Introduction on writing computer vision programs:

For the programming tasks in this course, we will keep things simple, efficient, and convenient. Since this is not a software engineering class, we will not develop complete, user friendly applications but just implement some computer vision algorithms and play around with them. And we will do this in the plain, old C programming language. There are several reasons for this:

1. C compilers are freely available for any platform and operating system, and the tiny image reading/writing library was written for you should be compatible with all of them
2. You most likely know C already or can quickly learn it because of its similarity to Java, and C++
3. The compiled code will execute extremely fast, which will be very helpful for some transformations and machine learning approaches
4. Programming in a low-level language without specific libraries will force you to think about every elementary step in the algorithms and really understand how they work.

Reading digital Images:

Regarding the digital images, we are going to work with the simplest file format, which is called *Netpbm*. These are the files with extensions ".PBM" (black-and-white, 1 bit per pixel), ".PGM" (grayscale, 8 bits per pixel), and ".PPM" (remember RGB color, 24 bits per pixel). You can find out all about them here:
https://en.wikipedia.org/wiki/Netpbm_format

Template you will use to write your code:

The author of the book "Hands on Computer Vision" wrote a very small C library for reading, writing, creating, and deleting images in these formats, which should be easy to understand and use.

You can download the "netpbm.zip" file containing
"netpbm.c" and "netpbm.h" files, plus a sample program named "main.c" and the sample images "sample.ppm" and "text_image.ppm" from Canvas in folder "Project1"

Setting up your first project:

You should put all files into the same directory and include the C files into a project for your compiler. When you compile and run the sample program, it should load the "sample.ppm" image, produce several variants of it, and write them to the same directory. This program should be very easy to understand and will show you how to implement your own image processing and computer vision algorithms.

Notice that this library only works with the binary versions of the Netpbm formats. There are also ASCII versions of these formats, which are horribly inefficient and should be avoided. While the Netpbm format is not as common as, for instance, the JPEG or PNG formats, there is free software available for all common operating systems that can read, write, and convert Netpbm files. I recommend the cross-platform image editor GIMP (http://www.gimp.org), and for Windows also the image viewer IrfanView (http://www.irfanview.com/). For our purpose, the advantage of the Netpbm format is that it is easy to understand and only contains the information we need.

Now you can start the fun part …

Binary Image Processing

After having played around with the library and read lecture 5 in binary image processing, you should be ready for your first project. We will start with binary image processing.

1. Download the binary image "text_image.ppm" from Canvas, folder "Project1" into the same directory as the library.
2. Put all of your code, which has to use the netpbm library, in a file named "project1.c" in the same directory.
3. Please do not modify the library files. Your program should do the following things:

PART I (Due Sep 20th at 6:00M)

You will see that the image contains some text with a lot of positive and negative noise. In order to remove it:

a) Change the image to black and white using a threshold θ = 128.

   Note that the library loads binary images by assigning each pixel an intensity value of 0 (black) or 255 (white) and also expects these values (actually, using a threshold θ = 128) for when writing an image. It is implemented this way, instead of using 0 for white and 1 for black, so that all image types can be handled equally. In other words, while images are in memory, they have no PBM/PGM/PPM type assigned.

b) Save the black and white image as “black-white.pbm”
c) Implement the expanding and shrinking operations and let your program execute a sequence of them on the image.
d) Write the resulting, hopefully nicer-looking, image into a file named “image_cleaned.pbm.”

Let’s move into more interesting task...

Awesome job! You are Done. Submit Part I

Submitting Information:

- Use the code I provided
- You should have all code in a zip file
- Submit your work on Canvas.