The code for this homework is available at http://cs.gmu.edu/~kosecka/cs682/code/.

1. **(2) Edge detection** Test the `edge` function in MATLAB and hand in images of edge maps under two different choices of threshold and report what the thresholds are. Use the 'canny' option to get the Canny edge detector. Do `help edge` to see how to use the function. Test the function `bwlabel` for computing the connected components of the edges obtained as an output of the edge detector. This is often useful intermediate representation used in variety of problems. Show the results of using these functions for at least two threshold settings of the edge function on an image chosen from the hw3 directory.

2. **(5) Corner detection** In this problem you will use Matlab implementation of the Harris corner detector which you can download from http://cs.gmu.edu/~kosecka/cs682/code/harris
   corners.m. This includes comments on the input parameters and how to run the detector. Your task is to run this corner detector on sample images provided in the same, visualize the results and answer some questions below. Here is a set of Matlab commands demonstrating its use.

   ```matlab
   >> im = rgb2gray(imread('house1.jpg'));
   >> corners = harris_corner(im, 7,1.5);
   >> imshow(img); hold on;
   >> plot(corners(:,1), corners(:,2), 'ro');
   ```

   This reads the image from file house1.jpg, runs the corner detector using a gaussian kernel of width 7 pixels and standard deviation 1.5. The last three commands display the image and the detected corners superimposed on it.

   (a) For fixed parameter values, run the detector on house1.jpg and house1-rotated.jpg. The latter image is a rotated copy of the former. If we rotate the input image, do the detected corner position rotate by the same amount? Justify your answer based on your observations.

   (b) If we scale down the input image, are all the detected corner positions scaled accordingly? You can test this experimentally by comparing the corner detection on the images house1.jpg, house1-2down.jpg, house1-4down.jpg. Each image in this sequence is half the size of its predecessor. Justify your answer based on your observations.

3. **Correspondences.** The problem of establishing correspondences between points detected in two different views of the same object is at heart of many important problems and applications in Computer Vision. The difficulty of the problem is emphasized by the type of changes the object/camera undergo between two views. This exercise will let you examine some approaches to the problem.

   Use the Harris feature corner code from the previous exercise, select the features in the first image and find the corresponding points in the second image. This you can do either by selecting feature points in the second image or examining all locations in the second image. One common way how to measure the similarity between two feature points is to take a
window of a particular size centered at the respective locations and computed so called SSD
sum-of-squared-differences similarity measure between two windows defined as follows

\[
SSD(w_1, w_2) = \sum_{x_1 \in w_1, x_2 \in w_2} (I(x_1) - I(x_2))^2
\]

Implement function SSD which takes two images (windows in this size and returns the similari-
ty score between them.

(a) Using this function to find correspondences between pairs of images cathedral1.bmp
    and cathedral4.bmp and visualize the matches found by your algorithm. Use function
    \texttt{rgb2gray} to convert color images to grayscale.

(b) Repeat the same exercise for SIFT detector, descriptor and SIFT matching procedure,
    which you can download from \url{www.vlfeat.org} and hand again the correspondences for
    the image pair.

(c) Using the SIFT features, design and describe an algorithm for finding the object given
    in the query image \texttt{query.JPG} in the database images \texttt{db*.JPG} and show the results and
    describe the algorithm.

Submit the SSD correspondences code and results of correspondences/matching algorithms
and comment on the difference between the two strategies, which one is better for which pair
and the description of the matching strategy for the last problem. You can also try other
feature detector(descriptor pairs which are available in vlfeat library. The tutorials how you
use the code are provided on the vlfeat website.

Note: To visualize the result make a new image putting the two images side-by-side and
connect the corresponding features by plotting lines originating in one view and finishing in
another. This is done easiest in Matlab in 3 lines of code. Function \texttt{appendimages.m} to make
a composite image is available in the code directory on cs682/code web site.