# ECE 533 FALL 2015 LAB 1
# BASICS AND SPATIAL FILTERING

In the first part of this lab, we will become familiar with some basic image processing operations (reading and saving images, examining their representation, simple manipulations like rotation, translation, and resizing, display options, and saving results. In the second part, we will examine spatial filtering methods for smoothing, sharpening, and contrast enhancement.

*You may use either MATLAB or Python for this and all coursework. However, examples will be provided in MATLAB.*

## A. Basics

1. Choose a grayscale image with which you may experiment. There are several options in the Moodle "Test images" folder. Read this image into memory and store it as `im`; *e.g.,*

   ```
   im = imread('saturn.png');
   ```

   To see the result of this operation, type `whos im` at the prompt. What does this tell you about your image? How big is it?

2. Depending on the file type, your image may be stored using uint8 or doubles. (The whos command will tell you which you have.) For many of the tasks we will undertake in this class, we will need doubles; I read the image using

   ```
   im = double(imread('saturn.png'));
   ```

3. What are the minimum and maximum values in `im`?

4. Now display your image. There are two options you will use the most: `image` and `imagesc`. Look at the help files; what is the difference between these two? What does the code

   ```
   k1 = 0; k2 = 100;
   figure(3);
   imagesc(im,[k1,k2]);
   colorbar;
   colormap gray;
   axis image;
   ```

   do? Explain each line. What happens for different values of $k_1$ and $k_2$?

5. Now load a color image into Matlab, again using `imread`. What are the minimum and maximum values? What is the result of the `whos` command? Does `imagesc` work as expected? What about `image`? (HINT: You may need to try `image(im/255)` and no colormap.)

6. Your color image is stored as a stack of three images; the first contains the red content of each pixel, the second green, and the third blue. Familiarize yourself with changing the color content of the image. E.g.

```
figure(15);
subplot(221);image(imC/255);axis image;title('orig');
subplot(222);imagesc(imC(:,:,1),[0,255]);axis image;
title('red');colormap gray;
subplot(223);imagesc(imC(:,:,2),[0,255]);axis image;
title('green');colormap gray;
subplot(224);imagesc(imC(:,:,3),[0,255]);axis image;
title('blue');colormap gray;
linkaxes
load cmapRGB
figure(16);imagesc(imC(:,:,1),[0,255]);axis image;
title('red');colormap(cmap_red)
figure(17);imagesc(imC(:,:,2),[0,255]);axis image;
title('green');colormap(cmap_green)
figure(18);imagesc(imC(:,:,3),[0,255]);axis image;
title('blue');colormap(cmap_blue)
```

7. Rotate your image by 90 degrees. There are built-in functions like `imtransform` and `imrotate` with which you may experiment, but show how you would do this rotate via simple matrix manipulations as well.

8. Translate your image by 5 pixels to the left. Again, you may experiment with `imtransform`, but also demonstrate how this may be done with simple matrix manipulations.

9. Resize image so it is half as big in one direction. Again, you may experiment with `imresize`, but also demonstrate how this may be done with simple matrix manipulations.

10. Save your transformed image to a file using the `imwrite` command.

---

## B. Spatial filtering

The image smoothing filters we have discussed in class can be used to implement many of the "blur" and "sharpen" filter operations in Adobe Photoshop. For this lab, we will implement our own versions of many of these filters.

1. Implement a spatial smoothing filter. (HINT: the Matlab command `fspecial` has several useful filters you may use, and `conv2` performs two-dimensional convolution.) How does the filter performance change as you change the size of the filter? What if you leave the filter size constant but vary the values of the filter coefficients? What is the difference between a boxcar and Gaussian filter?

2. Implement a spatial edge-emphasizing filter. Again, what is the impact of different choices of filter coefficients? Does the emphasis of edges depend on the edge orientation? How is this related to the filter coefficients?

3. Apply your smoothing filter to your image, and then apply an edge-emphasizing filter to the smoothed image. Is this result the same or different than if we reversed the order of the two filter operations? Why or why not?

4. Generate a sharpening filter by combining the above smoothing and edge-emphasizing filters. HINT: this is a kind of *unsharp masking*. Explain how the same process can be implemented with a single filter.

5. Implement a median filter. (For this lab, do NOT use a built-in function like `medfilt2`. Also implement max and min filters.

6. Experiment with $\gamma$ filtering for contrast enhancement. In particular, if `im` is your original image and `im_gam` is your filtered image, then for each pixel `i` the $\gamma$ filter performs the following operation:

   ```
   im_gam[i] = im[i]^gamma;
   ```

   What is the effect of this filter? How does the effect change with the value of $\gamma$, particularly when it is less than or greater than 1? Does it help if your image pixel values are between 0 and 1?

7. Next experiment with the effect of these filters on a noisy image. To generate a noisy image, you can use the following code:

   ```
   im = double(imread('myImage.jpg'));
   sigma = 15;
   noise = randn(size(im))*sigma;
   im_noisy = im + noise;
   figure(19);
   subplot(121);imagesc(im,[min(im(:)), max(im(:))]);
   axis image;colormap gray;
   subplot(122);imagesc(im_noisy,[min(im(:)), max(im(:))]);
   axis image;colormap gray;
   linkaxes
   ```

   Now apply the filters you developed above to the noisy image. Do the filters eliminate or emphasize noise?

You should choose a test image of a scene which contains some smooth regions, some edges or boundaries, and some texture. You should then produce a report describing how you implemented each filter and showing the result of your MATLAB filter applied to your test image for each of the different filters.

---

## C. Homework problems

Solve book problems 3.15, 3.16, 3.18(a), 3.21, and 3.28.

---

## D. Deadline

Reports, including images, descriptions, and code, should be turned in via Moodle. Turn in the result of "publish" in Matlab or iPython Notebooks. Written problem solutions should also be submitted digitally. If you photograph or scan hand-written solutions, make sure they are legible. Due by 2:30pm on Sept. 14.