A Review on OpenCV

Research · August 2015

DOI: 10.13140/RG.2.1.2269.8721

CITATIONS
3

READS
7,391

1 author:

Mumtazimah Mohamad
Universiti Sultan Zainal Abidin | UniSZA
58 PUBLICATIONS **285** CITATIONS

SEE PROFILE

A Review on OpenCV

Mumtazimah Mohamad[1], Md Yazid Mohd Saman[2], Muhammad Suzuri Hitam[3]
Department of Computer Science
Faculty Science and Technology
Universiti Malaysia Terengganu (UMT)
Mengabang Telipot,
21030 Kuala Terengganu, Terengganu Darul Iman

1.0 Introduction to OpenCV

OpenCV (**Open** Source **Computer Vision** Library) is an Application Peripheral Interface (API) developed by Intel which can be used for many image processing and computer vision applications. OpenCV officially launched in 1999 and the project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls.

OpenCV library is a collection of algorithms and C/C++ functions and a few classes that implement some Image processing and computer vision algorithms. There is active development on interfaces for Phyton, Ruby, Matlab and other languages. OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV is written in optimised C and can take advantage of multicore processors. If there is a need to further automatic optimisation on Intel architecture that help, we need Intel's Integrated Primitives (IPP) libraries which consist of low-level optimized routines in many different algorithmic area. OpenCV automatically uses appropriate IPP libray at runtime if that library is installed. OpenCV contains over 500 function that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision and robotics.

The principles behind the creation of the library is to aid commercial uses of computer vision in human-computer interface, robotics, monitoring, biometrics and security by providing a free and open infrastructure where the distributed efforts of the vision community can be consolidated and performance optimized. OpenCV support for vision is extensively including routine support for input, display, and storage of movies and single images. OpenCV uses DirectX, which is a set of APIs developed by Microsoft for creating multimedia applications and games.

One of the OpenCV goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. There are several goals of OpenCV in outset which are following:

- Advance vision research by providing not only open but also optimised code for basic vision infrastructure.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance-vision based commercial applications by making portable, performance-optimised code available for free- with a license that did not required commercial applications to be open or free themselves

1.1 The advantage of OpenCV

[1] mumtaz@udm.edu.my
[2] yazid@umt.edu.my
[3] suzuri@umt.edu.my

OpenCV provide a set of image processing functions and computer vision applications. The functions are optimized for Intel architecture processors and are particularly effective with MMX technology. The OpenCV Library is a way of establishing an open source vision community that will make better use of up-to-date opportunities to apply computer vision in growing PC environment and mobile platform. The library is open and has platform-independent interface and supplied with whole C sources.

OpenCV was designed to be portable. It was originally written to compile across Borland C++, Microsoft Visual Studio C++, and the Intel compilers. C and C++ code had to be fairly standard in order to make cross-platform support easier. OpenCV library is multi platform, and runs on both Windows and Linux Operating System. OpenCV is quickly gaining popularity for developing real-time computer vision applications. Some examples of applications include face recognizers, object recognizers, and motion trackers, just to name a few. The library has especially gained popularity in the computer vision research community. It allows researchers to get demos or research projects up and running quickly, and take advantage of the large collection of algorithms that are already available.

1.2 Image Processing or Computer Vision

The use of term of computer vision and image processing is commonly interleaved. In contrary, there is a gap between computer vision and image processing. Image processing is of low-level processing of still or video image, while computer vision is high-level processing of still or video image. OpenCV is specifically designed to an advent to computer vision development. OpenCV aimed at providing the basic tools needed to solve computer vision problem. High level library will be sufficient to solve more complex problems in computer vision. Basic components in the library are complete enough to create complete solution to almost any computer vision problem.

1.2.1 Low-Level Processing (Image Processing)

Image processing is among **low-level processes** that involve primitive operations such as image pre-processing to reduce noise. Low level image processing data are comprised of original images represented by matrices composed of brightness values (Sonka, Hvalac and Boyle ,1999). Low level methods usually use very little knowledge about the content of images. Low level characterised by the fact that both its inputs and outputs is image. Low level image processing sequence involved pre- processing and image segmentation.

Image processing involves changing the nature of an image either improvement of pictorial information or human interpretations or processing of image data for storage, transmission, and representation for autonomous machine perception (McAndrew, 2004). Image processing typically attempts to accomplish one of three things

- restoring images
- enhancing images
- understanding images

Restoration takes a corrupted image and attempts to recreate a clean original. Enhancement alters an image to makes it's mean and clearer to human observers. Understanding usually attempts to mimic the human visual system in extracting meaning from an image.

1.2.2 High-Level Processing (Computer Vision)

On the other hand, computer vision is high-level processing. High level processing involves in understanding or "making sense" of a group of recognized objects, as image analysis and finally perform the cognitive functions associated with human vision. High-level computer

vision tries to imitate human cognition and the ability to make decisions according to the information contained in the image. For example object size, shape, and mutual relations between objects in the image. High level data are usually expresses in symbolic form. (Sonka, Hvalac and Boyle,1999). High level vision tries to extract and order image processing steps using all available knowledge – image understanding

Computer vision is the transformation of a still or video image data onto either a decision or a new representation (Bradski & Kaehler, 2008). All such transformations are done for achieving some particular goal. The input data may include some contextual information such as "The camera mounted in the car" or "laser range finder indicates an object is 1 meter away". The decision might be the existence of a person in a scene or the existence of 14 tumour cells on the slide. A new representation might mean turning a colour image into a grey scale image or removing camera motion from an image sequence.

1.2.3 Computer Vision and Image Processing

Both Image processing and computer vision can be machine vision where, computer or a machine, receives a grid of numbers from camera or disk with no built-in pattern recognition, or automatic control of focus. The following figure shows a sampling of an image and its grid of numbers. Computer "sees" is just a grid of number. Any given number within that grid has a rather noise component and so by itself gives us little information.

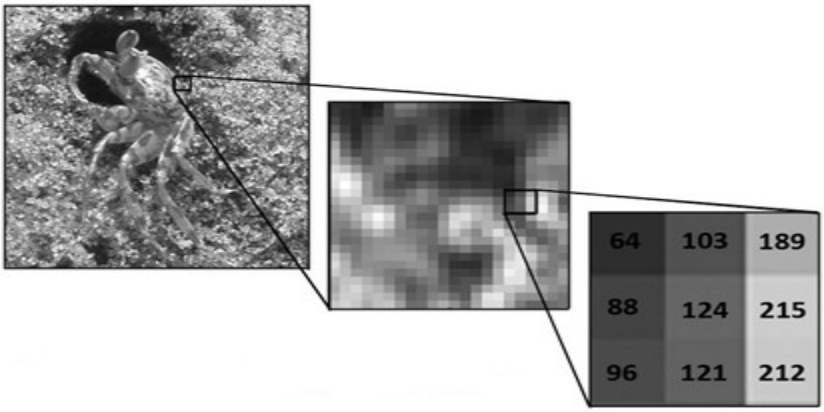

Figure 1.0 Sampling of an image (left) and its grid of number (right)

Generally, in image processing, the input of the process is image and the output processed would be a processed image. The best example of image processing is compression or edge detection. While computer vision use the image either still image or video image to convert it into symbols. The close example of computer vision such as face recognition, object tracking and et cetera.

1.3 OpenCV functionalities

OpenCV support the the major functionality and categories as follows:

- *General computer-vision and image-processing algorithms (mid- and low-level APIs)*

Many standard computer-vision algorithms can be experimented using these interfaces without having to code them. These include filtering, edge, line, and corner detection, ellipse fitting, image pyramids for multi-scale processing, template matching, colour conversion, morphological operations, histograms, various transforms (Fourier, discrete

cosine, and distance transforms), and more. Not to forget, image data manipulation which includes allocation, release, copying, setting and conversion.

- *High-level computer vision modules*

OpenCV includes several high-level capabilities. In addition to face-detection, recognition, and tracking, it includes optical flow (using camera motion to determine 3D structure), camera calibration, and stereo. Camera calibration functions can be used in finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence. It also includes Image and video I/O (file and camera based input, image/video file output) and motion analysis (optical flow, motion segmentation, tracking)

- *AI and machine-learning methods*

  Computer-vision applications often require machine learning or other AI methods. Some of these are available in OpenCV's Machine Learning package. e.g. Object recognition (eigen-methods, HMM)

- *Image sampling and view transformations*

It's often useful to process a group of pixels as a unit. Using matrix and vector manipulation and linear algebra routines, OpenCV includes interfaces for extracting image sub regions, random sampling, resizing, warping, rotating, and applying perspective effects.

- *Methods for creating and analyzing binary (two-valued) images*

Binary images are frequently used in inspection systems that scan for shape defects or count parts. A binary representation is also convenient when locating an object to grasp. Structural analysis that can be used for image are connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, or Delaunay triangulation.

- *Methods for computing 3D information*

These functions are useful for mapping and localization - either with a stereo rig or with multiple views from a single camera.

- *Math routines for image processing, computer vision, and image interpretation*

OpenCV includes math commonly used algorithms from linear algebra, statistics, and computational geometry.

- *Graphics*

OpenCV enable function to write text and draw on images (line, conic or polygon). In addition to various fun and creative possibilities, these functions are useful for labeling and marking. For example, a written program that detects objects can easily label images with their sizes and locations.

- *GUI methods*

OpenCV includes its own windowing interfaces. While these are limited compared to what can be done on each platform, they provide a simple, multi-platform API to display images, accept user input via mouse and its handling or keyboard, and implement slider controls or scrollbars.

- *Data structures and algorithms*

OpenCV efficiently and dynamically store, search, save, and manipulate large lists, collections (also called sets), queues, sets, trees, and graphs .

- *Data persistence*

These methods provide convenient interfaces for storing various types of data to disk and retrieving them later.

1.5 Basic Library Element

OpenCV library module is basically divided into five modules.

i. CV module

The CV component contains the basic image processing and higher-level computer vision algorithms that provide following features:

- Filters, geometrical transformations, color space transforms
- Image analysis (feature selection, morphology, contour retrieval, histograms)
- Structural analysis (shape descriptors, planar subdivisions)
- Motion analysis and object tracking
- Object/face detection
- Camera calibration and elements of 3D reconstruction

ii. CXCORE

CXCORE contains basic data type definitions. For example, the data structures for image, point, and rectangle are defined in cxtypes.h. CXCORE also contains linear algebra and statistics methods, the persistence functions, and error handlers. Some of the graphics functions for drawing on images are located here as well. Other than that, CXCORE also provides basic data structure as follows:

- mathematics and matrix and algorithm, linear algebra support
- Simple operations on dense arrays
- Matrix algebra, math functions, RNG
- DFT, DCT
- Serialization to XML/YAML (data persistence)
- Drawing functions (2D graphics)
- Complex data structures: sparse matrices, growing sequences, graphs

iii. Highgui

Highgui contains the multi-platform windowing capability, Input and Output routines, GUI and functions for storing and loading video and images such as

- Image and video input and output
- Image/video acquisition.
- Simple GUI facilities (all OpenCV visual samples use HighGUI)

iv. ML

Ml is Machine Learning library which includes many statistical classifiers and clustering tools.

v. CVAUX

CVAUX is described in OpenCV's documentation as containing obsolete and experimental code. However, the simplest interfaces for face recognition are in this module. The code behind them is specialized for face recognition, and they're widely used for that purpose. CVAUX is not particularly documented in OpenCV directory. Other CVAUX function such as :

- face detection, camera calibration, motion analysis
- 3D vision: stereo calibration, trifocal tensor, bundle adjustment
- Stereo correspondence, graph cliques
- Face details detection and tracking
- Shape matching and skeletons
- Textures

vi. CVCAM

CVCAM contains interfaces for video access through DirectX on 32-bit Windows platforms. However, HighGUI also contains video interfaces . In this article, I'll cover only the interfaces in HighGUI. They're simpler to use, and they work on all platforms

1.6 Relation between OpenCV and other libraries

OpenCV is designed to be used together Intel Image Processing Library (IPL) which is a set of low-level image processing functions. It also uses Integrated Performances Primitives (IPP) on lower-level, which provides cross-platform interface to highly-optimized low level functions that perform image processing and computer vision operations. OpenCV exploits the highly optimised code in IPP to speed itself up. The improvement in speed from using IPP can be substantial. Performance can be upgraded by using this library to run vision code in real time. OpenCV is no depend in any way on IPP but if IPP present, OpenCV can take advantage of IPP by loading IPP's dynamic link libraries to further enhance its speed.

1.7 OpenCV Coding Style

The core libraries (cv, cvaux) are written in C/C++, so the document concerns only the codes written in C/C++.

- **File Names**

All the file names of cv and cvaux libraries must obey the following rules:

- All the files of CV library have prefix cv.
- Mixed C/C++ interface headers have .h extension
- Pure C++ interface headers have .hpp extension
- Implementation files have .cpp extension
- Names are written all in lower case because of compatibility with POSIX.

- **File Structure**

Every file starts with BSD-compatible license, which template can be found in Contributors_BSD_License.htm file. Other rules for both header and implementation files include:

- Maximal line length is 90 symbols, not including end-of-line characters
- No tabulation used

- Indentation is four spaces, so the tabulations should be replaced with 1-4 spaces, depending on the starting column

Header files must use guarding macros, protecting the files from repeated inclusion. Mixed C/C++ interface header files contain extern "C"{}, surrounding C definitions. Source files must include precomp.h header before other headers, in order to make precompiled headers mechanism in Visual C++ work properly.

- **Naming conventions**

OpenCV uses mixed-case style identifiers for external functions, types and class methods. Macros are written with all capital letters, words are separated by underscore. All the external names or internal names, visible across several files, must have prefixes:

- Cv for external functions
- Icv for internal functions
- Cv for data structures (C structures, enumerations, unions, classes)
- CV_ for external and some of internal macros
- ICV_ for internal macros.

2 Installation guide in Windows Environment

2.2 Required Software

To configure and build a project using OpenCV, we need to have the following software:

i. Visual Studio, and the MSDN library

- Microsoft Visual Studio .Net 2005 (alternatively Microsoft Visual Studio Express)
- Visual Studio 2005 Service Pack 1 Update for Windows Vista (can be downloaded at msn.com)
- Microsoft Windows SDK .NET Framework 3.0 (can be downloaded at msn.com)
- Microsoft Redistributable. (Optional, due to different of OS used)

ii. DirectX SDK Updates and DirectX Extras

After completing above installation, we need also to have Microsoft DirectX 9.0 SDK Update (December 2004 or latest) and Microsoft DirectX 9.0 SDK Update (December 2004 or latest) Extras. Both of them can be downloaded from http://www.microsoft.com. Once this file has successfully downloaded, double click it and follow the instructions to install it. Next, install the extras by clicking the setup file downloaded. Double click it and extract the files to a temporary folder. The folder created at this location will be a folder called “Extras”. Take this folder and put it in the DirectX 9.0 SDK folder, which by default should have the location C:\Program Files\Microsoft DirectX 9.0 SDK (December 2004). This completes the installation of DirectX. Where DirectX is stored is very important since later we’ll be specifying paths so Visual C++ 6.0 can find DirectX.

iii. OpenCV

The main OpenCV site is http://www.intel.com/research/mrl/research/opencv/ and the openCV version can downloaded from http://www.sourceforge.com. Download the latest version by finding the package called “opencv-win”, and click the download button. On the next page, click “OpenCV_b4a.exe” under “beta4”. Now, choose a close mirror site to download the file from. A self-extracting zip file of size 12.4 MB called “OpenCV_b4a.exe”

will be downloaded. Double click the file and follow the instructions to install OpenCV v4.0-Beta, which requires at least 36.2 MB. Install OpenCV in the default folder, i.e., “C:\Program Files\OpenCV”. Leave the check box under additional tasks checked so the system will add the specified path to the system PATH variable. This guide covers installation on Windows XP. However, OpenCV may be installed on other operating systems including Windows 95/98/2000/NT and Linux.

2.2 Configuring Visual C++ and OpenCV

There are some configurations needed to make sure Visual C++ can work with OpenCV. After completing the above installation, we need to begin with setting up the system PATH variable.

Figure 2.1 : Setting Up System Path (1)

This is due to link DLLs to OpenCV library. Open control panel and click the “Advanced tab” in System windows. Choose “*Environment Variable*”. Click the *Environment Variables* that is in highlighted. The window like the adjacent figure will appear. Click *Path* and then edit by going to the end of the variable value line. Type "C:\Program Files\OpenCV\bin" and click on "OK" and once again click "OK" until the window "System Properties" window disappears.

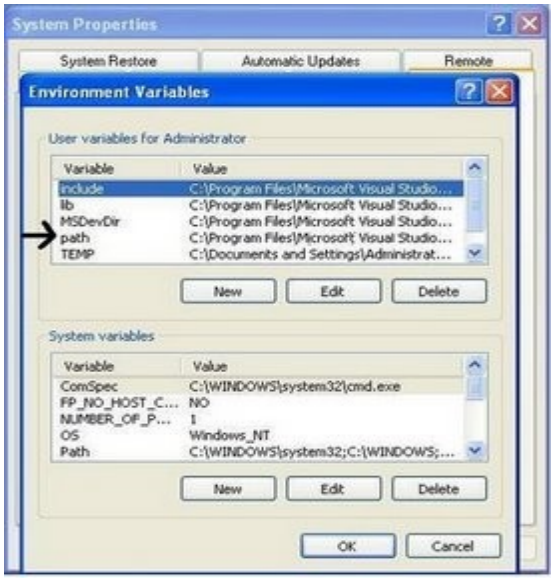

Figure 2.2: Setting Up System Path (2)

It might be required to restart Visual C++ when execution of instance failed after successful build. Lastly, to use OpenCV library in Visual C++ we need to customize global option in Visual C++.

2.3 Customize Global Options

In Visual C++ application, select ***Tools*** menu and choose ***Options***. In the listing, choose **Projects** and **VC++ Directories**. First, select **Library files** from the "**Show Directories for**" List Box. Click the **Insert New** icon, and locate the folder where you have installed.

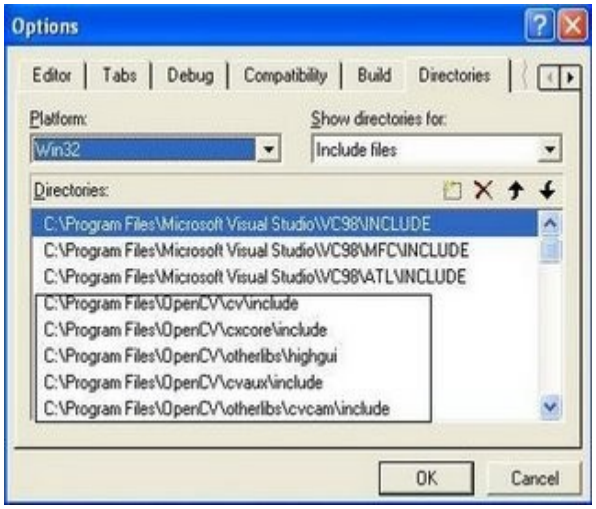

Figure 2.3: Setting Up Global Setting in MSVS (1)

Consider that OpenCV is installed in "C:/Program Files/OpenCV" by default. In the include files list, locate and add:

- "C:\Program Files\OpenCV\cv\include"
- "C:\Program Files\OpenCV\cxcore\include"
- "C:\Program Files\OpenCV\otherlibs\highgui"
- "C:\Program Files\OpenCV\cvaux\include"
- "C:\Program Files\OpenCV\otherlibs\cvcam\include"

These are the paths to be copied and make sure that you have copied and pasted them with the correct drop down options being "Win32" and "Include files".

Next, choose source files in the list box, and locate and add the following directories:

"C:\Program Files\OpenCV\cv\src"
"C:\Program Files\OpenCV\cxcore\src"
"C:\Program Files\OpenCV\cvaux\src"
"C:\Program Files\OpenCV\otherlibs\highgui"
"C:\Program Files\OpenCV\otherlibs\cvcam\src\windows"

Next, choose library files in the list box, and locate and add the path

- C:\Program Files\Microsoft DirectX 9.0 SDK (February 2005)\Lib\x86
- C:\Program Files\OpenCV\lib

Next, choose executable files in the list box, locate and add the following path

- C:\Program Files\Microsoft DirectX 9.0 SDK (February 2005)\Utilities\Bin\x86
- C:\Program Files\OpenCV\bin

Click Ok on the Options dialog.

Downloading and installing OpenCV in Linux and MacOs X may vary to Windows due to different environment.

2.4 Getting OpenCV Working

2.4.1 **Create New Project:**

Within Developer Studio create new application:

- Select from menu "File"->"New..."->"Projects" tab.

- Choose Visual C++ and choose "Win32 Application" or "Win32 console application" - the latter is the easier variant and both the sample projects have this type. So here again just the window is only different everything else is the same.

You may Choose "Win32 Console Application" and then type the name of the project and click "ok" and click to choose an empty project. After you click on finish you should be getting another window which says that no files would be added to the project just click finish.

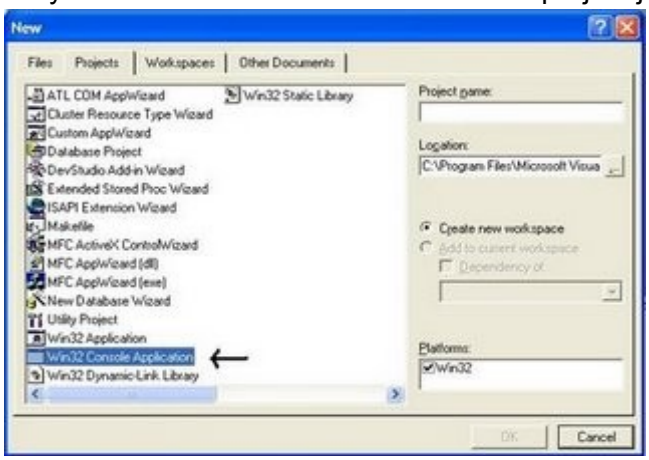

Figure 2.4 : Creating New Project (Step 1)

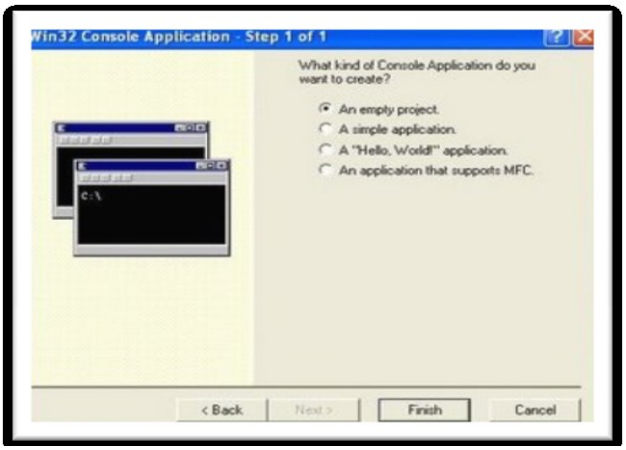

Figure 2.5 : Creating New Project (Step 2)

After completing to create project, add dependency projects into workspace. Choose from menu: "Project" and click "Properties".

- Choose "Linker" tab -> "Input" category -> "Additional Dependencies:". Add the paths to all neccessary import libraries (cxcore[d].lib cv[d].lib highgui[d].lib cvaux[d].lib cvcam[d].lib) .

Go to "Projects" and then click on "Settings" then you will find a window as follows. Click on the "Link" tab and then follow the next figure.

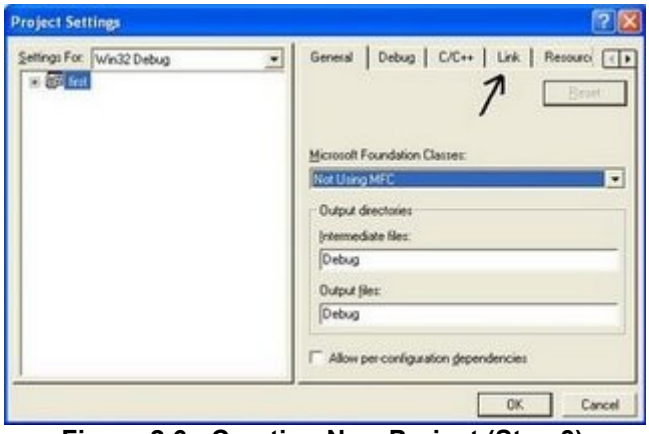

Figure 2.6 : Creating New Project (Step 3)

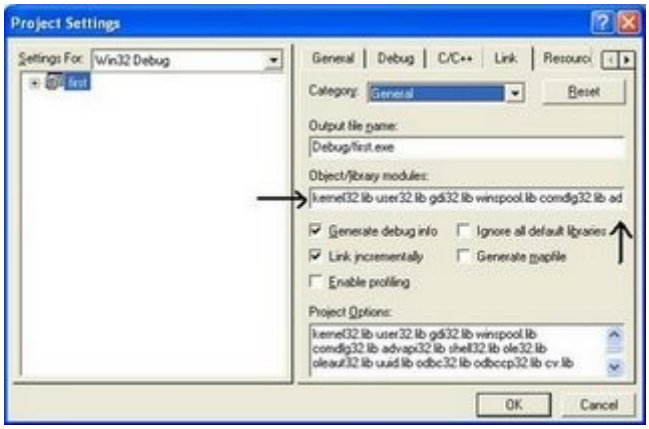

Figure 2.7 : Creating New Project (Step 4)

In this figure go to the end of the line which is indicated by the arrows and type ***"cv.lib cxcore.lib highgui.lib "*** and ensure the spaces after every word and click "OK". Most OpenCV programs need to include cv.h and highgui.h.The remaining header files are included by these top-level headers. If the header files left in multiple directories (by default installation), make sure the compiler's include path contains these directories. Visual Studio linker will need both the library path and the names of the static libraries to use (Figure 2.6 and 2.7). The static libraries are cxcore.lib, cv.lib, and highgui.lib. Later, the static library need to be added due to the requirement of the program type, e.g. face recognition need cvaux.lib to inserted here.

This is the end of configuration of installation. However, if the problem occurred during compiling the sample program, there should be thorough check to the problem due to inconsistency with update version either in Windows version, DirectX version and also Visual Studio version.

3.0 Some Basic Programming with OpenCV

After completing step-by-step above, the programming can be started. Add an item of C++ source code and a “hello.cpp” (as in Figure 3) can be a start program to explore the openCV programming.

```
/* Program : hello.cpp
This is a simple, introductory OpenCV program. The program reads an image
from a file, inverts it, and displays the result.*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cv.h>
#include <highgui.h>

int main(int argc, char *argv[])
{
  IplImage* img = 0;
  int height,width,step,channels;
  uchar *data;
  int i,j,k;

  if(argc<2){
    printf("Usage: main <image-file-name>\n\7");
    exit(0);
  }
    img=cvLoadImage(argv[1]); // load an image
  if(!img){
    printf("Could not load image file: %s\n",argv[1]);
    exit(0);
  }
  // get the image data
  height    = img->height;
  width     = img->width;
  step      = img->widthStep;
  channels  = img->nChannels;
  data      = (uchar *)img->imageData;
  printf("Processing a %dx%d image with %d channels\n",height,width,
channels);
  cvNamedWindow("mainWin", CV_WINDOW_AUTOSIZE); // create a window
  cvMoveWindow("mainWin", 100, 100);
```

```
  // invert the image
  for(i=0;i<height;i++) for(j=0;j<width;j++) for(k=0;k<channels;k++)
    data[i*step+j*channels+k]=255-data[i*step+j*channels+k];

    cvShowImage("mainWin", img ); // show the image
  cvWaitKey(0); // wait for a key
  cvReleaseImage(&img); // release the image
  cvDestroyWindow("mainWin");
  return 0;
}
```

When compiled and run from the command line with a single instrument, this program loads an image into memory and displays it on the screen. It then waits until the user presses a key, at which time it closes the window and exits.

`IplImage* img = 0;`

This line creates an image with allocated memory needed for image data structure. IplImage structure is the OpenCV construct is commonly deal with. OpenCV uses this structure to handle all kinds of images; single channel, multi channel, integer-valued, floating-point-valued, et cetera.

`img=cvLoadImage(argv[1]);`

Function cvLoadImage() is a high level routine that determines the file format to be loaded based on the file name. This function can read a variety of image formats, including BMP, DIB, JPEG, JPE, PNG, PBM, PGM, PPM, SR, RAS and TIFF. A pointer to an allocated image data structure is then returned to manipulate the image and image data.

`cvNamedWindow("mainWin", CV_WINDOW_AUTOSIZE);`

This is another function from high-level function of highgui library, opens a window on the screen that can contain and display an image. This function is assigned to the window named "mainWin". The future highgui calls will interact with this window will refer by this name. The second argument defines the properties of the window. It may be set either to 0 (default value) or to `CV_WINDOW_AUTOSIZE`. In this case of 0, the size of the windows will be the same regardless of the image size, and the image will be scaled to fit within the window. But for the latter case, the window will expand or contract automatically when an image is loaded to accommodate the image's true size.

`cvMoveWindow("mainWin", 100, 100);`

This also another function of highgui module. This function direct window to move at location 100, 100.

`cvShowImage("mainWin", img );`

After IplImage* pointer is created, it can be displayed it in an existing window with scShowImage () function requires that a named window already exist which is created by cvNamedWindow() earlier. The function causes the copy action to window buffer then the window be redrawn with the appropriate image in it, and the window will resize itself as appropriate if it was created using `CV_WINDOW_AUTOSIZE`.

The `cvWaitKey(0)` function asks the program to stop and wait for a keystroke. If positive argument is given, the program will wait for that number of milliseconds and then continue even if nothing is pressed. If the argument is set to 0 or to negative, the program will wait indefinitely for a keypress.

`cvReleaseImage(&img );`

This function will free the allocated memory. The pointer img will be set to NULL.

`cvDestroyWindow("mainWin");`

This function will close the window and de-allocate aby associated memory usage including internal buffer, which holds a copy of the pixel information from *img. For a simple program, the call for *cvReleaseImage* and *cvDestroyWindow* is not really needed since the resources and windows of the application are closed automatically by the operating system upon exit.

The following program is an example of simple OpenCV program for displaying video.

```
/* This is a simple, introductory OpenCV program for video display.*/

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

int main(int argc, char **argv)
{
  cvNamedWindow("Main Window", CV_WINDOW_AUTOSIZE); // create a window
  CvCapture* capture=cvCreateFileCapture(argv[1]);
  IplImage* frame;
  while(1) {
     frame =cvQueryFrame(capture);
  if(!frame)
     break;
     cvShowImage("Video", frame ); // show the image
     char c=  cvWaitKey(33); // wait for a key
     if(c==27)
       break;
     }
  cvReleaseCapture(&capture ); // release the imagecapture
  cvDestroyWindow("Main Window");
}
```

```
CvCapture* capture=cvCreateFileCapture(argv[1]);
```
The function cvCreateFileCapture takes as its argument the name of AVI file to be loaded and then returns a pointer to CvCapture structure. CvCapture structure contains all of information about the AVI file being read, including state information. In other word, CvCapture initialised to the beginning of AVI.

```
frame =cvQueryFrame(capture);
```
This function is for reading AVI file, in this case in looping. cvQueryFrame() function takes as its argument a pointer to a CvCapture structure. It grab the next video frame into memory (CvCapture has ready-allocated memory). A pointer is then returned to that frame. It is not necessarily to call cvrelease() for this frame pointer because it will be released when CvCapture structure is released.

```
char c=  cvWaitKey(33); // wait for a key
     if(c==27)
```
After displaying the frame, the system will wait for 33 milliseconds (ms). If the user hits a key, then c will be set to ASCII value of that key; if not, then it will be set to -1. If the user hits the Esc key (ASCII 27), then we will exit the read loop. If 33 ms is over without a key hit, it will execute and loop again.

```
cvReleaseCapture(&capture );
```
After exiting loop,because of there was no more video data or because the user hit the Esc key. CvReleaseCapture() will free memory associated with CvCapture structure and close any open file handles to the AVI file.

3.1 Data types supported

There are few fundamental types helper data types introduced to make OpenCV API simpler and more uniform. The fundamental data types includes:

- Array-like types: IplImage (IPL image) and CvMat for matrix
- Expandable and mixed type collections – CvSeq, CvSet, CvGraph and CvHistogram (multidimensional)
- Helper data types – CvPoint (2D point), CvSize (width and height), CvTermCriteria (termination criteria for iteration), CvMoments (spatial moments) and many more.

3.2 OpenCV library

3.2.1 CV library

The CV library contains Image Processing ,Structural Analysis, Motion Analysis Algorithms. The common useful functions such as:

- `void cvCanny` : for edge-detection
- `void cvGoodFeaturesToTrack`: determine strong corners on image
- `void cvCvtColor` :Converts image from one color space to another
  - (`CV_BGR2XYZ, CV_RGB2XYZ, CV_XYZ2BGR, CV_XYZ2RGB`)
- `void cvPyrSegmentation` :Implements image segmentation
- `void cvMoments` :*Calculates all moments up to third order of a polygon or rasterized shape*
- `CvHistogram* cvCreateHist` :*Creates histogram*
- `CvSeq* cvConvexHull2` :Finds convex hull of point set

3.2.2 CXCORE library

There are useful CXCORE library structures such as:

- `CvPoint` :*2D point with integer coordinates*
- `CvPoint2D32f`:*2D point with floating-point coordinates*
- `CvMat` :*Multi-channel matrix*
- `IplImage`*: IPL image header*

CXCORE function of OpenCV Operations that are useful and commonly used such as follows:

- `IplImage* cvCreateImage`*(CvSize size, int depth, int channels);*
- `IplImage* cvCloneImage`*(const IplImage* image);*
- `void cvSetImageROI`*(IplImage* image, CvRect rect);*
- Statistics
- Linear Algebra

3.2.3 HIGHGUI library

The general idea behind its design is to have a small set of directly useable functions to interface your computer vision code with the environment.

i. Window management

- Create and position a window:

```
cvNamedWindow("win1", CV_WINDOW_AUTOSIZE);
cvMoveWindow("win1", 100, 100); // offset from the UL corner of the
screen
```

- Load an image:

```
IplImage* img=0;
img=cvLoadImage(fileName);
```

```
if(!img) printf("Could not load image file: %s\n",fileName);
```

- Display an image:
  ```
  cvShowImage("win1",img);
  ```
  Can display a color or grayscale byte/float-image. A byte image is assumed to have values in the range . A float image is assumed to have values in the range . A color image is assumed to have data in BGR order.

- Close a window:
  ```
  cvDestroyWindow("win1");
  ```

- Resize a window:
  ```
  cvResizeWindow("win1",100,100); // new width/heigh in pixels
  ```

ii. Input handling

- Handle mouse events:
  o Define a mouse handler:
  ```
  void mouseHandler(int event, int x, int y, int flags,
  void* param){
  switch(event){
  case CV_EVENT_LBUTTONDOWN:
  if(flags & CV_EVENT_FLAG_CTRLKEY)
  printf("Left button down with CTRL pressed\n");
  break;
  case CV_EVENT_LBUTTONUP:
  printf("Left button up\n");
  break;
  }
  }
  ```
  x,y: pixel coordinates with respect to the UL corner

  ```
  event: CV_EVENT_LBUTTONDOWN, CV_EVENT_RBUTTONDOWN,
  CV_EVENT_MBUTTONDOWN,
  CV_EVENT_LBUTTONUP, CV_EVENT_RBUTTONUP,
  CV_EVENT_MBUTTONUP,
  CV_EVENT_LBUTTONDBLCLK, CV_EVENT_RBUTTONDBLCLK,
  CV_EVENT_MBUTTONDBLCLK,
  CV_EVENT_MOUSEMOVE:

  flags: CV_EVENT_FLAG_CTRLKEY, CV_EVENT_FLAG_SHIFTKEY,
  CV_EVENT_FLAG_ALTKEY,
  CV_EVENT_FLAG_LBUTTON, CV_EVENT_FLAG_RBUTTON,
  CV_EVENT_FLAG_MBUTTON
  ```

  o Register the handler:
  ```
  mouseParam=5;
  cvSetMouseCallback("win1",mouseHandler,&mouseParam);
  ```

- Handle keyboard events:

o The keyboard does not have an event handler.
o Get keyboard input without blocking:
```
int key;
key=cvWaitKey(10); // wait 10ms for input
```

o Get keyboard input with blocking:
```
int key;
key=cvWaitKey(0); // wait indefinitely for input
```

o The main keyboard event loop:

```
while(1){
key=cvWaitKey(10);
if(key==27) break;
switch(key){
case 'h':
...
break;
case 'i':
...
break;
}
}
```

- Handle trackbar events:

o Define a trackbar handler:

```
void trackbarHandler(int pos)
{
printf("Trackbar position: %d\n",pos);
}
```

o Register the handler:

```
int trackbarVal=25;
int maxVal=100;
cvCreateTrackbar("bar1", "win1", &trackbarVal ,maxVal ,
trackbarHandler);
```

o Get the current trackbar position:

```
int pos = cvGetTrackbarPos("bar1","win1");
```

o Set the trackbar position:

```
cvSetTrackbarPos("bar1", "win1", 25);
```

For video input and output, function and structure that are commonly used.

- `CvCapture* capture = 0;`
- `capture = cvCaptureFromCAM or cvCaptureFromAVI`
- For every frame in video
  - `IplImage* frame = 0;`
  - `frame = cvQueryFrame( capture );`
  - `image = cvCreateImage( cvGetSize(frame), 8, 3 );`
  - `image->origin = frame->origin;`
  - `cvCopy( frame, image, 0 );`

References

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV:Computer Vision with OpenCV Library.* California: O'Reilly Media Inc.

Internet Article originally appeared in SERVO Magazine, January 2007. Reprinted by permission of T & L Publications, Inc.