Image Enhancement Report

EQ2330 Image and Video Processing, Project 1

Harald Nordgren
haraldnordgren@gmail.com

November 25, 2015

Summary

I have investigated spatial and frequency domain image enhancement techniques in Matlab on the 512x512 demo image "Lena".

1 Introduction

This project was about restoring images the were degraded using three types of techniques.

The first sub-assigment deals with the dynamic range of greyscale images where intensity values are found in the range [0,255]. Looping over each pixel value and summing the pixel values into bins (one bin for all intensity values 0, one for 1, and so on) creates a histogram. A normal-looking image will have a histogram that uses the whole dynamic range and spreads somewhat evenly.

Using the function

$$g(x,y) = min(max(\lfloor 0.2 \cdot f(x,y) + 50 \rceil, 0), 255) \tag{1}$$

the dynamic range can be lowered but squeezing the values together, giving a duller-looking image. Assuming an image that for some reason starts out with a small dynamic range, histogram equalization can stretch the dynamic range to the entire 8-bit spectrum, giving a more lively apperance. The cumulative sum of the normalized histogram for the low-contrast image can be calculated as follows

$$s_k = 255 \sum_{j=0}^{k} p_r(r_j) \tag{2}$$

where the dynamic range the dynamic range is stretched to [0,255]. p is the probability for each intensity value in the original image, and the sum for p over the entire numerical axis is 1 by definition. Mapping the intensity values of the low-contrast image to these summed values equalizes its histogram.

In the second sub-assignment, the `mynoisegen` function generates two types of noise that are applied to the original images, and I then attempt to recover it. The Gaussian noise is additive, after which I requnatize the image to bring it back to the [0,255] interval. The salt-and-pepper noise sets certain random pixel values to 0 or 255, which represent black and white, respectively giving the impression that the image has been salted and peppered.

The attempts to recover the noised image realy of two very similar methods. The mean filter uses a kernel of a certain size (here a 3x3 matrix with every value equal to 1/9) which is convoluted with the noised image, effectively averaging each pixel value with its closest neighbors. The median filter replaces each pixel with the median of the neighboring pixels, meaning that these values have to be sorted first. Both methods remove high-frequency noise alongside legitimate information from the image.

In the final sub-assignment, the image is blurred by convoluting with a Gauss kernel generated by `myblurgen` and then quantized to give additive noise, modeled as

$$f(x, y) = g(x, y) * h(x, y) + \eta(x, y) \tag{3}$$

The task is then to deblur the image while only using the blurring function and the variance of the noise.

2 System Description

In the first sub-assignment, I plotted the histogram for the three versions of the demo image, using `subplot`. To calculate histograms all matrices are first converted to an array vector with `(:)` and divided each value by the total sum before plotting as a bar graph.

To lower the contrast I iterated over each pixel and calculate (1), and to equalize the histogram I used `cumsum` and then mapped the low-contrast values to the summed function.

I used mynoisegen to add Gaussian and Salt-and-pepper noise to the original, and then tried removing it with mean and median filtering. The mean filter replaces each value with the mean of the surrounding values. The median replaces it with the median (that is, sorting the values by size and choosing the middle one). The mean filter is quicker to calculate – no sorting is needed – but the median filter is likely to give slightly better results when dealing with outlier noise, which is why it gives such a good performance for the salt-and-pepper noise.

In the last task, I applied Gaussian blur by convolution the original images with a 8x8 kernel generated by myblurgen, and quantized the this image to 8-bit – giving additive noise. Given the symmetry of the blur kernel (leading to a zero determinant) it is obvious that a simple inverse filter would not do. Using the variance of the quantization noise and the blurring function, I used `deconvwnr` to deblur the image.

To avoid ringings along the border of the image, I had to first run `edgetaper`. I then plotted original, degenerated and deblurred images, along with their fourier spectra.

3 Results

For a continuous probability density function we expect equalization to give a completely flat histogram, but for out quantized data it will show slight deviations from that but will be as close as possible.

4 Conclusions

Present your conclusions in this section. Remember that conclusions are not just another summary. Your report, excluding references and appendix, should fit in 4-5 A4-pages. Therefore, make sure to write concisely and to the point, describing everything of importance. Writing a report takes time, which is why you should start early. If you have any questions about the assignment

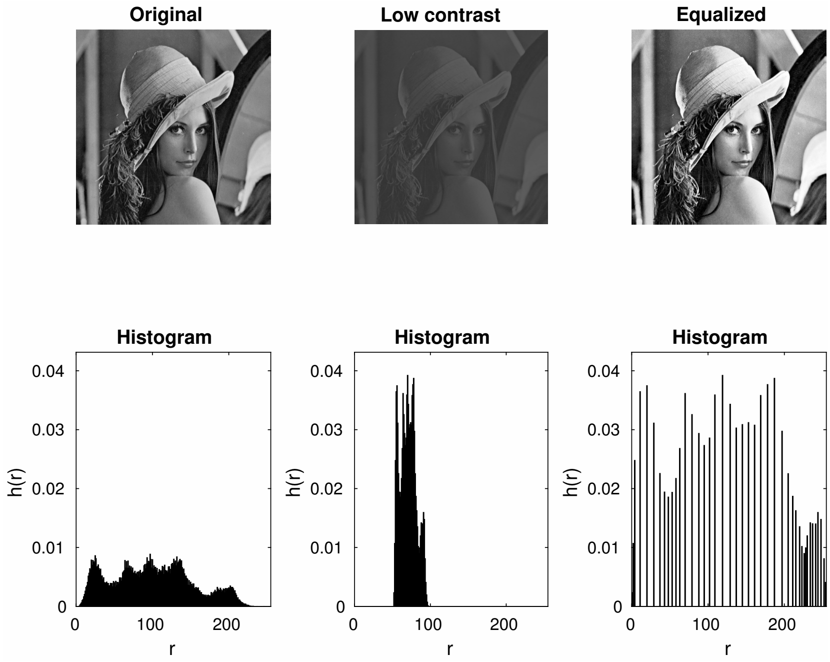

Figure 1: Sub-assigment 2.1

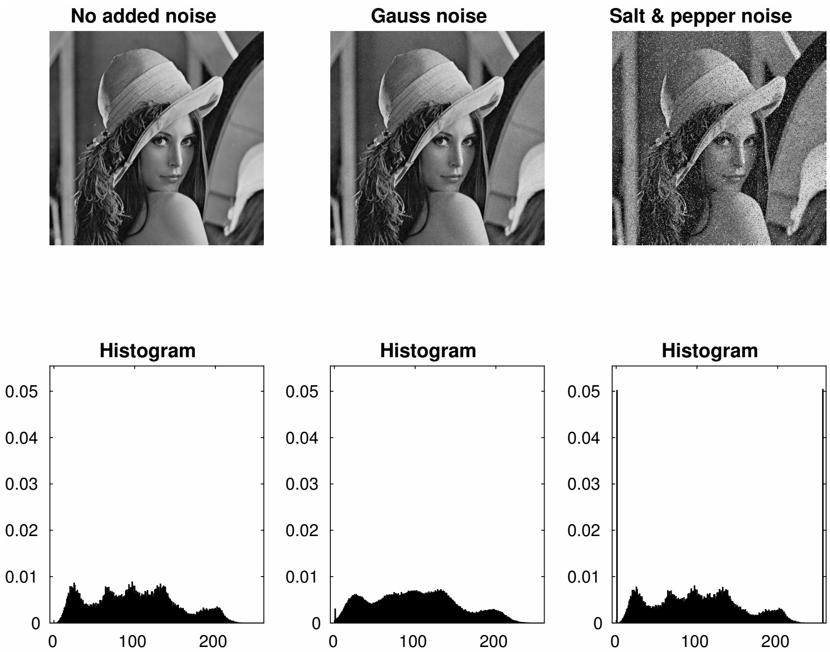

Figure 2: Sub-assigment 2.2 Noised added

ask the teaching assistants in time. Name your report pdf-file in the format `20YYpX_author1_author2.pdf`, where `author1` and `author2` are surnames of the authors.

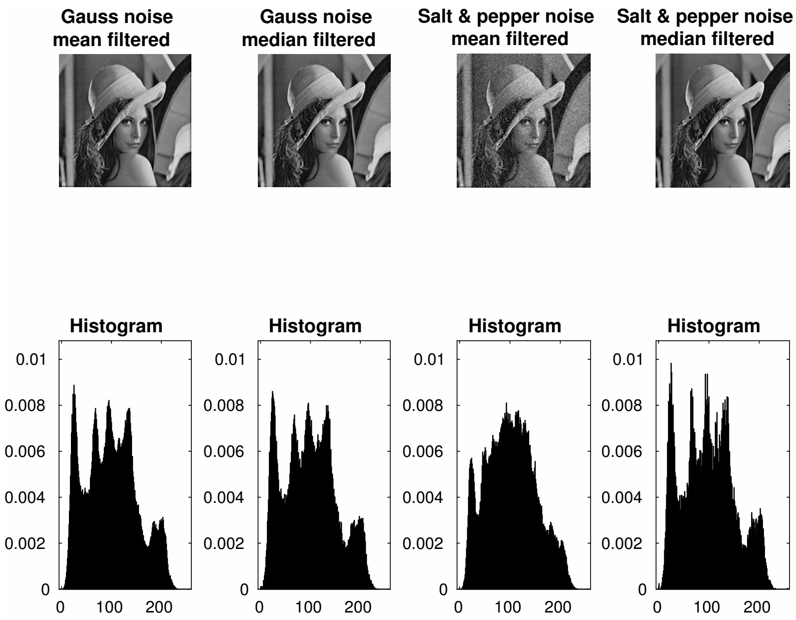

Figure 3: Sub-assigment 2.2 Filtered

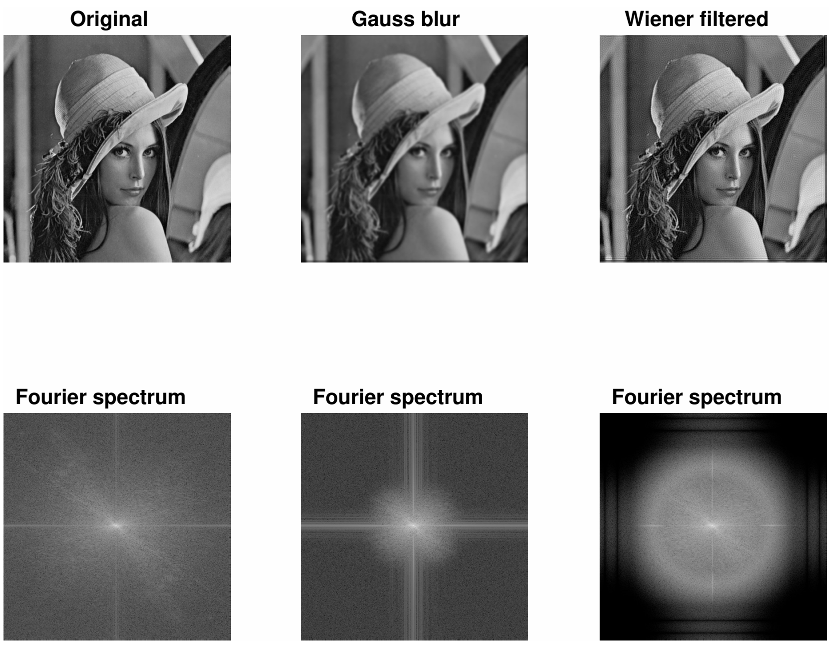

Figure 4: Sub-assigment 3

Appendix

Who Did What

Describe in detail how the project work was divided between the authors. This template was written by Ermin Kozica in LaTeX 2_ε. A good introduction to LaTeX $2_\varepsilon$is available at [2]. You can write your report in other programs as well.

MatLab code

Include the well documented MatLab code that you have used.

```
function h = histogram(f)
% A function that calculates the histogram of matrix f.

N = numel(f); % The number of elements in f
h = ...
```

References

[1] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Prentice Hall, 2nd ed., 2002

[2] Tobias Oetiker et al., *The Not So Short Introduction to LaTeX 2_ε*, Available: http://tobi.oetiker.ch/lshort/lshort.pdf, Last accessed: March 17, 2009