OpenCV Tutorial 3 - Chapter 4

Author: Noah Kuntz (2009)
Contact: nk752@drexel.edu

Keywords: OpenCV, computer vision, highgui, interface, gui

My Vision Tutorials Index

This tutorial assumes the reader:
(1) Has a basic knowledge of Visual C++
(2) Has some familiarity with computer vision concepts
(3) Has read the previous tutorials in this series

The rest of the tutorial is presented as follows:

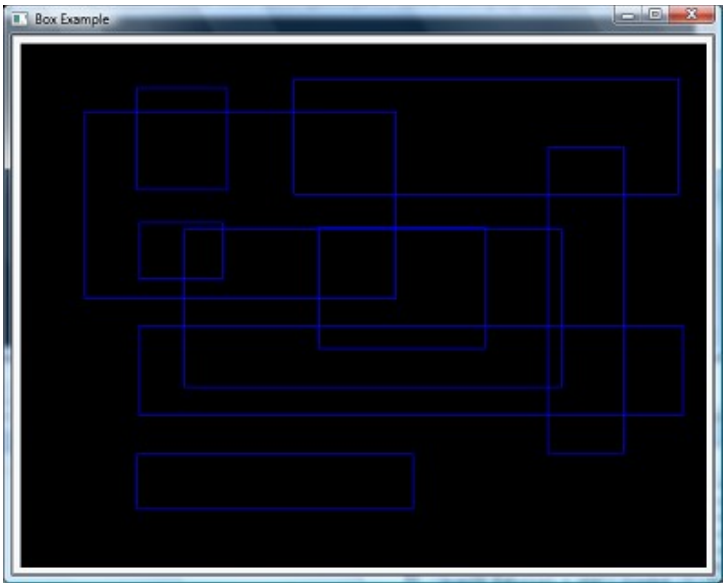
Important Note!

More information on the topics of these tutorials can be found in this book: Learning OpenCV: Computer Vision with the OpenCV Library

Step 1: HighGUI Overview

Chapter 4 covers the HighGUI library. Some of this material is repeated from the initial tutorial, including the basics of displaying images, and reading and writing video. These topics are covered in greater detail in the text. For this tutorial two examples are presented for using mouse events and sliders, both for creating interfaces.

Step 2: Using Mouse Events



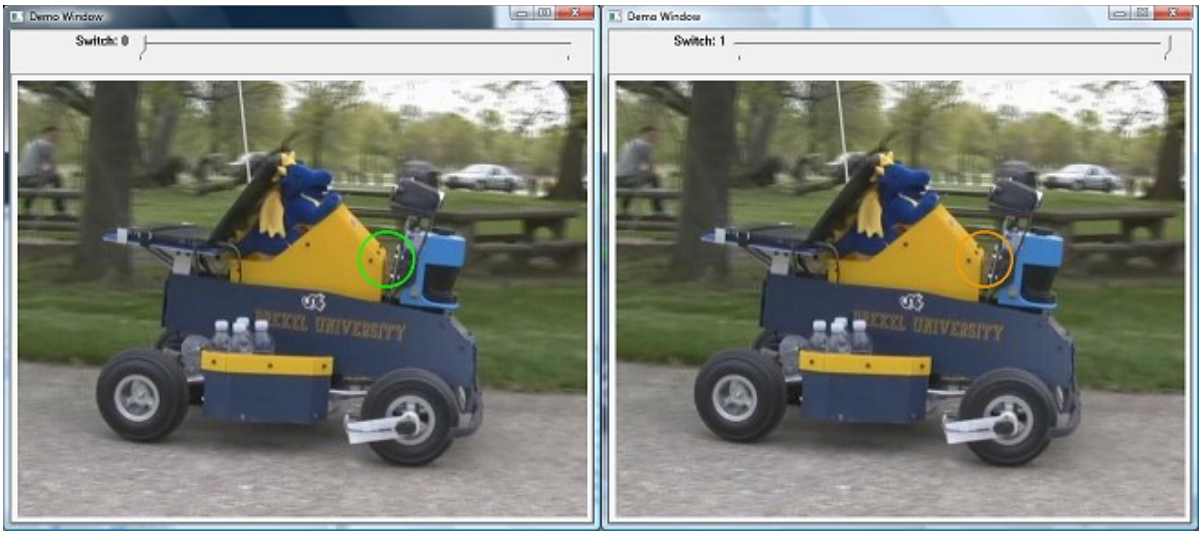
Squares drawn with the mouse

One simple graphical interface is using mouse clicks to draw. This program creates a rectangle when you click the mouse on the window, and stretches the rectangle until you release the mouse. This requires the creation of a mouse callback. For the main function we use a while loop that keeps redrawing each box as its created. Boxes are drawn and windows are created as learned in previous tutorials. The main new thing is the creation of the mouse callback. This is initialized with *cvSetMouseCallback*. The callback itself is an arbitrarily named function of the form *my_mouse_callback(int event, int x, int y, an int flags, void* param)* where x and y are the mouse position and event is a code representing what mouse action occured. The callback then uses a switch statement to perform the actions needed for this program when the mouse is moved, clicked, or released as specified by the event. Here is the code:

```
void my_mouse_callback( int event, int x, int y, int flags, void* param );

CvRect box;
bool drawing_box = false;

void draw_box( IplImage* img, CvRect rect ){
        cvRectangle( img, cvPoint(box.x, box.y), cvPoint(box.x+box.width,box.y+box.height),
                                cvScalar(0xff,0x00,0x00) );
}

// Implement mouse callback
void my_mouse_callback( int event, int x, int y, int flags, void* param ){
        IplImage* image = (IplImage*) param;

        switch( event ){
                case CV_EVENT_MOUSEMOVE:
                        if( drawing_box ){
                                box.width = x-box.x;
                                box.height = y-box.y;
                        }
                        break;

                case CV_EVENT_LBUTTONDOWN:
                        drawing_box = true;
                        box = cvRect( x, y, 0, 0 );
```

```
                        break;

                case CV_EVENT_LBUTTONUP:
                        drawing_box = false;
                        if( box.width < 0 ){
                                box.x += box.width;
                                box.width *= -1;
                        }
                        if( box.height < 0 ){
                                box.y += box.height;
                                box.height *= -1;
                        }
                        draw_box( image, box );
                        break;
        }
}

int _tmain(int argc, _TCHAR* argv[])
{
        const char* name = "Box Example";
        box = cvRect(-1,-1,0,0);

        IplImage* image = cvLoadImage( "MGC.jpg" );
        cvZero( image );
        IplImage* temp = cvCloneImage( image );

        cvNamedWindow( name );

        // Set up the callback
        cvSetMouseCallback( name, my_mouse_callback, (void*) image);

        // Main loop
        while( 1 ){
                cvCopyImage( image, temp );
                if( drawing_box )
                        draw_box( temp, box );
                cvShowImage( name, temp );

                if( cvWaitKey( 15 )==27 )
                        break;
        }

        cvReleaseImage( &image );
        cvReleaseImage( &temp );
        cvDestroyWindow( name );

        return 0;
}
```

Step 3: Using Sliders for Buttons



Using a switch to change the color of a circle

Another basic interface element is creating buttons. The *cvCreateTrackbar* function can be used to create a basic interface in the form of a slider with as many possible positions as is required. With just two positions it works like a basic flip switch. In this example a callback is created similiar to the use of the mouse, but somewhat simpler. A loop is used to change the color of a drawn circle each time the switch is toggled. Here is the code:

```
int g_switch_value = 0;
int colorInt = 0;

// Trackbar/switch callback
void switch_callback( int position ){
        if( position == 0 ){
                colorInt = 0;
        }else{
                colorInt = 1;
        }
}

int _tmain(int argc, _TCHAR* argv[])
{
        const char* name = "Demo Window";
        int radius = 30;
        int thickness = 2;
        int connectivity = 8;
        CvScalar green = CV_RGB(0,250,0);
        CvScalar orange = CV_RGB(250,150,0);

        IplImage* src1 = cvLoadImage( "MGC.jpg" );
        CvPoint pt2 = cvPoint(405,195);

        cvNamedWindow( name, 1 );
        cvShowImage(name, src1);


        // Create trackbar
        cvCreateTrackbar( "Switch", name, &g_switch_value, 1, switch_callback );

        // Loop to update the circle color
        while( 1 ) {
```

```
                if( colorInt == 0)
                        cvCircle(src1,pt2,radius,green,thickness,connectivity);
                else
                        cvCircle(src1,pt2,radius,orange,thickness,connectivity);
                cvShowImage(name, src1);
                if( cvWaitKey( 15 ) == 27 )
                        break;
        }

        cvReleaseImage( &src1 );
        cvDestroyWindow( name );

        return 0;
}
```

Final Words

This tutorial's objective was to show how to use some additional HighGUI functions. You should be able to extend these functions to create basic interfaces for OpenCV programs.

Click here to email me.
Click here to return to my Tutorials page.