**NUS-ISS**
Vision Systems

Module 3 - Foundations of computer vision system (2) - Local feature and representation, part 2

Dr. Tan Jen Hong
Lecturer & Consultant
Institute of System Science
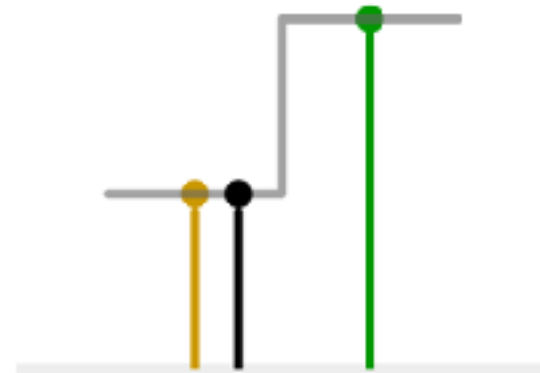National University of Singapore
issjht@nus.edu.sg

Learning objectives

- Rescale / resize image

- Perform image translation and rotation

- Perform histogram equalization

- Perform image thresholding

vse/m2.2/v1.2

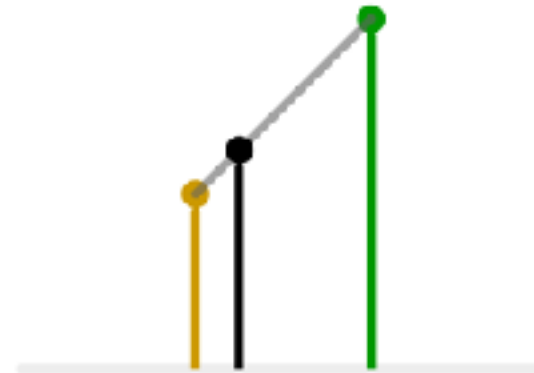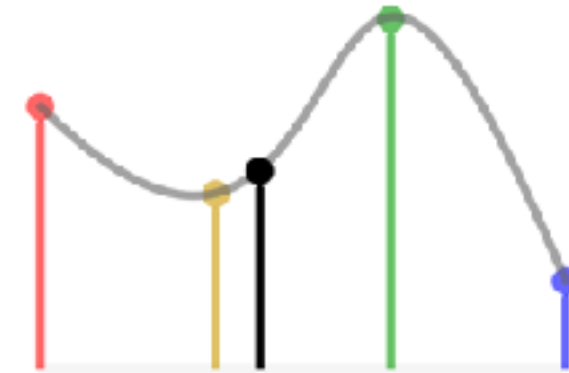# What is interpolation?
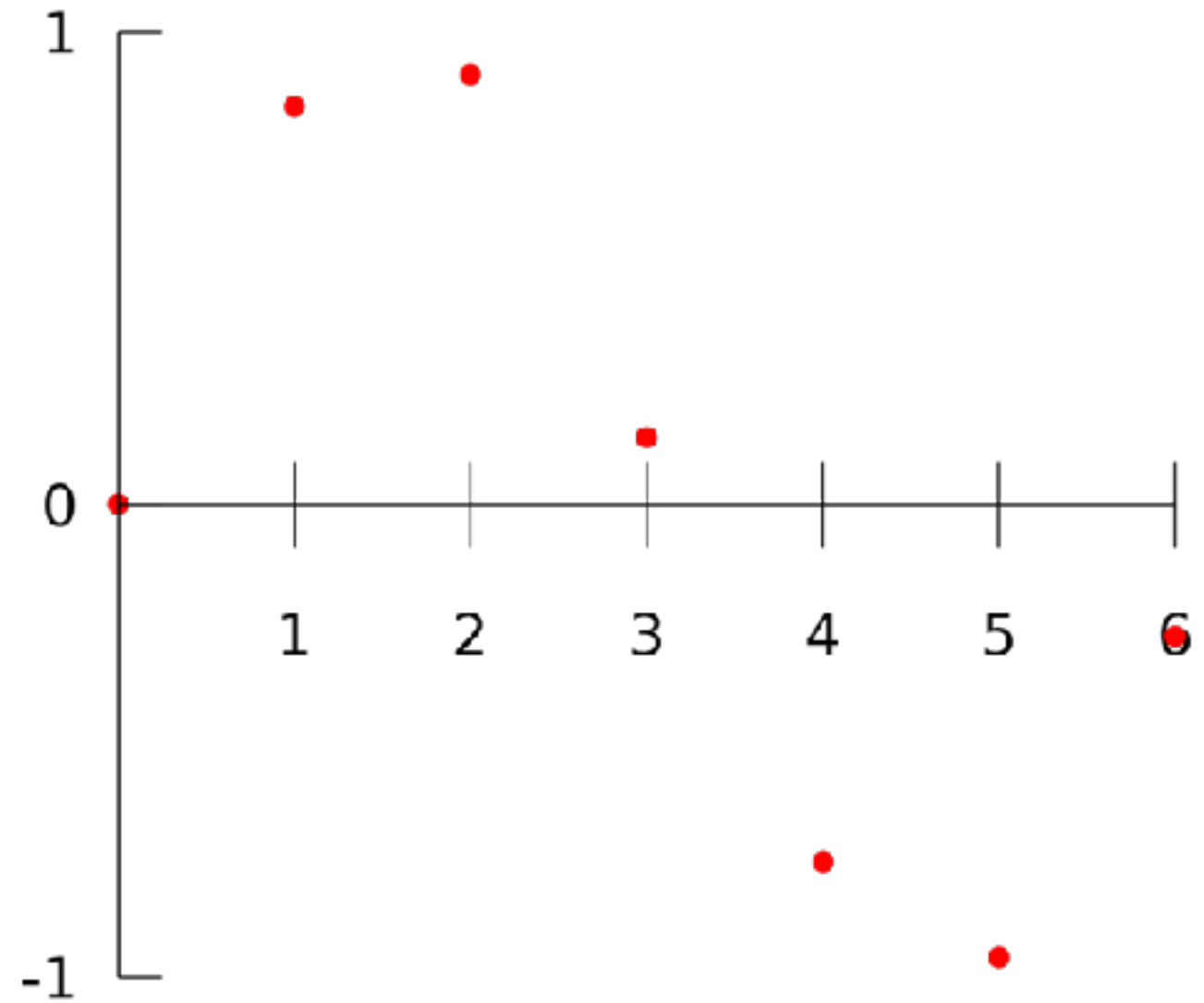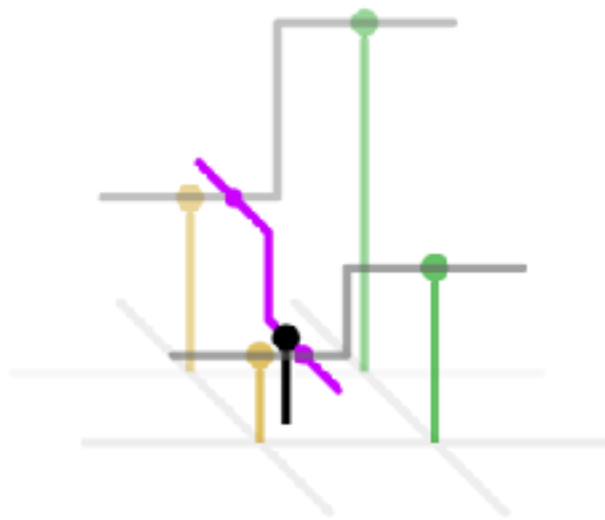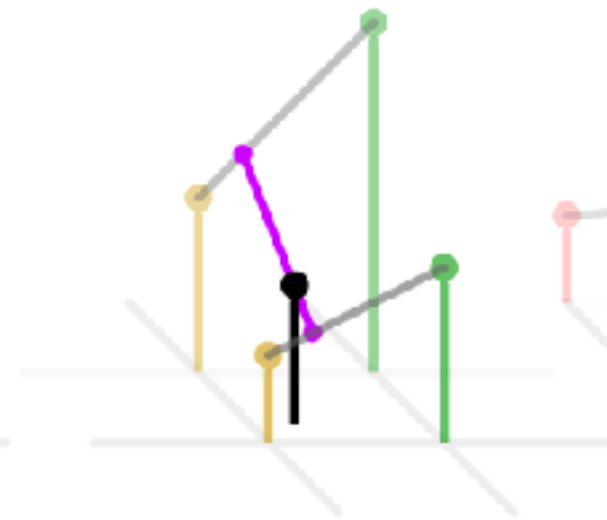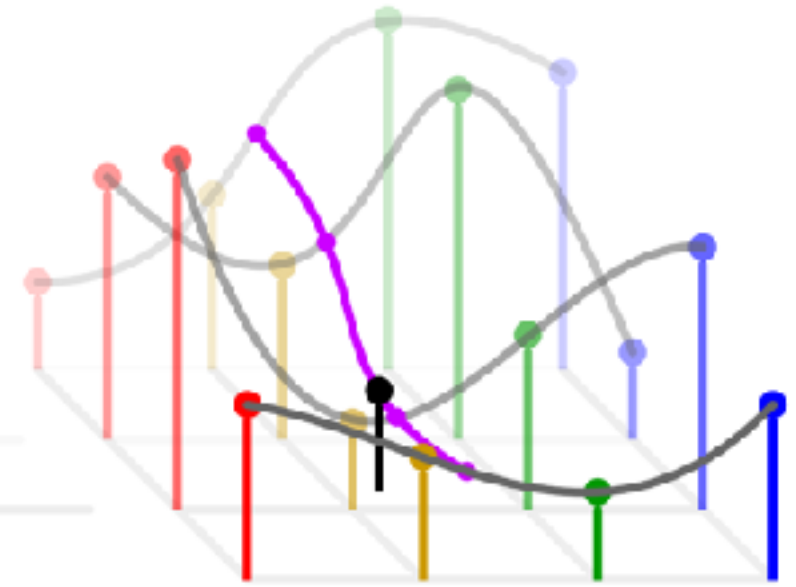
vse/m2.2/v1.2

Types of interpolation
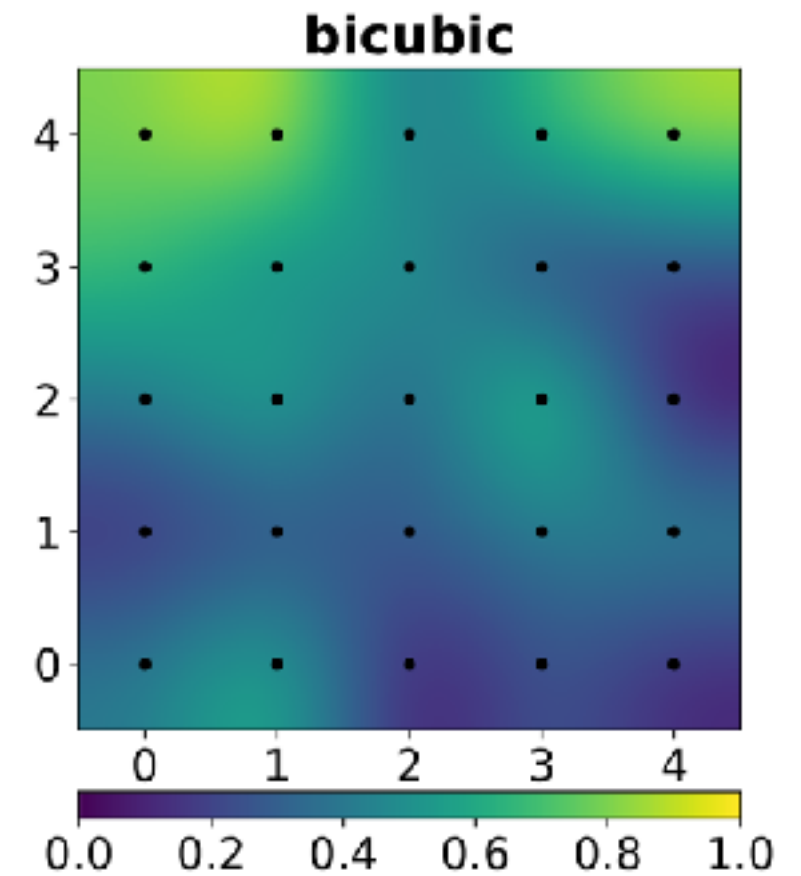

1D nearest-neighbour

Linear

Cubic

2D nearest-neighbour

Bilinear

Bicubic

vse/m2.2/v1.2

NUS | iss

Types of interpolation

2D nearest-neighbour   Bilinear   Bicubic

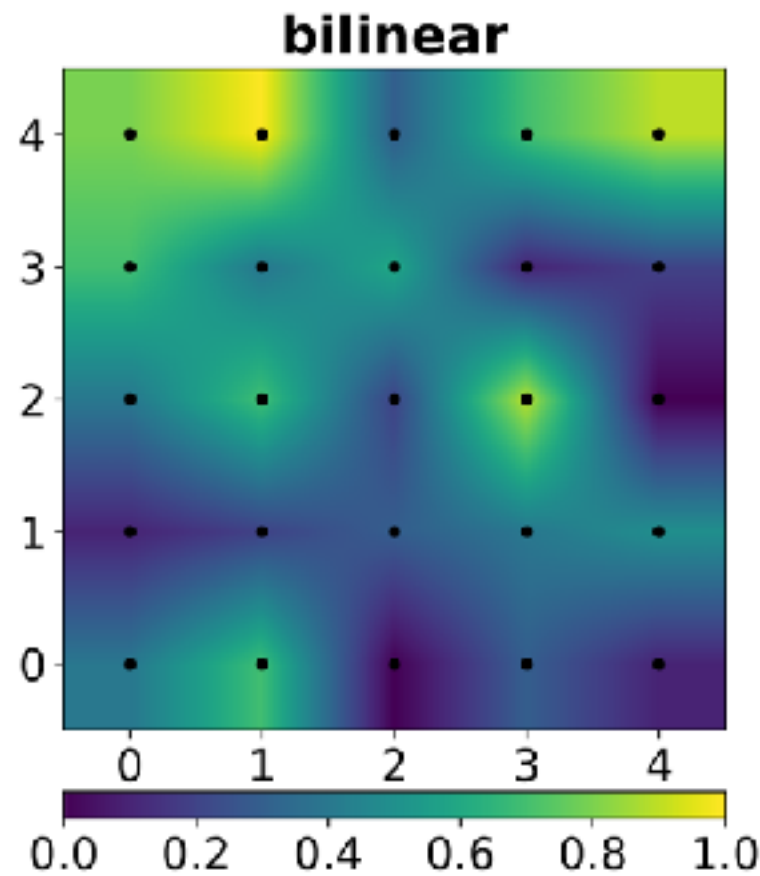nearest   bilinear   bicubic

Source: https://en.wikipedia.org/wiki/Bicubic_interpolation
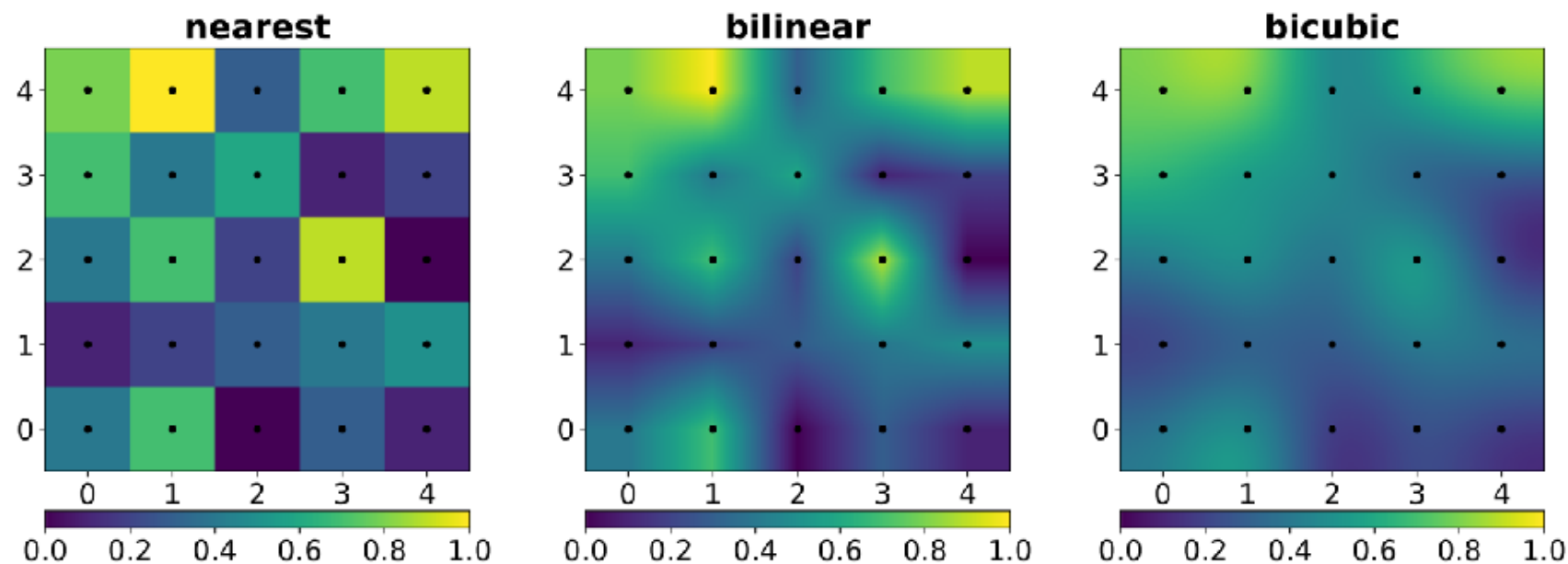
NUS | iss
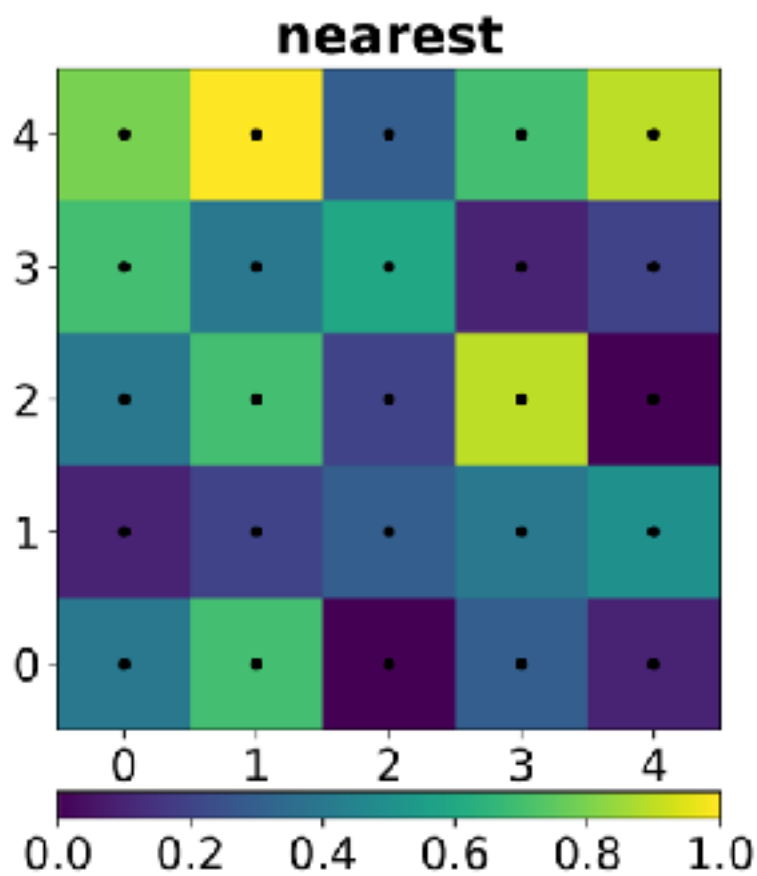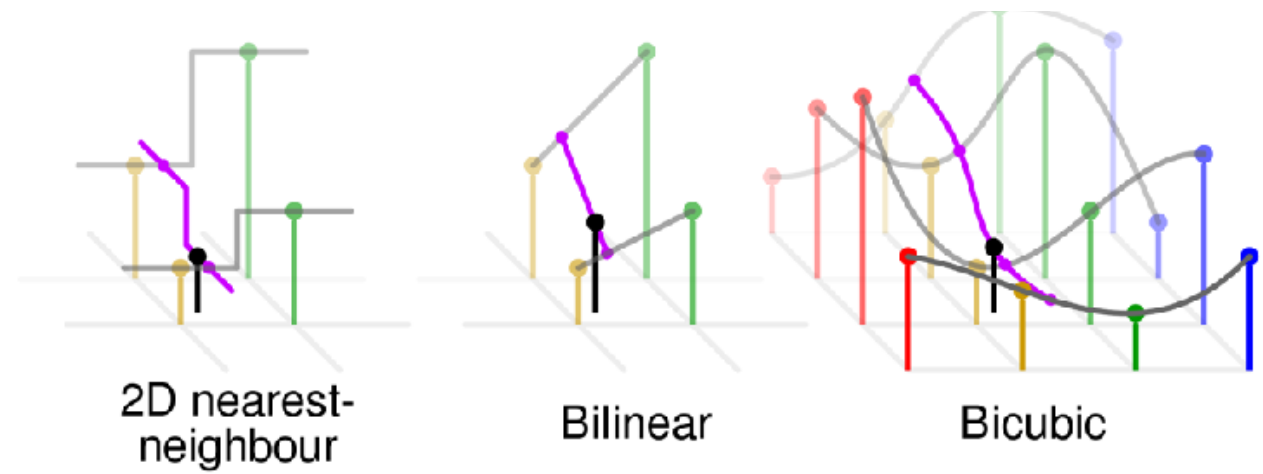
Roles of interpolation in image processing?

- Whenever we perform a geometric transformation on image, interpolation is always involved

- The choice of interpolation method affects the output and the calculation speed

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

vse/m2.2/v1.2

Rescale image
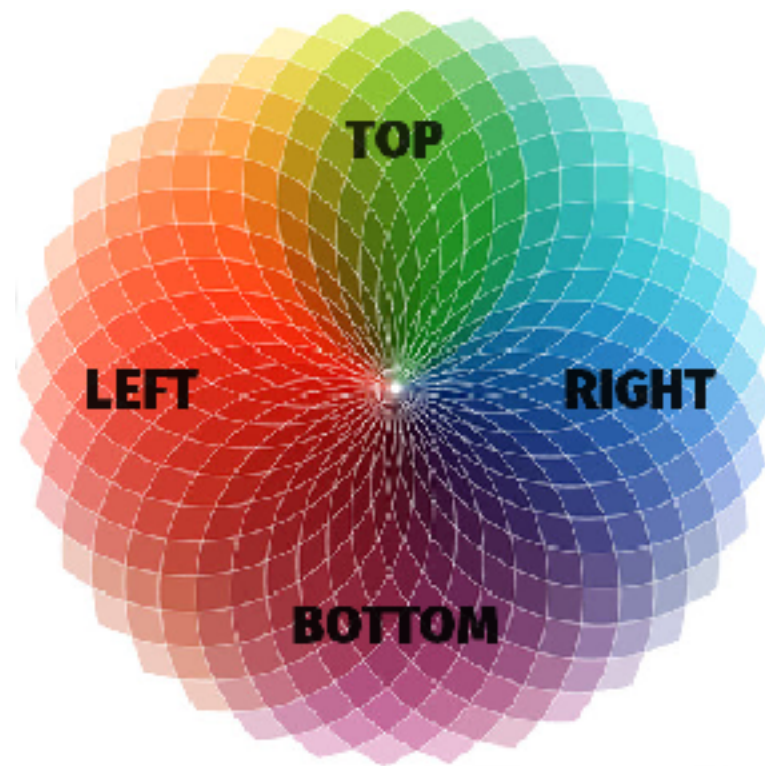
- Load the image and the necessary libraries

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

> src   = cv2.imread('cc.png')
```

- Check the shape and the type of the image

```
> src.shape
: (501, 501, 3)

> src.dtype
: dtype('uint8')

> print('img height: %d' % (src.shape[0]))
: img height: 501
```


cc.png

Rescale image

• Two ways to rescale image. First method: specify the output size

```
> rsz1 = cv2.resize(src,        width  height
                    output size  dsize=(300,200),
interpolation method  interpolation=cv2.INTER_AREA)
```
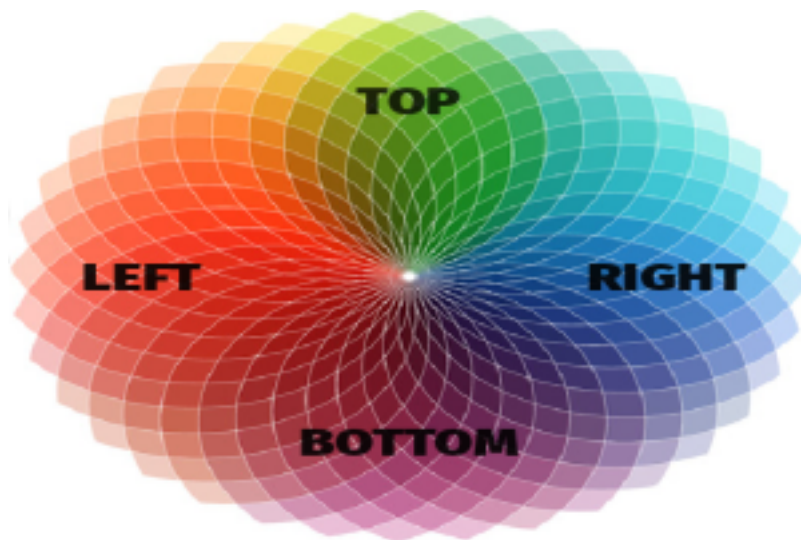
• Check the shape of the image

```
> rsz1.shape
: (200, 300, 3)

> print('img height: %d' % (rsz1.shape[0]))
: img height: 200
```


rsz1

Rescale image

- Second method: specify scale factor in x and y directions

```
> rsz2 = cv2.resize(src,
          output size    dsize=None,
 scale factor for width  fx=0.5,
scale factor for height  fy=0.7,
   interpolation method  interpolation=cv2.INTER_LINEAR)
```

- `dsize` must be set to `None` to avoid error

- Check the shape of the image

```
> rsz2.shape
: (351, 250, 3)

> print('img height: %d' % (rsz2.shape[0]))
: img height: 351
```

rsz2

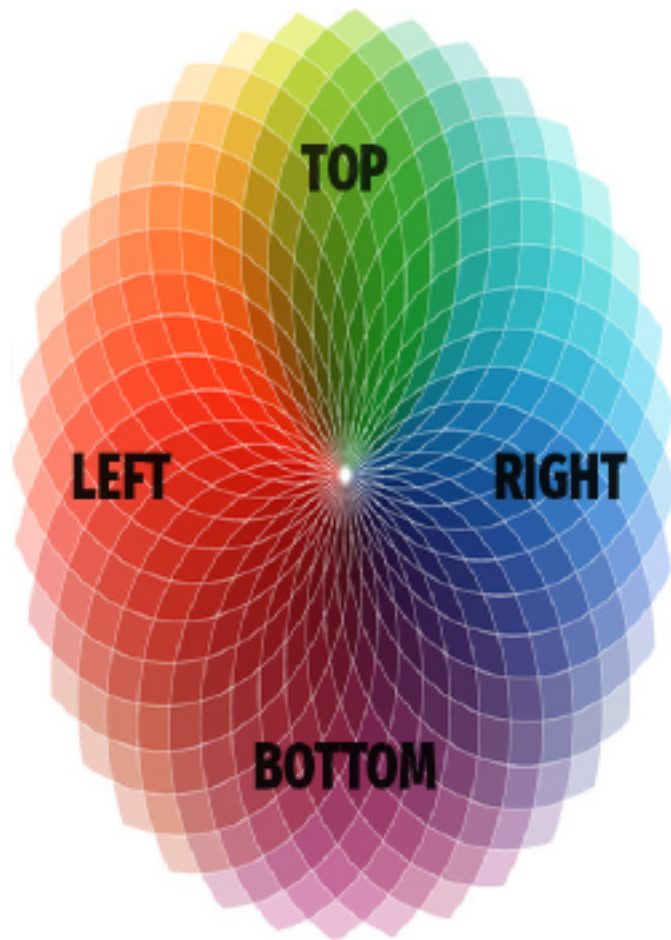NUS | iss

Interpolation methods

- Opencv provides a number of interpolation methods for geometric transformation

- Some of the common used methods
  ```
  cv2.INTER_AREA
  cv2.INTER_LINEAR
  cv2.INTER_CUBIC
  cv2.INTER_NEAREST
  ```

- 'NEAREST' has the fastest operation, but output is crude

- 'AREA' is good for shrinking image; 'CUBIC' good for enlarging image. But both operations are slow

- 'LINEAR' is somewhere in between

```
cv2.resize(src,
        dsize,
        interpolation=cv2.INTER_AREA)
```

NUS | iss

Enlarging image

- Enlarging image rarely give satisfying output; avoid if possible

- Plan your processing flow carefully; keep the original source in your flow if possible

```
> sml = cv2.resize(src,
                   dsize=(150,150),
                   interpolation=cv2.INTER_LINEAR)


> bg1 = cv2.resize(sml,
                   dsize=(500,500),
                   interpolation=cv2.INTER_NEAREST)


> bg2 = cv2.resize(sml,
                   dsize=(500,500),
                   interpolation=cv2.INTER_CUBIC)
```
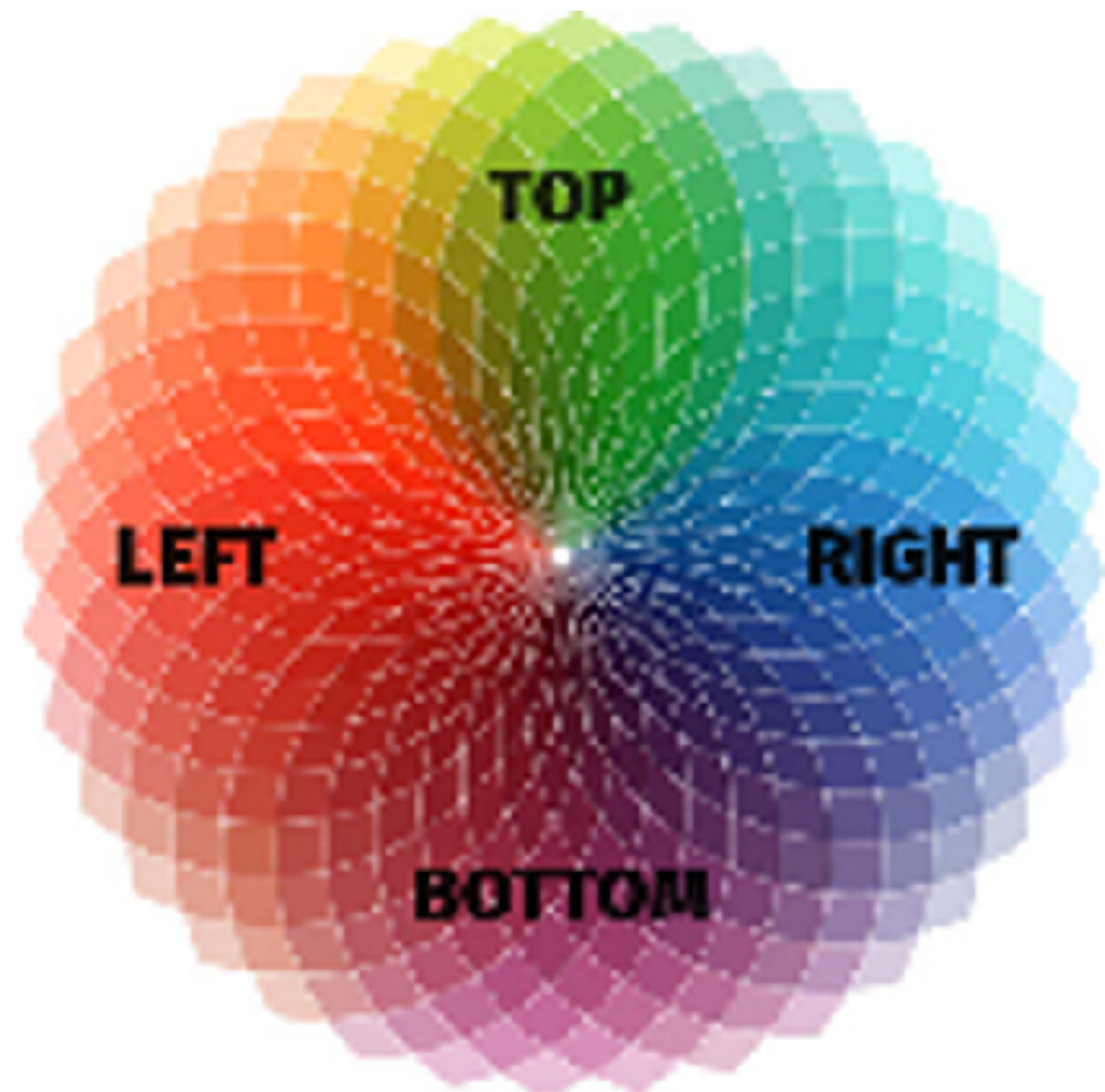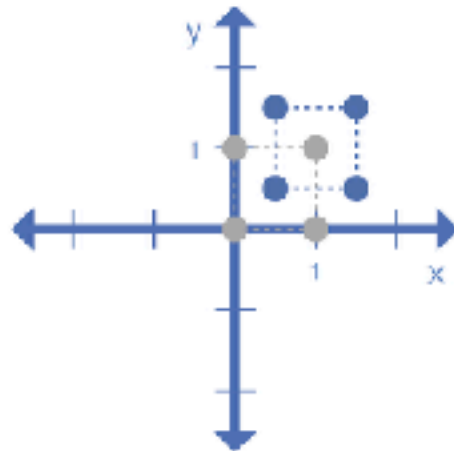

sml

NUS | iss
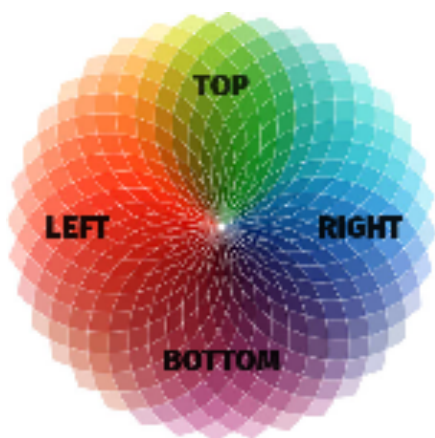
Enlarging image
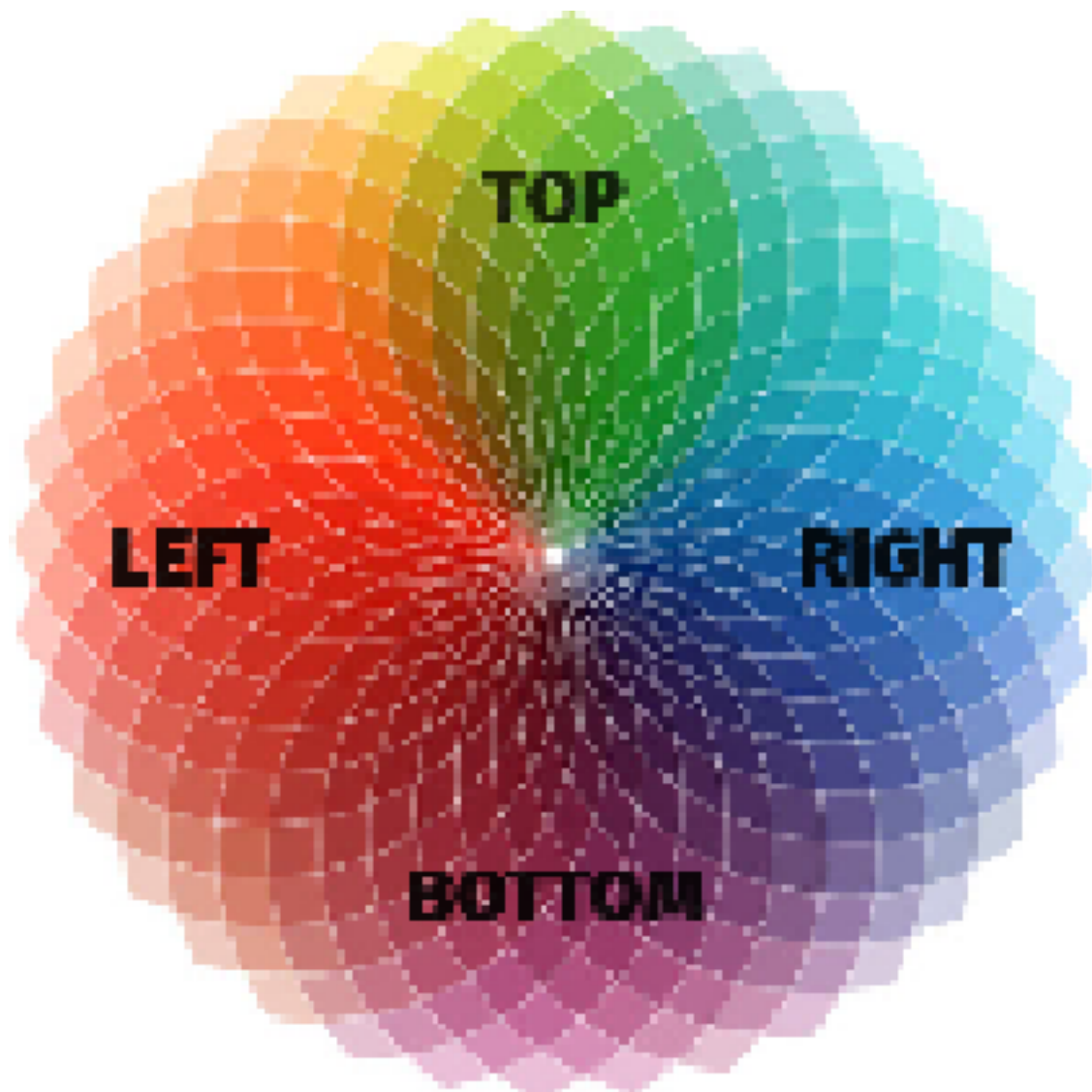
nearest

cubic

Affine transformation

- Mathematically, scaling, translating, rotating an image are considered as affine transformation

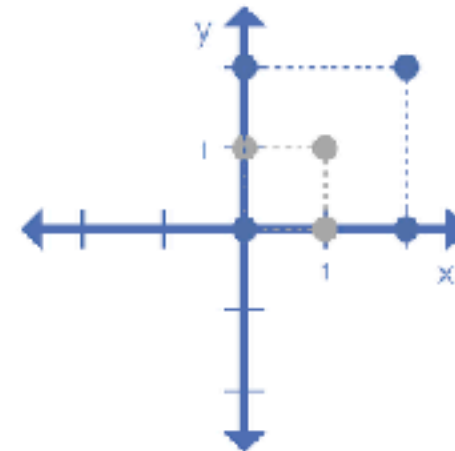- These transformations is ruled by a single matrix

Translate

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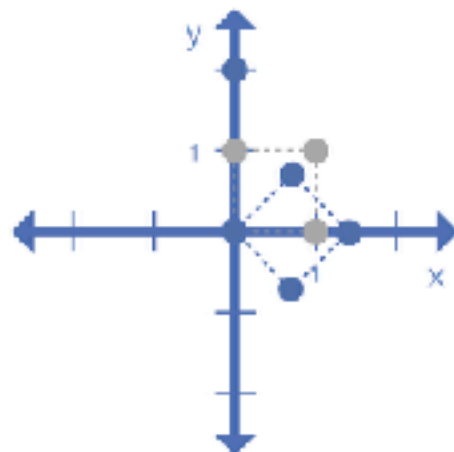
Scale

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$c = s = \sin(45°)$

Shear

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

neutrium.net

Source: https://neutrium.net/mathematics/basics-of-affine-transformation/

NUS | iss

Affine transformation

• In the matrix the last row is always having the values [0, 0, 1] for image processing purpose

• Thus in opencv, the last row is removed, and the ruling matrix is of this form

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{tx} \\ a_{21} & a_{22} & a_{ty} \end{bmatrix}$$

• Except for translation, we generally do not manually calculate each value in the matrix

• We try to use function to get the required transformation matrix

Translate

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

Scale

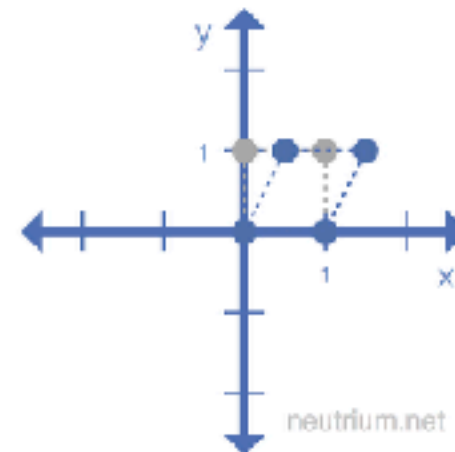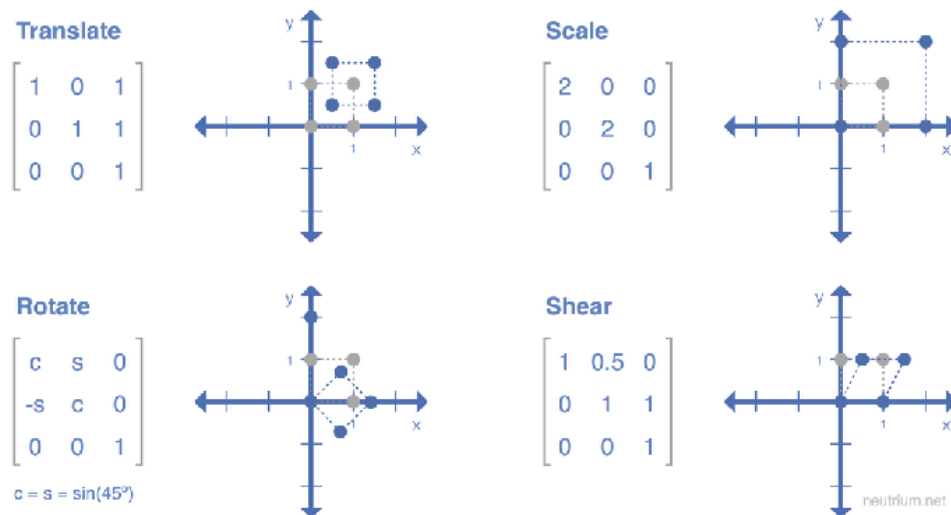$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

c = s = sin(45°)

Shear

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

neutrium.net

Source: https://neutrium.net/mathematics/basics-of-affine-transformation/

NUS iss

Translation

- For the translation, the transformation matrix looks like this

$$M = \begin{bmatrix} 1 & 0 & a_{tx} \\ 0 & 1 & a_{ty} \end{bmatrix}$$

- This assumes only translation operation is applied; no scaling or rotation is performed.

- a_{tx} stands for the amount of distance for the image to be shifted along x axis (in pixels)
  - positive value shifts the image to the right, negative to the left

- Similarly, a_{ty} stands for the shift in y axis
  - positive value shifts the image to the bottom, negative to the top

Translate

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

NUS iss

Translation

tr
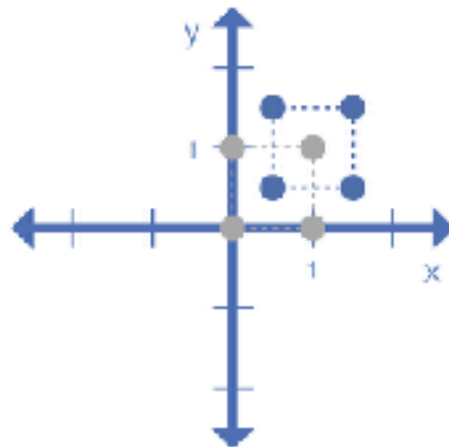
- To translate an image in opencv, the first thing we do is to create the transformation matrix

a_{tx}

```
> Mt = np.float32([[1, 0, 0],
                   [0, 1, 25]])
```
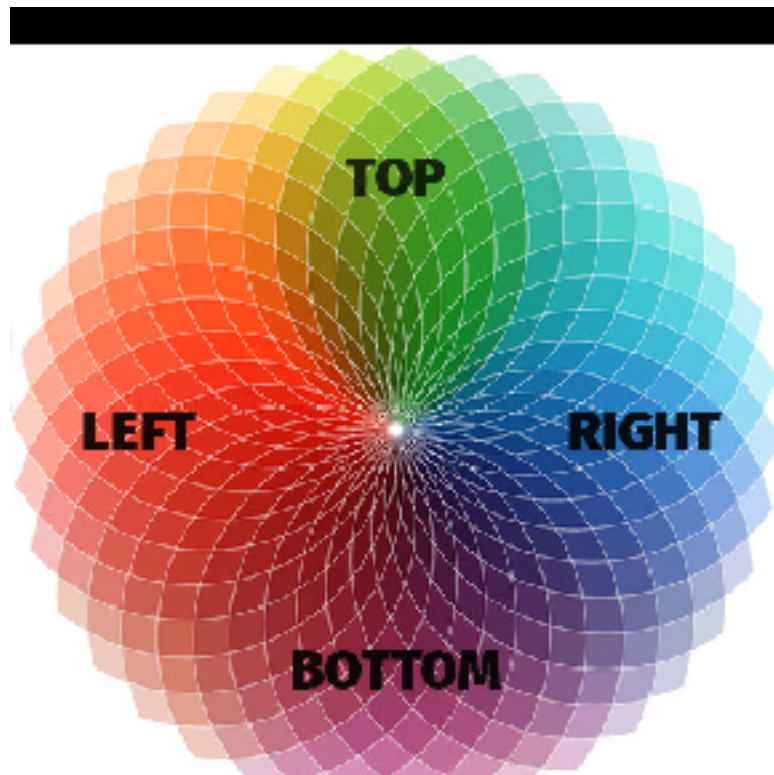
a_{ty}

- Must use `np.float32` to create the transformation matrix, else error

- `Mt` shifts an image downward by 25 pixels

```
> tr = cv2.warpAffine(src,
       transformation matrix  Mt,
              output size  (501,501))
                           height  width
```

Rotation

- To create the transformation matrix for rotation, we do

```
> (row,clm,_) = src.shape
```

```
                                    x        y
> Mr = cv2.getRotationMatrix2D((clm/2,row/2),  rotation center
                                90,  rotation angle
                                1)   scale
```

- Opencv provides the function to generate transformation matrix for rotation about a center point in an image

- The rotation angle is in degree, positive value for anti-clockwise, neative value for clockwise

```
> rt = cv2.warpAffine(src,
    transformation matrix  Mr,
            output size  (501,501))
                         height  width
```

RIGHT

TOP

BOTTOM

LEFT

rt

NUS | iss
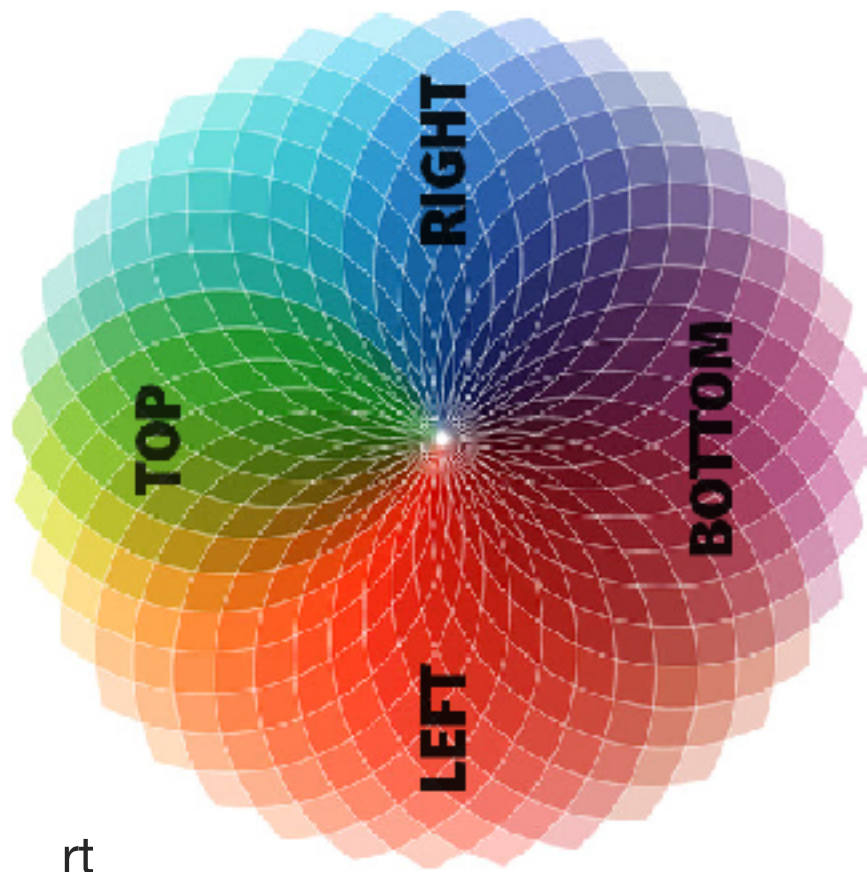
A very important use case
On geometric transformation

Source: https://rock-it.pl/images-augmentation-for-deep-learning-with-keras/

vse/m2.2/v1.2
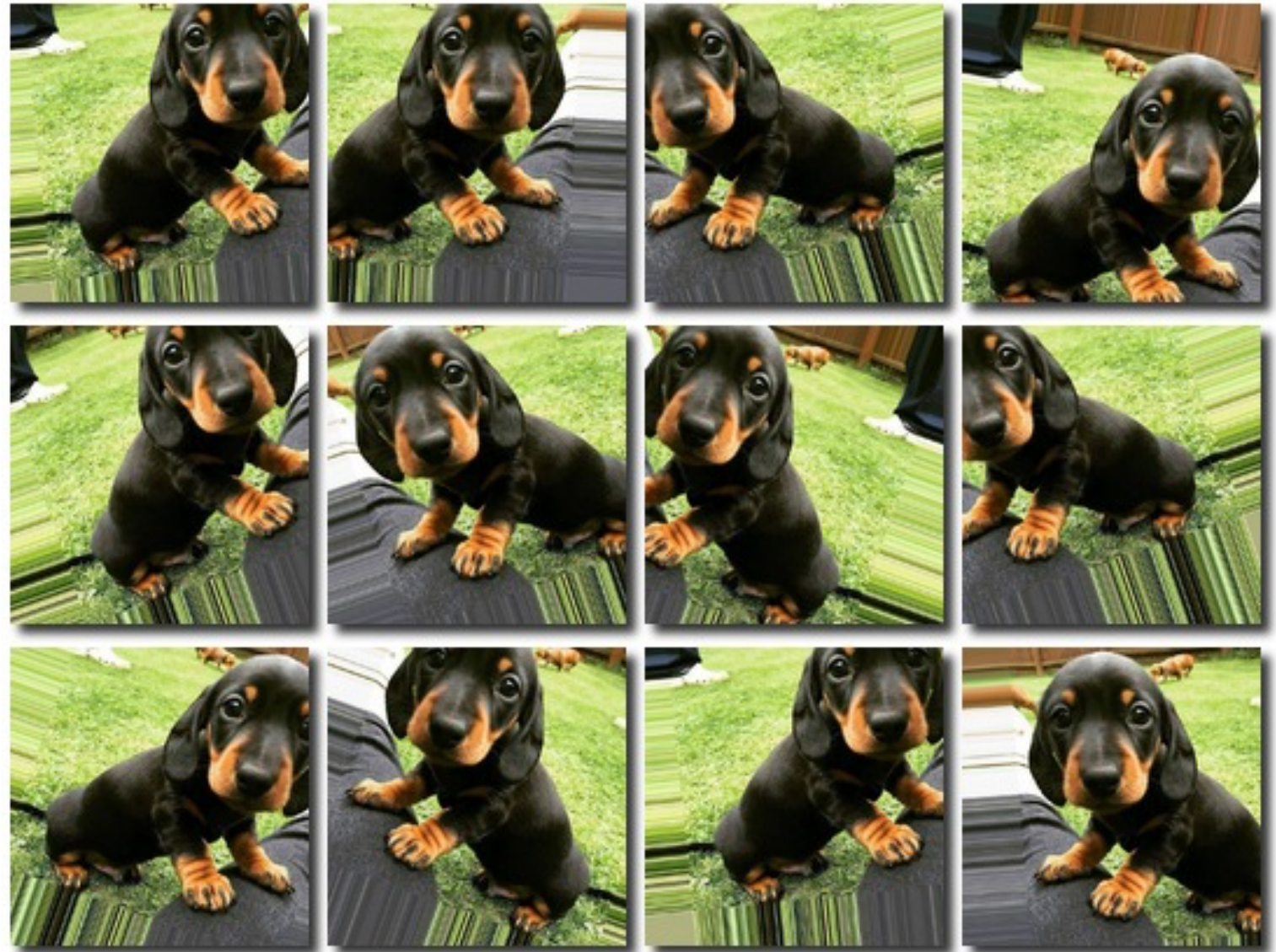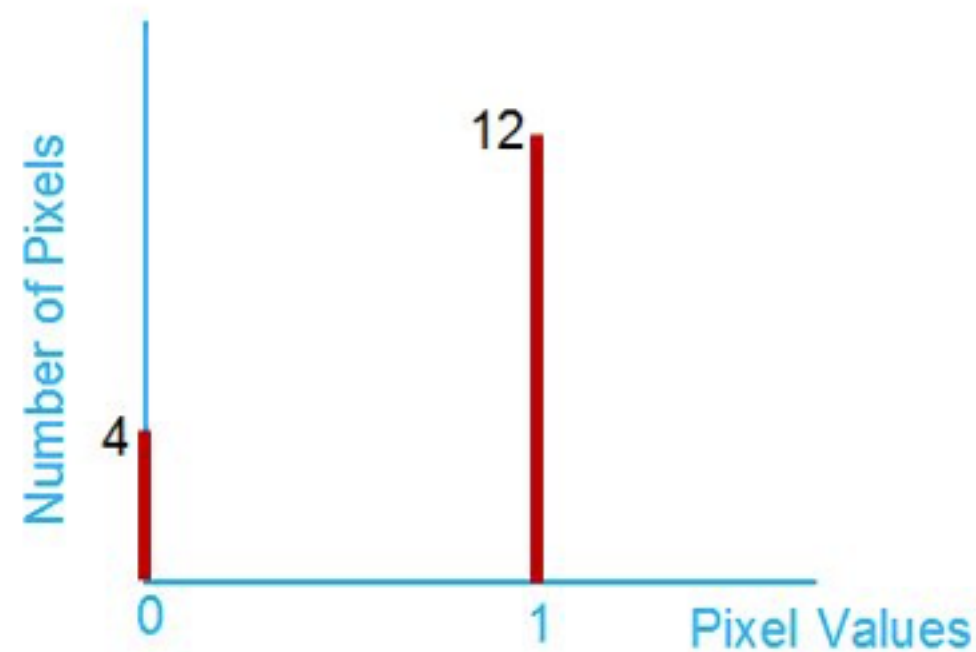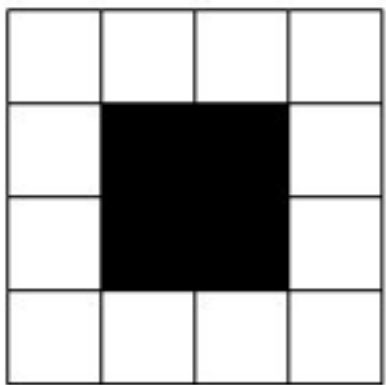
Image histogram

Image histogram

- Image histogram: a graph of pixel intensity (along x-axis) vs number of pixels (y-axis)

- x-axis has all the intensity levels

- y-axis indicates the amount of pixels in image that have the particular intensity-level value

1-bit

2-bit

3-bit

4-bit

vse/m2.2/v1.2

Image histogram

- 4 x 4 monochrome image (black or white) image

- Count the number of black pixels (there are 4)

- Count the number of white pixels (there are 12)
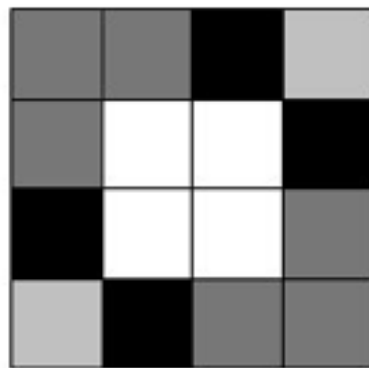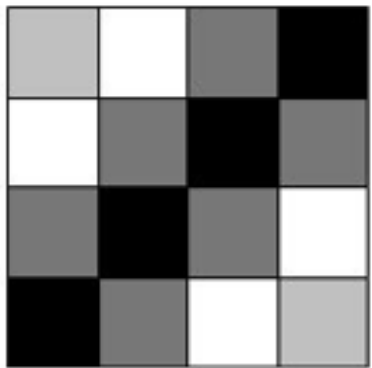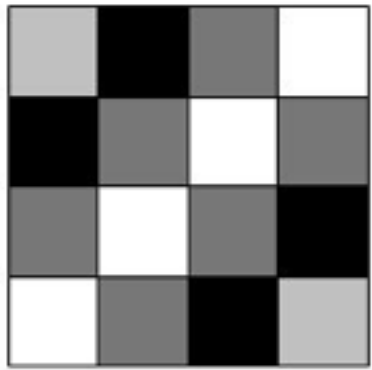
vse/m2.2/v1.2

Image histogram

- 4 x 4 gray scale image

- Gray scale image are generally 8-bit, and thus it has 256 levels

- Note: in image processing, we sometimes group several levels together as one bin

Source: https://www.allaboutcircuits.com/technical-articles/image-histogram-characteristics-machine-learning-image-processing/
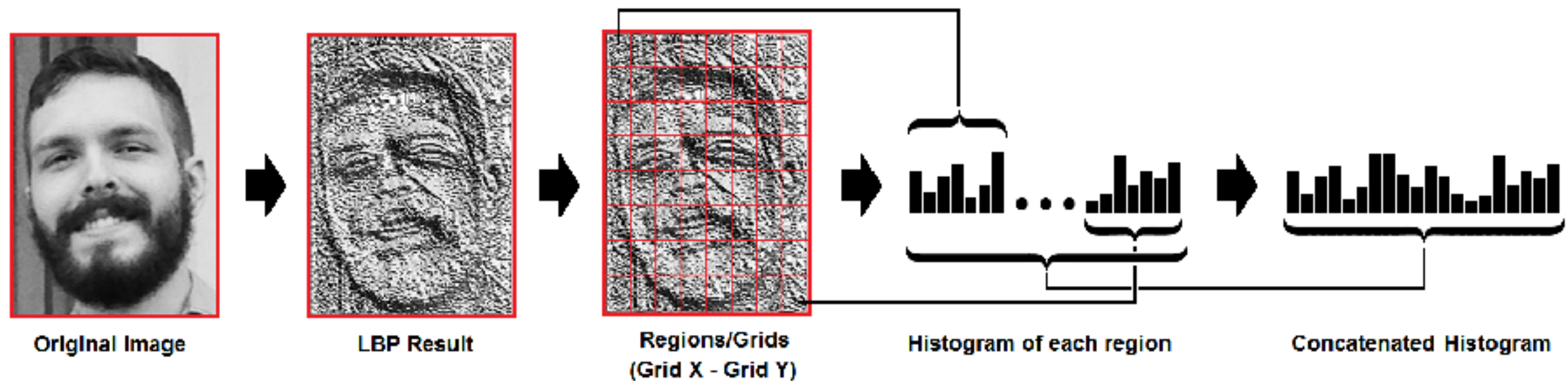
vse/m2.2/v1.2
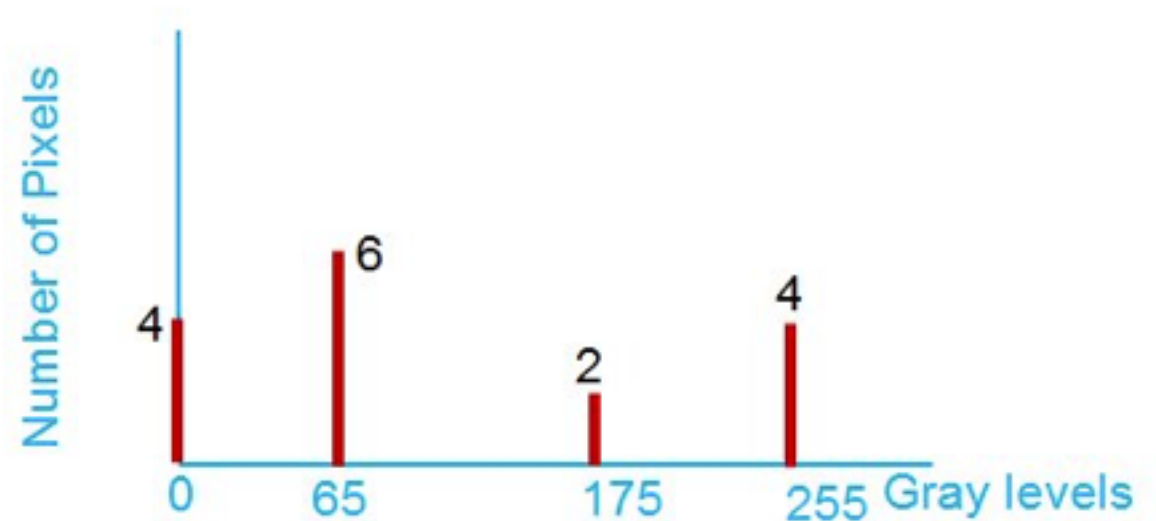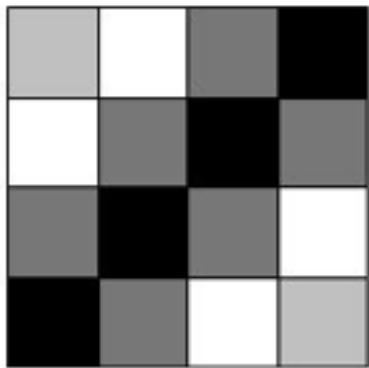
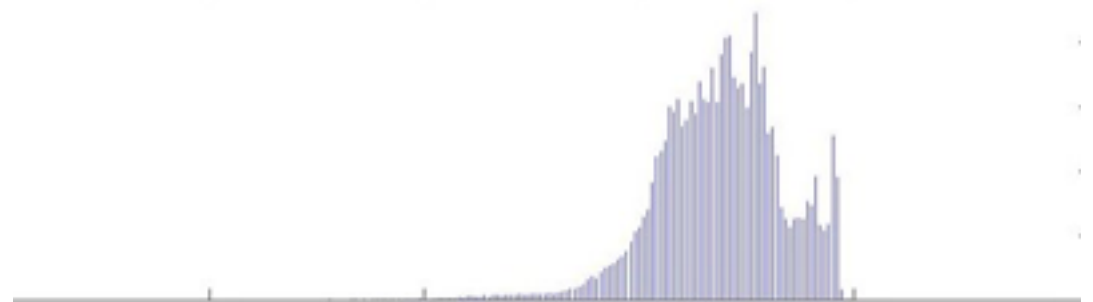# Image histogram

- Problem with histogram: provides only statistical information

- Zero information about the spatial distribution of the pixel values
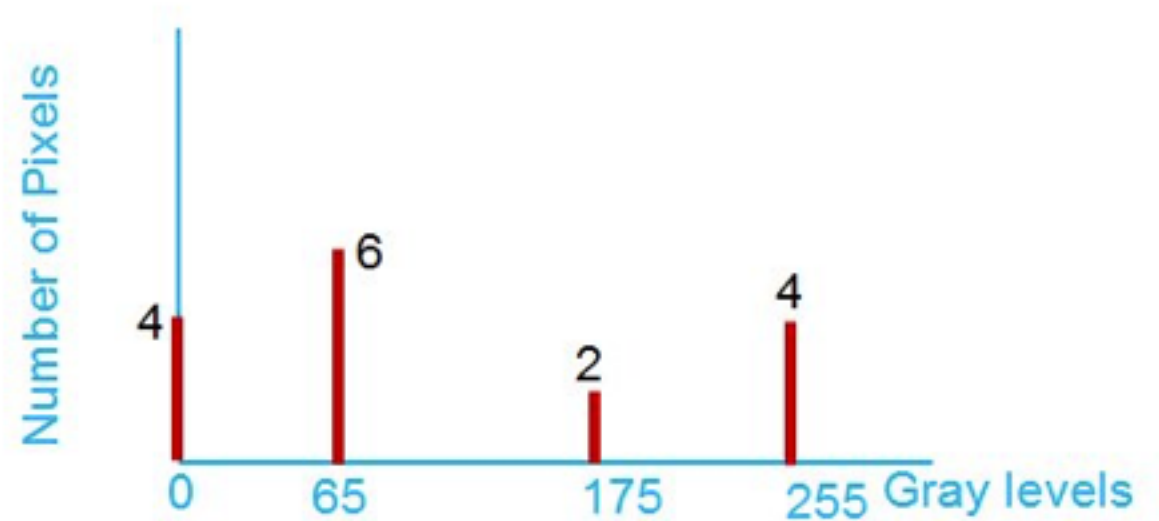
Image histogram

Original Image → LBP Result → Regions/Grids (Grid X - Grid Y) → Histogram of each region → Concatenated Histogram

Source: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

Image histogram

• Image of high contrast and low contrast

Source: https://www.allaboutcircuits.com/technical-articles/image-histogram-characteristics-machine-learning-image-processing/

Image histogram

kwg.jpg

- Let us load an image

```
> kwg   = cv2.imread('kwg.jpg')
```

- Check the shape

```
> kwg.shape
: (700, 477, 3)
```

- Grayscale image by right, should have only one channel

- However, many times grayscale images are still saved in 3 channels

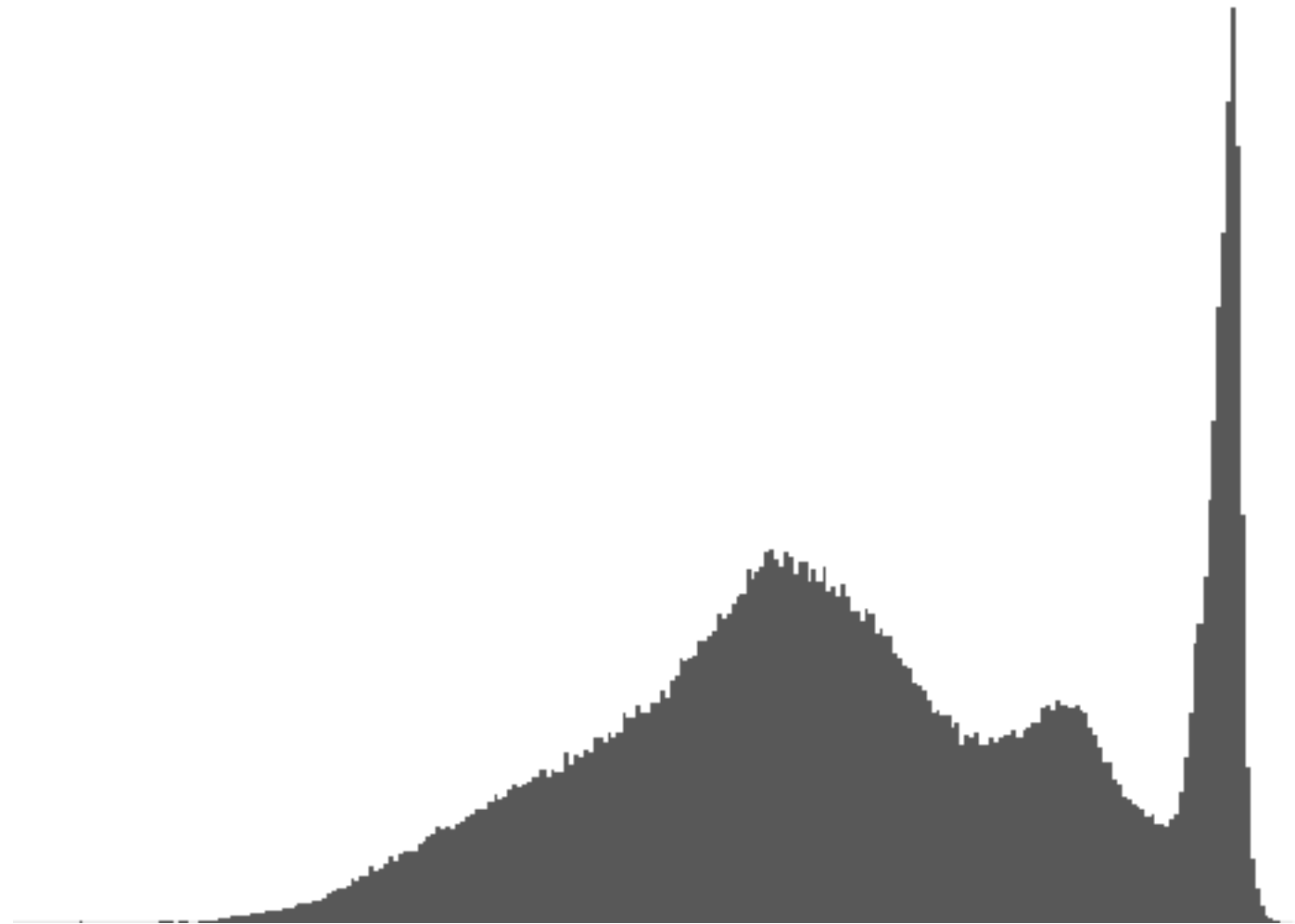- To retrieve just one channel for processing, we do

```
> kwg   = kwg[:,:,0]
```

Image histogram

```
> plt.hist(kwg.ravel(),
  number of bins   256,
          range    [0,255],
  colour of the bar  facecolor='black')
```


kwg.jpg

NUS | iss
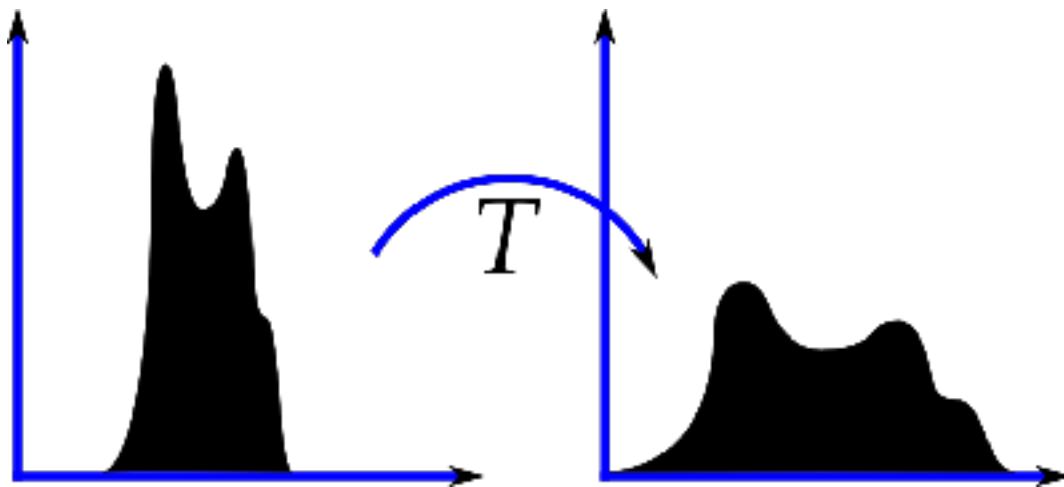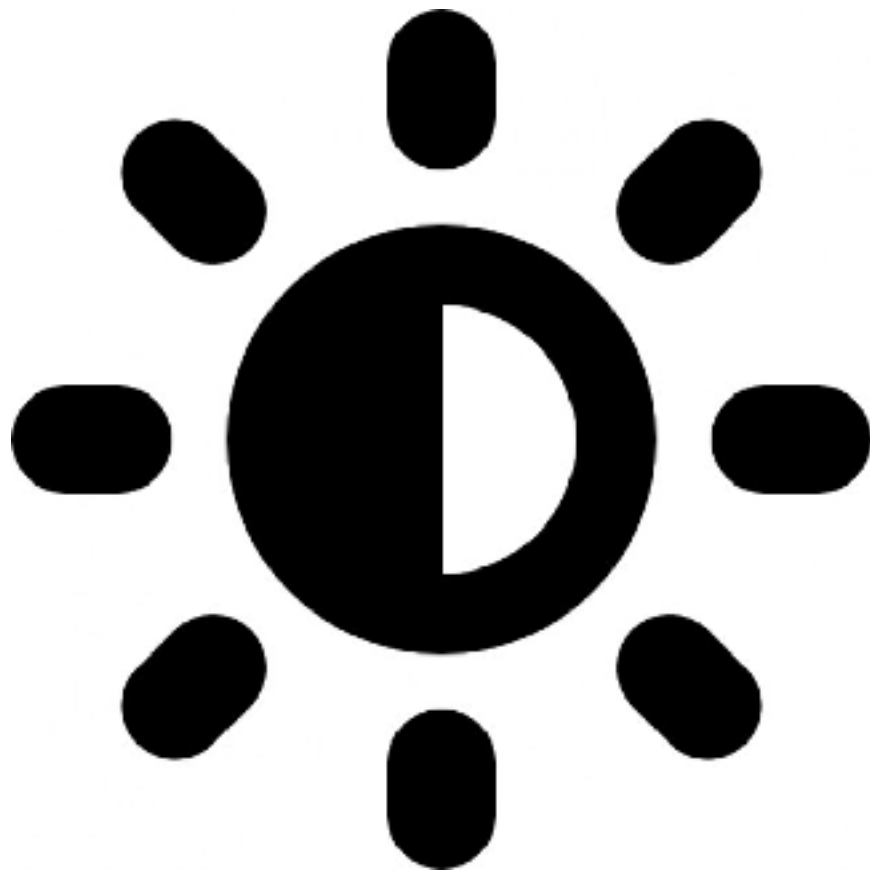
Histogram equalization

- Histogram equalization: a method to adjust contrast in an image

- Achieved by spreading out most frequent intensity values

- Often produce unrealistic effects in photographs, but useful for scientific images

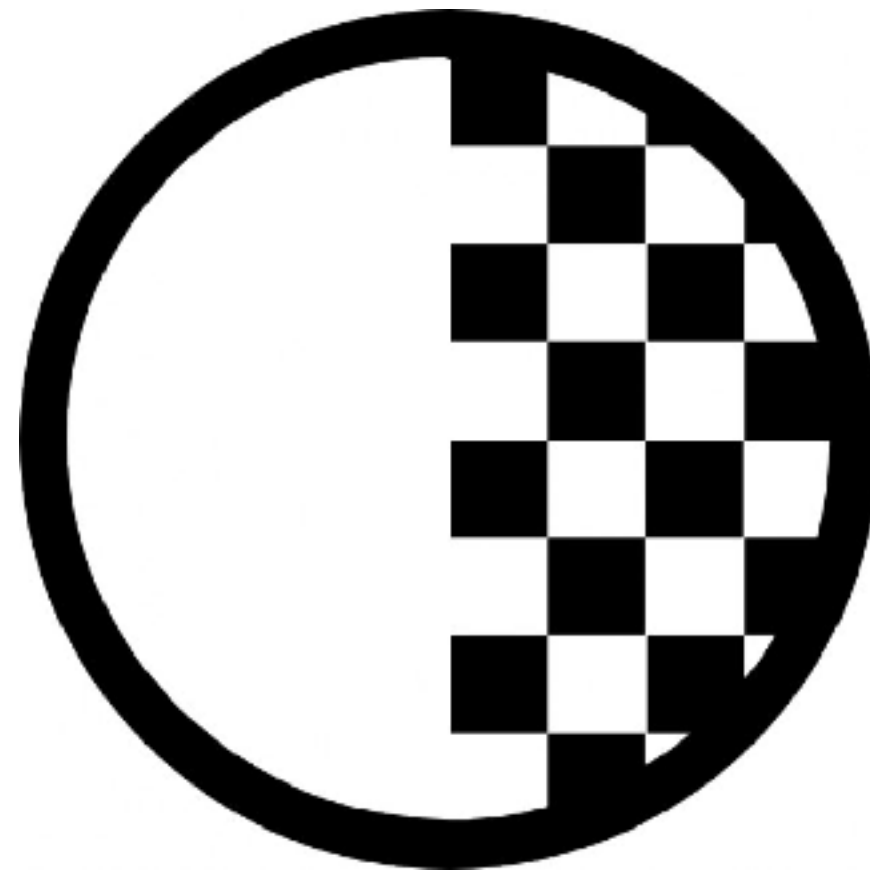- Calculation is not computationally expensive

Source: https://commons.wikimedia.org/wiki/
File:Histogrammspreizung.png

vse/m2.2/v1.2

Why do we need to adjust contrast?
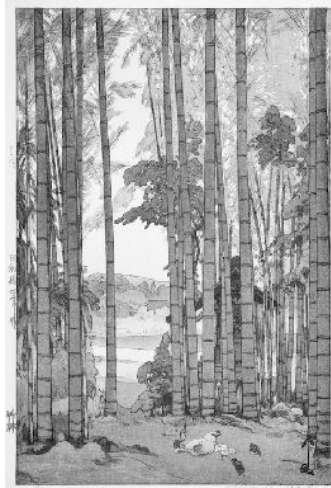
vse/m2.2/v1.2

Histogram equalization

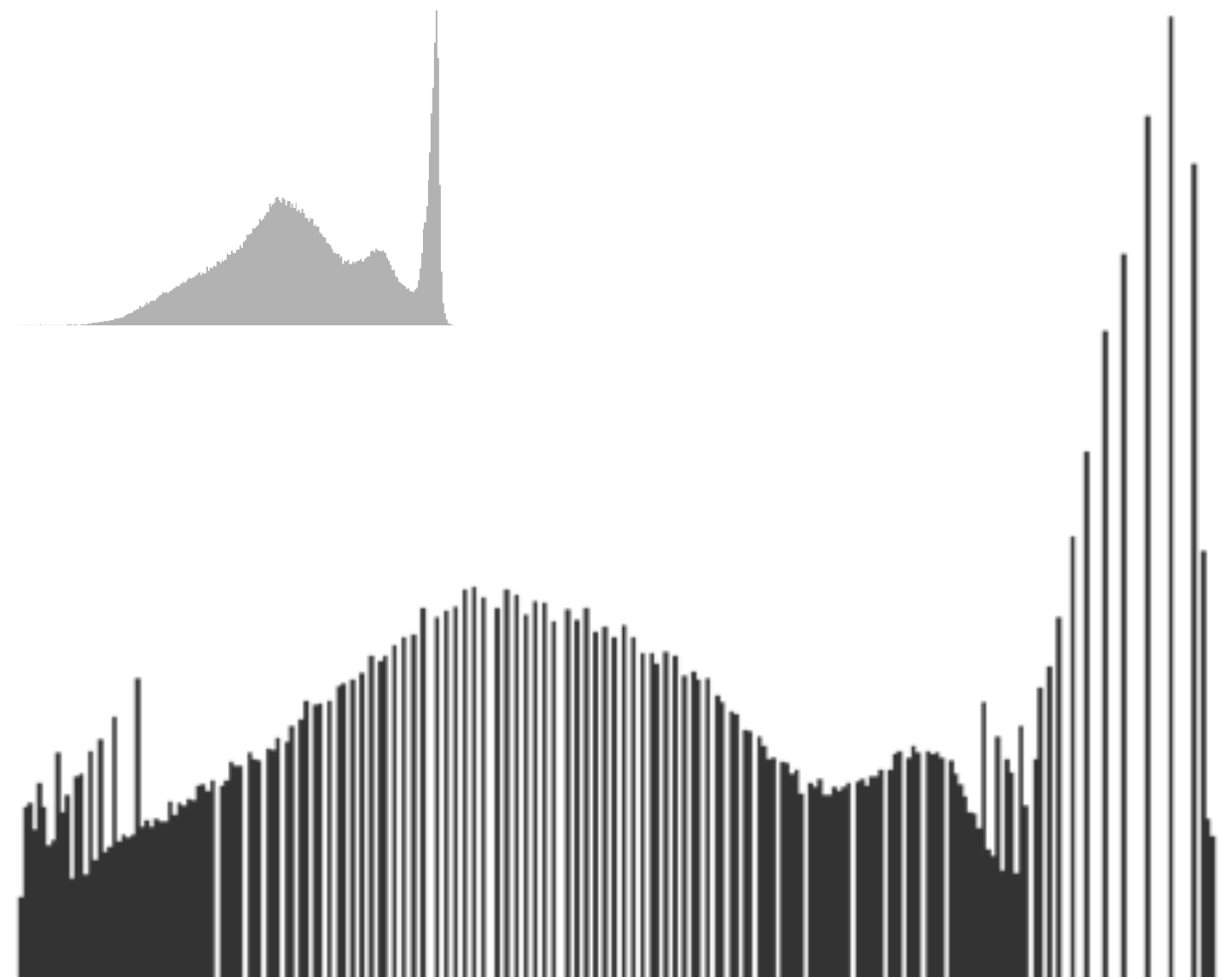- To perform histogram equalization, simply do

```
> heq = cv2.equalizeHist(kwg)
> plt.hist(heq.ravel(),
    number of bins  256,
          range  [0,255],
  colour of the bar  facecolor='black')
```


kwg

heq
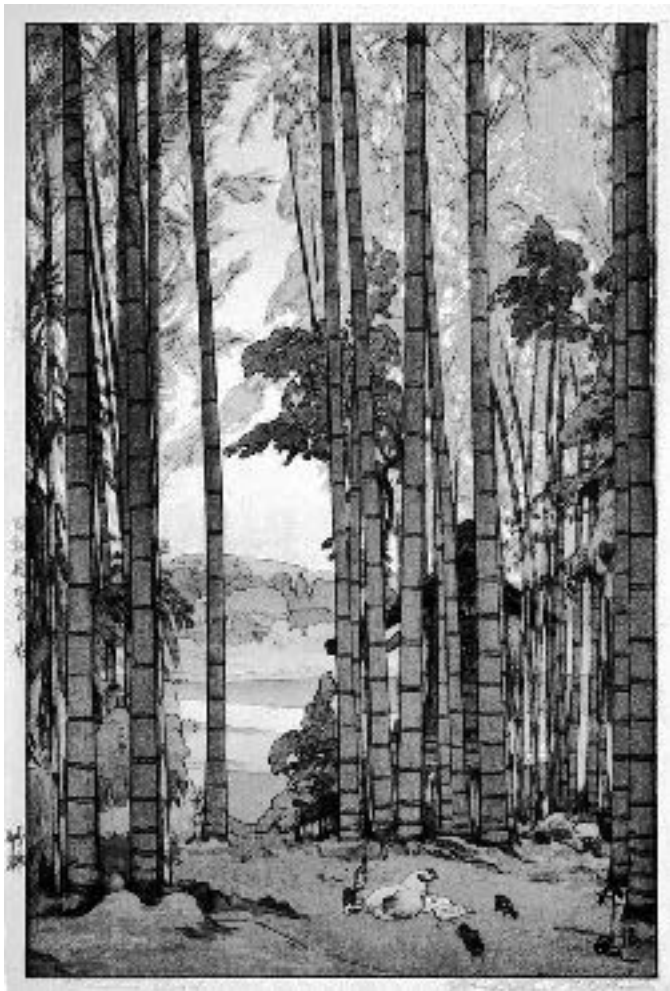
Contrast limited adaptive histogram equalization (CLAHE)
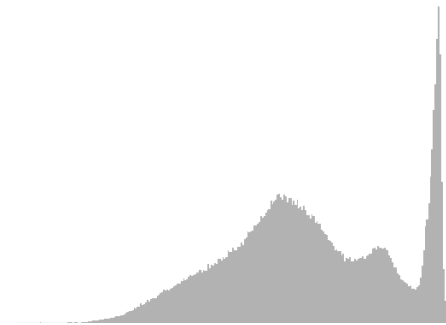
```
> clahe  = cv2.createCLAHE()    create CLAHE object
> hcl    = clahe.apply(kwg)     use the object to
> plt.hist(hcl.ravel(),         perform equalization
            256,
            [0,255],
            facecolor='black')
```


kwg

hcl

Comparison

Histogram equalization

CLAHE
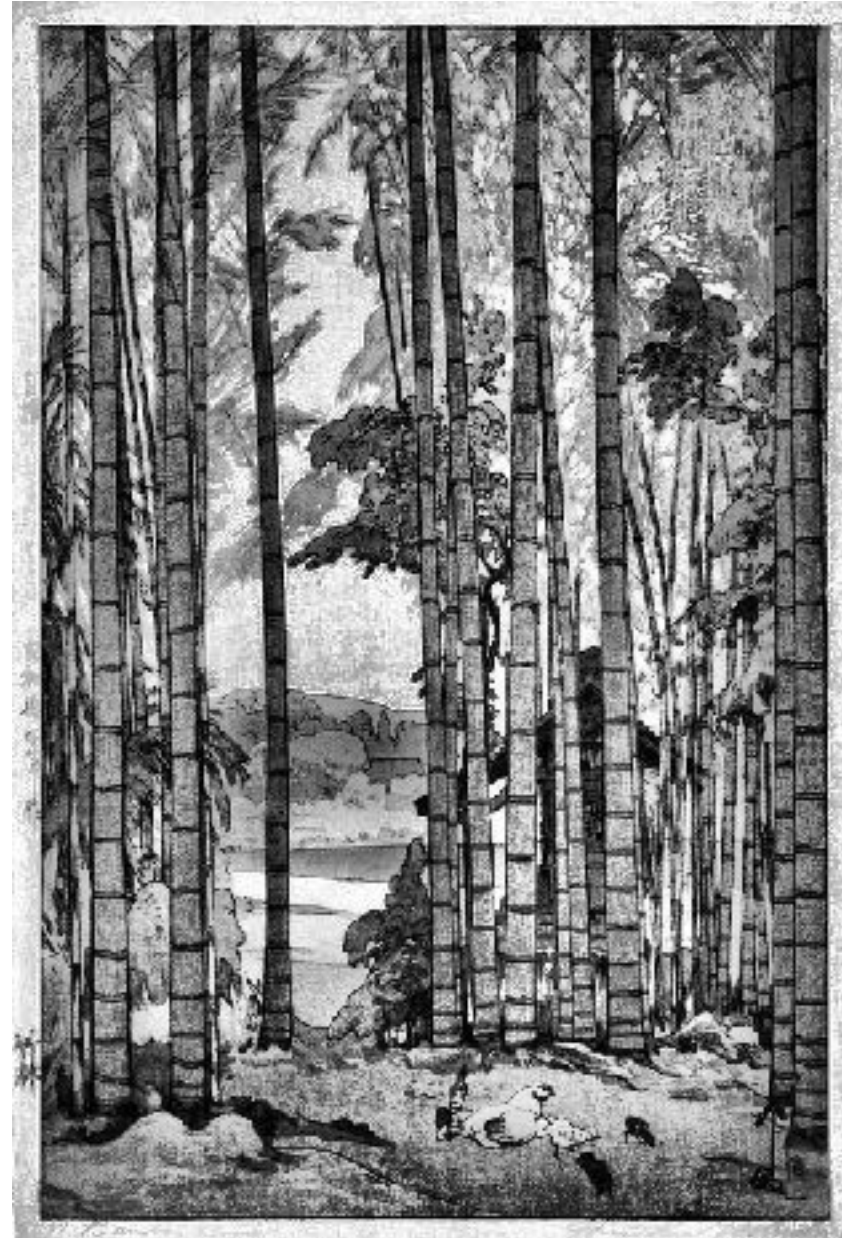
vse/m2.2/v1.2

NUS | iss

Image thresholding

- Image thresholding: a process that converts intensity values that exceeds/below a threshold to one value, and the rest as other value(s)

- Usually we use 1 or 255 to denote the pixels/points that we want, and 0 for points/pixels to be ignored/ discarded

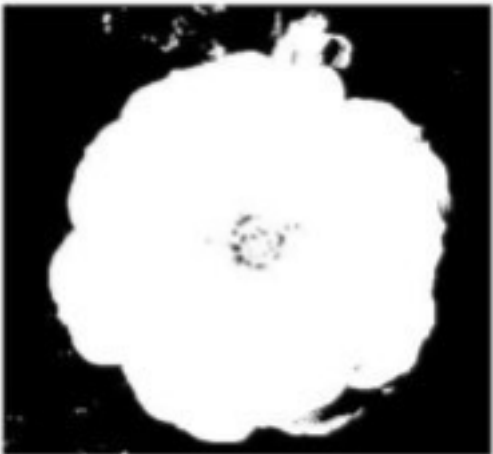- This is often used to isolate region of interest

Source: http://www.ni.com/tutorial/2916/en/

NUS | iss
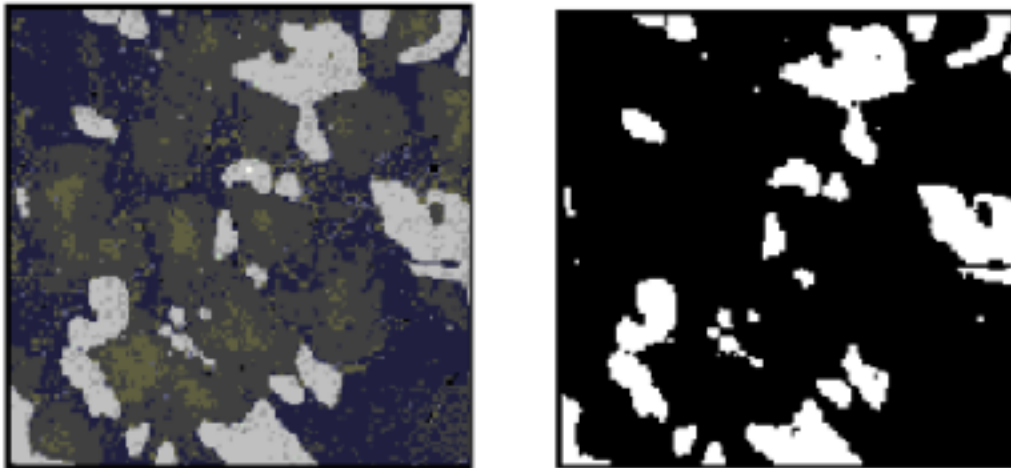
Image thresholding

• When we do thresholding, we are in fact performing segmentation using the image histogram

original image

thresholded image

vse/m2.2/v1.2

Image thresholding

- Opencv provides several methods to do thresholding

```
> gr       = cv2.imread('gr.png')
> (_,thr) = cv2.threshold(gr[:, :, 0],
         threshold value  85,
         assigned value  255,
         threshold style  cv2.THRESH_BINARY)
```

- Thresholding can be easily implemented in simple python; just one/two for loops will do

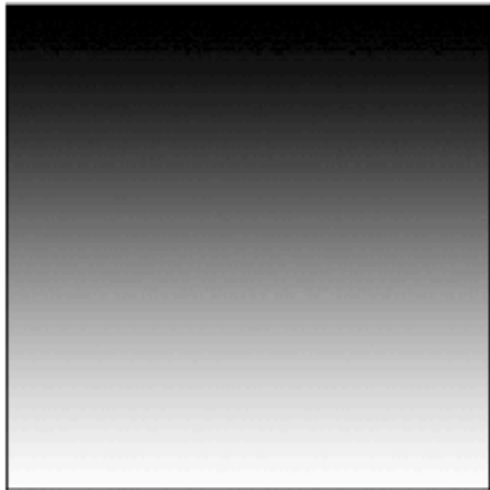- But that will be slow; using opencv function gives faster implementation

gr.png

vse/m2.2/v1.2

NUS | iss

Thresholding style

```
cv2.threshold(gr[:, :, 0],
              85,
              255,
              ThresStyle)
```
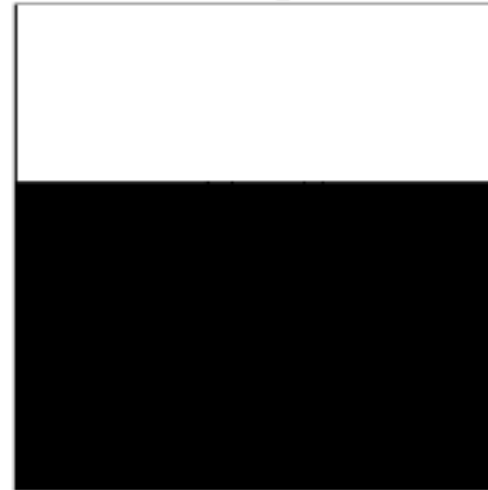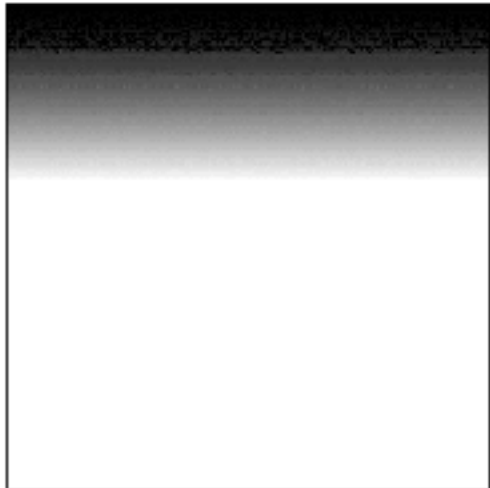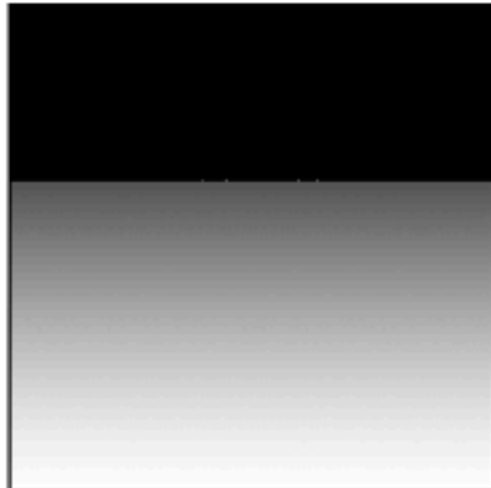

GR.PNG

BINARY

BINARY_INV

TRUNC

TOZERO

TOZERO_INV