A Basic Introduction to OpenCV for Image Processing

Qing Chen (David)

E-mail: qchen@discover.uottawa.ca

DiscoverLab
School of Information Technology & Engineering
University of Ottawa

January 15, 2007

Outline

- 1. Introduction
- 2. Image data structure in OpenCV
- 3. Basic operations for images
- 4. Working with videos
- 5. References and resources

1. Introduction

- General description
  - Open source computer vision library in C/C++.
  - Optimized and intended for real-time applications.
  - OS/hardware/window-manager independent.
  - Generic image/video loading, saving, and acquisition.
  - Both low and high level API.
  - Provides interface to Intel's Integrated Performance Primitives (IPP) with processor specific optimization (Intel processors).

1. Introduction

- Features:
  - Image data manipulation (allocation, release, copying, setting, conversion).
  - Image and video I/O (file and camera based input, image/video file output).
  - Matrix and vector manipulation and linear algebra routines.
  - Various dynamic data structures (lists, queues, sets, trees, graphs).
  - Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
  - Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
  - Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence).
  - Motion analysis (optical flow, motion segmentation, tracking).
  - Object recognition (eigen-methods, HMM).
  - Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
  - Image labeling (line, conic, polygon, text drawing).

1. Introduction

- OpenCV modules:
  - *cv* - Main OpenCV functions.
  - *cvaux* - Auxiliary (experimental) OpenCV functions.
  - *cxcore* - Data structures and linear algebra support.
  - *highgui* - GUI functions.

2. Image data structure in OpenCV

- Load and display an image in OpenCV:

```
#include "cv.h" //main OpenCV functions
#include "highgui.h" //OpenCV GUI functions¯include <stdio.h>

int main()
{
        /* declare a new IplImage pointer, the basic
           image data structure in OpenCV */
        IplImage* newImg;
        /* load an image named "apple.bmp", 1 means
           this is a color image */
        newImg = cvLoadImage("apple.bmp",1);
        //create a new window
        cvNamedWindow("Window", 1);
        //display the image in the window
        cvShowImage("Window", newImg);
        //wait for key to close the window
        cvWaitKey(0);
        cvDestroyWindow( "Window" ); //destroy the window
        cvReleaseImage( &newImg ); //release the memory for the image
        return 0;
}
```

2. Image data structure in OpenCV

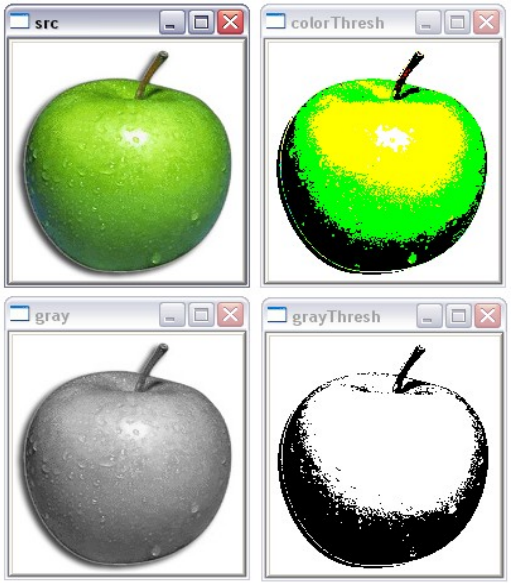
- IplImage is the basic image data structure in OpenCV

- IPL image:

```
IplImage
  |-- int  nChannels;     // Number of color channels (1,2,3,4)
  |-- int  depth;         // Pixel depth in bits:
  |                       //   IPL_DEPTH_8U, IPL_DEPTH_8S,
  |                       //   IPL_DEPTH_16U,IPL_DEPTH_16S,
  |                       //   IPL_DEPTH_32S,IPL_DEPTH_32F,
  |                       //   IPL_DEPTH_64F
  |-- int  width;         // image width in pixels
  |-- int  height;        // image height in pixels
  |-- char* imageData;    // pointer to aligned image data
  |                       // Note that color images are stored in BGR order
  |-- int  dataOrder;     // 0 - interleaved color channels,
  |                       // 1 - separate color channels
  |                       // cvCreateImage can only create interleaved images
  |-- int  origin;        // 0 - top-left origin,
  |                       // 1 - bottom-left origin (Windows bitmaps style)
  |-- int  widthStep;     // size of aligned image row in bytes
  |-- int  imageSize;     // image data size in bytes = height*widthStep
  |-- struct _IplROI *roi;// image ROI. when not NULL specifies image
  |                       // region  to be processed.
  |-- char *imageDataOrigin; // pointer to the unaligned origin of image data
  |                          // (needed for correct image deallocation)
  |
  |-- int  align;         // Alignment of image rows: 4 or 8 byte alignment
  |                       // OpenCV ignores this and uses widthStep instead
  |-- char colorModel[4]; // Color model - ignored by OpenCV
```

3. Basic operations for images

- Threshold




```
#include "cv.h"
#include "highgui.h"
#include "math.h"

int main()
{
    IplImage* src;
    IplImage* colorThresh;
    IplImage* gray;
    IplImage* grayThresh;
    int threshold = 120, maxValue = 255;
    int thresholdType = CV_THRESH_BINARY;

    src = cvLoadImage("apple.bmp", 1);
    colorThresh = cvCloneImage( src );
    gray = cvCreateImage( cvSize(src->width, src->height), IPL_DEPTH_8U, 1 );
    cvCvtColor( src, gray, CV_BGR2GRAY );
    grayThresh = cvCloneImage( gray );

    cvNamedWindow( "src", 1 );    cvShowImage( "src", src );
    cvNamedWindow( "gray", 1 );    cvShowImage( "gray", gray );

    cvThreshold(src, colorThresh, threshold, maxValue, thresholdType);
    cvThreshold(gray, grayThresh, threshold, maxValue, thresholdType);

    cvNamedWindow( "colorThresh", 1 );    cvShowImage( "colorThresh", colorThresh );

    cvNamedWindow( "grayThresh", 1 );    cvShowImage( "grayThresh", grayThresh );

    cvWaitKey(0);

    cvDestroyWindow( "src" );
    cvDestroyWindow( "colorThresh" );
    cvDestroyWindow( "gray" );
    cvDestroyWindow( "grayThresh" );
    cvReleaseImage( &src );
    cvReleaseImage( &colorThresh );
    cvReleaseImage( &gray );
    cvReleaseImage( &grayThresh );

    return 0;
}
```

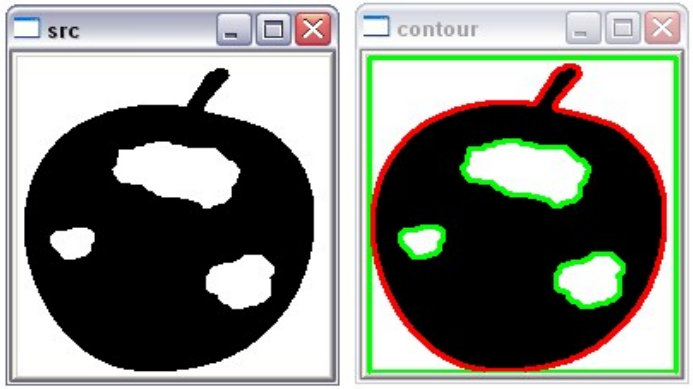
3. Basic operations for images

- Canny edge detection

```
#include "cv.h"
#include "highgui.h"

int main()
{
   IplImage* newImg; // original image
   IplImage* grayImg; // gray image for the conversion of the original image
   IplImage* cannyImg; // gray image for the canny edge detection

    //load original image
    newImg = cvLoadImage("apple.bmp",1);
    //create a single channel 1 byte image (i.e. gray-level image)
    grayImg = cvCreateImage( cvSize(newImg->width, newImg->height), IPL_DEPTH_8U, 1 );
    //convert original color image (3 channel rgb color image) to gray-level image
    cvCvtColor( newImg, grayImg, CV_BGR2GRAY );
    cannyImg = cvCreateImage(cvGetSize(newImg), IPL_DEPTH_8U, 1);

    // canny edge detection
    cvCanny(grayImg, cannyImg, 50, 150, 3);

    cvNamedWindow("src", 1);
    cvNamedWindow("canny",1);
    cvShowImage( "src", newImg );
    cvShowImage( "canny", cannyImg );

    cvWaitKey(0);

    cvDestroyWindow( "src" );
    cvDestroyWindow( "canny" );
    cvReleaseImage( &newImg );
    cvReleaseImage( &grayImg );
    cvReleaseImage( &cannyImg );

    return 0;
}
```

3. Basic operations for images

- Contour detection

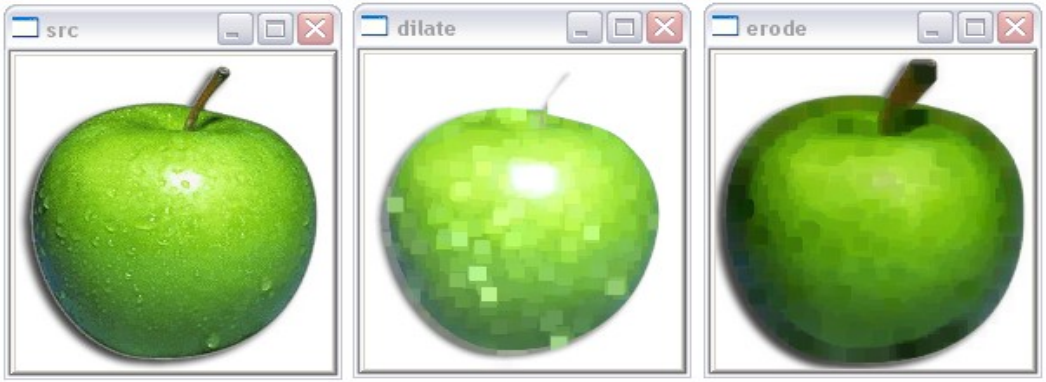



```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"

int main()
{
  IplImage* newImg = NULL;
  IplImage* grayImg = NULL;
  IplImage* contourImg = NULL;

  //parameters for the contour detection
  CvMemStorage * storage = cvCreateMemStorage(0);
  CvSeq * contour = 0;
  int mode = CV_RETR_EXTERNAL;
  mode = CV_RETR_CCOMP; //detect both outside and inside contour

  cvNamedWindow("src", 1);
  cvNamedWindow("contour",1);
  //load original image
  newImg = cvLoadImage("applebw.bmp",1);
  //create a single channel 1 byte image (i.e. gray-level image)
  grayImg = cvCreateImage( cvSize(newImg->width, newImg->height), IPL_DEPTH_8U, 1 );
  //convert original color image (3 channel rgb color image) to gray-level image
  cvCvtColor( newImg, grayImg, CV_BGR2GRAY );
  cvShowImage( "src", newImg );
  //make a copy of the original image to draw the detected contour
  contourImg = cvCreateImage(cvGetSize(newImg), IPL_DEPTH_8U, 3);
  contourImg=cvCloneImage( newImg );

  //find the contour
  cvFindContours(grayImg, storage, &contour, sizeof(CvContour), mode, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));
  //draw the contour
  cvDrawContours(contourImg, contour, CV_RGB(0, 255, 0), CV_RGB(255, 0, 0), 2, 2, 8);
  cvShowImage( "contour", contourImg );
  cvWaitKey(0);
  cvDestroyWindow( "src" );  cvDestroyWindow( "contour" );
  cvReleaseImage( &newImg );   cvReleaseImage( &grayImg );  cvReleaseImage( &contourImg );
  cvReleaseMemStorage(&storage);
  return 0;
}
```

3. Basic operations for images

- Dilate/Erode

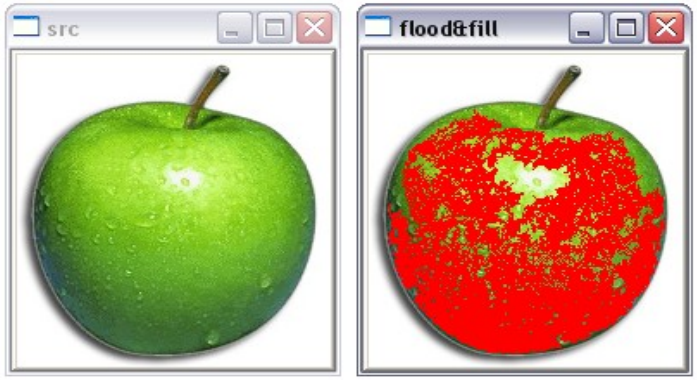
```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"

int main()
{

  IplImage* newImg = NULL;
  IplImage* dilateImg = NULL;
  IplImage* erodeImg = NULL;

  cvNamedWindow("src", 1);
  cvNamedWindow("dilate",1);
  cvNamedWindow("erode",1);

  //load original image
  newImg = cvLoadImage("apple.bmp",1);
  cvShowImage( "src", newImg );

  //make a copy of the original image
  dilateImg=cvCloneImage( newImg );
  erodeImg=cvCloneImage( newImg );

  //dilate image
  cvDilate(newImg,dilateImg,NULL,4);
  //erode image
  cvErode(newImg,erodeImg,NULL,4);

  cvShowImage( "dilate", dilateImg );
  cvShowImage( "erode", erodeImg );

  cvWaitKey(0);

  cvDestroyWindow( "src" );  cvDestroyWindow( "dilate" );  cvDestroyWindow( "erode" );
  cvReleaseImage( &newImg );   cvReleaseImage( &dilateImg );  cvReleaseImage( &erodeImg );
  return 0;
}
```



3. Basic operations for images

- Flood and Fill

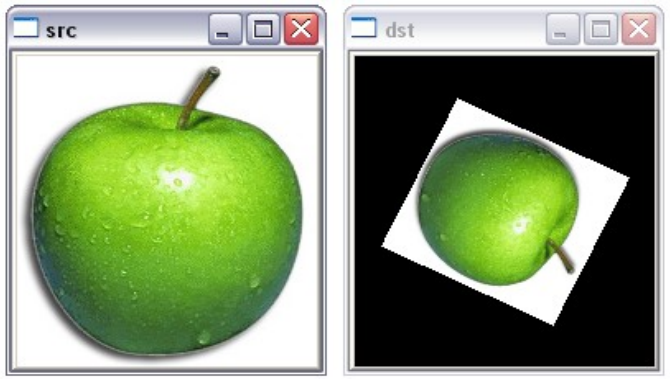
```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"

int main()
{
  IplImage* newImg = NULL;
  IplImage* ffImg = NULL;

  //flood and fill parameters
  int lo_diff, up_diff; //the low and up flood randge which can be adjusted
  CvConnectedComp comp;
  CvPoint floodSeed; //the original pixel where the flood begins
  CvScalar floodColor;
  lo_diff=8;
  up_diff=8;
  floodColor = CV_RGB( 255, 0, 0 ); //set the flood color to red

  cvNamedWindow("src", 1);
  cvNamedWindow("flood&fill",1);

  //load original image
  newImg = cvLoadImage("apple.bmp",1);
  cvShowImage( "src", newImg );
  //make a copy of the original image
  ffImg=cvCloneImage( newImg );
  floodSeed=cvPoint(60,60); //flooding start from pixel(60, 60)

  //Flood and Fill from pixel(60, 60) with color red and the flood range of (-8, +8)
  cvFloodFill( ffImg, floodSeed, floodColor, CV_RGB( lo_diff, lo_diff, lo_diff ),
                                CV_RGB( up_diff, up_diff, up_diff ), &comp, 8,  NULL);

  cvShowImage( "flood&fill", ffImg );

  cvWaitKey(0);
  cvDestroyWindow( "src" );  cvDestroyWindow( "flood&fill" );
  cvReleaseImage( &newImg );   cvReleaseImage( &ffImg );
  return 0;
}
```



3. Basic operations for images

- Rotate and Scale

```
#include "cv.h"
#include "highgui.h"
#include "math.h"

int main()
{
    IplImage* src;
    IplImage* dst;
    int delta;
    int angle;

        src = cvLoadImage("apple.bmp", 1);
        dst = cvCloneImage( src );
        delta = 1; angle = 0;
        cvNamedWindow( "src", 1 );
        cvShowImage( "src", src );
        for(;;)
        {
            float m[6];
            double factor = (cos(angle*CV_PI/180.) + 1.1)*3;
            CvMat M = cvMat( 2, 3, CV_32F, m );
            int w = src->width;
            int h = src->height;

            m[0] = (float)(factor*cos(-angle*2*CV_PI/180.));
            m[1] = (float)(factor*sin(-angle*2*CV_PI/180.));
            m[2] = w*0.5f;
            m[3] = -m[1];
            m[4] = m[0];
            m[5] = h*0.5f;

            cvGetQuadrangleSubPix( src, dst, &M, 1, cvScalarAll(0));

            cvNamedWindow( "dst", 1 ); cvShowImage( "dst", dst );

            if( cvWaitKey(5) == 27 )
                break;

            angle = (angle + delta) % 360;
        }
    return 0;
}
```



4. Working with videos

- Video capture from a file:
  - CvCapture* cvCaptureFromFile(const char* filename);
- Video capture from a camera:
  - CvCapture* cvCaptureFromCAM(int index);
  - example:

```
// capture from video device #0
CvCapture* capture = cvCaptureFromCAM(0);
```

4. Working with videos

- Grab a frame:
  - cvGrabFrame(CvCapture* capture);
- Get the image grabbed with cvGrabFrame:
  - cvRetrieveFrame(CvCapture* capture);
  - example:

```
IplImage* img = 0;

if(!cvGrabFrame(capture)){ // capture a frame
printf("Could not grab a frame\n\7");
exit(0); }

//retrieve the captured frame
img=cvRetrieveFrame(capture);
```

- Release the capture source:
  - cvReleaseCapture(&capture);
- For a better understanding of video processing with OpenCV, refer to the face detection example under the dir: C:\Program Files\OpenCV\samples\c\facedetect.c

5. References and resources

- http://www.intel.com/technology/computing/opencv/index.htm
  OpenCV official webpage.
- http://opencvlibrary.sourceforge.net/
  OpenCV documentation and FAQs.
- http://tech.groups.yahoo.com/group/OpenCV/
  OpenCV forum at Yahoo Groups.
- http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow/cvision.htm
  This is a good walkthrough for OpenCV and the Microsoft DirectShow technology by Prof. Robert Laganière of university of Ottawa. The configuration of OpenCV for MS .Net is also included.
- http://ai.stanford.edu/~dstavens/cs223b/stavens_opencv_optical_flow.pdf
  This is another OpenCV introduction focused on Optical Flow, the installation of OpenCV is also included.

About myself

- I am a Ph.D. candidate at the DiscoverLab, School of Information Technology and Engineering, University of Ottawa. My general research interests include image processing and computer vision. My current research topic is focused on real-time vision-based hand gesture recognition for Human Computer Interface (HCI).
  For more information about me, please refer to my webpage: http://www.discover.uottawa.ca/~qchen