CS-111 Programing Assignment 1: OpenCV Setup and Image Filtering

Submission instructions:

Please **submit your code and the PDF file in a single zip file** to Canvas.
You must also **submit the same PDF file** to Gradescope.
BOTH submissions are required for full points.
Your work is due by **11:59 p.m. on Wednesday, the 10th of April**.

Introduction:

Programming assignments for this course are designed for you to get hands on experience in writing image processing code. The preferred programming language is C/C++ and you will use OpenCV library for reading and writing images. You are expected to write complete, stand-alone functions that perform the desired operations on the input images.

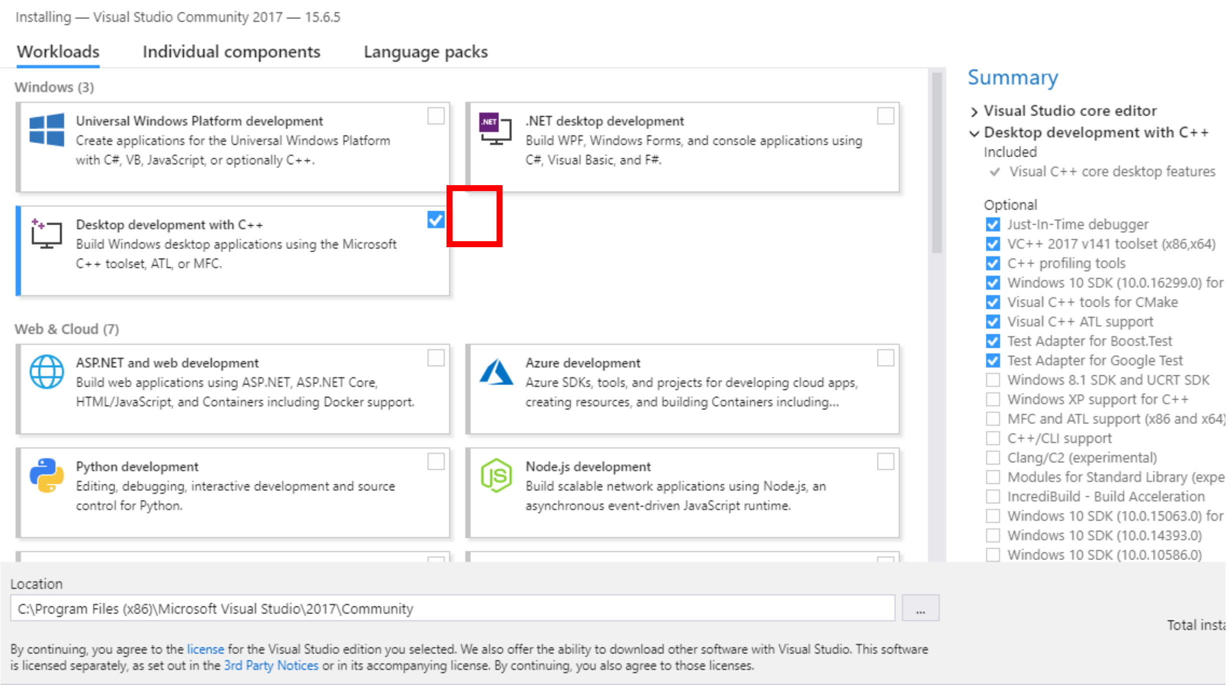
You are only allowed to use some OpenCV functions and you cannot replace the function you have to develop with its built-in functions. You will be required to submit your output image along with your function's code.

The first part of this assignment goes through the steps to set up your workspace for your programming assignments. This assignment only provides guidance for Visual Studio on Windows. If you are using other operating systems or you prefer other IDEs, please refer to online guidance for setting up your workspace. The second part of the assignment concerns image filtering.

OpenCV Installation:

I. Installing Visual Studio

a. Download free Visual Studio from https://www.visualstudio.com/free-developer-offers/ and download the Visual Studio Community.
b. Install Visual Studio and when Visual Studio Installer is up, choose only "Desktop Development with C++" and install.



Installing — Visual Studio Community 2017 — 15.6.5

Windows (3)

☐ **Universal Windows Platform development** Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

☐ **.NET desktop development** Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

☑ **Desktop development with C++** Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Web & Cloud (7)

☐ **ASP.NET and web development** Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

☐ **Azure development** Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...

☐ **Python development** Editing, debugging, interactive development and source control for Python.

☐ **Node.js development** Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Summary

- Visual Studio core editor
- Desktop development with C++

Included

- ✓ Visual C++ core desktop features

Optional

- ☑ Just-In-Time debugger
- ☑ VC++ 2017 v141 toolset (x86,x64)
- ☑ C++ profiling tools
- ☑ Windows 10 SDK (10.0.16299.0) for
- ☑ Visual C++ tools for CMake
- ☑ Visual C++ ATL support
- ☑ Test Adapter for Boost.Test
- ☑ Test Adapter for Google Test
- ☐ Windows 8.1 SDK and UCRT SDK
- ☐ Windows XP support for C++
- ☐ MFC and ATL support (x86 and x64)
- ☐ C++/CLI support
- ☐ Clang/C2 (experimental)
- ☐ Modules for Standard Library (expe
- ☐ IncrediBuild - Build Acceleration
- ☐ Windows 10 SDK (10.0.15063.0) for
- ☐ Windows 10 SDK (10.0.14393.0)
- ☐ Windows 10 SDK (10.0.10586.0)

Location

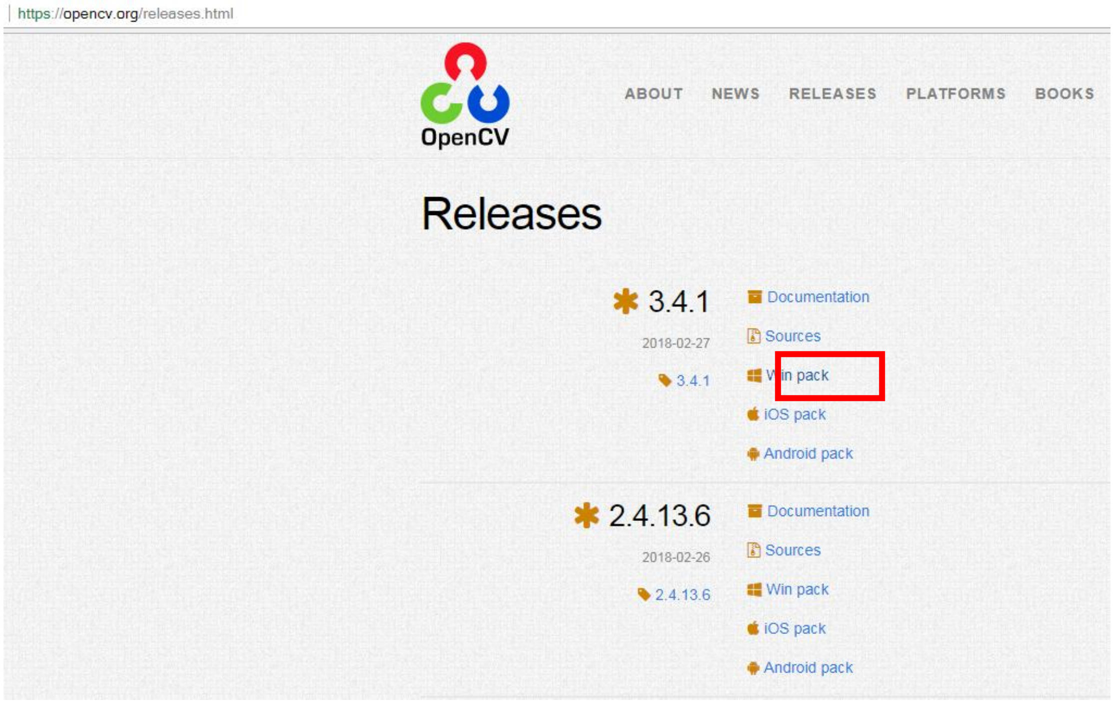
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

c. Once Visual Studio is installed, create a new "Windows Console Application" project.

II. Installing OpenCV

a. Download "Win Pack" from the latest released OpenCV version (Here 3.4.1) from https://opencv.org/releases.html



https://opencv.org/releases.html

Releases

3.4.1
2018-02-27
3.4.1

- Documentation
- Sources
- Win pack
- iOS pack
- Android pack

2.4.13.6
2018-02-26
2.4.13.6

- Documentation
- Sources
- Win pack
- iOS pack
- Android pack

b. Run the executable and unzip the content in a specific location on your computer (e.g. C:\)

c. Go to your Visual Studio project (that you created in the last step) and change the following properties:

i. In “Configuration Properties -> C/C++ -> General -> Additional Include Directories” add “<Your OpenCV location>\opencv\build\include” (e.g. “C:\opencv\build\include”)

ii. In “Configuration Properties -> Linker -> General -> Additional Library Directories” add “<Your OpenCV location>\opencv\build\x64\vc15\lib” (e.g. “C:\opencv\build\x64\vc15\lib”)

iii. In “Configuration Properties -> Linker -> Input -> Additional Dependencies” add “opencv_world341d.lib” for “Debug” configuration and “opencv_world341.lib” for “Release” configuration.

iv. Add the “<Your OpenCV location>\opencv\build\x64\vc15\bin” (e.g. “C:\opencv\build\x64\vc15\bin”) to your system “Environmental Variables” PATH. Then, restart your machine to apply the changes to the system PATH.

III. Start manipulating images

a. Now your project is ready to use OpenCV. Normally, images are color images, however, in this course we also will work on gray images (no color) for simplicity. An image is like a big matrix of the size of the image where each pixel has value that specifies the color or brightness of that pixel. Normally color images are RGB, which means each pixel has red, green and blue values. These values can have different ranges but the normal range for ordinary images are from 0 to 255, which means one byte per channel.

b. In the following code snippet, you will see how to read a color image, convert it to a gray image and save the new image. To use OpenCV commands in code, you need to include <opencv2\opencv.hpp> which contains OpenCV headers. The basic data structure in OpenCV is *Mat*, which is a matrix that can hold an image. Using the functions *imread* and *imwrite*, you can read and write images, respectively. Each pixel value can be read as OpenCV data structures *Vec3b* (for color images) or *uchar* (for grayscale images). You can create an empty *Mat* by specifying the size and the depth of each pixel. Here we use CV_8UC1 for gray images and CV_8UC3 for color images. If you have any confusion on how to use OpenCV or if you want to learn more about its features, you can read its very well defined documentations at https://docs.opencv.org/3.4.1/

```
#include "stdafx.h"
#include <opencv2\opencv.hpp>


using namespace std;
using namespace cv;

int main()
{
        Mat I = imread("alberta.jpg", CV_LOAD_IMAGE_COLOR);
        Mat J(I.rows, I.cols, CV_8UC1);
        for (int i = 0; i < I.rows; i++) {
                for (int j = 0; j < I.cols; j++) {
                        Vec3b pixel = I.at<Vec3b>(i, j);
                        int b = pixel[0];
                        int g = pixel[1];
                        int r = pixel[2];

                        J.at<uchar>(i, j) = uchar((b + g + r) / 3);
                }
        }
        imwrite("alberta2.jpg", J);
    return 0;
      }
```

c. One simple way to convert a color image to a gray image is taking average of the each red, green and blue channels, as you can see in the above code. However, our eyes are more sensitive to some colors than the others. According to **Rec.ITU-R BT.601-7** standard, the following formula should be used to correctly convert a color image to a gray image:

```
0.2990 * R + 0.5870 * G + 0.1140 * B
```

Modify the above code snippet to correctly convert the RGB image to gray image and apply it to all the color input images. Submit your code with the result images.

d. To learn more about manipulating color images, please read a color image and switch the color channels with each other.

Modify the above code snippet to replace the green with blue, red with green, and blue with red channels. Apply changes to all the color input images. Submit all result images with your code.

Image Filtering

IV. Filtering is a core operation in image processing. It has wide variety of applications ranging from video processing to computer vision. It is important to get familiar with implementing image filtering. In this section, you will learn how to apply a very simple filter on input images.

A box filter is a square filter that averages all of the image pixels covered by the filter. The output (i.e. filtered) pixel corresponds to the center pixel of the input pixels covered by the filter kernel. The filter kernel for a 3x3 box filter is:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

For each pixel in the input image, this filter adds that pixel’s value and those of its eight neighboring pixels and calculates their average. The output pixel value will be the rounded average value. You should write a function with the following signature:

```
BoxFilterGray3(Mat I);
```

which applies the above filter on all input gray images and returns the result. Read the input images as gray by calling the flag CV_LOAD_IMAGE_GRAYSCALE and then use these input images.

Implement the above function and apply the filter on all of the input images. Submit output images along with the code.