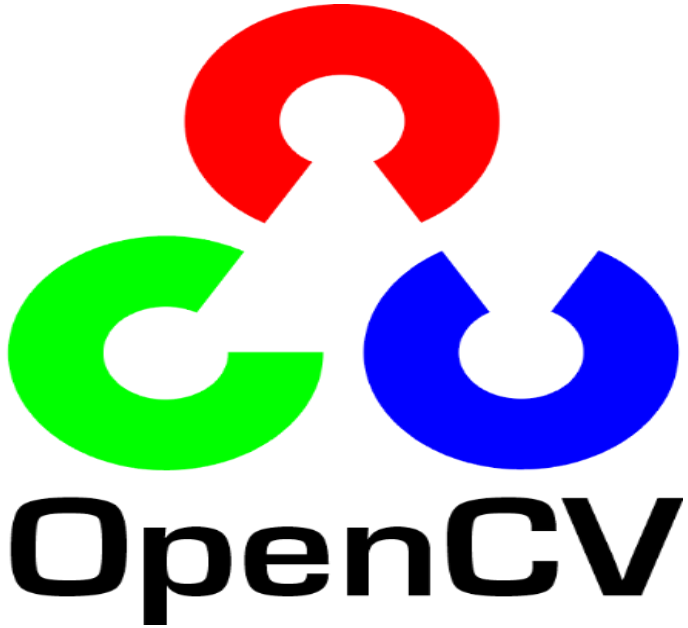
Computer Vision

Ch2. OpenCV Introduction

Prof. Po-Yueh Chen (陳伯岳)
E-mail: pychen@cc.ncue.edu.tw
Ext: 8440
NCUE CSIE



Website: https://opencv.org/

OpenCV Introduction (1/2)

✓ OpenCV (*Open source computer vision*)

- **OpenCV is a powerful library of programming functions.**
  - **Real-time**
  - **Deep Learning**

OpenCV Introduction (2/2)

- **In this course we are using *ver. 4.0.1.***

OpenCV 4.0.1 Document site:

- **https://docs.opencv.org/4.0.1/**

C++ with OpenCV beginning (1/6)

➢ Library & Header

- #include <iostream>
- #include <string>
- #include <sstream>
- using namespace std;

- #include <opencv2/opencv.hpp>
- #include <opencv2/core.hpp>
- #include <opencv2/highgui.hpp>
- using namespace cv;

C++ with OpenCV beginning (2/6)

➢ Open file

- **imread(".../Data.jpg");**

File path

Ex: **Mat** Example = imread("D:/Data/MyData/MyCode/Data.jpg");

C++ with OpenCV beginning (3/6)

➢ Open file

➢ **Mat Lady = imread(".../Lena.jpg");**

File path

Ex: Mat Example = imread("D:/Data/MyData/MyCode/Data.jpg");

➢ **Mat Dog = imread(".../Corgi.jpg");**

C++ with OpenCV beginning (4/6)

- **Show the executed result of image**

  - **namedWindow("title_name");**
    // Create a window and name the window's title.

  - **moveWindow("title_name", x, y);**
    // To set up the window's coordinate on the monitor.

  - **imshow("title_name", variable);**
    // Show the image in the window.

C++ with OpenCV beginning (5/6)

- **Show the executed result of image**

  - **namedWindow("Lena", WINDOW_GUI_NORMAL);**
    //To allow user can adjust the size of window.
  - **resizeWindow("Lena", 512, 512);**

  - **namedWindow("Corgi", WINDOW_AUTOSIZE);**
    //To auto-size the window of image, user cannot adjust size.

C++ with OpenCV beginning (6/6)

- **Stop & Shutdown windows**

  - **waitKey(0);**

  - **destroyWindow(“Example_name");**

  - **destroyAllWindows();**

➢ Codes for Demo#1

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
#include <opencv2/opencv.hpp>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
using namespace cv;

int main(){
        Mat Lena = imread("D:/lena.jpg");
        Mat Dog = imread("D:/Corgi.jpg");
```

➢ Codes for Demo#1

```
namedWindow("Lena", WINDOW_GUI_NORMAL);
namedWindow("Dog", WINDOW_AUTOSIZE);
moveWindow("Lena", 10, 10);
moveWindow("Dog", 520, 10);

imshow("Lena", Lena);
resizeWindow("Lena", 512, 512);
imshow("Dog", Dog);

waitKey(0);
destroyWindow("Lena");
destroyWindow("Dog");
return 0;
}
```

➢ Codes for Demo#1(Extended Practice)

```
//Multi-windows generated test
for (int i = 0; i < 10; i++)
{
        ostringstream dogN;
        dogN << "Dog" << i;
        namedWindow(dogN.str());
        moveWindow(dogN.str(), 20 * i, 20 * i);
        imshow(dogN.str(), Dog);
}
        waitKey(0);
        destroyAllWindows();
```

Demo#1

Practice

OpenCV Basic Types (1/7)

- **Scalar**
  - ✓ **For pixel colors.**

Syntax:

```
Scalar(B_value, G_value, R_value);
// Scalar(Blue level-value, Green level-value, Red level-value)

Scalar(gray_value); // Scalar(Gray level-value)
```

OpenCV Basic Types (2/7)

- **Scalar**
  - ✓ **For pixel colors.**

Ex:

```
Scalar(60, 120, 180);
// Scalar(Blue level-value, Green level-value, Red level-value)

Scalar(168); // Scalar(Gray level-value)
```

OpenCV Basic Types (3/7)

- **Point**
  - ✓ **For specific pixel.**

Syntax:

```
Point p_name(int x, int y);

Point p_name;
p_name.x = int x;
p_name.y = int y;
```

Ex:

```
Point pt1(40, 50);
```

Ex:

```
Point pt2;
pt2.x = 30;
pt2.y = 60;
```

OpenCV Basic Types (4/7)

- **Size**
  - ✓ **For image size (aspect).**

Syntax:

```
Size S_name(int width, int height);
```

Ex:

```
Size size1(30, 60);
```

Ex:

```
Size size2;
size2.width = 40;
size2.height = 50;

int myArea = size2.area();
```

OpenCV Basic Types (5/7)

- **Rect**
  - ✓ **For a rectangular region in an image.**

Syntax:

```
Rect(int x, int y, int width, int height);
```

Ex:

```
Rect rect1(30,60,300,200);
```

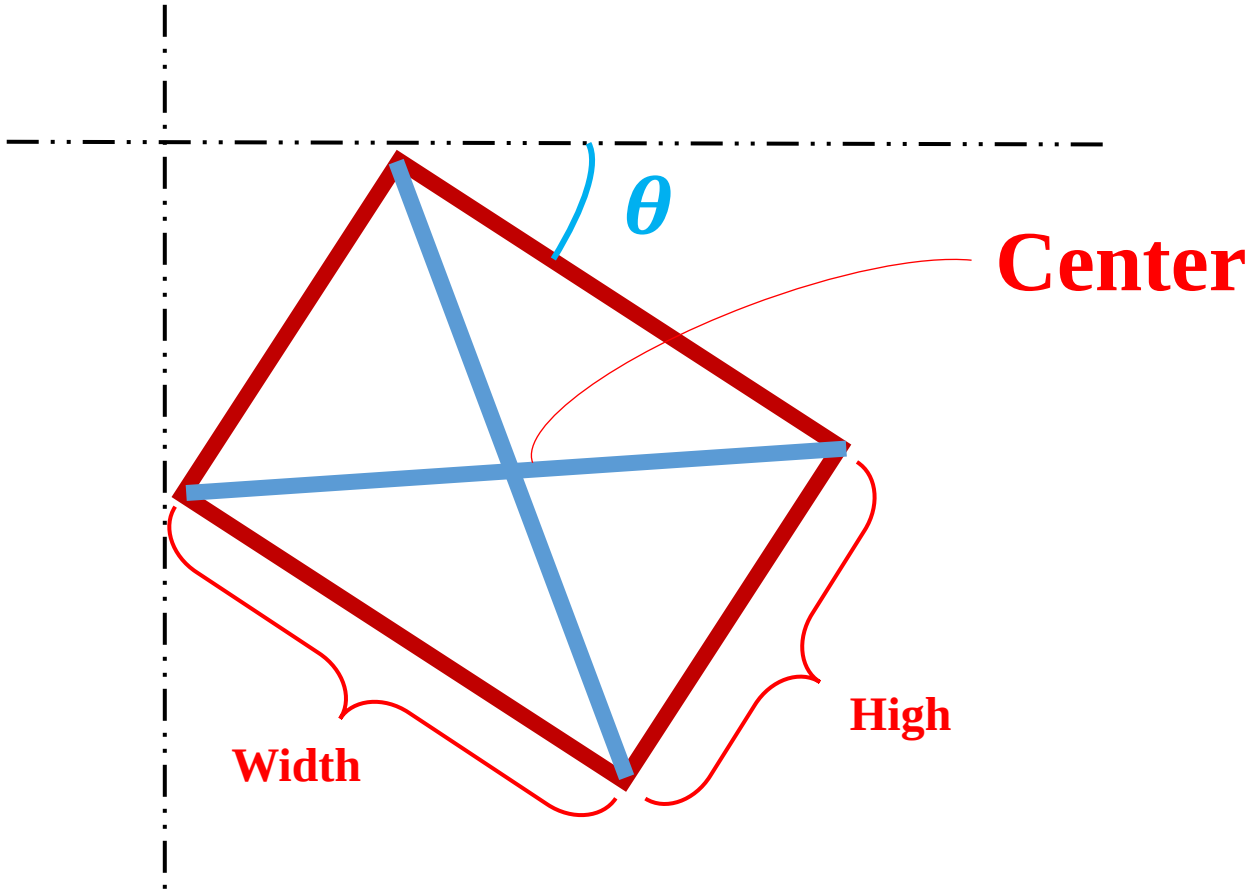
Ex:

```
Rect rect2;
rect2.x = 50;
rect2.y = 60;
rect2.width = 100;
rect2.height = 200;
```

OpenCV Basic Types (6/7)

➢ RotatedRect

- **center(Point2f)**
- **size(Size2f)**
- **angle(float)**



OpenCV Basic Types (7/7)

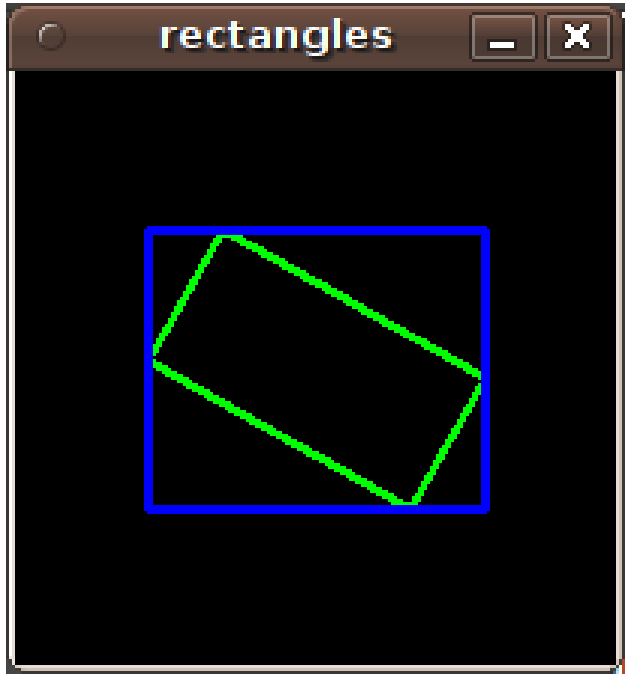
➢ RotatedRect

Ex:

```
Mat test_image(200, 200, CV_8UC3, Scalar(0));
RotatedRect rRect = RotatedRect(Point2f(100,100), Size2f(100,50), 30);

Point2f vertices[4];  //Declare 4 spaces for vertices.
rRect.points(vertices);
for (int i = 0; i < 4; i++){
        line(test_image, vertices[i], vertices[(i+1)%4], Scalar(0,255,0), 2);
}
Rect brect = rRect.boundingRect(); //Define the blue rectangle's position.
rectangle(test_image, brect, Scalar(255,0,0), 2); //Draw the blue rectangle.

imshow("rectangles", test_image);
waitKey(0);
```

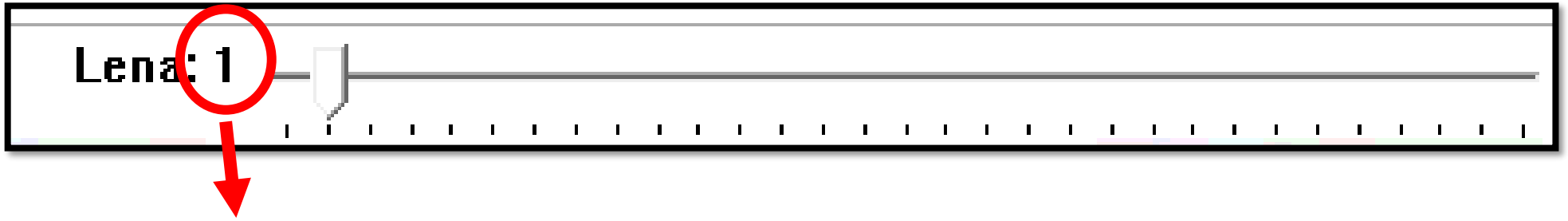


Source: https://docs.opencv.org/3.4.3/db/dd6/classcv_1_1RotatedRect.html

Practice

OpenCV GUI (1/8)

- **Track bar event**
  - ✓ **Create a Track bar**



Syntax:

createTrackbar("trackbar_name", "window_name", int *value, int count, onChange, void* data);

Ex:

createTrackbar(“Lena", “Lady", &blurAmount, 30, onChange, &data);

OpenCV GUI (2/8)

- **Track bar event**
  - ✓ **Get the track bar position.**

Syntax:

```
getTrackbarPos("trackbar_name", "window_name");
```

Ex:

```
getTrackbarPos ("Corgi", "Dog");
```

OpenCV GUI (3/8)

➢ Mouse events

CV_EVENT_MOUSEMOVE：Mouse moves.

CV_EVENT_LBUTTONDOWN：Left Button clicks.

CV_EVENT_RBUTTONDOWN：Right Button clicks.

CV_EVENT_MBUTTONDOWN：Middle Button clicks.

CV_EVENT_LBUTTONUP： Left Button release.

CV_EVENT_RBUTTONUP：Right Button release.

CV_EVENT_MBUTTONUP：Middle Button release.

CV_EVENT_LBUTTONDBLCLK： Left Button double clicks.

CV_EVENT_RBUTTONDBLCLK： Right Button double clicks.

CV_EVENT_MBUTTONDBLCLK： Middle Button double clicks.

OpenCV GUI (4/8)

- **Mouse event**
  - ✓ **To detect the mouse motion.**

Syntax:

setMouseCallback("window_name", Mouse call back status, void* data);

Ex:

setMouseCallback("Corgi", onMouse, &data);

OpenCV GUI (5/8)

- **Mouse event**
  - ✓ **To make a mark on the window.**

Syntax:

```
circle(Mat& img, Point center, int radius, const Scalar& color, int thickness);
```

Scalar(B, G, R)

Enter a negative value or CV_FILLED to fill the circle.

OpenCV GUI (6/8)

- **Demo**

```
int blurAmount = 1;
int main(int argc, const char** argv)
{
        Mat Dog = imread("D:/corgi.jpg");
        namedWindow("Corgi");

        createTrackbar("Corgi", "Corgi", &blurAmount, 30, onChange, &Dog);
        onChange(blurAmount, &Dog);
        setMouseCallback("Corgi", onMouse, &Dog);

        waitKey(0);
        destroyWindow("Corgi");
        return 0;
}
```

OpenCV GUI (7/8)

➢ Track bar event demo

- static void onChange(int pos, void* userInput);

```
static void onChange(int pos, void* userInput)
{
        if (pos <= 0)
        return;
        Mat imgBlur;
        Mat * img = (Mat*)userInput;  // Get the input image data's pointer.
        blur(*img, imgBlur, Size(pos, pos)); // Size() = Blur level
        imshow("Corgi", imgBlur);
}
```

OpenCV GUI (8/8)

➢ Mouse event demo

- static void onMouse(int event, int x, int y, int, void* userInput);

```
static void onMouse(int event, int x, int y, int, void* userInput)
{
        if (event != EVENT_LBUTTONDOWN)
                return;

        Mat * img = (Mat*)userInput;

        circle(*img, Point(x, y), 30, Scalar(0, 255, 255), -5);
        onChange(blurAmount, img);
}
```

Demo#2

Practice

GUI functions

Thanks!

Any questions?