
Intelligent Control Systems

Image Processing (1)

— Basic Concepts and Introduction of OpenCV —

Shingo Kagami

Graduate School of Information Sciences,

Tohoku University

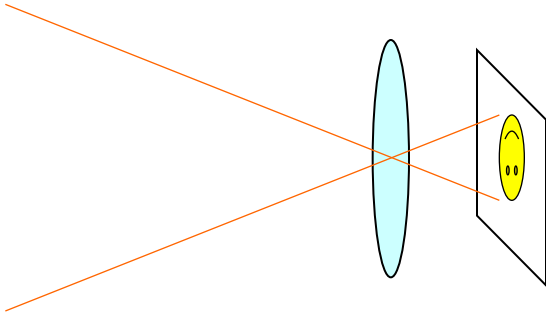
swk(at)ic.is.tohoku.ac.jp

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

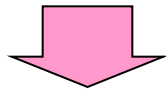
Outline

- General Concepts of Image Processing
- Image Processing Library: OpenCV
- Single Frame Image Processing
- Video Processing (i.e. Successive Multiple Frames)

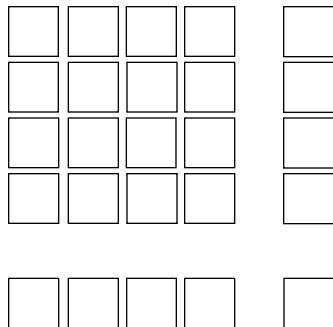
Digital Images



Analog distribution of light intensity



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

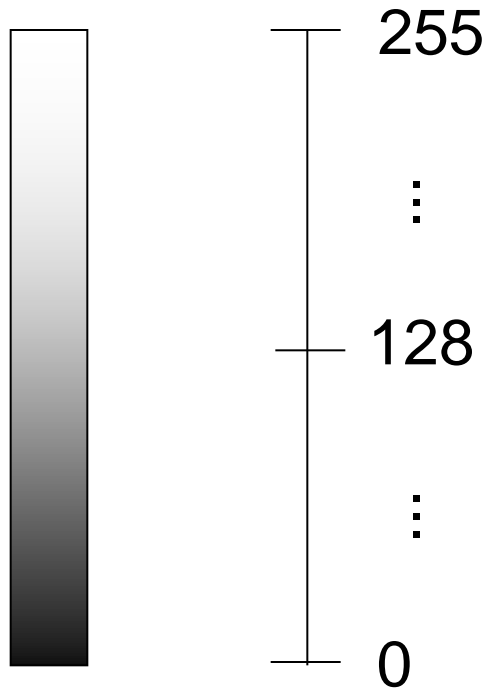


A digital image:
2-D array of pixel values

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

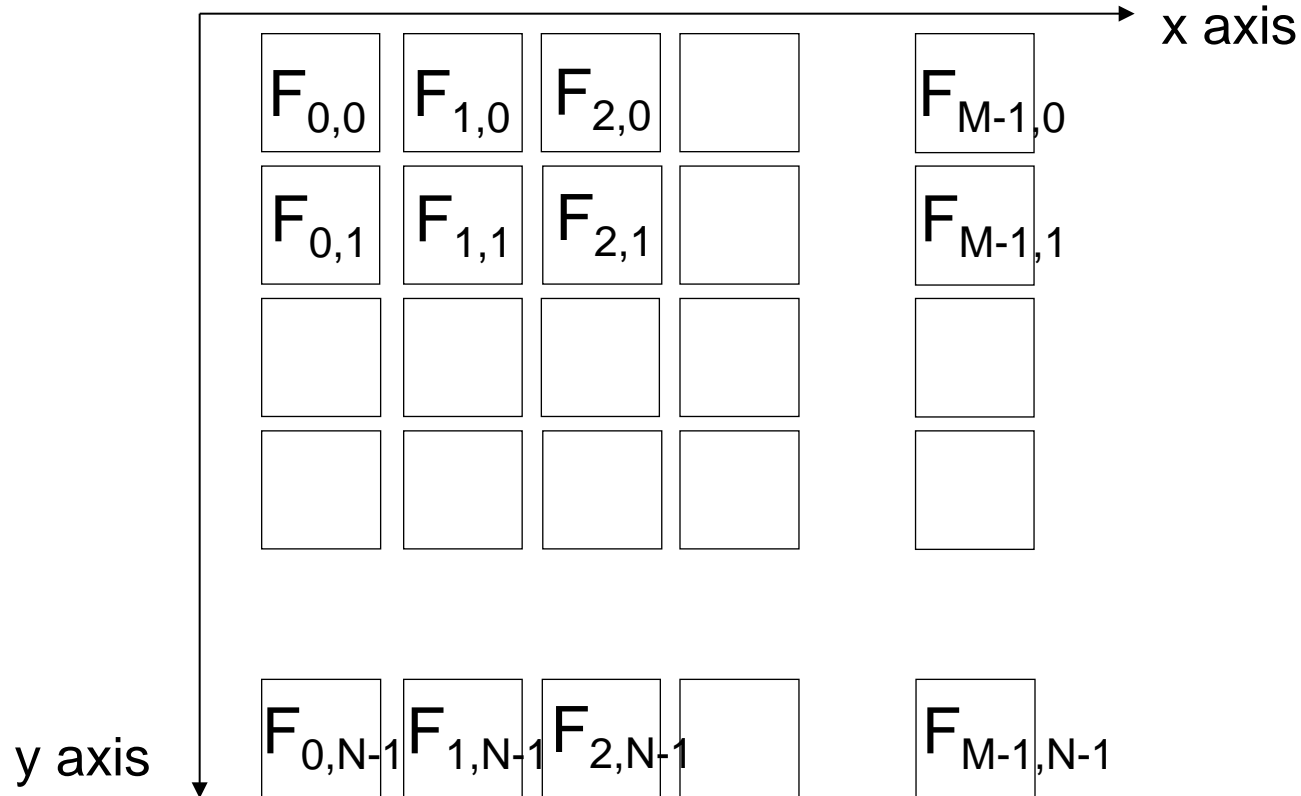
cf. binary image (= 1-bit grayscale)

Expression of a Digital Image

$M \times N$ pixels digital image:

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

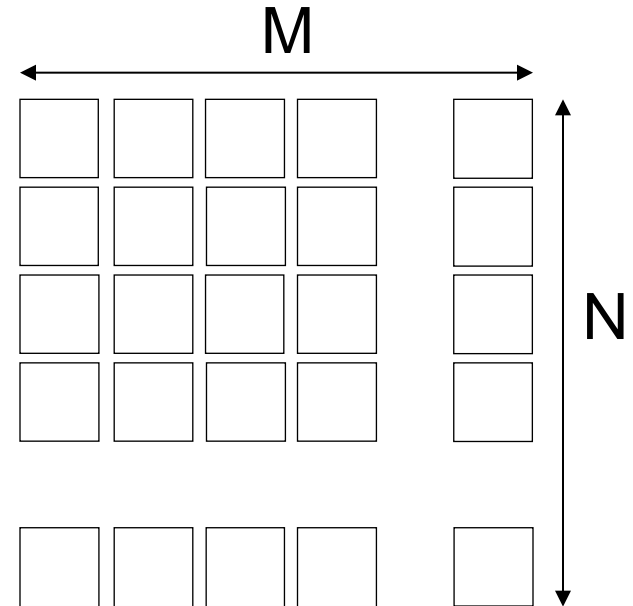
Pixel value at (x, y) : $F_{x,y}$



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

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;
        }
    }
}
```

Outline

- General Concepts of Image Processing
- Image Processing Library: OpenCV
- Single Frame Image Processing
- Video Processing (i.e. Successive Multiple Frames)

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

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.
- Main language: C++
 - also supports: python, Java

Are you OK with

C++?

Quotes

- C++ is Good for the Economy, It Creates Jobs! - Bjarne Stroustrup
- C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg. — Bjarne Stroustrup
- Being really good at C++ is like being really good at using rocks to sharpen sticks. — Thant Tessman
- If C++ has taught me one thing, it's this: Just because the system is consistent doesn't mean it's not the work of Satan. — Andrew Plotkin
- It has been discovered that C++ provides a remarkable facility for concealing the trival details of a program — such as where its bugs are. — David Keppel
- C++ is an insult to the human brain — Niklaus Wirth
- C++: an octopus made by nailing extra legs onto a dog. — Steve Taylor

C to C++

C++ may seem to be too complicated to completely understand.

But that's true in designing and implementing your own classes, and just using classes and functions written by anyone else can be much easier (as long as they are appropriately designed...)

This lecture will also provide minimal guides for migration from C to C++ whenever needed

Setting Up OpenCV

Download and Install the binary package for your platform from <http://opencv.org/>

Additionally, you will need:

- to add Library Path
- to add Include Path
- to add Executable Path
- to add Library Files

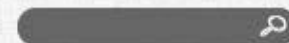
Go to

<http://docs.opencv.org/doc/tutorials/tutorials.html>

and see “Introduction to OpenCV” to find your platform guide



- ABOUT
- DOWNLOADS
- DOCUMENTATION
- PLATFORMS
- SUPPORT
- CONTRIBUTE



DONATE



OPENCV (OPEN SOURCE COMPUTER VISION)

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

QUICK LINKS:

- [Online documentation](#)
- [User Q&A forum](#)
- [Report a bug](#)
- [Build farm](#)
- [Books](#)
- [Developer site](#)

LATEST DOWNLOADS

2016-05-19
VERSION 2.4.13

- [OpenCV for Windows](#)
- [OpenCV for Linux/Mac](#)
- [OpenCV for iOS](#)

My Environment for Demonstration

- Microsoft Windows 7 Professional SP1, 64 bit
- Microsoft Visual Studio 2012 Express Edition (msvc11)
 - free!
- OpenCV 2.4.13
 - prebuilt binaries are extracted into C:\OpenCV2.4.13 (so that we have C:\OpenCV2.4.13\build and C:\OpenCV2.4.13\sources folders)

Detailed Setup for msvc11 (1/2)

For each **project property**:

- In **Configuration Properties – Linker – General – Additional Library Directories**, add:
 - `$(OPENCV_DIR)\lib`
- In **Configuration Properties - C/C++ - General – Additional Include Directories**, add:
 - `$(OPENCV_DIR)\..\..\include`
- In **Configuration Properties – Linker – Input – Additional Dependencies**, add (at least):
 - `opencv_core2413.lib opencv_imgproc2413.lib`
`opencv_highgui2413.lib`

OpenCV version

With **Windows environments variables**:

OPENCV_DIR: set to `C:\OpenCV2.4.13\build\x64\vc11`

PATH: including `%OPENCV_DIR%\bin`

x86 for Win32

vc12 for Visual Studio 2013

Detailed Setup for msvc11 (2/2)

- For **Debug** configuration (instead of Release), specify
opencv_core2413d.lib opencv_imgproc2413d.lib
opencv_highgui2413d.lib
 - In general, the programs built in Release Configuration is much far faster than the ones in Debug Config.
- Sometimes we want to see console outputs after the program terminates. In order to keep the console window from vanishing, set **Configuration Properties – Linker – System – Subsystem** to /SUBSYSTEM:CONSOLE
- **File – Export Template** may be useful in avoiding tedious project-wise setups

Outline

- General Concepts of Image Processing
- Image Processing Library: OpenCV
- Single Frame Image Processing
- Video Processing (i.e. Successive Multiple Frames)

Typical Image Processing Code (for a single frame)

```
#include "opencv2/opencv.hpp"

int main()
{
    cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_GRAYSCALE);
    cv::Mat output;

    cv::threshold(input, output, 128, 255, cv::THRESH_BINARY);

    cv::namedWindow("disp", cv::WINDOW_AUTOSIZE);
    cv::imshow("disp", output);
    cv::waitKey(0);

    return 0;
}
```

C++ namespace

```
#include "opencv2/opencv.hpp"

int main()
{
    cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_GRAYSCALE);
    cv::Mat output;

    cv::threshold(input, output, 128, 255, cv::THRESH_BINARY);

    cv::namedWindow("disp", cv::WINDOW_AUTOSIZE);
    cv::imshow("disp", output);
    cv::waitKey(0);

    return 0;
}
```

`cv::` is a namespace prefix to avoid the conflicts of names of functions, types, variables etc. Do you feel they are annoying?

C++ namespace

```
#include "opencv2/opencv.hpp"

int main()
{
    using namespace cv;
    Mat input = imread("lena.jpg", IMREAD_GRAYSCALE);
    Mat output;

    threshold(input, output, 128, 255, THRESH_BINARY);

    namedWindow("disp", WINDOW_AUTOSIZE);
    imshow("disp", output);
    waitKey(0);

    return 0;
}
```

using namespace directive can be used for compact codes (but with higher risk of conflicts).

cv::Mat type

```
#include "opencv2/opencv.hpp"

int main()
{
    int thresh = 128, max = 255;
    cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_GRAYSCALE);
    cv::Mat output;

    cv::threshold(input, output, thresh, max, cv::THRESH_BINARY);

    cv::namedWindow("disp", cv::WINDOW_AUTOSIZE);
    cv::imshow("disp", output);
    cv::waitKey(0);

    return 0;
}
```

cv::Mat is a class for a matrix or an image defined in OpenCV. `input` and `output` are instances of class `cv::Mat`, just as `thresh` and `max` are instances of type `int`.

C++ class

An extension to C struct:

```
struct MyComplexInt {
    int re;
    int im;
};

int main()
{
    struct MyComplexInt x;
    x.re = 10;
    x.im = 20;
}
```

C struct can have member variables

C++ class

```
class MyComplexInt {
public:
    int re;
    int im;
    double magnitude() {
        return sqrt(re * re + im * im);
    }
};

int main()
{
    MyComplexInt x;
    x.re = 10; x.im = 20;
    double mag = x.magnitude();
}
```

C++ class can also have member functions (methods).

C++ reference type

```
cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_GRAYSCALE);  
cv::Mat output;
```

```
cv::threshold(input, output, thresh, max, cv::THRESH_BINARY);
```

how is it possible to write to output?

```
void conjugate(MyComplexInt& a) {  
    a.im = -a.im;  
}
```

```
int main()  
{  
    MyComplexInt x;  
    conjugate(x);  
}
```

a becomes a reference (alias)
to the real argument x, so x in
the main becomes modified.

Thresholding Implementation Example

```
int main()
{
    int thresh = 128, max = 255;
    cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_GRAYSCALE);
    int cols = input.cols, rows = input.rows;
    cv::Mat output(rows, cols, CV_8U);

    for (int j = 0; j < rows; j++) {
        for (int i = 0; i < cols; i++) {
            if (input.at<uchar>(j, i) >= thresh) {
                output.at<uchar>(j, i) = max;
            } else {
                output.at<uchar>(j, i) = 0;
            }
        }
    }

    ...
}
```

C++ constructor/destructor

```
class MyComplexInt {  
public:  
    MyComplexInt(int r, int i) { re = r; im = i; }  
    MyComplexInt() { re = 0; im = 0; }  
    ~MyComplexInt() { /* do nothing */ }  
    int re;  
    int im;  
};
```

overloading
is allowed

```
int main()  
{  
    MyComplexInt x(10, 20); // constructor called  
} // destructor called
```

A constructor and destructor are called when an instance is generated and is expired, respectively, so that your library can do initialization and cleanup automatically (if needed).

C++ template

```
class MyComplexInt {
public:
    int re;
    int im;
};
```

```
class MyComplexDouble {
public:
    double re;
    double im;
};
```

```
class MyComplexUchar {
public:
    uchar re;
    uchar im;
};
```

...

Tedious!

C++ template

```
template<typename T>
class MyComplex {
public:
    T re;
    T im;
};

int main( )
{
    MyComplex<int> x;
    MyComplex<uchar> y;
    MyComplex<double> z;
}
```

The compiler automatically does kind of copy&paste of codes.
Can be used not only for classes but for functions, methods, etc.

Now you make out ...

```
cv::Mat output(rows, cols, CV_8U);
```

Constructor of `cv::Mat` is called with arguments `rows`, `cols` (`rows × cols` matrix), and `CV_8U` (8-bit unsigned).

```
for (int j = 0; j < rows; j++) {
    for (int i = 0; i < cols; i++) {
        if (input.at<uchar>(j, i) >= thresh) {
            output.at<uchar>(j, i) = max;
        } else {
            output.at<uchar>(j, i) = 0;
        }
    }
}
```

Template method `at` of class `cv::Mat` is called with template argument `uchar`. (to access the specified pixel value considering each pixel is 8-bit unsigned)

Outline

- General Concepts of Image Processing
- Image Processing Library: OpenCV
- Single Frame Image Processing
- Video Processing (i.e. Successive Multiple Frames)

Typical Image Processing Code (for video frames)

```
int main()
{
    cv::VideoCapture cap(0);
    cv::Mat input, output;
    cv::namedWindow("disp", cv::WINDOW_AUTOSIZE);

    while (1) {
        cap >> input;
        cv::cvtColor(input, input, CV_BGR2GRAY);
        cv::threshold(input, output, 128, 255, cv::THRESH_BINARY);
        cv::imshow("disp", output);
        if (cv::waitKey(30) > 0) {
            break;
        }
    }

    return 0;
}
```

0-th camera;
You can also specify a video filename here

wait for 30 ms at maximum.
If any key is hit within 30 ms, keycode is returned

C++ operator overload

```
class MyComplexInt {  
public:  
    int re;  
    int im;  
};
```

```
MyComplexInt operator+(MyComplexInt lhs, int rhs) {  
    MyComplex ret;  
    ret.re = lhs.re + rhs; ret.im = lhs.im;  
    return ret;  
}
```

```
int main()  
{  
    MyComplexInt x; x.re = 10; x.im = 20;  
    MyComplexInt y = x + 5;  
}
```

Summary

- Basic structure of image processing:
 - for a single frame
 - for a multiple (successive) frames
- Implementation examples in OpenCV

- Basic Concepts of C++:
 - namespace
 - class (constructor / destructor)
 - reference type
 - template
 - function overload, operator overload

References

- R. Szeliski: Computer Vision: Algorithms and Applications, Springer, 2010.
- A. Hornberg eds.: Handbook of Machine Vision, Wiley-VCH, 2006.
- G. Bradski and A. Kaebler: Learning OpenCV, O'Reilly, 2008.
- OpenCV Documentation: <http://docs.opencv.org/index.html>

(in Japanese)

- デジタル画像処理編集委員会, デジタル画像処理, CG-ARTS協会, 2015.
- 田村: コンピュータ画像処理, オーム社, 2002.

Sample codes are available at

<http://www.ic.is.tohoku.ac.jp/~swk/lecture/>