OpenCV Tutorial 6 - Chapter 7

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

Keywords: OpenCV, computer vision, image processing, histograms

My Vision Tutorials Index

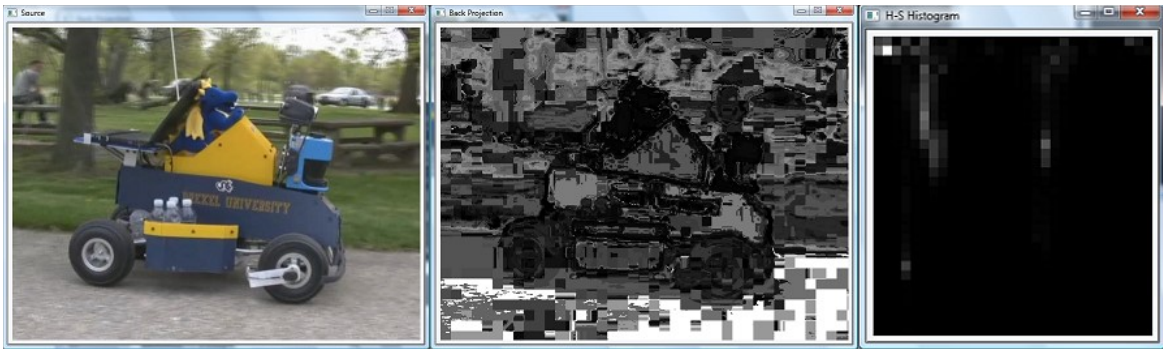
This tutorial assumes the reader:
(1) Has a basic knowledge of Visual C++
(2) Has some familiarity with computer vision concepts
(3) Has read the previous tutorials in this series

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: Histograms and Back Projection



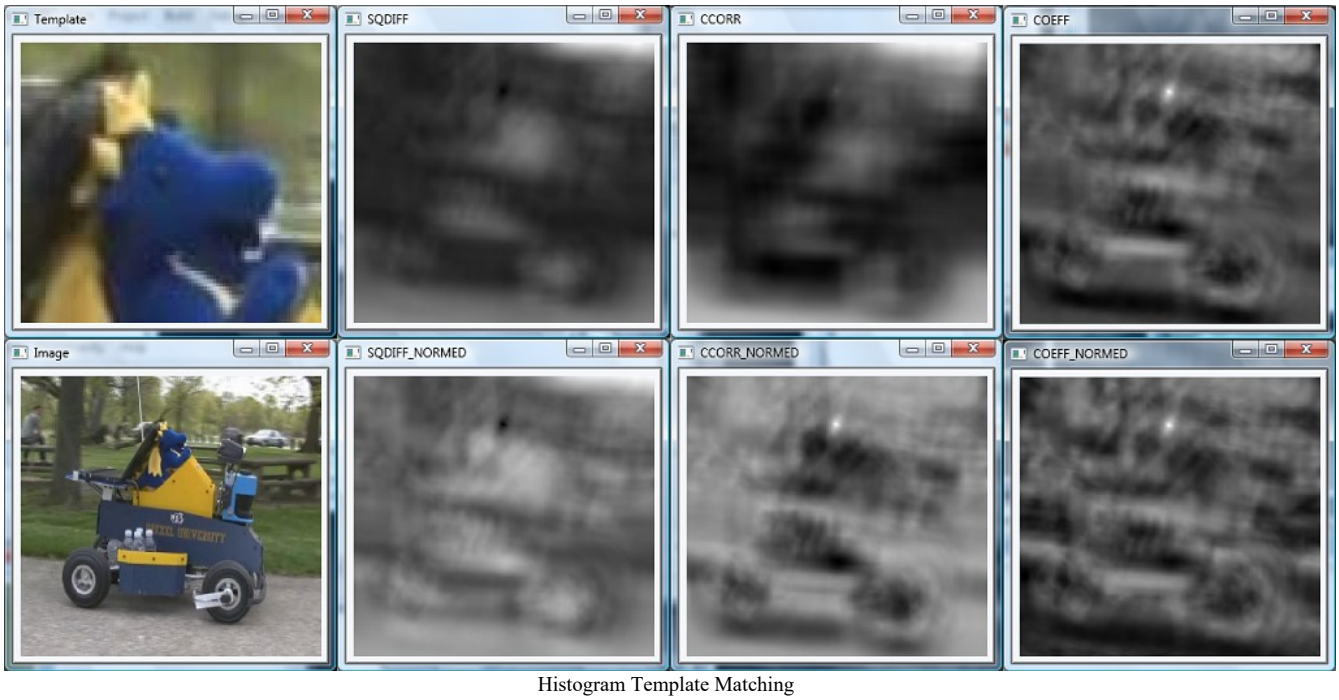
Original Image, Back Projection, and Histogram

Histograms are one of the classic methods to analyze images. A Histogram classifies aspects of an image into bins to determine the correlation between images, or a feature in an image. OpenCV provides the *CvHistogram* structure, created with *cvCreateHist(int dims, int* sizes, int type, float** ranges = NULL, int uniform = 1)* Consult the book for more details of the histogram data structure. Histograms bins are accessed in a similiar way to normal matrices, for example with *cvGetHistValue_2D(CvHistogram* hist, int idx0, idx1)*. The following example computes a histogram for the intensity of an image based on the HSV colorspace. The histogram is also normalized, and *cvCalcBackProject* is used to map the histogram results back on to the image to better illustrate the results. Here is the code:

```
int _tmain(int argc, _TCHAR* argv[])
{
	// Set up images
	IplImage* img = cvLoadImage("MGC.jpg");
	IplImage* back_img = cvCreateImage( cvGetSize( img ), IPL_DEPTH_8U, 1 );

	// Compute HSV image and separate into colors
	IplImage* hsv = cvCreateImage( cvGetSize(img), IPL_DEPTH_8U, 3 );
	cvCvtColor( img, hsv, CV_BGR2HSV );

	IplImage* h_plane = cvCreateImage( cvGetSize( img ), 8, 1 );
	IplImage* s_plane = cvCreateImage( cvGetSize( img ), 8, 1 );
	IplImage* v_plane = cvCreateImage( cvGetSize( img ), 8, 1 );
	IplImage* planes[] = { h_plane, s_plane };
	cvCvtPixToPlane( hsv, h_plane, s_plane, v_plane, 0 );

	// Build and fill the histogram
	int h_bins = 30, s_bins = 32;
	CvHistogram* hist;
	{
		int hist_size[] = { h_bins, s_bins };
		float h_ranges[] = { 0, 180 };
		float s_ranges[] = { 0, 255 };
		float* ranges[] = { h_ranges, s_ranges };
```

```
            hist = cvCreateHist( 2, hist_size, CV_HIST_ARRAY, ranges, 1 );
        }
        cvCalcHist( planes, hist, 0, 0 ); // Compute histogram
        cvNormalizeHist( hist, 20*255 ); // Normalize it

        cvCalcBackProject( planes, back_img, hist );// Calculate back projection
        cvNormalizeHist( hist, 1.0 ); // Normalize it



        // Create an image to visualize the histogram
        int scale = 10;
        IplImage* hist_img = cvCreateImage( cvSize( h_bins * scale, s_bins * scale ), 8, 3 );
        cvZero ( hist_img );

        // populate the visualization
        float max_value = 0;
        cvGetMinMaxHistValue( hist, 0, &max_value, 0, 0 );

        for( int h = 0; h < h_bins; h++ ){
                for( int s = 0; s < s_bins; s++ ){
                        float bin_val = cvQueryHistValue_2D( hist, h, s );
                        int intensity = cvRound( bin_val * 255 / max_value );
                        cvRectangle( hist_img, cvPoint( h*scale, s*scale ),
                                               cvPoint( (h+1)*scale - 1, (s+1)*scale - 1 ),
                                               CV_RGB( intensity, intensity, intensity ),
                                               CV_FILLED );
                }
        }

        // Show original
        cvNamedWindow( "Source", 1) ;
        cvShowImage( "Source", img );

        // Show back projection
        cvNamedWindow( "Back Projection", 1) ;
        cvShowImage( "Back Projection", back_img );

        // Show histogram equalized
        cvNamedWindow( "H-S Histogram", 1) ;
        cvShowImage( "H-S Histogram", hist_img );

        cvWaitKey(0);

        cvReleaseImage( &img );
        cvReleaseImage( &back_img );
        cvReleaseImage( &hist_img );

        return 0;
}
```

Step 2: Template Matching



Histogram Template Matching

Another powerful operation with a histogram is template matching. Here a template image is compared with a larger image to find the area most similiar to the template in the larger image. The function *cvMatchTemplate(const CvArr* image, const CvArr* templ, CvArr* result, int method)* is used to do the matching. The last input here chooses the method of template matching. We use all six methods, starting with square difference matching, which simply takes the squared difference between the template and image. A match is 0 and bad matches are large values. Then there is correlation matching which multiplies the template against the image, so a large value is a good match and a very bad match s 0. The last general type of matchin is correlation coefficient matching, which matches the mean of the image relative to the mean of template, with a good match being 1, no match being 0, and a mismatch being as low as -1. The normalized versions of these are also tested, and generally provide more visually clear results. Here is the code:

```
int _tmain(int argc, _TCHAR* argv[])
{
        IplImage *src, *templ, *ftmp[6]; // ftmp will hold results
        int i;

        // Read in the source image to be searched
        src = cvLoadImage("MGC.jpg");

        // Read in the template to be used for matching:
        templ = cvLoadImage("template.jpg");

        // Allocate Output Images:
        int iwidth = src->width - templ->width + 1;
        int iheight = src->height - templ->height + 1;
        for(i = 0; i < 6; ++i){
                ftmp[i]= cvCreateImage( cvSize( iwidth, iheight ), 32, 1 );
        }

        // Do the matching of the template with the image
        for( i = 0; i < 6; ++i ){
                cvMatchTemplate( src, templ, ftmp[i], i );
                cvNormalize( ftmp[i], ftmp[i], 1, 0, CV_MINMAX );
        }

        // DISPLAY
        cvNamedWindow( "Template", 0 );
        cvShowImage( "Template", templ );
        cvNamedWindow( "Image", 0 );
        cvShowImage( "Image", src );
        cvNamedWindow( "SQDIFF", 0 );
        cvShowImage( "SQDIFF", ftmp[0] );
        cvNamedWindow( "SQDIFF_NORMED", 0 );
        cvShowImage( "SQDIFF_NORMED", ftmp[1] );
        cvNamedWindow( "CCORR", 0 );
        cvShowImage( "CCORR", ftmp[2] );
        cvNamedWindow( "CCORR_NORMED", 0 );
        cvShowImage( "CCORR_NORMED", ftmp[3] );
        cvNamedWindow( "COEFF", 0 );
        cvShowImage( "COEFF", ftmp[4] );
        cvNamedWindow( "COEFF_NORMED", 0 );
        cvShowImage( "COEFF_NORMED", ftmp[5] );

        cvWaitKey(0);

        return 0;
}
```

Final Words

This tutorial's objective was to show how to use some basic histogram functions.

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