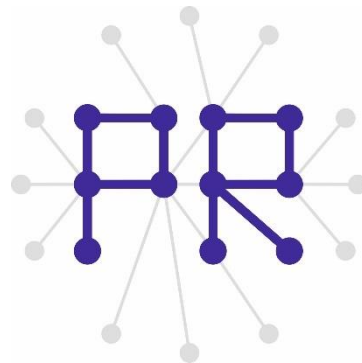


Multimedia Retrieval Exercise Course

2 Basic Knowledge about Images in OpenCV

Kimiaki Shirahama, D.E.

Research Group for Pattern Recognition
Institute for Vision and Graphics
University of Siegen, Germany



Last lesson's Program

- Read an image and show it in a dialog

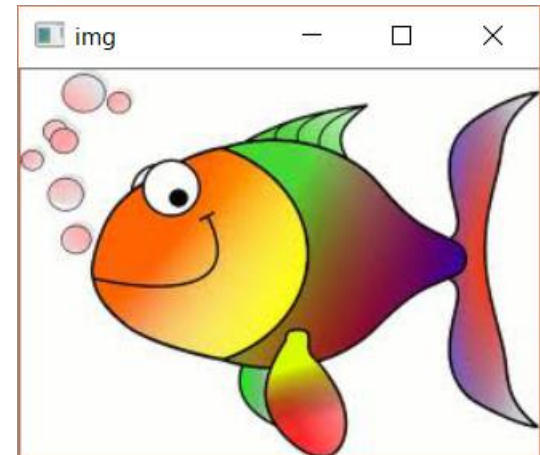
```
#include "stdafx.h" // This is a head file specific to my project
```

```
#include <opencv2/core.hpp>
```

```
#include <opencv2/highgui.hpp>
```

```
int main(int argc, const char* argv[]){  
    cv::Mat img = cv::imread("C:\\\\opencv\\\\sources\\\\samples\\\\data\\\\HappyFish.jpg", 1);  
    cv::namedWindow("img", cv::WINDOW_AUTOSIZE);  
    cv::imshow("img", img);  
    cv::waitKey(0);  
    cv::destroyAllWindows();  
    return 0;  
}
```

Please change the image filename depending on your environment.



Overview of Today's Lesson

1. cv::Mat

- Structure of cv::Mat
- Printing out the Header Information
- How are pixel values stored with cv::Mat
- Printing out Pixel Values
- Shallow and deep copies

2. Useful functions in highgui

OpenCV Information

Useful web page: <http://opencv.org/documentation.html>
(Especially, <http://docs.opencv.org/3.3.0/>)

opencv2/

- core.hpp: Basic data structure and functions in OpenCV
 - imgproc.hpp: Functions for image processing and analysis
 - imgcodecs.hpp: Functions for loading and writing image files
 - videoio.hpp: Functions for loading and writing video files
 - highgui.hpp: Functions for creating and managing GUIs
(highgui.hpp includes core.hpp, imgcodecs.hpp and videoio.hpp)
 - ml.hpp: Functions for machine learning methods
 - dnn.hpp: Functions for deep learning methods
- etc.

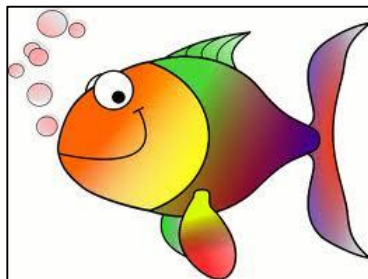
cv::Mat

Container for storing an image

- **Data header**

- cols: Image width
- rows: Image height
- dims: Number of dimensions
(An image has 2 dimensions corresponding to the row and column)
- channels: Number of channels used to represent a pixel
(3 channels for an RGB image, 1 channel for a gray-scale image)
- depth: Data type used to represent one channel
(A usual image where the R/G/B value of a pixel range from 0 to 255.
This means one channel is represented by "8-bit unsigned char (integers)")

- **Pointer to data**

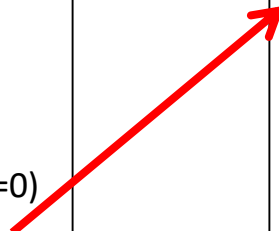


Memory (RAM)

cv::Mat

cols:259
rows: 194
dims: 2
channels: 3
depth: CV_8U (=0)
Pointer to data

Actual data (pixel values)



Printing out the Header Information

Please add to the last lesson's code some sentences to show image's width, height, dimensionality, channels and depth.

```
cout << ">> Width: " << img.cols << ", height:" << img.rows << ", dims:" << img.dims << endl;  
cout << ">> Depth:" << img.depth() << "(=" << CV_8U << "), channels:" << img.channels() << endl;
```

1. Please refer to http://docs.opencv.org/3.3.0/d3/d63/classcv_1_1Mat.html for how to access each piece of information.
2. To output texts to a terminal, “cout” in C++ is more convenient than “printf” because the former can automatically recognise the variable types.

```
#include <iostream>
```

```
using namespace std; // If this is missing, “cout” has to be written as “std::cout”
```

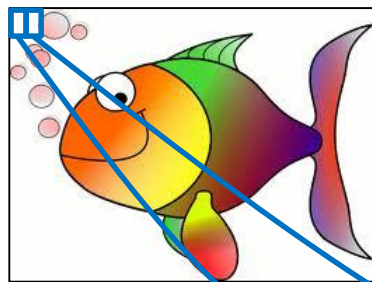
3. In Visual C++, a terminal will quickly disappear after reaching the end of a program. To keep the terminal appearing, one simple approach is as follows:

```
int a = 0;
```

```
cin >> a; // Wait until some value is input into “a”
```

cv::Mat in More Detail

3-channel color image



Memory (RAM)

cv::Mat

cols:259
rows: 194
dims: 2
channels: 3
depth: CV_8U (=0)
Pointer to data

Actual data (pixel values)

B G R B G R - - - -

Each pixel with three channels is represented by three values (*the order is BGR*)

For 8-bit depth images, it is recommended to access one pixel (

B	G	R
---	---	---

) using cv::Vec3b.

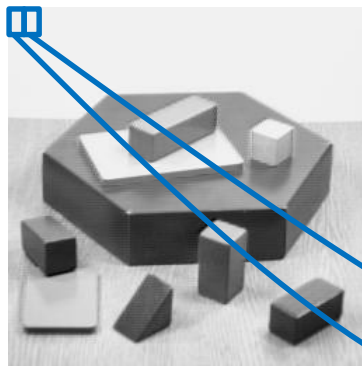
- Get the values of the pixel (x, y)
`cv::Vec3b p = img.at<cv::Vec3b>(y, x);`
Then, B, G and R values of (x, y) are p[0], p[1] and p[2], respectively.

- Get the pointer to the yth row
`cv::Vec3b *row = img.ptr<cv::Vec3b>(y);`
Then, B, G and R values of (x, y) are row[x][0], row[x][1] and row[x][2], respectively.

For another depth image, refer to http://docs.opencv.org/3.3.0/dc/d84/group_core_basic.html

For Your Information

Gray-scale image

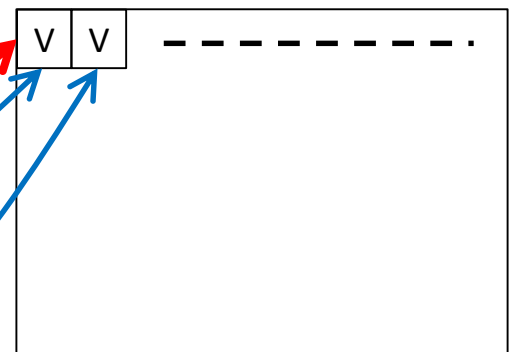


Memory (RAM)

cv::Mat

cols:256
rows: 256
dims: 2
channels: 1
depth: CV_8U (=0)
Pointer to data

Actual data (pixel values)



Each pixel has only one channel value (V)

For 8-bit depth images, it is recommended to access one pixel using uchar.

- Get the values of the pixel (x, y)
uchar p = img.at<uchar>(y, x);
- Get the pointer to the yth row
uchar *row = img.ptr<uchar>(y);
Then, the value of (x, y) are row[x].

Printing out Pixel Values

Please add to the last lesson's code sentences to show the values of each pixel

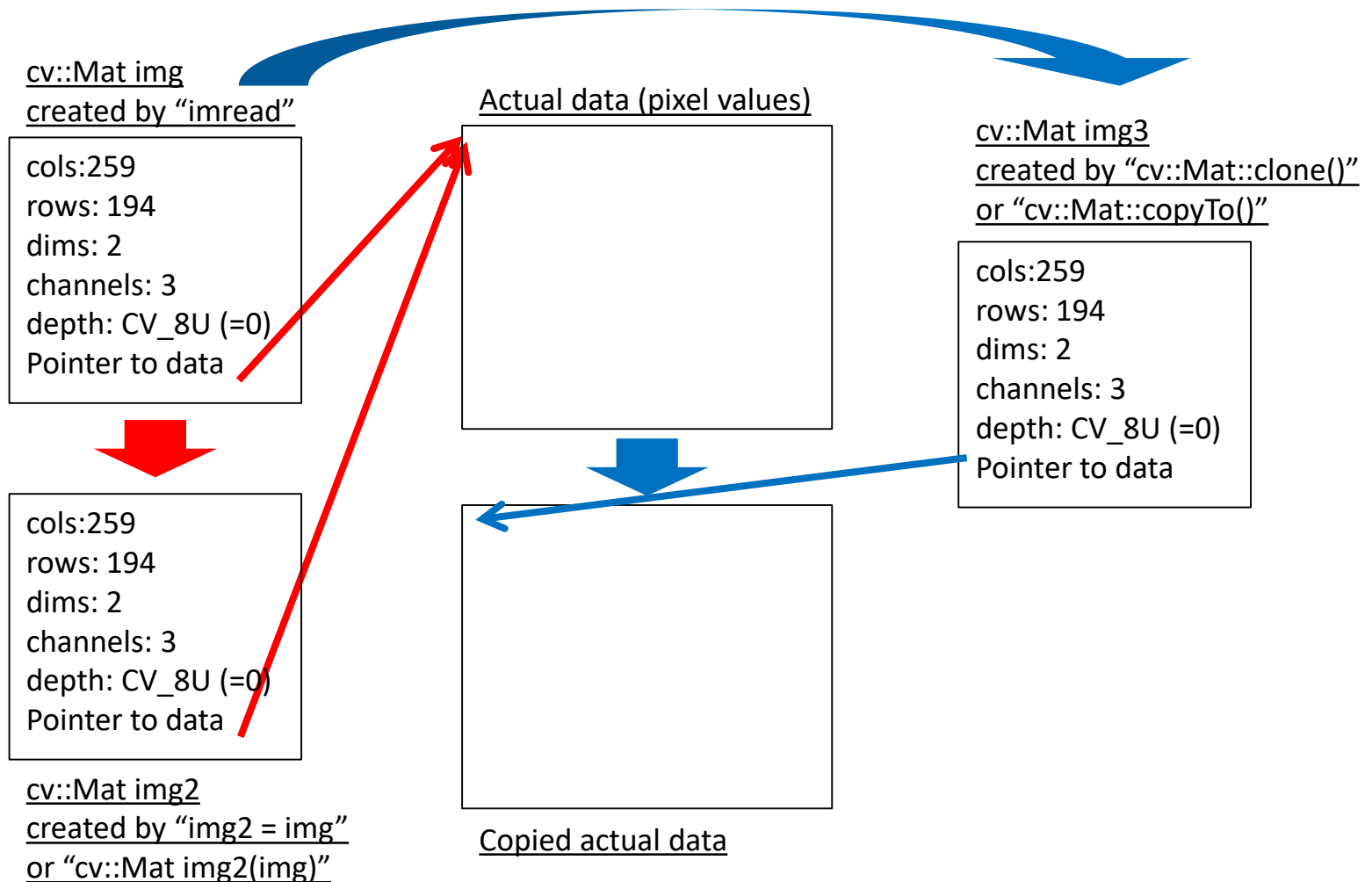
```
cout << "##### Using at #####" << endl;
for (int y = 0; y < img.rows; y++) {
    for (int x = 0; x < img.cols; x++) {
        cv::Vec3b p = img.at<cv::Vec3b>(y, x);
        if ( (y == 0 || y == img.rows - 1) && (x < 5 || x > img.cols - 5) )
            printf("(%d,%d): %u, %u, %u\n", x, y, p[0], p[1], p[2]);
    }
}

cout << "##### Using ptr #####" << endl;
for (int y = 0; y < img.rows; y++) {
    cv::Vec3b *row = img.ptr<cv::Vec3b>(y);
    for (int x = 0; x < img.cols; x++) {
        if ((y == 0 || y == img.rows - 1) && (x < 5 || x > img.cols - 5))
            printf("(%d,%d): %u, %u, %u\n", x, y, row[x][0], row[x][1], row[x][2]);
    }
}
```

NOTE: Variable type recognition of “cout” does not work for elements (unsigned char) of Vec3b. Use “printf” to show pixel values.

Shallow and Deep Copies

- **Shallow copy:** Only the header is copied
- **Deep copy:** Both of the header and actual data are copied



Useful Functions in highgui

1. void cv::namedWindow(const String &winname, int flags=WINDOW_AUTOSIZE)

The first variable specifies the name of the window where an image is shown.

2. void cv::imshow(const String &winname, InputArray mat)

Using the first variable, you can specify which window is used to show an image (mat)

3. void cv::destroyAllWindows(), void cv::destroyWindow(const String &winname)

Release the memory used for the window

4. int cv::waitKey (int delay=0)

The function is used to wait for some key input or “delay” milliseconds.

The return value represents which key is pressed.

For more detail, refer to https://docs.opencv.org/3.3.0/d7/dfc/group_highgui.html