**Problem 1 (20pts). Warm up**. Download the image folder for PS1 and choose two images and do the following: for each of these images

1.1 Load the image into your environment

1.2 Blur the image using Gaussian filter.

1.3 Display the result.

1.4 Compute the DFT (Discrete Fourier Transform) of the image. Please read Numpy FFT package. np.fft.fft2 and see here ([http://docs.scipy.org/doc/numpy/reference/routnes.fft.html](http://docs.scipy.org/doc/numpy/reference/routines.fft.html))

1.5 Display the magnitude of the DFT .

**Problem 2 (20 pts). Histogram equalization.** Compute the gray level (luminance) histogram for an image and equalize it so that the tones look better (and the image is less sensitive to exposure settings). You might want to use the following steps:

* 1. Convert the color image to luminance.
  2. Compute the histogram, the cumulative distribution, and the compensation transfer function (normalized CDF)
  3. Try to increase the “punch” in the image by ensuring that a certain fraction of pixels (say 5%) are mapped to pure black and white.
  4. Limit the local gain f’(I) in the transfer function. One way to do this is to limit

while performing the accumulation, keeping any unaccumulated values “in reserve”.

**Problem 3 (20pts) Separable filters**. Implement convolution with a separable kernel (Figure 3.14, Page 102 in http://szeliski.org/Book/). The input should be a grayscale or color image along with the horizontal and vertical kernels.

Please include examples of Gaussian filter, box filter, and Sobel filter.

Make sure you support the padding mechanisms described in class and your textbook (chapter 3.2). You will need this functionality for some of the later exercise. Please specify the kernels you used and display the original image and the output images after each horizontal and vertical operation. Here are a few padding options:

To compensate for this, a number of alternative padding or extension modes have been developed (Figure 3.13):

• zero: set all pixels outside the source image to 0 (a good choice for alpha-matted cutout images);

• constant (border color): set all pixels outside the source image to a specified border value;

• clamp (replicate or clamp to edge): repeat edge pixels indefinitely;

• (cyclic) wrap (repeat or tile): loop “around” the image in a “toroidal” configuration;

<https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.convolve.html>

You can use the optional parameter (mode)

**Problem 4 (10pts).** Write a function that finds the outline of simple objects in images (for example a square against white background) using image gradients. Your function input should be an image and output should be edges of the object. Please write your own code with Scipy/Numpy. Do not use edge detection packages in OpenCV.

**Problem 5 (10pts) Create a Github repository** of CSC589 and submit all of your code as separate .py files in a folder ProblemSet2 with clear comments.

Send your repository link in your report. Learn Github tutorial here:

<https://guides.github.com/activities/hello-world/>