

WorkShop 8 : ImageProcessing

Dr. Ekapol Chuangsuwanich

2110101 ComProg SEC 5

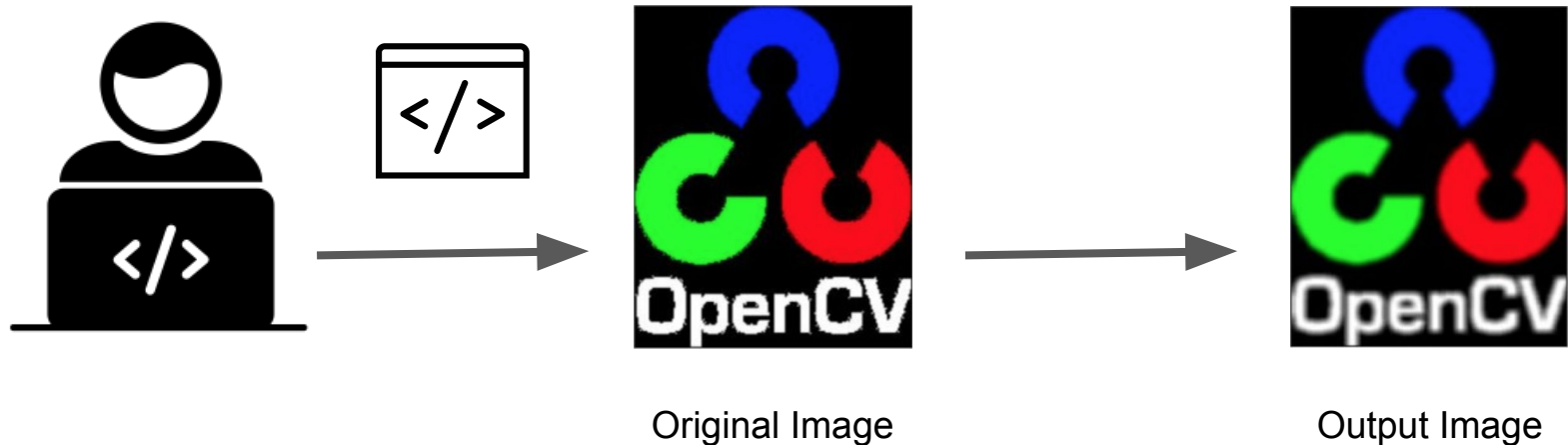
TODAY

Our Topics

- Digital Image Processing
- Application
- Machine Vision
- Image Representation
- Image Data
- Picture Element (Pixel)
- Spatial Resolution
- Image Cropping
- Image Scaling
- Colors
- Greyscale
- Image negative
- Sepia
- Convolution
- Workshop (Find distance)

Digital Image Processing

Digital image processing is the use of computer **algorithm** to perform **image processing** on **digital images**.



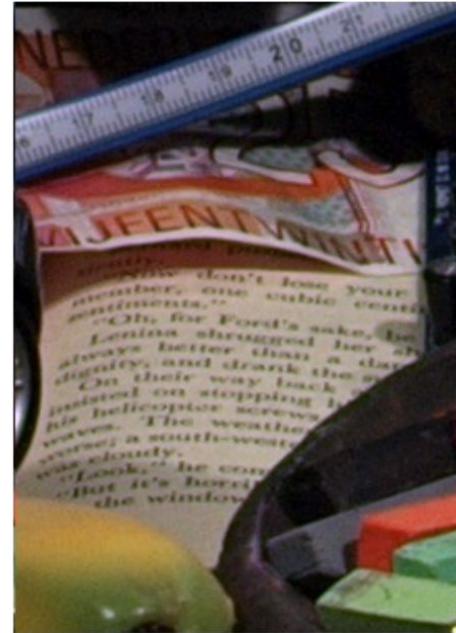
Digital Image Processing

Why do we need image processing?

- To **enhance** pictorial information for human perception.
- To **build** autonomous vision machines for various applications.
- To **make** storage and transmission more efficient.

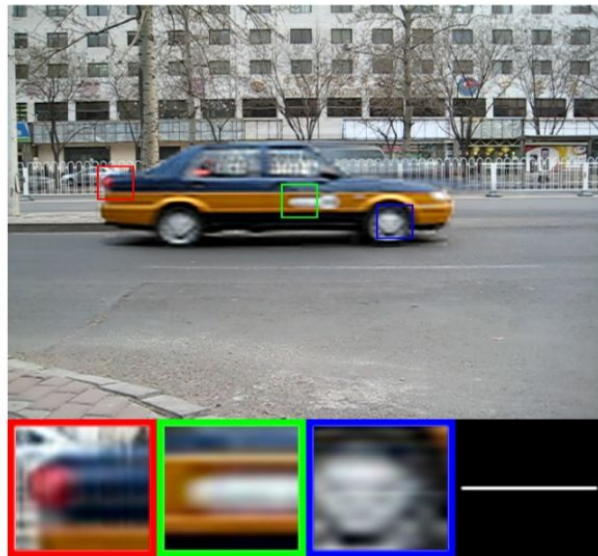
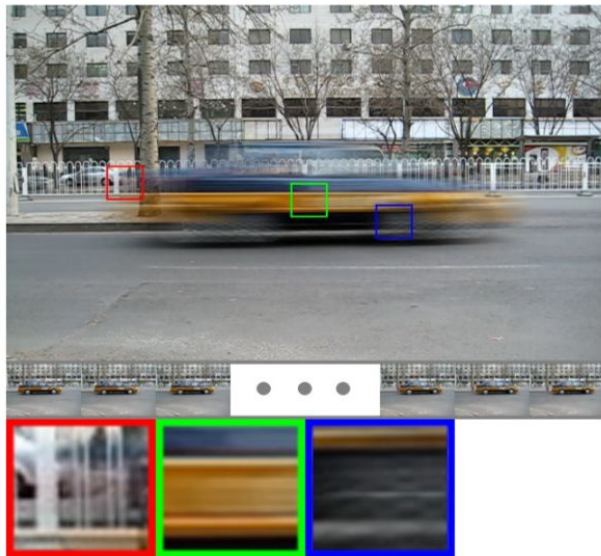
Application

Image Enhancement



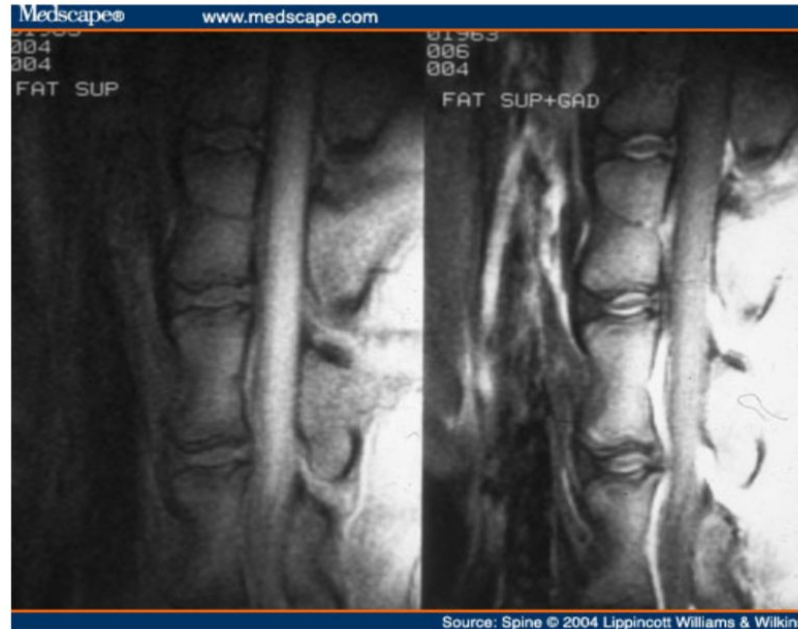
Application

Motion Deblurring



Application

Contrast Enhancement



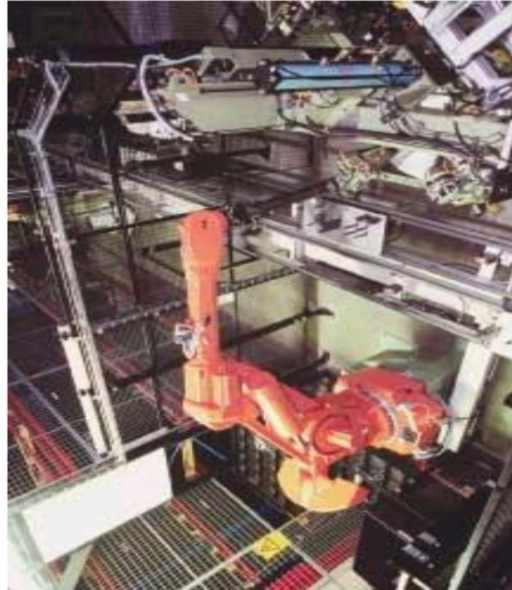
Application

Boundary Detection



Machine Vision

Vision-guided robots assemble wheel parts



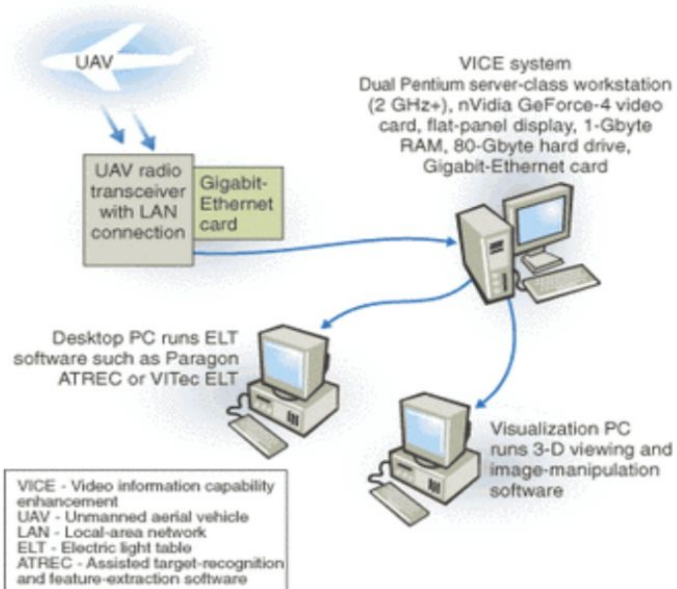
Machine Vision

Vision system grades moving food products



Machine Vision

Airborne imager tracks targets

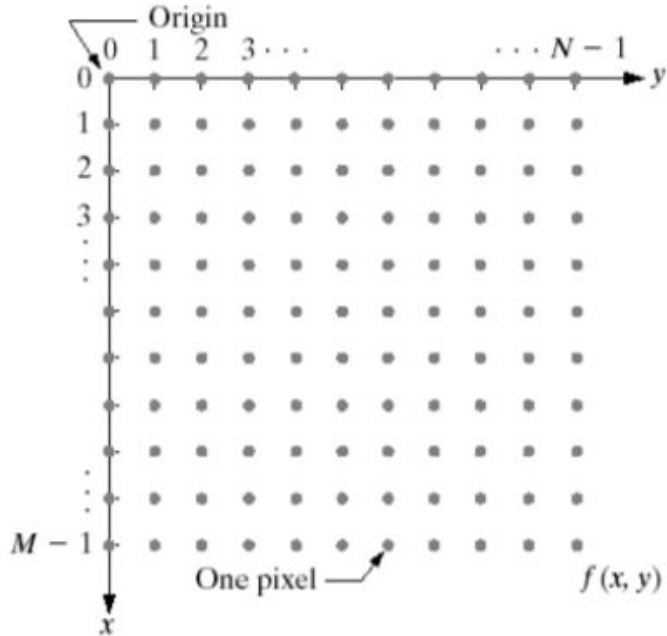


Machine Vision

Smart Vehicles



Image Representation



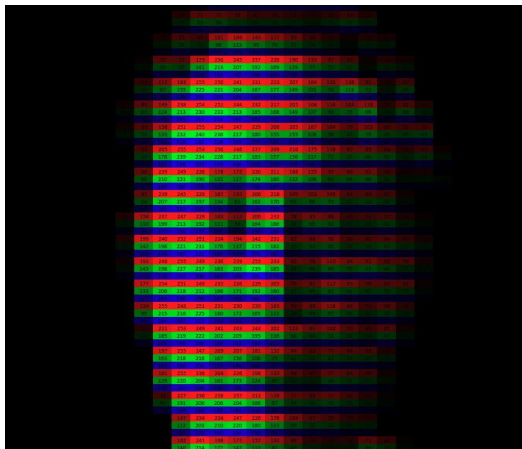
นิยาม

- “ภาพ” (Image) : ฟังก์ชันสองมิติ $f(x,y)$ โดยที่
 - x และ y เป็นพิกัดเชิงพื้นที่ (spatial coordinates)
 - ขนาดของฟังก์ชัน f คือ “ความเข้มของจุดภาพ” (intensity) หรือ gray level ของจุด x,y นี้
- “ภาพดิจิทัล” (Digital Image) : x,y,f are finite and discrete

Image Data



หากต้องการนำข้อมูลประเภท “ภาพ” มาใช้ในการประมวลผล ข้อมูลภาพจะถูกเก็บอยู่ในรูปแบบของ numpy array

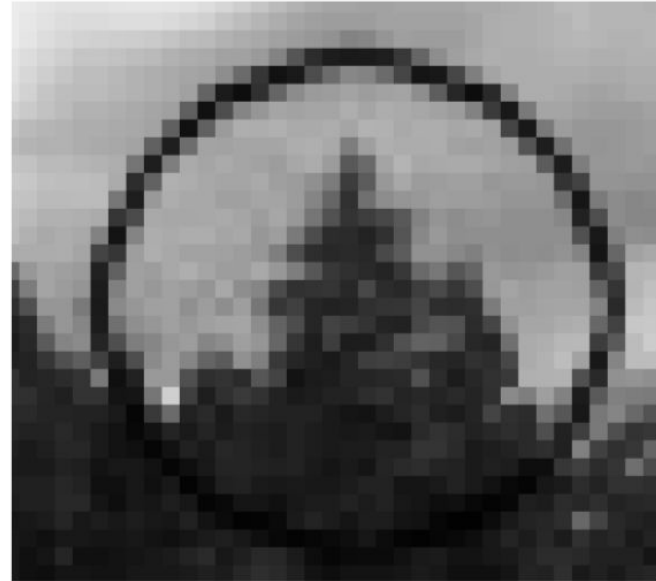


`img[: , : , :]`

ROW COLUMN CHANNEL

โดยทั่วไปค่าของสีในแต่ละ pixels จะถูกเก็บเป็นจำนวนเต็ม ที่มีค่าอยู่ระหว่าง 0 - 255

Picture Element (Pixel)



Spatial Resolution

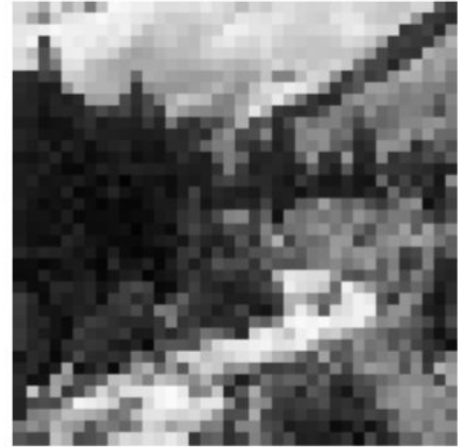
- A measure of the smallest discernible detail in an image



192x192



96x96



38x38

Image Cropping



บน

ล่าง

ซ้าย

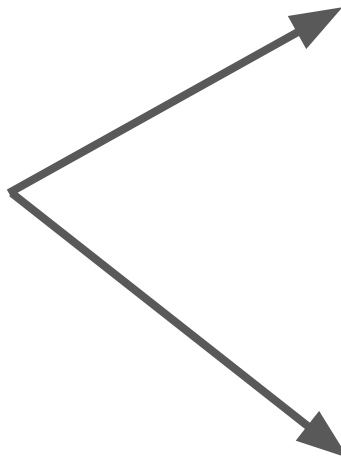
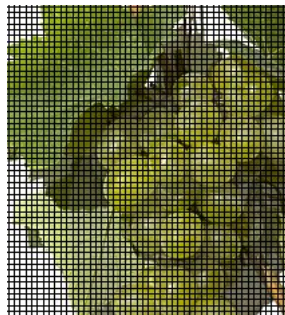
ขวา

```
img[10:230, 30:1200, :]
```

เราสามารถเลือก Crop ภาพได้ตามที่ต้องการ

โดยกำหนดจุดเริ่มต้นและสิ้นสุดของ Row และ Column

Image Scaling



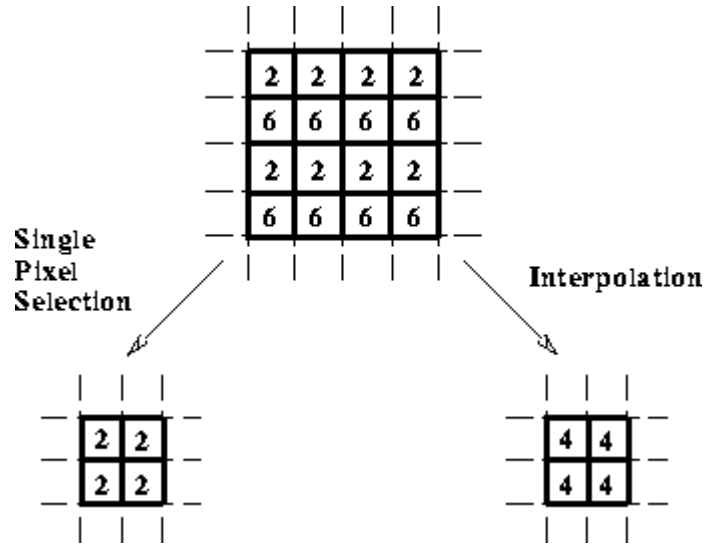
Scale Up 200%



Scale down 50%

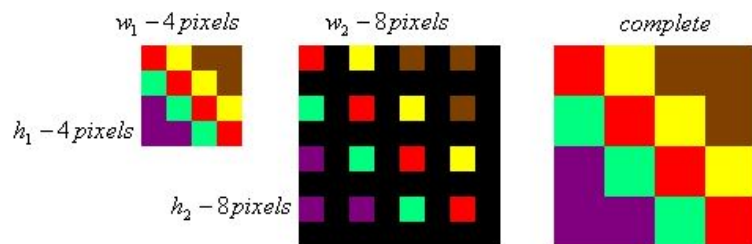
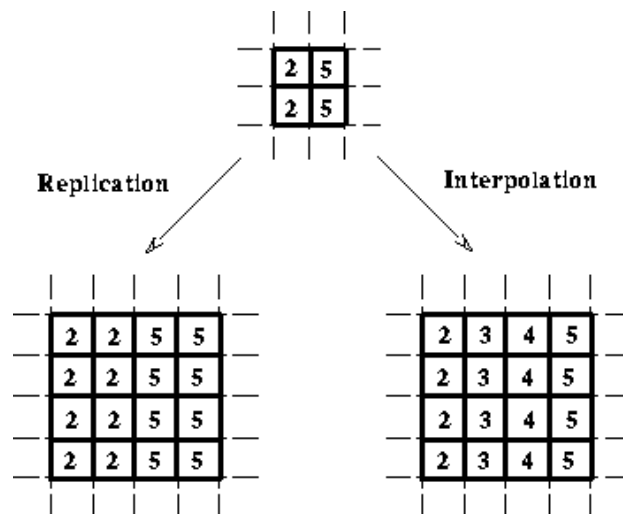


Scale-Down



In workshop we will use Interpolation algorithm by `convolve2d`.

Scale-Up



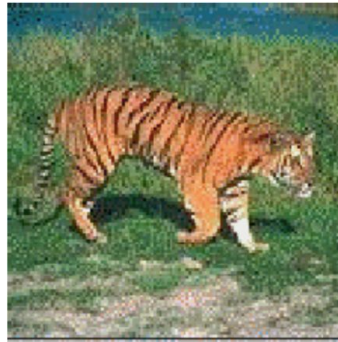
In work shop we will use Replication

Colors

- Our vision and action are influenced by an abundance of geometry and color information.
 - Traffic light
 - Search for a car in the parking lot.
- In the past, color image processing was limited, it has gained importance in recent years.

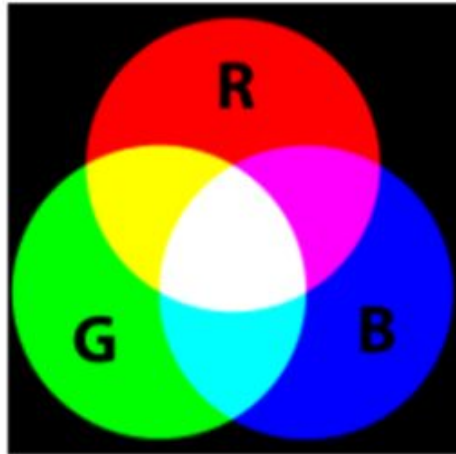
Colors Perception

- Color perception depends on both
 - the physics of light
 - complex processing by the eye-brain to integrate stimulus' properties with experience

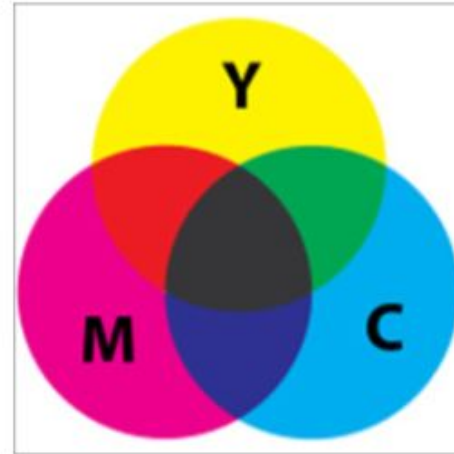


Pictures from Shapiro and Stockman's book

Colors Systems



Additive Color Mixture



Subtractive Color Mixture

Colors Image

ภาพสีทั่วไปจะเกิดจากการผสมสีระหว่าง R(แดง) G(เขียว) และ B(น้ำเงิน)
ในค่าที่แตกต่างกันจนเกิดเป็นภาพ



`img[: , : , :]`

- 0 หมายถึง สีแดง
- 1 หมายถึง สีเขียว
- 2 หมายถึง สีน้ำเงิน

Greyscale

นอกจากภาพสีแล้วก็ยังมีภาพขาวดำ ซึ่งมีลักษณะการเก็บข้อมูลคล้ายๆ ภาพสี กล่าวคือเป็น Numpy array เหมือนกัน เพียงแต่ Channel สีที่เพียงแค่ช่องเดียว เรียกว่าภาพ Greyscale



สำหรับค่าสีของภาพ Greyscale

0 คือ สีขาว

1 - 254 คือ สีเทาไล่ระดับตามความเข้ม

255 คือ สีดำ

Greyscale

การแปลงภาพสีให้เป็นภาพขาวดำด้วยวิธีการเฉลี่ยสี



เป็นการแปลงภาพสีที่มี 3 channel ให้เหลือเพียง 1 channel

$$\text{Gray}[i] = (R[i] + G[i] + B[i]) / 3$$

Greyscale

การแปลงภาพสีให้เป็นภาพขาวดำด้วย
วิธีการคูณด้วยค่าคงที่เฉพาะตัว



การแปลงภาพสีเป็นภาพขาว-ดำด้วยวิธีการเฉลี่ยค่าสี
เป็นวิธีที่อาจจะทำให้เกิดความคลาดเคลื่อนของสีที่แสดงผลได้
(อาจไม่เห็นความแตกต่างมากนัก)

วิธีที่ทำให้ได้ค่าสีที่ตรงคือการคูณด้วยค่าคงที่แล้วหาผลรวม

weights = [0.2989, 0.5870, 0.1140]

R **G** **B**

Image Negative

- ภาพลบ (negative image) ที่มีสเกลสีเทาอยู่ระหว่างค่า $[0, L-1]$ หาได้โดยใช้ negative transformation function

$$s = L - 1 - r$$



Sepia



เป็นการ process ภาพให้ภาพธรรมดาของเรา
ให้เป็นภาพสี Sepia ซึ่งทำให้อารมณ์ภาพดูเหมือนภาพเก่าๆ

$$R' = \text{np.minimum}(1.0, 0.393R + 0.769G + 0.189B)$$

$$G' = \text{np.minimum}(1.0, 0.349R + 0.686G + 0.168B)$$

$$B' = \text{np.minimum}(1.0, 0.272R + 0.534G + 0.131B)$$

Convolved

การทำคอนโวลูชัน (convolution)

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

- โดยที่ $w(x, y)$ คือ ตัวพรางขนาด $m \times n$

$f(x, y)$ คือ ภาพนำเข้า

$$a=(m-1)/2 \text{ และ } b=(n-1)/2$$

Convolved

ตัวพราง (Mask / Kernel)

- A **mask** is a set of pixel positions and corresponding values called **weights**.
- Each mask has an **origin**.

1	1	1
1	1	1
1	1	1

1	2	1
2	4	2
1	2	1

1
1
1

- Applying a mask on a $M \times N$ image (I) yields a $M \times N$ output image (J).

Convolved

INPUTS

R - Red channel

0	0	2	0	2
0	2	2	1	1
0	2	1	2	1
0	2	1	1	0
2	0	2	0	0

FILTERS ("MAGNIFYING GLASSES")

Weights 1

Red to feature map 1

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

CONV 1

From Red

-1	4	2
2	4	4
4	3	7

-1

$$\begin{aligned}
 &= (0 \times 0) + (0 \times 1) + (2 \times 1) + \\
 &\quad (0 \times 1) + (2 \times 1) + (2 \times -1) + \\
 &\quad (0 \times 1) + (2 \times -1) + (1 \times -1)
 \end{aligned}
 \quad
 \begin{aligned}
 &= 0 + 0 + 2 + \\
 &\quad 0 + 2 + (-2) + \\
 &\quad 0 + (-2) + (-1)
 \end{aligned}
 \quad
 \begin{aligned}
 &= 2 + \\
 &\quad 0 + \\
 &\quad -3
 \end{aligned}$$

9 multiplications for 1 layer

R - Red channel

1	2	3		
4	5	6		
7	8	9		

Weights 1

Red to feature map 1

A	B	C
D	E	F
G	H	I

From Red

?		

?

$$\begin{aligned}
 &= (1 \times A) + (2 \times B) + (3 \times C) + \\
 &\quad (4 \times D) + (5 \times E) + (6 \times F) + \\
 &\quad (7 \times G) + (8 \times H) + (9 \times I)
 \end{aligned}$$

9 multiplications for 1 layer

Applying Masks to Images

1	2	1
2	4	2
1	2	1

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

1	2	1
2	4	2
1	2	1

Input image I

Mask

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Apply mask to every pixels,
e.g., at $I(0,0)$

$$\begin{aligned}
 J(0,0) &= 40*1 + 40*2 + 80*1 \\
 &+ 40*2 + 40*4 + 80*2 + 40*1 \\
 &+ 40*2 + 80*1 = \mathbf{800}
 \end{aligned}$$

800	1120	1280
800	1120	1280
800	1120	1280

Temporary
image J

INPUTS

(Height x Width) x Depth (RGB)

(5 x 5) x 3

Stride = # of pixels to move glass

Stride = 1

No padding

Start

R - Red channel

0	0	2	0	2
0	2	2	1	1
0	2	1	2	1
0	2	1	1	0
2	0	2	0	0

G - Green channel

0	2	2	1	0
1	1	1	0	0
0	0	0	0	1
2	1	1	0	0
2	2	1	0	2

B - Blue channel

0	2	1	1	0
1	2	0	2	1
2	1	0	0	1
1	1	1	2	1
1	2	1	2	1

Applying Masks to Images with padding

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image *I*

1	2	1
2	4	2
1	2	1

Mask

40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80

Virtual image

Create a virtual image by expand top&bottom rows and left&right columns

Padding can be any value depend on each task.

INPUTS		
(Height x Width) x Depth (RGB)		
(5 x 5) x 3		
Stride = # of pixels to move glass		
Stride = 1		
Zero padding		

1. Start

R - Red channel

0	0	0	0	0	0	0
0	2	2	2	2	2	0
0	2	2	2	1	1	0
0	2	1	2	1	1	0
0	2	1	1	1	0	0
0	2	0	2	0	0	0
0	0	0	0	0	0	0

G - Green channel

0	0	0	0	0	0	0
0	2	2	1	0	0	0
0	1	1	1	0	0	0
0	0	0	0	0	1	0
0	2	1	1	0	0	0
0	2	2	1	0	2	0
0	0	0	0	0	0	0

B - Blue channel

0	0	0	0	0	0	0
0	2	2	1	1	0	0
0	1	2	0	2	1	0
0	2	1	0	0	1	0
0	1	1	1	2	1	0
0	1	2	1	2	1	0
0	0	0	0	0	0	0

Normalization

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image I

1	2	1
2	4	2
1	2	1

Mask

640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280

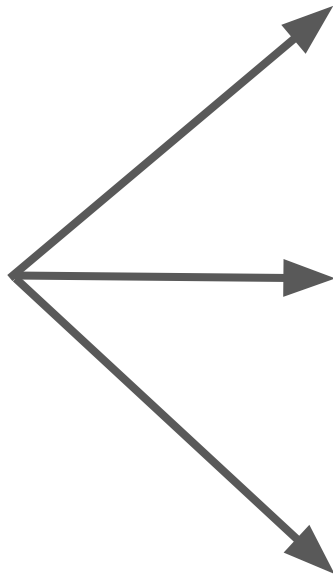
Temporary
image

**Normalize the
image J by dividing
by sum of the weights
(16) to obtain J'**

40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80

Normalized
Output image J'

Convolved Output



with kernel
 $\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$



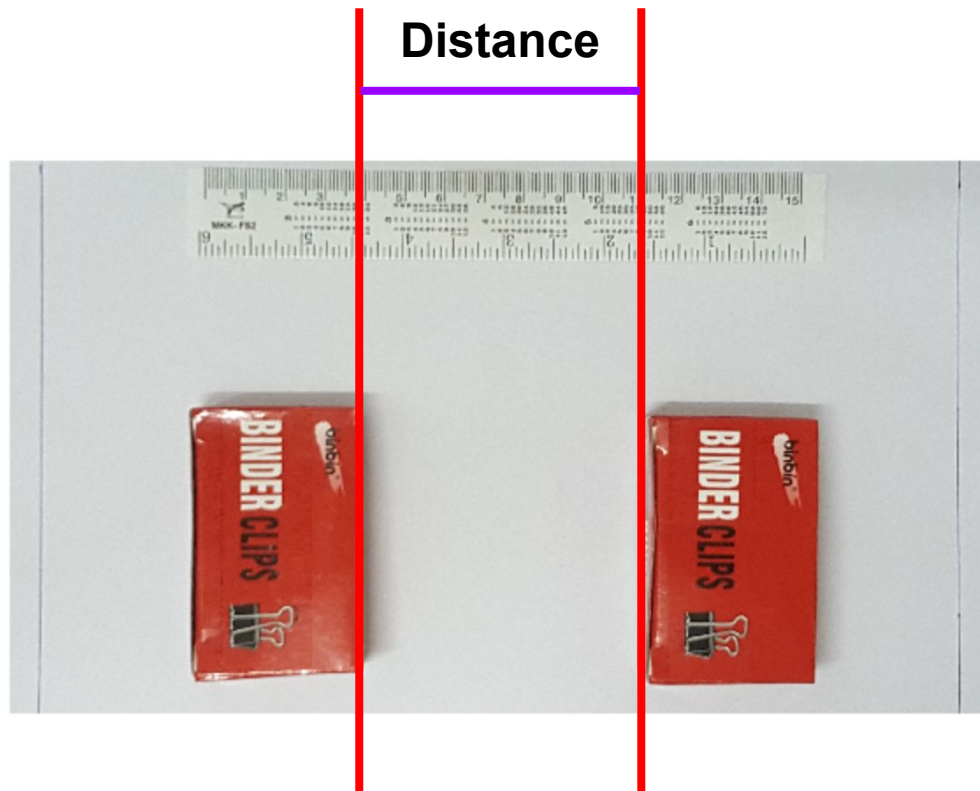
with kernel
 $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



with kernel
 $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
with negative input

Workshop

Find Distance : การหาระยะห่างระหว่างกล่อง



เปิดไฟล์ได้จาก Link ด้านล่าง

shorturl.at/dozEH

จะพบกันหน้าตาแบบนี้

ยังไม่สามารถแก้ไขได้
สังเกตได้ว่าเป็น Viewing อยู่

01-image_processing_with_numpy X +

← → ↻ 🔒 https://colab.research.google.com/drive/1_5jlbMztftqYIMAHIDHxWqU_G6S_JfH7fbcldid=hwAR3yPLV1CD3jvs6-evqGSN2mX0FFVNWAZKZCp40xgUR50c8WopHtspH6c ☆ 🔍

co 01-image_processing_with_numpy_student.ipynb ☆

File Edit View Insert Runtime Tools Help

OPEN IN PLAYGROUND

Table of contents Code snippets Files X

Workshop 4 : Image processing with Numpy

- Read and Crop
- Colours
- To Greyscale
- Image Negatives
- Sepla Image
- Convolved
- Test01
- Test02
- Image Scalling
- Scale-Up (x 2)
- Find Distance : การหาระยะทางระหว่างอนบิตร์
- Crop Ruler
- Convert to Greyscale
- Lab 2 : Universal distance measurement

Workshop 4 : Image processing with Numpy

ในภาษา Python มี library สำหรับจัดการรูปภาพมากมายเช่น OpenCV, SciKit-Image และ Pillow เป็นต้น ซึ่งจะช่วยให้เราจัดการกับรูปภาพในงานต่างๆได้เป็นอย่างดี
อย่างไรก็ตาม เป้าหมายของวีรกรรมนี้ต้องการให้เราได้ศึกษาพื้นฐานเทคนิคในการทำ image processing ด้วยเหตุนี้เราจึงเลือกใช้ numpy ในการจัดการกับรูปภาพเพราะเป็นเรื่องที่ง่าย
ได้ศึกษาแล้วในขั้นเขียน และใช้ matplotlib ในการนำเข้าและแสดงผลรูปภาพ

```
[ ] import numpy as np
import matplotlib.pyplot as plt
```

```
[ ] # Download File

!wget "https://github.com/5730279821-TA/workshop4_Resource/raw/master/hs4_resource.zip"
```

```
[ ] # Extract File

import zipfile
with zipfile.ZipFile("hs4_resource.zip","r") as zip_ref:
    zip_ref.extractall("")
```

Read and Crop

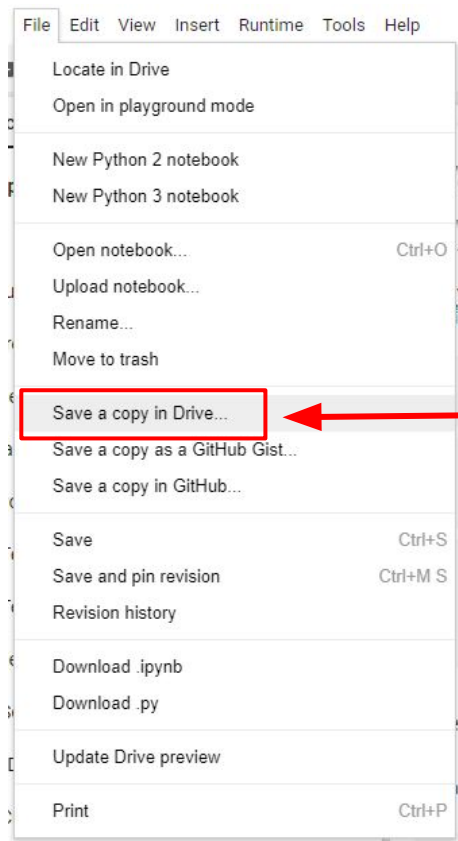
โหลดรูปภาพ "elon_musk.jpg"

```
[ ] im = plt.imread("elon_musk.jpg")
im.shape
```

จะเห็นว่าภาพได้ถูกโหลดเข้ามาใน array ที่มี dimension เป็น 451 x 300 x 3
ซึ่ง 2 ค่าแรก (451 และ 300) แสดงถึง pixel ในแกน Y (ความสูง) และ X(ความกว้าง) ตามลำดับ และค่าสุดท้าย (3) แสดงถึงค่าสี RGB ในภาพนั้น
ต่อมาเราจะมาดูว่าหน้าตาของภาพนั้นเป็นอย่างไร

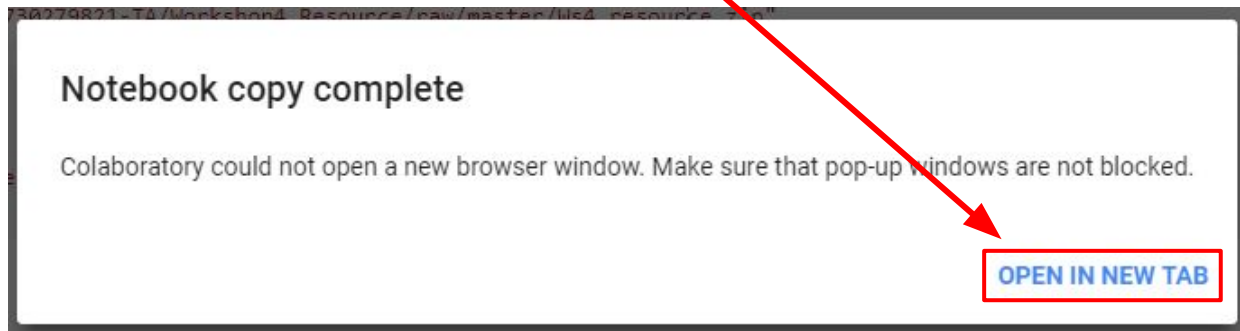
```
[ ] plt.figure(num=None, figsize=(5, 4))
plt.imshow(im)
plt.axis('off')
plt.show()
```

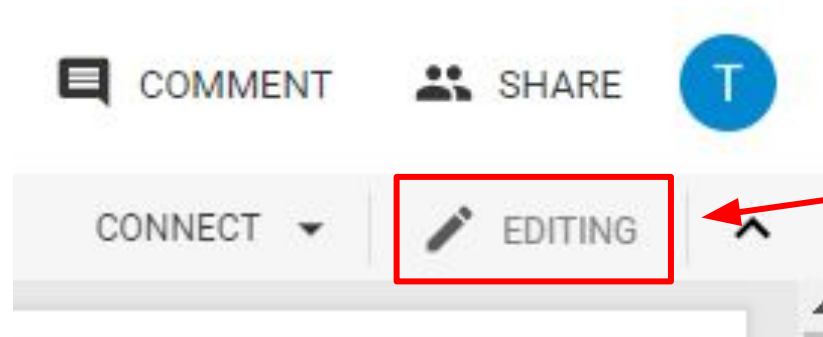
ภาพข้างต้นเป็นรูปของ Elon Musk ซีอีโอของบริษัท SpaceX
ถ้าหากว่าเราต้องการแค่บางส่วนของภาพของอย่างเช่นแค่ "ใบหน้าของ Elon Musk"



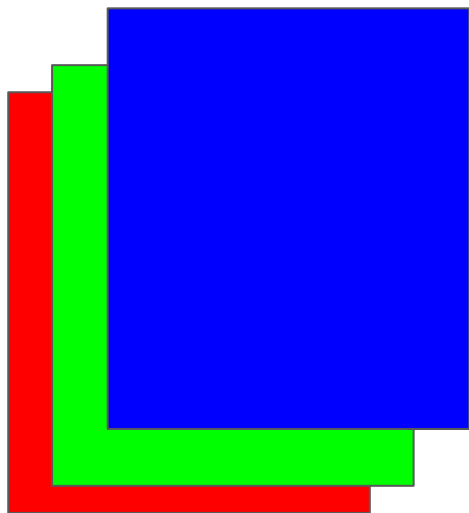
ไปที่ File เลือก save a copy in Drive...

เลือก OPEN IN NEW TAB

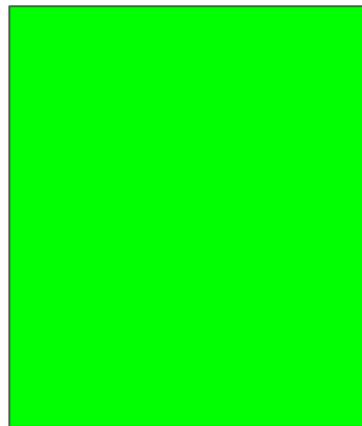




สังเกตว่าเราสามารถ
แก้ไขไฟล์ได้แล้ว



`img[:, :, 1] ->>>`



`imshow` มองเป็นขาวดำ



`imshow` มองภาพสี