Image Processing 1 – COMP4423 Computer Vision

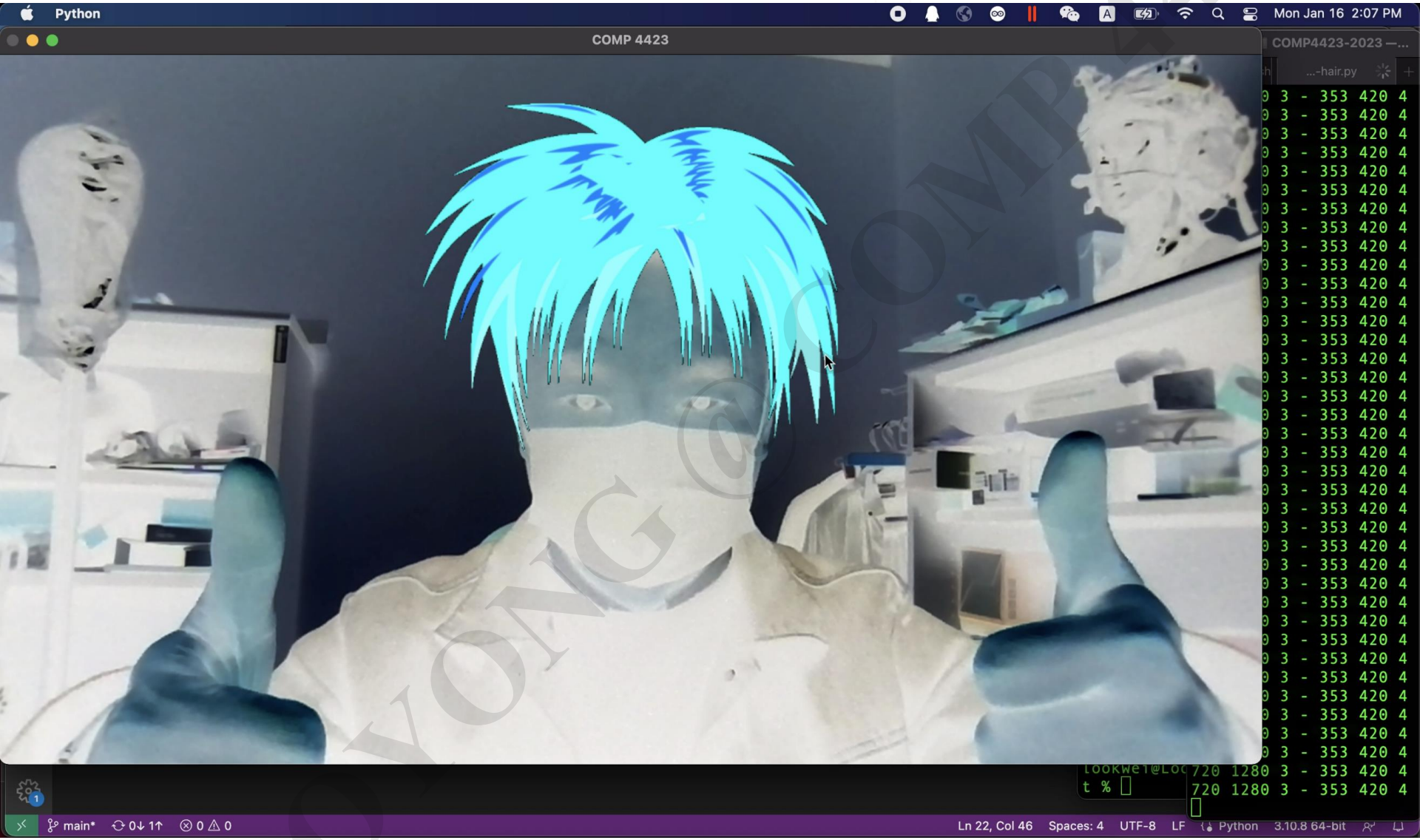
Xiaoyong Wei (魏驍勇)

x1wei@polyu.edu.hk

Department of Computing
電子計算學系

The Hong Kong Polytechnic University
香港理工大學

Opening Minds • Shaping the Future
啟迪思維 • 成就未來



Methods will be introduced in this lecture

Outline

>How Human/Computers see images

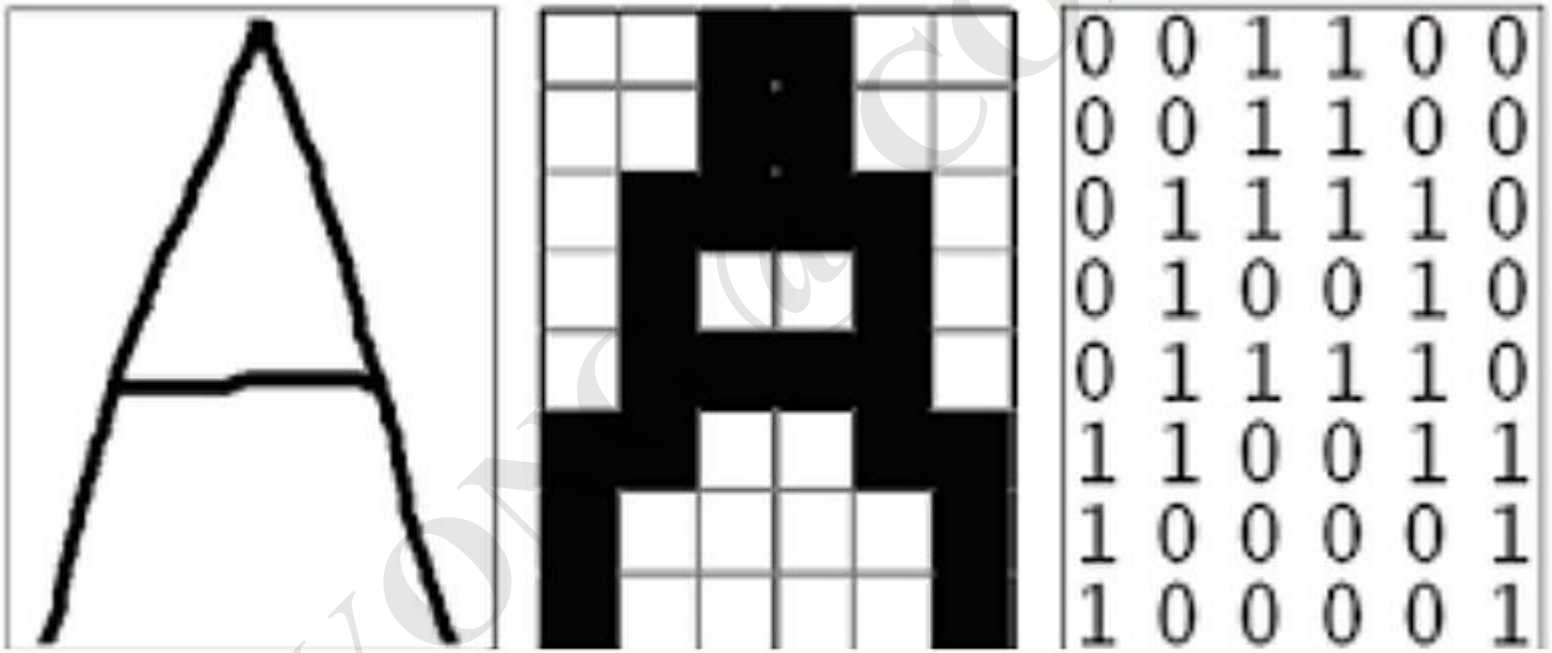
>Display the images

>Play with the images (colors, sizes, rotations)

>Play with the videos (files, webcam)

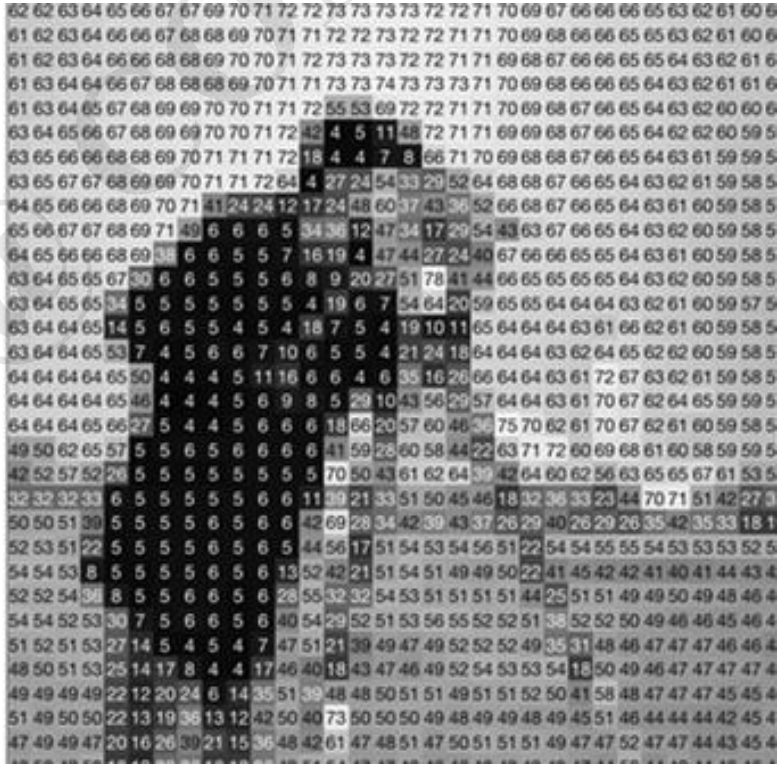
What is Computer Vision?

What human/computers see



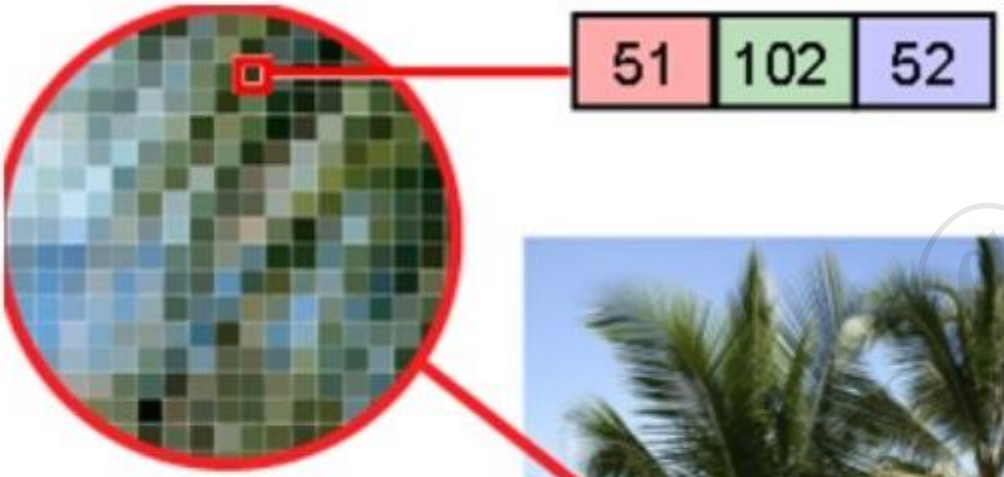
https://santanderglobaltech.com/en/computer-vision-vs-human-vision-this-is-how-computers-see/

What human/computers see



https://expressanalytics.com/blog/10-things-about-computer-vision-and-data-analytics-you-may-not-know/

What human/computers see



https://santanderglobaltech.com/en/computer-vision-vs-human-vision-this-is-how-computers-see/

What human/computers see

http://spsprashanth.blogspot.com/2016/08/rgb.html


TALK
IS CHEAP SHOW ME
THE CODE

Let's play with it!

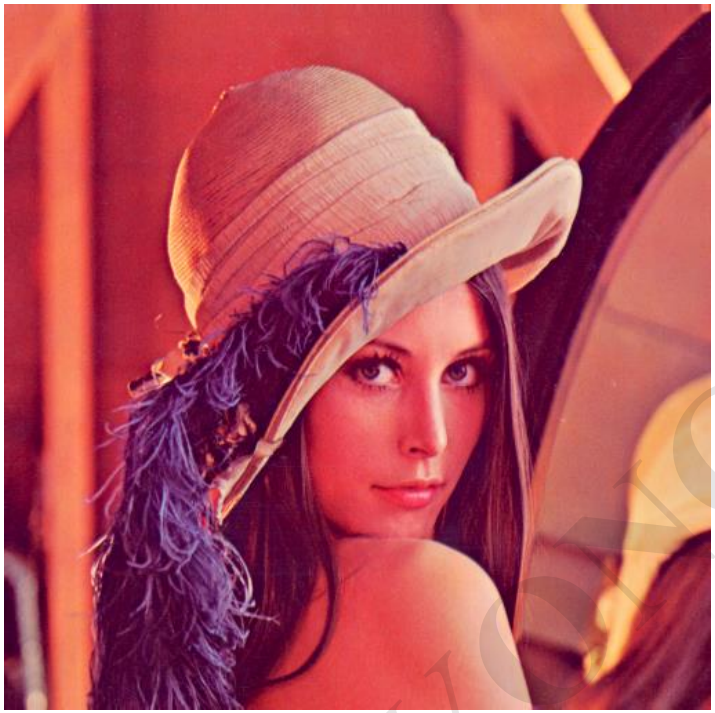
Get yourself a Google CoLab

>1. You need a Google Account

>2. Sign in at https://colab.research.google.com/?utm_source=scs-index

>3. You're all set!

Load an image



.bmp
Device-Independent Bitmap (BMP)

.jpg or .jpeg
Joint Photographic Experts Group (JPEG)

.tif or .tiff
Tagged Image File Format (TIFF)

Load an image

image.procesing1.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Image Processing 1: Let's play with the images

1. Load an image

There are several ways to load images. We show two examples of *PIL* and *skimage.io* here. Besides, to load an image with the url, help of *urllib.request*.

```
from PIL import Image
import urllib.request
import skimage.io

# save the image from the url
urllib.request.urlretrieve('https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna_%28test_image%29.png',
                           'lenna.png')

# load the image using PIL.Image
img_lenna = Image.open('lenna.png')
display(img_lenna)

# load the image using skimage.io
img_lenna2 = skimage.io.imread(fname='lenna.png')
skimage.io.imshow(img_lenna2)
```

Import the libraries. Recall the *include* in **C** or *using* in **C#**

Call a function

Load an image

image.procesing1.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

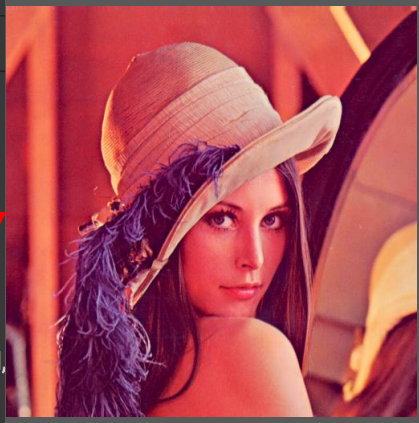
+ Code + Text

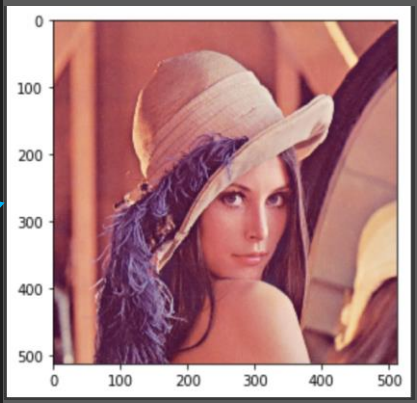
Image Processing 1: Let's play with the images

1. Load an image

There are several ways to load images. We show two examples of *PIL* and *skimage.io* here. Besides, to load an image with the url, help of *urllib.request*.

```
from PIL import Image
import urllib.request
import skimage.io

# save the image from the url
urllib.request.urlretrieve('https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna_%28test_image%29.png',
                           'lenna.png')

# load the image using PIL.Image
img_lenna = Image.open('lenna.png')
display(img_lenna)

# load the image using skimage.io
img_lenna2 = skimage.io.imread(fname='lenna.png')
skimage.io.imshow(img_lenna2)
```





See what it is

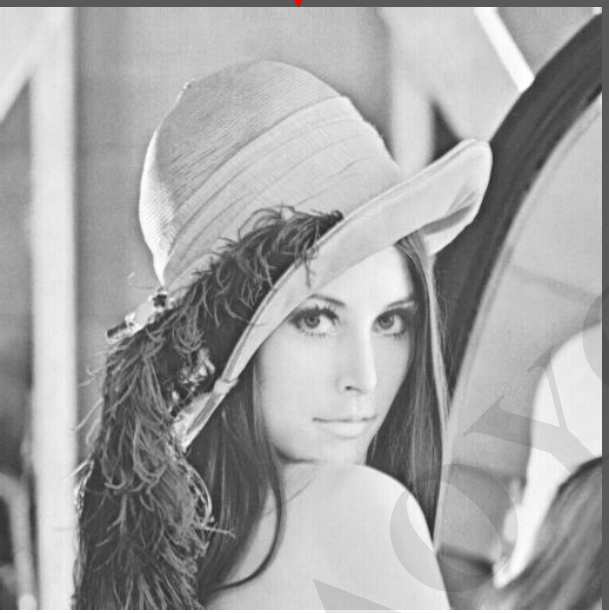
2. See what it is

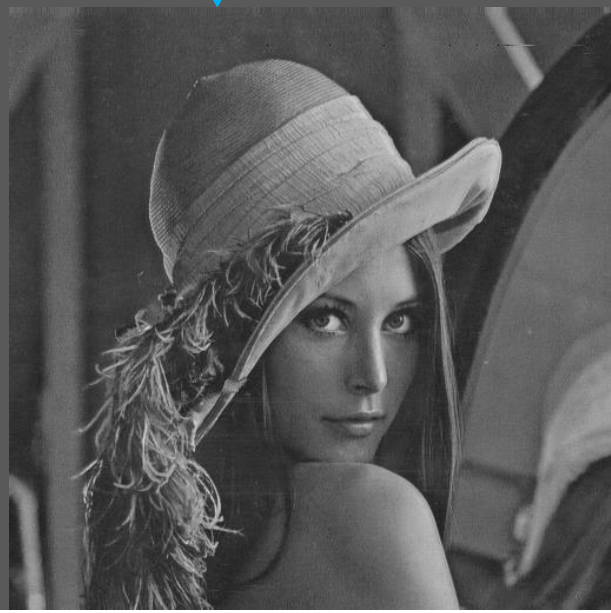
```
[43] # check out the image
     print('Format:\t'+img_lenna.format)
     print('Size:\t'+"%s"%(img_lenna.size,))
     print('Mode:\t'+img_lenna.mode)

     # check out the image as arrays
     import numpy as np
     im = np.array(img_lenna)

     print('Array Type:\t'+str(type(im)))
     print('Data Type:\t'+str(im.dtype))
     print('Array Shape:\t'+"%s"%(im.shape,))
```

```
Format: PNG
Size:   (512, 512)
Mode:   RGB
Array Type:     <class 'numpy.ndarray'>
Data Type:      uint8
Array Shape:    (512, 512, 3)
```

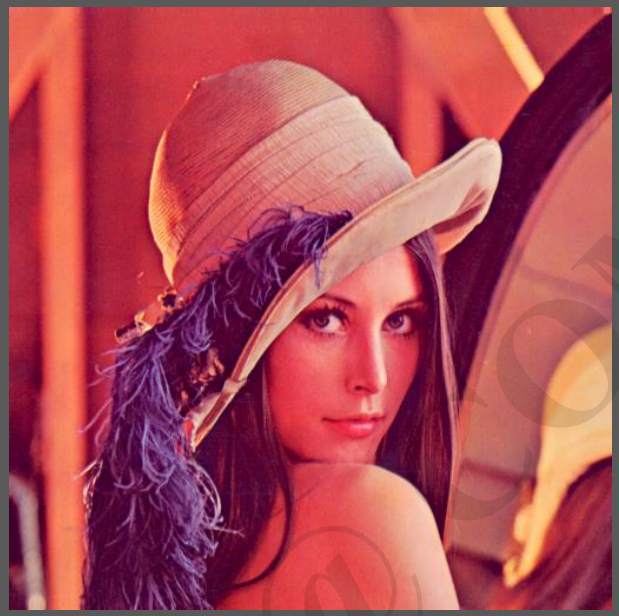
See what it is

```
# check out channels
R_img=Image.fromarray(im[:,:,0])
display(R_img)
G_img=Image.fromarray(im[:,:,1])
display(G_img)
B_img=Image.fromarray(im[:,:,2])
display(B_img)
```





What information do these channels tell?



The R channels are dominating. It results in a **warm** color image.

+

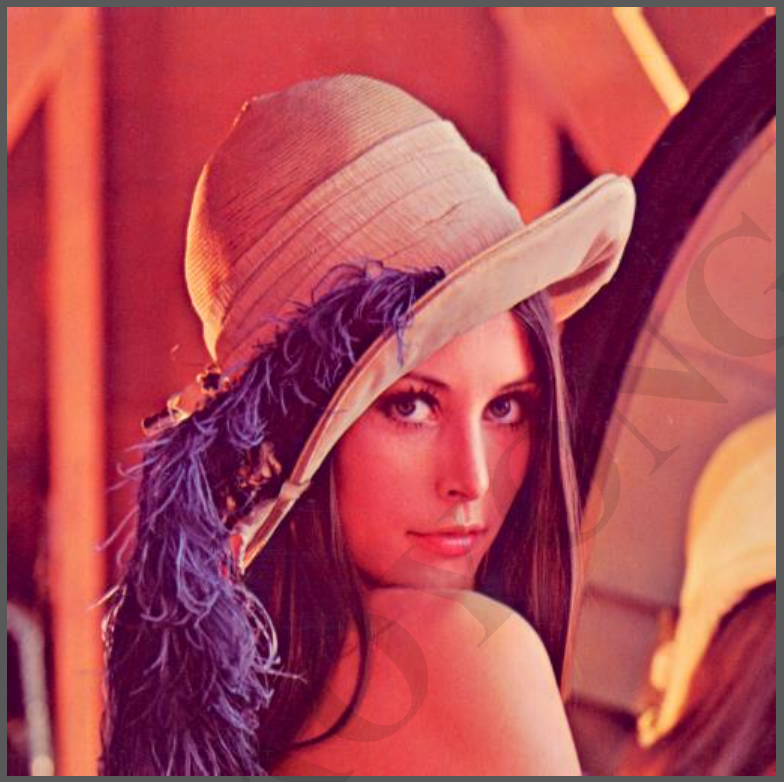
+

Can we make it a cool color?

We can see from above that the R channel is dominating, which results in a warm color image. Can we make it a cool color?

It can be done by simply "inverse" the value of each pixel within the range of [0, 255]. For example, for a value x (e.g., 30), it's inverted value is (255-x) (e.g., 255-30 = 225).

```
cool_img=Image.fromarray(255-im)
display(cool_img)
```



Other solutions?

It's too large to fit into my small screen.

Can you make it smaller?

Resize an image

Sure. We can resize an image (let's call it the source image hereafter) by generating a target image with pixels selected from the source. For example, to reduce the size by 50%, we can select one from two neighbour pixels into the target image.

```
from numpy.core.fromnumeric import shape
cool_array=255-im
width_new,height_new=int(im.shape[0]/2),int(im.shape[1]/2)

cool_array_small=np.ndarray((width_new,height_new,3),dtype=np.uint8)
for i in range(0,width_new):
  for j in range(0,height_new):
    cool_array_small[i,j,:]=cool_array[2*i,2*j,:]

cool_img_small=Image.fromarray(cool_array_small)
print(cool_array_small.dtype)
print(type(cool_array_small))
print(cool_img_small.size)
display(cool_img_small)
```

Resize an image

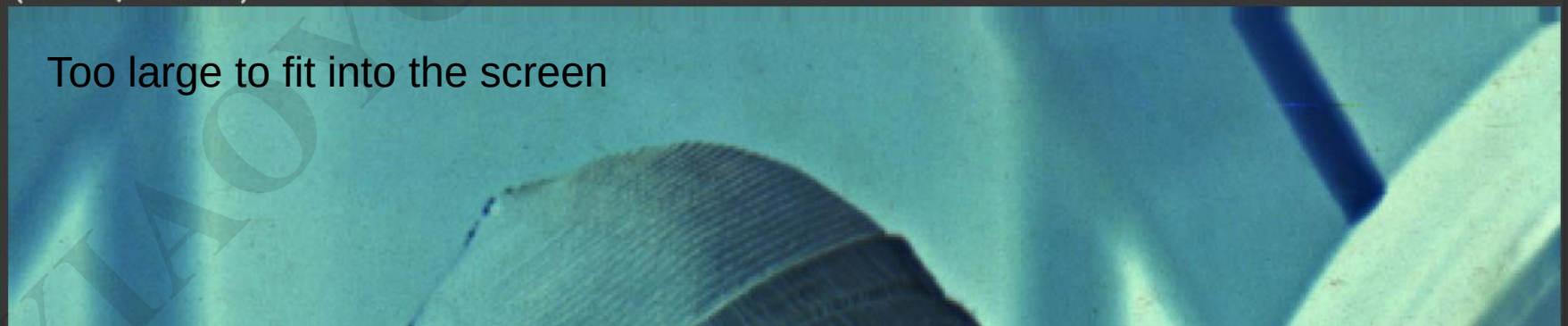
```
cool_array_small=np.ndarray((width_new,height_new,3),dtype=np.uint8)
for i in range(0,width_new):
  for j in range(0,height_new):
    cool_array_small[i,j,:]=cool_array[2*i,2*j,:]
```

```
uint8
<class 'numpy.ndarray'>
(256, 256)
```

Similarly, we can enlarge the image by duplicating the pixels to its neighbours in the target. Let's double the size of the source.

```
width_lg,height_lg=int(im.shape[0]*2),int(im.shape[1]*2)

cool_array_large=np.ndarray((width_lg,height_lg,3),dtype=np.uint8)
for i in range(0,width_lg):
  for j in range(0,height_lg):
    cool_array_large[i,j,:]=cool_array[int(i/2),int(j/2),:]

cool_img_large=Image.fromarray(cool_array_large)
print(cool_array_large.dtype)
print(type(cool_array_large))
print(cool_img_large.size)
display(cool_img_large)
```

```
uint8
<class 'numpy.ndarray'>
(1024, 1024)
```



It's cool! Can we enlarge an image into any size?

Let's try enlarge it by 10 times using the same code and see what happens.

Enlarge the image by 10 times

What happened?

Image Interpolation

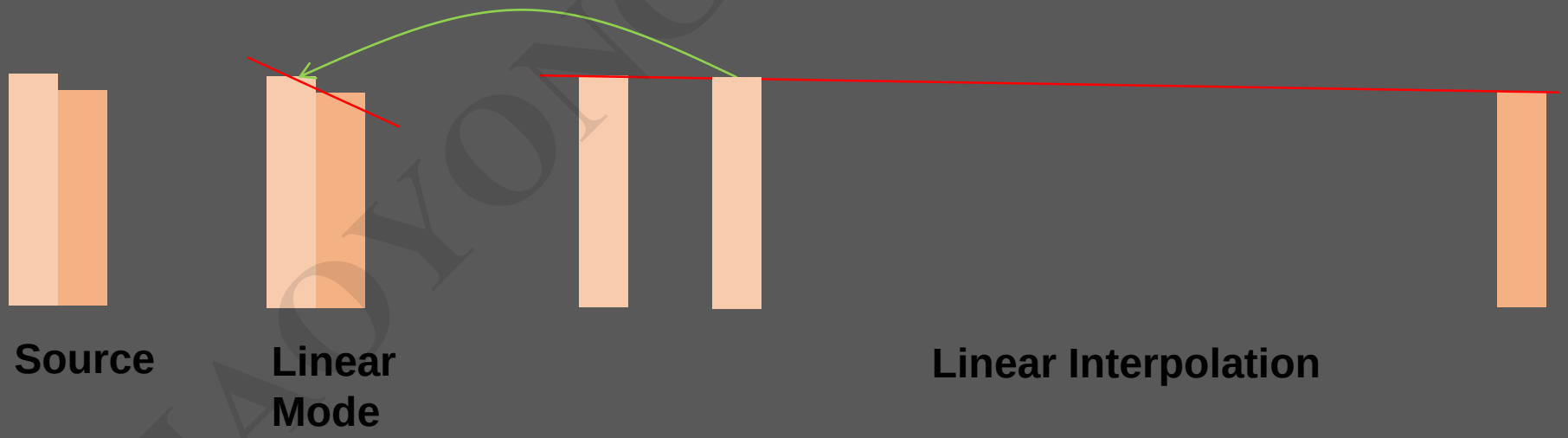
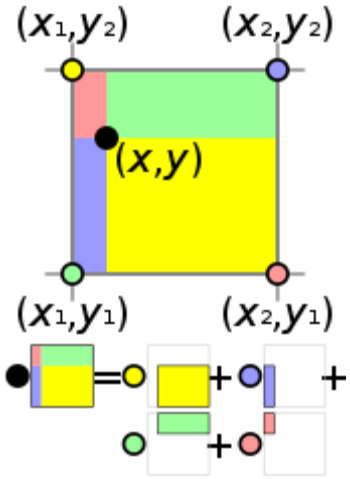
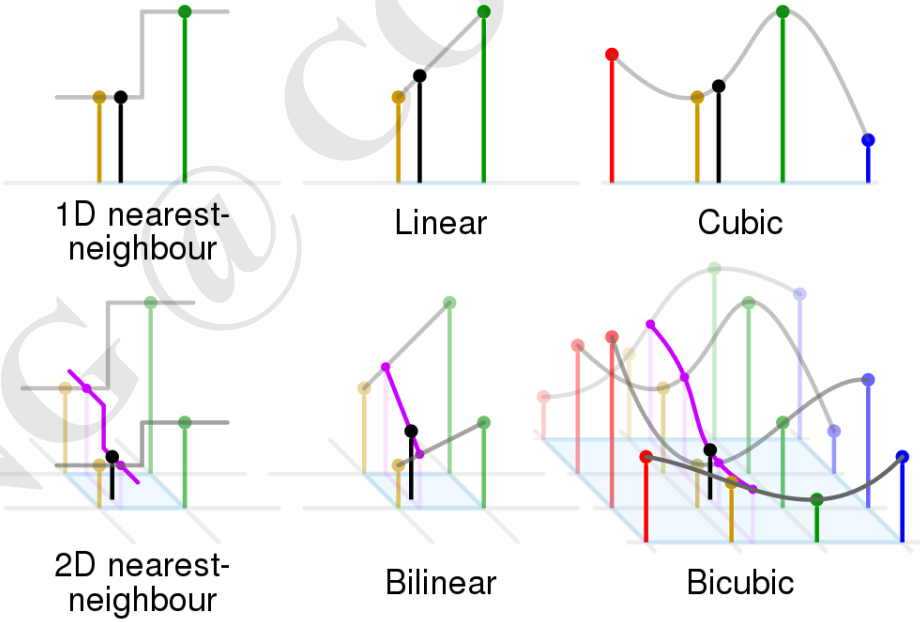


Image Interpolations





https://en.wikipedia.org/wiki/Bicubic_interpolation

Great!
What if I have to watch it from bed?

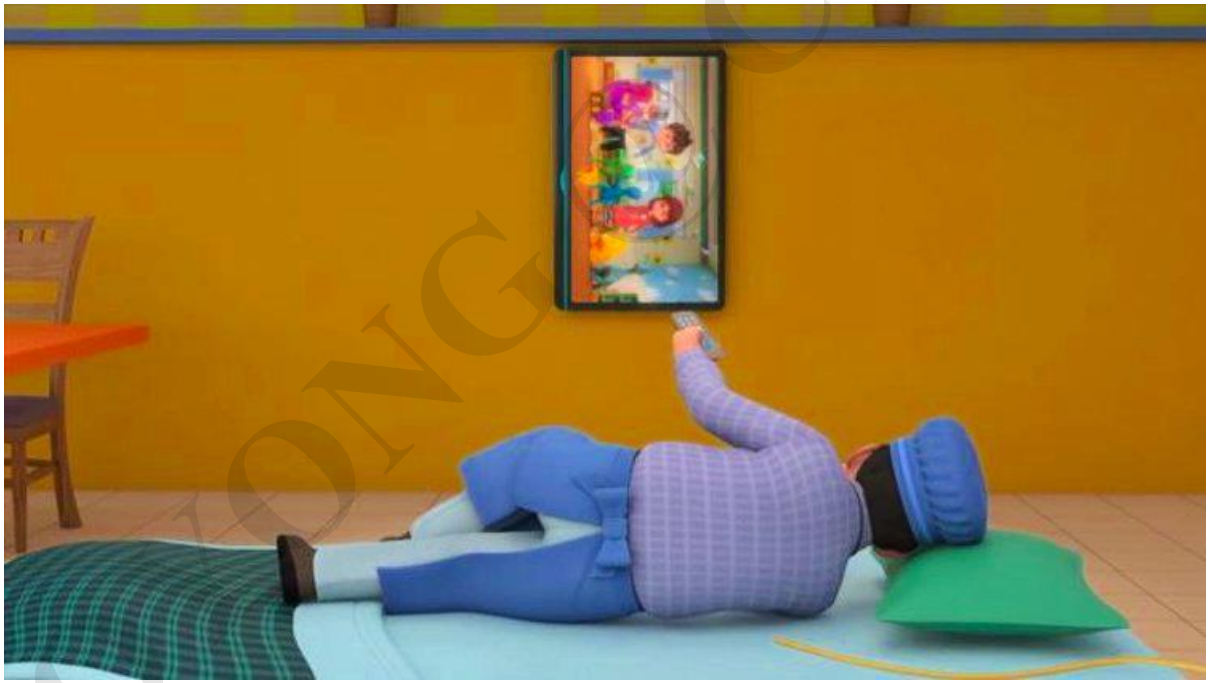


Image Rotation

$$x_2 = cos(\theta) * (x_1 - x_0) - sin(\theta) * (y_1 - y_0) + x_0$$

$$y_2 = sin(\theta) * (x_1 - x_0) + cos(\theta) * (y_1 - y_0) + y_0$$

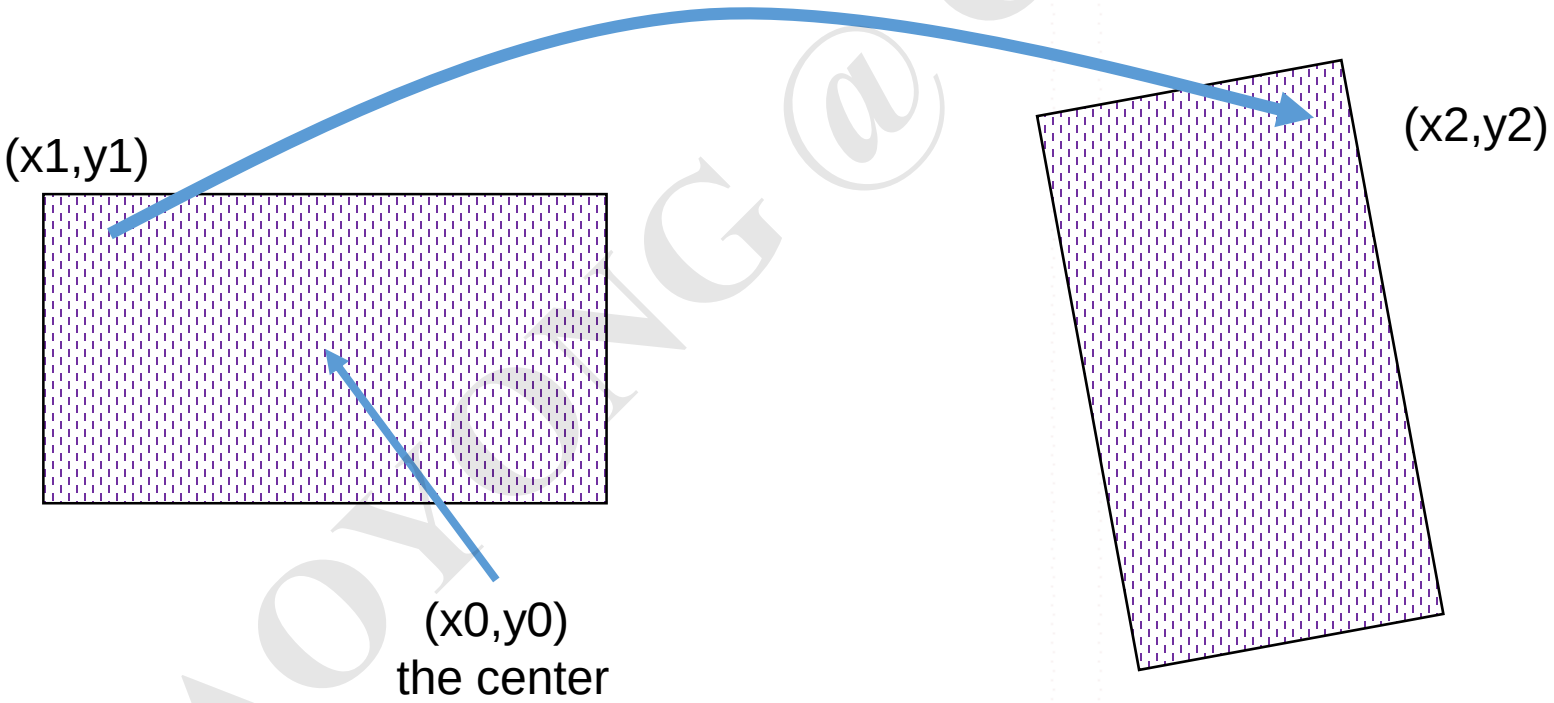


Image Rotation

Great! Let's take a step further. Say, we want to watch the image when we're lying on the bed, and it would be better to rotate the image clockwisely by 90 degree so that we don't have to rotate our screeen.

```
width_cw90,height_cw90=cool_array.shape[1],cool_array.shape[0]

cool_array_cw90=np.ndarray((width_cw90,height_cw90,3),dtype=np.uint8)
for i in range(0,width_cw90):
  for j in range(0,height_cw90):
    y0,x0=int(height_cw90/2),int(width_cw90/2)
    y1,x1=j,i
    y2,x2=min((x1-x0)+y0,width_cw90-1),min(-(y1-y0)+x0,height_cw90-1)
    cool_array_cw90[i,j,:]=cool_array[x2,y2,:]

cool_img_cw90=Image.fromarray(cool_array_cw90)
print(cool_img_cw90.size)
display(cool_img_cw90)
```

```
cool_array_cw90=np.ndarray((width_cw90,height_cw90,3),dtype=np.uint8)
for i in range(0,width_cw90):
  for j in range(0,height_cw90):
    y0,x0=int(height_cw90/2),int(width_cw90/2)
    y1,x1=j,i
    y2,x2=min((x1-x0)+y0,width_cw90-1),min(-(y1-y0)+x0,height_cw90-1)
    cool_array_cw90[i,j,:]=cool_array[x2,y2,:]
```

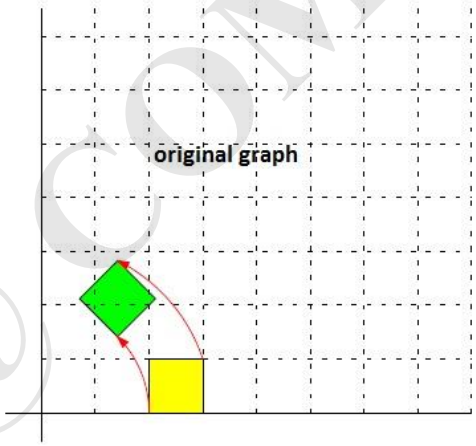
Image Rotation

To rotate $45°$ about the origin, we apply the matrix

$$\begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix} = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

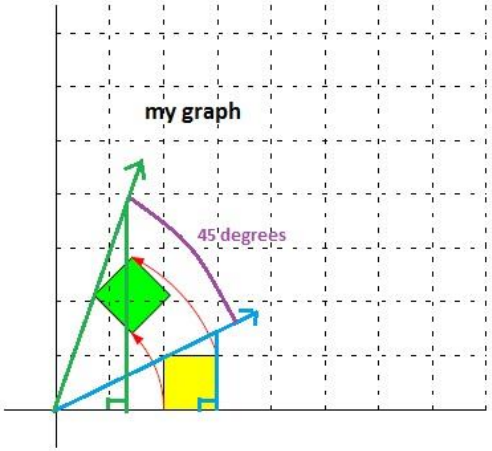
Note: $\frac{\sqrt{2}}{2} = \cos 45° = \sin 45°$, so this is the same as

$$\begin{pmatrix} \cos 45° & -\sin 45° \\ \sin 45° & \cos 45° \end{pmatrix}$$



Counter Clockwise $\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$

Clockwise $\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$



Geometric Manipulations as Matrix Transformations

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Rotation **Translation** **Scaling**

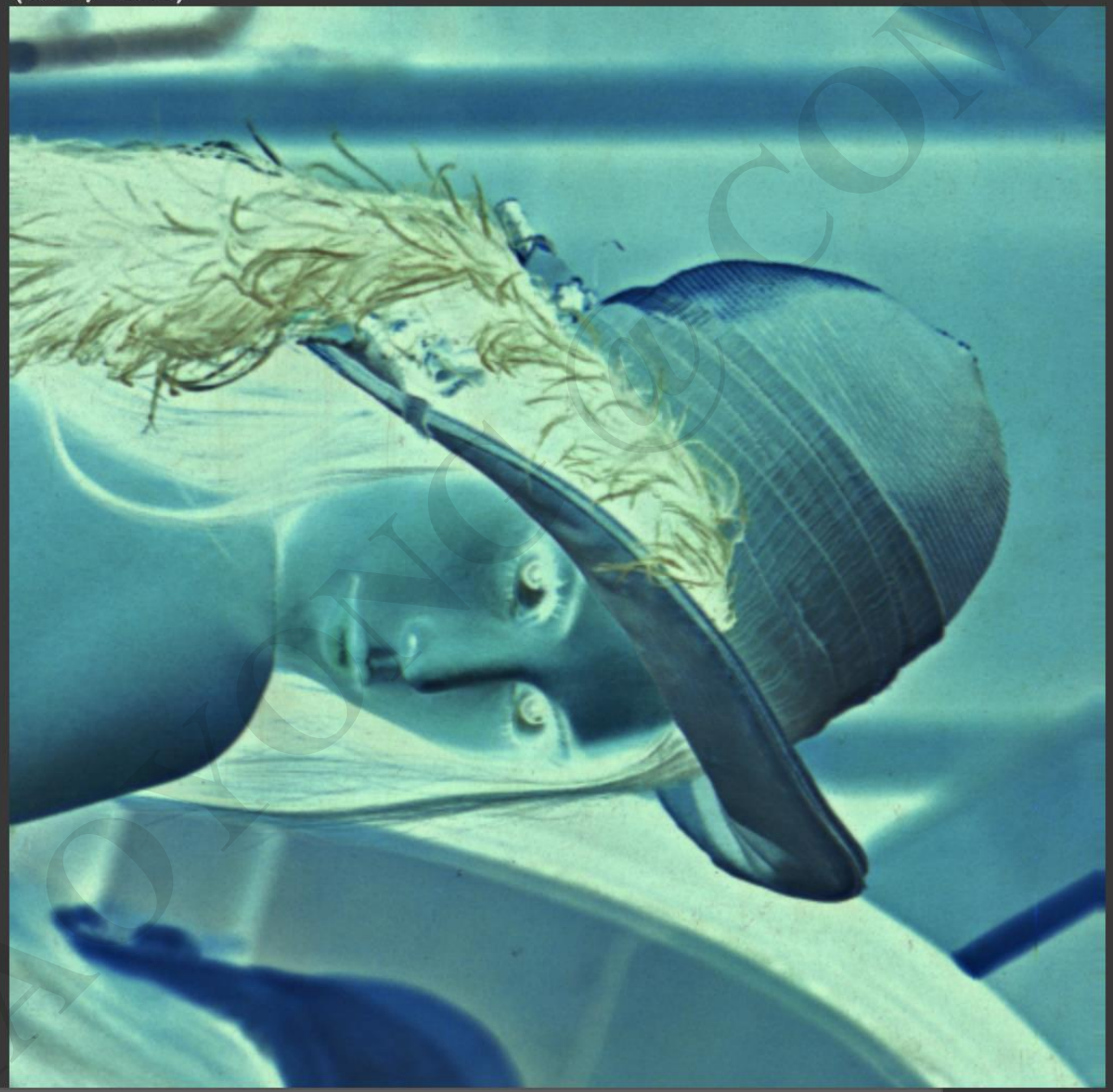
Congratulations! You just grasped the basis of image processing.

But …

There are much easier ways …

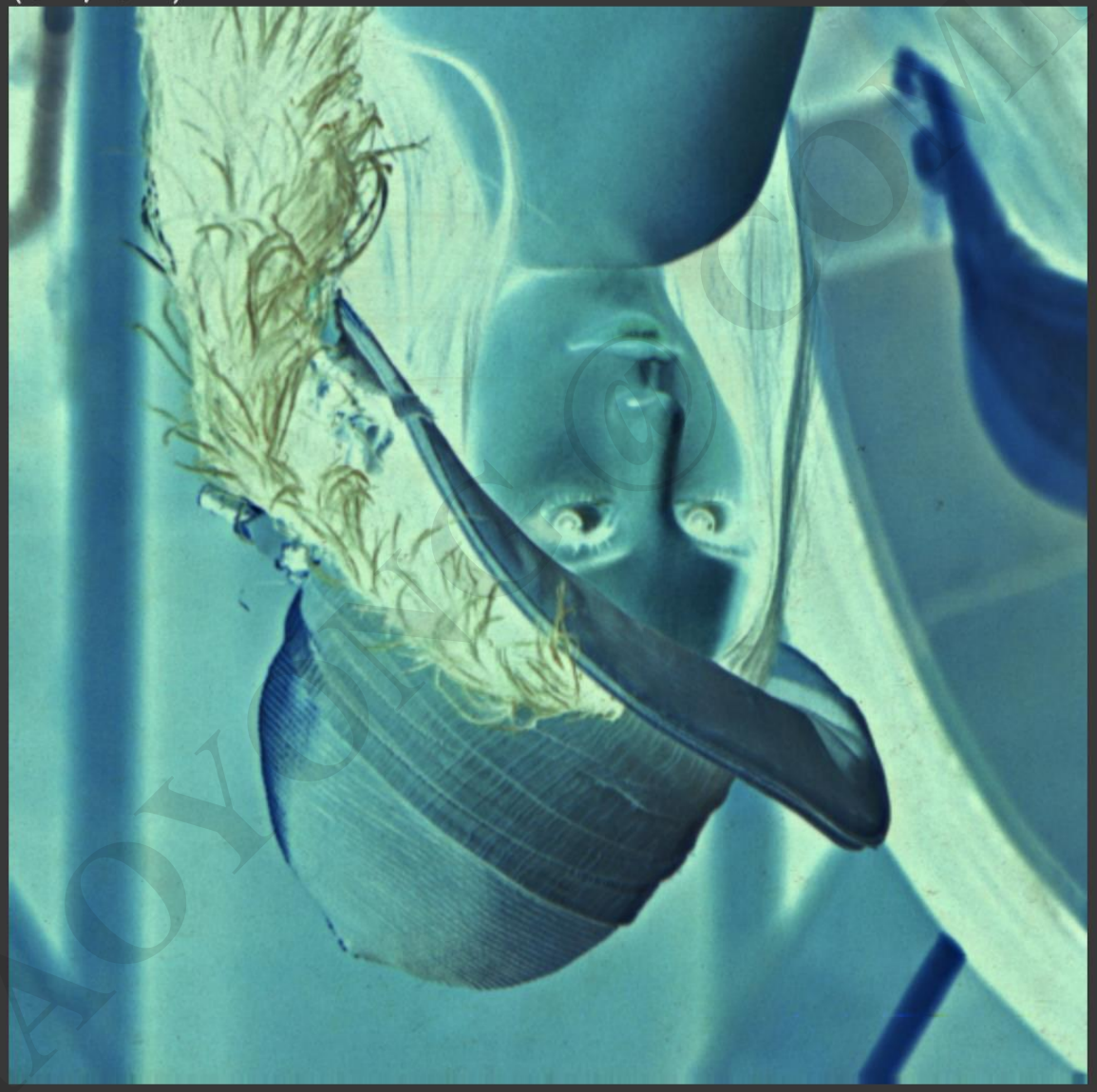
```
import cv2 #import OpenCV

img_rotate_90_clockwise = cv2.rotate(cool_array, cv2.ROTATE_90_CLOCKWISE)
cool_img_cw90=Image.fromarray(img_rotate_90_clockwise)
print(cool_img_cw90.size)
display(cool_img_cw90)
```

```
(512, 512)
```



```
img_flip_ud = cv2.flip(cool_array, 0)
cool_img_new=Image.fromarray(img_flip_ud)
print(cool_img_new.size)
display(cool_img_new)
```

```
(512, 512)
```



```
# convert the image into gray
im_gray = cv2.cvtColor(cool_array, cv2.COLOR_BGR2GRAY)
# binarize the image using a threshold
th, im_gray_th_otsu = cv2.threshold(im_gray, 128, 192, cv2.THRESH_OTSU)
cool_img_new=Image.fromarray(im_gray_th_otsu)
print(cool_img_new.size)
display(cool_img_new)
```

```
(512, 512)
```

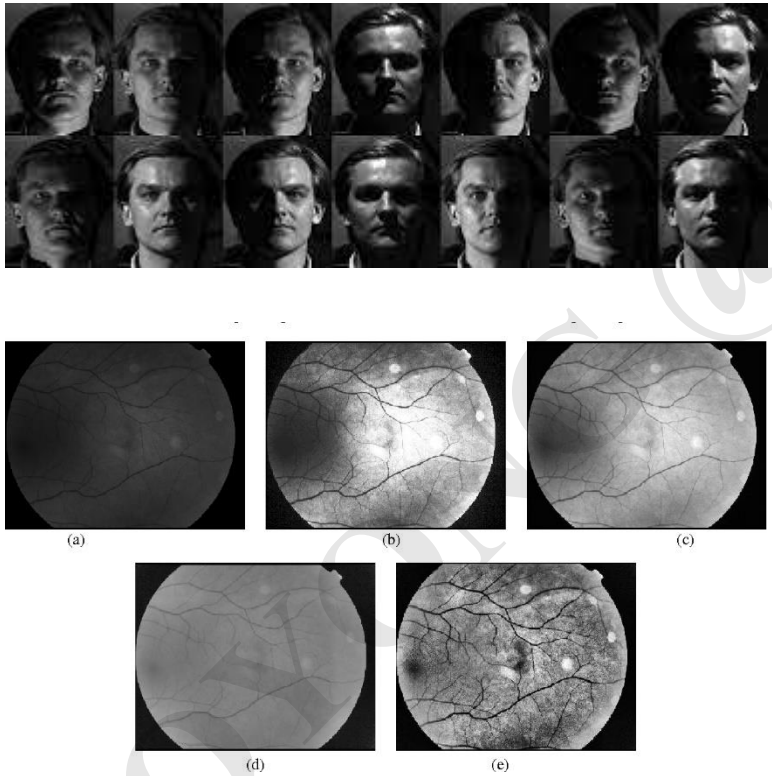
```
cool_img_lg=cool_img.resize((800,300),Image.BICUBIC)
display(cool_img_lg)
```

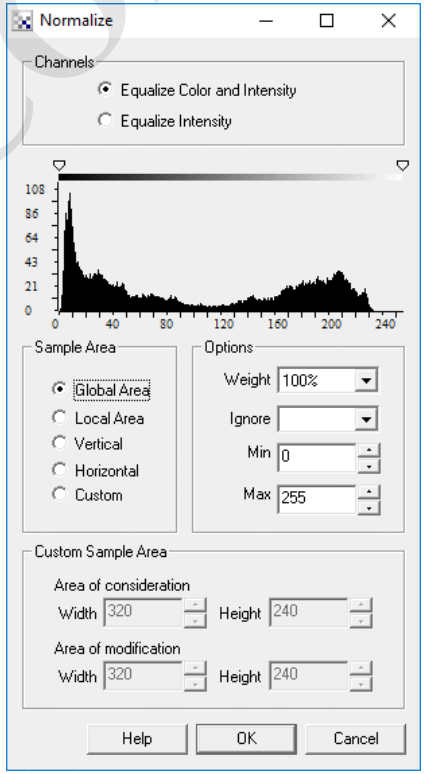
Why didn't you tell us earlier?

Why would I have to learn those? Where do the fancy stuff in the Lecture 1 go?

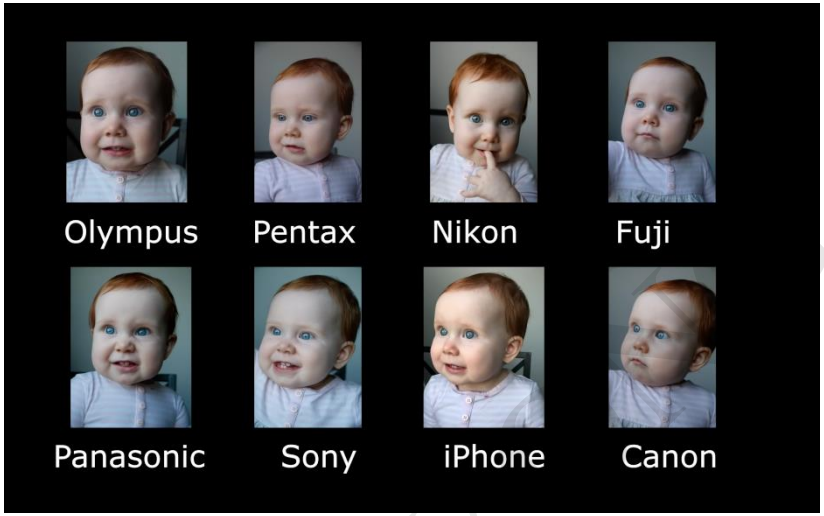
Image preprocessing are the steps taken before the model training.

Illumination Normalization





Sensor Gap and Color Normalization



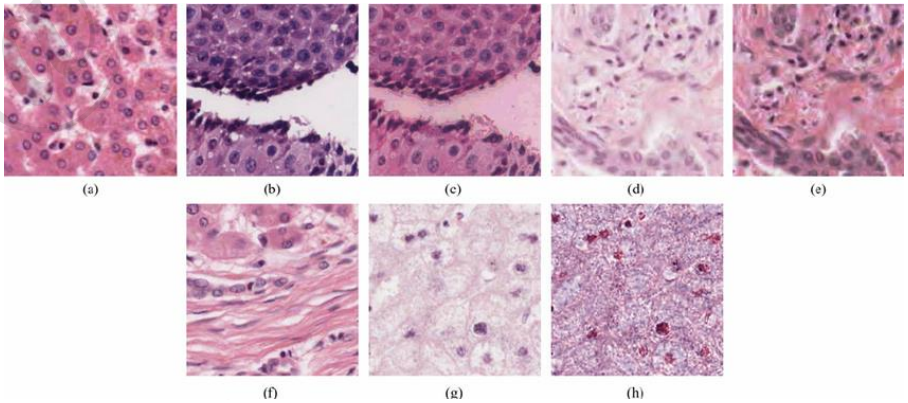


Image Registration

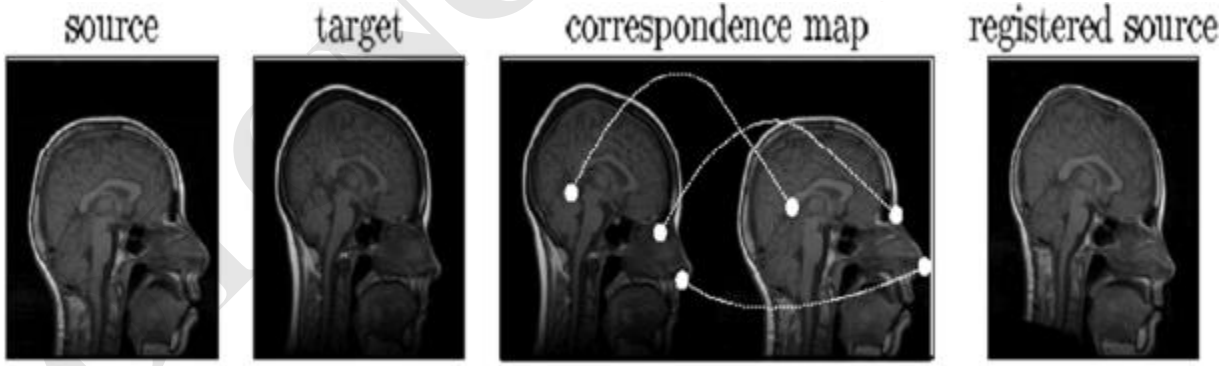


Figure 1.2: Image Registration Process

Let's play with videos

Load video stream (webcam)

```python
import cv2

# declare a video capture object
vid = cv2.VideoCapture(0)

while True:
    # capture the video frame by frame
    ret, frame_raw = vid.read()
    frame = cv2.flip(frame_raw, 1) # optional
    # display the resulting frame
    cv2.imshow('COMP 4423', frame)

    # quit when 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release the cap object
vid.release()
# destroy all the windows
cv2.destroyAllWindows()
```

Invert the images (frames)

```python
import cv2

# declare a video capture object
vid = cv2.VideoCapture(0)
inversion_on=False

while True:
    # capture the video frame by frame
    ret, frame_raw = vid.read()
    frame = cv2.flip(frame_raw, 1) # optional
    # get dimensions of the frame
    h, w, c = frame.shape
    print(h,w,c)
    # display the resulting frame
    cv2.imshow('COMP 4423', 255-frame if inversion_on else frame)

    key=cv2.waitKey(1) & 0xFF
    # quit when 'q' is pressed
    if  key== ord('q'):
        break
    if key==ord('i'):
        inversion_on=not inversion_on

# release the cap object
vid.release()
# destroy all the windows
cv2.destroyAllWindows()
```

Put the hair on

```python
import cv2,numpy as np

# declare a video capture object
vid = cv2.VideoCapture(0)
inversion_on=False

# load the redhair image
im_readhair = cv2.imread('redhair.png', cv2.IMREAD_UNCHANGED)
hh,wh,ch=im_readhair.shape
# resize the hair image
im_readhair = cv2.resize(im_readhair, (int(wh*0.7),int(hh*0.7)), interpolation = cv2.INTER_AREA)
# find theindices of non-transparent pixels
non_trans=np.argwhere(im_readhair[:,:,3]>0)

h2,w2,c2=im_readhair.shape
alpha = im_readhair[:,:,3]
hair_on=False

while True:
    # capture the video frame by frame
    ret, frame_raw = vid.read()
    frame = cv2.flip(frame_raw, 1) # optional
    # get dimensions of the frame
    h, w, c = frame.shape
    print(h,w,c,'-',h2,w2,c2)
```

Put the hair on (cont.)

```
21      ret, frame_raw = vid.read()
22      frame = cv2.flip(frame_raw, 1) # optional
23      # get dimensions of the frame
24      h, w, c = frame.shape
25      print(h,w,c,'-',h2,w2,c2)
26
27      # put the hair on
28      if hair_on:
29          top,left=50,int((w-w2)/2)
30          #frame[top:top+h2,left:left+w2,:]=im_readhair[:,:,0:3]
31          frame[top+non_trans[:,0],left+non_trans[:,1],:]=im_readhair[non_trans[:,0],non_trans[:,1],0:3]
32
33      # display the resulting frame
34      cv2.imshow('COMP 4423', 255-frame if inversion_on else frame)
35
36      key=cv2.waitKey(1) & 0xFF
37      # quit when 'q' is pressed
38      if  key== ord('q'):
39          break
40      if key==ord('i'):
41          inversion_on=not inversion_on
42      if key==ord('h'):
43          hair_on=not hair_on
44
45  # release the cap object
46  vid.release()
47  # destroy all the windows
48  cv2.destroyAllWindows()
```

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學
Department of Computing
電子計算學系

Happy New Year!