Assignment 4 - Signal and Image Processing 2014

Martin Jørgensen, tzk173

September 24, 2014

Contents

1 Task 1

The code for creating Figure 1 can be found in the appendix.

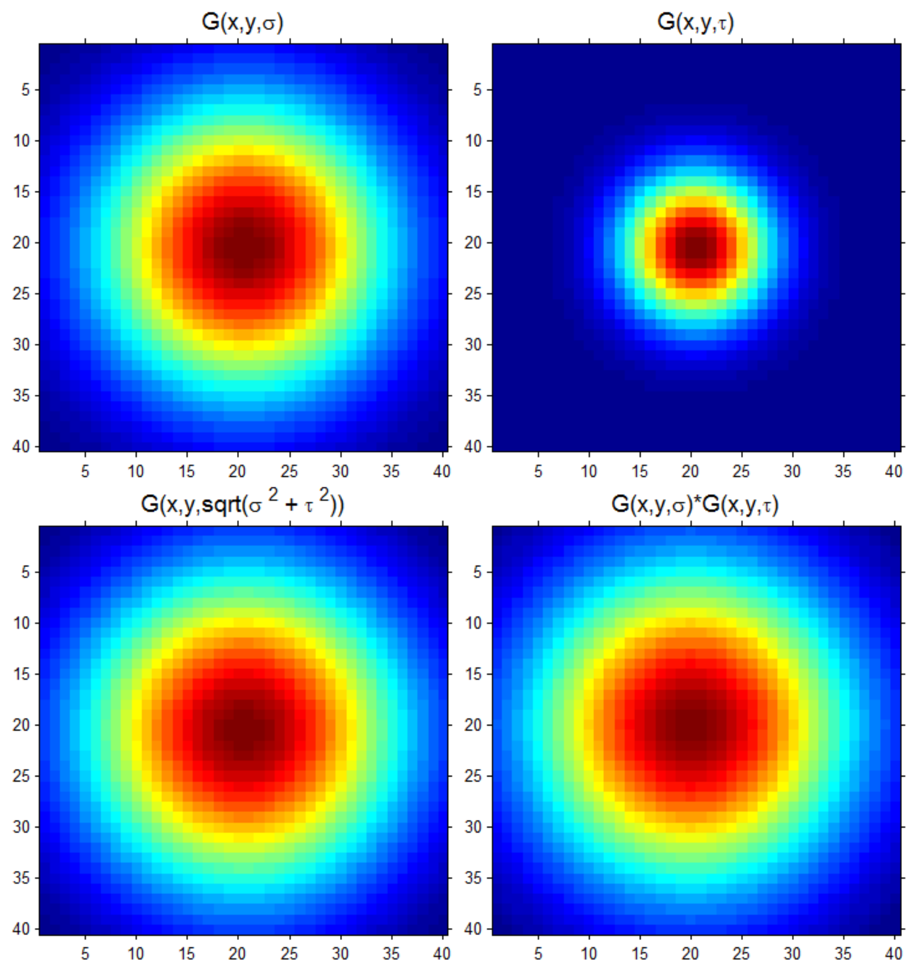

Figure 1: The different kernels produced for the task.

Figure 1 shows the four kernels used for this task. The top left is the kernel produced with the deviation value of $\sigma = 10$, and the upper right us the kernel for deviation value $\tau = 5$. The lower left figure us the kernel produced when the deviation value is set to $\sqrt{\sigma^2 + \tau^2}$, while the lower right shows the resulting kernel from convolving the two upper kernels. There is a number of remarks to be made here. In order to eliminate errors from “border cases”, the size of the kernels was set to be 200×200 so that all the kernels would have 0 weight values around the edges. The kernels are then cropped so we can see the most interesting part, the centres.

The two last lower kernels look very similar but it is worth noticing that the kernel produced by convolution have tiny “spikes/rays” emanating from the centre along the x- and y-axis. This becomes almost invisible when the images are zoomed back out and will only cause minor differences when applied to pictures.

I made several experiments with different kernel sizes and ratios between σ and τ and all yielded this same result. The pictures from the experiments are omitted to preserve space. But can easily be recreated by changing the values in the code in Figure 5 in the appendix.

2 Task 2

Part a

First, lets substitute equation 5 into equation 6 from the assignment we get

$$\begin{aligned}
H(x,y,\tau) &= I_{xx}(x,y,\tau) + I_{yy}(x,y,\tau) \quad (1)\\
&= \tau^{\gamma(2+0)}\frac{\partial^2(x,y,\tau)}{\partial x^2} + \tau^{\gamma(0+2)}\frac{\partial^2 I(x,y,\tau)}{\partial y^2} \quad (2)\\
&= \tau^2\frac{\partial^2 I(x,y,\tau)}{\partial x^2} + \tau^2\frac{\partial^2 I(x,y,\tau)}{\partial y^2} \quad (3)\\
&= \tau^2\left(\frac{\partial^2 I(x,y,\tau)}{\partial x^2} + \frac{\partial^2 I(x,y,\tau)}{\partial y^2}\right) \quad (4)\\
&= \tau^2\left(I(x+1,y,\tau) + I(x-1,y,\tau) - 4I(x,y,\tau) + I(x,y+1,\tau) + I(x,y-1,\tau)\right) \quad (5)
\end{aligned}$$

which is the analytical expression we are looking for.

Part b/c/d

I am unsure how to do partial derivatives on the convolution operation needed in order to do a proper derivation of $I(x,y,\tau)$ in the analytical case. I tried using Matlab to convolve the gaussian and still maintain the symbols needed to do derivation and equation solving later, but was unable to do so.

Note that this problem also persists for Task 3, which I was unable to complete.

3 Task 3

See Task 2B.

4 Task 5

The code for creating Figure 2 can be found in the appendix.

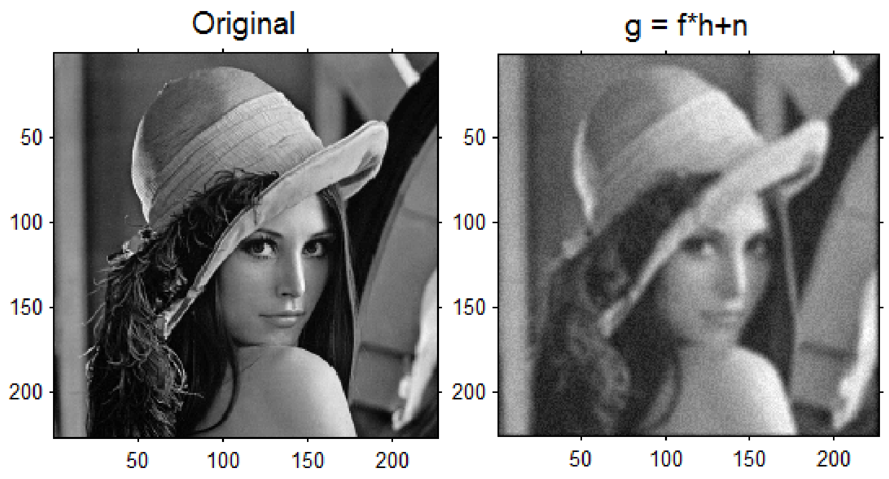

Figure 2: Image before and after the `LSI` operation.

The formula for calculating the LSI is given as $g(x, y) = f(x, y) * h(x, y) + n(x, y)$ where f is the input signal, h is a kernel and n is the noise function.

5 Task 5

The code for creating Figure 3 can be found in the appendix.

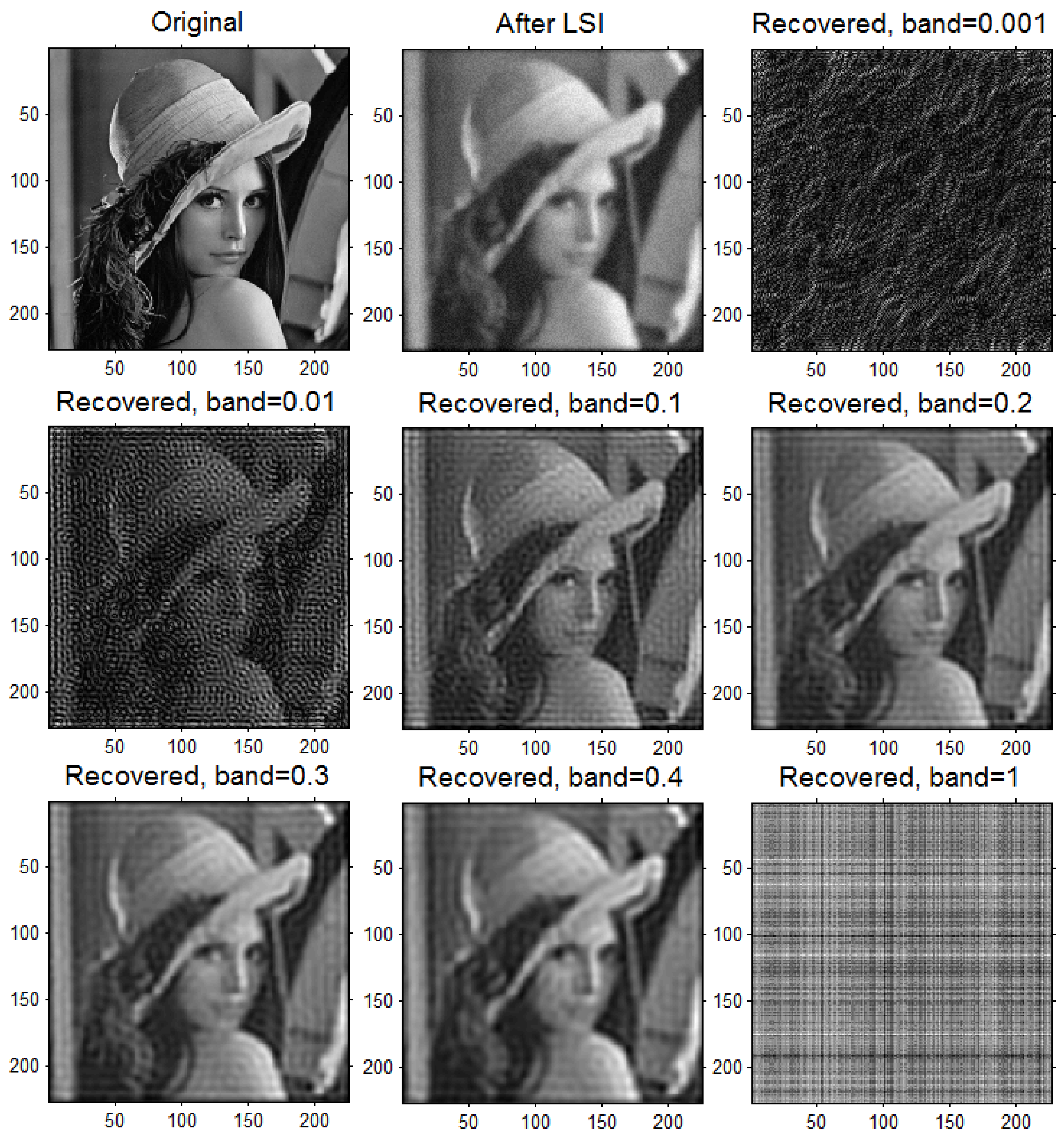

Figure 3: Original image, noisy image and the image after testing the `ILSI` function with different frequency band filters.

Figure 3 shows the original *lena.tif* image as well as the noisy version and different recovered version. The recovered version are produced by using the inverse filtering technique and applying a band pass filter in order to eliminate noise. I tried several different frequencies for the filter, the ones showed in the figure are a select few that represent the most obvious changes in the output. We can see that the filter works best around 0.1 and 0.2. Higher and lower causes the image to degrade into noise again. It is worth noticing that the noise is different for high and low frequency filters. It is worth noticing that Matlab makes a render error when thee frequency is 1, at this value, all of the data gets stripped and all the pixels take on the same value 104.7xxx where the last digits very from run to run, but it is always the same value for all pixels. So the image should be monotone.

6 Task 6

The code for creating Figure 4 can be found in the appendix.

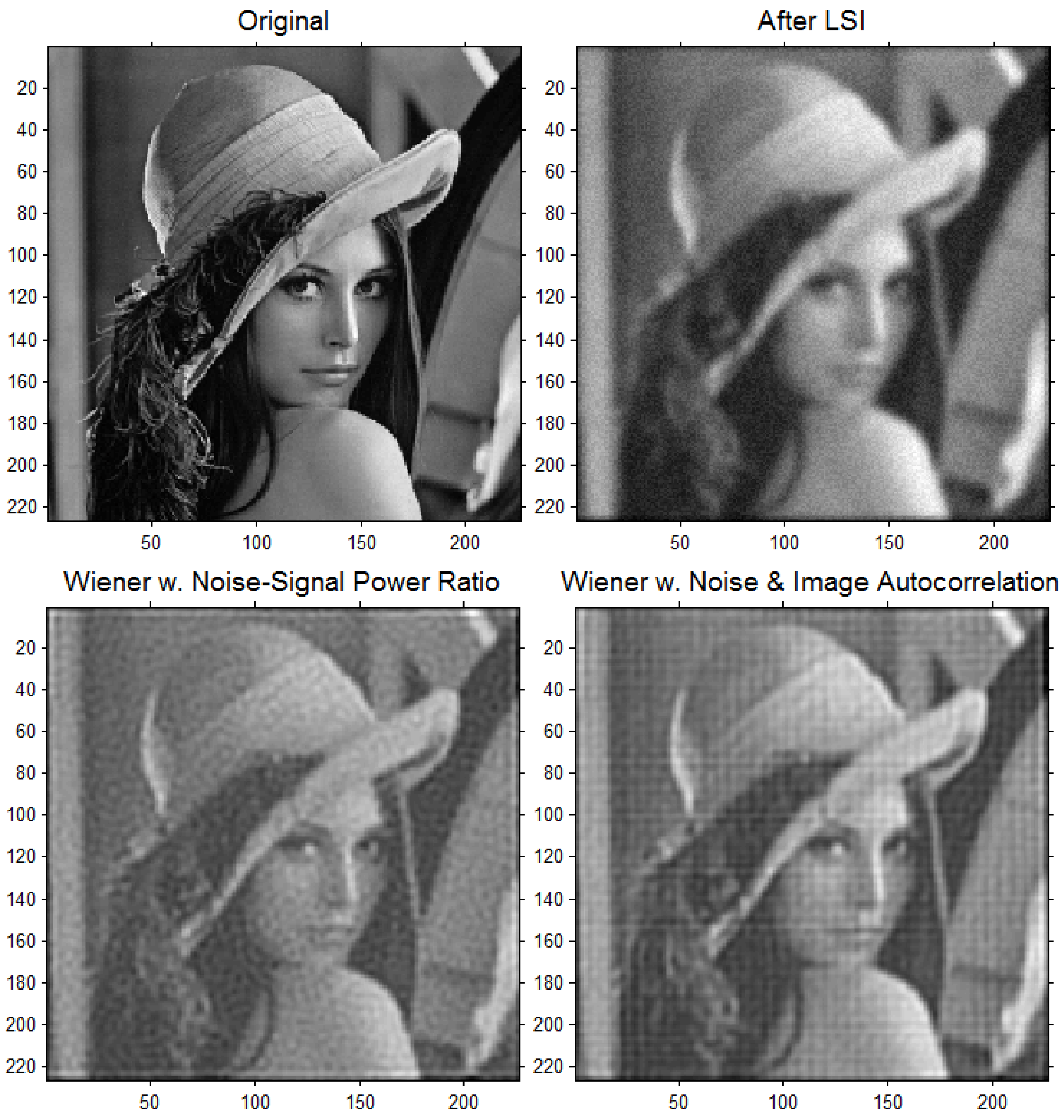
Original
After LSI
Wiener w. Noise-Signal Power Ratio
Wiener w. Noise & Image Autocorrelation
20
40
60
80
100
120
140
160
180
200
220
50
100
150
200

Figure 4: Original *lena.tif* along with the noisy version and the result of two different Wiener filter applications.

Figure 4 shows the application of the Wiener filter in two different ways, one is with the noise-signal power ration, and one is using autocorrelation data generated from the noise and the image respectively.

A Appendix

Code for Task 1

```
% Solution for part 1 of Assignment 4.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I = double(imread('lena.tif'));

% Set appropriate values.
sigma = 10.0;
tau = 5.0;
sigtau = realsqrt(sigma^2+tau^2);
hsize = 200;
% This produces a 40x40 cutout of the kernels.
cropsize = 39;
start = (hsize/2)-cropsize/2;

% Create kernels.
G1 = fspecial('gaussian', hsize, sigma);
G2 = fspecial('gaussian', hsize, tau);
G3 = fspecial('gaussian', hsize, sigtau);
G4 = conv2(G1,G2,'same');

% Crop them.
cG1 = imcrop(G1, [start start cropsize cropsize]);
cG2 = imcrop(G2, [start start cropsize cropsize]);
cG3 = imcrop(G3, [start start cropsize cropsize]);
cG4 = imcrop(G4, [start start cropsize cropsize]);


h = figure(411); set(h,'Color','White'); colormap(jet);
subplot(2,2,1); imagesc(cG1); axis image; set(gca,'TickDir','out');
title('G(x,y,\sigma)','FontSize',14);
subplot(2,2,2); imagesc(cG2); axis image; set(gca,'TickDir','out');
title('G(x,y,\tau)','FontSize',14);
subplot(2,2,3); imagesc(cG3); axis image; set(gca,'TickDir','out');
title('G(x,y,sqrt(\sigma ^2 + \tau ^2))','FontSize',14);
subplot(2,2,4); imagesc(cG4); axis image; set(gca,'TickDir','out');
title('G(x,y,\sigma)*G(x,y,\tau)','FontSize',14);
```

Figure 5: Code for producing the figure for the first task. (../p1.m)

Code for Task 4

`LSI` function

```
function [ g ] = LSI( f, h, n )
%LSI Calculates the Linear Shift Invariant
    g = conv2(f,h,'same') + n;
end
```

Figure 6: LSI function. (../LSI.m)

Creating the figure

```
% Solution for part 5 of Assignment 4.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I = double(imread('lena.tif'));

h = fspecial('gaussian', 6, 2);
n = rand(size(I)).*20;

I1 = LSI(I,h,n);

h = figure(441); set(h,'Color','White'); colormap(gray);
subplot(1,2,1); imagesc(I); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(1,2,2); imagesc(I1); axis image; set(gca,'TickDir','out');
title('g = f*h+n','FontSize',14);
```

Figure 7: Code for producing the figure for the fourth task. (../p4.m)

Code for Task 5

ILSI function

```
function [ f ] = ILSI( g, h, band )
%ILSI - Loosely applied from example 6.1 in the book.
H = fftshift(psf2otf(h, size(h)));

G = fftshift(fft2(g));
idxs = find(H > band);
F = zeros(size(G)); F(idxs)=G(idxs) ./ H(idxs);
f = abs(ifft2(F));

end
```

Figure 8: ILSI function. (../ILSI.m)

Creating the figure

```
% Solution for part 5 of Assignment 4.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I = double(imread('lena.tif'));

% Create kernel and noise.
h = fspecial('gaussian', size(I), 2);
n = rand(size(I)).*20;

% Create image with noise.
I1 = LSI(I,h,n);

h1 = figure(451); set(h1,'Color','White'); colormap(gray);
subplot(3,3,1); imagesc(I); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(3,3,2); imagesc(I1); axis image; set(gca,'TickDir','out');
title('After LSI','FontSize',14);

i=3;
for freq = [0.001 0.01 0.1 0.2 0.3 0.4 1]
    I2 = ILSI(I1,h,freq);
    subplot(3,3,i); imagesc(I2); axis image; set(gca,'TickDir','out');
    title(strcat('Recovered, band=',num2str(freq)),'FontSize',14);
    i = i + 1;
end
```

Figure 9: Code for producing the figure for the fifth task. (../p5.m)

Code for Task 6

```matlab
% Solution for part 6 of Assignment 4.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I = double(imread('lena.tif'));

% Create kernel and noise.
h = fspecial('gaussian', size(I), 2);
n = rand(size(I)).*20;

% Create image with noise.
I1 = LSI(I,h,n);

% Create correlation data
NP = abs(fftn(n)).^2;
NPOW = sum(NP(:))/numel(n);
NCORR = fftshift(real(ifftn(NP)));

IP = abs(fftn(I)).^2;
IPOW = sum(IP(:))/numel(I);
ICORR = fftshift(real(ifftn(IP)));

NSR = NPOW ./ IPOW;

% Wiener filter using both  power ratio and auto correlation.
I2 = deconvwnr(I1, h, NSR);
I3 = deconvwnr(I1, h, NCORR,ICORR);

h1 = figure(461); set(h1,'Color','White'); colormap(gray);
subplot(2,2,1); imagesc(I); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(2,2,2); imagesc(I1); axis image; set(gca,'TickDir','out');
title('After LSI','FontSize',14);
subplot(2,2,3); imagesc(I2); axis image; set(gca,'TickDir','out');
title('Wiener w. Noise-Signal Power Ratio','FontSize',14);
subplot(2,2,4); imagesc(I3); axis image; set(gca,'TickDir','out');
title('Wiener w. Noise & Image Autocorrelation','FontSize',14);
```

Figure 10: Code for producing the figure for the sixth task. (../p6.m)