Assignment 3 - Signal and Image Processing 2014

Martin Jørgensen, tzk173

September 18, 2014

Contents

1 Fourier Transform – Theory

Question (a)

What is the difference between a Fourier series and the Fourier Transform?

Page 115 from the book gives the following definitions:

Fourier Series breaks down a periodic signal into harmonic functions of discrete frequencies.

Fourier Transform breaks down a non-periodic signal into harmonic functions of continuously varying frequencies.

The series best used for a signal that have obvious periods and repeats while the transform can be applied to any signal, by assuming that the signal period is going towards ∞.

Question (b)

Prove that the continuous Fourier transform of a real and even function is real and even.

I am not great with formalism, but since the point of Fourier transforms is to represent a function/signal it only makes sense that if the input is real and even, the output needs to be as well. Below is some argumentation that should further strengthen the notion that this is indeed the case.

Since we integrate from $-\infty$ to ∞, we will evaluate for both the positive and negative value for any n. The only imaginary part of the transform comes from the exponential function, so lets take a look at that

$$e^{a+bi} = e^{a}(cos(b) + i \cdot sin(b))$$

In our case $a = 0$ so the exponential function so we have $1(cos(b) + i \cdot sin(b))$ instead. so for both n and $-n$ we get:

$$e^{i\frac{2\pi n}{\lambda}} = 1(cos(\frac{2\pi n}{\lambda}) + i \cdot sin(\frac{2\pi n}{\lambda}))$$
$$e^{-i\frac{2\pi n}{\lambda}} = 1(cos(\frac{2\pi n}{\lambda}) + -i \cdot sin(\frac{2\pi n}{\lambda}))$$

When we in the end integrate over both of these, the imaginary and negative imaginary components will "cancel each other out" and this ensures that the resulting transform is real.

Question (c)

Derive the continuous Fourier transform of $\delta(x - d) + \delta(x + d)$ for some constant d

Assuming the δ is the Dirac Impulse Function (which always integrate to 1), it does not matter if you add anything to the "x" it just shifts along the x-axis. Since there is 2 delta functions added together, the resulting $F(k)$ is 2. (See table 5.2 on page 123), this is just for a single delta function though.

Question (d)

Consider the box function

$$b_a(x) = \begin{cases} 1/a & \textbf{if } |x| \leq \frac{a}{2} \\ 0 & \textbf{otherwise.} \end{cases}$$

show that

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

Since integration is the area under the graph, and we have a box function, we can rewrite the integration as a simple $a = l \times w$ formulae. This can then be solved to be 1.

$$\begin{aligned} 1 &= \frac{1}{a}2\frac{a}{2} \\ &= \frac{2}{a}\frac{a}{2} \\ &= 1 \end{aligned}$$

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 $sinc(x) = \frac{sin(x)}{x}$.**

$$\begin{aligned} B(k) &= \frac{1}{ak\pi} sin\left(\frac{ak}{2}\right) \\ &= \frac{1}{\pi}\frac{1}{ak} sin\left(ak\frac{1}{2}\right) \\ &= \frac{1}{2\pi}\frac{1}{ak\frac{1}{2}} sin\left(ak\frac{1}{2}\right) \\ &= \frac{1}{2\pi}\frac{sin\left(ak\frac{1}{2}\right)}{ak\frac{1}{2}} \\ &= \frac{1}{2\pi} sinc\frac{ak}{2} \end{aligned}$$

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?**

By the tip we have $\lim_{x\to 0}\frac{sin(x)}{x} = 1$, that means that if $a \to 0$ for $\frac{1}{2\pi} sinc\left(\frac{ak}{2}\right)$ we have $\lim_{a\to 0}\frac{1}{2\pi} sinc\left(\frac{ak}{2}\right) = \frac{1}{2\pi}1$ which equals $\lim_{a\to 0} B(k) = \frac{1}{2\pi}$.

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

Since the transformation into the frequency domain is reversible, there must be a correspondence between the two. So if b has compact support in normal space, it will also have it in the frequency space k.

2 Fourier Transform – Practice

Question (a)

Use Matlab to calculate the power spectrum of *lena.tif*. Apply the function `fftshift` and interpret the result representation of the image.

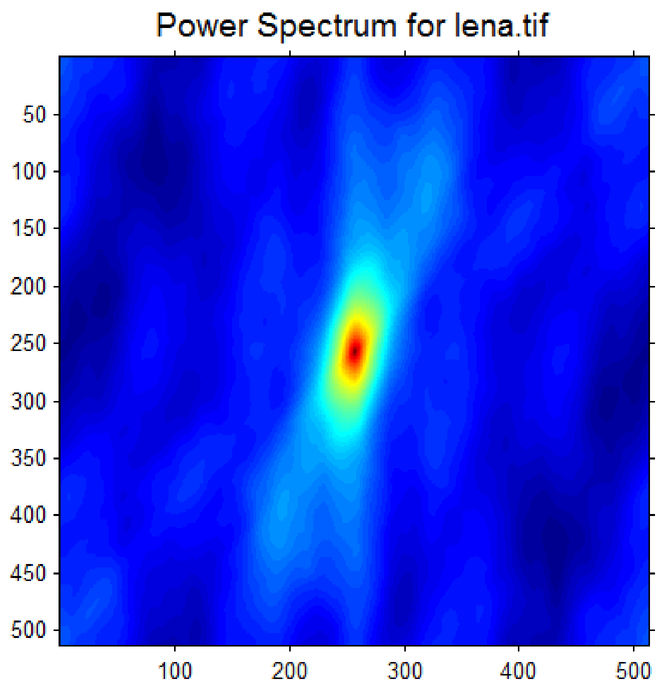

Figure 1: The power spectrum of *lena.tif* rendered with the `jet` color map.

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

I am not quite sure what the power spectrum represents, but after discussing with my group we agreed that it is the change in frequency/intensity in an area of the image.

Question (b)

> **Write two programs: 1 that implements convolution as a nested for loop of the spatial representation of the kernel and image, 2 that implements the same convolution using Fast Fourier Transformation. Compare the two both in terms of the result and the computation time for a number of kernel sizes and image sizes.**

The code to produce Figure 2 can be found in the appendix along with the code for the requested functions. Please be aware that the time is calculated as an average of 10 iterations, which may take some time on slower computers.

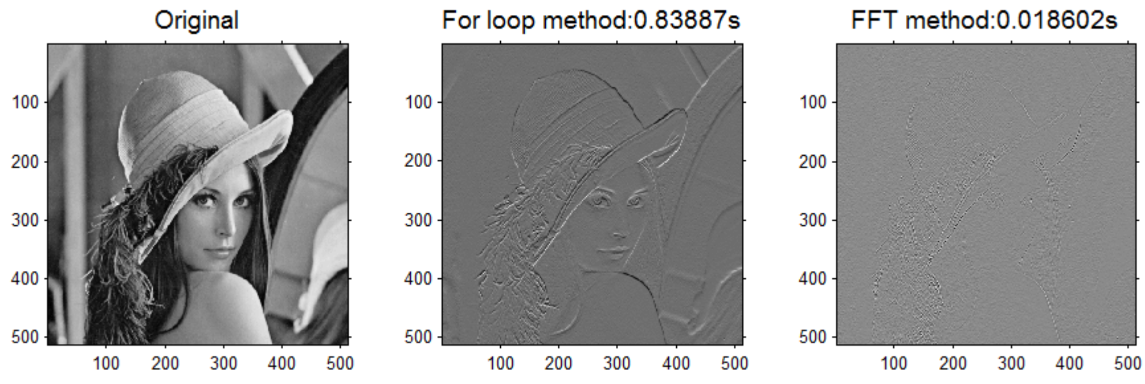

Figure 2: The result of running my convolution methods as well as the time for a single run in seconds.

As the figure shows I had some issues making the FFT method give as clear output as the for loop method. So while it is currently much faster than the for loop method, the result is also less visible. In small version of the image it can be hard to see that the FFT method actually produces an image from the edge detection kernel, but when it is magnified it's easy to make our the contours.

It is also clear that the FFT method is vastly faster than the loop method that applies the kernel to every pixel, one by one.

Question (c)

> **Write a program that adds the function $a_0cos(v_0x + w_0y)$ to *lena.tif*, and evaluate and describe the power spectrum of the result. Design a filter, which removes and such planar waves given v_0 and w_0.**

The code for producing Figure 3 can be found in the appendix along with the code for adding and attempting to remove the noise. Please note that the code uses a third party command from Matlab file exchange[1] to let me use different colormaps in the same figure. This command is the work of John Iversen.[2]

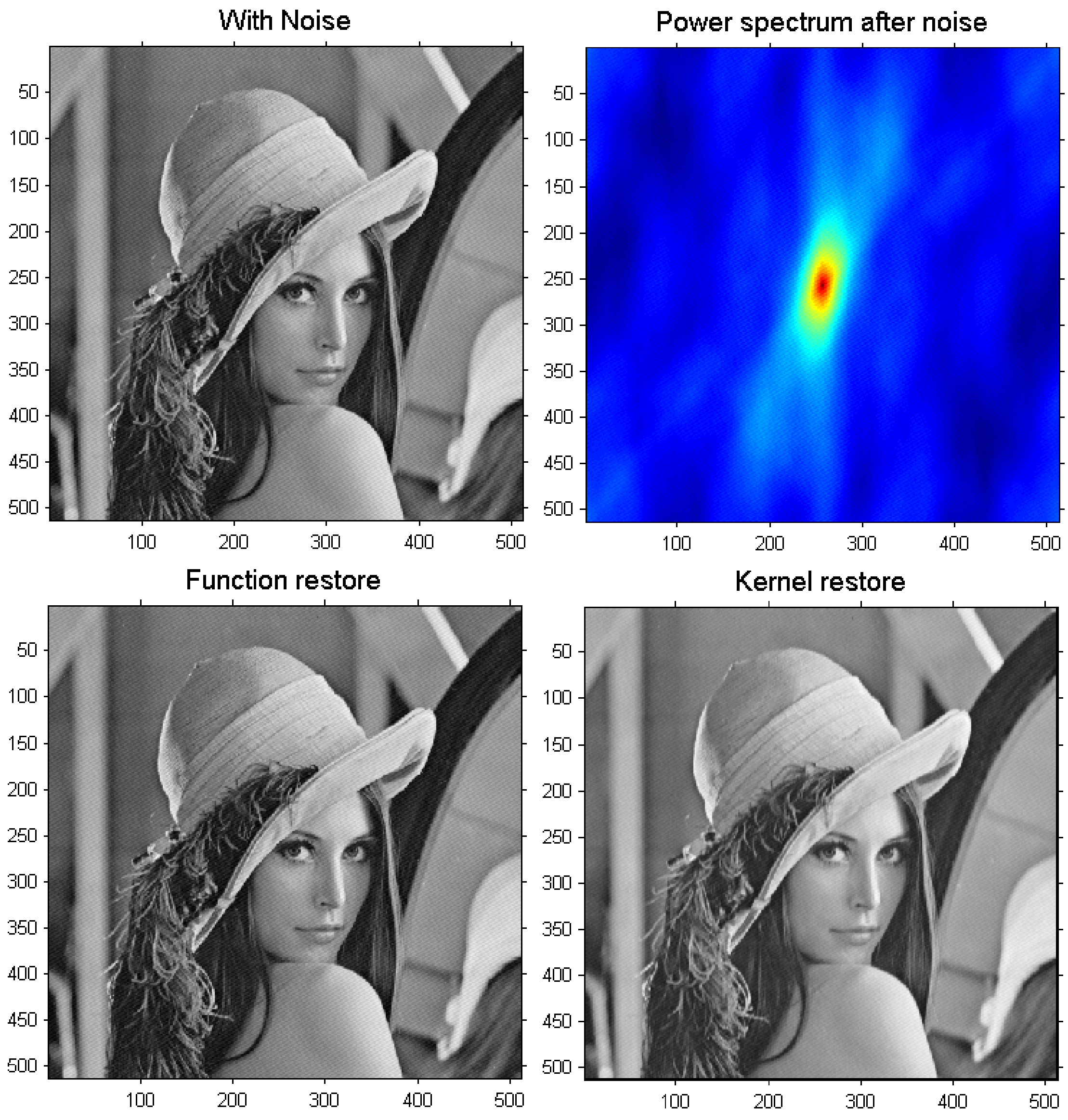

Figure 3: `lena.tif` with noise, power spectrum and after 2 different removal attempts.

We can see that the added noise carries over to the power spectrum.

It is possible to remove much of the noise using a simple smoothing kernel, that is a kernel that takes some values from the area around it and mixes it with the current pixel value.

[1]`http://www.mathworks.com/matlabcentral/fileexchange/7943-freezecolors---unfreezecolors`

[2]John Iversen, 2005-10 - john_iversen@post.harvard.edu

Figure 3 is created by applying the kernel

$$k = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

This kernel is not based in the values $v0$ and $w0$ of course. I was unable to find a proper way to utilize these values since the kernel is unaware of it's current position, it will not be able to scale the values correctly (by x and y) anyway.

I tried to construct a function to counteract the noise based on the fact that we know most of the function that creates the noise. Since the noise addition is done by this formulae $I_2 = I + a_0 cos(v_0 x + w_0 y)$ and we know $cos(v_0 x + w_0 y)$ we can simple subtract that from the pixel value. This of course means we don't know the factor a_0 but it produces reasonable results, even though they are not as good as the simple smoothing kernel used above.

Question (d)

> **Write the function `scale`, which implements convolution with a isotropic Gaussian kernel, parametrized with the its standard deviation - the scale. Apply it to *lena.tif* for a range of scales.**

The assignment text seems a little vague or ambiguous, so I chose to interpret this as we should create a function where you give an image and a "scale". The function will then set a standard deviation for you. The code for producing Figure 4 and the code for the `scale` function can be found in the appendix.

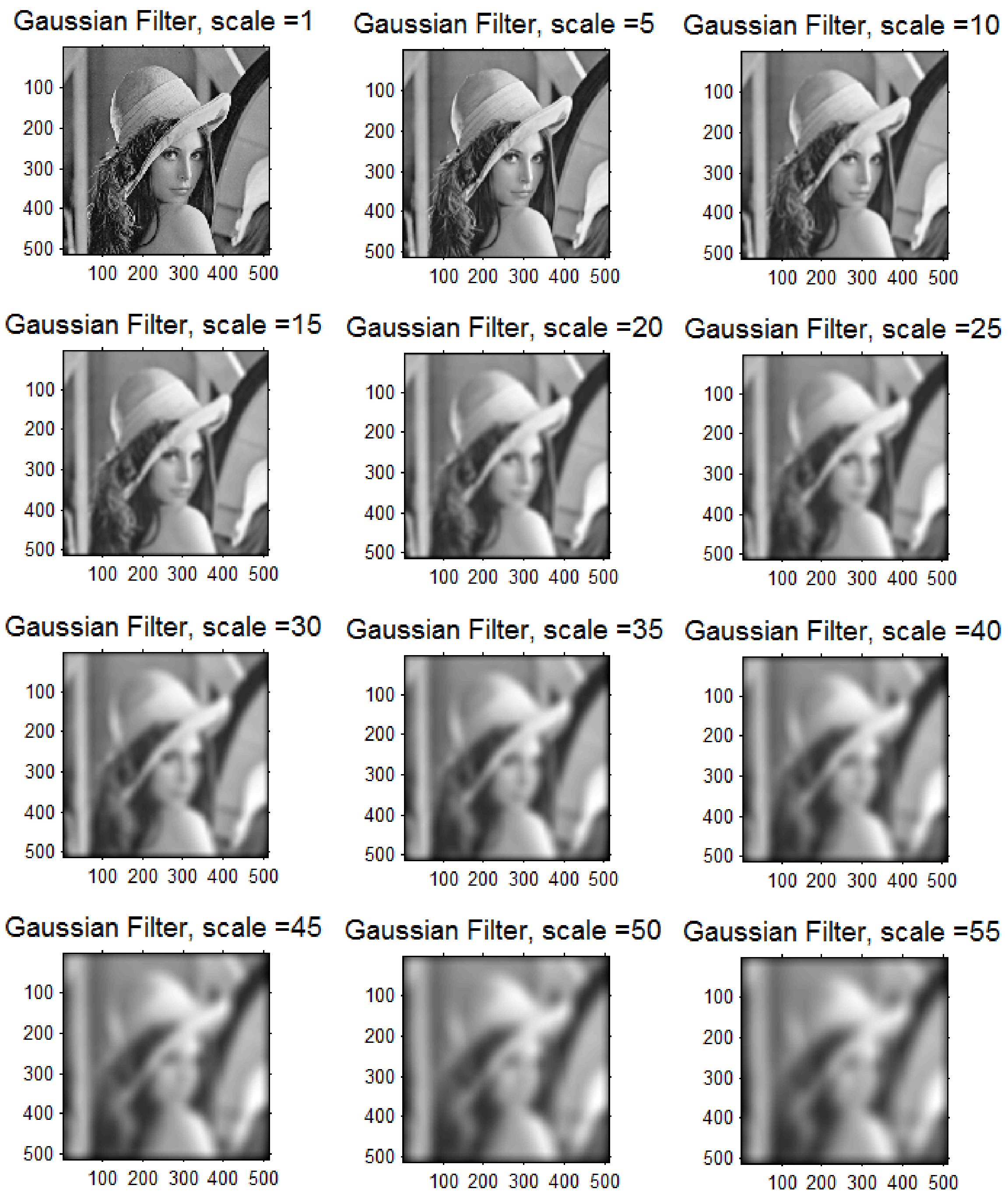
Gaussian Filter, scale =1
Gaussian Filter, scale =5
Gaussian Filter, scale =10
Gaussian Filter, scale =15
Gaussian Filter, scale =20
Gaussian Filter, scale =25
Gaussian Filter, scale =30
Gaussian Filter, scale =35
Gaussian Filter, scale =40
Gaussian Filter, scale =45
Gaussian Filter, scale =50
Gaussian Filter, scale =55
100
200
300
400
500

Figure 4: The scale function used on *lena.tif* with different sizes.

The function sets the standard deviation (σ) based in the scale, it σ value is set to be one third or the size, based on a tip from the assignment last week.

Question (e)

Spatial derivatives may be written as the multiplication of a kernel in the Fourier Domain. Derive the exact relation and discuss its practicality.

Question (f)

Implement a function which takes 2 derivative orders and a 2-dimensional image, and returns the derivative of the image using FFT.

A Appendix

Code For Part 2

Code for Part 2.a

```matlab
% Solution for part 2.a of Assignment 3.
% Written by: Martin Jrgensen, tzk173

clear all;

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

%PS = abs(fft2(I)).^2;
PS = abs(fftn(I)).^2;
SPS = fftshift(ifftn(PS));

h = figure(211); colormap(jet); set(h,'Color','White');
imagesc(SPS); axis image; set(gca,'TickDir','out');
title('Power Spectrum for lena.tif','FontSize',14);
```

Figure 5: Creates a figure with the power spectrum. (../p2a.m)

Code for Part 2.b

```
% Solution for part 2.b of Assignment 3.
% Written by: Martin Jrgensen, tzk173

clear all;

I1 = double(imread('lena.tif'));
%k = [8 12 8;
%     12 20 12;
%     8 12 8];

k = [1,2,1;
     0,0,0;
    -1,-2,-1]; % Edge detection
% I want to run 5 iterations to make sure we have more sample points for
% the time, can be increased as needed. :)
reps = 10;

% Take time for the for loop version.
tic
for i=1:reps
    I2 = forconv(I1,k);
end
fortime = toc/reps;

% Take time for the FFT version.
tic
for i=1:reps
    I3 = fftconv(I1,k);
end
ffttime = toc/reps;

% Construct meaningfull titles.
t1 = 'Original';
t2 = strcat('For loop method: ', num2str(fortime) ,'s');
t3 = strcat('FFT method: ', num2str(ffttime) ,'s');

% Display.... EVERYTHING!
h = figure(321); colormap(gray); set(h,'Color','White');
subplot(1,3,1); imagesc(I1); axis image; set(gca,'TickDir','out');
title(t1,'FontSize',14);

subplot(1,3,2); imagesc(I2); axis image; set(gca,'TickDir','out');
title(t2,'FontSize',14);

subplot(1,3,3); imagesc(I3); axis image; set(gca,'TickDir','out');
title(t3,'FontSize',14);
```

Figure 6: For creating the figure. (../p2b.m)

```matlab
function [ g ] = forconv( f, k )
%forconv Does convulution using for loops.

fs = size(f);
g = zeros(fs);
for y = 2:fs(1)-1
    for x = 2:fs(2)-1
        m = [f(x-1, y-1) f(x, y-1)  f(x+1, y-1);
             f(x-1, y)   f(x, y)    f(x+1, y);
             f(x-1 ,y+1) f(x, y+1)  f(x+1, y+1)];
        M = m*k;
        g(x,y) = M(2,2);
    end
end

end
```

Figure 7: The For loop convolution method. (../forconv.m)

```matlab
function [ g ] = fftconv( f, k )
%fftconv Uses FFT to convolve a kernel on an image.
sf = size(f);
ff = fft2(f);
fk = freqz2(k,sf(1),sf(2));

fmr = ff .* fk;
g = ifft2(fmr);

end
```

Figure 8: The For FFT convolution method. (../fftconv.m)

Code for Part 2.c

```matlab
% Solution for part 2.c of Assignment 3.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I1 = imread('lena.tif');
a0 = 10;
v0 = 1.5;
w0 = 0.9;
k = [2 2 2;
     2 3 2;
     2 2 2];

I2 = addfunc(I1, a0, v0, w0);
I3 = defunc(I2, v0, w0);
I4 = forconv(I2, k);

PS = abs(fftn(I2)).^2;
SPS = fftshift(ifftn(PS));

h = figure(331); set(h,'Color','White');
subplot(2,2,1); imagesc(I2); axis image; set(gca,'TickDir','out');
title('With Noise','FontSize',14); colormap(gray); freezeColors;
subplot(2,2,2); imagesc(SPS); axis image; set(gca,'TickDir','out');
title('Power spectrum after noise','FontSize',14); colormap(jet); freezeColors;
subplot(2,2,3); imagesc(I3); axis image; set(gca,'TickDir','out');
title('Function restore','FontSize',14); colormap(gray); freezeColors;
subplot(2,2,4); imagesc(I4); axis image; set(gca,'TickDir','out');
title('Kernel restore','FontSize',14);
```

Figure 9: For creating the figure. (../p2c.m)

```matlab
function [ I2 ] = addfunc( I1, a0, v0, w0 )
%addfunc a0 is assumed to be the value of the pixel.

Is = size(I1);
I2 = zeros(Is);

for x = 1:Is(1)
    for y=1:Is(2)
        I2(x,y) = I1(x,y)+ a0*cos(v0*x + w0*y);
    end
end
end
```

Figure 10: Function for adding the “noise”. (../addfunc.m)

```matlab
function [ I2 ] = defunc( I1, v0, w0 )
%defunc Defuncicating stuff.

Is = size(I1);
I2 = zeros(Is);

for x = 1:Is(1)
    for y=1:Is(2)
        I2(x,y) = I1(x,y)-cos(v0*x + w0*y);
    end
end
end
```

Figure 11: Function for removing the “noise”. (../defunc.m)

Code for Part 2.d

```
% Solution for part 2.d of Assignment 3.
% Written by: Martin Jrgensen, tzk173

clear all;

% Read base image.
I1 = imread('lena.tif');

h = figure(225); set(h,'Color','White'); colormap(gray);
i = 1;
for s=[1 5 10 15 20 25 30 35 40 45 50 55]
    I2 = scale(I1, s);
    subplot(4,3,i); imagesc(I2); axis image; set(gca,'TickDir','out');
    title(strcat('Gaussian Filter, scale = ',num2str(s)),'FontSize',14);
    i = i + 1;
end
```

Figure 12: For creating the figure. (../p2d.m)

```
function [ I2 ] = scale( I, scale )
%scale
    gf = fspecial('gaussian', scale, scale/3);
    I2 = filter2(gf, I);
end
```

Figure 13: `scale` function. (../scale.m)