SD 575 Image Processing

Fall 2023

Lab 4 - Restoration (Python)
Due Friday November 24 at 11:59 pm

Note: All lab reports will be submitted on Learn to group dropboxes.

For help on how to use the Python functions mentioned throughout the lab, please type `help` followed by the function (e.g., `help imread`).

NOTE: if you are using jupyter notebooks you may need to install a component for the Adaptive Filtering section. In your environment, run *pip install ipympl*

1 Overview

The goal of this lab is to provide some hands-on experience with image restoration concepts in frequency domain.

For the first part of this lab, we will study how to enhance images in frequency domain. The following images will be used for testing purposes:

- cameraman.tif
- degraded.tif

These images can be found on Learn.

2 Image Restoration in the Frequency Domain

Let us now study two frequency domain based image restoration methods: inverse filtering and Wiener filtering. Load the Camerman image (adjust intensities to range of 0 to 1). First, create a disk blur function of radius of 4 and apply it to the image in the frequency domain:

```
h_d = disk(4)
h = np.zeros((256,256))
h[0:9,0:9] = h_d / np.sum(h_d)
```

```
h = np.roll(h, (-5,-5))
h_freq = np.fft.fft2(h)
f_blurfreq = h_freq*np.fft.fft2(f)
f_blur = np.real(np.fft.ifft2(f_blurfreq))
```

Plot the blurred image and the corresponding PSNR. Now apply inverse filtering to it by dividing the image by the blurring function `h_freq` and plot the result.

1. Compare the restored image with the original image and the blurred image. How does the restored image and the PSNR differ from the blurred image? Is it better or worse? Why?

Now add zero-mean Gaussian noise with a variance of 0.002 to the blurred image using *skimage.util.random_noise*. Apply inverse filtering to the noisy blurred image. Plot the restored image and the PSNR.

2. Compare the restored image with the restored image from the previous step. How does the restored image and the PSNR differ from the previous restored image? Is it better or worse? Why?
3. Can you draw any conclusions about inverse filtering when applied to noise degraded images?

Let us study the Wiener filter for image restoration. Apply Wiener filtering on the noisy blurred image used in the previous step. Use the `restoration.wiener` function and pass in the disk blur function as the point-spread function (PSF) and an appropriate approximation of noise-to-signal ratio (NSR). Note, that `restoration.wiener` expects the PSF to be centered, so be sure to do this with `np.fft.fftshift`.

4. Compare the restored image with the restored image from the previous step. How does the restored image and the PSNR differ from the previous restored image? Is it better or worse? Why? Explain it in context with the concept behind Wiener filtering.
5. Can you draw any conclusions about Wiener filtering when applied to noise degraded images?

3 Adaptive Filtering

Linear shift-invariant (LSI) filters do not take into account any structure in the image; the degree of smoothing is the same over all parts of an image. It is known, however, that people are less sensitive to noise in regions of high image detail.

Lee developed a filter which adapts to the detail in an image. The form of the filter is

$$\hat{f}(i,j) = K(i,j)g(i,j) + (1 - K(i,j))\overline{g(i,j)}$$

$$K(i,j) = \frac{\sigma_g^2(i,j) - \sigma_{noise}^2}{\sigma_g^2(i,j)}$$

where $\overline{g(i,j)}$ and $\sigma_g^2(i,j)$ are the mean and variance of the intensity levels of the pixels in the $m \times m$ neighborhood of pixel (i,j), respectively.

Let us now study noise reduction using Lee filter. Load `degraded.tif` and adjust intensities between $[0,1]$. Note that this filter requires an estimate of the noise variance. In class, you learned a method to estimate the noise variance. In this lab, an easier empirical method is presented. One way to estimate the noise variance is to capture a "flat" region of the image and calculate the variance of the intensity levels.

6. Use a flat region of `degraded.tif` and estimate the variance of the noise (Use the provided Selector class, see the example usage in selector_ex.py/selector_ex.ipynb).

Use the following code to compute the local mean and variance of the image on the 5×5 neighborhood of each pixel:

```
                    mn = np.ones((5,5))/25
        local_mean = signal.convolve(I, mn, mode='same')
local_var = signal.convolve(I**2, mn, mode='same') - local_mean**2
```

where, `local_mean(i,j)` and `local_var(i,j)` are $\overline{g(i,j)}$ and $\sigma_g^2(i,j)$, respectively. Now, you can compute elements of matrix K and denoise the image using Lee formulation. Plot the original and denoised image, with PSNR.

7. Compare this result to that using a Gaussian low pass filter with standard deviation of 30 (in the frequency domain). Note the performance in areas of high and low detail.
8. Try varying your estimate of the noise variance both above and below the value you got from your flat region. How does this change the filter's results? Why?
9. Try changing the size of the filter neighborhood to be smaller and larger than 5×5. How does this change the results? Why?

4 Report

Include in your report:

- A brief introduction.
- Pertinent graphs and images (properly labelled).
- Include code (can be included in appendix).
- Include responses to all questions. Enumerate your answers to each question (e.g., 1., 2. or Q1, Q2, etc.).
- A brief summary of your results with conclusions.