Exploiting visual redundancy in images for image enhancement

Goal:

The goal of the project is to exploit the redundancy of visual data within images for the purpose of image enhancement (in this project - for image denoising).

Description:

The underline assumption is that small image patches (e.g. 7x7) tend to repeat themselves within natural images. Therefore, for each patch in the image we will look for similar patches elsewhere in the image, and integrate their visual information to remove the noise (the common visual information will be preserved, and the uncorrelated noise will be removed).

Details

1. Your program should do the following:

    1. Given a noisy image
    2. For each pixel take its surrounding patch (of size h x w) and measure its distance with respect to all the other patches in the image. Use Sum-of-Square distances (SSD) as a distance measure
    3. Find k "nearest neighbors" (i.e. k most similar patches).
    4. For each of these "nearest" patches assign weight = EXP(-distance/sigma^2)
    5. New pixel value = weighted average of the center pixels of the k "nearest" patches (the average should include the pixel of the original patch as well).
    6. Iterate steps 2-5 with the new improved image (but each time search for k "nearest" patches in the original noisy input image).

2. Write a function called "denoise.m" that performs the following:

    - Input:
        - noisy image
        - sigma value

    - number of nearest neighbors (k)
    - size of the patch -- two dimensional vector [h w]
- Output:
    - the denoised image

3. Apply your function on the following gray images: clown, car and color images: clown_color, car_color

4. For finding k nearest neighbors (patches) you can use this code (sorry, it is a bit slow... By the way ANN = approximate nearest neighbor)
To use the code just unzip the file, you will get a folder '@ann' and file ann_class_compile.m, run the following commands in Matlab:

```
>> mex -setup % setup your local matlab compiler
>> ann_class_compile
```

You will need to use functions 'ksearch' and 'ann'.

Notes:

- Use 5 iterations.
- You can use matlab's 'im2col' to produce the patches.
- The program should work BOTH for gray-scale and color images. (Note that color images have 3 channels whereas gray scale images have 1 channel).
- In your report indicate:
    - What is the best sigma to use?
    - How many nearest neighbors are needed?
    - What is a good patch size?
- Compare the results to simple noise cleaning with a **gaussian** filter and to noise cleaning with a **median** filter. Use the same window size (h x w) for the comparison.

The function prototype:

```
denoised_image = denoise(im,sigma,num_nn,patchsize);
```

Code Submission: The source code should be a one-file function named "denoise" (do not send multiple files). Note that many functions can be included in one MATLAB m-file, each function starting with the key-word "function". The function denoise must be the first function in the file, which should be named "denoise.m"

Good Luck!