Image Processing Challenge

Overview

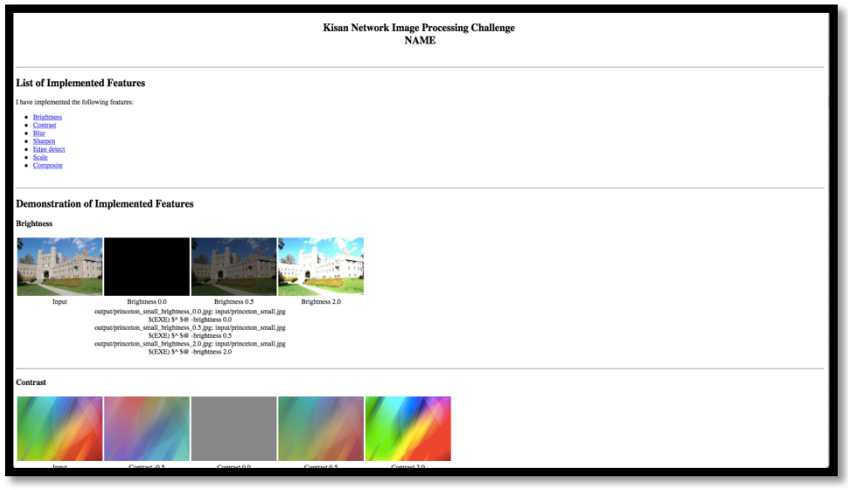
In this assignment you will create a simple image processing program. The operations that you implement will mostly be filters that take an input image, process the image, and produce an output image.

Getting Started

You should download the skeleton code from http://bit.ly/2J2uEYl as a starting point for the assignment. The zip file contains the correct directory structure for submissions, helpful source code, and a template writeup html file (see README.txt in the zip file for details). **To implement the image processing features listed above, you should add code to the appropriate functions within src/R2Image.cpp**, add commands to the NMakefile / Makefile (at root directory level; do not edit the Makefile within the src sub directory) to test them (not needed unless you are making some changes), and add sections to the writeup.html (not needed unless you are making some changes) to present and discuss your results.

How writeup.html looks when the directory is downloaded:

How writeup.html looks when the challenge has been successfully completed:



Implementing the Image Processing Filters

The following is a list of features that you need to implement (listed roughly from easiest to hardest). Refer to the http://bit.ly/2IXuKjS for more details on the implementation of each filter and example output images.

Luminance Operations
Brightness**:** Change the brightness of an image by scaling RGB values by a factor.
Contrast**:** Change the contrast of an image using the method described in http://bit.ly/2wWVtrQ.

Linear Filtering Operations
Blur: Blur an image by convolving it with a Gaussian low-pass filter.
Sharpen**:** Apply a linear sharpening filter to the image.
Edge detect: Detect edges in an image by convolving it with an edge detection kernel.

Resampling Operations
Scale**:** Scale an image up or down in resolution by a real valued factor.

Composite Operations
Composite**:** Compose one image with a second image using the ***over*** operator, with a third image as a matte (alpha).

To get full credit for any of the operations requiring resampling (scale), you must support three different sampling methods (point, linear, and Gaussian) and compare the results for at least one example. The sampling method to be used by imgpro is controlled by arguments specified before the resampling operation on the command line. For example,

imgpro in.jpg out.jpg -point_sampling -rotate 1 -gaussian_sampling rotate 3

will rotate the input image by 1 radian using point sampling and then rotate the result by 3 radians using Gaussian sampling. For Gaussian sampling, choose sensible Gaussian filter dimensions. For scaling, this requires adapting the extent of the Gaussian.

Running the Program

The image processing program, imgpro, runs on the command line. It reads an image from a file (the first program argument) and writes an image to a file (the second program argument). In between, it processes the image using the filters specified by subsequent command line arguments. For example, to add 50% random noise to an image in.jpg and save the result in the image out.jpg, you would type the following command (the noise filter has already been implemented, so you can test this command right away):

% imgpro in.jpg out.jpg -noise 0.5

For each available image filter there may be one or more optional parameters (e.g., noise takes a magnitude). To see the complete list of filters and their parameters, type:

% imgpro –help

If you specify more than one filter on the command line, they are applied in the order that they are found. For example,

% imgpro in.jpg out.jpg -contrast 1.2 -scale 0.5 0.5

would first increase the contrast of the input image by 20%, and then scale down the resolution of the result by 50% in both x and y directions. Of course, you are welcome to create your own filters and add program arguments for them by editing imgpro.cpp. However, please do not change the program arguments for the filters already provided.

Submitting

Create a .zip file containing the contents of this directory and Submit it at the Dropbox link

https://www.dropbox.com/request/40ndMMef2wwavrZwgEfm

The src directory should have all code required to compile and link your program (including the files provided with the assignment), along with a Visual Studio Solution file and a Makefile to rebuild the code. We will not attempt to grade assignments that neither build in Visual Studio nor with GNU Make under Mac OS X.

We will test your code on Linux by running make in your src/ subdirectory, followed by make in the main assignment directory to create the output images. Please ensure that your code builds/runs in this way before submitting (it will unless you alter the Makefiles in any way). Your Makefile should generate all the images in the output directory referenced by your writeup.

To save space, please submit images in .jpeg format (not .bmp or. ppm) and remove binaries and backup files from the src directory before submitting -- i.e., run make clean (under Linux or Mac OS) and execute "Clean Solution" on the "Build menu" in MS Visual Studio.

Note that you are expected to use good programming style at all times, including meaningful variable names, a comment or three describing what the code is doing, etc.

Hints

A few hints:

- Do the simplest filters first!
- Look at the http://bit.ly/2IXuKjS.
- There are functions to manipulate pixel components and pixels in R2Image. [cpp/h]. You may find them helpful while writing your filters.
