Deconvolution and Texture Analysis
Signal and Image Processing

March 22, 2018

1. Inverse filtering

1.1. Write a program that takes an image, a kernel and a realisation of a noise source, and which returns the linear, shift invariantly (LSI) degraded result.

1.2. Implement direct inverse filtering and apply it to a LSI degraded image of `trui.png` (you are suppose to implement this on your own - not use functions from skimage). Experiment with different noise levels. Discuss this approach's ability to recover the original image.

1.3. Repeat the above exercise by implementing the Wiener filter - eq. (5.8-6) in Gonzales and Woods (you are suppose to implement this on your own - not use functions from skimage). Experiment with different settings of the constant K approximating the fraction of noise and signal power spectra in eq. (5.8-2). Also experiment with different noise levels. Discuss this approach's ability to recover the original image.

2. Texture analysis

2.1. Compute the intensity histograms of the six texture images `canvas1-a-p001.png`, `cushion1-a-p001.png`, `linseeds1-a-p001.png`, `sand1-a-p001.png`, `seat2-a-p001.png`, and `stone1-a-p001.png` and visualise these in the report. Also compute the χ^2-distance between each pair of the six histograms and report them in a table. The χ^2-distance between two normalized n-bin histograms $\mathbf{h} = (h_1, \ldots, h_n)$ and $\mathbf{g} = (g_1, \ldots, g_n)$, with $\sum_{i=1}^n h_i = 1$ and $\sum_{i=1}^n g_i = 1$, is given by this expression

$$\chi^2(\mathbf{h}, \mathbf{g}) = \sum_{i=1}^{n} \frac{(h_i - g_i)^2}{h_i + g_i} \ .$$

(Notice that you have to handle the case where both bins in the two histograms are zero.)

Is it possible to distinguish the six texture classes using intensity histograms as features of the texture?

2.2. We will implement a modified version of the texture analysis approach proposed by Varma and Zisserman [1] using the MR8 filter bank (Maximum Response 8 filters).

i. Implement the MR8 filter bank consisting of the oriented Gaussian derivative filters up to order 2 including the zeroth order term (the Gaussian

filter itself) and the Laplacian filter. The MR8 filter bank assigns 8 filter responses to each pixel in the processed image, but is contains more than 8 filters. The filter responses is reduced to eight by picking maximum responses over different orientations of the oriented derivative filters.
We will use the convention that a Gaussian filter $G(x, y; \sigma)$ applied to an image I leads to the filter response image

$$L(x, y; \sigma) = (I * G)(x, y; \sigma)$$

and we will denote the responses to Gaussian derivative filter of order $n+m$ as

$$L_{x^n y^m}(x, y; \sigma) = (I * G_{x^n y^m})(x, y; \sigma)$$

Also recall that we can compute the oriented 1st and 2nd order derivatives (oriented by the angle θ) with

$$\begin{aligned} L_\theta(x, y; \sigma) &= \cos(\theta) L_x(x, y; \sigma) + \sin(\theta) L_y(x, y; \sigma) \\ L_{\theta\theta}(x, y; \sigma) &= \cos^2(\theta) L_{xx}(x, y; \sigma) + 2\cos(\theta)\sin(\theta) L_{xy}(x, y; \sigma) + \sin^2(\theta) L_{yy}(x, y; \sigma) \end{aligned}$$

The filter responses of the filter bank applied to an image I is given by

$$\begin{aligned} L(x, y; \sigma) & & (1) \\ \nabla^2 L(x, y; \sigma) &= L_{xx}(x, y; \sigma) + L_{yy}(x, y; \sigma) & (2) \\ \max_\theta L_\theta(x, y; \sigma) & & (3) \\ \max_\theta L_{\theta\theta}(x, y; \sigma) & & (4) \end{aligned}$$

Furthermore, each derivative filter should be computed at 3 different scales (i.e. values of the parameter σ). We suggest that you use $\sigma = 10$ pixels for $L(x, y; \sigma)$ and $\nabla^2 L(x, y; \sigma)$ and the values $\sigma = 1, 2$, and 4 pixels for the 1st and 2nd order oriented derivative filters. This leads to that the filter bank returns in total 8 filter responses. Finally we need to choose a set of orientations and here we suggest you use these angles $\theta = 0, \pi/6, 2\pi/6, 3\pi/6, 4\pi/6$, and $5\pi/6$. This means that the two maximization operators in (3) and (4) should be taken over the filter responses for the 6 angles.
Apply your filter bank implementation to the `linseeds1-a-p001.png` texture image and illustrate the 8 filter respons images for the filter bank.

ii. Next we will learn a set of textons to describe the six texture classes given by the training images `canvas1-a-p001.png`, `cushion1-a-p001.png`, `linseeds1-a-p001.png`, `sand1-a-p001.png`, `seat2-a-p001.png`, and `stone1-a-p001.png`.
Learning textons from the 8 filter responses of our filter bank can be thought of as finding clusters in the space of the 8 filter response vectors. For each of the six images apply the filter bank. At each pixel you can now form an 8 dimensional vector of the filter responses. Collect these vectors for all pixels in the six images - this form your training dataset. Apply the K-means algorithm with $K = 60$ (10 clustes per texture class, in our case we have 6 classes) to the training set of filter response vectors. You may use `sklearn.cluster.KMeans` or any other implementation of the K-means algorithm.

iii. Compute texton histograms for each of the six texture images. That is, give each of the textons learnt in the previous question a unique label in the form of an integer. Next apply the filter bank to an image obtaining 8 filter responses (a vector) for each pixel. For each pixel find the texton (K-means cluster center) that the 8 filter response vector is closest to and assign the pixel this texton label. Now compute a histogram of occurrences of the texton labels in the image.
Compute the χ^2-distance between each pair of the six texton histograms and report them in a table. Show the histograms in the report and discuss if you think it is possible to discern between the six textures based on these histograms.

iv. Compute the texton histogram of the image `cushion1-a-p012.png` and find the closest matching texton histogram among the six training images using the χ^2-distance. Do you get the correct texture class?

References

[1] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1/2):61–81, April 2005.