COMP27112
Computer Graphics and Image Processing
Coursework Assignment 4
HelloCV

Dr Tim Morris

2018-2019

1 Introduction

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.[1]

This very powerful library is widely used in the image processing industry. Some of the most interesting applications include live object detection in vehicle cameras, horizon detection (which you will be doing in a later lab) in airplanes, noise reduction in medical imaging and many others.

First and foremost, let's start with the basics. In this (unmarked and unsupervised) lab exercise, you will learn how to set up an OpenCV project correctly, load an image file, create a window and display the image into it. Don't get distracted by the fact that this exercise is unmarked. You will definitely need to know all of these things throughout this part of the course unit.

2 Setting Up

You'll have created a folder for your COMP27112 coursework during earlier sessions. You'll need a subfolder in it called **cwk4**. Create it and move to it. You'll create code in this folder.

3 HelloCV

Before we dive into coding, we need to link the gcc compiler with the OpenCV library. You can create a script so that you don't have to type the command every time you need to compile.

g++ HelloCV.cpp -o HelloCV ‘pkg-config –libs –cflags opencv‘

Use your favorite text editor to create a new C++ file. (Although this is technically a C++ file, we're using the C subset only.) Along with the standard input/output library, we will need to import the <opencv2/core/core.hpp> and <opencv2/highgui/highgui.hpp> libraries. <core.hpp> provides basic OpenCV structures and operations, while <highgui.hpp> gives us the ability to create and work with windows to display information to the user.

To check that everything is working correctly, let's print out the installed OpenCV major and minor versions in the console. To do that, we have been provided with macros representing the two integers: CV_MAJOR_VERSION and CV_MINOR_VERSION.

Listing 1: C code needed to print the versions

```
1
2 #include <stdio.h>
3 #include <opencv2/core/core.hpp>
4 #include <opencv2/highgui/highgui.hpp>
5
6 int main(int argc, char *argv[])
7 {
8   //Print the OpenCV version
9   printf("OpenCV version: %d.%d\n", CV_MAJOR_VERSION,
```

```
10                                                  CV_MINOR_VERSION);
11     return 0;
12 }
```

Compile the code with the script that you created and run the binary output. You should get "OpenCV version: 2.4" printed in the terminal. If you get an error, check you've included everything correctly, check for typos, before moving on to the next parts.

4 Time to start using OpenCV

As you've read in the introduction section, we will simply load an image file into memory and display it in a window using OpenCV.

First, we will need to provide the code with an image path, either by using command-line arguments, user input in the terminal or any other way that you can think of. Make sure to check that the path has been provided and exit the program otherwise.

Now that we have a path to the image, we need to load it into memory. For this, OpenCV has a function named **imread** in the **cv** namespace which takes the path as the first argument and the image colour as the second argument. You can find all the details in the OpenCV documentation[2], but for now you will probably want to use CV_LOAD_IMAGE_COLOR as the second argument. This function loads the image into a **cv::Mat** object.

Now that we have the image (you should check that the image is loaded in the **data** member of the cv::Mat, you can look up the documentation to find out how), we need a window to display it in. Again, OpenCV (remember highgui.hpp?) has a function called **cv::namedWindow** which does this for us. This takes the window name as the first argument and the window flags as the second argument. We want the window size to match the image size, so we use CV_WINDOW_AUTOSIZE as the flag.

All that's left is to add the image to the window. **cv::imshow** does just that. The first argument is the name of the window you want to add the image to and the second argument is the pointer to the image.

At this point, if you compile and run the program you will notice that nothing shows (try it!). That is because our program terminates and closes the window before we can see anything. A useful command in this case is **cv::waitKey(0)**. The function waits for the user to press a key in order to move on with the code. The numerical argument defines how long the function will wait, look up the documentation to see how it's interpreted.

That's it. Compile and run the code. A window should pop up with the title that you gave and the image.

5 Submit

submit the code you wrote to display the image.

References

[1] OpenCV website, https://opencv.org/about.html

[2] OpenCV documentation, https://docs.opencv.org/2.4/index.html