Intelligent Control Systems

Image Processing (1)
— Basic Concepts and Introduction of OpenCV —

Shingo Kagami

Graduate School of Information Sciences,

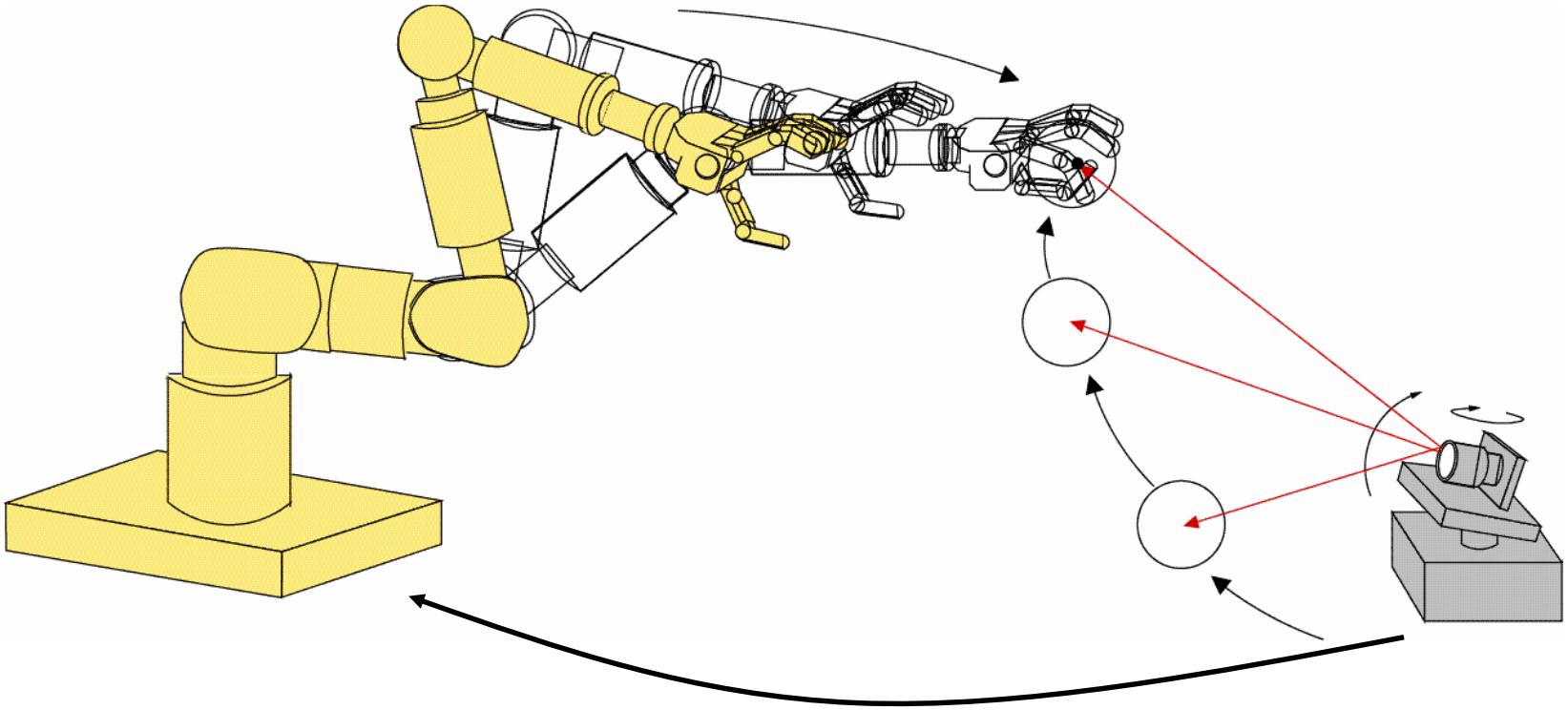
Tohoku University

swk(at)ic.is.tohoku.ac.jp

http://www.ic.is.tohoku.ac.jp/ja/swk/

Basic Motivation

e.g. Vision-based Control of Robots



- image acquisition (will be covered in TESP2018 lecture; Aug 7)
- **image processing**
- robot control (have been covered by Prof. Hashimoto's part)

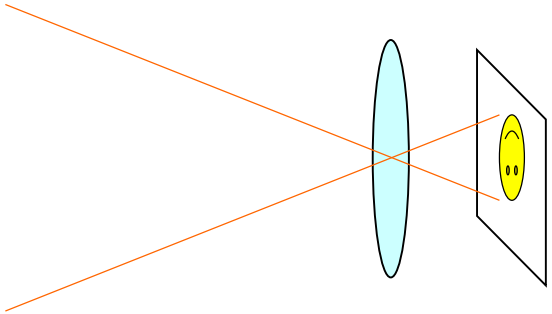
Schedule (tentative)

We focus on theories and implementations of basic visual tracking methods, which give foundations of image processing for visual servoing.

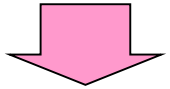
July 6: Intro: Image Processing Programming
July 13: Image Processing Basics (Filtering, Colors)
July 20: Object Tracking (1)
July 27: Object Tracking (2)

Aug 9: Final Report due

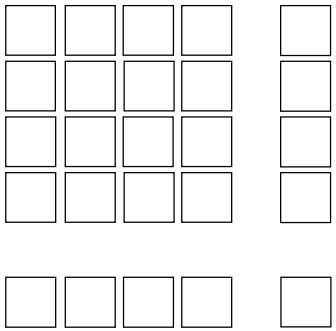
Digital Images



Analog distribution of light intensity



2-D discretization (into pixels)
quantization of intensity (ADC)

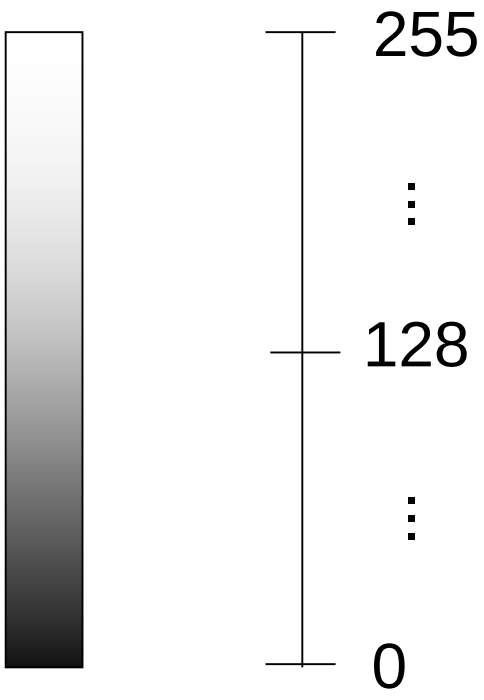


A digital image:

2-D array of pixel values

Pixel Value

(analog) light intensity; illuminance; voltage
(digital) pixel value; intensity value; gray level; grayscale value



quantized into [0, 255] integer:
8-bit grayscale image

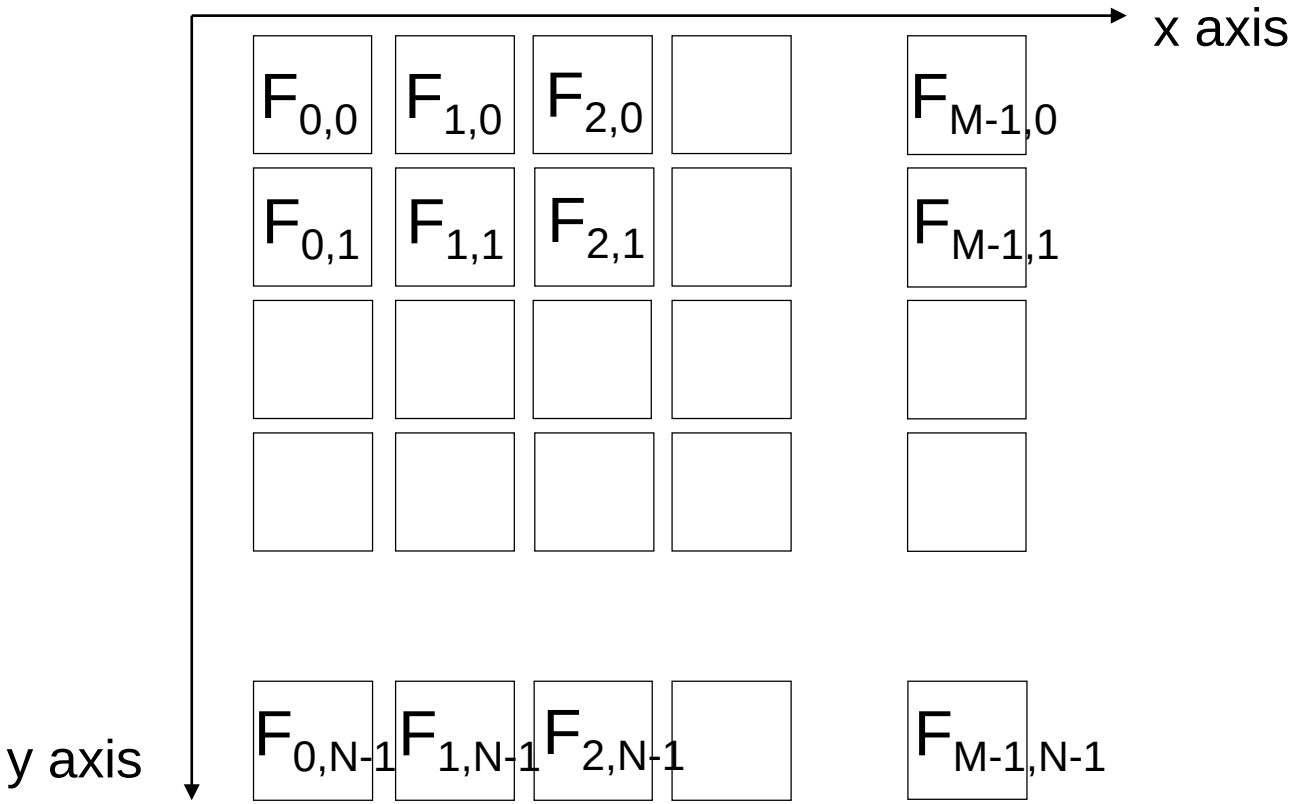
cf. binary image (= 1-bit grayscale)

Expression of a Digital Image

M £ N pixels digital image:

$\{ F_{x,y} \}$, $x = 0, 1, \cdots, M-1$, $y = 0, 1, \cdots, N-1$

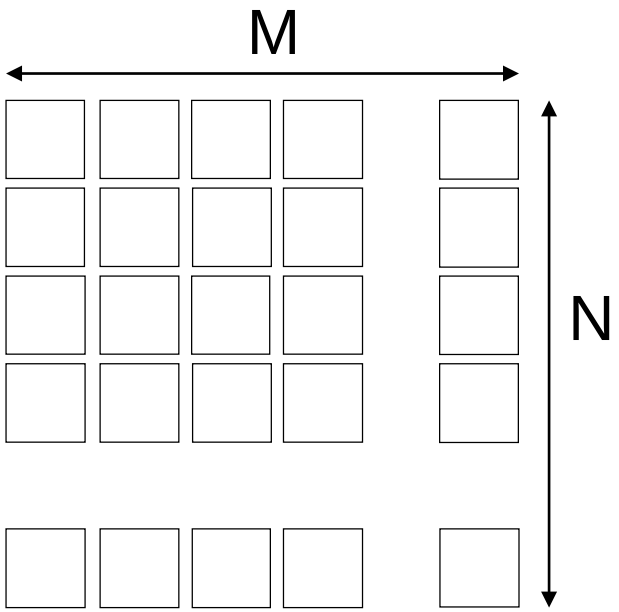
Pixel value at (x, y): $F_{x,y}$



Example in C

```
#define M 640
#define N 480
unsigned char image[M * N];

image[M * y + x] = 30;
// F(x, y) := 30
```

8-bit



- 2-D array is not convenient in C (e.g. not flexible in sizes)
- 1-D array is often preferred

A Simple Example Code in C

binarization (or thresholding)

```
#define M 640
#define N 480
#define THRESHOLD 128
unsigned char image[M * N];
int i, j;

for (j = 0; j < N; j++) {
    for (i = 0; i < M; i++) {
        if (image[M * j + i] >= THRESHOLD) {
            image[M * j + i] = 255;
        } else {
            image[M * j + i] = 0;
        }
    }
}
```

Image Processing Libraries

- Writing image processing programs by your own in this way is possible, but cumbersome
- How do you read image/video from a file or a camera?
- How do you display images?

There are many libraries or toolkits convenient for image processing, and using some of them is a good choice

In this lecture, we use OpenCV for Python

note: our goal is not to show you detailed features of OpenCV, but to lecture basic concepts of image processing by using OpenCV as a tool

OpenCV (Open Computer Vision Library)

http://opencv.org/

- A de facto standard library in Computer Vision community
  - implements many state-of-the-art algorithms
  - offers simple but easy-to-use I/O and GUI

- Runs on Windows, Mac, Linux, Android, iOS etc.
- Language: C++ (also supports: **python**, Java)

Python

https://www.python.org/

A lightweight scripting language

- easy to write, easy to run
- becoming popular particularly in scientific data processing
- a lot of useful modules are available

Setup (for Windows)

A portable package for this class is available (prepared in USB memories). It requires 6 GB disk space.

How this package is prepared:

Anaconda 5.3 with Python 3.6 for Windows (64 bit)
https://www.anaconda.com/

- Install for “just me”
- Destination: C:¥ic2018¥Anaconda3
- Uncheck all the Advanced Options

arbitrary folder of your choice

In Command Prompt with PATH variable set (see next page):

```
> pip install opencv-python
> pip install opencv-contrib-python
```

Running Codes

Run Command Prompt (cmd.exe):

```
> set ANACONDA_DIR=C:¥ic2018¥Anaconda3
> set PATH=%ANACONDA_DIR%;%ANACONDA_DIR%¥Scripts;%ANACOND
A_DIR%¥Library¥bin;%PATH%
```

Within this Command Prompt, the installed version of python is active.

```
> cd C:¥ic2018¥sample
> python thresh.py

> start spyder
```

(Spyder: Tools -> Preferences -> General -> Advanced Settings -> Language)

Interactive Execution

In IPython window of spyder, you can interactively execute Python codes.

```
cd C:/ic2018/sample/
import cv2
img = cv2.imread('lena.jpg')
cv2.imshow('testwin', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hit any key on the image window to proceed

```
import matplotlib.pyplot as plt
plt.imshow(img)  # Oops!
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

OpenCV uses (Blue, Green, Red) order to encode pixel colors, while matplotlib uses (Red, Green, Blue)

Full Python Code for Thresholding (1/2)

```
import cv2
import numpy as np

def threshold_impl(src, thresh, maxval):
    width = src.shape[1]
    height = src.shape[0]
    dest = np.zeros_like(src)

    for j in range(height):
        for i in range(width):
            if src[j, i] > thresh:
                dest[j, i] = maxval
            else:
                dest[j, i] = 0
    return dest
```

OpenCV module

numpy module

definition of a function

Note that indentation matters

zero-initialized image with the same size as `src`

`range(n)` generates a list `[0, 1, 2, …, n-1]`

Indices for pixel access are given in [Y-axis, X-axis] order (= [row, column] order)

Full Python Code for Thresholding (2/2)

```
if __name__ == '__main__':
    input = cv2.imread('lena.jpg', cv2.IMREAD_GRAYSCALE)
    retval, thresh_cv = cv2.threshold(input, 128, 255,
                                      cv2.THRESH_BINARY)
    thresh_np = np.full_like(input, 255) * (input > 128)
    thresh_my = threshold_impl(input, 128, 255)

    cv2.imshow("test1", thresh_cv)
    cv2.imshow("test2", thresh_np)
    cv2.imshow("test3", thresh_my)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

An idiom to define "main" routine

`imread()` method in `cv2` module is called

(window name, image)

Wait infinitely until any key is pressed, while refreshing the graphics

A function can return multiple values

Numpy module

http://www.numpy.org/

A de factor standard library for scientific computing with Python

- numpy array is used to represent images in OpenCV (Note: different from OpenCV in C++, which uses cv::Mat class)

list in Python is flexible but inefficient

```
x = [0, 1, "apple", [2, 4]]
```

numpy array is efficient because it is a straightforward array with fixed data type

```
import numpy as np
x = np.array([[1,2,3], [4,5,6], [7,8,9]])
```

Video Processing Example: Thresholding

```
if __name__ == '__main__':
    cap = cv2.VideoCapture('vtest.avi')
    #cap = cv2.VideoCapture(0)

    while True:
        retval, input = cap.read()
        if retval == False:
            break
        input = cv2.cvtColor(input, cv2.COLOR_BGR2GRAY)
        thresh_img = threshold_impl(input, 128, 255)

        cv2.imshow("test", thresh_img)
        key = cv2.waitKey(30)
        if key == ord('q'):
            break

    cv2.destroyAllWindows()
```

images are captured from video file or a camera device (specified by integer index)

break if no image is available

grayscale image is assumed

Just-In-Time Compilation by numba

https://numba.pydata.org/

Iterating through the pixels using for loops is extremely slow in Python.

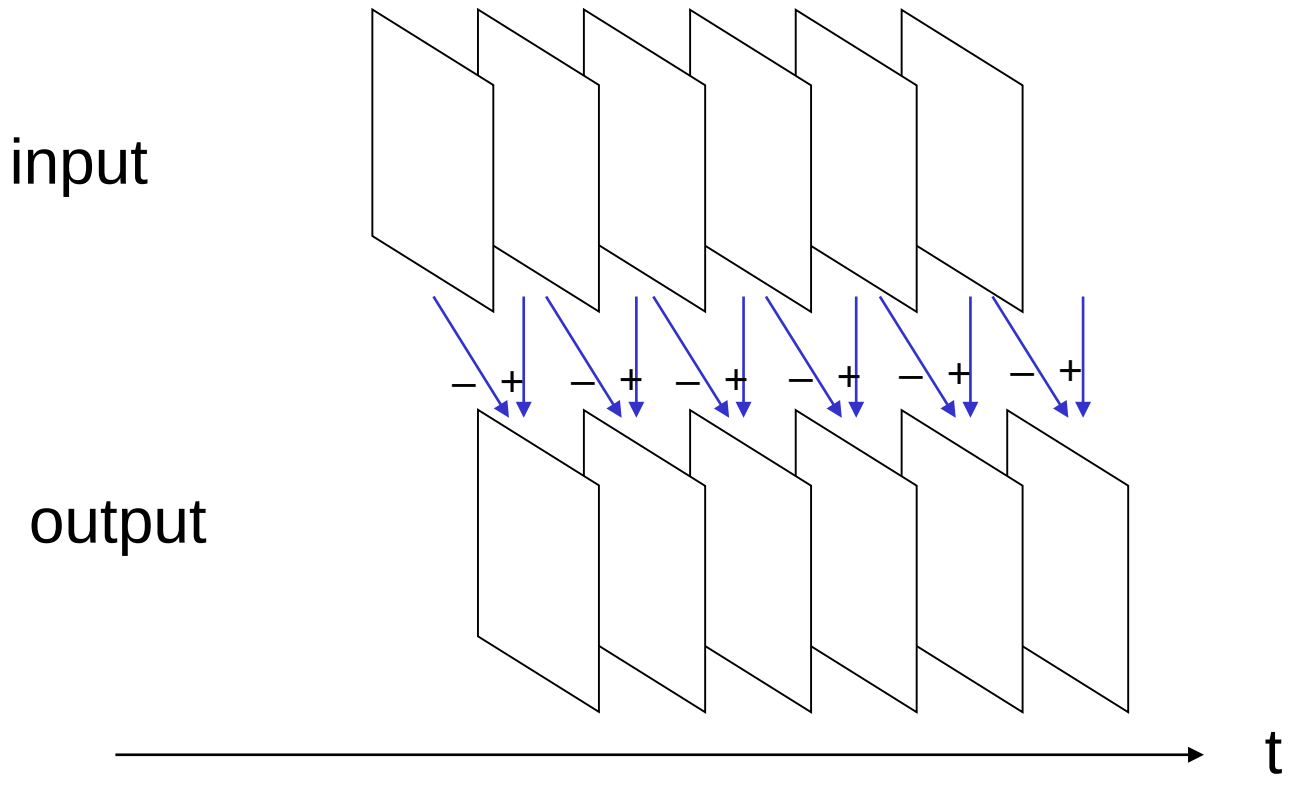
- Good practice is to use numpy methods thoughtfully, but it is not main focus of this course
- We use numba module as a workaround: the function is compiled when it is executed for the first time and therefore runs fast for the second time and on

```
from numba import jit

@jit
def threshold_impl(src, thresh, maxval):
    …
```

Trackbars

```
if __name__ == '__main__':
    cv2.namedWindow('test')
    cv2.createTrackbar('thresh', 'test', 128, 255, doNothing)
    cap = cv2.VideoCapture('vtest.avi')
    while True:
        retval, input = cap.read()
        if retval == False:
            break
        input = cv2.cvtColor(input, cv2.COLOR_BGR2GRAY)
        thresh_val = cv2.getTrackbarPos('thresh', 'test')
        thresh_img = threshold_impl(input, thresh_val, 255)
        cv2.imshow("test", thresh_img)
        key = cv2.waitKey(30)
        if key == ord('q'):
            break
    cv2.destroyAllWindows()
```

Video Processing Example: Frame Difference



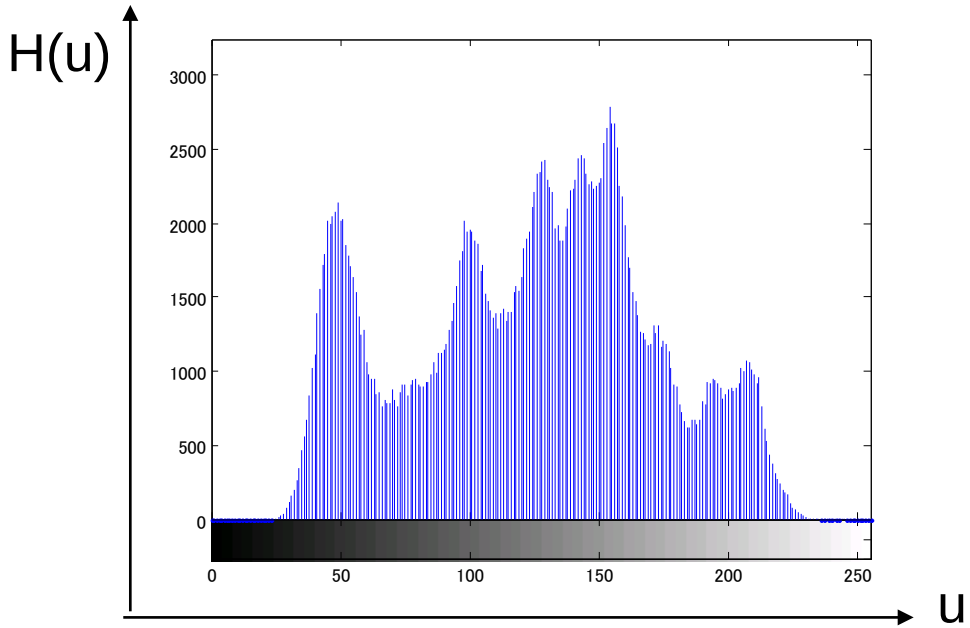
Frame Difference

```
if __name__ == '__main__':
    cap = cv2.VideoCapture('vtest.avi')
    retval, prev_frame = getGrayFrame(cap)
    while True:
        retval, input = getGrayFrame(cap)
        if retval == False:
            break
        diff_img = np.uint8((np.int16(input) -
                             np.int16(prev_frame)) / 2 + 128)
        prev_frame = input

        cv2.imshow("test", diff_img)
        key = cv2.waitKey(30)
        if key == ord('q'):
            break

    cv2.destroyAllWindows()
```

converted to 16-bit integer image to deal with negative values, and then converted back to 8-bit unsigned integer

wait for 30 ms until any key is hit;
Break if the pressed key code is that of ‘q’

Histogram of Pixel Values



$$\boldsymbol{H} = \{H_u\}_{u=1,2,\cdots,m}, \quad H_u = \sum_{\boldsymbol{x} \in S(u)} 1$$

where S(u) is a set of pixels having values belonging to the bin u

Plotting Histogram

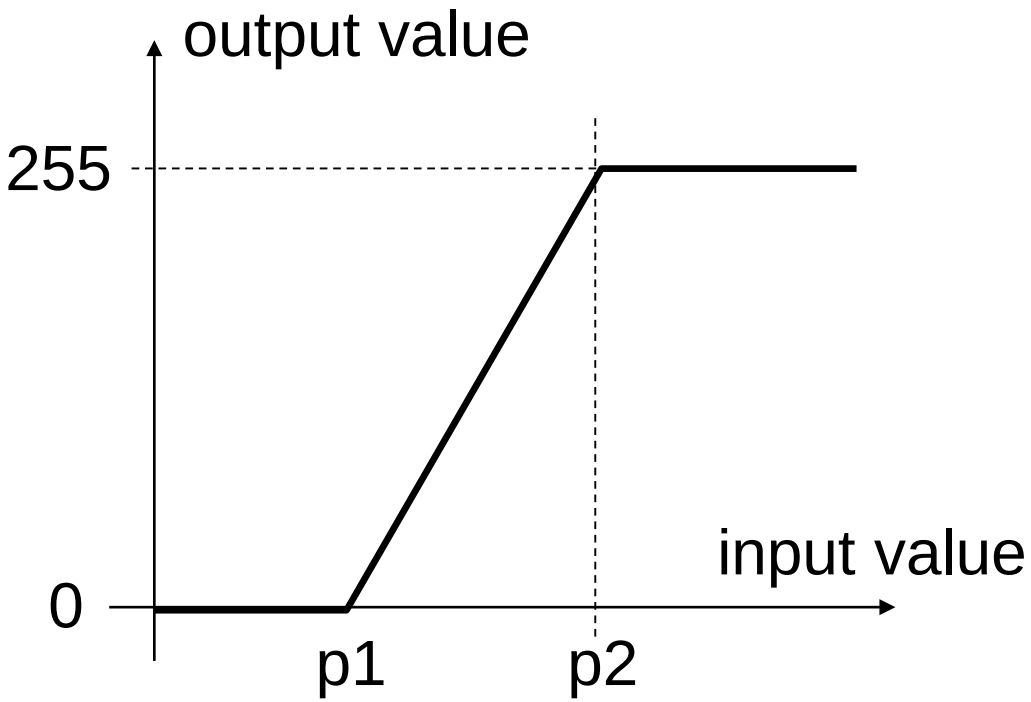
https://matplotlib.org/

matplotlib is a plotting library with MATLAB-like interface

```
import matplotlib.pyplot as plt

def drawHistogram(src):
    width = src.shape[1]
    height = src.shape[0]
    pix_val_hist = np.zeros(256)
    for j in range(height):
        for i in range(width):
            pix_val_hist[src[j, i]] += 1
    plt.bar(range(256), pix_val_hist)
```

abscissa values ordinate values

Converting Pixel Values



clip values between 0 and 255

```
converted_img = np.uint8(np.clip(
            (255 * (np.int32(input) - vmin)) / (vmax - vmin),
            0, 255)
        )
```

References

Slides and sample codes are available at
http://www.ic.is.tohoku.ac.jp/~swk/lecture/

Reference manuals for OpenCV and numpy are in:

- https://docs.opencv.org/3.4.1/
- http://www.numpy.org/

- A. Kaehler, G. Bradski: Learning OpenCV 3, O'Reilly, 2017. (詳解OpenCV 3, オライリー・ジャパン, 2018)
  Unfortunately, the codes are in C++