LAB 9

Fall 2010, BESE- 13 & 14

Image Processing with EmguCV

Objective

The purpose of today's lab is to introduce you to the EmguCV Library that is developed to perform Image Processing and Computer Vision operations in .NET languages such as C#, VB, VC++ etc. This lab covers fundamental datatypes and data structures defined in EmguCV library and a sample program to elaborate basic EmguCV operations for loading, showing, transforming and saving image. By the end of this lab you should be comfortable enough to read an image, performing basic transformations and writing processed image to disk.

NOTE: the code and functions explored in this LAB have basis from OpenCV (Lab-07) therefore their specs are not re-explained.

Tasks for Today

1. Introduction to EmguCV

EmguCV is an open source and cross platform .NET library. It encapsulates Intel® OpenCV image processing library and provides an easy access to OpenCV functions from dot NET languages such as Visual C#, Visual Basic, and Visual C++. In addition to the features it inherits from OpenCV library it also incorporates a variety of coding examples to facilitate users with ease of performing image processing tasks.

2. Working with EmguCV in Microsoft Visual C# 2008. NET

Unlike OpenCV, EmguCV offers support for dot NET compatible languages such as Visual C#, Visual Basic, Visual C++, and IronPython. EmguCV is a wrapper to the Intel® image processing library therefore it is defined and distributed over several namespace and classes. The core namespace in EmguCV is Emgu which then defined and comprise other namespaces.

Following is a chart depicting how EmguCV wraps structures, enumerations and functions of OpenCv in hierarchy:

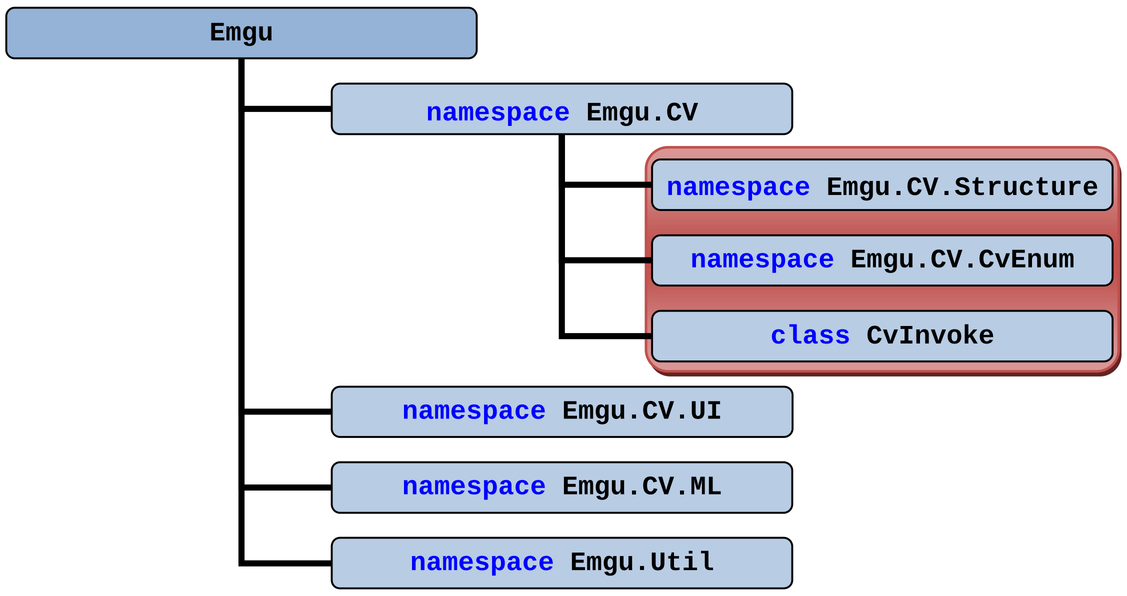


Figure 1: Core modules in EmguCV with hierarchical organization

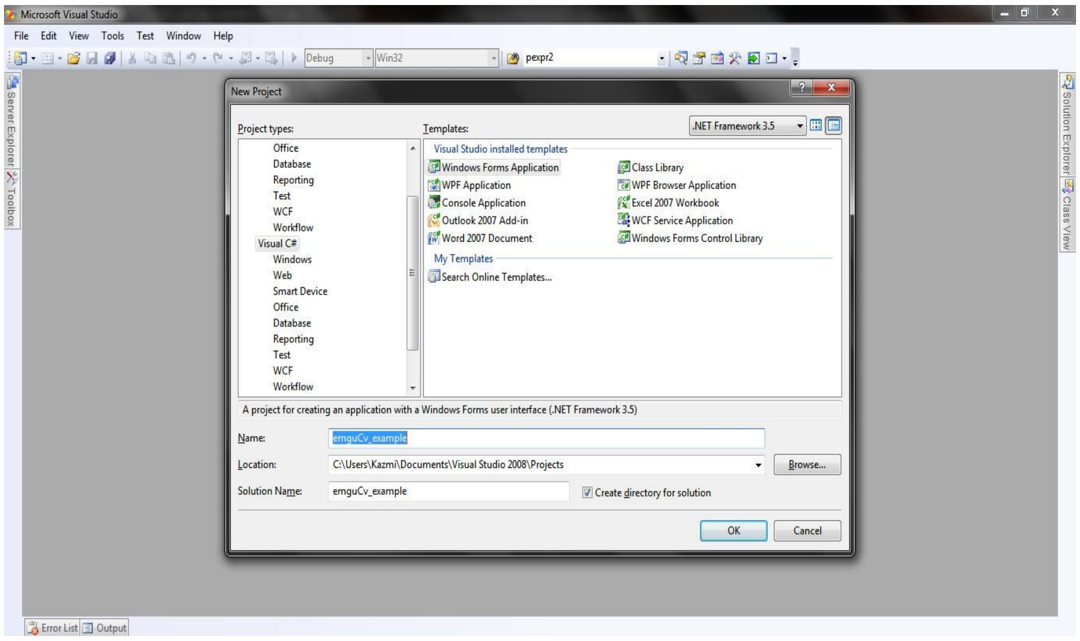
- namespaces **`Emgu.CV.Structure`** and **`Emgu.CV.Enum`** contain direct mapping to the structures and enumerations in OpenCV.
- A class **`CvInvoke`**, defined in **`Emgu.CV`** namespace, acts as a wrapper to the access the functions in OpenCV.
- **`Emgu.CV.UI`** defines user controls such as **`ImageBox`** to handle fast and robust image display.
- **`Emgu.CV.ML`** namespace defines and provides access to Machine learning features and functionality.
- **`Emgu.Util`** namespace contains classes that provide cross-platform support and other features to get an advantage of .NET technology.

2.1. Configuring EmguCV for Microsoft Visual C# 2008 dot NET:

After the OpenCV has been successfully installed and all the system PATHS are set, follow the steps as listed below:

[Step–01]: Create a New Project: go to File → New → Project.

[Step–02]: Select *Visual C#* for *Project Types* and *Windows Forms Application* under Templates. Give name to your project (e.g. *emguCv_example*) and finalize by clicking OK button, as shown below:



A project for creating an application with a Windows Forms user interface (.NET Framework 3.5)

Name: emguCv_example

Location: C:\Users\Kazmi\Documents\Visual Studio 2008\Projects

Solution Name: emguCv_example

[✓] Create directory for solution

Figure 2: Create New Visual C# Project

[Step–03]: Find *Solution Explorer* on the right side of current window; Right-Click the folder named *References* and select *Add Reference…* from menu. Select *Browse* tab from dialog box; browse to the folder **`“C:\Program Files\emgucv 2.1.0.793\bin”`** and select the following files to add:

- **`Emgu.CV.dll`**
- **`Emgu.CV.UI.dll`**
- **`Emgu.CV.Util.dll`**
- **`Emgu.CV.ML.dll`** *(add if Machine Learning features are required)*

2.2. Writing an Eample Program

[Step–01]: Find *Toolbox* on the left side of current window, select **`Button`** control from *Common Controls* and drop it on the **`Form`**. Next, Right-Click at this button and select *Properties* from context menu. Scroll to the property **`Name`** and set its value e.g. “**`btnShowImage`**”. Scroll to property **`Text`** and give the string to be display on button face e.g. **`“Show Image”`**.

[Step–02]: To keep the procedure simple just Double-Click the **`Button`** and system will direct to the *Code View* creating a **`Click`** event handler for this **`Button`**.

[Step–03]: Unlike VC++, where we include header files, in C# we add the namespaces that are supposed to be used in the program. A namespace can be included in a program with **`using`** keyword followed by name of the **`namespace`** (terminated by a semicolon "`;`"). Following is a list of namespaces required to be included in project before writing a program:

- **`using   Emgu.CV;`**
- **`using   Emgu.CV.Structure;`**
- **`using   Emgu.CV.CvEnum;`**
- **`using   Emgu.Util;`**

Finally, write the following sample code into the body of a button event handler. This will read an image from the specified path and display it once the button is clicked:

```
// name of the window that will display image
String window = "Emgu Example";

// path to image
String filename = "C:\\Users\\Pictures\\Sample Pictures\\img6.jpg";

// create window to display and image
IntPtr img = CvInvoke.cvLoadImage(filename,
                             LOAD_IMAGE_TYPE.CV_LOAD_IMAGE_ANYCOLOR);

// show image
CvInvoke.cvShowImage(window, img);

// wait for the user to press a key
CvInvoke.cvWaitKey(0);

// destory the created window
CvInvoke.cvDestroyWindow(window);
```

Resizing an Image and Saving it…

Drag and drop another **`Button`** onto the **`Form`** and write following code:

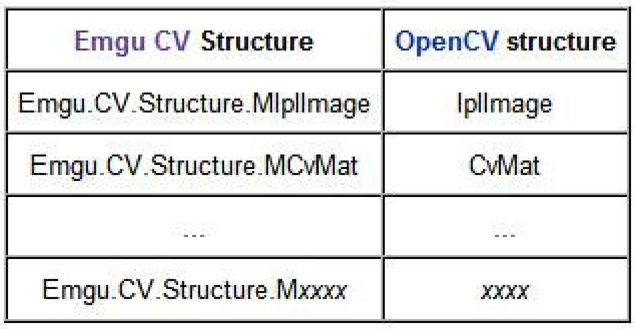
```
// create a 320x240 empty image
IntPtr new_image = CvInvoke.cvCreateImage(new Size(320, 240),
                                        IPL_DEPTH.IPL_DEPTH_8U, 3);

// resizeing image
if (image != null)
{
      CvInvoke.cvResize(image, new_image, INTER.CV_INTER_LINEAR);
      CvInvoke.cvSaveImage("D:\\new_image.jpg", new_image);
}
```

Reading/Writing Image's Pixel Value… taking image negative

```
MCvScalar value = new MCvScalar();

for (int y = 0; y < 320; y++)
    for (int x = 0; x < 240; x++)
    {
        value = CvInvoke.cvGet2D(new_image, x, y);

        value.v0 = 255 - value.v0;
        value.v1 = 255 - value.v1;
        value.v2 = 255 - value.v2;

        CvInvoke.cvSet2D(new_image, x, y, value);
    }
```

Primitive Data Types / Structures in EmguCV…

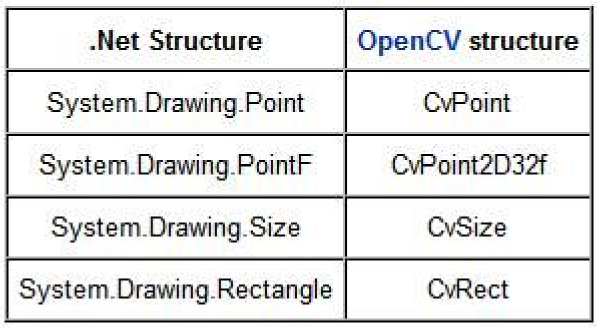
This type of structure is a direct mapping to OpenCV structures.



Emgu CV Structure	OpenCV structure
Emgu.CV.Structure.MIplImage	IplImage
Emgu.CV.Structure.MCvMat	CvMat
...	...
Emgu.CV.Structure.Mxxxx	xxxx

Figure 3: EmguCV structure vs. OpenCV structure

EmguCV used some existing structures in .NET to represent structures in OpenCV.



.Net Structure	OpenCV structure
System.Drawing.Point	CvPoint
System.Drawing.PointF	CvPoint2D32f
System.Drawing.Size	CvSize
System.Drawing.Rectangle	CvRect

Figure 4: .NET structures equivalent to OpenCV structures

References

- http://www.emgu.com/wiki/index.php/Main_Page