WorkShop 8 : ImageProcessing

Dr. Ekapol Chuangsuwanich

2110101 ComProg SEC 5

Some part copy from
2110433 Computer Vision

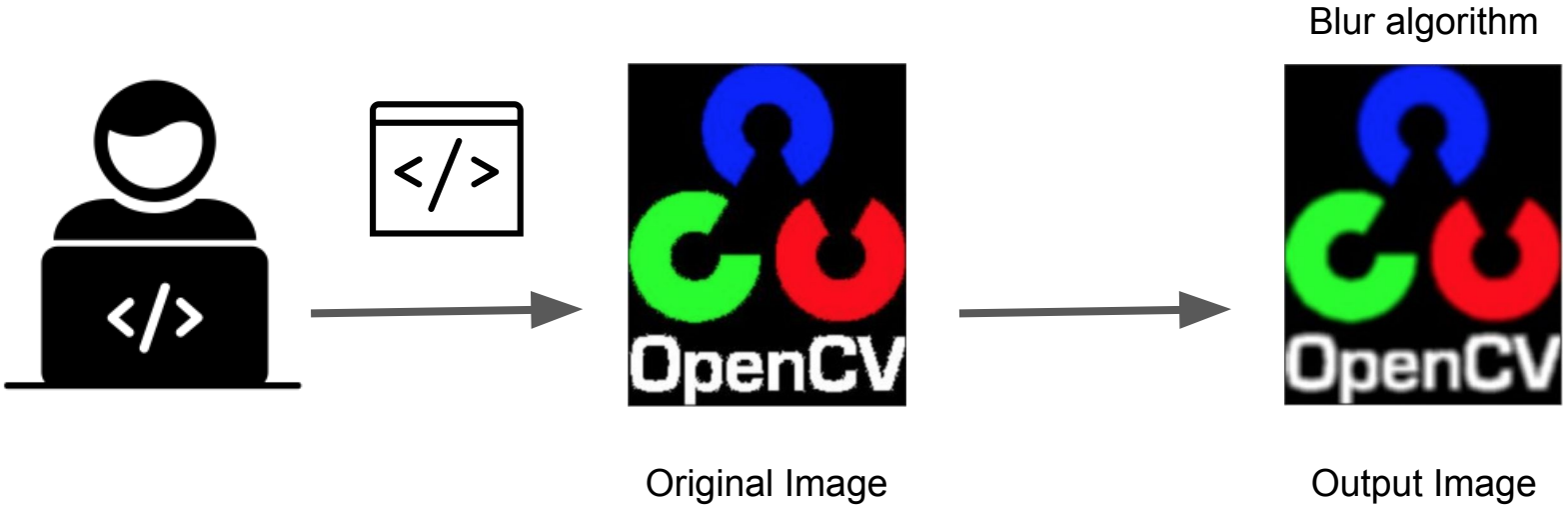
TODAY

Our Topics

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

Digital Image Processing

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



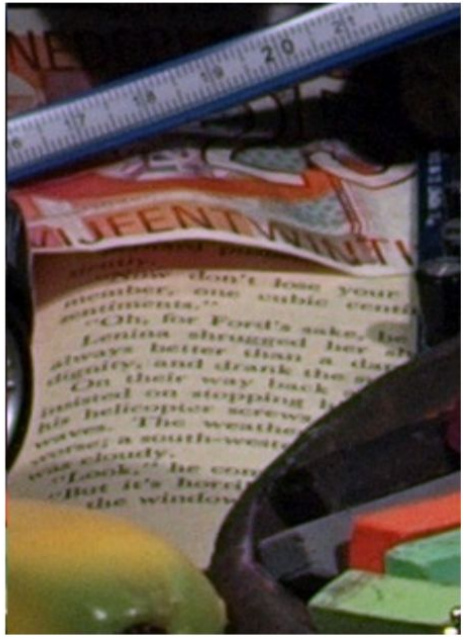
Digital Image Processing

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.

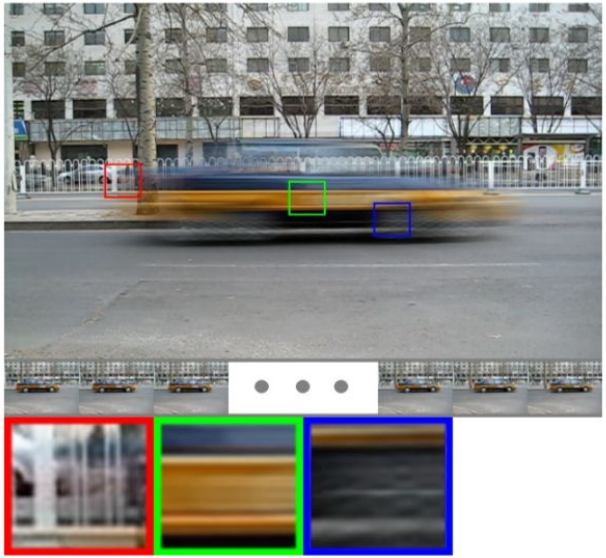
Application

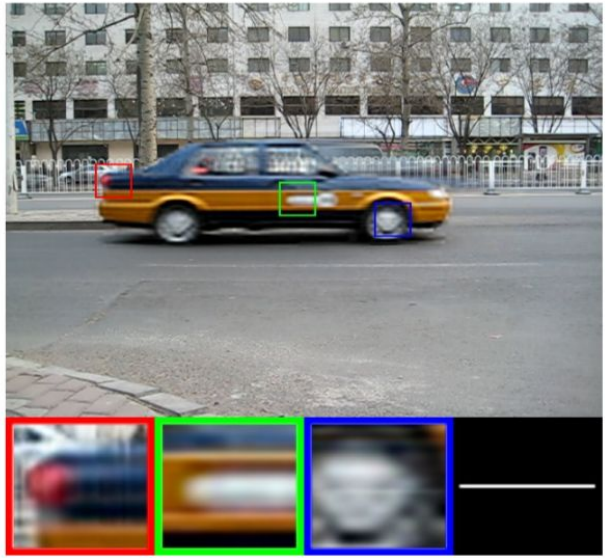
Image Enhancement



Application

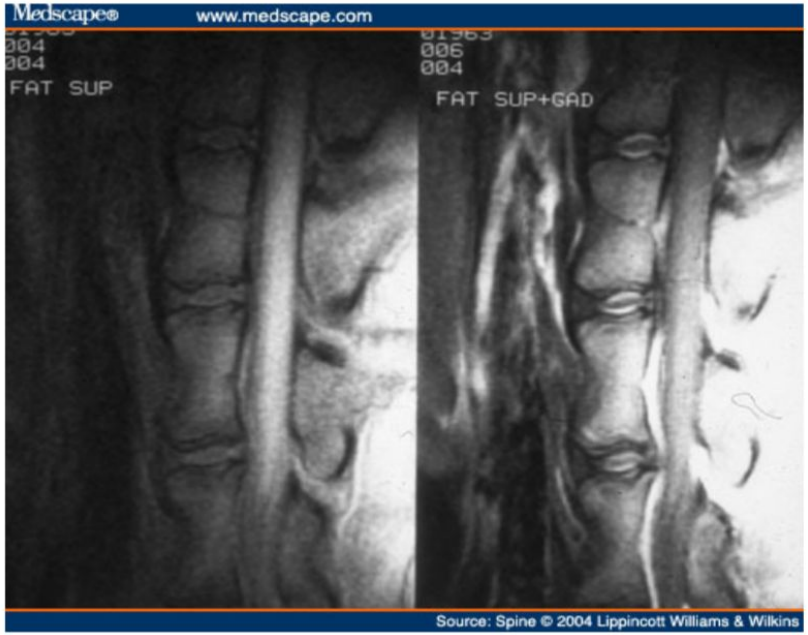
Motion Deblurring





Application

Contrast Enhancement

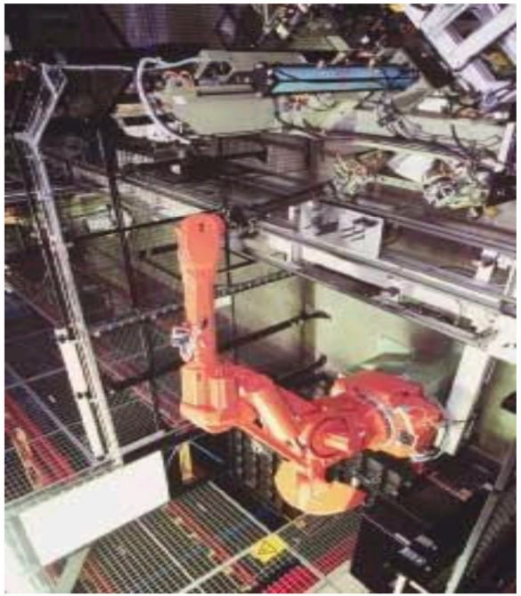


Application

Boundary Detection

Machine Vision

Vision-guided robots assemble wheel parts

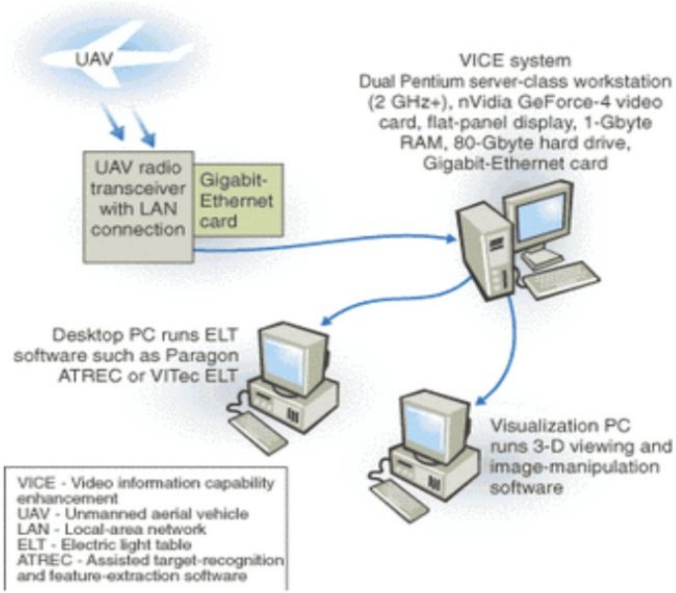


Machine Vision

Vision system grades moving food products

Machine Vision

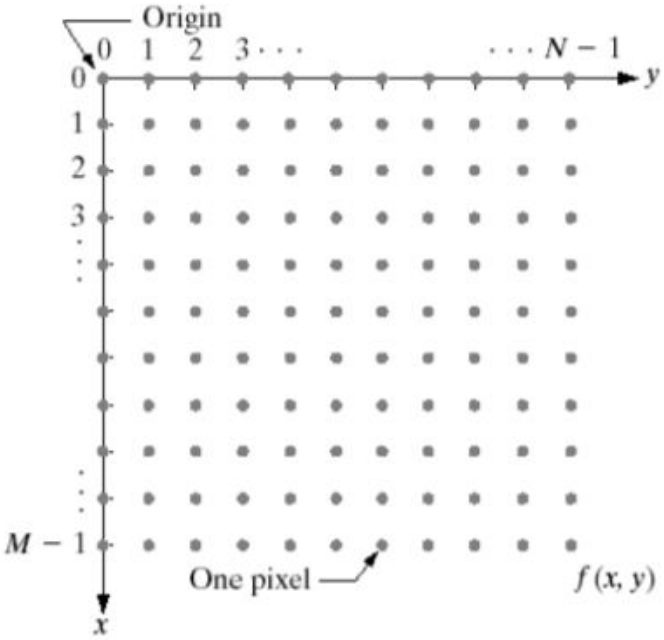
Airborne imager tracks targets



Machine Vision

Smart Vehicles

Image Representation



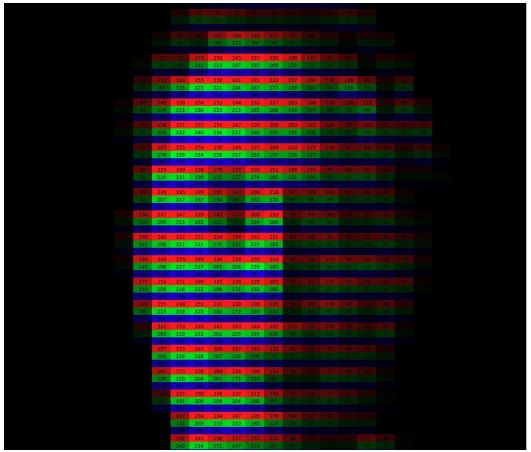
นิยาม

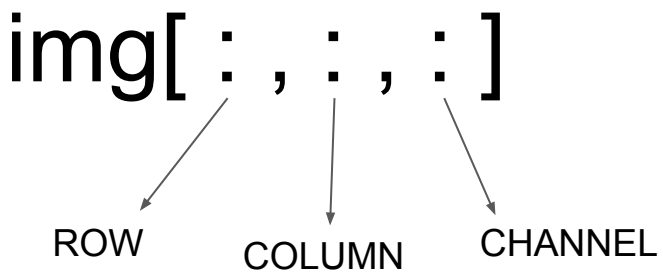
- “ภาพ” (Image) : ฟังก์ชันสองมิติ f(x,y) โดยที่
  - x และ y เป็นพิกัดเชิงพื้นที่ (spatial cooordinates)
  - ขนาดของฟังก์ชัน f คือ “ความเข้มของจุดภาพ” (intensity) หรือ gray level ของจุด x,y นี้

- “ภาพดิจิทัล” (Digital Image) : x,y,f are finite and discrete

Image Data

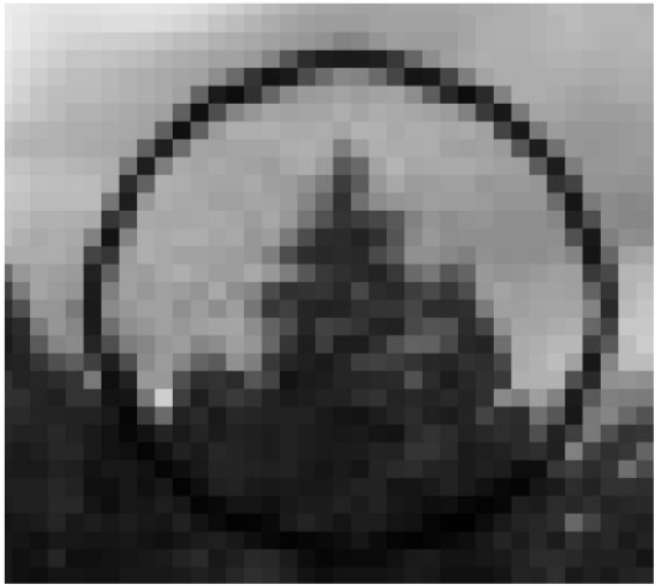
หากต้องการนำข้อมูลประเภท “ภาพ” มาใช้ในการประมวลผล
ข้อมูลภาพจะถูกเก็บอยู่ในรูปแบบของ numpy array





โดยทั่วไปค่าของสีในแต่ละ pixels จะถูกเก็บเป็นจำนวนเต็ม
ที่มีค่าอยู่ระหว่าง 0 - 255

Picture Element (Pixel)

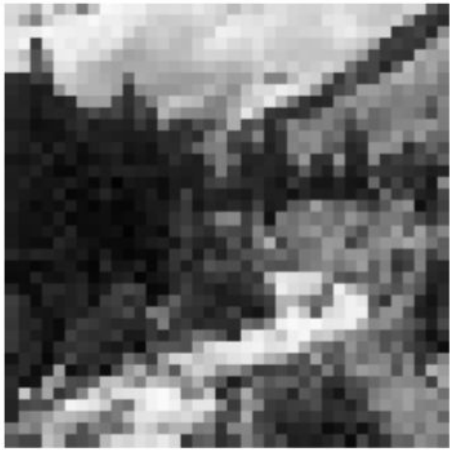


Spatial Resolution

- **A measure of the smallest discernible detail in an image**

192x192

96x96



38x38

Image Cropping

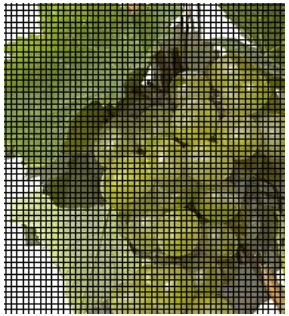
บน ล่าง ซ้าย ขวา

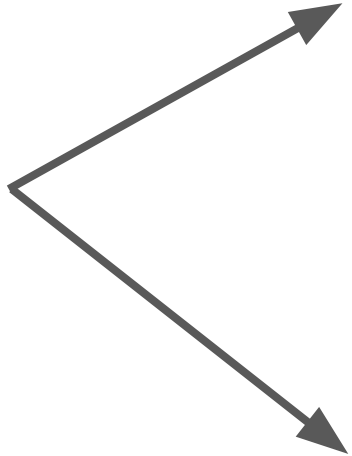
```
img[10:230,30:1200, :]
```

เราสามารถเลือก Crop ภาพได้ตามที่ต้องการ

โดยกำหนดจุดเริ่มต้นและสิ้นสุดของ Row และ Column

Image Scaling

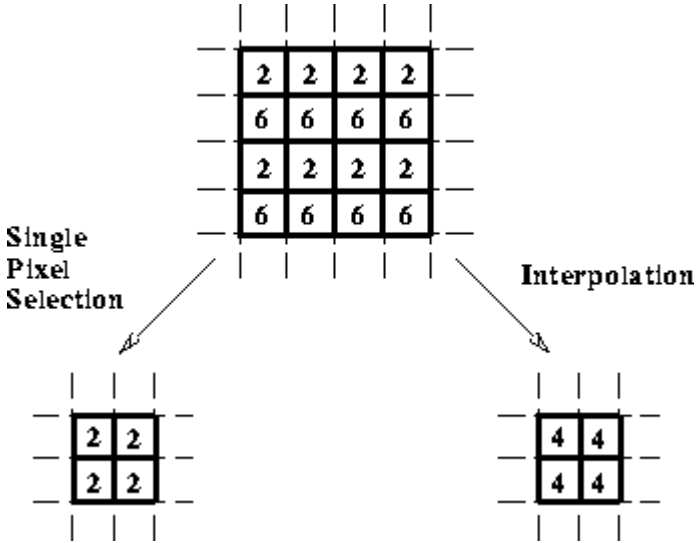




Scale Up 200%

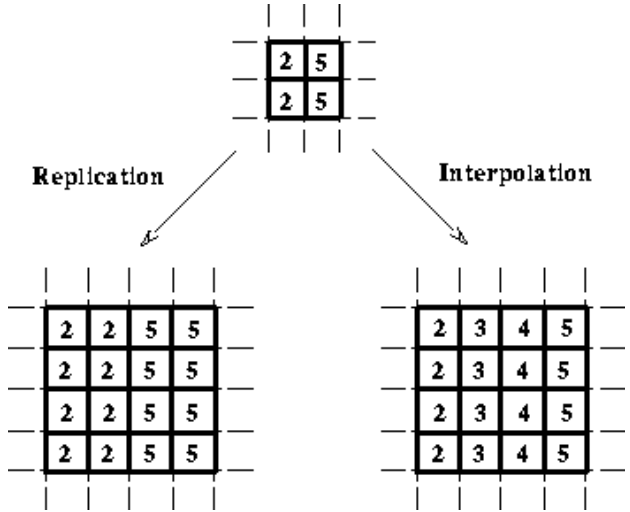
Scale down 50%

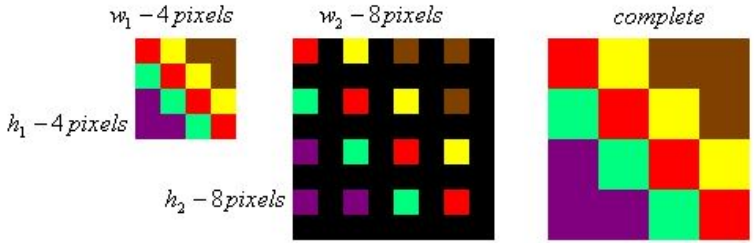
Scale-Down



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.

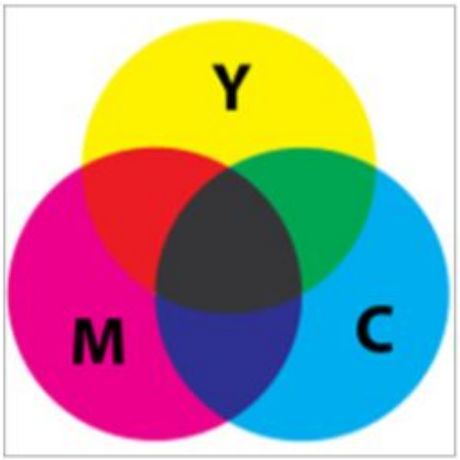
Colors Perception

- 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 สีทีเพียงแค่ช่องเดียว เรียกว่าภาพ Greyscale

สำหรับค่าสีของภาพ Greyscale

0 คือ สีขาว

1 - 254 คือ สีเทาไล่ระดับตามความเข้ม

255 คือ สีดำ

Greyscale

การแปลงภาพสีให้เป็นภาพขาวดำด้วยวิธีการเฉลี่ยสี

เป็นการแปลงภาพสีที่มี 3 channel ให้เหลือเพียง 1 channel

$$Gray[i] = (R[i] + G[i] + B[i]) / 3$$

Greyscale

การแปลงภาพสีให้เป็นภาพขาวดำด้วย
วิธีการคูณด้วยค่าคงที่เฉพาะตัว

การแปลงภาพสีเป็นภาพขาว-ดำด้วยวิธีการเฉลี่ยค่าสี
เป็นวิธีที่อาจจะทำให้เกิดความคลาดเคลื่อนของสีที่แสดงผลได้
(อาจไม่มีเห็นความแตกต่างมากนัก)

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' = np.minimum(1.0, 0.393R + 0.769G + 0.189B)

G' = np.minimum(1.0, 0.349R + 0.686G + 0.168B)

B' = np.minimum(1.0, 0.272R + 0.534G + 0.131B)

Convoluted

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 x n

  $f(x,y)$ คือ ภาพนำเข้า

  $a=(m-1)/2$ และ $b=(n-1)/2$

Convoluted

ตัวพราง (Mask / Kernel)

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

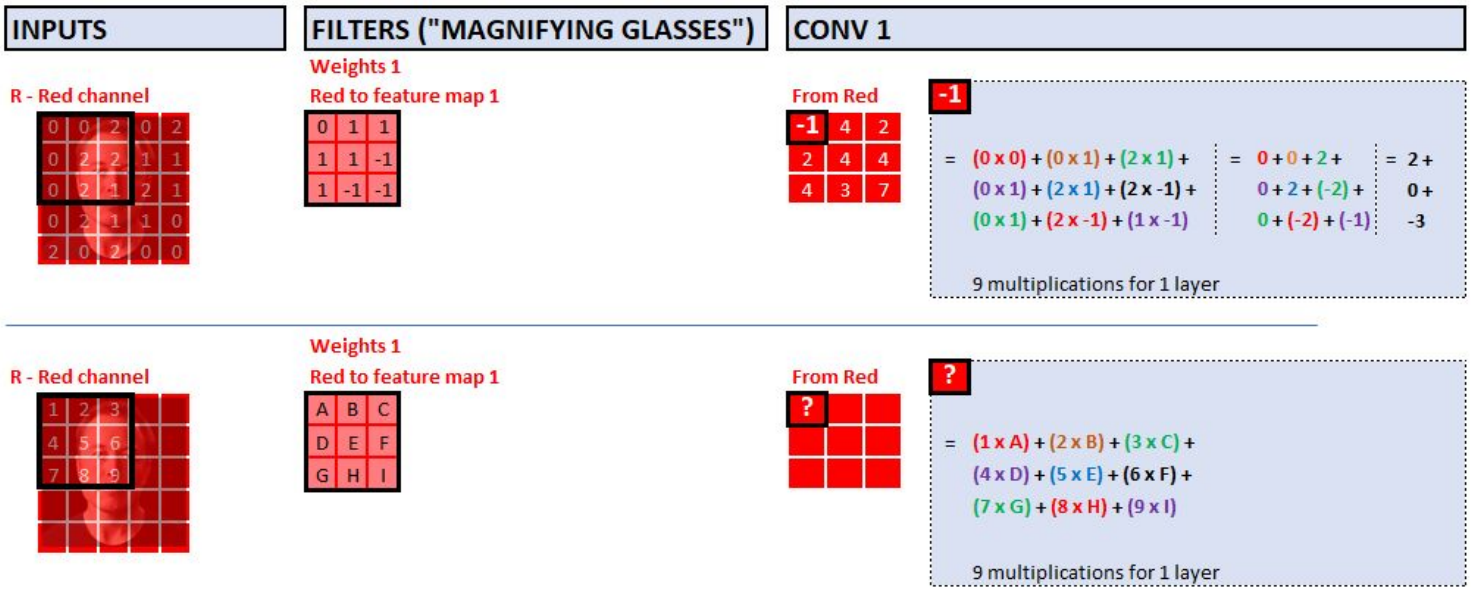
1	1	1
1	**1**	1
1	1	1

1	2	1
2	**4**	2
1	2	1

1
1
1

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

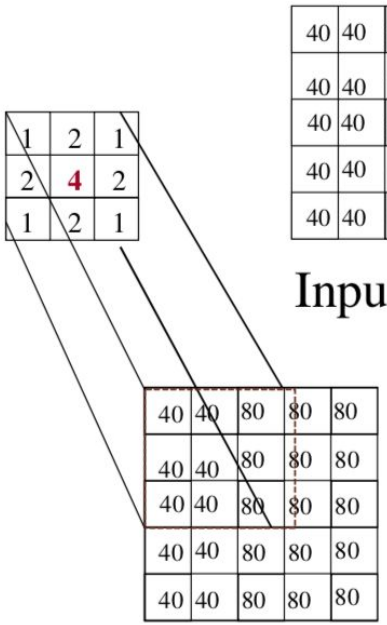
Convoluted


INPUTS
FILTERS ("MAGNIFYING GLASSES")
CONV 1
Weights 1
R - Red channel
Red to feature map 1
From Red
-1
= (0 x 0) + (0 x 1) + (2 x 1) + (0 x 1) + (2 x 1) + (2 x -1) + (0 x 1) + (2 x -1) + (1 x -1)
= 0 + 0 + 2 + 0 + 2 + (-2) + 0 + (-2) + (-1)
= 2 + 0 + -3
9 multiplications for 1 layer
Weights 1
R - Red channel
Red to feature map 1
From Red
?
= (1 x A) + (2 x B) + (3 x C) + (4 x D) + (5 x E) + (6 x F) + (7 x G) + (8 x H) + (9 x I)
9 multiplications for 1 layer

Applying Masks to Images

INPUTS
(Height x Width) x Depth (RGB)
(5 x 5) x 3
Stride = # of pixels to move glass
Stride = 1
No padding

Start



Input image ***I***

1	2	1
2	4	2
1	2	1

Mask

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

J(0,0) = 40*1 + 40*2 + 80*1 + 40*2 + 40*4 + 80*2 + 40*1 + 40*2 + 80*1 = **800**

800	1120	1280
800	1120	1280
800	1120	1280

Temporary image ***J***

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	0	0	2	0	2	0
0	0	2	2	1	1	0
0	0	2	1	2	1	0
0	0	2	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	0	2	2	1	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	0	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'***

Convoluted Output

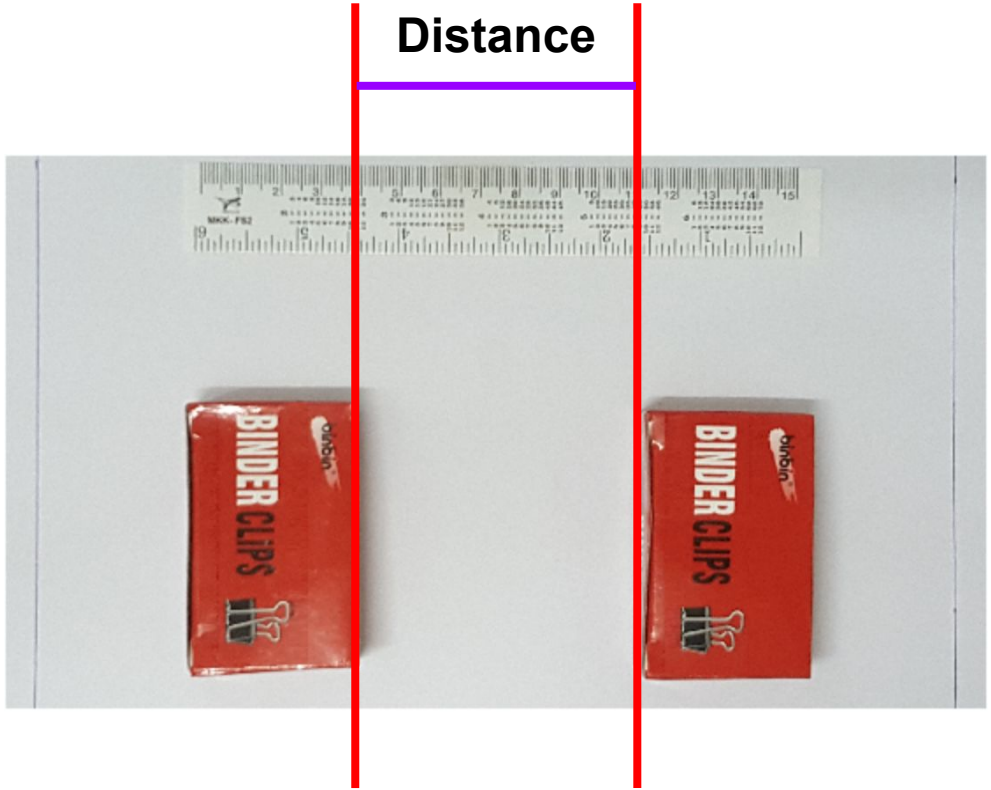
with kernel
[[1/9, 1/9, 1/9],
 [1/9, 1/9, 1/9],
 [1/9, 1/9, 1/9]]

with kernel
[[1, 1, 1],
 [1, -8, 1],
 [1, 1, 1]]

with kernel
[[1, 1, 1],
 [1, -8, 1],
 [1, 1, 1]]
with negative input

Workshop

Find Distance : การหาระยะห่างระหว่างกล่อง

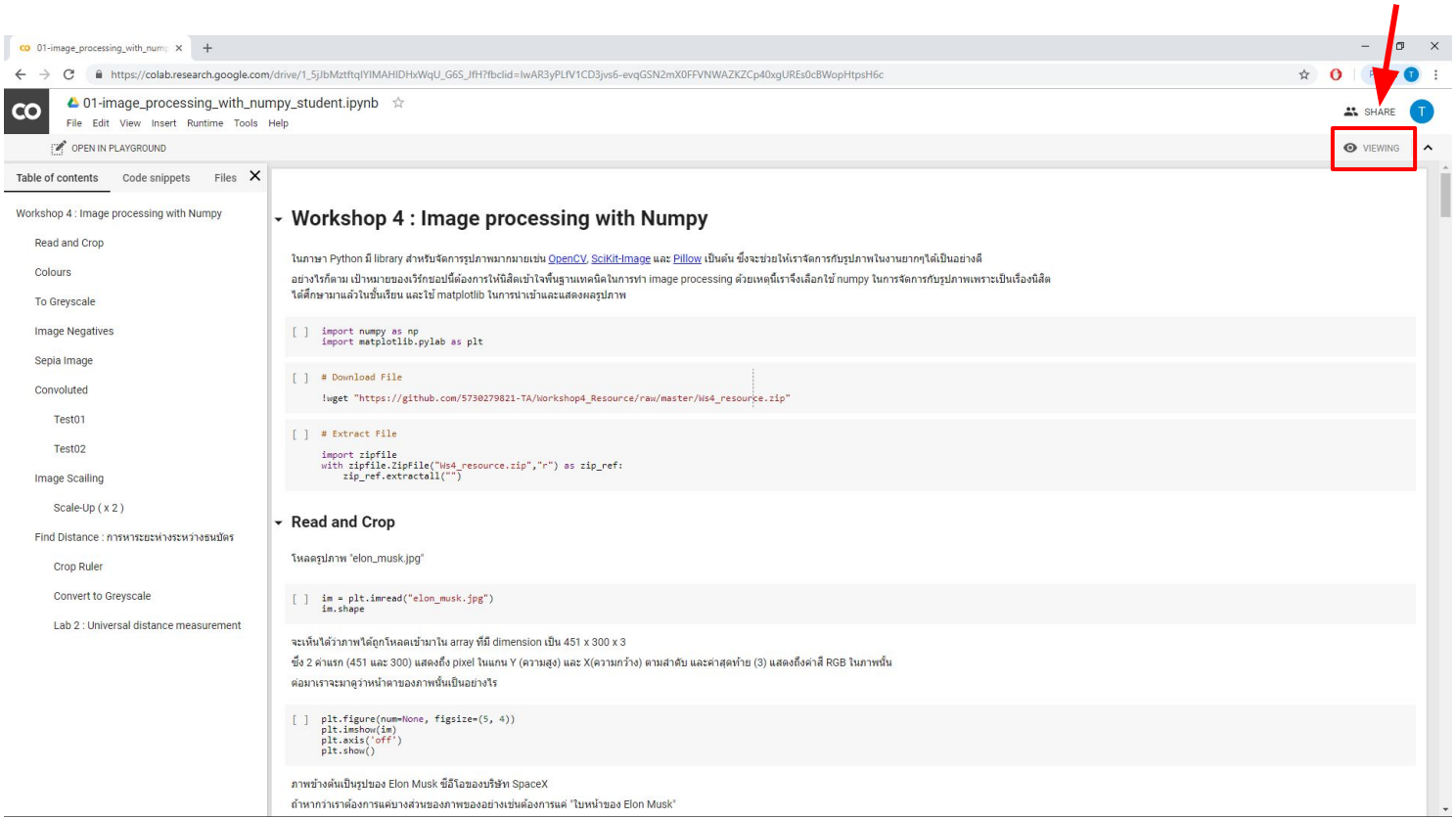


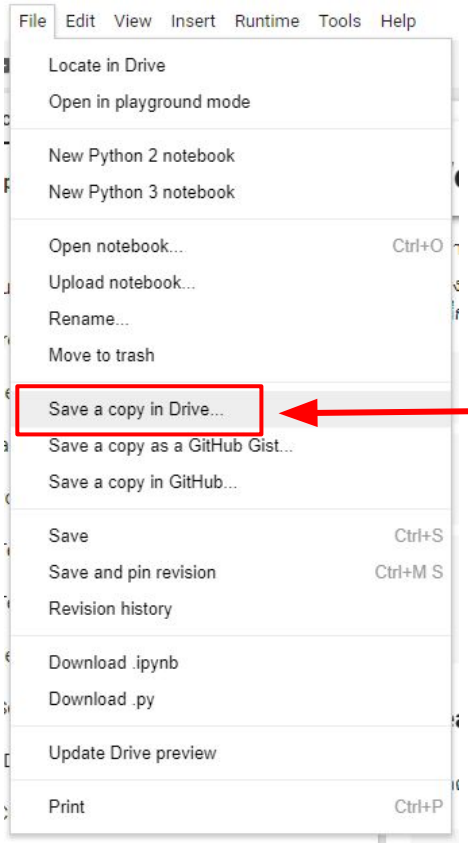
เปิดไฟล์ได้จาก Link ด้านล่าง

shorturl.at/dozEH

จะพบกันหน้าตาแบบนี้

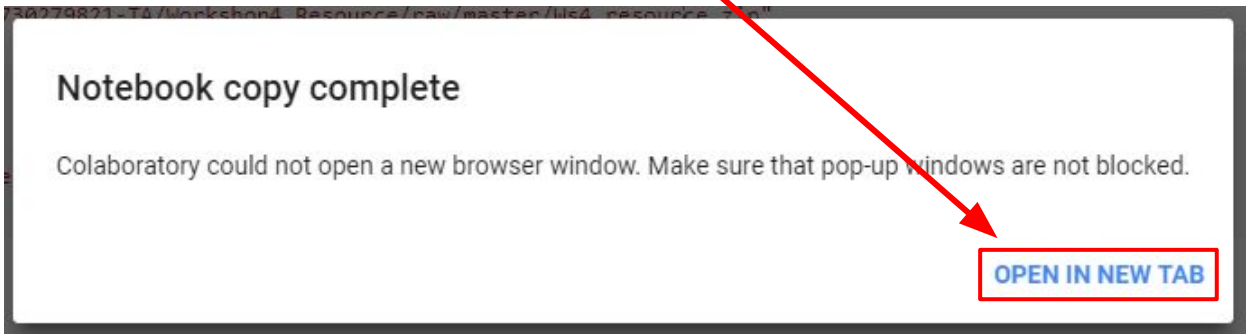
ยังไม่สามารถแก้ไขได้
สังเกตได้ว่ายังเป็น Viewing อยู่

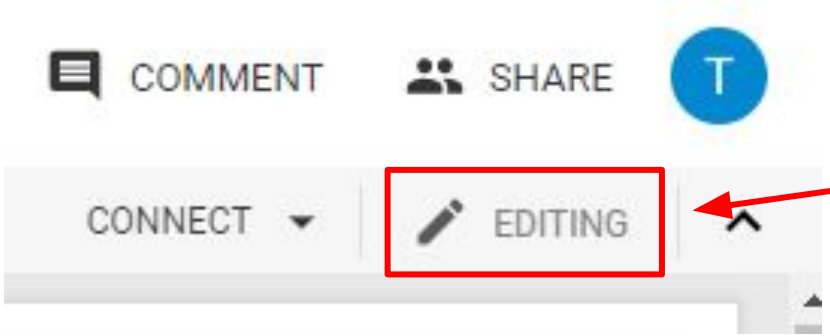




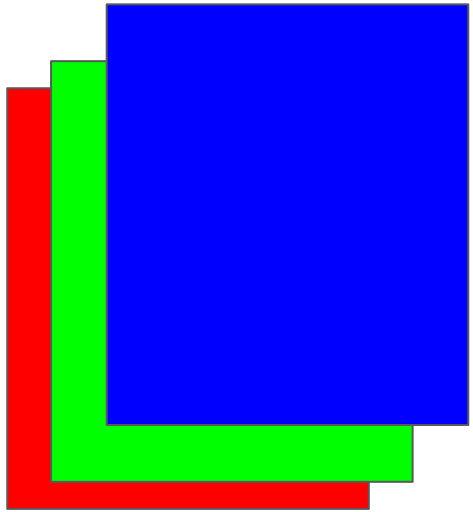
ไปที่ File เลือก save a copy in Drive...

เลือก OPEN IN NEW TAB

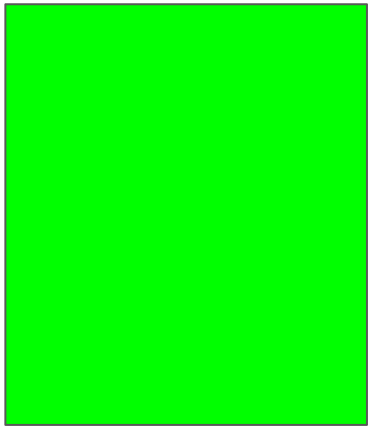




สังเกตว่าเราสามารถ
แก้ไขไฟล์ได้แล้ว



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



imshow มองเป็นขาวดำ

imshow มองภาพสี