OpenCV Tutorial 7 - Chapter 8

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

Keywords: OpenCV, computer vision, image processing, contours

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:

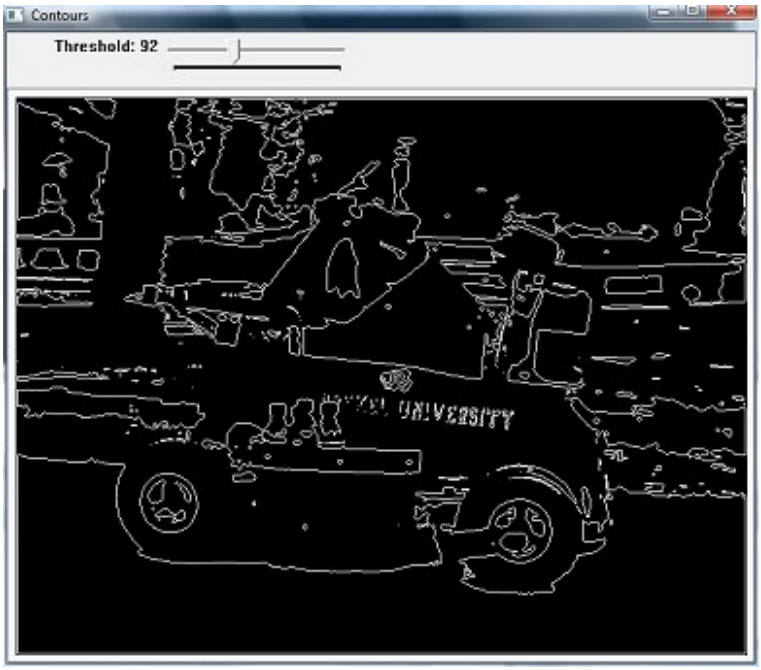
Important Note!

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

Step 1: Sequences

In order to work with contours, several other OpenCV functions and data types must be considered. OpenCV uses an entity called memory storage for dynamic objects. One type of memory storage is a sequence, which is a linked list of objects. A sequence is created like this: *CvSeq* yourvariable = cvCreateSeq(int seq_flags, int header_size, int elem_size, CvMemStorage* storage*. Individual elements in a sequence can be accessed with *char* cvGetSeqElem(seq, index)*. A sequence can be used as a stack with functions like push, *char* cvSeqPush(CvSeq* seq, void* element = NULL)* and pop, *char* cvSeqPop(CvSeq* seq, void* element = NULL)*. Sequence objects can also be converted to arrays. Sequences are used for storing contours. For more in depth details about the flags for creating sequences and various methods of accessing and manipulating sequence elements, please see the book.

Step 2: Contour Examples

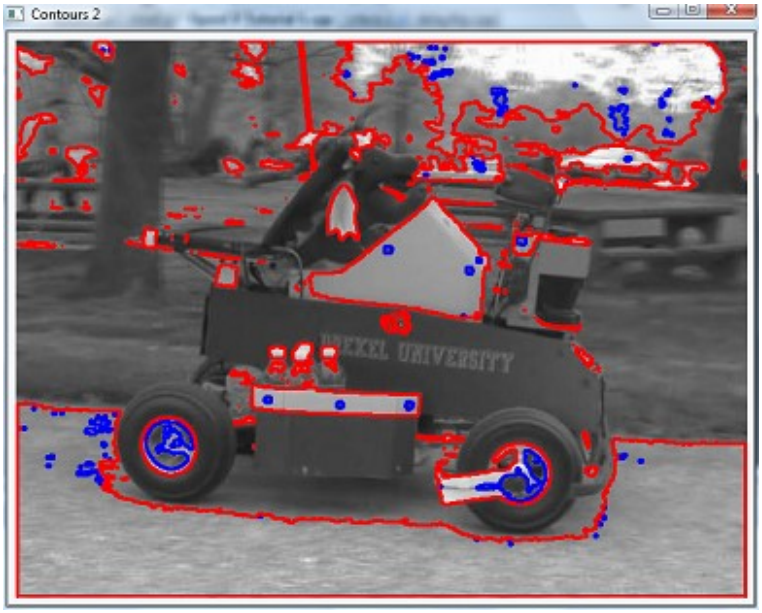


Contours at Various Thresholds

Contours are sequences of points defining a line/curve in an image. Contour matching can be used to classify image objects. The following is a basic example for finding contours at various thresholds. *cvThreshold* is used to binarize the image, and then the contours are found with *int cvFindContours(CvArr* image, CvMemStorage* storage, CvSeq** first_contour, int headersize = sizeof(CvContour), int mode = CV_RETR_LIST, int method = CV_CHAIN_APPROX_SIMPLE, CvPoint offset = cvPoint(0,0))*. Then the contours are drawn using *void cvDrawContours(CvArr* img, CvSeq* contour, CvScalar external_color, CvScalar hole_color, int max_level, int thickness = 1, int line_type = 8, CvPoint offset = cvPoint(0,0))*. A trackbar is used to control the threshold. Here is the code:

```
IplImage*       g_image = NULL;
IplImage*       g_gray = NULL;
int             g_thresh = 100;
CvMemStorage*   g_storage = NULL;

void on_trackbar(int){
        if( g_storage == NULL ){
                g_gray = cvCreateImage( cvGetSize( g_image ), 8, 1 );
                g_storage = cvCreateMemStorage(0);
        } else {
                cvClearMemStorage( g_storage );
        }

        CvSeq* contours = 0;
        cvCvtColor( g_image, g_gray, CV_BGR2GRAY );
        cvThreshold( g_gray, g_gray, g_thresh, 255, CV_THRESH_BINARY );
        cvFindContours( g_gray, g_storage, &contours );
        cvZero( g_gray );
        if( contours ){
                cvDrawContours(
                        g_gray,
```

```
                        contours,
                        cvScalarAll(255),
                        cvScalarAll(255),
                        100 );
        }
        cvShowImage( "Contours", g_gray );
}


int _tmain(int argc, _TCHAR* argv[])
{

        g_image = cvLoadImage( "MGC.jpg" );
        cvNamedWindow( "Contours", 1 );
        cvCreateTrackbar( "Threshold", "Contours", &g_thresh, 255, on_trackbar );
        on_trackbar(0);
        cvWaitKey();

        return 0;
}
```



Finding and Drawing Contours on an Input Image

This example draws external and internal contours with different colors, and draws them one at a time so that by pressing escape you can cycle through the contours. Here is the code:

```
int _tmain(int argc, _TCHAR* argv[])
{
        cvNamedWindow( "Contours 2", 1 );
        IplImage* img_8uc1 = cvLoadImage( "MGC.jpg", 0 );
        IplImage* img_edge = cvCreateImage( cvGetSize(img_8uc1), 8, 1 );
        IplImage* img_8uc3 = cvCreateImage( cvGetSize(img_8uc1), 8, 3 );

        cvThreshold( img_8uc1, img_edge, 128, 255, CV_THRESH_BINARY );
```

```
        CvMemStorage* storage = cvCreateMemStorage();
        CvSeq* first_contour = NULL;

        int Nc = cvFindContours(
                img_edge,
                storage,
                &first_contour,
                sizeof(CvContour),
                CV_RETR_LIST );

        int n=0;
        printf( "Total Contours Detected: %d\n", Nc );
        CvScalar red = CV_RGB(250,0,0);
        CvScalar blue = CV_RGB(0,0,250);

        for( CvSeq* c=first_contour; c!=NULL; c=c->h_next ){
                cvCvtColor( img_8uc1, img_8uc3, CV_GRAY2BGR );
                cvDrawContours(
                        img_8uc3,
                        c,
                        red,            // Red
                        blue,           // Blue
                        1,                      // Vary max_level and compare results
                        2,
                        8 );
                printf( "Contour #%dn", n );
                cvShowImage( "Contours 2", img_8uc3 );
                printf( " %d elements:\n", c->total );
                for( int i=0; itotal; ++i ){
                        CvPoint* p = CV_GET_SEQ_ELEM( CvPoint, c, i );
                        printf(" (%d,%d)\n", p->x, p->y );
                }
                cvWaitKey();
                n++;
        }

        printf( "Finished all contours.\n");
        cvCvtColor( img_8uc1, img_8uc3, CV_GRAY2BGR );
        cvShowImage( "Contours 2", img_8uc3 );
        cvWaitKey();

        cvDestroyWindow( "Contours 2" );

        cvReleaseImage( &img_8uc1 );
        cvReleaseImage( &img_8uc3 );
        cvReleaseImage( &img_edge );

        return 0;
}
```

Step 3: Other Contour Functions

I want to touch on more processing that can be done with contours, please see the text to understand how to implement these functions. A contour can be approximated as a polygon with *cvApproxPoly()*. The dominant points can be found from a contour, a dominant point being on that is on a corner essentially,

using *cvFindDominantPoints()*. Other characteristic like length and the the bounding box can be measured. Contours can be matched using moments, Hu moments, hierarchical matching, convexity characteristics, and geometrical histograms.

Final Words

This tutorial's objective was to show how to use sequences, and find contours in images, and assorted functions related to contour matching.

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