Assignment 3
Signal & Image Processing

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

September 25, 2014

Abstract

In this assignment we examine the Fourier series and Fourier transform.

Contents

1 Theory

1.1 Fourier Series and Transform Differences

The Fourier series produce an approximation of a function by way of a weighted sum of complex exponentials (or harmonic functions), the accuracy of which increases with the number of terms employed. The Fourier transform is a synthesis of a Fourier series (summation) of discrete frequencies into a function of continuous (integral) frequencies.

1.2 Continuous Fourier Transforms

Consider the complex exponential term of the Fourier transform $e^{-ik_x x}$. Because the integral is done over the interval $[-\infty;\infty]$, for any intermediate discrete value a of the exponent we have that;

$$e^{ia} = e^0(\cos a + i \sin a) = \cos a + i \sin a \tag{1}$$
$$e^{-ia} = e^0(\cos a - i \sin a) = \cos a - i \sin a \tag{2}$$

From this general example, we see that the imaginary part of the equations cancel out for each symmetric term over the integral, making it real.

1.3 Derivation of Fourier Transform

Regardless of the constant d the integral over the Dirac (or impulse-) function will stay at exactly 1. So, each of the terms must become 1, and so the Fourier transform for entire expression $\delta(x-d)+\delta(x+d)$ must then be 2.

1.4 The Box Function

Consider the box function

$$b_a(x) = \begin{cases} \frac{1}{a} & \text{iff. } |x| \leq \frac{a}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Show that

i. $\int_{-\infty}^{\infty} b_a(x)dx = 1$

For the general case, we can say that the integral will always have a length over the x-axis of $2\frac{a}{2}$ and the height of it will always be $\frac{1}{a}$. If we consider the formula for the area of a box and apply these findings we have that $A = \frac{1}{a}\cdot 2\frac{a}{2}$. Reducing this, we find that the area is always equal to 1.

$$\frac{1}{a}\cdot 2\frac{a}{2} = \frac{1}{a}\cdot\frac{2a}{2} = \frac{1}{a}\cdot a = \frac{a}{a} = 1 \tag{4}$$

ii. The continuous Fourier transform of b using (5.10) is $B(k) = \frac{1}{ak\pi}\sin\frac{ak}{2}$. Rewrite $B(k)$ using the $\operatorname{sinc}(x) = \frac{\sin x}{x}$ function.

$$\frac{1}{ak\pi}\sin\left(\frac{ak}{2}\right) = \frac{1}{\pi}\frac{1}{ak}\sin\left(ak\frac{1}{2}\right) = \frac{1}{2\pi}\frac{1}{ak\frac{1}{2}}\sin\left(ak\frac{1}{2}\right) = \frac{1}{2\pi}\frac{\sin\left(ak\frac{1}{2}\right)}{ak\frac{1}{2}} = \frac{1}{2\pi}\operatorname{sinc}\left(\frac{ak}{2}\right) \tag{5}$$

iii. $\lim_{a\to 0} B(k) = \frac{1}{2\pi}$ (Hint: $\lim_{x\to 0} \frac{\sin x}{x} = 1$). Does this prove an entry in Table 5.2?

Since $\lim_{x\to 0} \frac{\sin x}{x} = 1$, then letting $a \to 0$ for $\frac{1}{2\pi}\text{sinc}\left(\frac{ak}{2}\right)$ we have that $\lim_{a\to 0} \frac{1}{2\pi}\text{sinc}\left(\frac{ak}{2}\right) = \frac{1}{2\pi} \cdot 1$, which yields $\lim_{a\to 0} B(k) = \frac{1}{2\pi}$.

iv. The filter b has compact support in space (only uses a small set of neighbouring pixels in x). Is the same true in the frequency domain, k? Explain your answer.

Because of the inherent revertibility of Fourier transforms, such information of b must carry over from spatial domain into the frequency domain, so as to make the frequency domain revertible. Therefore, this must be true of the frequency domain as well.

2 Practice

2.1 Power Spectrum

To be honest I'm not quite sure what the power spectrum describes, but it must have to do with frequency of sorts, since it is based on a transform in that domain. I would guess that the 2-dimensional representation thereof shows the frequency variance within the image.

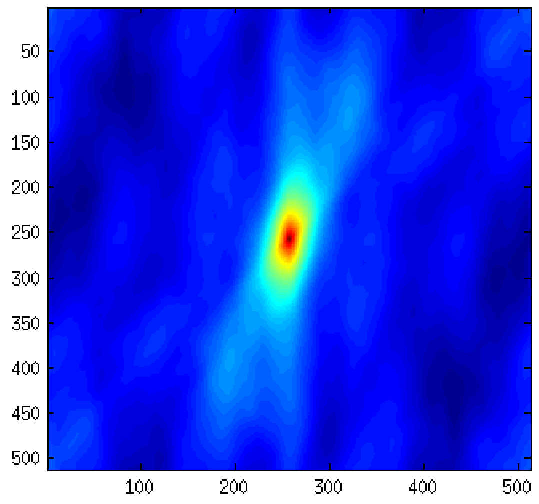

Figure 1: Comparison of convolution methods

2.2 Spatial Representation

The time required to compute the result using the fast fourier transform convolution method (0.05 secs) was by far faster than the for-loop method (2.5 secs).

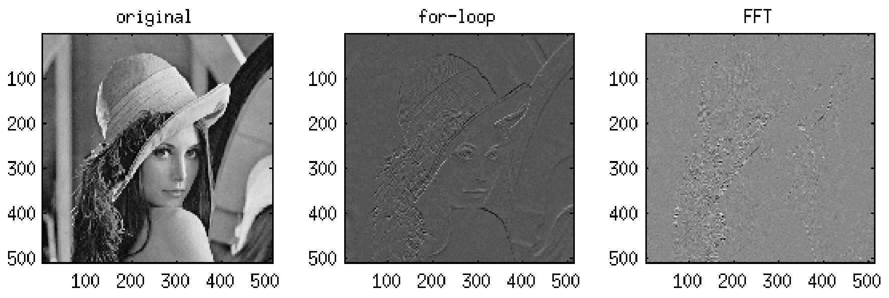

Figure 2: Comparison of convolution methods

As apparent from figure (**??**) above, however, the results were nothing alike. Since the for-loop method produced what I would have expected, this lead me to think I have a problem with the FFT convolution implementation **??**.

2.3 Planar Wave Filter

I'm not entirely sure how a filter could ever remove noise that is dependent on the x and y coordinates. If my understanding of the question is correct I would argue that since the noise introduced is structural in nature, so must such a filter be, and thus a kernel will not suffice. We can, however, remove the noise using an inverse operation on the image (see appendix **??**).

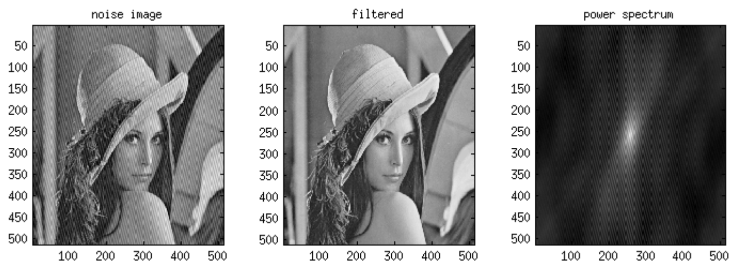

Figure 3: Results from wave filtering, and power spectrum

On the left we have the image with noise added, in middle we have the filtered image, and on the right the power spectrum of the unfiltered image.

2.4 Isotropic Gaussian kernel

The assignment text for this task was very ambiguous, so I've done something along the lines of what is asked, but I'm not entirely sure if it is what was meant to be done. I've used the FFT techniques offered in the book to make a function which basically wraps the gaussian filter in a function.

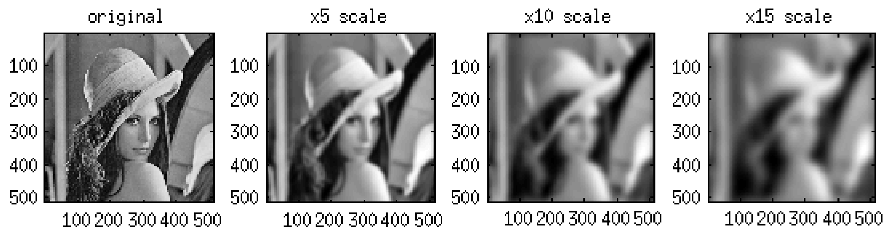

Figure 4: Different scales applied to an image

2.5 Spatial Derivatives

I didn't have time for this one :(

2.6 Derivative Orders

I didn't have time for this one :(

A Appendix

A.1 Power Spectrum

```
%% assignment 2.a

I = double(imread('images/lena.tif'));
P = abs(fftn(I)).^2;
R = fftshift(real(ifftn(P)));

figure (1);
colormap(gray);
imagesc(I);

figure (2);
colormap(jet);
imagesc(R);
```

Figure 5: Excerpt showing the solution for 2-a (../assignment.m)

A.2 Spatial Representation

```
%% assignment 2.b

I1 = double(imread('images/lena.tif'));
I2 = [512 512];

h = [1 2 1; 0 0 0; -1 -2 -1];

% for-loop convolution
tic;
for x = 2:511
    for y = 2:511
        m = [I1(x - 1, y - 1)    I1(x, y - 1)     I1(x + 1, y - 1);
             I1(x - 1, y)        I1(x, y)         I1(x + 1, y);
             I1(x - 1, y + 1)    I1(x, y + 1)     I1(x + 1, y + 1)];
        M = m * h;
        I2(x,y) = M(2,2);
    end
end
for_t = toc;

tic;
I3 = ifft2( fft2(I1) .* freqz2(h, [512 512]) );
fft_t = toc;

colormap(gray);
subplot(1,3,1), imagesc(I1), axis image, title('original');
subplot(1,3,2), imagesc(I2), axis image, title('for-loop');
subplot(1,3,3), imagesc(I3), axis image, title('FFT');

for_t
fft_t
```

Figure 6: Excerpt showing the solution for 2-b (../assignment.m)

A.3 Planer Wave Filter

```
function [ out ] = someFunc ( I, a, v, w )
    s = size(I);
    R = [s(1) s(2)];
    for x = 1:s(1)
        for y = 1:s(2)
            R(x,y) = I(x,y) + a * cos(v*x + w*y);
        end
    end
    out = R;
end
```

Figure 7: Excerpt showing the noise function for 2-c (../someFunc.m)

```
%% assignment 2.c

clear all;

a = 25;
v = 6;
w = 3;

I = double(imread('images/lena.tif'));
I1 = someFunc(I, a, v, w);

h = [0 0 0;
     0 1 0;
     0 0 0];
h = h .* a;

for x = 2:511
    for y = 2:511
        I2(x,y) = I1(x,y) - a * cos(v*x + w*y); % works, but isn't a kernel
%          m = [I1(x - 1, y - 1)    I1(x, y - 1)     I1(x + 1, y - 1);
%               I1(x - 1, y)        I1(x, y)         I1(x + 1, y);
%               I1(x - 1, y + 1)    I1(x, y + 1)     I1(x + 1, y + 1)];
%          M = m * h;
%          I2(x,y) = M(2,2);
    end
end

P = abs(fftn(double(I1))).^2;
POW = fftshift(real(ifftn(P)));
```

Figure 8: Excerpt showing the solution for 2-c (../assignment.m)

A.4 Isotropic Gaussian Kernel

```
subplot(1,3,1), imagesc(I1), title('noise image');
subplot(1,3,2), imagesc(I2), title('filtered');
subplot(1,3,3), imagesc(POW), title('power spectrum');

%% assignment 2.d

I = imread('images/lena.tif');
R1 = scale(I, 5);
R2 = scale(I, 10);
R3 = scale(I, 15);

colormap(gray);
```

Figure 9: Excerpt showing the solution for 2-d (../assignment.m)

References

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