OpenCV Tutorial 2 - Chapter 3

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

Keywords: OpenCV, computer vision, data type, alpha

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: Data Type Concepts

Chapter 3 is largely concerned with some key data types in open cv, for matrices and specifically images, and the various basic functions for manipulating them. I suggest that you read the chapter to full understand these topics, as I cannot repeat all of the information from the book. I will briefly touch on some of the basics in this section before going on to a couple examples.

Some Basic Data Types:
CvPoint - point in an image (x,y)
CvSize - size of an image (width, height)
CvRect - portion of an image (x, y, width, height)
CvScaler - RBGA value for a pixel (val[4])
cvMat* - matrix (rows, cols, type) for a 2 dimensional matrix

Creating a Matrix
The most basic way to make a matrix is: *CvMat* mat = cvCreateMat(5, 5, CV32FC1);* And it can be accessed with: *float element_3_2 = CV_MAT_ELEM(*mat, float, 3, 2);* There are better more elegant ways to access matrix values, in particular via pointer incrementation. This code would sum all the elements in a three-channel matrix:

```
float sum( const CvMat* mat ){
        float s = 0.0f;
        for(int row=0; rowrows; row++ ){
```

```
                const float* ptr = (const float*)(mat->data.ptr + row * mat->step);
                for( col=0; colcols; col++){
                        s += *ptr++;
                }
        }
}
```

The other important thing to note from this chapter is the IplImage Data Structure:

```
typedef struct _IplImage {
        int                     nSize;
        int                     ID;
        int                     nChannels;
        int                     alphaChannel;
        int                     depth;
        char                    colorModel[4];
        char                    channelSeq[4];
        int                     dataOrder;
        int                     origin;
        int                     align;
        int                     width;
        int                     height;
        struct _IplROI*         roi;
        struct _IplImage*       maskROI;
        void*                   imageId;
        struct _IplTileInfo*    tileInfo;
        int                     imageSize;
        char*                   imageData;
        int                     widthStep;
        int                     BorderMode[4];
        int                     BorderConst[3];
        char*                   imageDataOrigin;
} IplImage;
```

Some important factors here are obviously the width and height, then depth in terms of colors, IPL_DEPTH_8U being the most common, and nChannels for whether the image is grayscale (1), RGB (3), or RGBA (4).

Step 2: Alpha Blend with ROI

Part of an image alpha blended with OpenCV

One simple operation to perform on an image is an alpha blend. To perform this on just part of an image can be achieved by setting a region of interest *cvSetImageROI* and then performing a blend with *cvAddWeighted*. Check the chapter for many other matrix operations.

```
int _tmain(int argc, _TCHAR* argv[])
{
        IplImage* src1 = cvLoadImage( "MGC.jpg" );
        IplImage* src2 = cvLoadImage( "wheel.jpg" );
        int x = 280;
        int y = 80;
        int width = 60;
        int height = 60;
        double alpha = 0.5;
        double beta = 0.5;
        cvSetImageROI(src1, cvRect(x,y,width,height));
        cvAddWeighted(src1, alpha, src2, beta, 0.0, src1);
        cvResetImageROI(src1);
        cvNamedWindow("Alpha_blend", 1);
        cvShowImage("Alpha_blend", src1);
        cvWaitKey();
}
```

Step 3: Drawing and Text

Lines and text drawn on an image

Another basic image manipulation is adding lines, shapes, and text. In this example a line is drawn, a circle is drawn, and text is added to the image. Points must be created with *cvPoint* to do any of these actions, and scalers must be created to represent colors, by using *CV_RGB(r,g,b)*. We are able to draw a line with *cvLine (src1,pt1,pt2,color,thickness,connectivity)*, and a circle with *cvCircle (src1,pt2,radius,blue,thickness,connectivity)* Text is a little more complex, first a font must be initialized with *CvFont font1; cvInitFont(&font1,CV_FONT_HERSHEY_DUPLEX,hscale,vscale,shear,thickness,line_type)*, and then the text can be created with *cvPutText(src1,text,pt1,&font1,blue)*. Here is the code:

```
int _tmain(int argc, _TCHAR* argv[])
{
        IplImage* src1 = cvLoadImage( "MGC.jpg" );

        // Line variables
        CvPoint pt1 = cvPoint(250,60);
        CvPoint pt2 = cvPoint(405,195);
        CvScalar red = CV_RGB(250,0,0);
        int thickness = 2;
        int connectivity = 8;

        // Circle variables
        int radius = 30;
        CvScalar blue = CV_RGB(0,0,250);

        // Text variables
        const char* text = "testing";
        double hscale = 1.0;
        double vscale = 0.8;
        double shear = 0.2;
        int thickness2 = 1;
        int line_type = 8;

        CvFont font1;
        cvInitFont(&font1,CV_FONT_HERSHEY_DUPLEX,hscale,vscale,shear,thickness,line_type);

        cvLine(src1,pt1,pt2,red,thickness,connectivity);
```

```
        cvCircle(src1,pt2,radius,blue,thickness,connectivity);
        cvPutText(src1,text,pt1,&font1,blue);

        cvNamedWindow("Drawing_and_Text", 1);
        cvShowImage("Drawing_and_Text", src1);
        cvWaitKey();

        return 0;
}
```

Final Words

This tutorial's objective was to show how to do some additional image manipulation. The book should be followed to learn more about the new data types introduced.

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