COSC 190, AI, Computer Vision, and Cognition
HW#3, Due Weds April 19th in class
Instructor: Dr. Fred Park

1. **Image Blur Model:** Load the cameraman.tif image in and apply 3 blurring kernels to it via convolution:

   - Disk of radius 1, 3, 5, 15
   - Motion blur of length 50 at an angle of 45 degrees as explained in class
   - A Gaussian blur of standard deviation 5.

   Plot the clean image and the blurred ones, 1 plot involving subplots for each blur. Explain the qualitative differences between them.

2. **Image Noise Model I:** We assume an additive noise model. If f is the given clean image and η the noise that is mean zero and normally distributed (i.e. Gaussian white noise). The forward degradation model is the following:

   $$f_n(x, y) = f(x, y) + \eta(x, y).$$

   Apply the noise to the 'eight.tif' image with standard deviation equaling 1,5,10, and 40. What do you notice? Why does increasing the standard deviation increase the noise? Can you think of a method to remove the noise for moderate noise levels? Code it! (Hint, look at appropriate operations in a neighborhood of a pixel).

3. **Image Noise Model II:** We assume a salt and pepper noise model. This occurs when a certain percentage of image pixels are set to either 0 or 255. This can be from information loss through wireless systems. In Matlab, you can use the following code to add salt and pepper noise to an image: `fn = imnoise(f,'salt & pepper', percent_pixels);`
   where the 'percent_pixels' is the total percent of pixels that are affected in decimal form. E.g. 0.01 = 1%. Add this type of noise to an image where 1%, 10%, and 50% of the pixels are corrupted. Can you think of a method to remove salt and pepper noise for moderate noise levels? Code it! (Hint, look at appropriate operations in a neighborhood of a pixel).

4. **Image Edge Detection via Filtering:** Create an appropriate filter that gives you better information about the jumps in an image. Use this to create an improved edge detector than from your previous assignment. Plot the original image and the detected edges. Compare to your previous method and show that a marked increase in accuracy is obtained.

5. **Shape Representation:** Can you think about how to mathematically represent the following shapes:

   - A square
   - A circle
   - A rectangle
   - A triangle
   - A non-convex shape of your choice.

   For the above, write down as much mathematical information as possible. Now, try to represent those shapes computationally in Matlab. How would you do this? Can you use points on the boundary? Or some other descriptor? Is your representation invariant under rotation, translation, and scaling? The idea of mapping a given shape to some mathematical quantity is known as a transformation. The said mathematical quantity is the shape signature. Given two square shapes where one is double the size of the other and also rotated and translated, how would you compare their shape signatures? What would be the obvious metric? What are the limitations?