Signal processing
Lab 2: Linear denoising

Wednesday 18/02/2015

Abstract

The objective of this Lab is to get familiar with processing images with matlab and implement a simple image denoising technique. We will first add noise to an image using a gaussian noise model of standard deviation σ_n. Then we will find a reconstruction of the original image using a gaussian filter of standard deviation σ_r. Finally, we will search for the best σ_r to use for a given value of σ_n.

Deliverable: To be submitted before Wednesday 04/03/2015.

1 Noise model

Remember that you can get information about Matlab commands and their parameters by typing `help` ⟨`command`⟩ in Matlab Command Window.

You will find images in the **images** folder. A starting code, which already implements part of what follows, is given: you will have to complete that code.

In image denoising, we are given an image $\mathbf{y}$ which is a noisy observation of a "ground truth" image $\mathbf{x_0}$. The pixels sensors on a camera can for instance locally perturb the image because of physical processes. Image compression can also introduce noise into an image. The goal of image denoising is to reconstruct an image $\mathbf{x}$ from $\mathbf{y}$ that recovers the original image $\mathbf{x_0}$ as closely as possible, removing the effect of the noise.

In this work, we artificially introduce noise to an image $\mathbf{x_0}$, and then try to recover this image. This way, one may study the performance of our denoising algorithm for a particular noise model.

The noise model that we will use is one of the simplest possible: a additive gaussian model, also called *white noise*. According to this model, the noise perturbs the original image $\mathbf{x_0}$ by adding a random variable to each of its pixels

$$x^{i,j} = x_0^{i,j} + \xi^{i,j}$$

where the random variables are independent and follow a centered normal law

$$p(\xi^{i,j}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\xi^{i,j})^2}{2\sigma_n^2}} .$$

The only parameter of the noise model is the standard deviation of the normal law, σ_n.

In Matlab, one may simulate a additive gaussian noise model by using the function `randn` which returns an array of centered normal variables with unit variance. The line

$$\mathtt{y} = \mathtt{x0} + \mathtt{sigma\_n} * \mathtt{randn}(\mathtt{size}(\mathtt{x0}));$$

adds the gaussian noise to $x0$, scaling the output of `randn` so that the noise has the desired standard deviation, σ_n.

The beginning of the code, already implemented, loads the image **lena.tiff** into the array x_0 using Matlab's `imread` command and converts it to grayscale. The images are then shown using the `imshow` command.

Question 1.1. For a noise setting $\sigma_n = 0.08$, include the ground truth image $\mathbf{x_0}$ and the noisy image $\mathbf{y}$, in your report.

2 Linear denoising

The goal is to recover an image $\mathbf{x}$, close to the ground truth $\mathbf{x_0}$, from the noisy image $\mathbf{y}$. In what follows, we will consider a translation invariant linear denoising operator. As seen in the course, such an operator is parametrized by a convolution of the input with a kernel $\mathbf{h}$

$$\mathbf{x} = \mathbf{x_0} \star \mathbf{h}$$

and in our case, we will use a gaussian kernel with bandwidth (standard deviation) σ_r. We will compute 2-D convolution using Matlab's imfilter command.

The first step is to implement a gaussian kernel. In its continuous form, this kernel is simply a 2-dimensional centered gaussian

$$h(\mathbf{x}) = a \exp\left(-\frac{\mathbf{x}^2}{2{\sigma_r}^2}\right)$$

where σ_r is the standard deviation of the reconstructing filter and a is a normalization constant. In our case, the filter has to be discretized to an array of pixels.

Question 2.1. Write a function `h = gauss_kernel(sigma_r)` that returns a gaussian filter $\mathbf{h}$. In order to include any significant values in the output array, the filter window should contain all the values that lie within 3 standard deviations of the center. Visualize the output with $\sigma_r = 3$ pixels using the command `imagesc` (include the image in your report). If your code is right, the output should look similar to Figure 1.

Question 2.2. Use the function `imfilter` to compute the estimate $\mathbf{x} = \mathbf{x_0} \star \mathbf{h}$. What is the effect of `imfilter`'s optional third parameter? Include the image $\mathbf{x}$ in your report. Figure 2 shows the results you should obtain.

A quantitative measure of the denoising performance is given by the signal-to-noise (or SNR) ratio. For general signals, this ratio is given by

$$\mathrm{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}}$$

and often given in a decibel scale

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \cdot \log_{10} \mathrm{SNR}.$$

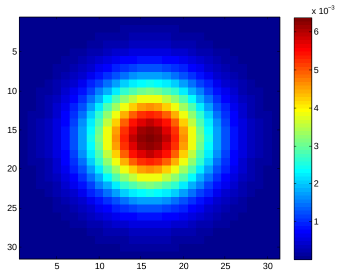

Figure 1: A gaussian kernel with standard deviation $\sigma_r = 3$.

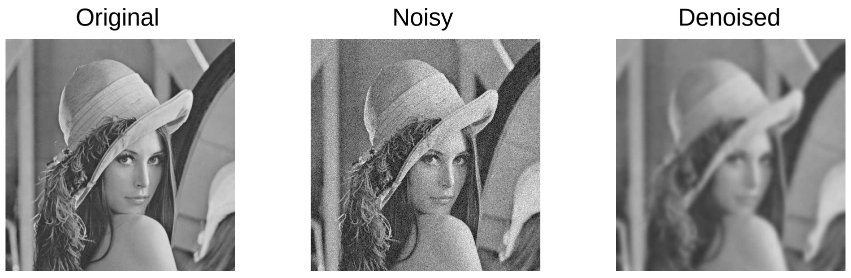

Figure 2: The denoising process using $\sigma_n = 0.08$ and $\sigma_r = 3$.

In the particular context of images, the SNR ratio becomes the PSNR ratio. In our case, this ratio is computed as

$$\text{PSNR} = -10 \cdot \log_{10}(\text{MSE})$$

where MSE is the average mean square error between the pixels of $\mathbf{x}$ and $\mathbf{x_0}$; if the images have size $n_x \times n_y$

$$\text{MSE} = \frac{1}{n_x n_y} \sum_{i,j} (x^{i,j} - x_0^{i,j})^2.$$

Note that in this study we built $\mathbf{y}$ from $\mathbf{x_0}$, and therefore we can compute the $PSNR$. In a real denoising application, such as the a denoising function on a phone's camera, one would not have access to $\mathbf{x_0}$.

Question 2.3. Compute the PSNR of the noisy image $\mathbf{y}$ and the reconstructed image $\mathbf{x}$ with regard to the original image $\mathbf{x_0}$. Report thes values. *Remark: be careful to use the `log10` function.*

Question 2.4. Does the denoising process removes the noise from the original image? Do we recover all the information from the original image?

3 Finding the best filter

In what follows, we are interested in finding the value of the bandwidth of the denoising filter, σ_r, giving the best signal-to-noise ratio, thus recovering the best approximation of the ground truth x_0. This optimal value will of course depend on the input noise level σ_n.

Question 3.1 Using calculations similar to Section 2, write a function

$$\texttt{PSNR} = \texttt{compute\_PSNR}(\texttt{x0}, \texttt{y}, \texttt{sigma\_r})$$

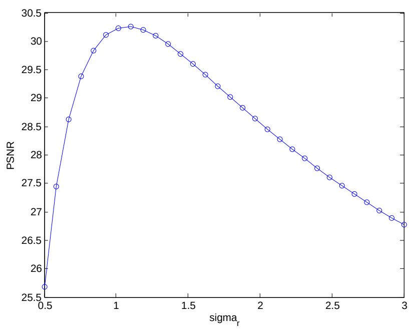

Figure 3: PSNR as a function of $\sigma_r = 3$ for an input noise $\sigma_n = 0.08$.

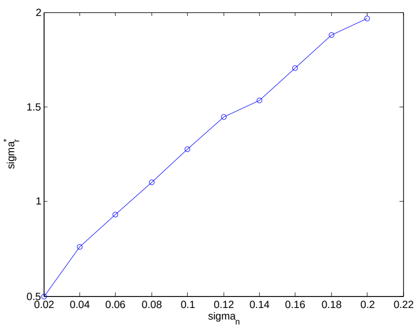

Figure 4: Optimal reconstructing bandwidth σ_r^* for different values of the input noise σ_n.

which computes the PSNR of a reconstruction $\mathbf{x}_{\sigma_r}$ computed from $\mathbf{y}$ using a gaussian filter of bandwidth σ_r.

Question 3.2 Use the code provided to draw a plot of the PSNR as a function of σ_r, for 30 values ranging from 0.5 to 3. If your code is correct, you should obtain the result shown on Figure 3.

Question 3.3 Among the values of σ_r considered, find the one that gives the best PSNR, σ_r^*. You will use matlab function `max` which returns the maximum of an array, and its index as an optional second argument. Report the value along with the best PSNR, and the image reconstructed at this optimal PSNR.

Question 3.4 Fill in function `sigma_r_star = best_sigma_r(x0, sigma_n)` which encapsulates this procedure: it takes an image $\mathbf{x_0}$ as input, adds gaussian noise with squared variance σ_n, and finds the best reconstructing bandwidth σ_r^*. Using the provided code, draw a plot of σ_r^* as a function of σ_n. Include this plot in your report; it should be similar to Figure 4.

Question 3.5 How does linear gaussian image denoising perform in recovering the ground truth images? Is the noise removed, do we recover the information of the ground truth ? Optionally, compare the denoising results with results performance on a few other images, among the one included in the `images` folder, or other images found on the Internet (less than 3 other images).