Assignment 4
Signal & Image Processing

Casper B. Hansen
University of Copenhagen
Department of Computer Science
fvx507@alumni.ku.dk

September 25, 2014

Abstract

...

Contents

1 Scale-space Operators

1.1 Convolution Theorem

To show that $G(x, y, \sigma) * G(x, y, \tau) = G(x, y, \sqrt{\sigma^2 + \tau^2})$ are equivalent I tried to match on the pixels for one-to-one equivalency using the `ginput` tool. Although they are very close, I found that they didn't match perfectly. I think this is due to how the convolution is carried out, as it seems to shift the entire image 1–2 pixels toward the upper-left, and hence some variance in the result might be expected. To counter this, I also tried the alternative built-in `conv2` method, but this produced a similar result.

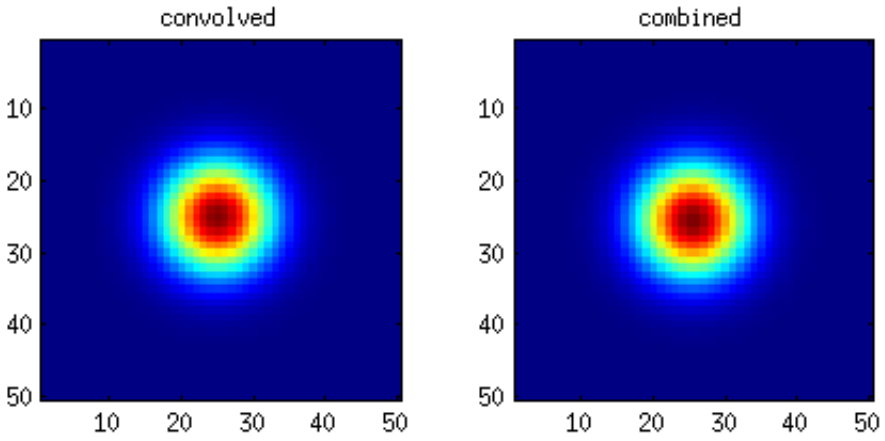

Figure 1: Self-convoluted result (left) and combined kernel (right)

As can be seen in figure (1) the results using window size of 50, and letting $\sigma = 5$ and $\tau = 2$ we see almost no difference in the appearance of the two filter kernels. The code used to produce the figure above can be reviewed in section A.1. Note that I have tried to account for the shifting behavior described.

1.2 Scale Normalized Derivatives

i. Analytical expression

I'm not quite sure what is meant by analytical expression, so I'v reduced the equation into discrete form — hopefully this is what was asked for.

$$H(x, y, \tau) = I_{xx}(x, y, \tau) + I_{yy}(x, y, \tau) = \tau^{\gamma(2+0)} \frac{\partial^2 I(x, y, \tau)}{\partial x^2} + \tau^{\gamma(0+2)} \frac{\partial^2 I(x, y, \tau)}{\partial y^2} \tag{1}$$

$$= \tau^2 \frac{\partial^2 I(x, y, \tau)}{\partial x^2} + \tau^2 \frac{\partial^2 I(x, y, \tau)}{\partial y^2} = \tau^2 \left[\frac{\partial^2 I(x, y, \tau)}{\partial x^2} + \frac{\partial^2 I(x, y, \tau)}{\partial y^2} \right] \tag{2}$$

$$= \tau^2 \left[I(x+1, y, \tau) + I(x-1, y, \tau) - 4I(x, y, \tau) + I(x, y+1, \tau) + I(x, y-1, \tau) \right] \tag{3}$$

ii. **Scales**

After fighting with MatLab for many hours, I had to give in to this. I failed to produce the values of τ sought after. My attempt, as it looks at this time can be found in A.3.

2 Inverse Filtering

2.1 Linear Shift Invariant

Figure (2) shows the result of the program code given in A.4.

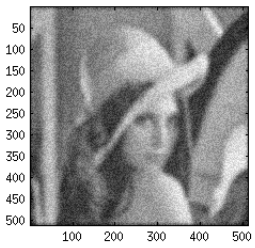

Figure 2: The linear shift invariantly (LSI) degraded result

2.2 Inverse and Band-pass Filtering

Considering the inverse filter $\frac{1}{H(k_x,k_y)}$ we see that for the resulting equation, when applying it to $G(k_x, k_y) + N(k_x, k_y)$, where $N(x, y)$ is the additive noise, we have that for instances of $G(k_x, k_y) < N(k_x, k_y)$ the noise term becomes dominant, an example of which is given in figure (3) (left) where we allow most of the frequencies to pass right through. Since noise often consists of high-frequency signals, the high-pass (see appendix A.5) filter allows for filtering out some of these signals, notably at the expense of $G(k_x, k_y)$ as well.

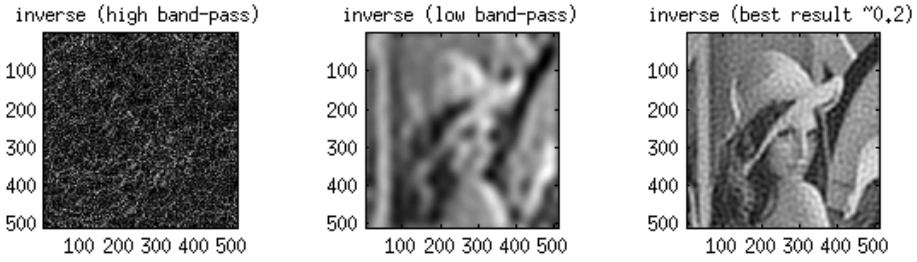

Figure 3: Inverse band-pass filter at different cut-off frequencies

From the tests I have run on this approach I find that the result is generally not good.

2.3 Wiener Filter

The Wiener filter attempts to overcome the limitations of the straight-forward —and naive— approach taken by simply filtering out at a fixed cut-off frequency, as is the case with the high-pass filter. It does so by weighting the contributions of $G(k_x, k_y)$ and $N(k_x, k_y)$, respectively, by way of the signal/noise power signal.

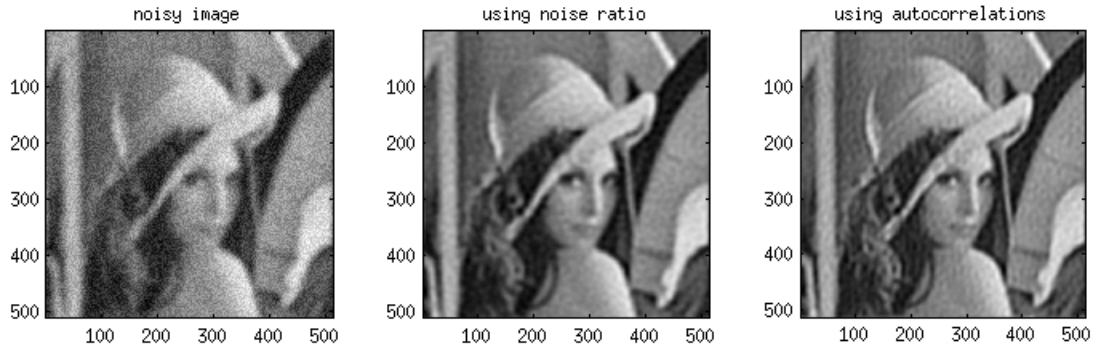

Figure 4: Inverse Wiener results

The results of using this filter is by far an improvement over the band-pass filter method.

A Appendix

A.1 Convolution Theorem

```
%% assignment 1

N = 50; sigma = 5.0; tau = 2.0;

I  = double(imread('images/lena.tif'));
G1 = scale(N, sigma);
G2 = scale(N, tau);
G3 = abs(fftshift(ifft2(fft2(G1) .* fft2(G2))));
G4 = scale(N, sqrt(sigma^2.0 + tau^2.0));

colormap(jet);
subplot(1,2,1), imagesc(G3), axis image, title('convolved');
subplot(1,2,2), imagesc(G4), axis image, title('combined');

[x, y] = ginput(1);
x = uint8(x);
y = uint8(y);

G3(x,y)
G4(x-1,y-1)
```

Figure 5: Convolution (../assignment.m)

A.2 Scale

```
function [ out ] = scale( N, sigma )
    % completely stolen from pp. 132 of the text book
      h = fspecial('gaussian', [N N], sigma);
      out = h;
end
```

Figure 6: Scale function (../scale.m)

A.3 Scale Extremals

```
22 %% assignment 2.b
23
24 clear all; clc;
25
26 syms x y t I(x,y)
27 G(x,y,t) = (1/2*pi*t^2) * exp(-(x^2 + y^2)/(2*t^2));
28
29 dx = diff(G, x, x);
30 dy = diff(G, y, y);
31
32 H = dx + dy;
33
34 S = solve([H, dx, dy], x, y, t);
35 [S.x, S.y, S.t]
```

Figure 7: Scale extremals (attempted) solution (../assignment.m)

A.4 LSI Degradation

```
37 %% assignment 4
38
39 I = double(imread('images/lena.tif'));
40 h = fspecial('gaussian', size(I), sigma);
41 n = rand(size(I)) .* 50;
42 g = conv2(I, h, 'same') + n;
43
44 colormap(gray);
45 subplot(1,1,1), imagesc(g), axis image;
```

Figure 8: LSI degradation code (../assignment.m)

A.5 High-pass Filter

```
47 %% assignment 5
48
49 N = 10; sigma = 5;
50 fq1 = 1e-6; fq2 = 1.0; fq3 = 0.2;
51
52 I = imread('images/lena.tif'); B = fft2(I); B = fftshift(B);
53 n = rand(size(I)) .* 100;
54 h = fspecial('gaussian', size(I), sigma);
55 H = psf2otf(h, size(h)); H = fftshift(H);
56 g = ifft2(H .* B); g = abs(g) + n;
57
58 % inverse, using a high frequency band-pass filter
59 G = fft2(g); G = fftshift(G);
60 indices = find(H > fq1);
61 F = zeros(size(G)); F(indices) = G(indices) ./ H(indices);
62 f = ifft2(F); f = abs(f);
63
64 % inverse, using a low frequency band-pass filter
65 J = fft2(g); J = fftshift(J);
66 indices = find(H > fq2);
67 F = zeros(size(J)); F(indices) = J(indices) ./ H(indices);
68 j = ifft2(F); j = abs(j);
69
70 % inverse, using a low frequency band-pass filter
71 K = fft2(g); K = fftshift(K);
72 indices = find(H > fq3);
73 F = zeros(size(J)); F(indices) = K(indices) ./ H(indices);
74 k = ifft2(F); k = abs(k);
75
76 colormap(gray);
77 subplot(1,3,1), imagesc(f), axis image, title('inverse (high band-pass)');
78 subplot(1,3,2), imagesc(j), axis image, title('inverse (low band-pass)');
79 subplot(1,3,3), imagesc(k), axis image, title('inverse (best result ~0.2)');
```

Figure 9: High-pass filtering (../assignment.m)

References

[1] Solomon & Breckon, Fundamentals of Digital Image Processing - A Practical Approach with Examples in Matlab, 2011