
Intelligent Control Systems

Image Processing (1)

— Basic Concepts and Introduction of OpenCV —

Shingo Kagami

Graduate School of Information Sciences,

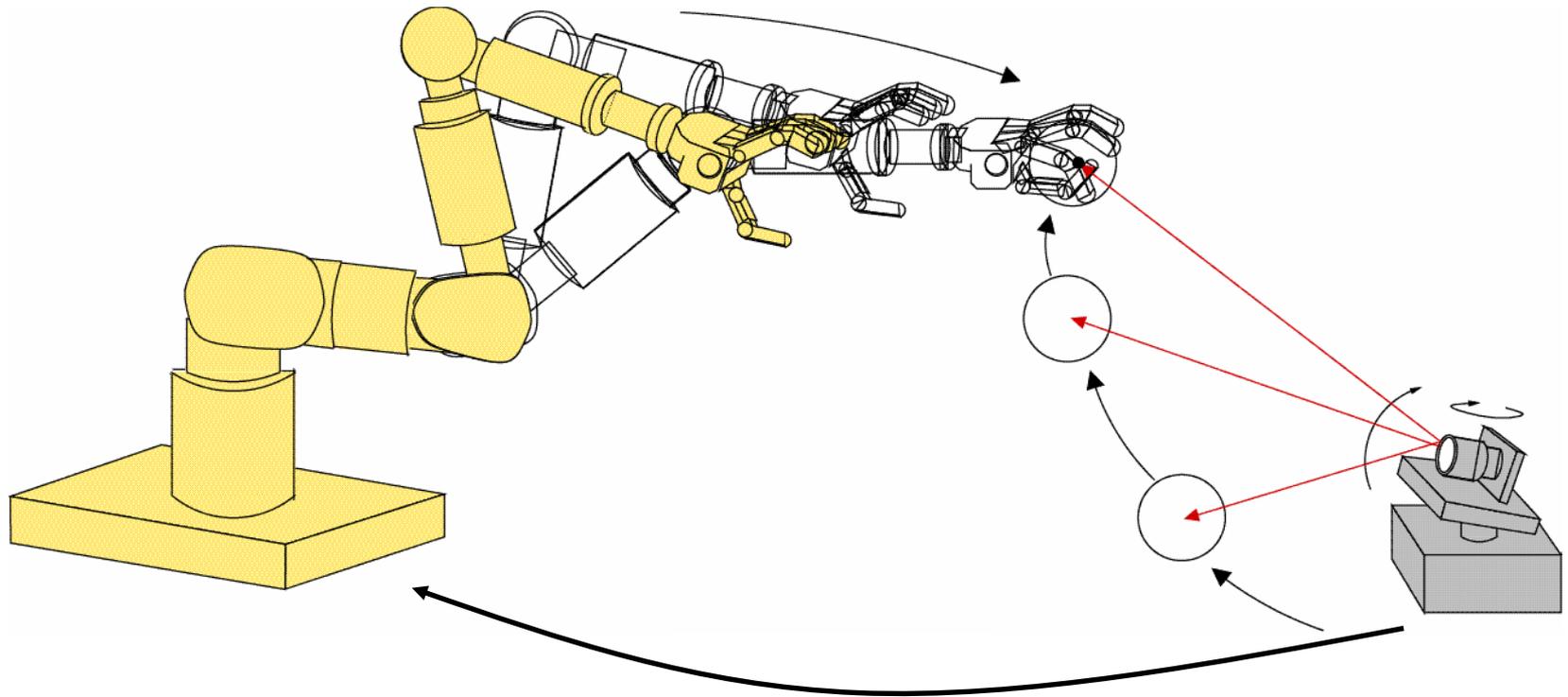
Tohoku University

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 (not covered this year)
- **image processing**
- robot control (have been covered by Prof. Hashimoto's part)

Schedule

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

July 10: Intro: Image Processing Programming

July 17: Image Processing Basics (Filtering, Colors)

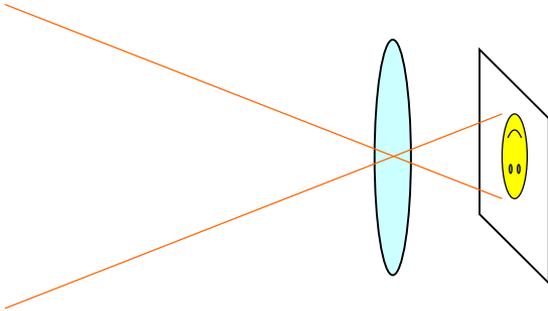
(July 24: Holiday)

July 31: Object Tracking (1)

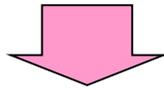
August 7: Object Tracking (2)

August 31: Final Report due

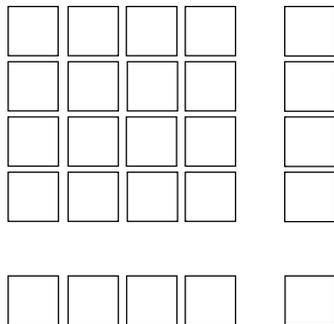
Digital Images



Analog distribution of light intensity



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

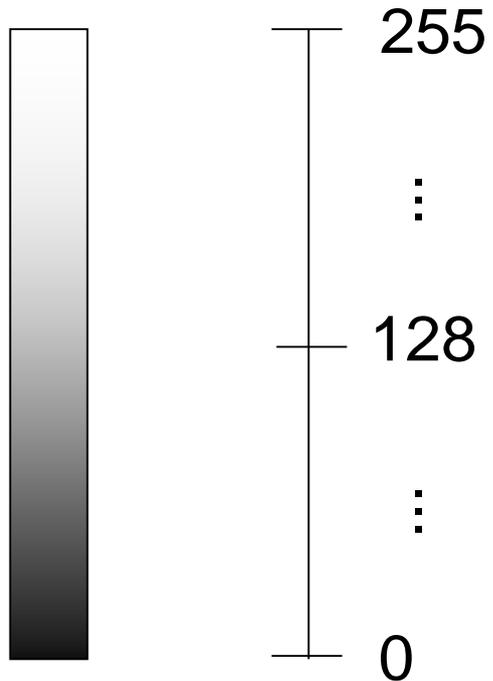


A digital image:
2-D array of pixel values

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

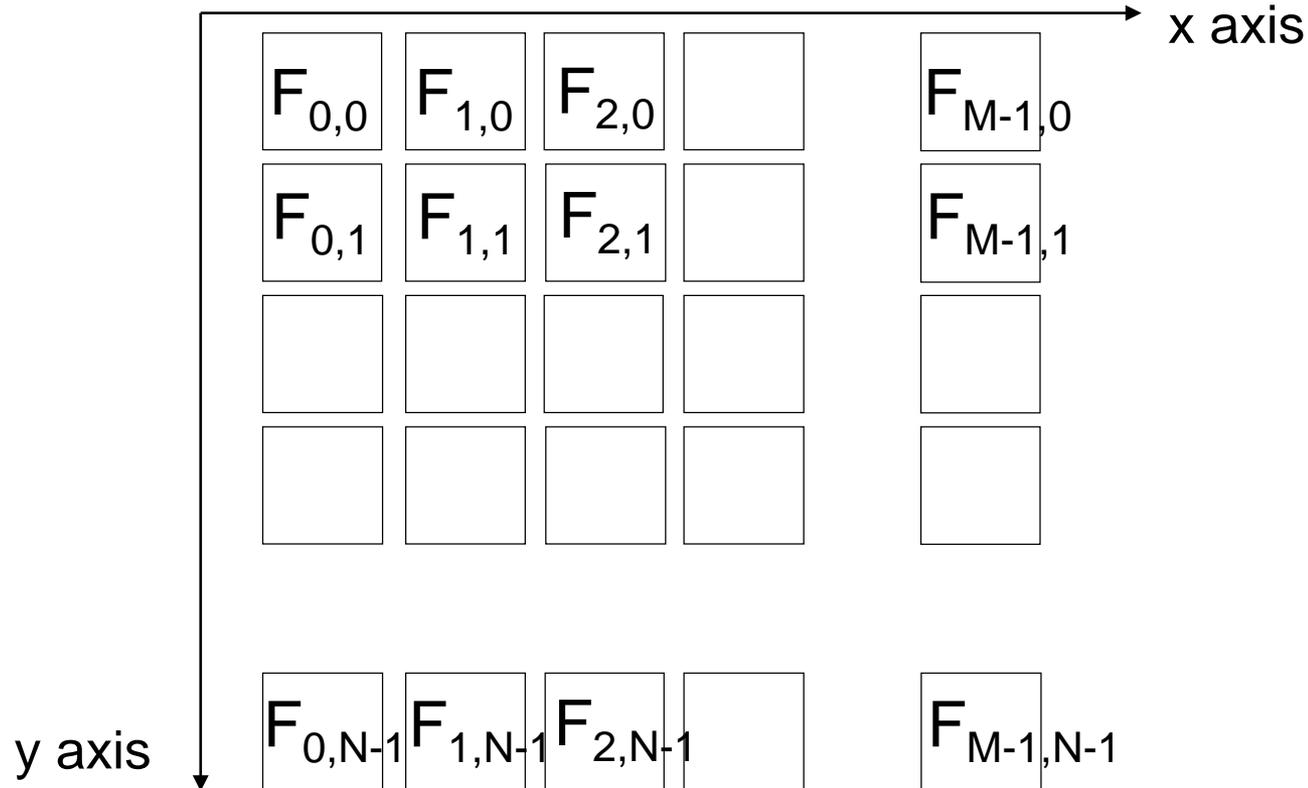
cf. binary image (= 1-bit grayscale)

Expression of a Digital Image

M × N pixels digital image:

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

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

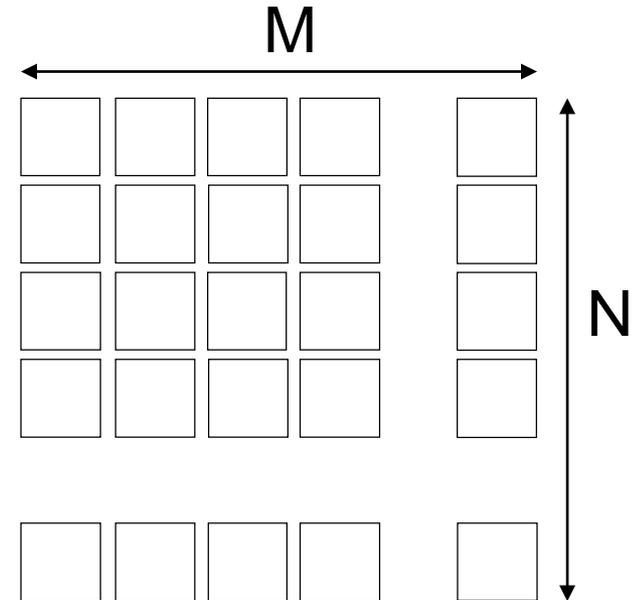


Example in C

```
#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

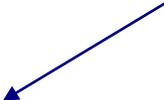
<https://www.python.org/>

A lightweight scripting language

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

In this lecture, its grammar is not explained in detail. (I hope the sample codes are understandable if you have some general knowledge of programming)

Recommended Installation Steps (Windows)

- It is recommended to set up an isolated Python execution environment for this class to try the sample codes
- The following steps will not affect any existing Python or OpenCV software environment
- Approx. 1 GB of disk space is needed
- Create a new folder `C:\ic2020`  arbitrary folder of your choice; avoid including white spaces
- If you are a git user, clone <https://github.com/shingo-kagami/ic.git> here
- Download and Install “Miniconda3 Windows 64-bit”
<https://docs.conda.io/en/latest/miniconda.html>
 - Install for “just me”
 - Destination: `C:\ic2020\miniconda3`
 - Uncheck all the Advanced Options
 - Go on with installation

Cont'd

- (If you didn't clone git repository) Download `ic_python_env.bat` (right-click "Raw") and put it in `C:¥ic2020` folder
https://github.com/shingo-kagami/ic/blob/master/ic_python_env.bat
- Run `C:¥ic2020¥ic_python_env.bat`
- In the command prompt opened, run the following commands

```
pip install opencv-python
pip install opencv-contrib-python
pip install numba
pip install matplotlib
pip install scipy
pip install spyder
```

Known Issue (does not affect our sample codes, hopefully)

```
ERROR: flake8 3.8.3 has requirement pyflakes<2.3.0,>=2.2.0, but
you'll have pyflakes 2.1.1 which is incompatible.
```

Running Sample Codes

(If you didn't clone git repository) Download sample codes, images, and movies (Click "Code" -> Download zip) and put them into `C:¥ic2020¥sample`

<https://github.com/shingo-kagami/ic>

Within the Command Prompt opened by `ic_python_env.bat`, the installed version of python is active.

```
cd C:¥ic2020¥sample
python 01_thresh.py
```

Or, you can launch an integrated development environment called Spyder.

```
spyder3
```

Note: not spyder but spyder3

If you want to change the language of Spyder, open in the Spyder menu:
Tools -> Preferences -> General -> Advanced Settings -> Language

Interactive Execution

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

```
cd C:/ic2020/sample/  
import cv2  
img = cv2.imread('lena.jpg')  
cv2.imshow('testwin', img)  
cv2.waitKey(0) Hit any key on the image window to proceed  
cv2.destroyAllWindows()  
  
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)

01_thresh.py:

```
import cv2
import numpy as np
```

OpenCV module
numpy module

```
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
```

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)           (window name, image)  
    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
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

01_thresh_video.py:

```
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

character code of 'q'

How to Switch the VideoCapture Source

```
import sys

...

cap_src = 'vtest.avi'
if len(sys.argv) == 2:
    if sys.argv[1].isdecimal():
        cap_src = int(sys.argv[1])
    else:
        cap_src = sys.argv[1]
cap = cv2.VideoCapture(cap_src)
```

sys.argv[0] is command name
sys.argv[1] is 1st argument

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 avoid iteration by using 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

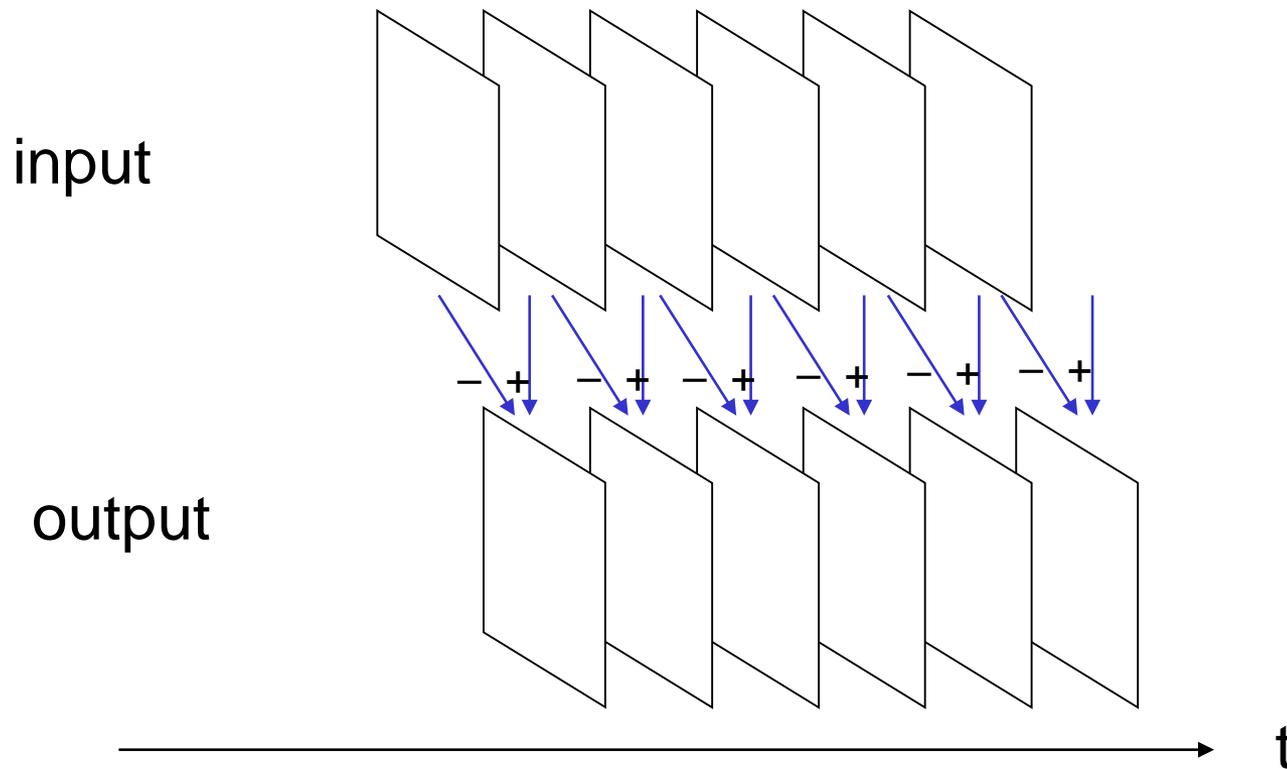
@jit
def threshold_impl(src, thresh, maxval):
    ...
```

Trackbars

01_thresh_video_trackbar.py:

```
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

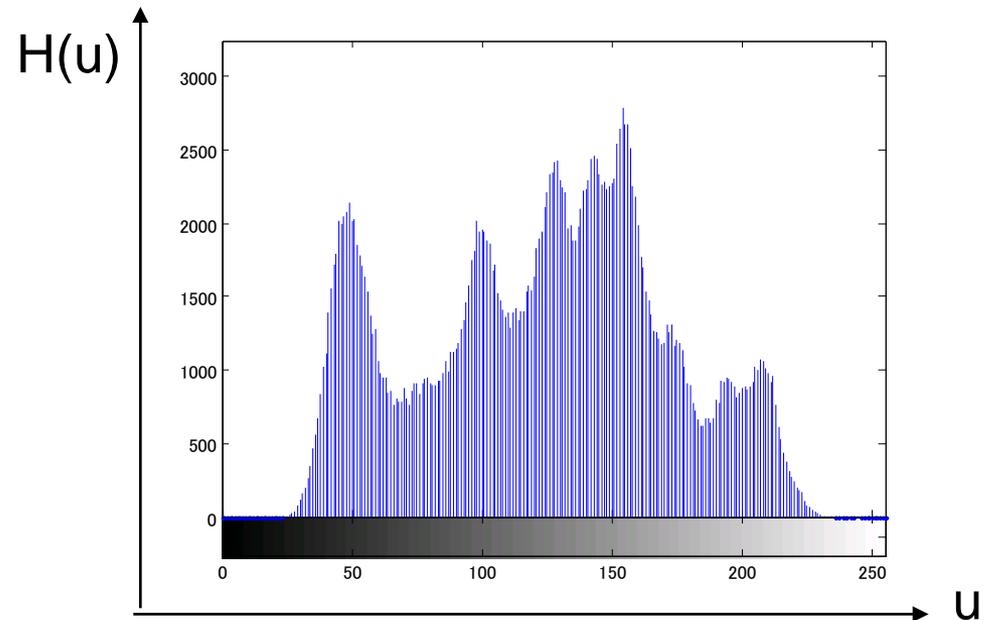
01_frame_diff.py:

```
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



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

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

01_histogram.py:

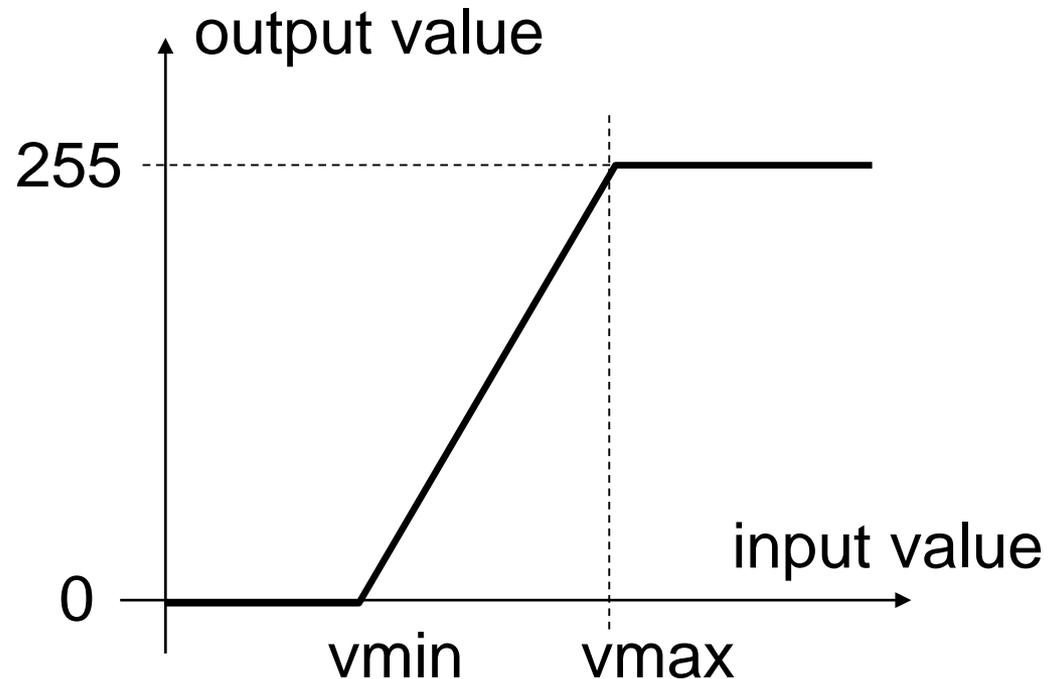
```
import matplotlib.pyplot as plt

def calcHistogram(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

pix_val_hist = calcHistogram(image)
plt.bar(range(256), pix_val_hist)
```

abscissa values ordinate values

Converting Pixel Values

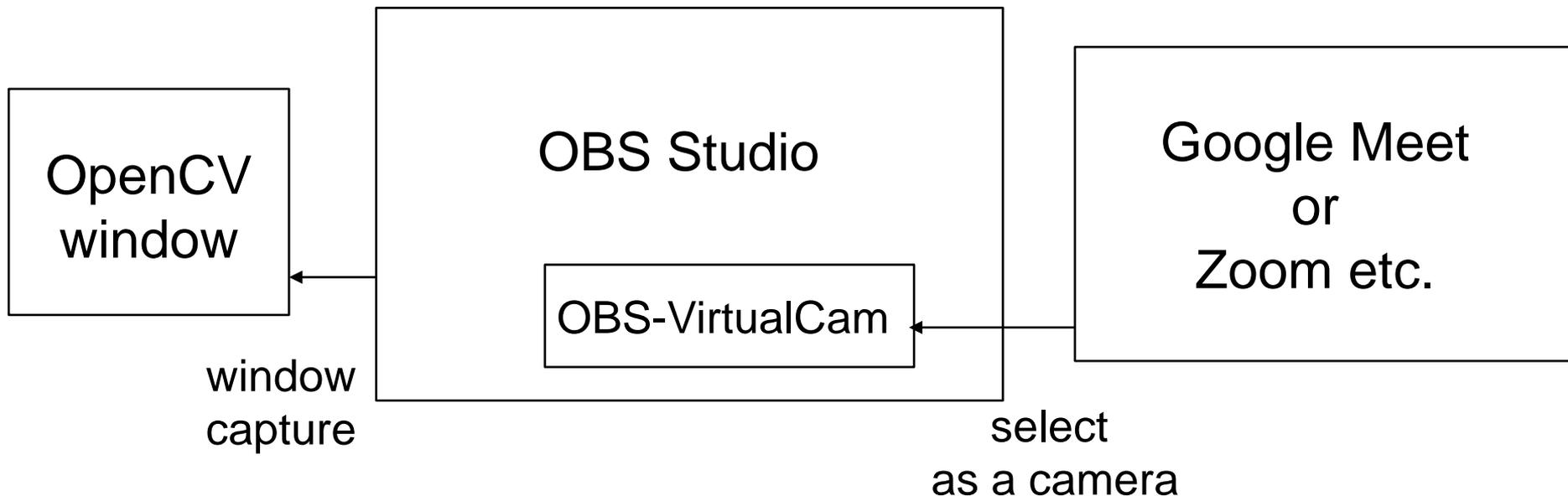


clip values between 0 and 255

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

Just a Fun Example

- OBS Studio (an open-source screen capture software)
- OBS-VirtualCam (a plugin for OBS Studio to simulate a webcam)



References

Sample codes are available at

<https://github.com/shingo-kagami/ic>

Reference manuals for OpenCV and numpy are in:

- <https://docs.opencv.org/master/>
 - https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
 - <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++