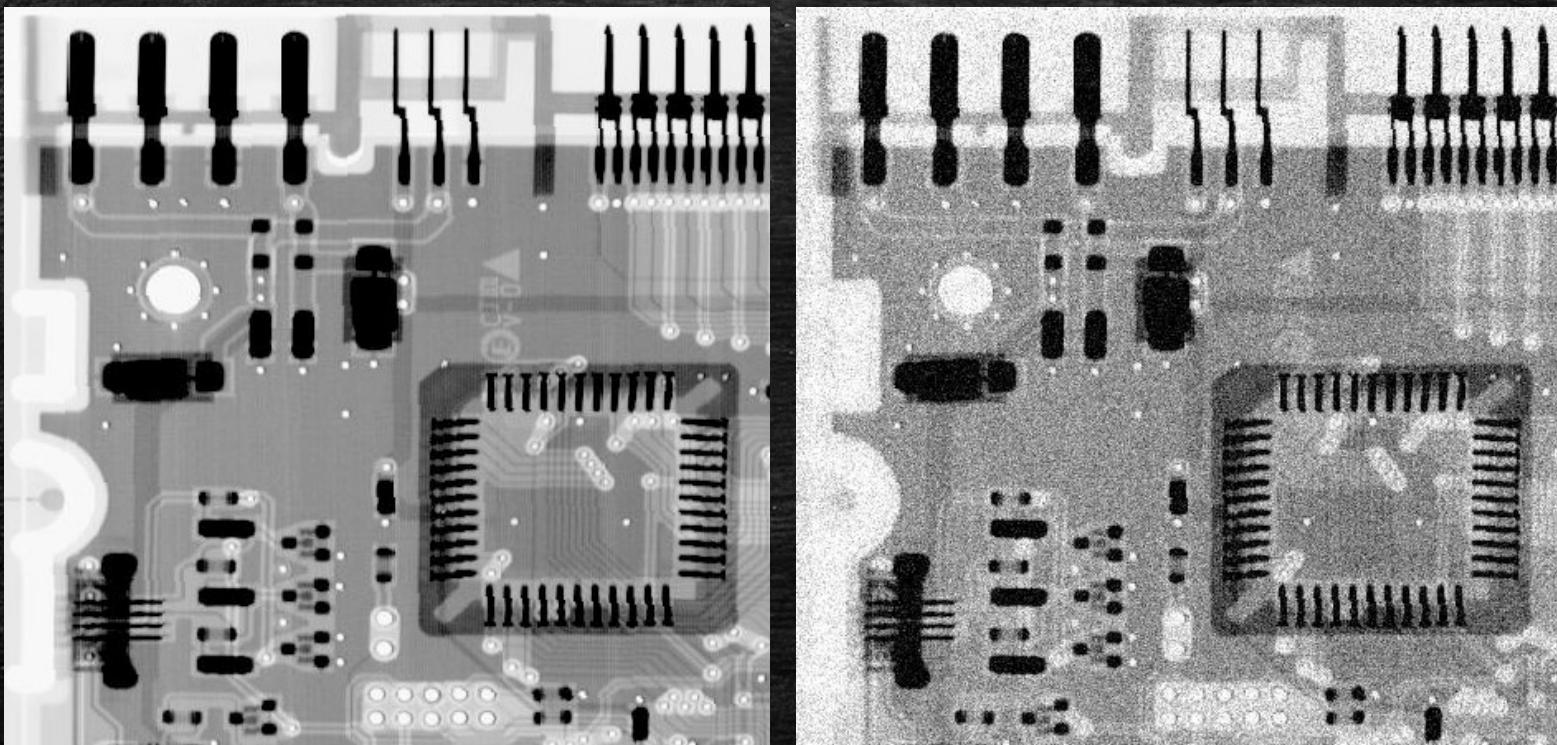
Geometric mean filter

- ใช้การคำนวณผลคูณของค่าในตัวแทนพิกเซลทั้งหมดใน 1×1 เทมเพลตตัวกรอง ดังนี้

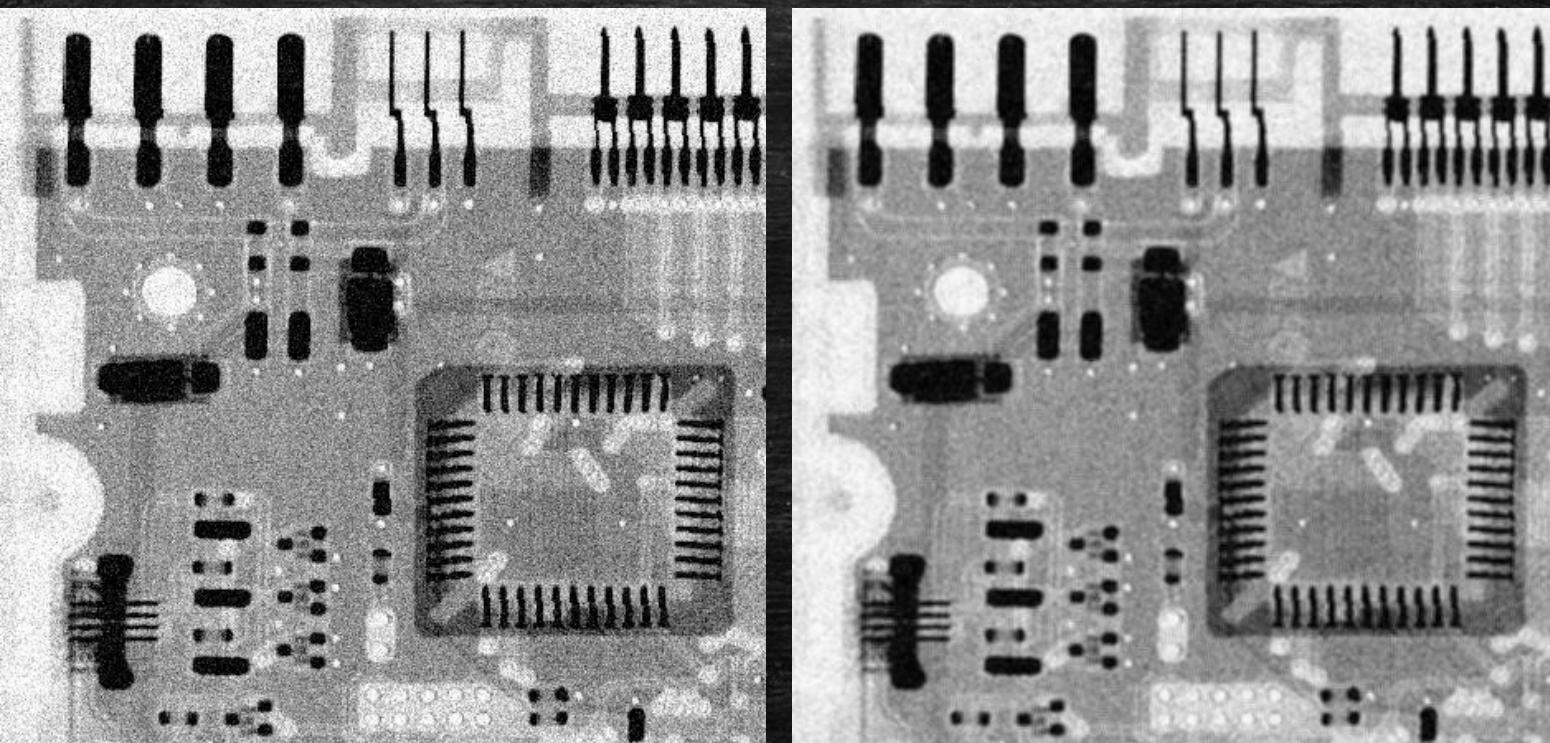
$$g(x, y) = \left[\prod_{(s, t) \in S_{xy}} f(s, t) \right]^{\frac{1}{mn}}$$

- ทำการปรับภาพให้เรียบ (smoothing) ได้เช่นเดียวกับการใช้วิธี arithmetic mean
- ข้อดี : ให้ภาพผลลัพธ์มีรายละเอียดและคมชัดมากกว่า วิธี arithmetic mean
- ข้อเสีย : การคำนวณจะทำได้ยากและใช้เวลามากขึ้น เนื่องจากต้องทำการคูณ และต้องยกกำลังส่วนกลับของผลคูณด้วย

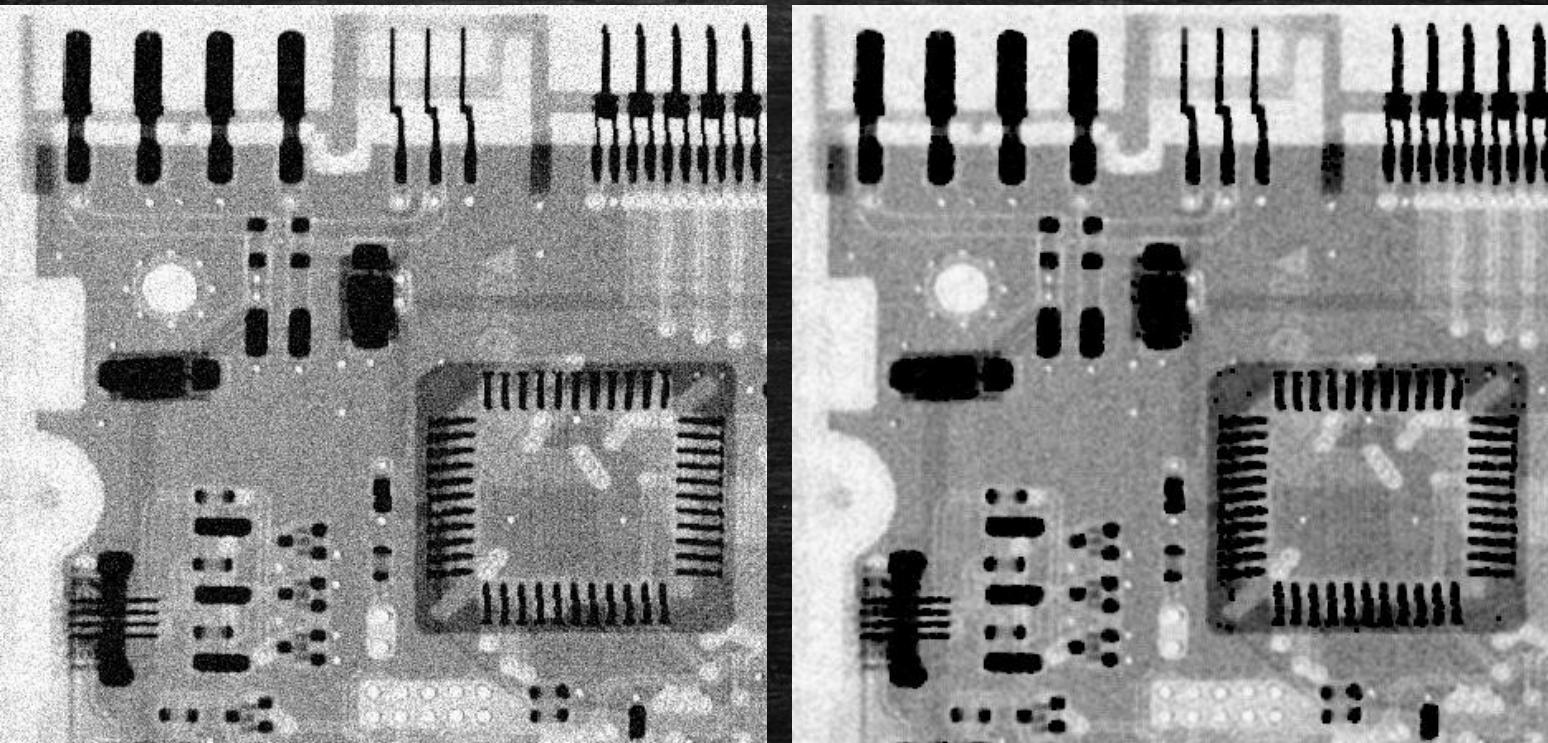
Example : X-ray image & Image corrupted by noise



Example : X-ray image & Result of Arithmetic mean filter



Example : X-ray image & Result of Geometric mean filter



Harmonic mean filter

- สมการเป็นดังนี้
$$g(x, y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{f(s, t)}}$$
- เทคนิคนี้ใช้ในการลดผลกระทบของ salt noise ได้ดี (ใช้ได้ดีกับสิ่งรบกวนชนิดอื่นๆ เช่น Gaussian noise)
- แต่ไม่ได้กับ pepper noise

Image with salt noise & Harmonic mean filter

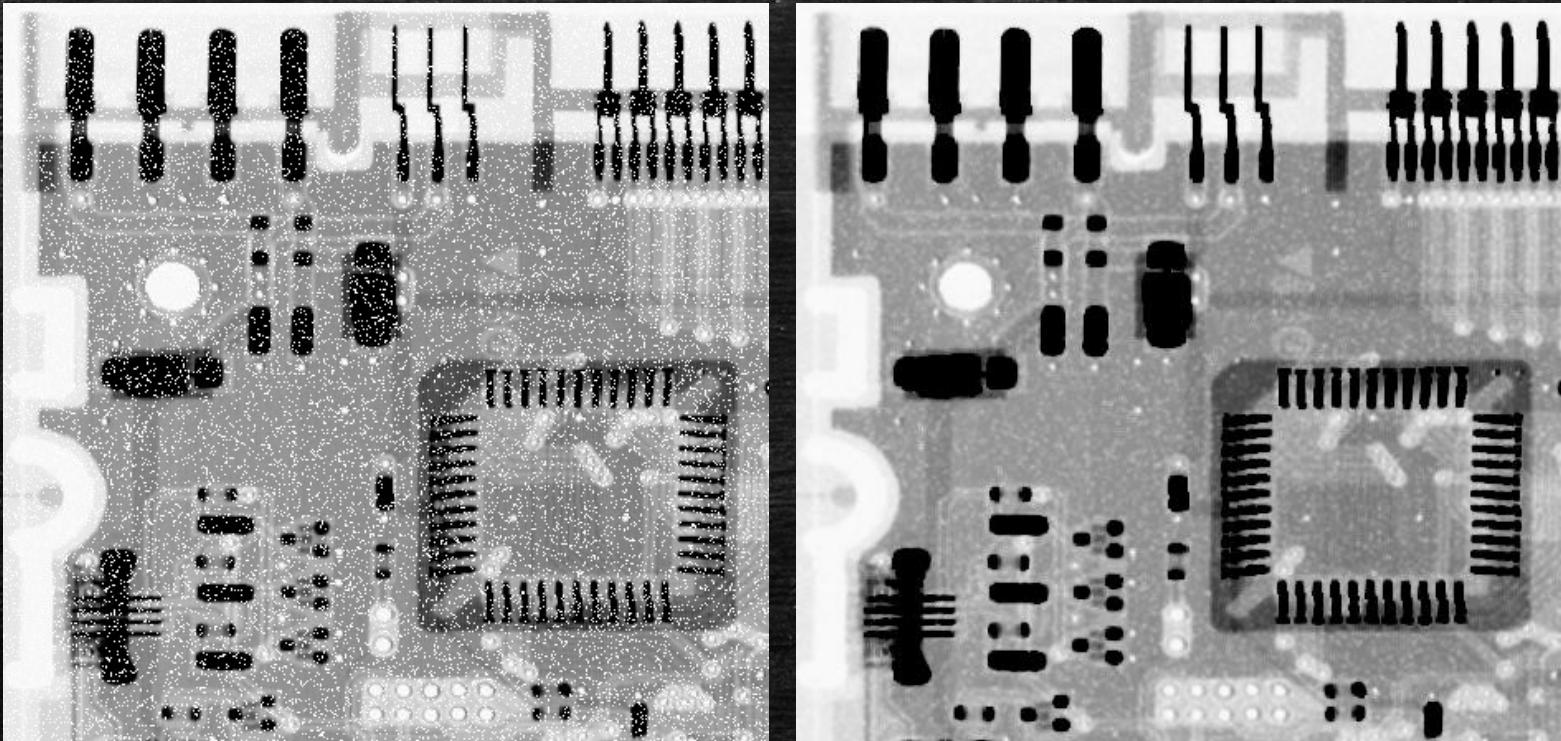
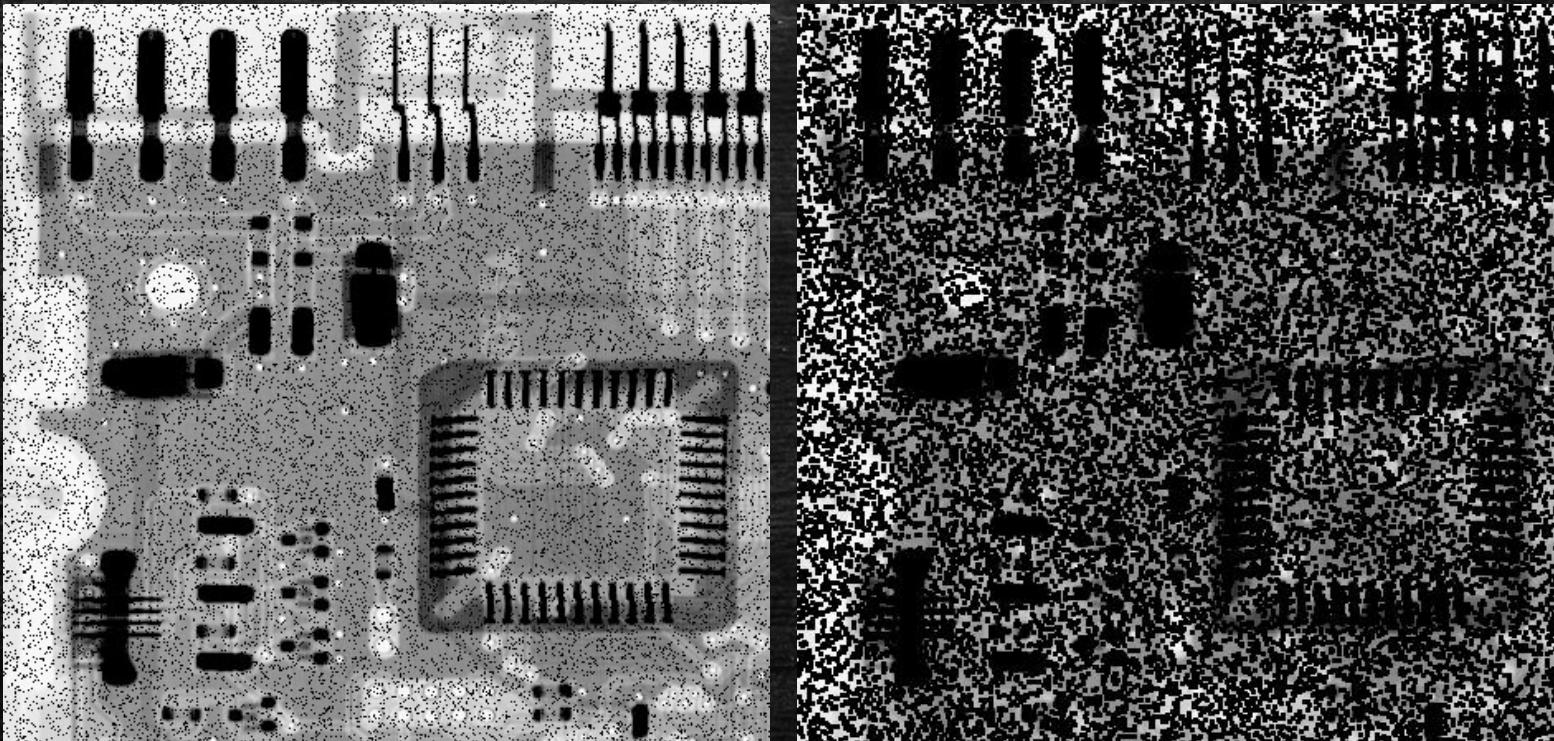


Image with pepper noise & Harmonic mean filter



Contra harmonic mean filter

- เทคนิคนี้สามารถกำจัดให้ลด pepper noise หรือ salt noise ได้ สมการเป็นดังนี้

$$g(x,y) = \frac{\sum_{(s,t) \in S_{xy}} f(s,t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} f(s,t)^Q}$$

- Q เป็นค่า order (กำลัง) ของ filter
 - กำจัดให้ Q เป็นค่าบวกจะใช้ลด pepper noise
 - กำจัดให้ Q เป็นค่าลบจะใช้ลด salt noise
- ข้อดี : ให้ผลลัพธ์ที่ดี ไม่ทำให้ภาพเบลอ ไม่ลดความคมชัดของภาพ
- ข้อเสีย : มีความซับซ้อน และการเลือกค่า Q ให้เหมาะสม

Image with pepper noise & Contra harmonic mean filter

$$Q = 1.5$$

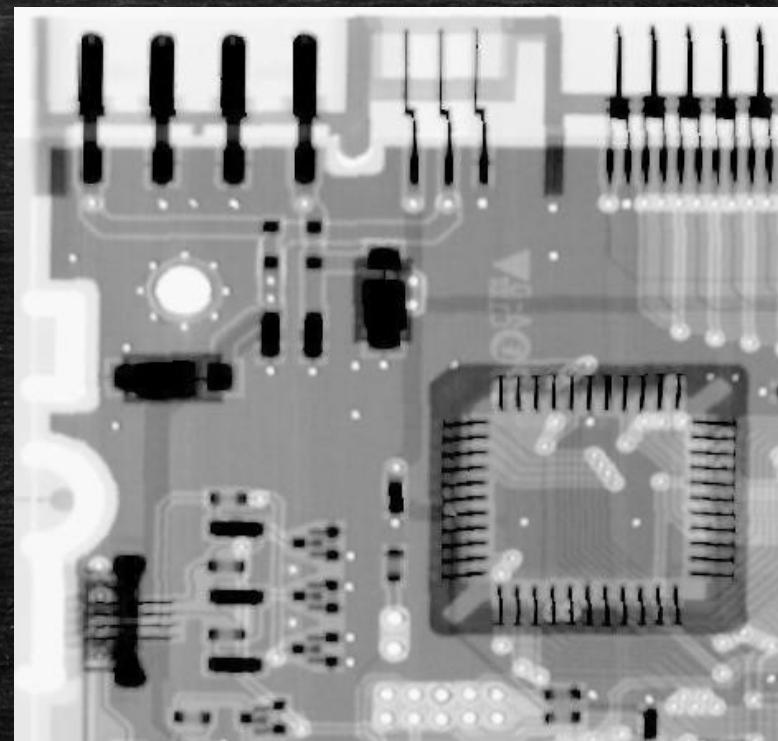
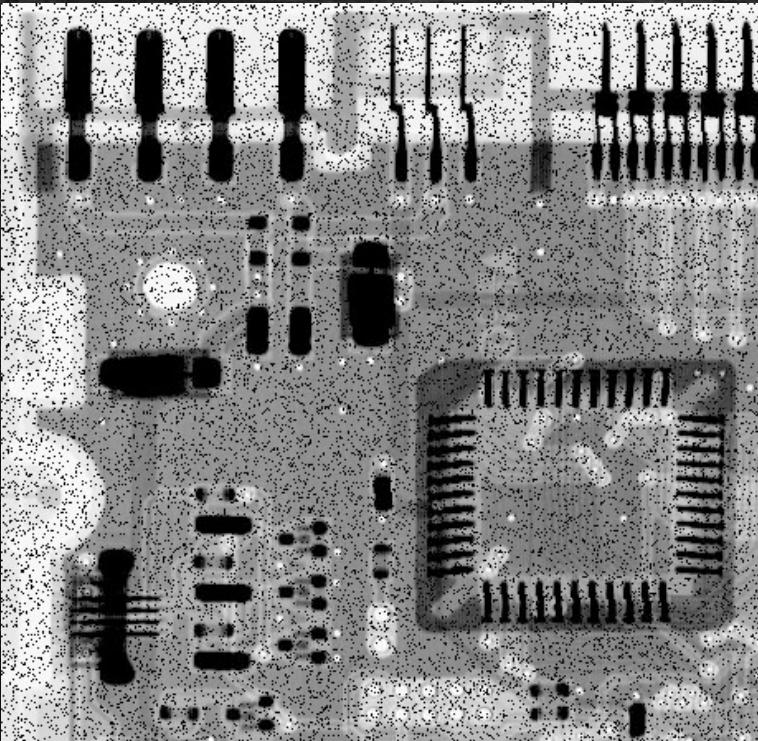


Image with pepper noise & Contra harmonic mean filter

$$Q = -1.5$$

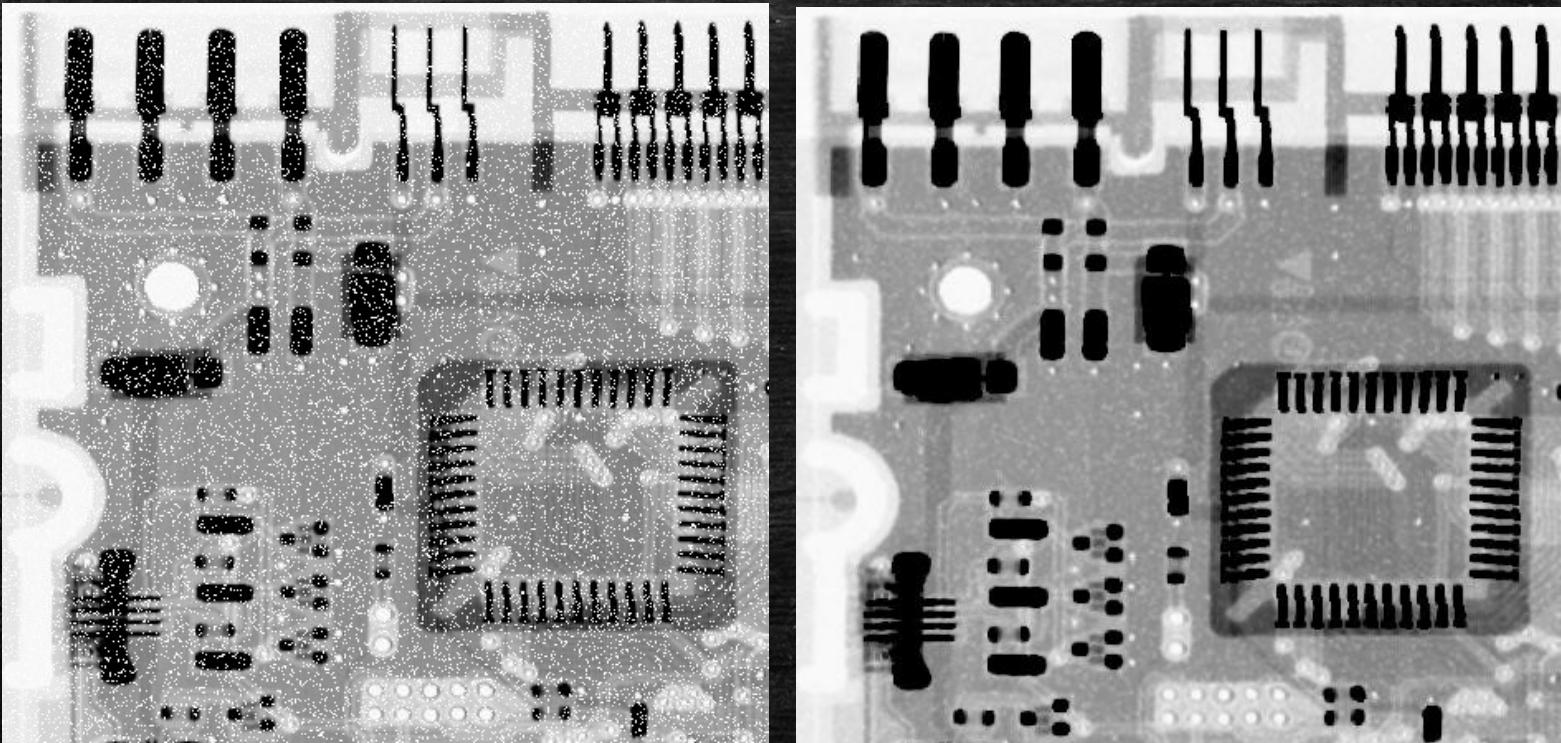


Image with pepper noise & Contra harmonic mean filter

**Results of selecting the wrong sign in
contraharmonic mean filter**

$$Q = -1.5$$

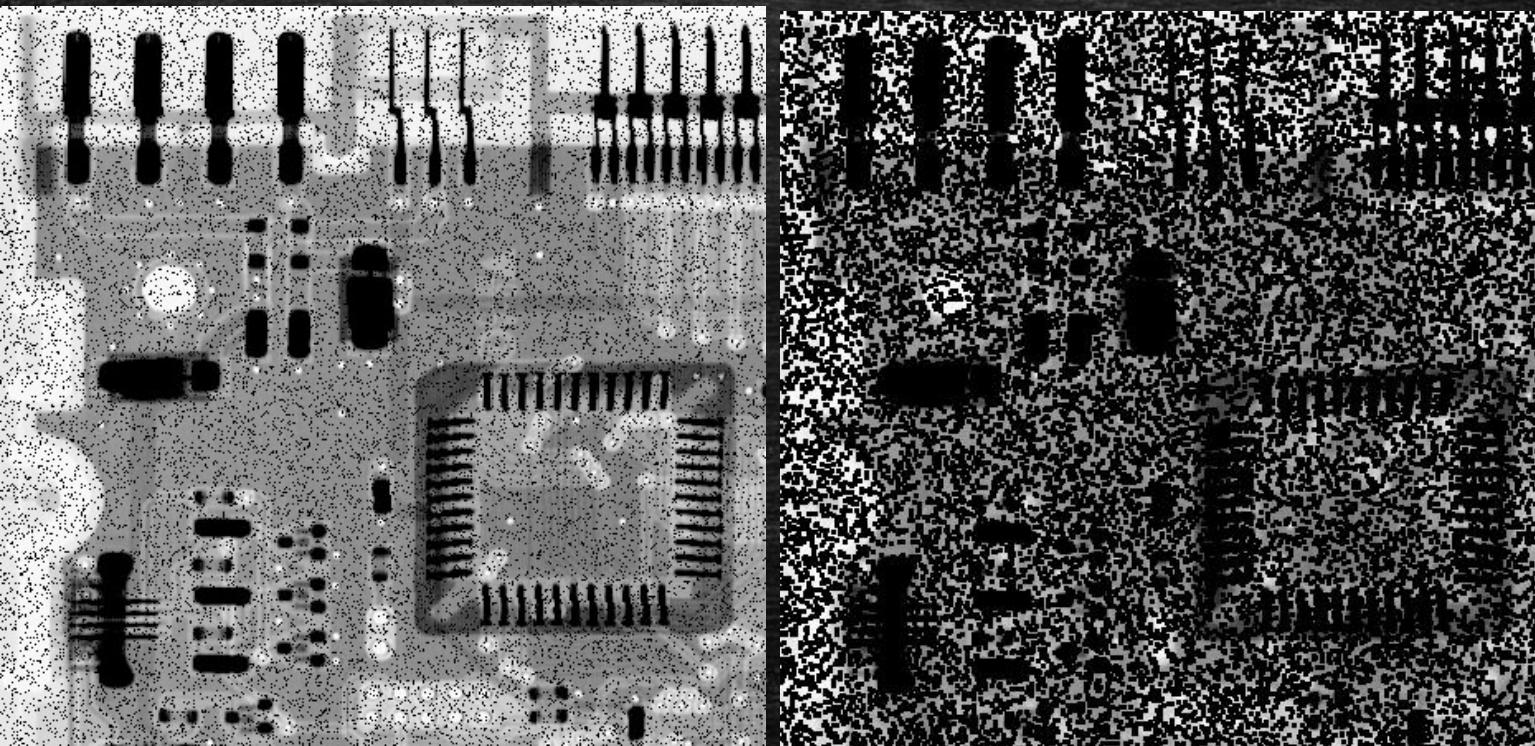


Image with salt noise & Contra harmonic mean filter

**Results of selecting the wrong sign in
contraharmonic mean filter**

$Q = 1.5$

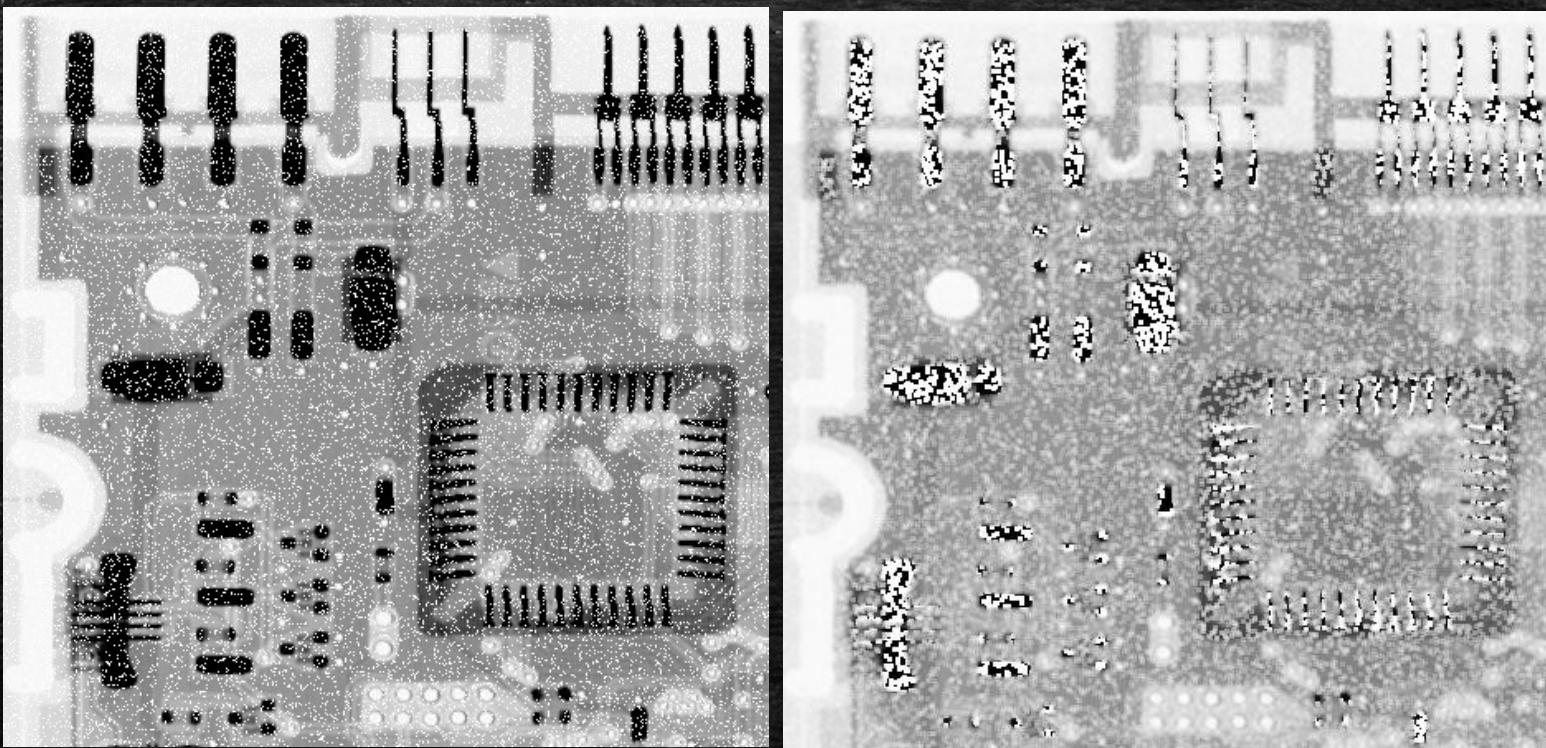
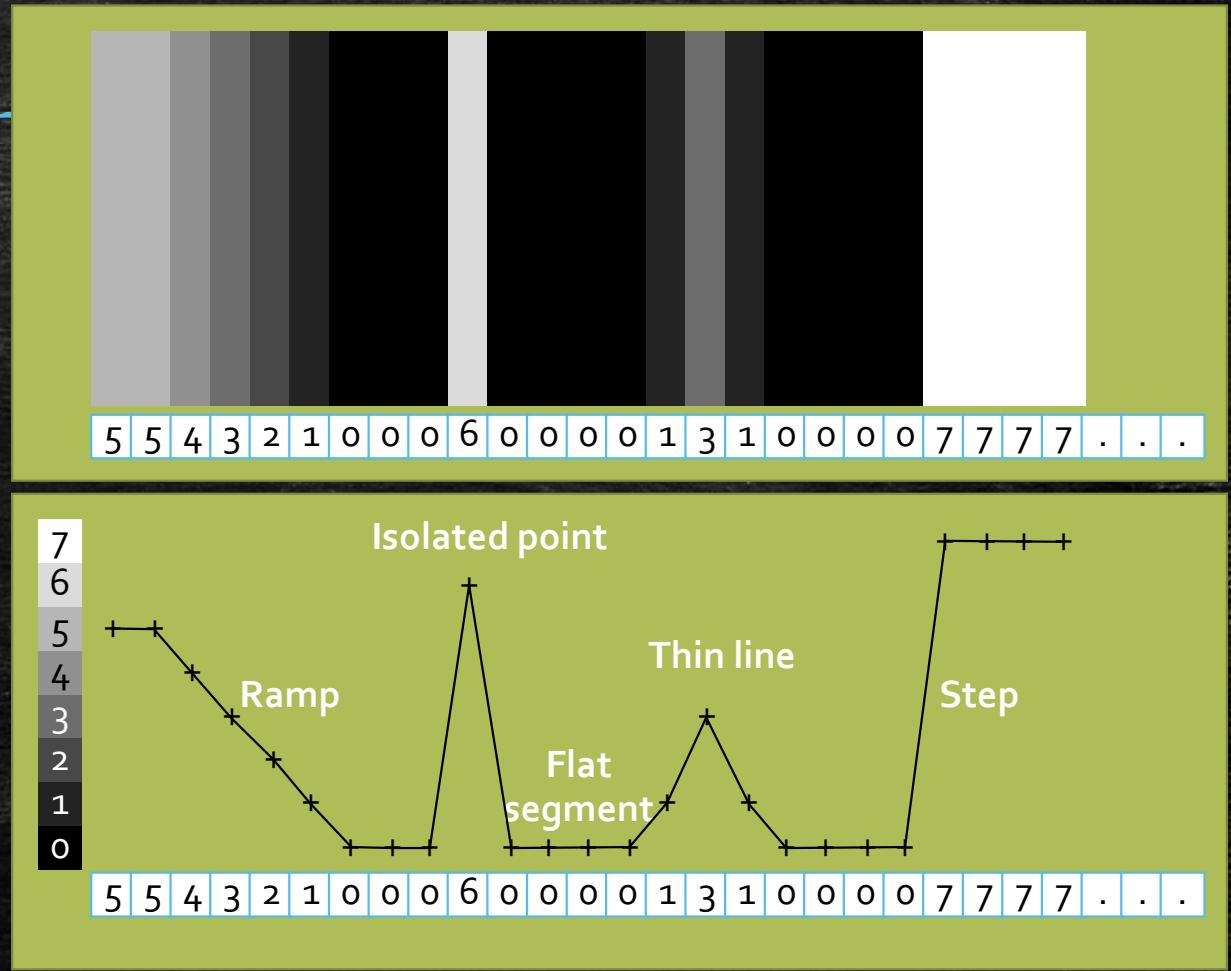


Image sharpening

- การที่ภาพเบลอ (ไม่ชัด หรือละเอียดภายในภาพน้อย) อาจเกิดจากผลกระทบจากสภาพแวดล้อม ประสิทธิภาพของ Sensor หรือเกิดจากกระบวนการลดสิ่งรบกวนในภาพ
- การปรับความคมชัดของภาพ ทำให้ภาพมีรายละเอียดดีขึ้น โดยการหาผลต่างในแกนตั้งแน่น (spatial differentiation) โดยใช้เทคนิค
 - First-order Derivative $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$
 - Second-order derivative $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}$.

Image sharpening



First-order Derivative

- เพื่อเน้นรายละเอียดที่ขอบของวัตถุภายในภาพ
- การหา First-order Derivative ในข้อมูลภาพ ทำโดยการหาผลต่าง ซึ่งมีวิธีในการหาผลต่างหลายวิธี โดยจะมีข้อกำหนดของค่า First-order Derivative ดังนี้
 - ต้องมีค่าเป็น 0 บริเวณจุดภาพที่มีค่าความเข้มแสงคงที่
 - ต้องมีค่าไม่เท่ากับ 0 บริเวณจุดภาพที่มีการเปลี่ยนแปลงระดับค่าความเข้มแสง หรือ เป็นลักษณะ Step หรือ Ramp
 - ต้องไม่มีค่าเป็น 0 ตลอดการเปลี่ยนแปลงช่วง Ramp

First-order Derivative

- การหา First-order Derivative สำหรับฟังก์ชัน 1 มิติของ $f(x)$ สามารถหาผลต่างได้ดังนี้ $\frac{\partial f}{\partial x} = f(x+1) - f(x)$



First-order Derivative

- เหมือนเดิมตัวกรอง ที่นิยมใช้สำหรับ first derivative ได้แก่ Prewitt kernel เป็น เหมือนเดิมตัวกรอง ที่ใช้สำหรับทำการเปลี่ยนแปลงความเข้มแสงในแนวแกน x และแกน y

$$w_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad w_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- w_x และ w_y เรียกอีกอย่างว่า Sobel operator

First-order Derivative

- แทนผลตัวกรอง อีกแบบที่นิยมคือ Robert cross-gradient operator ซึ่งเป็น filter ที่ใช้ในการตรวจการเปลี่ยนแปลงในแนวเส้นทแยงมุม มีค่าดังนี้ $w_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ $w_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

$$g(x, y) \approx |F_1| + |F_2| = |w_1 * f(x, y)| + |w_2 * f(x, y)|$$

$$|F_1| = |w_1 * f(x, y)| = |(-1)f(x, y) + (1)f(x-1, y-1)|$$

$$|F_2| = |w_2 * f(x, y)| = |(-1)f(x, y-1) + (1)f(x-1, y)|$$

First-order Derivative

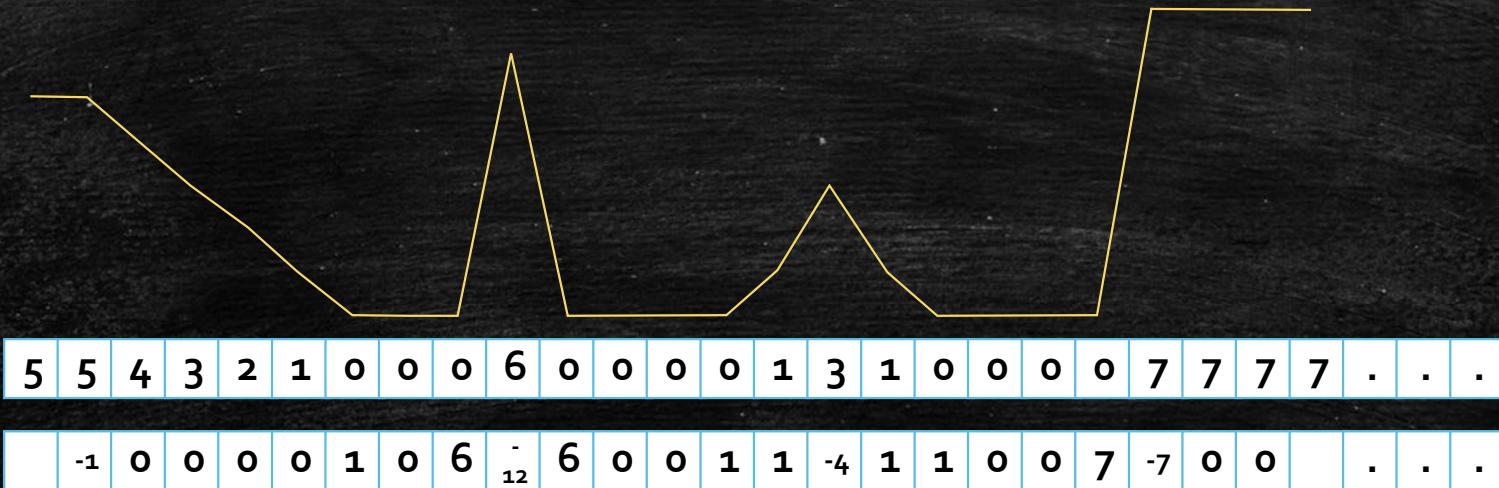
- ปัญหาในการคำนวณค่า Convolution ของ First-order Derivative ได้แก่การกำหนดตัวแปรสำหรับเก็บค่าผลลัพธ์ที่ได้
 - ข้อมูลภาพ f ปกติเป็น unsigned int
 - เทมเพลต พ เป็นจำนวนจริง
 - ข้อมูลภาพจากการประมวลผล g เป็นจำนวนจริง

Second-order derivative

- การหา Second-order Derivative ในข้อมูลภาพ คล้ายกับการหา First-order Derivative โดยจะมีข้อกำหนดของค่า Second-order Derivative ดังนี้
 - ต้องมีค่าเป็น 0 บริเวณจุดภาพที่มีค่าความเข้มแสงคงที่
 - ต้องมีค่าไม่เท่ากับ 0 บริเวณจุดภาพที่มีการเปลี่ยนแปลงระดับค่าความเข้มแสง หรือ เป็นลักษณะ Step หรือ Ramp
 - ต้องมีค่าเป็น 0 ตลอดการเปลี่ยนแปลงช่วง Ramp ซึ่งมี slope คงที่

Second-order Derivative

- การหา Second-order Derivative สำหรับฟังก์ชัน 1 มิติของ $f(x)$ สามารถหาผลต่างได้ดังนี้ $\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$



Second-order Derivative

- การหา Second-order Derivative สำหรับข้อมูลภาพที่เป็นฟังก์ชัน 2 มิติ $f(x,y)$ ได้ดังนี้
$$\nabla^2 F = \begin{bmatrix} F_x^2 \\ F_y^2 \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} \\ \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$
- รูปแบบของ isotropic filter ที่ง่ายที่สุดคือ Laplacian filter ซึ่งเป็น second-order derivative ของฟังก์ชันภาพ

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

Second-order Derivative

- จาก Second-order Derivative สำหรับฟังก์ชัน 1 มิติ

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

- เมื่อหา Second-order Derivative สำหรับฟังก์ชัน 2 มิติจะได้

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

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

Second-order Derivative

- เนื่องจาก Derivatives มีลักษณะเป็น linear operation ฟังก์ชัน Laplacian มีลักษณะเป็น linear operation เช่นกัน

$$\begin{aligned}\nabla^2 f &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\ &= f(x+1, y) + f(x-1, y) + \\ &\quad f(x, y+1) + f(x, y-1) - 4f(x, y)\end{aligned}$$

- สามารถนำมาสร้างเป็น เทมเพลตตัวกรอง ได้เป็น

$$w = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Second-order Derivative

- จากการใช้เทมเพลตตัวกรอง
- ถ้าหากภาพมีการหมุนทุก 90° จะให้ผลลัพธ์จากการ Convolution ไม่ต่างกัน
- หากต้องการให้ผลลัพธ์ไม่ซึ้งกับการหมุนภาพในทิศเส้นทแยงมุม ด้วยต้องปรับสมการ Derivatives เป็น

$$\nabla^2 f = \begin{bmatrix} f(x+1, y) + f(x-1, y) + f(x, y+1) + \\ f(x, y-1) + f(x-1, y-1) + f(x+1, y+1) + \\ f(x-1, y+1) + f(x+1, y-1) - 8f(x, y) \end{bmatrix} \quad w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Second-order Derivative

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

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

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

Second-order Derivative

- Laplacian เป็น derivative operator ที่
 - ให้ค่าผลพัลธ์สูง ตัวแทนที่ค่าความเข้มแสงเปลี่ยนแปลงอย่างไม่ต่อเนื่อง
 - ให้ค่าผลพัลธ์ต่ำ (ไม่นิ่น) ในตัวแทนที่มีการเปลี่ยนแปลงความเข้มแสงน้อย
- ดังนั้นเพื่อเน้นรายละเอียดของภาพโดยเฉพาะตรงขอบการเปลี่ยนแปลง เราทำได้โดยบวกค่าความเข้มแสงเดิมของภาพด้วยค่าความเข้มแสงที่ได้จากการทำ Laplacian filter

ค่ากึ่งกลางเพลตตัวกรอง เป็นค่าติดลบ

$$g(x, y) = f(x, y) - \nabla^2 f(x, y)$$

ค่ากึ่งกลางเพลตตัวกรอง เป็นค่าบวก

$$g(x, y) = f(x, y) + \nabla^2 f(x, y)$$

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

Second-order Derivative

- ในทางปฏิบัติ แทนที่จะต้องคำนวณผลลัพธ์จาก Laplacian filter แล้วนำไปหักออกหรือบวกเพิ่มกับค่าความเข้มแสงจริงของภาพ

$$\begin{aligned} g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) + 4f(x, y)] \\ &= 5f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] \end{aligned}$$

- สามารถทำได้โดยรวมผลของการบวกหรือลบเข้าไปทีตัว เพิ่มเพลิดตัวกรอง ได้ดังนี้

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 5 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

1	1	1
1	-8	1
1	1	1

Second-order Derivative

(a) shows an image of the North Pole of the moon

(b) shows the result of filtering this image with the Laplacian mask [1 1 1; 1 -8 1; 1 1 1]

(c) shows the scaled image since the Laplacian image contains both positive and negative values.

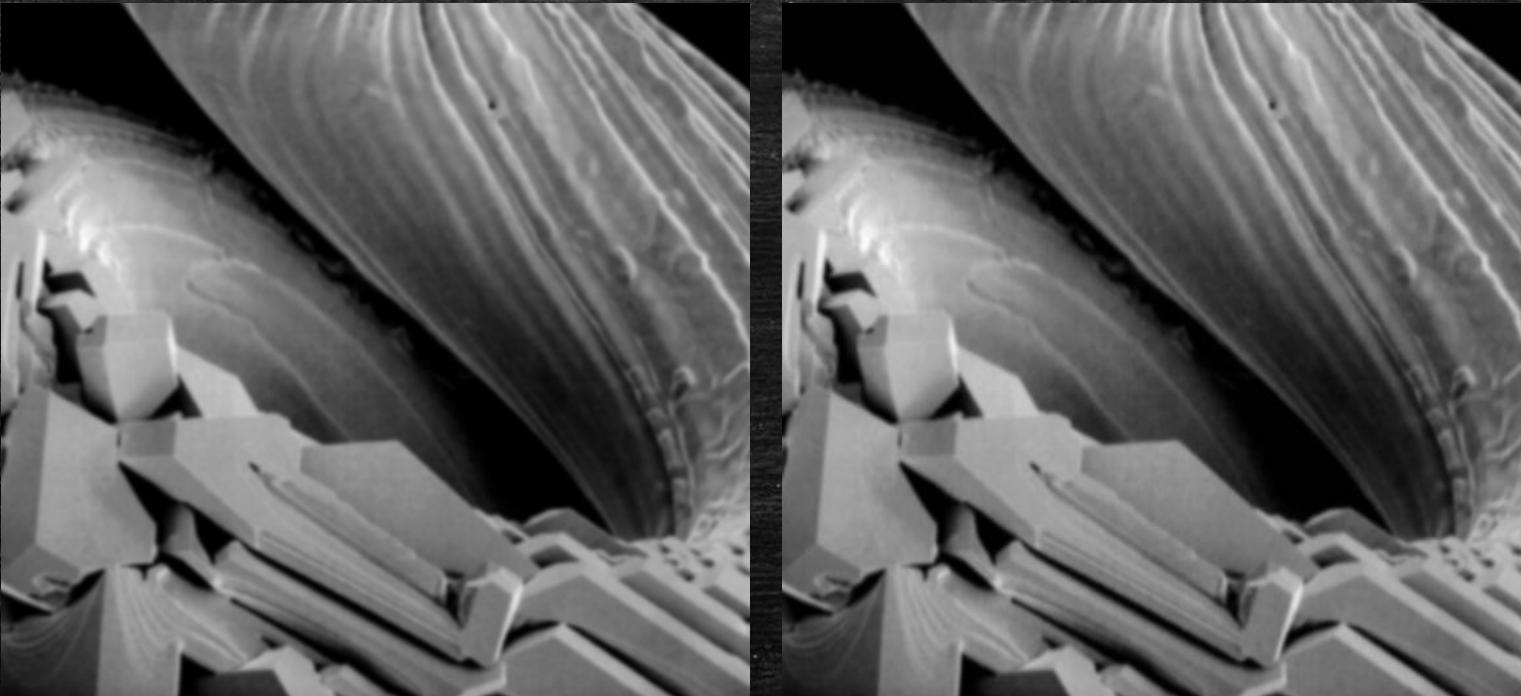
(d) shows the result obtained using

$$g(x, y) = f(x, y) - \nabla^2 f(x, y)$$



Second-order Derivative

0	-1	0
-1	5	-1
0	-1	0



Second-order Derivative

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

