OpenCV Tutorial

Using OpenCV with Microsoft Visual Studio .net 2005

Lecturer: Amir hossein khalili
Sharif university of technology
Nov 2007
OpenCV

**What is OpenCV?**
(from the documentation)

OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and a few C++ classes that implement some popular Image Processing and Computer Vision algorithms.

**The key features**
(from the documentation)

Cross-Platform API of C functions FREE for commercial and non-commercial uses

**What this means**

You can take advantage of high speed implementations of functions commonly used in Computer Vision/Image Processing.
Overview of OpenCV

CV
Image processing and vision algorithms

HighGUI
GUI, Image and Video I/O

CXXCORE
basic structures and algorithms, XML support, drawing functions

Cvaux
Auxiliary (experimental) OpenCv functions

CvCam
cross-platform module for processing video stream from digital video cameras

ML
Machine Learning methods
OpenCV

How to obtain the library
Available on Sourceforge
http://sourceforge.net/projects/opencvlibrary/
(Or use your favorite search engine)

How to install the library
(On Windows)
Download and Install the Executable
Configuring MSVS .net 2005

Creating the Project

A project is initially created by selecting:
File -> New -> Project

Create a “Win32 Console Application”

Make it an “Empty Project” by selecting the box under “Application Settings”
Configuring MSVS .net 2005

Create the First File
Right Click the “Source Files” Folder under the project name (“Test1” in this case)
Add -> Add new Item

Select “C++ file(.cpp)” and give it a name

Creating a file makes it possible to set “Additional Include Directives” in the C/C++ pane under the project properties.
Configuring MSVS .net 2005

In order to build projects using OpenCV the required libraries and directives must be included in the project’s properties.

Open the Properties Pane
Right Click the name of the project and select "Properties" ("Test1" in this case)
Configuring MSVS .net 2005

Set Additional Include Directives
Under the C/C++ tab select “General”

Select the “Additional Include Directories”

Add the full path to each of the folders which contain “.h” files required to use OpenCV
Be sure to include trailing “\”

Utilized Directives
D:\OpenCV\cvaux\include\  
D:\OpenCV\xcore\include\  
D:\OpenCV\cv\include\  
D:\OpenCV\other\libs\highgui\  
D:\OpenCV\other\libs\cvcam\include\  


Configuring MSVS .net 2005

Utilized Directives

..\..\cvaux\include\n..\..\cxcore\include\n..\..\cv\include\n..\..\otherlibs\highgui\n..\..\otherlibs\cvcam\include\n
Configuring MSVS .net 2005

Set Additional Dependencies
Under the Linker tab select "Input"

Select the "Additional Dependencies"

Add the full path to each of the ".lib" files required to use OpenCV

Be sure to keep the paths in quotes

Utilized Dependencies

"D:\OpenCV\lib\cv.lib"
"D:\OpenCV\lib\cvaux.lib"
"D:\OpenCV\lib\cxcore.lib"
"D:\OpenCV\lib\cv僵.intilib"
"D:\OpenCV\lib\highgui.lib"
Configuring MSVS .net 2005

Utilized Dependencies

“..\..\lib\cv.lib“
“..\..\lib\cvaux.lib“
“..\..\lib\cxcore.lib“
“..\..\lib\cvcam.lib“
“..\..\lib\highgui.lib“
Now that the environment is configured it would be a good idea to test it to make sure that a program will correctly build and run.

Testing the First Program
The enclosed code can be cut and pasted into the file created in the project space to test OpenCV

```c
#include <cv.h>
#include <highgui.h>

/*
   This will pop up a small box with "Welcome to OpenCV"
   as the text.
   @author: Amir Hossein Khalili a_khalili@ce.sharif.edu
   imitated from Gavin Page, gsp8334@cs.rit.edu
   @date: 1 March 2007
*/
int main( int argc, char** argv ) {
    //declare for the height and width of the image
    int height = 620;
    int width = 440;
    //specify the point to place the text
    CvPoint pt = cvPoint( height/4, width/2 );
    //Create an 8 bit, 3 plane image
    IplImage* hw = cvCreateImage(cvSize(height, width), 8, 3);
    //Clearing the Image
    cvSet(hw,cvScalar(0,0,0));
    //Initialize the font
    CvFont font;
    cvInitFont( &font, CV_FONT_HERSHEY_COMPLEX, 1.0, 1.0, 0, 1, CV_AA);
    //place the text on the image using the font
    cvPutText(hw, "Welcome To OpenCV", pt, &font,
              CV_RGB(150, 0, 150));
    //create the window container
    cvNamedWindow("Hello World", 0);
    //display the image in the container
    cvShowImage("Hello World", hw);
    //hold the output windows
    cvWaitKey(0);
    return 0;
}
```
Testing MSVS .net 2005

Now that the environment is configured it would be a good idea to test it to make sure that a program will correctly build and run.

Testing the First Program
The enclosed code can be cut and pasted into the file created in the project space to test OpenCV
At this point you should have a working OpenCV project. If the program is not working you should go back and carefully recheck the steps.

From here you can explore the documentation to review the functions available.

There are also a number of tutorials on the web including: http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow
or you can just search for them

You should also join the OpenCV Community located at: http://groups.yahoo.com/group/OpenCV/
As of today there are >15000 members available to answer questions. There is also a searchable message board where you can look up previous queries.
Memory management

- Why is Managing OpenCV objects Important?
  - Video, 30 frames per second
  - Each frame is an image
  - Images are arrays of pixels
  - A 640x480 image is 307,200 pixels
  - These must be represented in memory
  - How much memory does your machine have?

```c
void cvResize( const CvArr* src, CvArr* dst, int interpolation )
```

The metatype CvArr* is used only as a function parameter to specify that the function accepts arrays of more than a single type, for example IplImage*, CvMat* or even CvSeq*. The particular array type is determined at runtime by analyzing the first 4 bytes of the header.
Image data structure

IplImage

- int nChannels
- int depth
- int width
- int height
- char* imageData
- int dataOrder
- int origin
- int widthStep
- int imageSize
- struct _IplROI *roi
- char *imageDataOrigin
- int align
- char colorModel[4]

IPL_DEPTH_<bit_depth>(S|U|F)

cvCreateImage( CvSize size, int depth, int channels )
cvLoadImage( const char* filename, int iscolor=1 )
cvReleaseImage( IplImage** image )

When allocating IplImage in a loop be sure to deallocate in the loop as well

cvSetImageROI( IplImage* image, CvRect rect );

cvRect( int x, int y, int width, int height )
cvLoadImage

Supported formats:

- Windows bitmaps - BMP, DIB;
- JPEG files - JPEG, JPG, JPE;
- Portable Network Graphics - PNG;
- Portable image format - PBM, PGM, PPM;
- Sun rasters - SR, RAS
- TIFF files - TIFF, TIF.
# Functions

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Statistics</td>
<td>In region of interest: Count, Mean, STD, Min, Max, Norm, Moments, Hu Moments.</td>
</tr>
<tr>
<td>Morphology</td>
<td>Erode, dilate, open, close. Gradient, top-hat, black-hat.</td>
</tr>
<tr>
<td>Distance Transform</td>
<td>Distance Transform</td>
</tr>
<tr>
<td>Thresholding</td>
<td>Binary, inverse binary, truncated, to zero, to zero inverse.</td>
</tr>
<tr>
<td>Flood Fill</td>
<td>4 and 8 connected</td>
</tr>
<tr>
<td>Histogram (recognition</td>
<td>Manipulation, comparison, backprojection</td>
</tr>
<tr>
<td>Eigen Objects</td>
<td>Calc Cov Matrix, Calc Eigen objects, decomp. coefs. Decomposition and projection.</td>
</tr>
</tbody>
</table>
Sample Program

• Extracting edges with sobel

```c
#include "cv.h"
#include "highgui.h"
int main( int argc, char** argv )
{
    char* fileAddress="pic.png";
    IplImage* orginallImage = cvLoadImage(fileAddress,0);
    cvNamedWindow("Orginal Image");
    cvShowImage("Orginal Image", orginallImage);
    IplImage* edgeImage =
    cvCreateImage(cvGetSize(orginallImage),IPL_DEPTH_16S,1);
    cvSobel(orginallImage,edgeImage,0,1);
    IplImage* absEdgeImage =
    cvCreateImage(cvGetSize(orginallImage),IPL_DEPTH_8U,1);
    cvConvertScale(edgeImage,absEdgeImage,1);
    cvNamedWindow("Edge Image");
    cvShowImage("Edge Image", absEdgeImage);
    cvWaitKey(0);
    cvReleaseImage(&orginallImage);
    cvReleaseImage(&edgeImage);
    cvDestroyWindow("orginal Image");
    cvDestroyWindow("Edge Image");
}
```
## Sampling, Interpolation

<table>
<thead>
<tr>
<th>GetRectSubPix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieves pixel rectangle from image with sub-pixel accuracy</td>
</tr>
<tr>
<td>void cvGetRectSubPix( const CvArr* src, CvArr* dst, CvPoint2D32f center );</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GetQuadrangleSubPix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieves pixel quadrangle from image with sub-pixel accuracy</td>
</tr>
<tr>
<td>void cvGetQuadrangleSubPix( const CvArr* src, CvArr* dst, const CvMat* map_matrix );</td>
</tr>
</tbody>
</table>
Pyramids and the Applications

- void cvPyrDown( const CvArr* src, CvArr* dst, int filter=CV_GAUSSIAN_5x5 );
- void cvPyrUp( const CvArr* src, CvArr* dst, int filter=CV_GAUSSIAN_5x5 );
Canny Edge Detector

- `void cvCanny( const CvArr* image, CvArr* edges, double threshold1, double threshold2, int aperture_size=3 );`
Image Thresholding

Source picture

Fixed threshold

Adaptive threshold
Working with Histograms

IplImage* grayImage = cvCreateImage(cvSize(im->width, im->height), IPL_DEPTH_8U, 1);
cvCvtColor(im, grayImage, CV_BGR2GRAY);
CvRect rect = cvRect(0, 0, 500, 600);
cvSetImageROI(grayImage, rect);

The cvCvtColor function can be used to convert images to one of several color spaces. In order to work with a histogram the image will have to converted to a single plane.
Working with Histograms

CvHistogram* hist =
   cvCreateHist(1, &hist_size, CV_HIST_ARRAY, ranges, 1);

cvCalcHist( &grayImage, hist, 0, NULL );

cvGetMinMaxHistValue( hist, &min_value, &max_value, &min_idx, &max_idx);

[Diagram with nodes:
  Calculate the Histogram
  Grab Min/Max Values]
OpenCV has several other functions for working with histograms. These include:
- cvNormalizeHist
- cvThreshHist
- cvCompareHist

For more information about usage of these functions see the OpenCV documentation.
Morphological Operations

Image $I$

Erosion $I \ominus B$

Dilatation $I \oplus B$

Opening $IoB = (I \ominus B) \oplus B$

Closing $I \bullet B = (I \oplus B) \ominus B$

Grad($I$) = $(I \oplus B) -(I \ominus B)$

TopHat($I$) = $I - (I \ominus B)$

BlackHat($I$) = $(I \oplus B) - I$
Accessing image elements

- Assume that you need to access the $K$-th channel of the pixel at the $i$-row and $j$-th column. The row index is in the range $[0\text{-height}-1]$. The column index is in the range $[0\text{-width}-1]$. The channel index is in the range $[0\text{-nchannel}-1]$.

<table>
<thead>
<tr>
<th>Indirect access</th>
<th>Onother direct access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CvScalar s; s=cvGet2D(img,i,j); Int value = s.val[k]; s.val[k]=111; cvSet2D(img,i,j,s);</td>
<td>int height = img-&gt;height; int width = img-&gt;width; int step = img-&gt;widthStep/sizeof(float); int channels = img-&gt;nChannels; TYPE * data = (TYPE <em>)img-&gt;imageData; data[i</em>step+j*channels+k] = 111;</td>
</tr>
</tbody>
</table>

Direct access

Value =((TYPE *)(img->imageData + i*img->widthStep))[j*img->nChannels + 0]=111
### Some other useful data structures

**CvMat**

OpenCV uses the CvMat* as its general purpose matrix structure. It is managed in an equivalent style topllImage*

```c
cvCreateMat( int rows, int cols, int type );
cvReleaseMat( CvMat** mat );
```

**CvScalar**

4D vector : double val[4]

```c
CvScalar s = cvScalar(double val0, double val1, double val2, double val3)
```

**CvMatND**

Multi Dimensional version of CvMat

**CvSparseMat**

SPARSE N-dimensional array

```c
void cvSet( CvArr* arr, CvScalar value, const CvArr* mask=NULL )
```
Some other useful data structures

<table>
<thead>
<tr>
<th>CvSeq</th>
<th>CvMemStorage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Growable 1-D array, Link list, Queue,</td>
<td>Growing memory storage</td>
</tr>
<tr>
<td>Stack</td>
<td></td>
</tr>
</tbody>
</table>
| cvCreateSeq( int seq_flags, int header_size, int elem_size, CvMemStorage* storage ); | cvCreateMemStorage( int block_size=0 );
|                                            | cvClearMemStorage( CvMemStorage* storage ) |

Points

- CvPoint p = cvPoint(int x, int y);
- CvPoint2D32f p = cvPoint2D32f(float x, float y);
- CvPoint3D32f p = cvPoint3D32f(float x, float y, float z);

Rectangular dimensions

- CvSize r = cvSize(int width, int height);
- CvSize2D32f r = cvSize2D32f(float width, float height);

Rectangular dimensions with offset

- CvRect r = cvRect(int x, int y, int width, int height);
Working with video sequences

**Initializing Capture**
- From a camera
  - CvCapture* capture = cvCaptureFromCAM(0);
- From a file
  - CvCapture* capture = cvCaptureFromAVI("infile.avi");

**Grabbing, decompressing and returning frame**
- cvQueryFrame
  - IplImage* cvQueryFrame( CvCapture* capture );
- The returned image should not be released or modified by user

**Processing the image**

**NULL**

**Releasing the capture source**
- cvReleaseCapture
  - the image captured by the device is allocated /released by the capture function. There is no need to release it explicitly
  - cvReleaseCapture(&capture);
Motion Analysis and Object Tracking

- Background subtraction
- Motion templates
- Optical flow
- Active contours
- Estimators
Background subtraction

• describes basic functions that enable building statistical model of background for its further subtraction.
• Background statistics functions:
  ✓ Average
  ✓ Standard deviation
  ✓ Running average
  \[
  \mu_{ij}^t = \alpha \cdot I_{ij}^t + (1 - \alpha) \cdot \mu_{ij}^{t-1}, \quad 0 \leq \alpha \leq 1
  \]
Motion templates

- To generate motion template images that can be used to rapidly determine where a motion occurred, how it occurred, and in which direction it occurred.
  - Object silhouette
  - Motion history images
  - Motion history gradients
  - Motion segmentation algorithm

![Image of silhouette, MHI, and MHG]
Optical Flow

- Block matching technique
- Horn & Schunck technique
- Lucas & Kanade technique
- Pyramidal LK algorithm
- 6DOF (6 degree of freedom) algorithm
Active Contours

- Snake energy: \( E = E_{\text{int}} + E_{\text{ext}} \)
- Internal energy: \( E_{\text{int}} = E_{\text{cont}} + E_{\text{curv}} \)
- External energy: \( E_{\text{ext}} = E_{\text{img}} + E_{\text{con}} \)
- Two external energy types:

\[
E_{\text{img}} = -I, \\
E_{\text{img}} = -\| \text{grad}(I) \|, \\
E = \alpha \cdot E_{\text{cont}} + \beta \cdot E_{\text{curv}} + \gamma \cdot E_{\text{img}} \Rightarrow \text{min}
\]
Estimators

- Kalman filter
- ConDensation filter
Saving a video file

Initializing a video writer

Writing frames to video file

Is there more

Releasing the video writer

CvVideoWriter *writer = 0;
int isColor = 1;
int fps = 25; // or 30
int frameW = 640; // 744 for firewire cameras
int frameH = 480; // 480 for firewire cameras
writer=cvCreateVideoWriter("out.avi",
CV_FOURCC(‘P’,’I’,’M’,’1’),
fps,cvSize(frameW,frameH),isColor);

IplImage* img = 0;
int nFrames = 50;
for(i=0;i<nFrames;i++)
{
    Img=cvQueryFrame(capture);
    cvWriteFrame(writer,img);
}

cvReleaseVideoWriter(&writer);
# Possible Codecs for saving

<table>
<thead>
<tr>
<th>Codec</th>
<th>fourcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-1</td>
<td>CV_FOURCC(‘P’,‘I’,‘M’,‘1’)</td>
</tr>
<tr>
<td>motion-jpeg</td>
<td>CV_FOURCC(‘M’,‘J’,‘P’,‘G’)</td>
</tr>
<tr>
<td>MPEG-4.2</td>
<td>CV_FOURCC(‘M’,‘P’,‘4’,‘2’)</td>
</tr>
<tr>
<td>MPEG-4.3</td>
<td>CV_FOURCC(‘D’,‘I’,‘V’,‘3’)</td>
</tr>
<tr>
<td>MPEG-4</td>
<td>CV_FOURCC(‘D’,‘I’,‘V’,‘X’)</td>
</tr>
<tr>
<td>H263</td>
<td>CV_FOURCC(‘I’,‘2’,‘6’,‘3’)</td>
</tr>
<tr>
<td>FLV1</td>
<td>CV_FOURCC(‘F’,‘L’,‘V’,‘1’)</td>
</tr>
</tbody>
</table>

A codec code of -1 will open a codec selection window (in windows).
Thank You! Questions?