**Image Analysis and Interpretation (04 24086) and**

**Computer Vision (EE4H) OpenCV Laboratory Class**

**2011-12**

# Introduction

This laboratory class and exercises is designed to give you a quick introduction to using OpenCV. It is not intended to be an exhaustive explanation of all that can be done with OpenCV.

The first comments is a warning that some algorithms in OpenCV only work on certain image types. This is regrettable, especially as no warning message is given. Therefore, consider if it is reasonable to expect the operation that you wish to perform to be valid on the image that you have. For example would you expect an FFT to work on an 68 bit grey –level input image or on a 24 bit colour image. The routine will produce a floating point result image pair of phase and magnitude, likely to each be of floating point values. Thus it might be expected that the input image should be floating point.

# Creating a project

Start: Microsoft Visual Studio 2008

Select: Create Project

Select: C++ and win32 console application.

Enter project name and, if necessary adjust path to project location.

Click Ok

When Project wizard appears click: Next

Select: Empty project and un check pre-compiled headers.





## Enter IDE paths

Select: *Tools, Options* and in the window that appears select *VC++ Directories*.

On the right select: *Include files* and enter the path to the OpenCV include directory, e.g. *C:\OpenCV2.1\include\opencv*



On the rightselect: *Library files* and enter the path to the Library files, e.g. *OpenCV2.1\lib*

Likewise the path to the sources should be added.



In Solution explorer, highlight *sources*, right click and select *Add New Item*

Name new item with required source file name ending “.c”

Double left click the new file and enter the code.



Build the project by selecting *Build, Build solution*. There will be linker errors.

Select *Project Properties*

In the window that appears, on the left select *Linker, Input*

On the right select type the library list into the window of *Additional Dependencies* i.e. *cxcore210.lib c210.lib highgui210.lib cvaux210.lib*



Build the project again.

Now you can execute the program. The program in this example is designed to execute from the command line with the image file as an argument. You may wish to place the image file in the same directory as the executable so that a local reference will find the program and the file.

There are more exercises than you can complete in the scheduled lab session. Exercises 3 and 4 are open ended; aim to implement a program that attempts at least one of these operations even if you do not have time to achieve a good result.

# Exercise 1 (15 minutes)

Create a program to load a still image and display it using the example code from the lectures on OpenCV.

# Exercise 2 (1 hr 15 minutes)

Create a program that will load a video sequence. A suitable avi format file can be found in the intranet directory specified below.

The function to use is **cvCaptureFromAVI(“video.avi”)**

You will need to extract a frame from the video sequence as described in the code fragment below.

**#include <stdio.h>**

**#include <cv.h>**

**#include <highgui.h>**

**#define TRUE -1**

**#define FALSE 0**

**IplImage \*img=0;**

**int status;**

**cvCapture \*capture=0;**

**capture = cvCaptureFromAVI(“video.avi”);**

**if (capture != FALSE)**

**{status = cvGrabFrame(capture);**

 **if status != FALSE;**

 **{/\* process image \*/**

 **}**

 **else printf(“Could not grab a frame\n\7”);**

 **}**

**else printf(“Video file: video.avi does not exists”);**

This code fragment is different to that in the tutorial at www.cs.iit.edu/~agam/cs512/lect-notes/pencv-intr/opencv-intr.html. Some of the differences are highlighted in the following questions. If you cannot answer the questions ask for assistance. It is good to use material from the internet but be careful that some is not moderated and is not entirely accurate. This tutorial is useful but there is a lack of attention to good programming style.

*Questions*

1. Why is the logic test “**!= FALSE**” used ?

2. Which is strictly correct C syntax: **IplImage\* img=0; or IplImage \*img=0;**

3. Why is it good to avoid using **exit(0),** as used in the code example provide by Gady Agam ?

## Exercise 2a

Now convert the colour image to grey-level and display it in a second window

## Exercise 2 b

Having done that compute and edge image and display that in a third window.

# Exercise 3 (2 hours)

Now create a new programme and apply colour segmentation to detect skin tones. Using the video sequence from the intranet directory specified below. You may wish to use the back projection function for this. There is a simple example of the use of this function in the sample C code for **camshaftdemo.c**

# Exercise 4 (2 hours)

Implement a simple background subtraction scheme using the video sequence from the intranet directory specified below.

# Images and video clips

There are images and AVI video clips on the intranet pages under my name and the subdirectory “Multimodal”.

**Image Analysis and Interpretation (04 24086) and**

**Computer Vision (EE4H) AForge.NET Laboratory Class**

**2011-12**

# Introduction

This laboratory class and exercises is designed to give you a quick introduction to using AForge.NET. Should you wish to install AForge.NET on your PC, it is obtainable from <http://www.aforgenet.com/>. AForge is not as extensive as OpenCV but it allows you to develop your programs using C# and still has an good library of algorithms especially in the area of motion/video analysis.

The libraries are subdivided into basic image processing and application areas.

**AForge.Imaging**, which is the biggest library of the framework so far, contains different image processing routines, which are aimed to help as in image enhancement/processing, as in some computer vision tasks:

* [Linear color correction filters](http://www.aforgenet.com/framework/features/color_correction_filters.html) (RGB/HSL/YCbCr correction, brightness/contrast/saturation correction);
* [Nonlinear color correction filters](http://www.aforgenet.com/framework/features/nonlinear_color_correction_filters.html) (contrast stretch, histogram equalization, color remapping, gamma correction);
* [Image re-coloring filters](http://www.aforgenet.com/framework/features/image_recoloring_filters.html) (grayscale, sepia, hue modifier, rotate channels, invert);
* [Pixel filtering by color](http://www.aforgenet.com/framework/features/color_filtering.html) (RGB, HSL, YCbCr color spaces);
* [Color channels manipulations](http://www.aforgenet.com/framework/features/color_channels_manipulation.html) (RGB and YCbCr color spaces);
* [Color reduction](http://www.aforgenet.com/framework/features/color_reduction.html) (including color dithering);
* [Binarization filters](http://www.aforgenet.com/framework/features/binarization.html) (threshold, threshold with carry, ordered dithering, Bayer dithering, Floyd-Steinberg dithering, Burkes dithering, Jarvis-Judice-Ninke dithering, Sierra dithering, Stucki dithering);
* [Adaptive binarization](http://www.aforgenet.com/framework/features/adaptive_binarization.html) (simple image statistics, iterative thresholding, Otsu thresholding);
* [Adaptive local thresholding](http://www.aforgenet.com/framework/docs/html/0ad5b988-5613-d62a-22a9-cf41e39c139f.htm);
* [Convolution filters](http://www.aforgenet.com/framework/features/convolution_filters.html) (mean, blur, sharpen, edges, Gaussian blur, custom convolution filters);
* [Mathematical morphology filters](http://www.aforgenet.com/framework/features/morphology_filters.html) (erosion, dilatation, opening, closing, top hat, bottom hat, hit-and-miss);
* [Edge detectors](http://www.aforgenet.com/framework/features/edge_detectors_filters.html) (homogeneity, difference, sobel, canny);

Etc

**AForge.Vision** [library](http://www.aforgenet.com/framework/features/motion_detection_2.0.html) consists of different motion detection and motion processing routines.

**AForge.Math** library contains different math related algorithms, which are used as internally by the framework, as can be used as they are by users. Some of the most interesting algorithm are:

* 3D pose estimation using [*POSIT*](http://www.aforgenet.com/framework/docs/html/a7e937d4-2a3d-8bdf-5dbf-1058a3df2a60.htm) and [*Coplanar POSIT*](http://www.aforgenet.com/framework/docs/html/3110e73c-d8f4-7fda-44f5-3713ca554ddb.htm) algorithms (see [dedicated article](http://www.aforgenet.com/articles/posit/));
* [*Simple shape checker*](http://www.aforgenet.com/framework/docs/html/a4c8c190-d752-79bc-7fb6-76f1b3c3d4a7.htm) (see [dedicated article](http://www.aforgenet.com/articles/shape_checker/));
* [*Graham convex hull*](http://www.aforgenet.com/framework/docs/html/eb434e52-9781-2f51-f092-8a76db2814b9.htm) algorithm;
* etc.

**AForge.Video** library contains different classes, which provide access to video data. Nice to have it taking into account the amount of image processing stuff in the framework.

* [Access to JPEG and MJPEG streams](http://www.aforgenet.com/framework/features/jpeg_video_streams.html), which enables access to IP cameras;
* [Access to USB web cameras](http://www.aforgenet.com/framework/features/directshow_video.html), capture devices and video files through DirectShow interface;
* [Reading/writing AVI files](http://www.aforgenet.com/framework/features/avi_files.html) using Audio for Windows interface;
* [Reading/writing video files](http://www.aforgenet.com/framework/features/ffmpeg.html) using [FFmpeg library](http://www.ffmpeg.org/);
* *[Support of Microsoft Kinect sensor;](http://www.aforgenet.com/framework/docs/html/ec99e402-afc6-f187-4269-8e640ef5a26f.htm)*
* *[Support of XIMEA cameras;](http://www.aforgenet.com/framework/docs/html/fd048340-51bf-7219-39cb-886dc1866e96.htm)*
* [*Asynchronous video source*](http://www.aforgenet.com/framework/docs/html/01c875c5-3556-d700-ec3b-4023bfe81a19.htm) wrapper.

**AForge.Robotics** library contains some classes to manipulate some robotics kits:

* [Lego Mindstorm RCX](http://www.aforgenet.com/framework/features/lego_robotics.html#rcx) Robotics kit;
* [Lego Mindstorm NXT](http://www.aforgenet.com/framework/features/lego_robotics.html#nxt) Robotics kit;.
* [Qwerk](http://www.aforgenet.com/framework/features/terk_robots.html#qwer) robotics board;
* [Surveyor SRV-1 Blackfin](http://www.aforgenet.com/framework/features/surveyor_robotics.html#srv1) robot;
* [Surveyor Stereo Vision System](http://www.aforgenet.com/framework/features/surveyor_robotics.html#svs) robotics board.

**AForge.Neuro** [library](http://www.aforgenet.com/framework/features/neural_networks.html) consists of some common neural network architectures' implementations and their learning algorithms:

* Multi-layer feed forward networks utilizing activation function;
* Distance networks (Kohonen SOM, for example);
* Simple perceptron's learning, Delta rule learning, Back Propagation learning, Kohonen SOM learning, Evolutionary learning based on Genetic Algorithm;
* Activation functions (threshold, sigmoid, bipolar sigmoid).

**AForge.Genetic** [library](http://www.aforgenet.com/framework/features/genetic_algorithms.html) consists of classes aimed to solve different tasks from Genetic Algorithms (GA), Genetic Programming (GP) and Gene Expression Programming (GEP) areas:

* GA chromosomes (binary, short array, double array), GP tree based chromosome and GEP chromosome;
* Selection algorithms (elite, roulette wheel, rank);
* Common fitness functions (1/2D function optimization, symbolic regression, time series prediction).
* Population class to handle chromosomes.

**AForge.Fuzzy** [library](http://www.aforgenet.com/framework/features/fuzzy_sets.html) consists of classes to perform different fuzzy computations, starting from using basic fuzzy sets and linguistic variables and continuing with complete inference system, which is capable of running set of fuzzy rules evaluating requested fuzzy variable.

**AForge.MachineLearning** library contains some classes from machine learning area:

# Creating a project

More straightforward than OpenCV. Just create a Windows Form Application (C#) and you’re good to go! All you need to do is then need to add references to the AForge dll’s which you require. You then simply include the *using* statements in your program appropriate to the namespaces you require.







# In the following exercises

# Exercise 1 (15 minutes)

Create a program to load a still image and display it using the static method *DecodeFromFile* of the *ImageDecoder* class which is in the *AForge.Imaging.Formats* namespace. Print out the image height and width and number of bit per pixel. This information is returned as an output argument from the method.

**Exercise 2 (1 hr 15 minutes)**

Create a program that will load and display a video sequence. AForge is able to handle both avi and mpeg file formats. The following code fragments show how an event is triggered from acquiring a new frame and an associated event handler which does the processing on a frame. Each frame is returned as a Bitmap object. This can easily be displayed in a PictureBox component by setting its Image property.

private void loadVideoFromFile(String fileName)

{

 videoSource = new FileVideoSource(fileName);

// New frame event handler, which is invoked on each new available

 // video frame

videoSource.NewFrame += new NewFrameEventHandler(video\_NewFrame);

}

private void video\_NewFrame(object sender, NewFrameEventArgs eventArgs)

{

 // Get new frame

 Bitmap frame = (Bitmap)(eventArgs.Frame.Clone());

 //…. Process frame

}

# Exercise 3 (2 hours)

Implement a simple background subtraction scheme.

For test data try the CAVIAR sequence database which is an extensive database of people walking in shopping centres and is available in avi <http://www.eee.bham.ac.uk/spannm/Test%20Video%20Footage.html> and mpeg <http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/> formats.