Industrial Computer Vision

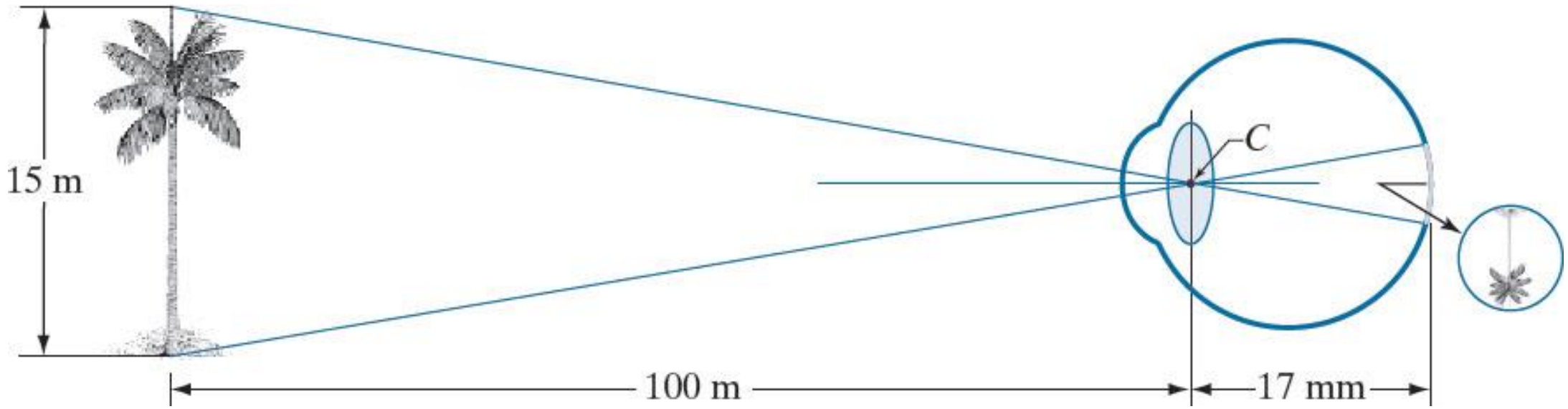
- Image Input/Output & GUI

2nd lecture, 2022.09.14

Lecturer: Youngbae Hwang

Image formation

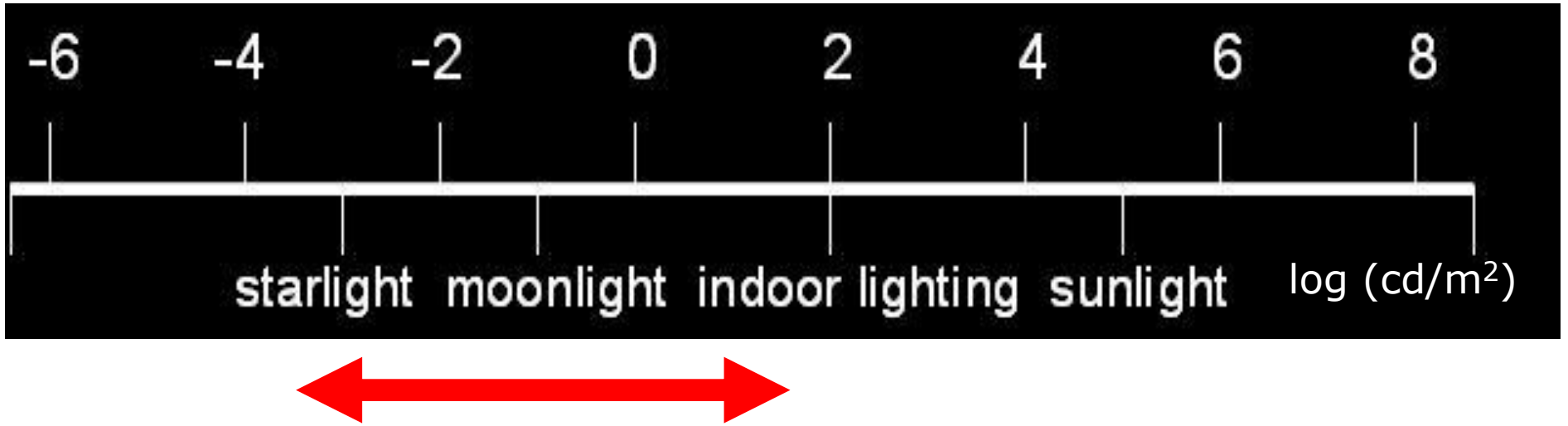
- Image formation in the eye
  - Distance between center of lens and retina (focal length) vary between 14-17 mm.
  - Image length h = 17(mm) x (15/100)



Range of human visual system

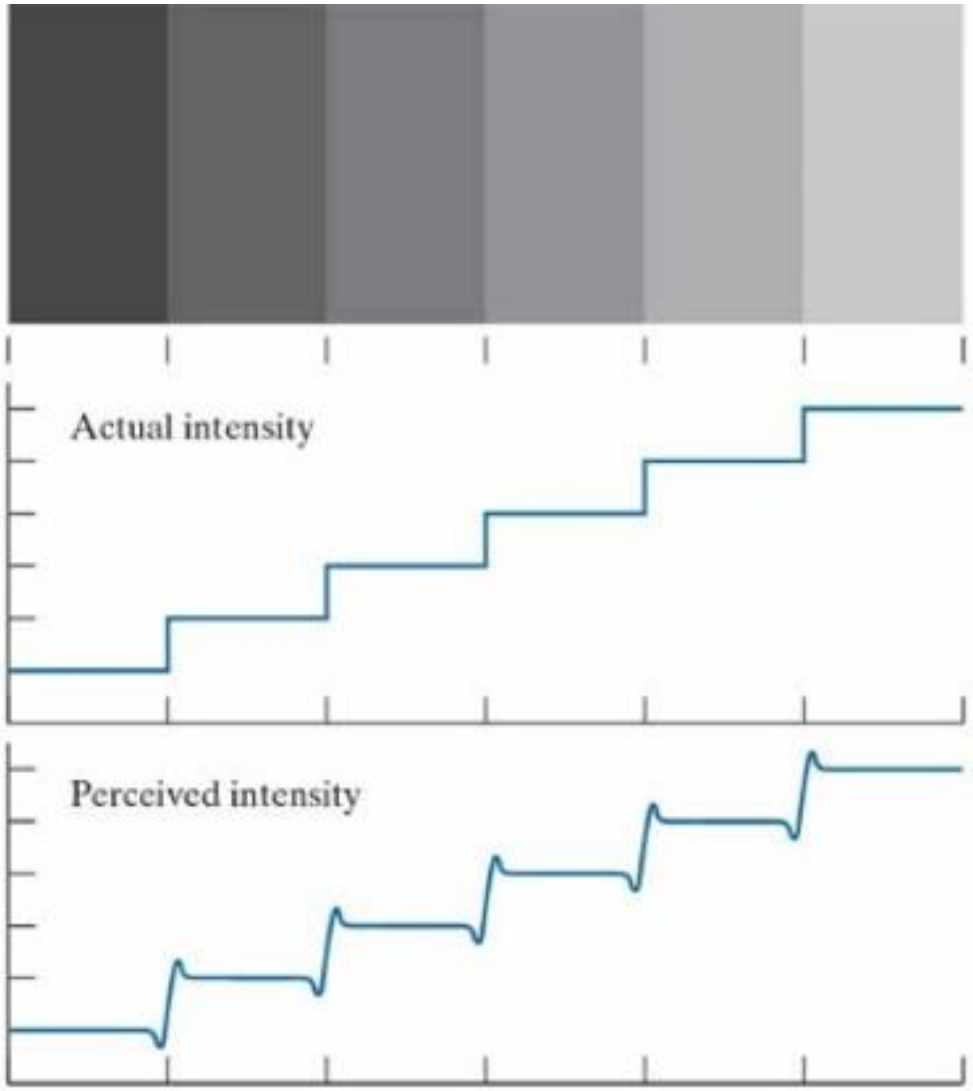


Human simultaneous luminance vision range
(5 orders of magnitude)



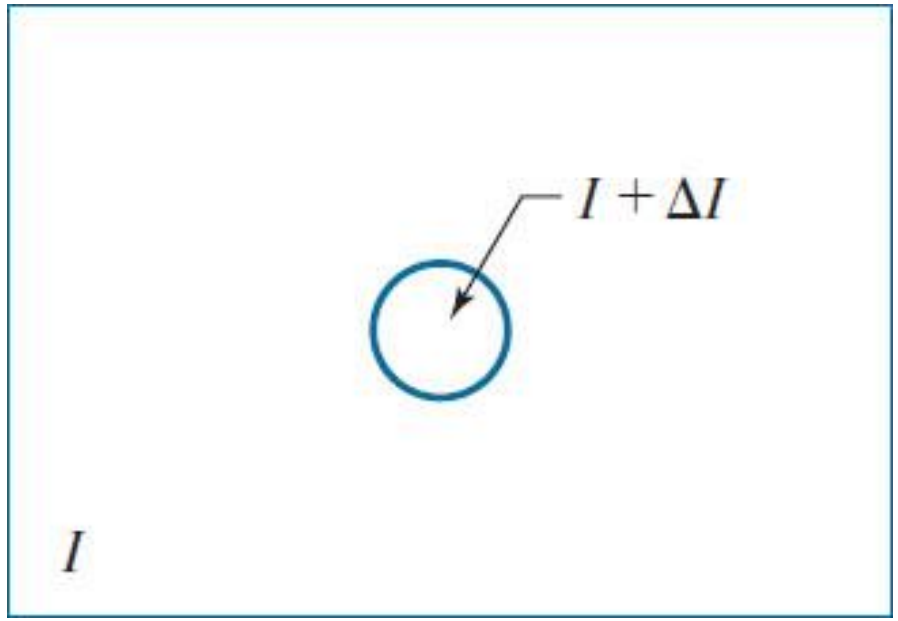
Perceived intensity

- Illustration of the Mach band effect. Perceived intensity is not a simple function of actual intensity

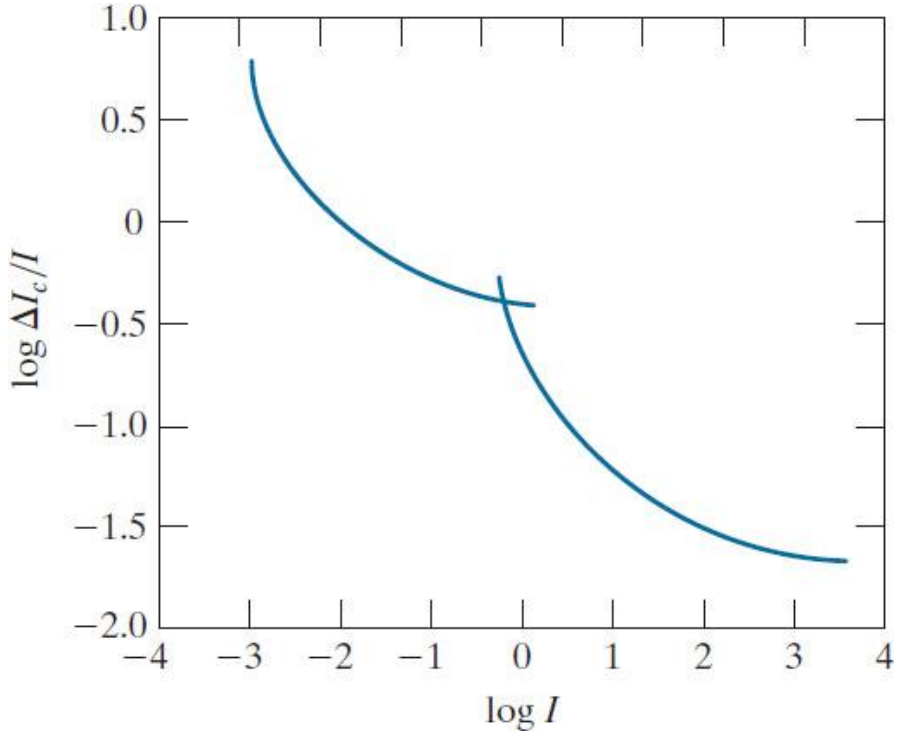


Bright discrimination

- Perceivable changes at a given adaptation level



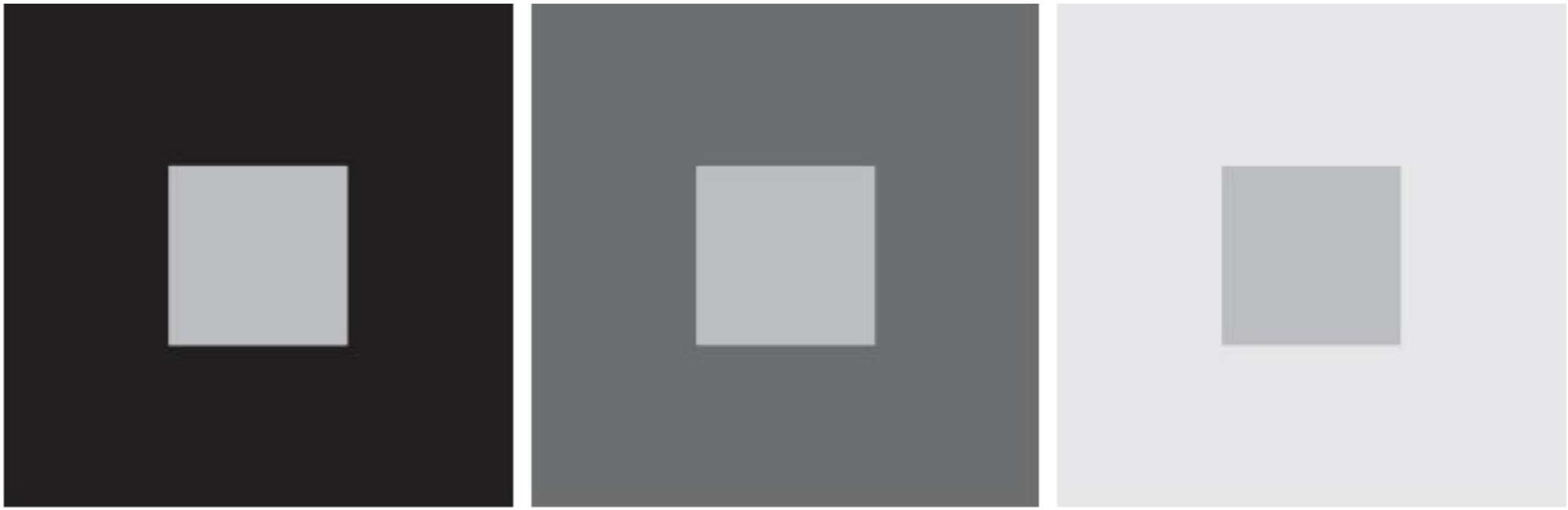
Experimental setup used
to characterize brightness discrimination



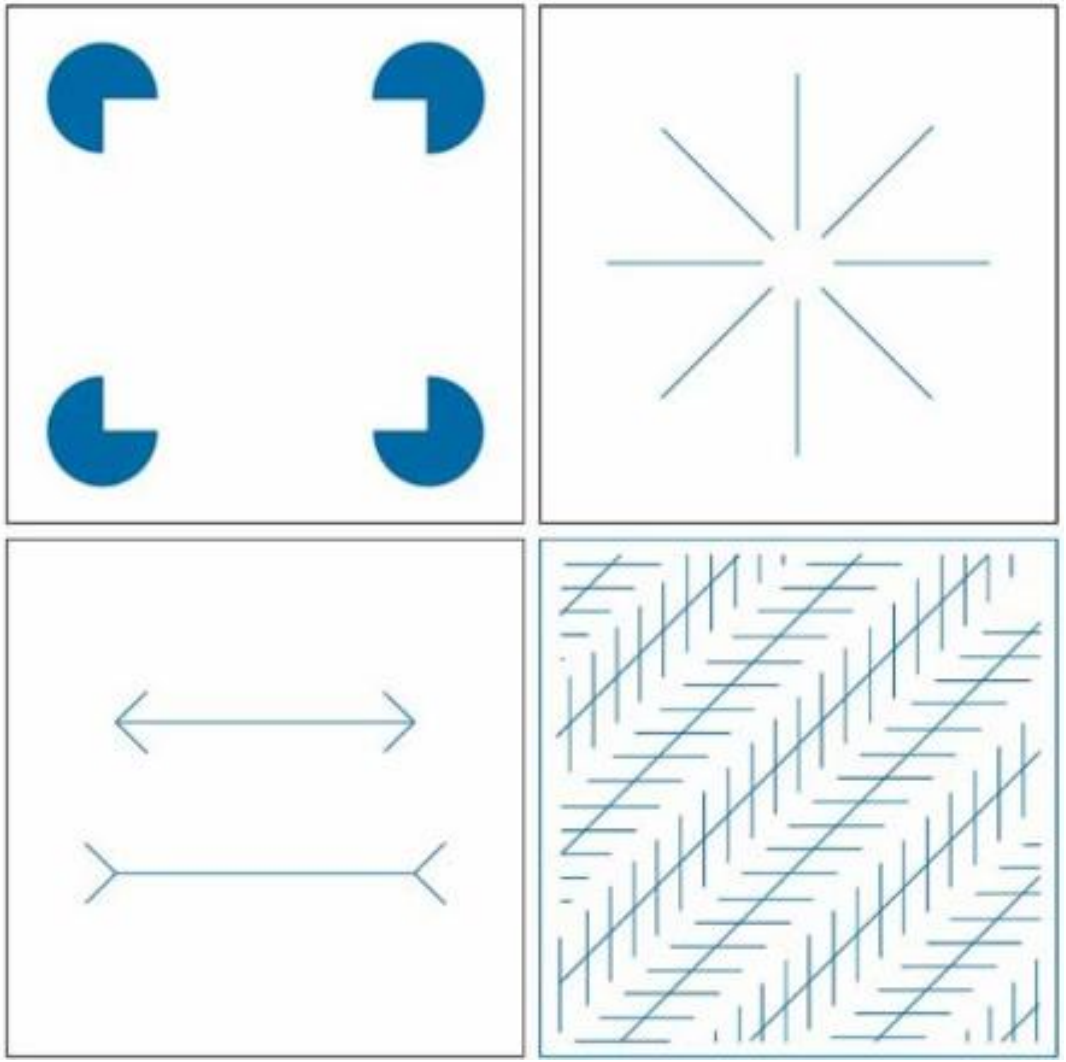
Weber ratio as a function of intensity

Simultaneous constrast

- All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

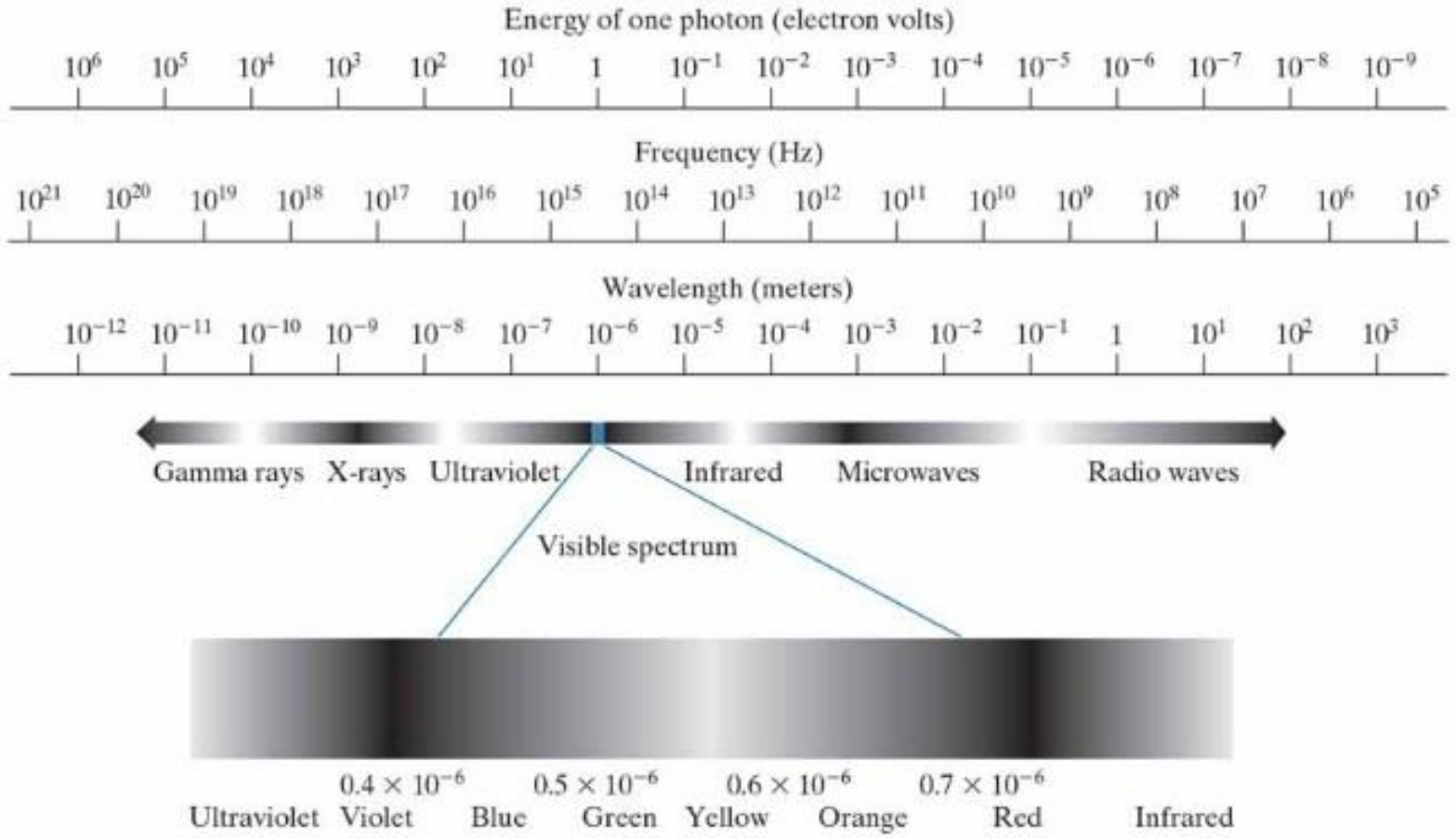


Some well-known optical illusions



Electromagnetic spectrum

- $c = \frac{\lambda}{T} = \lambda\nu \rightarrow \nu = \frac{c}{\lambda}$ $\quad \lambda_{vg} \approx 0.55\,\mu m \quad \nu = \frac{3 \cdot 10^8\, m/s}{0.55 \cdot 10^{-6} m} = \frac{3}{0.55} 10^{14} \frac{1}{s} \approx 5.45 \cdot 10^{14} Hz$
- $E = h\nu = 4.13 \cdot 10^{-15}\, eVs \cdot 5.45 \cdot 10^{14} Hz$
  $\approx 22 \cdot 10^{-1} eV = 2.2eV$



Light

- Light is a particular type of electromagnetic wave
- The colors that humans perceive are determined by the light *reflected* by the object:

  - all the light reflected: white object

  - some components (of the visible spectrum) absorbed, some reflected: color (wavelength reflected).

  - Light reflected/absorbed at the same rate for all wavelengths: *monochromatic* light.

- Thus we speak of *intensity* or *gray level*

Light

- Properties of light sources/reflected light:
  - Chromatic light (colors): from 0.43 μm to 0.79 μm wavelength
- *Radiance*: Total amount of energy out of the light source (Watts)
- *Luminance*: Amount of light perceived from a light source (lumen) ex. Stars
- *Brightness*: earlier a synonymous of luminance, is now a subjective measurement of light perceived from a light source.

Image acquisition

(a) Single sensing element.

(b) Line sensor.

(c) Array sensor.

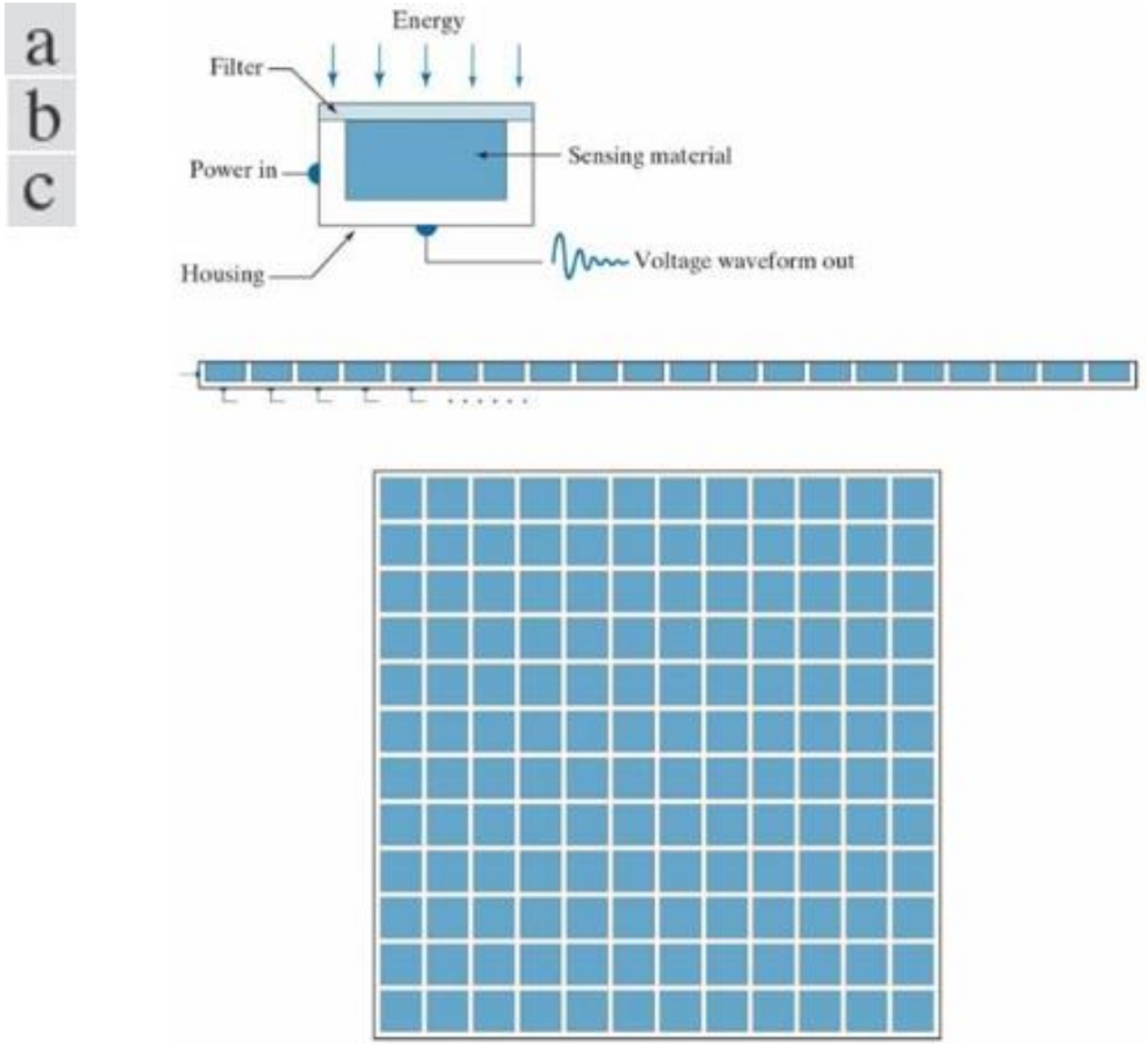
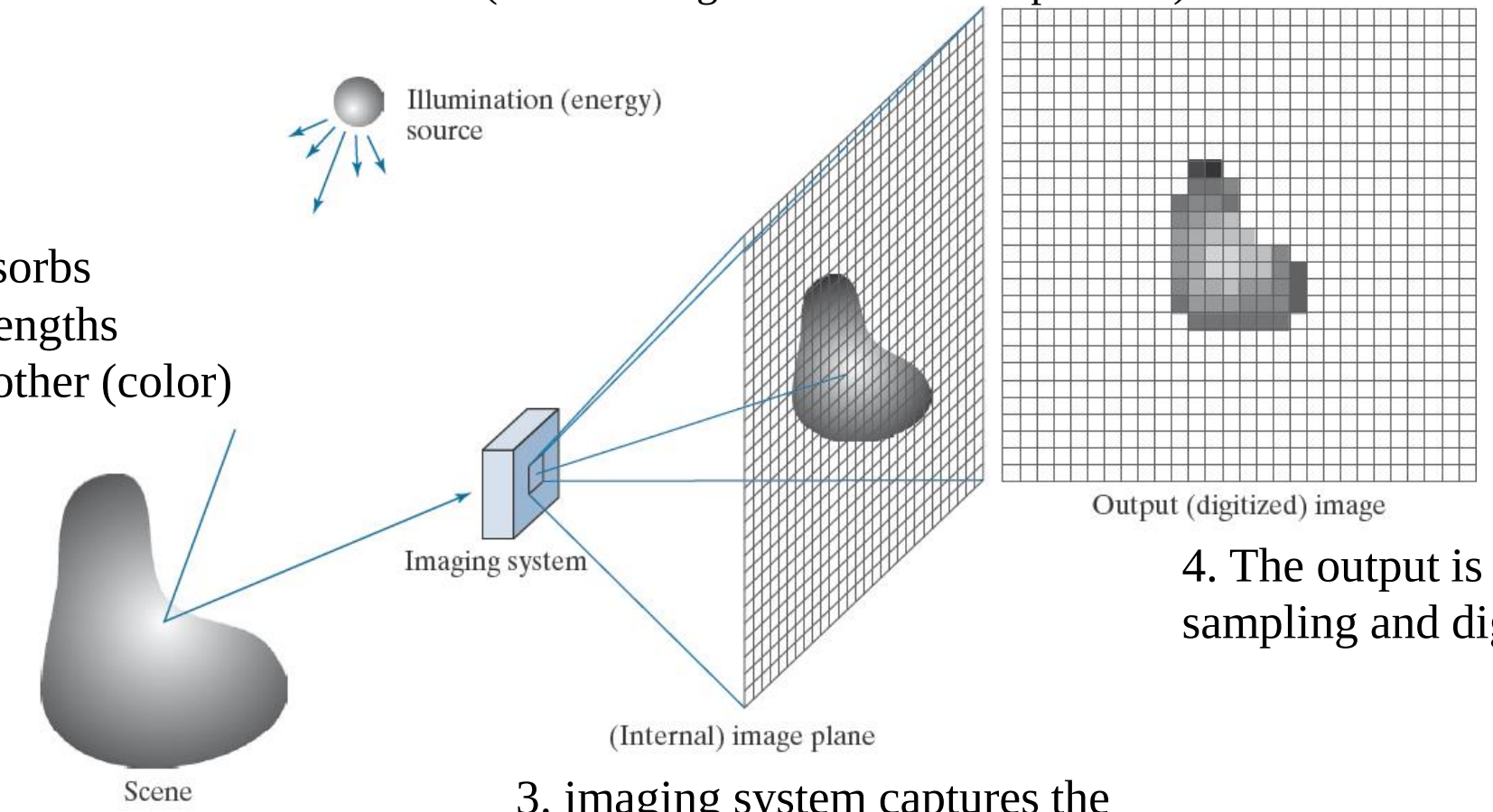


Image acquisition process

1. Illumination source emits light
(all wavelengths in the visible spectrum)

2. Object absorbs
Some wavelengths
and reflects other (color)



4. The output is obtained through
sampling and digitalization

3. imaging system captures the
energy of the reflected wavelengths

a
b c d e

What is Digital Image Processing

- Digital Image
  - A two-dimensional function x and y are spatial coordinates $f(x, y)$
  - The amplitude of f is called intensity or gray level at the point (x, y)

- Digital Image Processing
  - Process digital images by means of computer, it covers low-, mid-, and high-level processes
  - low-level: inputs and outputs are images
  - mid-level: outputs are attributes extracted from input images
  - high-level: an ensemble of recognition of individual objects

- Pixel
  - the elements of a digital image

A simple image formation model

$$f(x, y) = i(x, y) \cdot r(x, y)$$

$f(x, y)$: intensity at the point (x, y)

$i(x, y)$: illumination at the point (x, y)

(the amount of source illumination incident on the scene)

$r(x, y)$: reflectance/transmissivity at the point (x, y)

(the amount of illumination reflected/transmitted by the object)

where $0 < i(x, y) < \infty$ and $0 < r(x, y) < 1$

A simple image formation model

In practice, for any point (x_0, y_0) of the image, we require

$$i_{min} r_{min} = L_{min} \leq \ell = f(x_0, y_0) \leq L_{max} = i_{max} r_{max}$$

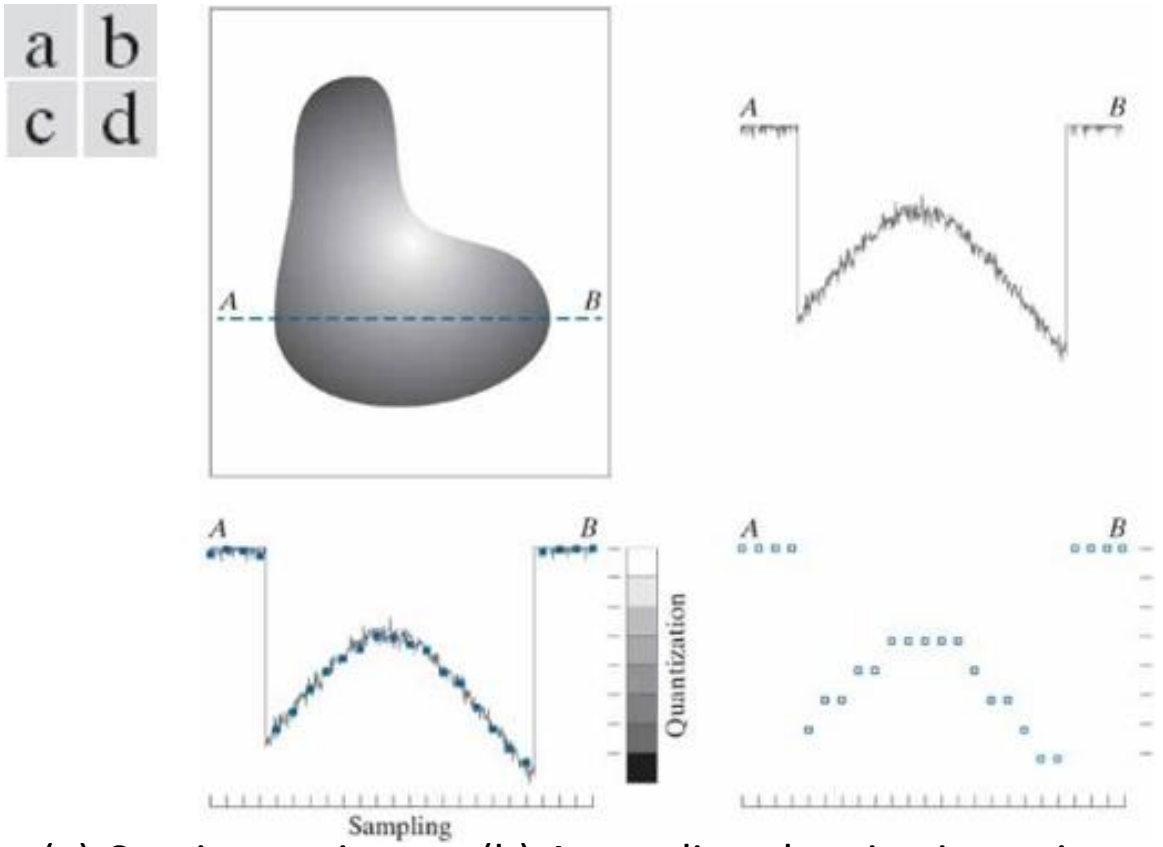
where L_{max} is required to be finite

$[L_{min}, L_{max}]$ is the *intensity* scale (also *gray scale*) of the image.

Common practice: $[L_{min}, L_{max}]$ is shifted to $[0, L-1]$, where $\ell = 0$ is black and $\ell = L - 1$ is white.

Basic concepts in sampling and quantization

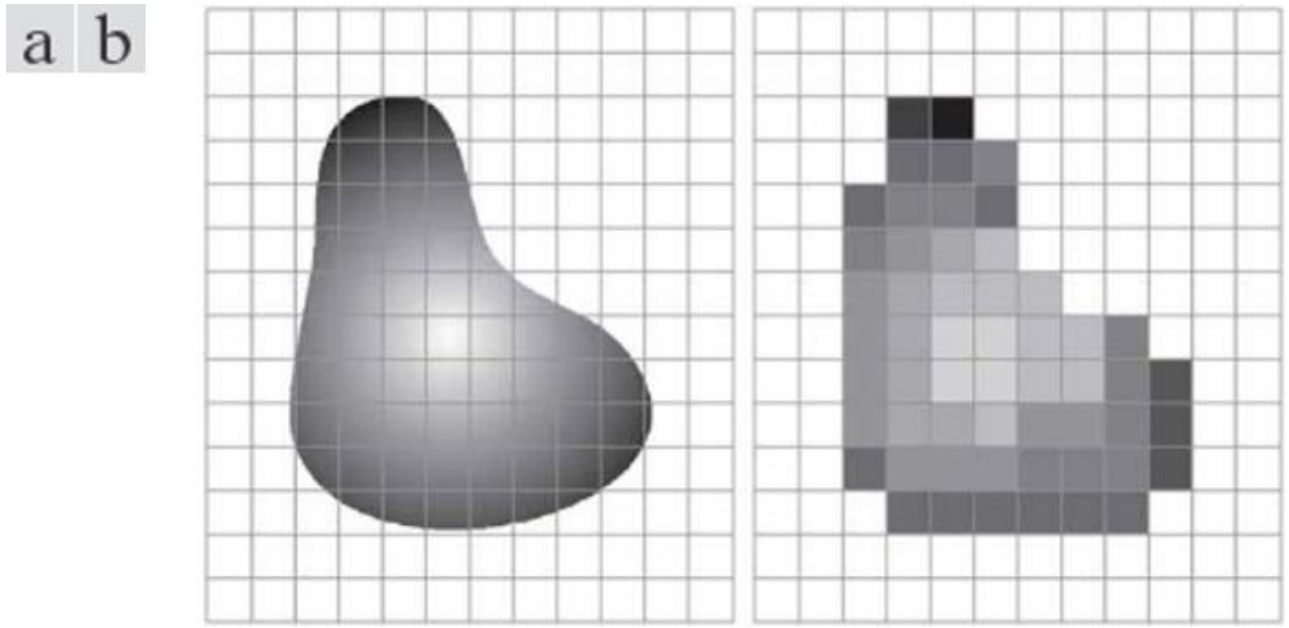
- Image sampling and quantization:
  - From continuous (+noise)
  - to discrete (digitalized)



(a) Continuous image. (b) A scan line showing intensity variations along line **AB** in the continuous image. (c) Sampling and quantization. (d) Digital scan line.

Basic concepts in sampling and quantization

- Because of sampling, the image will be described by a finite set of points (pixels)



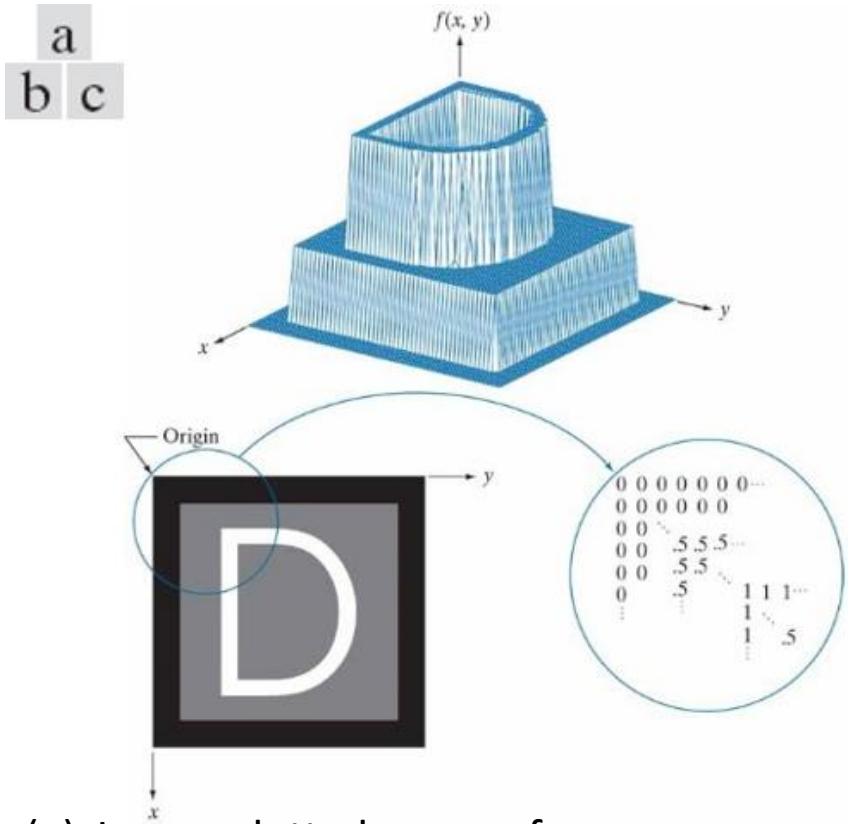
(a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Representing digital images

- The representation of an MxN numerical array as

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

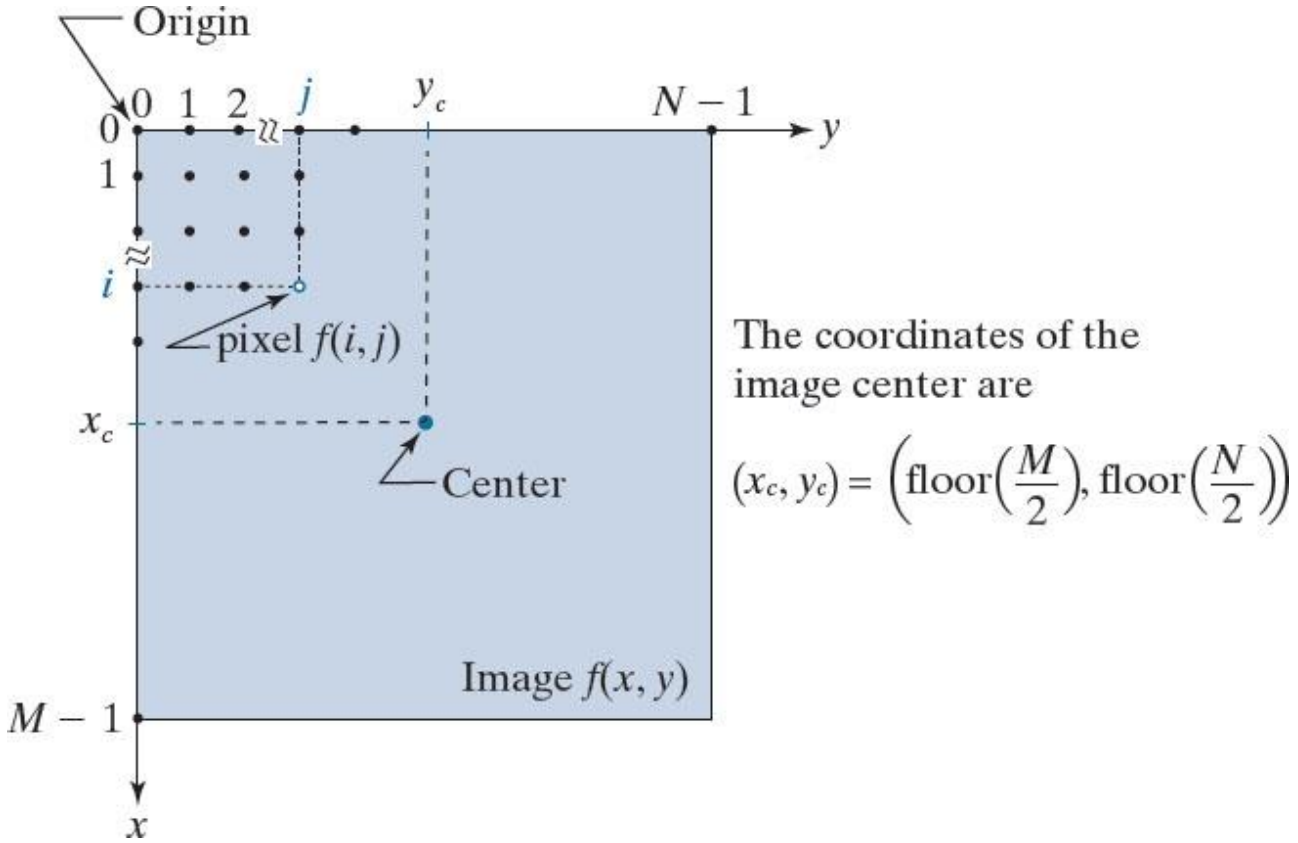
$$A=\begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \dots & \dots & \dots & \dots \\ a_{M-1,0} & a_{M-1,1} & \dots & a_{M-1,N-1} \end{bmatrix}$$



(a) Image plotted as a surface.
(b) Image displayed as a visual intensity array.
(c) Image shown as a 2-D numerical array.

Coordinate convention

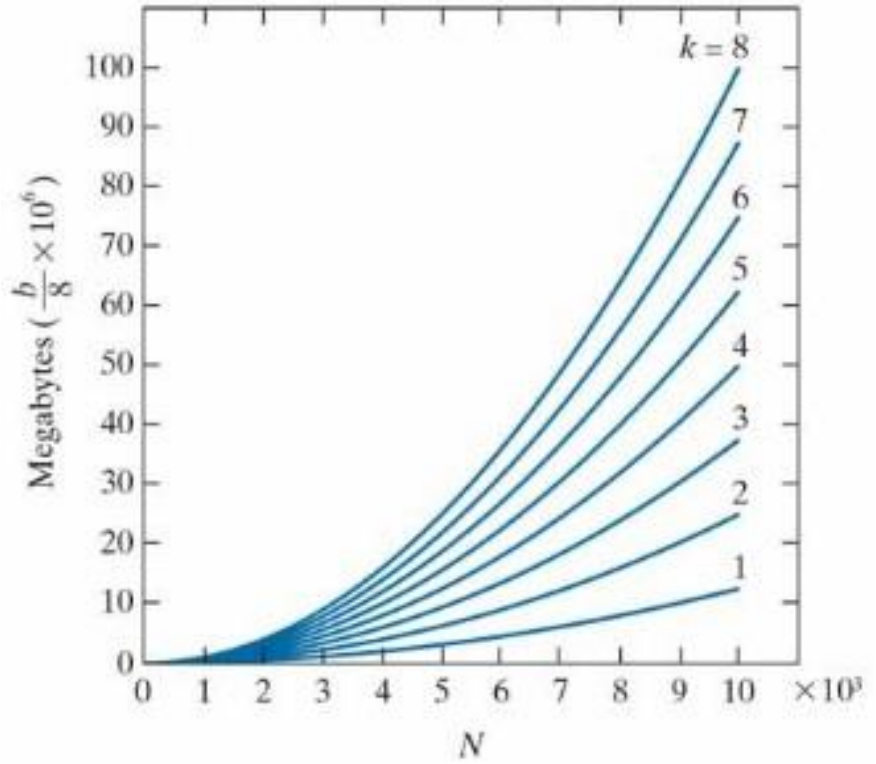
- Coordinate convention used to represent digital images.
  - Because coordinate values are integers, there is a one-to-one correspondence between **x** and **y** and the rows (**r**) and columns (**c**) of a matrix.



Storing digital images

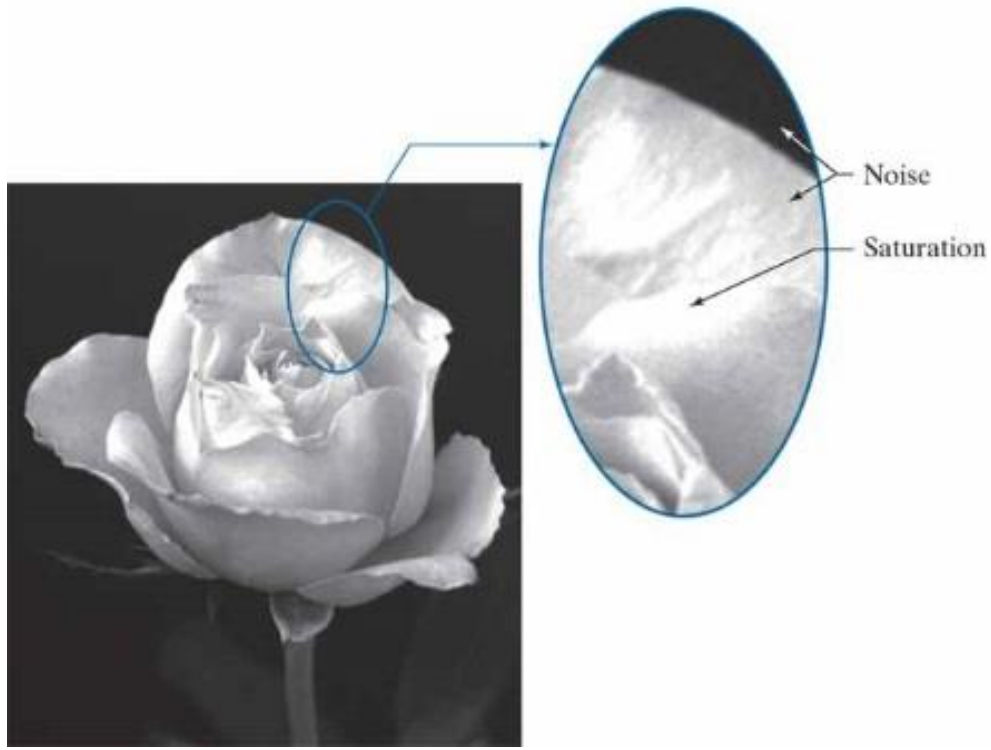
- Discrete intensity interval [0, L-1], $L=2^k$
  - The number b of bits required to store a M × N digitized image

$$b = M \times N \times k = N^2 k \text{ (when M=N)}$$



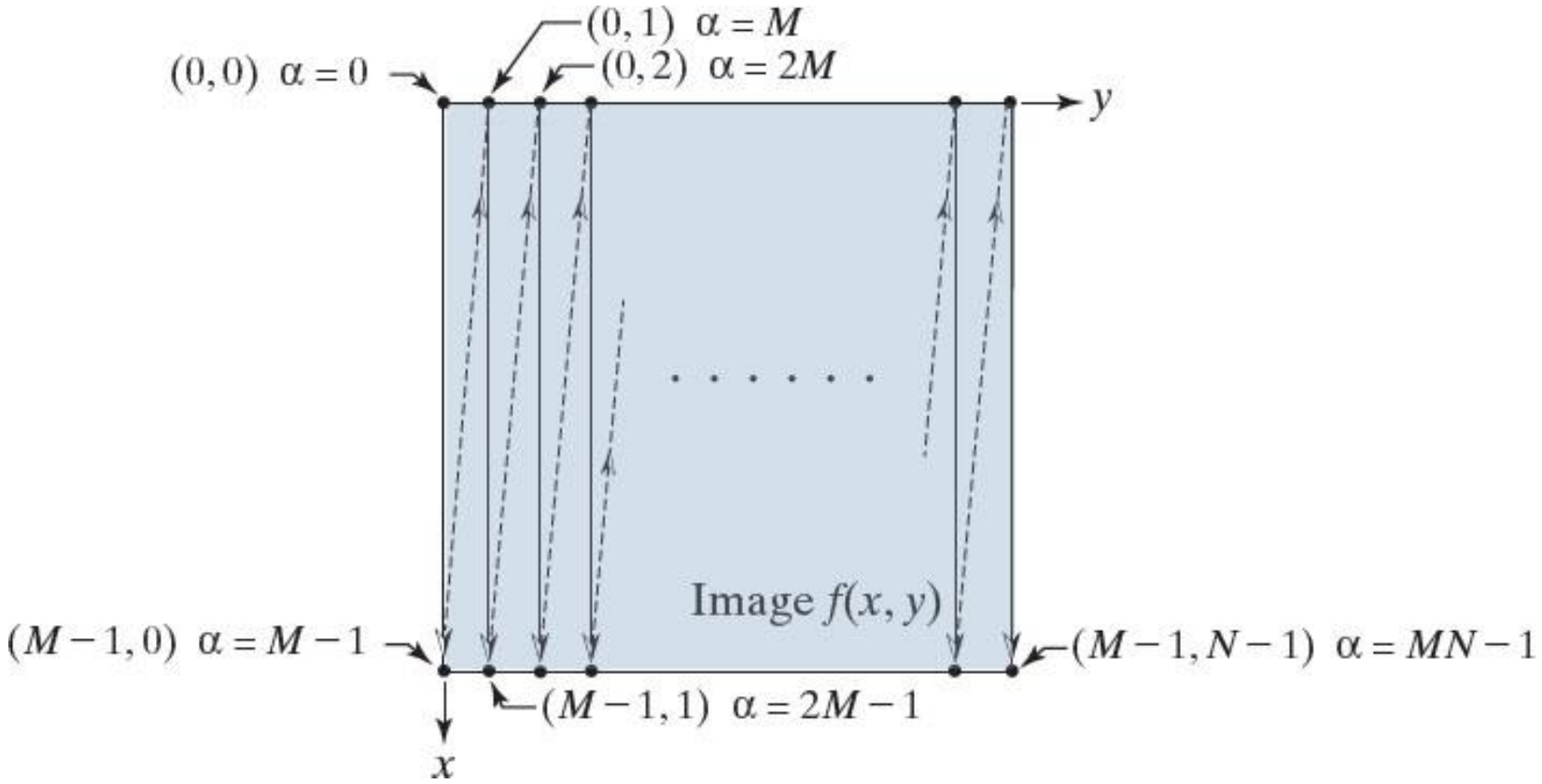
Some attributes of the imaging system/images

- *Dynamic range* of an imaging system: ratio between the maximum and minimum detectable intensity level of the system.
- *Saturation*: highest value beyond which intensity levels are clipped (to a constant value)
- *Noise:* grainy texture pattern
- Contrast: difference in intensity between the highest and lowest intensity level in an image



Linear vs. coordinate indexing

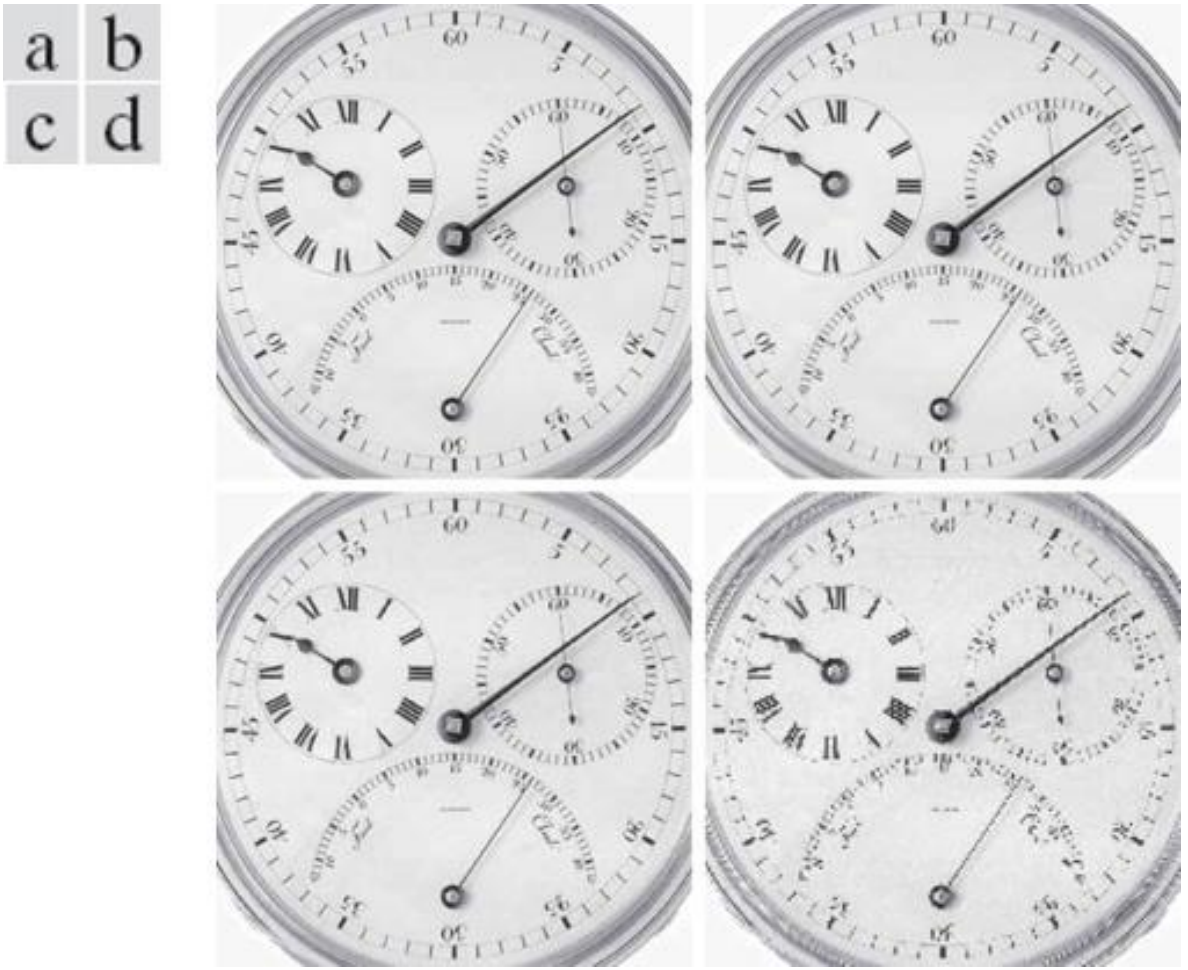
- $\alpha = My + x$
  - $x = \alpha \; mod \; M$
  - $y = (\alpha - x)/M$



Spatial and intensity resolution

- *DPI : dots per inch*
- *Spatial resolution:*

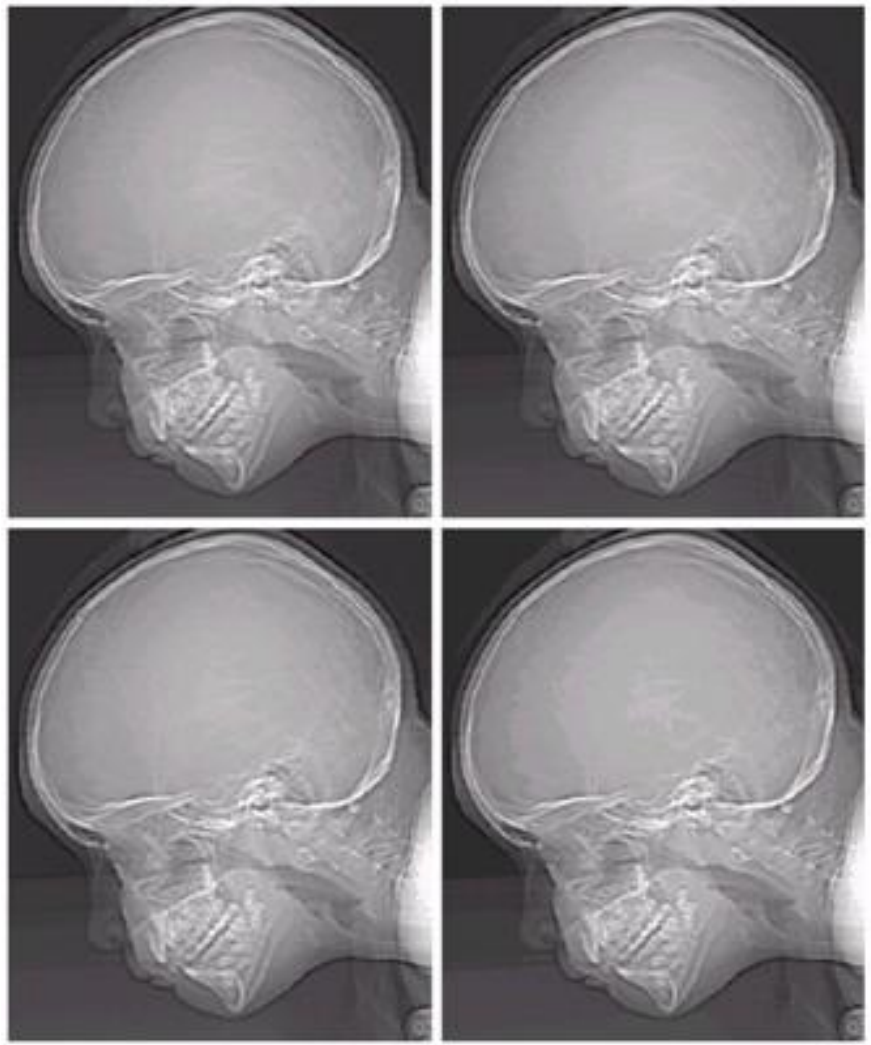
  is the measure of the smallest discernible detail in an image.
- Relates number of pixels to spatial dimension of the image.
- High spatial resolution: very detailed image
- Low spatial resolution: poor detailed image



Effects of reducing spatial resolution. The images shown are at: (a) 930 dpi, (b) 300 dpi, (c) 150 dpi, (d) 72 dpi.

Intensity resolution

- *Intensity resolution:* spatial resolution fixed, reduce *k,* the number of intensity levels.
  - [0, L-1]=[0, 2^k]
- Low intensity resolution might result in false contours



a b
c d

FIGURE 2.21
(a) 452 × 374, 256-level image. (b)–(d) Image displayed in 128, 64, and 32 gray levels, while keeping the spatial resolution constant.

Intensity resolution

- *Intensity resolution:* spatial resolution fixed, reduce *k,* the number of intensity levels.
  - [0, L-1]=[0, 2^k]
- Low intensity resolution might result in false contours

e f
g h

FIGURE 2.21 *(Continued)* (e)–(h) Image displayed in 16, 8, 4, and 2 gray levels. (Original courtesy of Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)

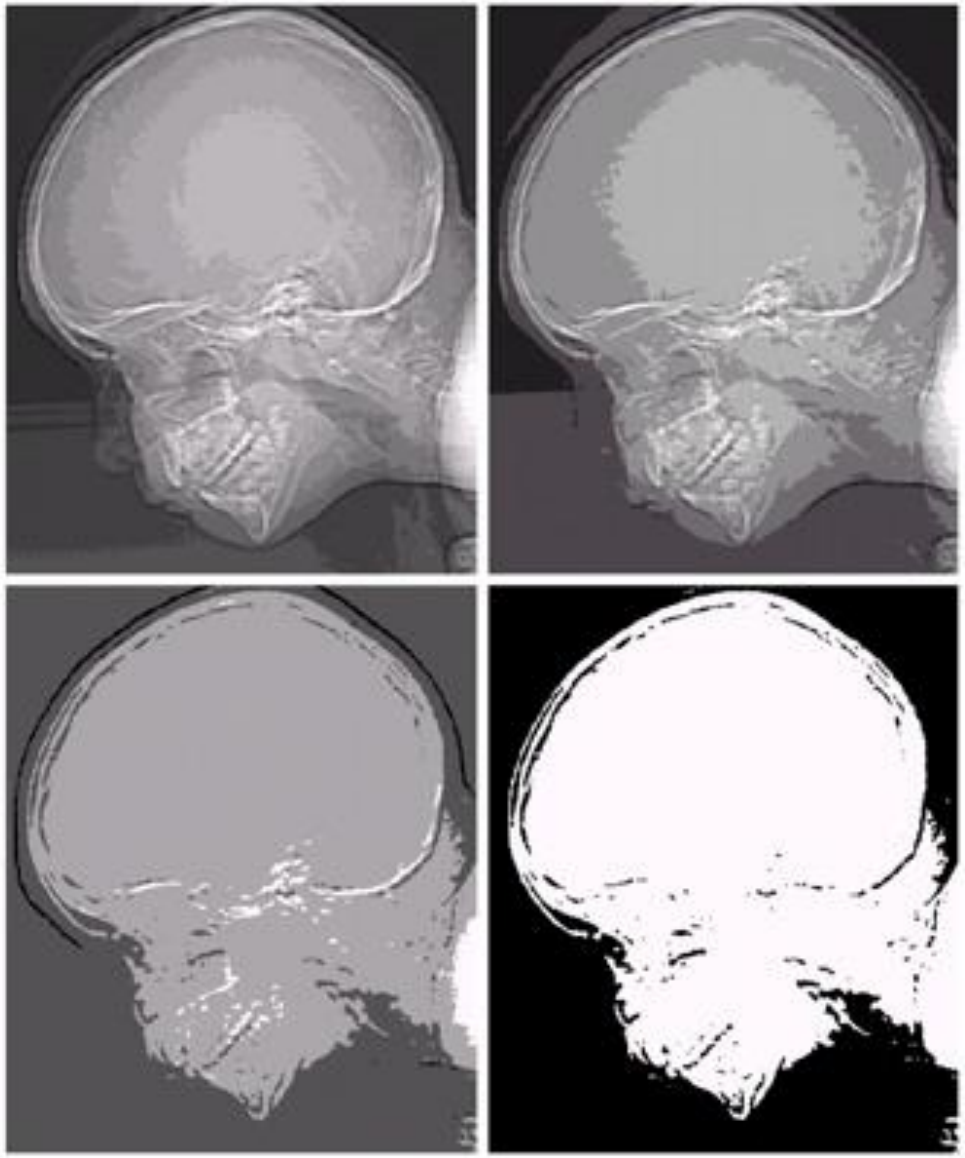


Image with various levels of detail

- What are the optimal values of *N* and *k* ?
  - No general rule, might depend on the level of detail of the image

a b c

(a) Image with a low level of detail. (b) Image with a medium level of detail. (c) Image with a relatively large amount of detail.

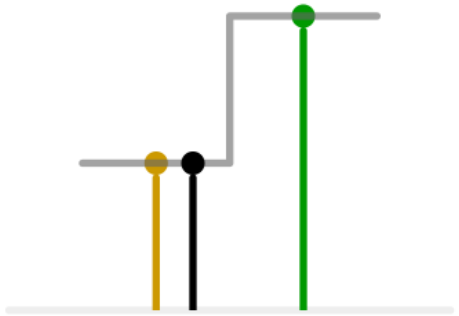
Image interpolation

- Is used for zooming, shrinking, rotating, geometric corrections, etc.
  (resampling methods)
- *Interpolation*: estimate values at unknown locations using known data values.

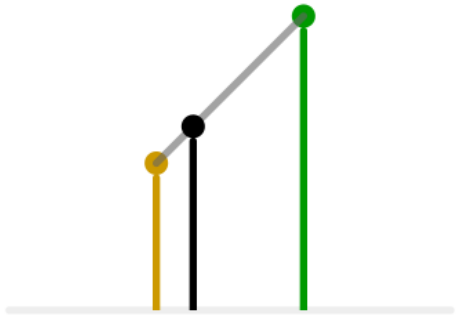
a b c

(a) Image reduced to 72 dpi and zoomed back to its original 930 dpi using nearest neighbor interpolation. (b) Image reduced to 72 dpi and zoomed using bilinear interpolation. (c) Same as (b) but using bicubic interpolation.

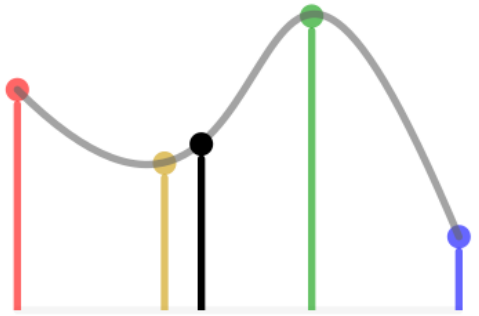
Image interpolation



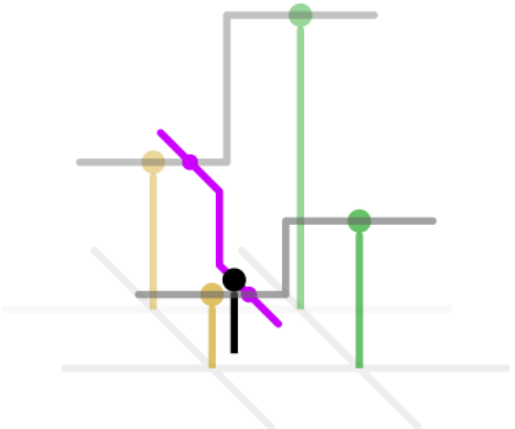
1D nearest-neighbour



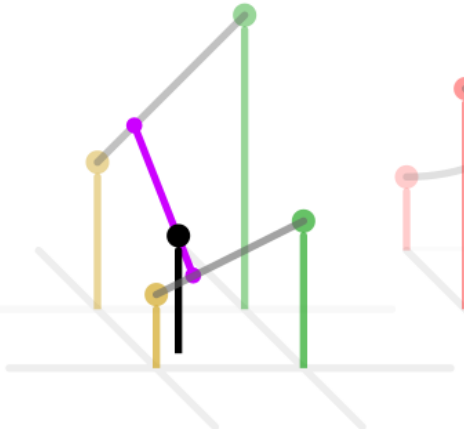
Linear



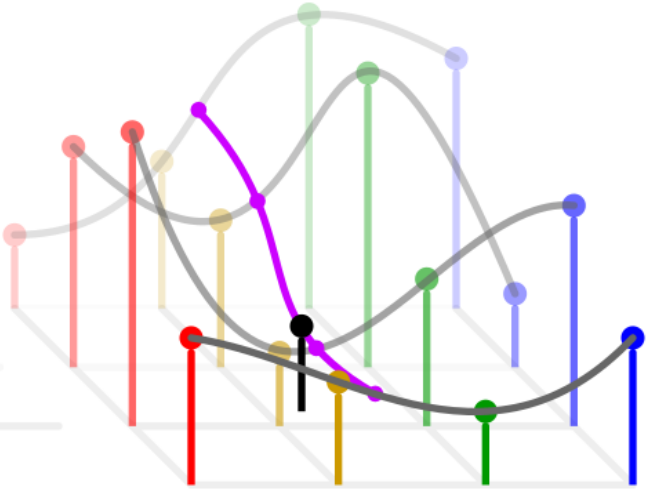
Cubic



2D nearest-neighbour



Bilinear



Bicubic

Image interpolation

- Nearest neighbor
  - find the closest pixel in the original grid and assign its value

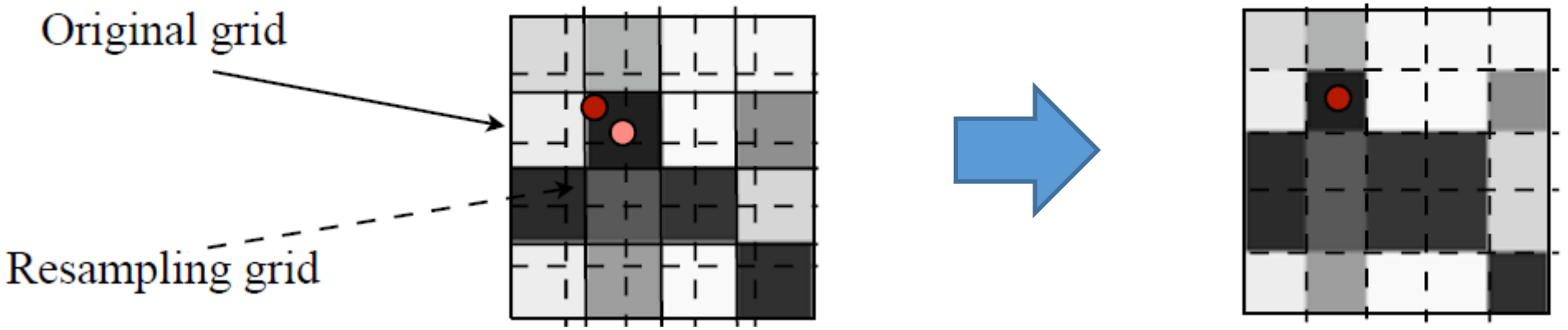
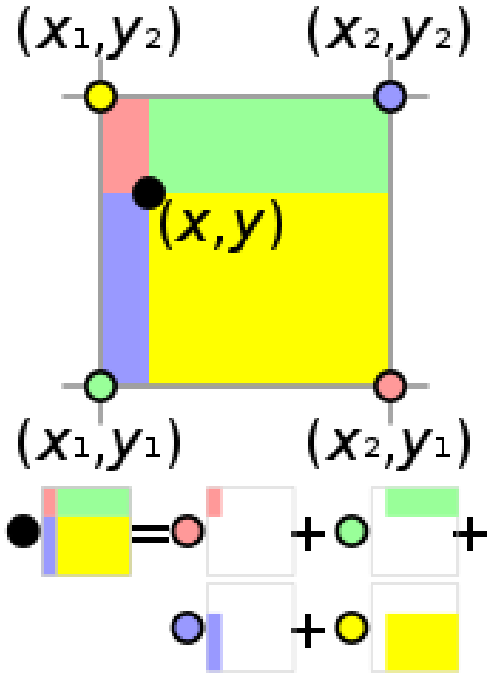
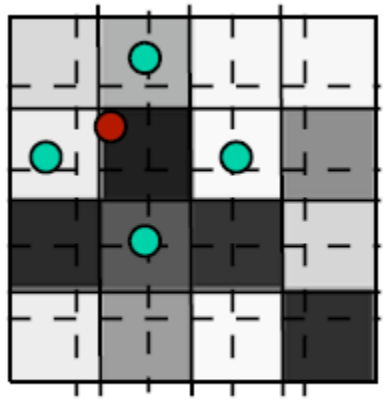
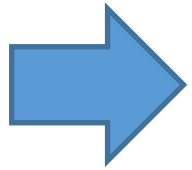


Image interpolation

- *Bilinear interpolation:* $v(x,y) = ax + by + cxy + d$
  - the coefficients *a, b, c, d* are computed using 4 neighbors







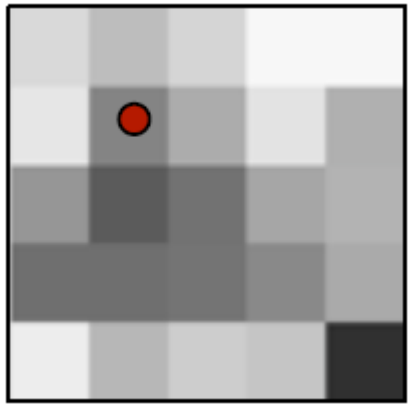
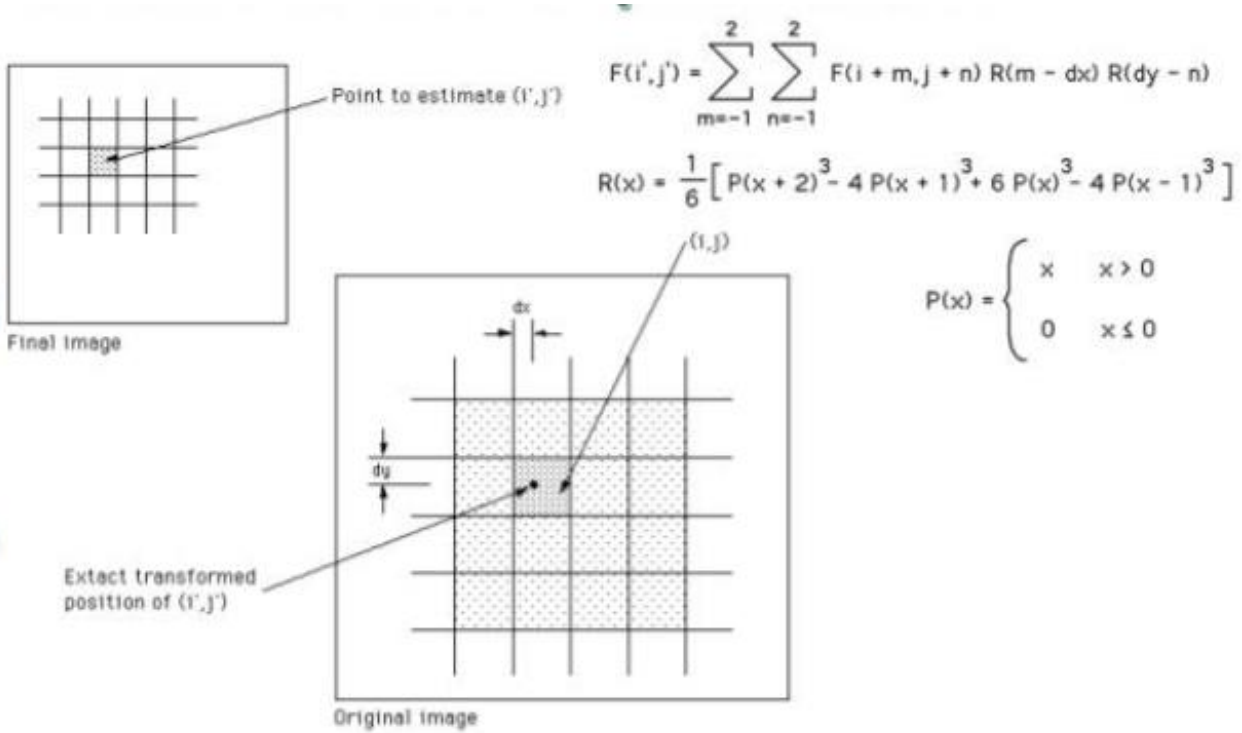
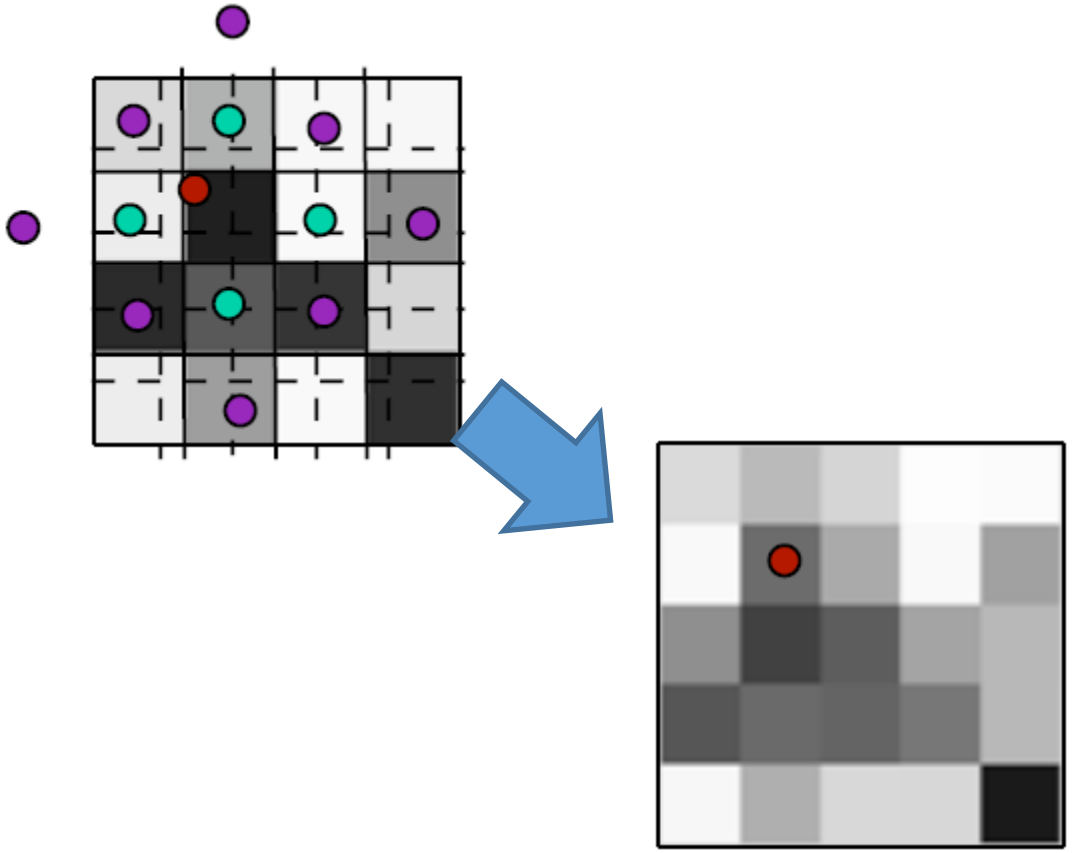


Image interpolation

- *Bicubic interpolation:* $v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{i,j}x^i y^j$
- the coefficients $a_{i,j}$ are computed using 16 nearest neighbors

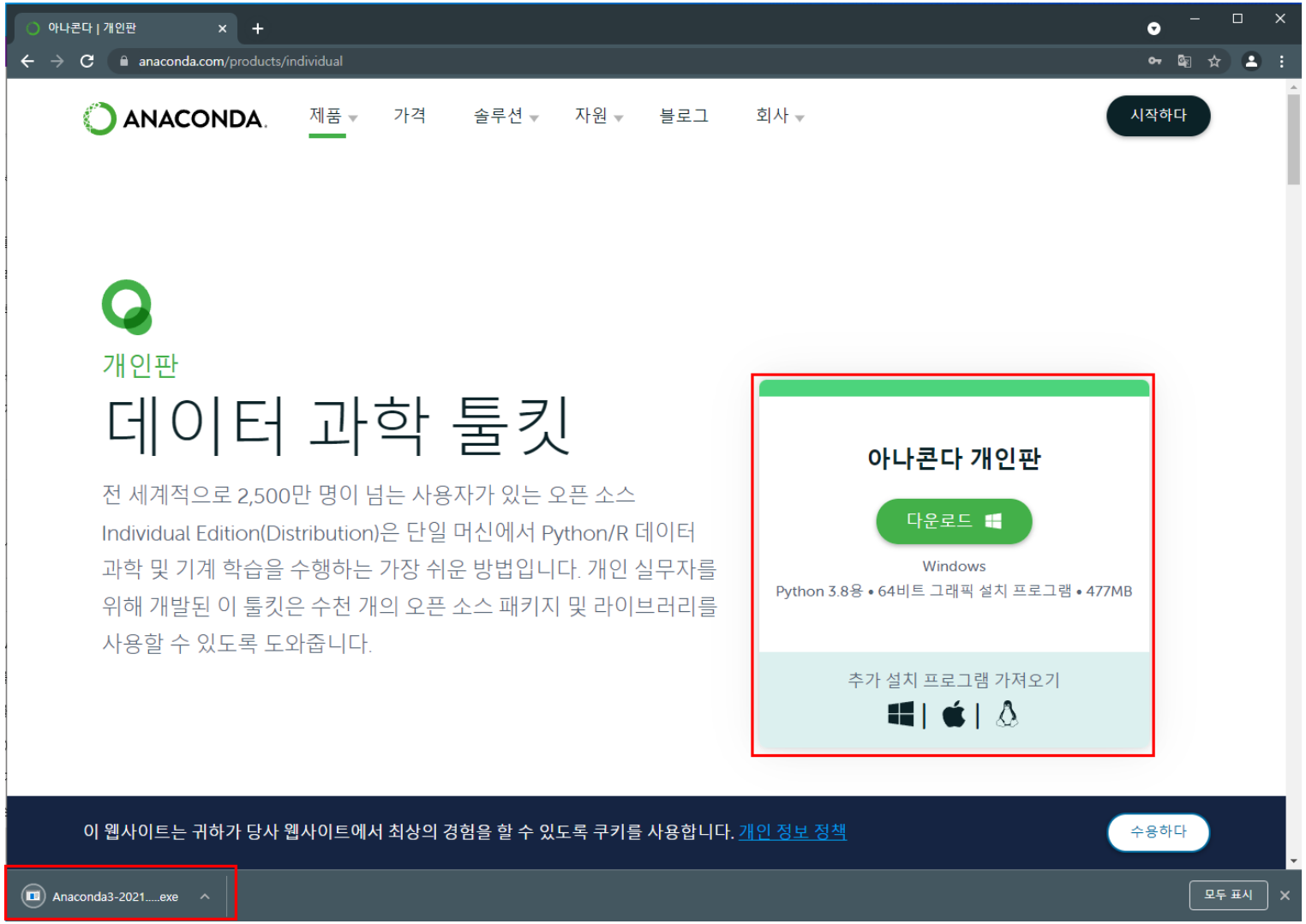




Python 설치 방식

- 특정 버전의 Python 을 설치
  - 가상환경을 만들기 위해서는 venv 명령을 통해 만듬
- Anaconda를 설치 후에 python 설치
  - Conda 가상환경을 만들고 각 가상환경에서 python 버전과 모듈의 버전을 다르게 설정
- Conda vs pip 차이점
  - 패키지를 받아오는 주소가 다르고 설치하는 모듈과 패키지가 다를 수 있음
  - Pip는 해당 패키지만 설치하지만, conda는 python 패키지외 의존성이 필요한 다른 모듈도 같이 설치해 줌

Anaconda 설치


아나콘다 | 개인판
anaconda.com/products/individual
ANACONDA.
제품
가격
솔루션
자원
블로그
회사
시작하다
개인판
데이터 과학 툴킷
전 세계적으로 2,500만 명이 넘는 사용자가 있는 오픈 소스 Individual Edition(Distribution)은 단일 머신에서 Python/R 데이터 과학 및 기계 학습을 수행하는 가장 쉬운 방법입니다. 개인 실무자를 위해 개발된 이 툴킷은 수천 개의 오픈 소스 패키지 및 라이브러리를 사용할 수 있도록 도와줍니다.
아나콘다 개인판
다운로드
Windows
Python 3.8용 • 64비트 그래픽 설치 프로그램 • 477MB
추가 설치 프로그램 가져오기
이 웹사이트는 귀하가 당사 웹사이트에서 최상의 경험을 할 수 있도록 쿠키를 사용합니다. 개인 정보 정책
수용하다
Anaconda3-2021.....exe
모두 표시

Pycharm or Spyder or VS code 설치

https://www.jetbrains.com/lp/pycharm-anaconda/?=

PyCharm: the Python IDE for Professional D···

JetBrains PyCharm is a Python IDE for data science and web development with int elligent code completion, on-the-fly error checking, quick-fixes, and much more...

www.jetbrains.com

https://www.spyder-ide.org/

Home — Spyder IDE

Download Ready to give Spyder a try? Let's get started! Want to join the community of scientists,···

www.spyder-ide.org

Conda 가상환경 설치

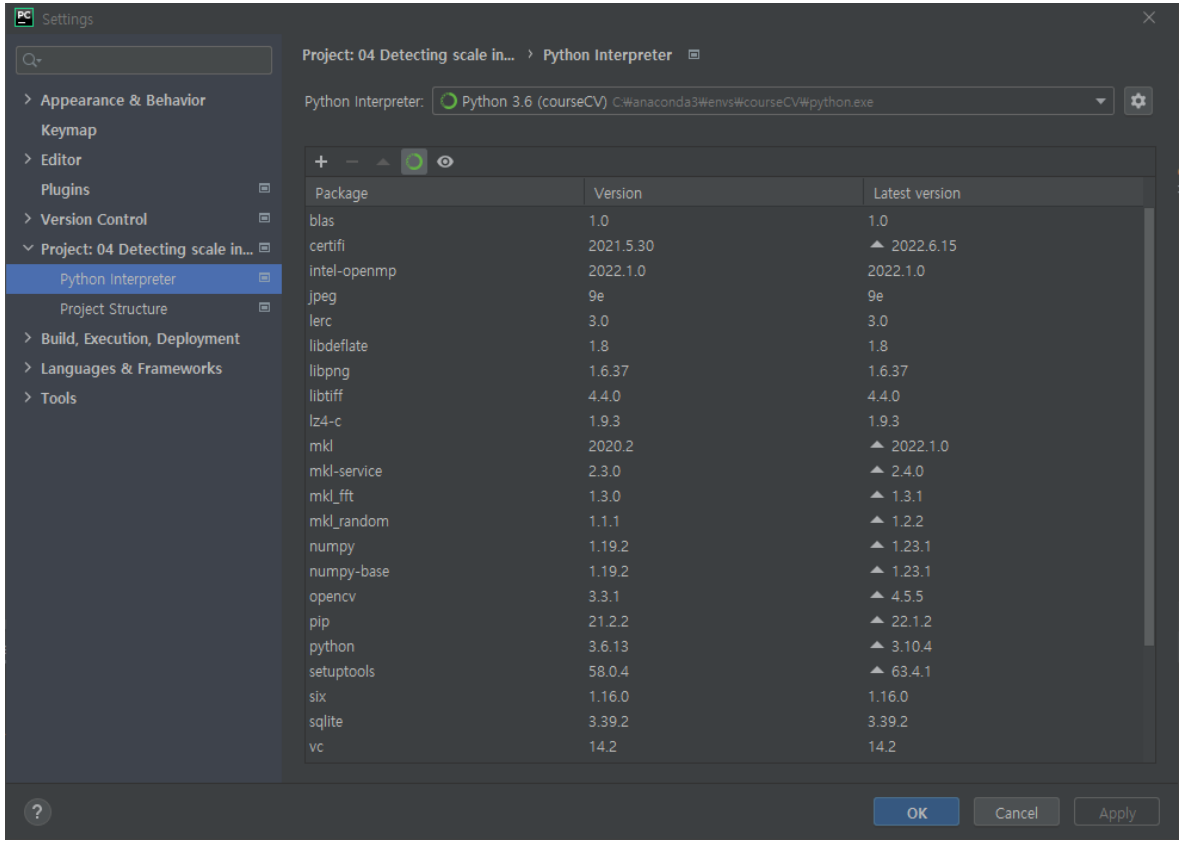
- conda create –n indCV(가상환경 이름) python-3.6
- conda activate indCV
- conda install opencv=3.3.1

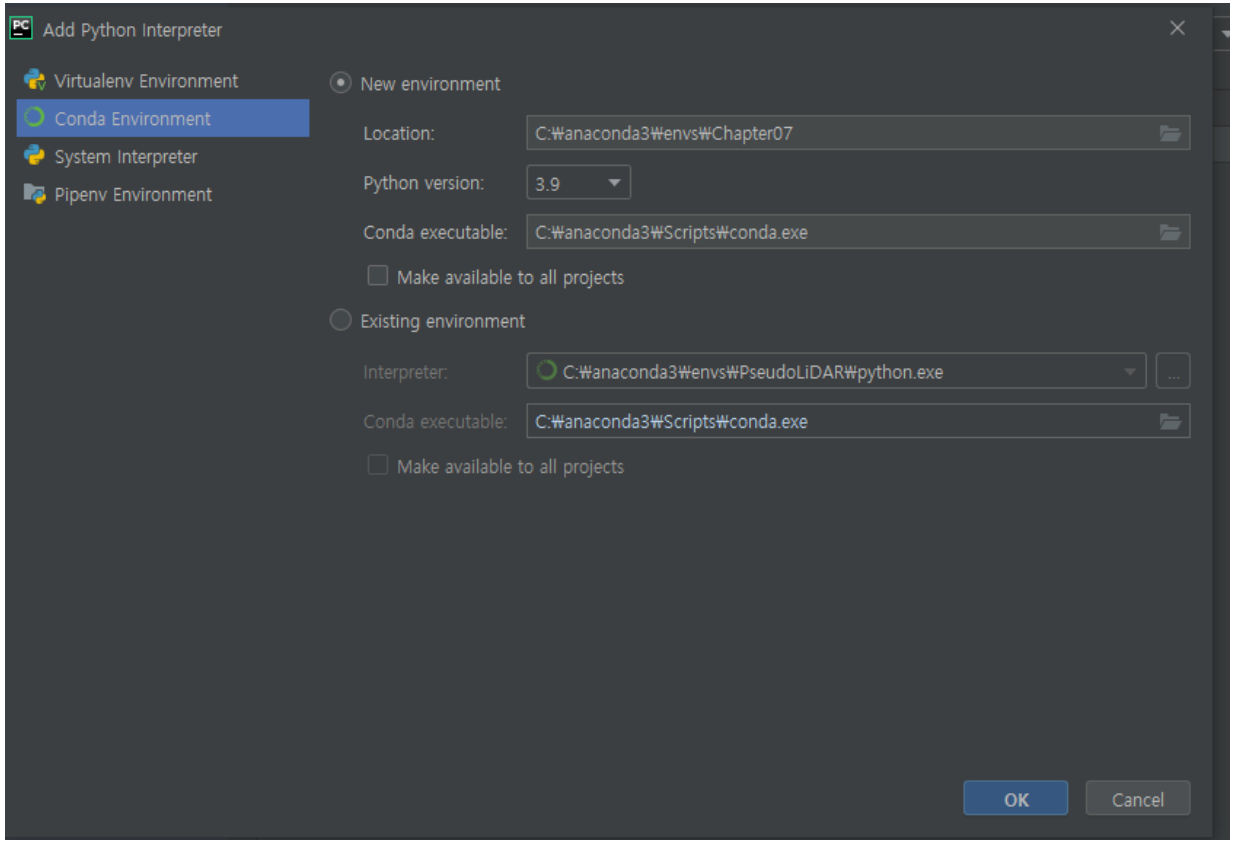
```
#
# Name                    Version                   Build  Channel
blas                      1.0                         mkl
certifi                   2021.5.30        py36haa95532_0
intel-openmp              2022.1.0         h59b6b97_3788
jpeg                      9e                  h2bbff1b_0
lerc                      3.0                 hd77b12b_0
libdeflate                1.8                 h2bbff1b_5
libpng                    1.6.37              h2a8f88b_0
libtiff                   4.4.0               h8a3f274_0
lz4-c                     1.9.3               h2bbff1b_1
mkl                       2020.2                      256
mkl-service               2.3.0            py36h196d8e1_0
mkl_fft                   1.3.0            py36h46781fe_0
mkl_random                1.1.1            py36h47e9c7a_0
numpy                     1.19.2           py36hadc3359_0
numpy-base                1.19.2           py36ha3acd2a_0
opencv                    3.3.1            py36h20b85fd_1
pip                       21.2.2           py36haa95532_0
python                    3.6.13              h3758d61_0
setuptools                58.0.4           py36haa95532_0
six                       1.16.0             pyhd3eb1b0_1
sqlite                    3.39.2              h2bbff1b_0
vc                        14.2                h21ff451_1
vs2015_runtime            14.27.29016         h5e58377_2
wheel                     0.37.1             pyhd3eb1b0_0
wincertstore              0.2              py36h7fe50ca_0
xz                        5.2.5               h8cc25b3_1
zlib                      1.2.12              h8cc25b3_3
zstd                      1.5.2               h19a0ad4_0

(courseCV) C:₩Users₩황영배>
```

```
C:₩Users₩Ybhwang>pip list
Package                 Version
----------------------- ---------
certifi                 2020.11.8
cycler                  0.10.0
kiwisolver              1.3.1
matplotlib              3.3.2
numpy                   1.19.3
opencv-contrib-python   3.3.1.11
opencv-python           3.3.1.11
Pillow                  8.0.1
pip                     18.1
pyparsing               2.4.7
python-dateutil         2.8.1
setuptools              40.6.2
six                     1.15.0
You are using pip version 18.1, however version 21.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:₩Users₩Ybhwang>python --version
Python 3.6.8
```

Pycharm 가상환경 설정

- File -> Settings -> Project: -> Python Interpreter





Reading image from file

```python
import argparse

import cv2

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
params = parser.parse_args()

img = cv2.imread(params.path)

# Check if image was successfully read.
assert img is not None

print('read {}'.format(params.path))
print('shape:', img.shape)
print('dtype:', img.dtype)

img = cv2.imread(params.path, cv2.IMREAD_GRAYSCALE)
assert img is not None
print('read {} as grayscale'.format(params.path))
print('shape:', img.shape)
print('dtype:', img.dtype)
```

OpenCV C++ Mat structure

```
class CV_EXPORTS Mat
{
public:
    // ... a lot of methods ...
    ...

    /*! includes several bit-fields:
         - the magic signature
         - continuity flag
         - depth
         - number of channels
     */
    int flags;
    //! the array dimensionality, >= 2
    int dims;
    //! the number of rows and columns or (-1, -1) when the array has more tha
    int rows, cols;
    //! pointer to the data
    uchar* data;

    //! pointer to the reference counter;
    // when array points to user-allocated data, the pointer is NULL
    int* refcount;

    // other members
    ...
};
```

```
for (int y = 0; y < height; y++) {

        for (int x = 0; x < width; x++) {

                uchar b = img_color.at<Vec3b>(y, x)[0];
                uchar g = img_color.at<Vec3b>(y, x)[1];
                uchar r = img_color.at<Vec3b>(y, x)[2];

                img_grayscale.at<uchar>(y, x) = (r + g + b) / 3.0;
        }
}
```

```
for (int y = 0; y < height; y++) {

        uchar *data_output = img_grayscale.data;


        for (int x = 0; x < width; x++) {

                uchar b = data_input[y * width * 3 + x * 3];
                uchar g = data_input[y * width * 3 + x * 3 + 1];
                uchar r = data_input[y * width * 3 + x * 3 + 2];


                data_output[width * y + x] = (r + g + b) / 3.0;
        }
}
```

OpenCV numpy.ndarray structure

```
>>> import numpy as np
>>> import cv2 as cv

>>> img = cv.imread('messi5.jpg')
```

You can access a pixel value by its row and column coordinates. For BGR image, it returns an array of Blue, Green, Red values. For grayscale image, just corresponding intensity is returned.

```
>>> px = img[100,100]
>>> print( px )
[157 166 200]

# accessing only blue pixel
>>> blue = img[100,100,0]
>>> print( blue )
157
```

You can modify the pixel values the same way.

```
>>> img[100,100] = [255,255,255]
>>> print( img[100,100] )
[255 255 255]
```

OpenCV numpy.ndarray structure

```
# accessing RED value
>>> img.item(10,10,2)
59

# modifying RED value
>>> img.itemset((10,10,2),100)
>>> img.item(10,10,2)
100
```

```
>>> print( img.shape )
(342, 548, 3)
```

Note

If an image is grayscale, the tuple returned contains or grayscale or color.

Total number of pixels is accessed by `img.size`:

```
>>> print( img.size )
562248
```

Image datatype is obtained by `img.dtype`:

```
>>> print( img.dtype )
uint8
```

```
>>> ball = img[280:340, 330:390]
>>> img[273:333, 100:160] = ball
```

Check the results below:

image

Resizing, Flipping

```python
import argparse
import cv2

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
params = parser.parse_args()
img = cv2.imread(params.path)
print('original image shape:', img.shape)

width, height = 128, 256
resized_img = cv2.resize(img, (width, height))
print('resized to 128x256 image shape:', resized_img.shape)

w_mult, h_mult = 0.25, 0.5
resized_img = cv2.resize(img, (0, 0), resized_img, w_mult, h_mult)
print('image shape:', resized_img.shape)

w_mult, h_mult = 2, 4
resized_img = cv2.resize(img, (0, 0), resized_img, w_mult, h_mult, cv2.INTER_NEAREST)
print('image shape:', resized_img.shape)

img_flip_along_x = cv2.flip(img, 0)
img_flip_along_x_along_y = cv2.flip(img_flip_along_x, 1)
img_flipped_xy = cv2.flip(img, -1)

# check that sequential flips around x and y equal to simultaneous x-y flip
assert img_flipped_xy.all() == img_flip_along_x_along_y.all()
```

cv2.resize(*img, dsize, fx, fy, interpolation*)

Parameters:

- **img** – Image
- **dsize** – Manual Size. 가로, 세로 형태의 tuple(ex; (100,200))
- **fx** – 가로 사이즈의 배수. 2배로 크게하려면 2. 반으로 줄이려면 0.5
- **fy** – 세로 사이즈의 배수
- **interpolation** – 보간법

- 0, for flipping the image around the x-axis (vertical flipping);
- > 0 for flipping around the y-axis (horizontal flipping);
- < 0 for flipping around both axes.

Saving image using lossy and lossless compression

```python
import argparse
import cv2

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
parser.add_argument('--out_png', default='../data/Lena_compressed.png',
                    help='Output image path for lossless result.')
parser.add_argument('--out_jpg', default='../data/Lena_compressed.jpg',
                    help='Output image path for lossy result.')
params = parser.parse_args()
img = cv2.imread(params.path)

# save image with lower compression - bigger file size but faster decoding
cv2.imwrite(params.out_png, img, [cv2.IMWRITE_PNG_COMPRESSION, 0])

# check that image saved and loaded again image is the same as original one
saved_img = cv2.imread(params.out_png)
assert saved_img.all() == img.all()

# save image with lower quality - smaller file size
cv2.imwrite(params.out_jpg, img, [cv2.IMWRITE_JPEG_QUALITY, 0])
```

Showing image in OpenCV window

```python
import argparse
import cv2

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
parser.add_argument('--iter', default=50, help='Downsampling-upsampling iterations number.')
params = parser.parse_args()
orig = cv2.imread(params.path)
orig_size = orig.shape[0:2]

cv2.imshow("Original image", orig)
cv2.waitKey(2000)

resized = orig

for i in range(params.iter):
    resized = cv2.resize(cv2.resize(resized, (256, 256)), orig_size)
    cv2.imshow("downsized&restored", resized)
    cv2.waitKey(100)

cv2.destroyWindow("downsized&restored")

cv2.namedWindow("original", cv2.WINDOW_NORMAL)
cv2.namedWindow("result")
cv2.imshow("original", orig)
cv2.imshow("result", resized)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Scrollbars in OpenCV window

```python
import cv2, numpy as np

cv2.namedWindow('window')

fill_val = np.array([255, 255, 255], np.uint8)


def trackbar_callback(idx, value):
    fill_val[idx] = value


cv2.createTrackbar('R', 'window', 255, 255, lambda v: trackbar_callback(2, v))
cv2.createTrackbar('G', 'window', 255, 255, lambda v: trackbar_callback(1, v))
cv2.createTrackbar('B', 'window', 255, 255, lambda v: trackbar_callback(0, v))

while True:
    image = np.full((500, 500, 3), fill_val)
    cv2.imshow('window', image)
    key = cv2.waitKey(3)
    if key == 27:
        break

cv2.destroyAllWindows()
```

Drawing 2D primitives

```python
import argparse
import cv2, random

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
params = parser.parse_args()
image = cv2.imread(params.path)
w, h = image.shape[1], image.shape[0]


def rand_pt(mult=1.):
    return (random.randrange(int(w * mult)),
            random.randrange(int(h * mult)))


cv2.circle(image, rand_pt(), 40, (255, 0, 0))
cv2.circle(image, rand_pt(), 5, (255, 0, 0), cv2.FILLED)
cv2.circle(image, rand_pt(), 40, (255, 85, 85), 2)
cv2.circle(image, rand_pt(), 40, (255, 170, 170), 2, cv2.LINE_AA)
cv2.line(image, rand_pt(), rand_pt(), (0, 255, 0))
cv2.line(image, rand_pt(), rand_pt(), (85, 255, 85), 3)
cv2.line(image, rand_pt(), rand_pt(), (170, 255, 170), 3, cv2.LINE_AA)
cv2.arrowedLine(image, rand_pt(), rand_pt(), (0, 0, 255), 3, cv2.LINE_AA)
cv2.rectangle(image, rand_pt(), rand_pt(), (255, 255, 0), 3)
cv2.ellipse(image, rand_pt(), rand_pt(0.3), random.randrange(360), 0, 360, (255, 255, 255), 3)
cv2.putText(image, 'OpenCV', rand_pt(), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 3)

cv2.imshow("result", image)
key = cv2.waitKey(0)
```

cv2.circle(*img, center, radian, color, thickness*)

Parameters:

- **img** – 그림을 그릴 이미지
- **center** – 원의 중심 좌표(x, y)
- **radian** – 반지름
- **color** – BGR형태의 Color
- **thickness** – 선의 두께, -1 이면 원 안쪽을 채움

cv2.line(*img, start, end, color, thickness*)

Parameters:

- **img** – 그림을 그릴 이미지 파일
- **start** – 시작 좌표(ex; (0,0))
- **end** – 종료 좌표(ex; (500. 500))
- **color** – BGR형태의 Color(ex; (255, 0, 0) -> Blue)
- **thickness** (*int*) – 선의 두께. pixel

cv2.putText(*img, text, org, font, fontSacle, color*)

Parameters:

- **img** – image
- **text** – 표시할 문자열
- **org** – 문자열이 표시될 위치. 문자열의 bottom-left corner점
- **font** – font type. CV2.FONT_XXX
- **fontSacle** – Font Size
- **color** – fond color

Handling user input from keyboard

```python
import argparse
import cv2, numpy as np, random

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
params = parser.parse_args()
image = cv2.imread(params.path)
image_to_show = np.copy(image)
w, h = image.shape[1], image.shape[0]


def rand_pt():
    return (random.randrange(w),
            random.randrange(h))

finish = False
while not finish:
    cv2.imshow("result", image_to_show)
    key = cv2.waitKey(0)
    if key == ord('p'):
        for pt in [rand_pt() for _ in range(10)]:
            cv2.circle(image_to_show, pt, 3, (255, 0, 0), -1)
    elif key == ord('l'):
        cv2.line(image_to_show, rand_pt(), rand_pt(), (0, 255, 0), 3)
    elif key == ord('r'):
        cv2.rectangle(image_to_show, rand_pt(), rand_pt(), (0, 0, 255), 3)
    elif key == ord('e'):
        cv2.ellipse(image_to_show, rand_pt(), rand_pt(), random.randrange(360), 0, 360, (255, 255, 0), 3)
    elif key == ord('t'):
        cv2.putText(image_to_show, 'OpenCV', rand_pt(), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 3)
    elif key == ord('c'):
        image_to_show = np.copy(image)
    elif key == 27:
        finish = True
```

Handling user input from mouse

```python
import argparse
import cv2, numpy as np

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='../data/Lena.png', help='Image path.')
params = parser.parse_args()
image = cv2.imread(params.path)
image_to_show = np.copy(image)

mouse_pressed = False
s_x = s_y = e_x = e_y = -1


def mouse_callback(event, x, y, flags, param):
    global image_to_show, s_x, s_y, e_x, e_y, mouse_pressed

    if event == cv2.EVENT_LBUTTONDOWN:
        mouse_pressed = True
        s_x, s_y = x, y
        image_to_show = np.copy(image)

    elif event == cv2.EVENT_MOUSEMOVE:
        if mouse_pressed:
            image_to_show = np.copy(image)
            cv2.rectangle(image_to_show, (s_x, s_y),
                          (x, y), (0, 255, 0), 1)

    elif event == cv2.EVENT_LBUTTONUP:
        mouse_pressed = False
        e_x, e_y = x, y
```

```python
cv2.namedWindow('image')
cv2.setMouseCallback('image', mouse_callback)

while True:
    cv2.imshow('image', image_to_show)
    k = cv2.waitKey(1)

    if k == ord('c'):
        if s_y > e_y:
            s_y, e_y = e_y, s_y
        if s_x > e_x:
            s_x, e_x = e_x, s_x

        if e_y - s_y > 1 and e_x - s_x > 0:
            image = image[s_y:e_y, s_x:e_x]
            image_to_show = np.copy(image)
    elif k == 27:
        break

cv2.destroyAllWindows()
```

Playing frame stream from video

```python
import cv2

capture = cv2.VideoCapture('../data/drop.avi')

while True:
    has_frame, frame = capture.read()
    if not has_frame:
        print('Reached end of video')
        break

    cv2.imshow('frame', frame)
    key = cv2.waitKey(500)
    if key == 27:
        print('Pressed Esc')
        break

cv2.destroyAllWindows()
```

다음의 프로그램을 작성하시오.

- Lena 이미지를 화면에 출력하시오. (**Showing image in OpenCV window** 예제 참고)
- 마우스를 이용해서 사각형(rectangle) 혹은 직선(line), 화살표직선(arrowedline)을 그리는 코드를 작성하시오. (**Drawing 2D primitives** 예제와 **Handling user input from mouse** 예제 참고)
- 키보드로 'r'을 입력받으면 사각형, 'l'를 입력받으면 직선, 'a'를 입력받으면 화살표직선을 그리는 모드로 전환하시오. (**Handling user input from keyboard** 예제 참고)
- 키보드로 'w'를 입력받으면 현재의 이미지를 'lena_draw.png'로 저장하시오. (**Saving image using lossy and lossless compression** 예제 참고)
- 키보드로 'c' 를 입력받으면 기존의 그렸던 사각형, 직선, 화살표직선을 모두 지우고 Lena 이미지만을 출력하시오
- 키보드로 'esc'를 입력받으면 코드를 종료하시오