**CSci 4968 and 6270**
**Computational Vision,**
**Fall Semester, 2011-2012**
**Homework 1**
**Due: Monday, September 19, 2011, 5 pm**

## Homework Guidelines

Your homework submissions will be graded on the following criteria:

- Correctness of your solution,
- Clarity of your code, and
- Quality of your output,
- Conciseness and clarity of your explanations,
- Where appropriate, computational efficiency of your algorithms and your implementations.

Clarity of code includes the usual properties of good code: clear and easy-to-follow logic; concise, meaningful comments; good use of indentation, spacing, varable naming, and blank lines (don't insert a blank line after each statement!) to make the functions easy to read.

Conciseness and clarity of explanations is extremely important: Image data is highly variable and unpredictable. Most algorithms you implement and test will not always work particularly well, and in many cases they will work poorly. Finding the breaking points of algorithms and evaluating their causes is an important part of understanding image analysis and computer vision.

Please submit your solutions via email to `cvstewart@gmail.com`. Submit a *pdf file* containing your write-ups and image results, Matlab diaries as necessary, and your m-files as well. Do NOT submit Word files or Powerpoint files — if you use these tools, save the results to pdf. Combine everything into a zip file; if you want you can create subfolders for each problem.

**PLEASE:** include the phrase "CV HW 1" on the subject line, so that I can automatically sort your email.

## Problems

The goal of this assignment to get started working with images and with Matlab. If you choose not to use Matlab, you must adapt your solution to the environment of your choice.

1. (**10 points**) The Matlab image processing toolbox has a histogram equalization function called `histeq` (as well as a histogram function called `imhist`). Using this function, or the equivalent in a C++ toolkit, find an example image for which histogram equalization produces what you consider to be a nice output image, and find an example image for which histogram equalization does a very poor job. These images must not be artificial. In fact, if you have access to a digital camera, you should take them yourself. (Remember to convert them to grayscale using `rgb2gray` before doing so.)

   Submit the images before and after histogram equalization, as well the histograms of each image (before equalization). Explain as best you can why histogram equalization works well in the one case and poorly in the other.

2. (**20 points**) Write a function that takes two arguments as input — a grayscale image $f$ and a floating point number in the interval $[0, 0.5)$ — and produces a grayscale image $f'$ as a result. The function should compute the cdf of the image. It should then compute two values, $g_0$ and $g_1$, as follows:

   - $g_0$ is the largest grayscale $g$ such that $\text{cdf}(g) \leq r$, and
   - $g_1$ is the smallest grayscale $g$ such that $\text{cdf}(g) \geq 1 - r$.

   Finally, it should create a linearly-stretched result image:

$$f'(i,j) = \begin{cases} 0 & f(i,j) \leq g_0 \\ 255 & f(i,j) \geq g_1 \\ \frac{255(f(i,j)-g_0)}{g_1-g_0} & \text{otherwise.} \end{cases}$$

   Submit the source code, diary from running Matlab, and example results, including histograms and images. Include whatever output you need to convince me that your computation is correct.

3. (**20 points**) Write a function that computes the result of convolving an image with a Gaussian kernel of standard deviation $\sigma$. It should take the image and the value $s$ (representing $\sigma$). It should form two one-dimensional kernels, each of size $2w + 1$ where $w = \lfloor 3\sigma \rfloor$. One kernel should be vertical and the other horizontal. The function should use function `imfilter` to do the actual work, with a call to `imfilter` to apply the vertical and subsequent call to `imfilter` to apply the horizontal kernel to the result. As an intermediate result, the function should output the contents of the two kernels. Remember to convert the image to single or double precision before running through the filter.

4. (**30 points**) Building on the result of the previous problem, implement and analyze the bilateral filtering algorithm discussed at the end of the lecture notes on image processing. The input to the function should be an image and the two Gaussian values, $\sigma_d$ and $\sigma_r$ (note that here $d$ signifies "domain" and $r$ signifies "range"). The output should be the resulting image. Experiment with and analyze different values of $\sigma_d$ and $\sigma_r$, especially $\sigma_r$. (Make sure you think about what the reasonable ranges should be for $\sigma_r$.) What happens when you apply the bilateral filter repeatedly without changing the values of $\sigma_d$ and $\sigma_r$? How does this compare to applying Gaussian smoothing repeatedly using the same value of $\sigma_d$?

   Those of you programming in Matlab will find the function `nlfilter` quite useful. This is faster and easier than writing double `for` loops to pass over the entire image. There are also faster methods than `nlfilter` that you can explore.