CS 663 - Assignment 5

Shaan Ul Haque 180070053
Samarth Singh 180050090
Niraj Mahajan 180050069

November 12, 2020

Question 1

Given the image taken when the outside scene is in focus g_1, and the image when the reflection off the window is in focus g_2, along with the respective blurring kernels h_1 and h_2, we have

$$g_1 = f_1 + h_2 * f_2$$

$$g_2 = f_2 + h_1 * f_1$$

By applying Fourier transform on these equations, we get,

$$G_1 = F_1 + H_2F_2$$

$$G_2 = F_2 + H_1F_1$$

On further solving the system of linear equations for F_1, F_2,

$$F_1 = \frac{G_1 - H_2G_2}{1 - H_1H_2}$$

$$F_2 = \frac{G_2 - H_1G_1}{1 - H_1H_2}$$

Now, to obtain f_1 and f_2, we simply take the Fourier Inverse Transform of F_1 and F_2

$$\boxed{f_1 = \mathcal{F}^{-1}(F_1)}$$

$$\boxed{f_2 = \mathcal{F}^{-1}(F_2)}$$

Problem in the solution obtained

The inherent problem in the formula derived is that the Fourier transforms of the blurring kernels act as low pass filters. So, for low frequencies, H_1H_2 will approach 1. If we look at the denominator of the formulae $1 - H_1H_2$, we notice that for low frequencies, the denominator tends to ∞. This is will amplify the noise around low frequencies and hence is not ideal.

Question 2

1D image

$$g(x) = (h * f)(x)$$

where g is the gradient image (in 1D), h is the convolution kernel to represent the gradient operation, and f is the original 1D image

Using convolution theorem in 1D

$$G(u) = H(u)F(u)$$

where G represents the Fourier Transform of g, F is the Fourier Transform of f and H is the Fourier Transform of h

g and h are known $\Rightarrow$ G and H are known

$$F(u) = \frac{G(u)}{H(u)}$$

i.e. We can get F by point-wise division of G by H
Once we get F, we can use inverse Fourier Transform to get f
Fundamental difficulties:

- **Division by zero**: H can be zero for some u, which will result in a 0/0 indeterminate form.
- **Uniqueness** of f: Adding a constant to f will not affect g, thus for same G and H we can have different f. But above approach will always result in only one F and f.
- **Different dimensions** of h and f matrix: We need G, F and H to be of the same dimension. g and f are of the same dimension but h can have different dimension. So we need to zero-pad h and g appropriately before calculating their Fourier transform.

2D image

$$g(x, y) = (h * f)(x, y)$$

where g is the gradient image (in 2D), h is the convolution kernel to represent the gradient operation, and f is the original 2D image

Using convolution theorem in 2D

$$G(u, v) = H(u, v)F(u, v)$$

where G represents the Fourier Transform of g, F is the Fourier Transform of f and H is the Fourier Transform of h

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

g and h are known $\Rightarrow$ G and H are known

$$F(u,v) = \frac{G(u,v)}{H(u,v)}$$

i.e. We can get F by point-wise division of G by H
Once we get F, we can use inverse Fourier Transform to get f

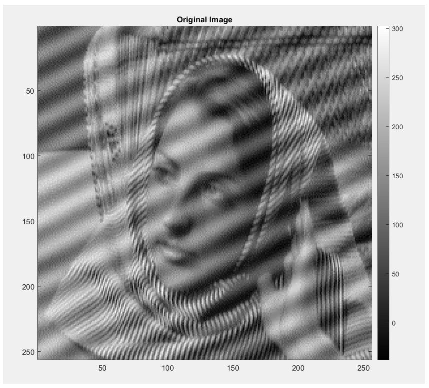
Fundamental difficulties:

- **Division by zero**: H can be zero for some u,v, which will result in a 0/0 indeterminate form
- **Uniqueness** of f: Adding a constant to f will not affect g, thus for same G and H we can have different f. But above approach will always result in only one F and f.
- **Different dimensions** of h and f matrix: We need G, F and H to be of the same dimension. g and f are of the same dimension but h can have different dimension. So we need to zero-pad h and g appropriately before calculating their Fourier transform.

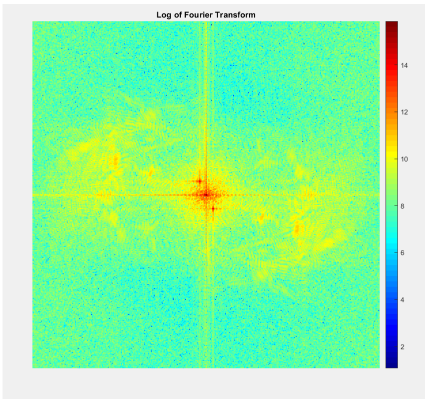
Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

1 Fourier Transform of the noisy image

The Fourier Transform of the image contained unusual peak at coordinates (266, 276) and (246, 236) which correspond to frequency pairs (-10,20) and (10, -20) respectively. The image and its Fourier Transform are shown in fig. below. This is justified also as we see



(a) Original Image



(b) Fourier Transform

Figure 1

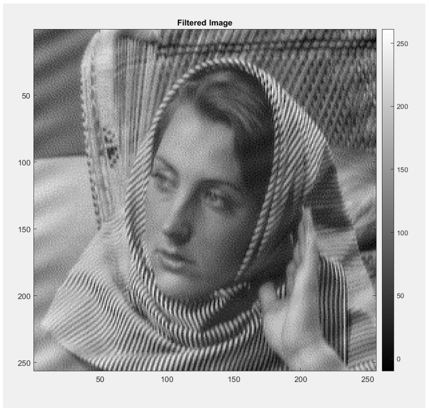
the noise pattern in the image are straight lines at certain angle. The Fourier Transform of corresponding to these patterns lies on the line perpendicular to these patterns, i.e, on the line $y = -2x$.

To suppress this pattern we design a notch filter which is 1 everywhere in the Fourier plane but at the points nearby (266, 276) and (246, 236) where it is zero.

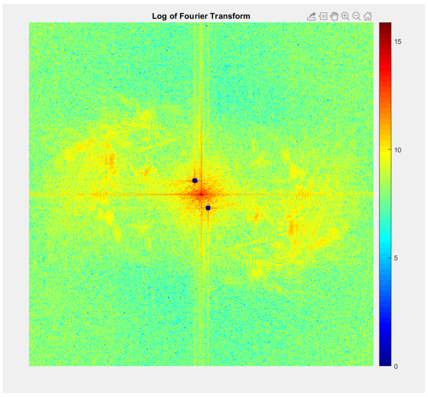
$$H(u, v) = 0 \ ; \ if \ (u - u_0)^2 + (v - v_0)^2 <= D^2$$

$$H(u, v) = 1; otherwise$$

After some hit and trial parameter $D^2 = 14$ was found to give good results. Images are shown below.



(a) Output Image



(b) Fourier Transform

Figure 2

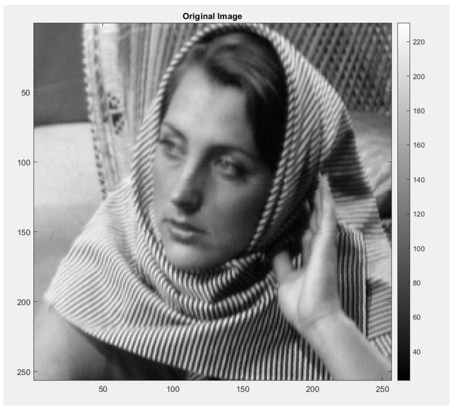
2 Code Usage

The code is small which consists of several small parts. Firstly, we extract the image from the .mat file then display the Fourier transform (after appropriate padding) to locate the frequency corresponding to noise. The second part consists of designing notch filter in the Fourier domain, then multiplying the filter with the Fourier Transform and obtain the inverse Fourier Transform of the image. Finally we extract the middle part of the image and display it.

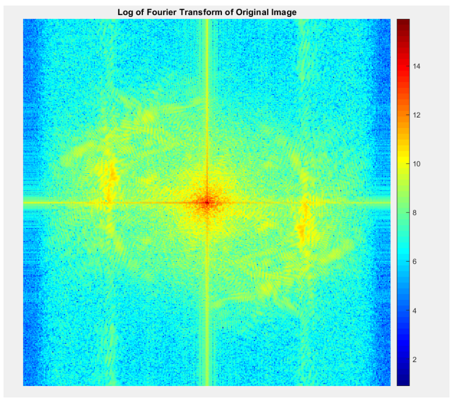
Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

1 Ideal LPF on the image

Original Image and its Fourier Transform are shown in fig. below.



(a) Original Image



(b) Fourier Transform

Figure 1

We want to pass this image through an ideal lpf and see its effect on the image. Ideal low pass filter is defined by the following equation:

$$H(u,v) = 1 \; ; \; if \;\; u^2 + v^2 <= D^2$$

$$H(u,v) = 0; otherwise$$

1.1 Cutoff frequency $|D| = 40$

The frequency response corresponding the filter is shown below.

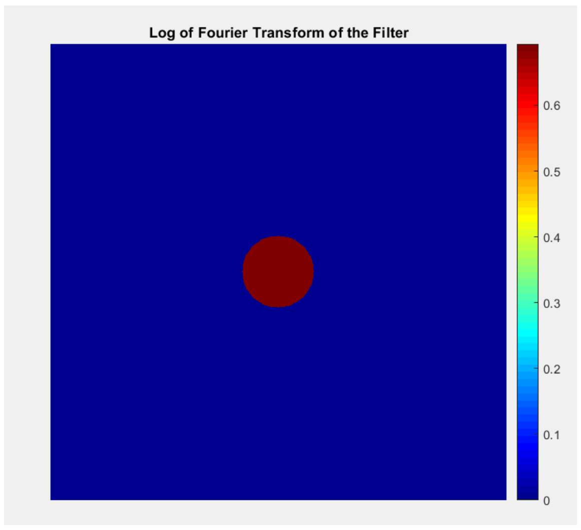
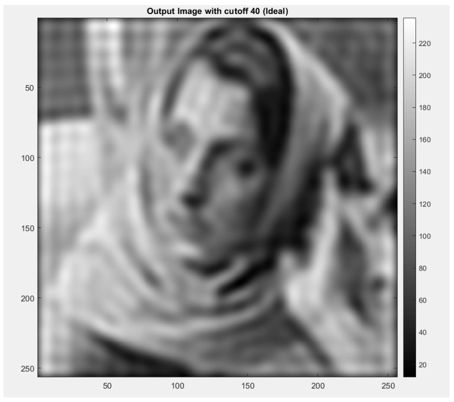
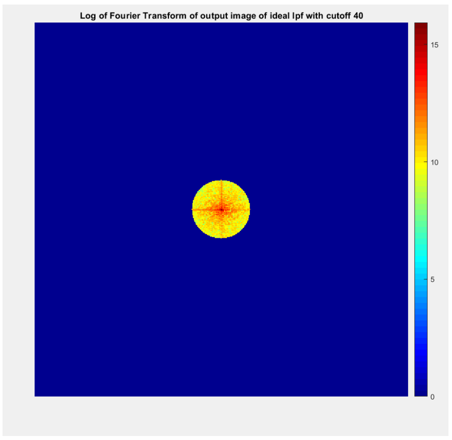


Figure 2: Frequency response of an ideal LPF with cutoff frequency $|\omega| = 40$

Image and the corresponding Fourier Transform are shown in fig. given below.



(a) Output Image after applying ideal LPF



(b) Fourier Transform of the Image

Figure 3

1.2 Cutoff frequency $|D| = 80$

The frequency response corresponding the filter is shown below.

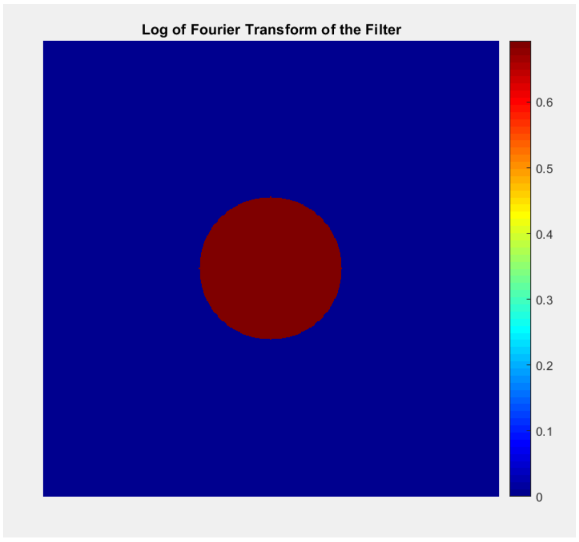
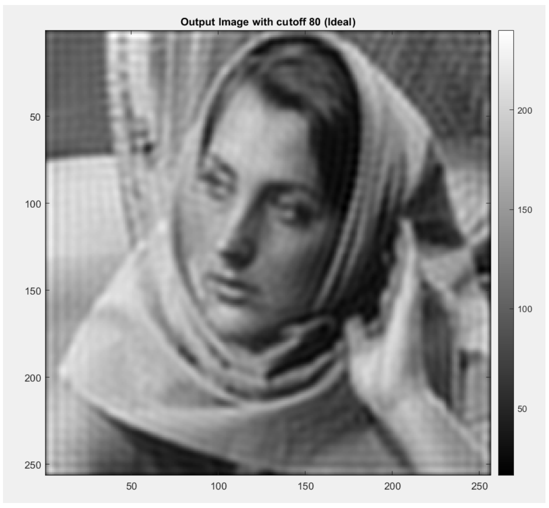
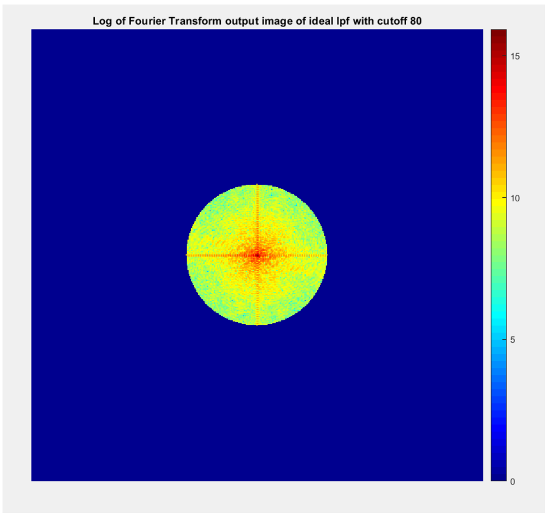


Figure 4: Frequency response of an ideal LPF with cutoff frequency $|\omega| = 80$

Image and the corresponding Fourier Transform are shown in fig. given below.



(a) Output Image after applying ideal LPF



(b) Fourier Transform of the Image

Figure 5

2 Gaussian low pass filter

2.1 Standard Deviation $\sigma = 40$

The frequency response corresponding the filter is shown below.

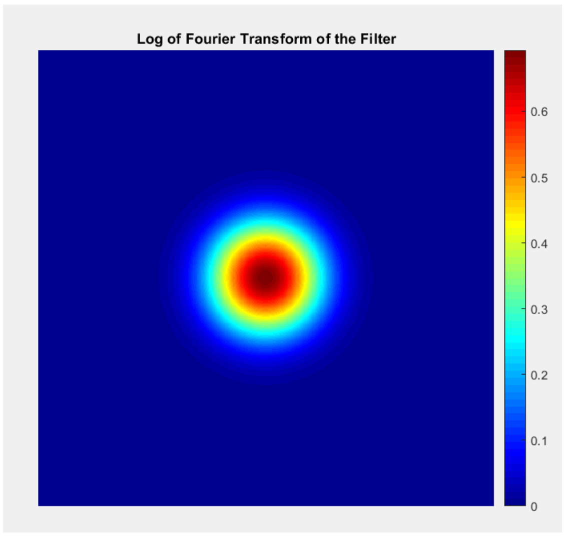
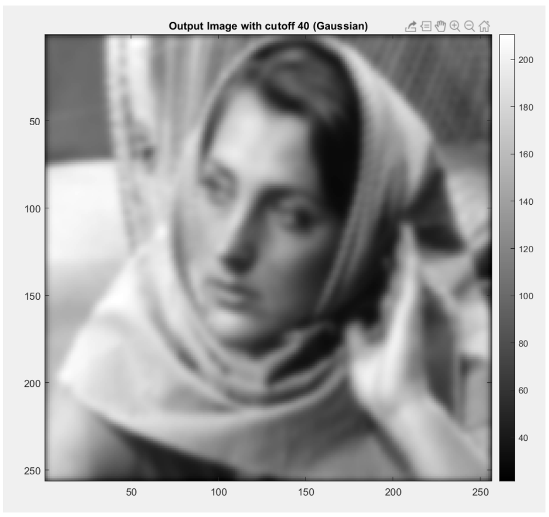
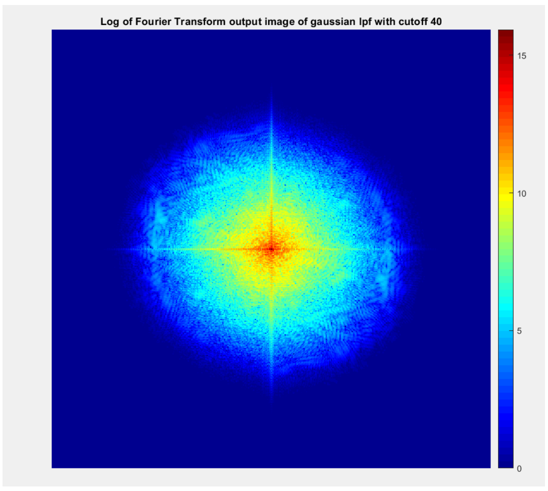


Figure 6: Frequency response of an Gaussian LPF with standard deviation $|\sigma| = 40$

Image and the corresponding Fourier Transform are shown in fig. given below.



(a) Output Image after applying Gaussian LPF



(b) Fourier Transform of the Image

Figure 7

2.2 Standard Deviation $\sigma = 80$

The frequency response corresponding the filter is shown below.

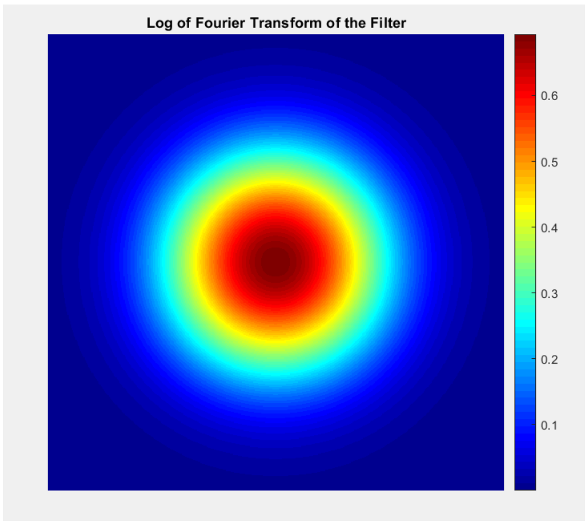
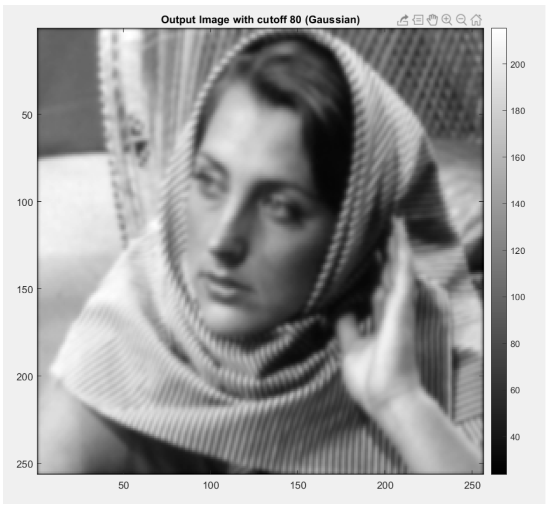


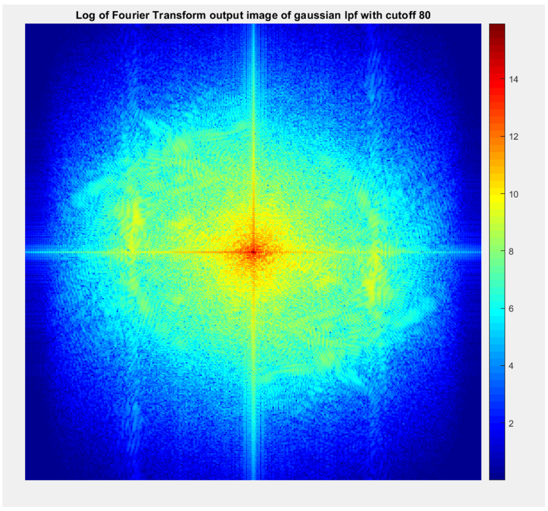
Figure 8: Frequency response of an Gaussian LPF with standard deviation $|\sigma| = 80$

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Image and the corresponding Fourier Transform are shown in fig. given below.



(a) Output Image after applying Gaussian LPF



(b) Fourier Transform of the Image

Figure 9

3 Explanation

We can see a stark difference in the output image which was expected also. The idea LPF corresponds to jinc function in the spatial domain which gives rippling or ringing effect in the image. As the size of the LPF is increased the ringing effect also decreases. The Fourier Transform of a Gaussian Filter gives a Gaussian kernel in the spatial domain also which is essentially Gaussian smoothening of the image.

4 Code Usage

The code is divided into three parts.

- Part 1 is just extracts the image from the folder and displays it and its Fourier Transform.
- Part 2 corresponds to an LTI system which is an ideal LPF with cutoff frequencies, f$\in \{40, 80\}$.
- Part 3 corresponds to an LTI system which is a Gaussian LPF with standard deviation, $\sigma \in \{40, 80\}$.

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Question 5

1. Code

```
function [x,y, impulse, visibility, log_fft] = getTranslation(I1,I2)
    [r1,c1] = size(I1);
    [r2,c2] = size(I2);
    assert(r1 == r2);
    assert(c1 == c2);

    % calculate FFTs
    F1 = fftshift(fft2(I1));
    F2 = fftshift(fft2(I2));

    % compute the cross-power spectrum, and the corresponding impulse
    Prod = (F1.*conj(F2))./(abs(F1.*F2) + 1e-50);
    log_fft = log(1+abs(Prod));
    prod = abs(ifft2(ifftshift(Prod)));
    impulse = prod/max(prod(:));

    % interpret the translation from the obtained impulse
    [~, linearIndexesOfMaxes] = max(prod(:));
    y = rem(linearIndexesOfMaxes-1,300);
    x = (linearIndexesOfMaxes-y-1)/300;
    startx = x;
    starty = y;
    if (x > idivide(int32(c1),2))
       x = x - c1;
    end
    if (y > idivide(int32(r1),2))
       y = y - r1;
    end

    % create a yellow circle around the impulse for better visibility
    visibility = cat(3,impulse,impulse,impulse);
    visibility = visibility/max(visibility(:));
    circle = zeros(20,20,3);
    for i = 1:20
        for j = 1:20
            if (abs((i-10)^2 + (j-10)^2 - 81) < 20)
                circle(i,j,:) = [1,1,0];
            end
        end
    end
    visibility(starty-9:starty+10, startx-9:startx+10,:) = visibility(
   starty-9:starty+10, startx-9:startx+10,:) + circle;
end
```

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

2. Image Registration without noise

In the noiseless case, the translation was obtained at $t_x = -30$ and $t_y = 70$.
Below are the corresponding plots. The 4th plot is just the 3rd plot with a marking circle added for better visibility. (Kindly zoom-in the pdf for better clarity). Assuming the rectangles are vertical, ie, the longer length is vertically aligned.

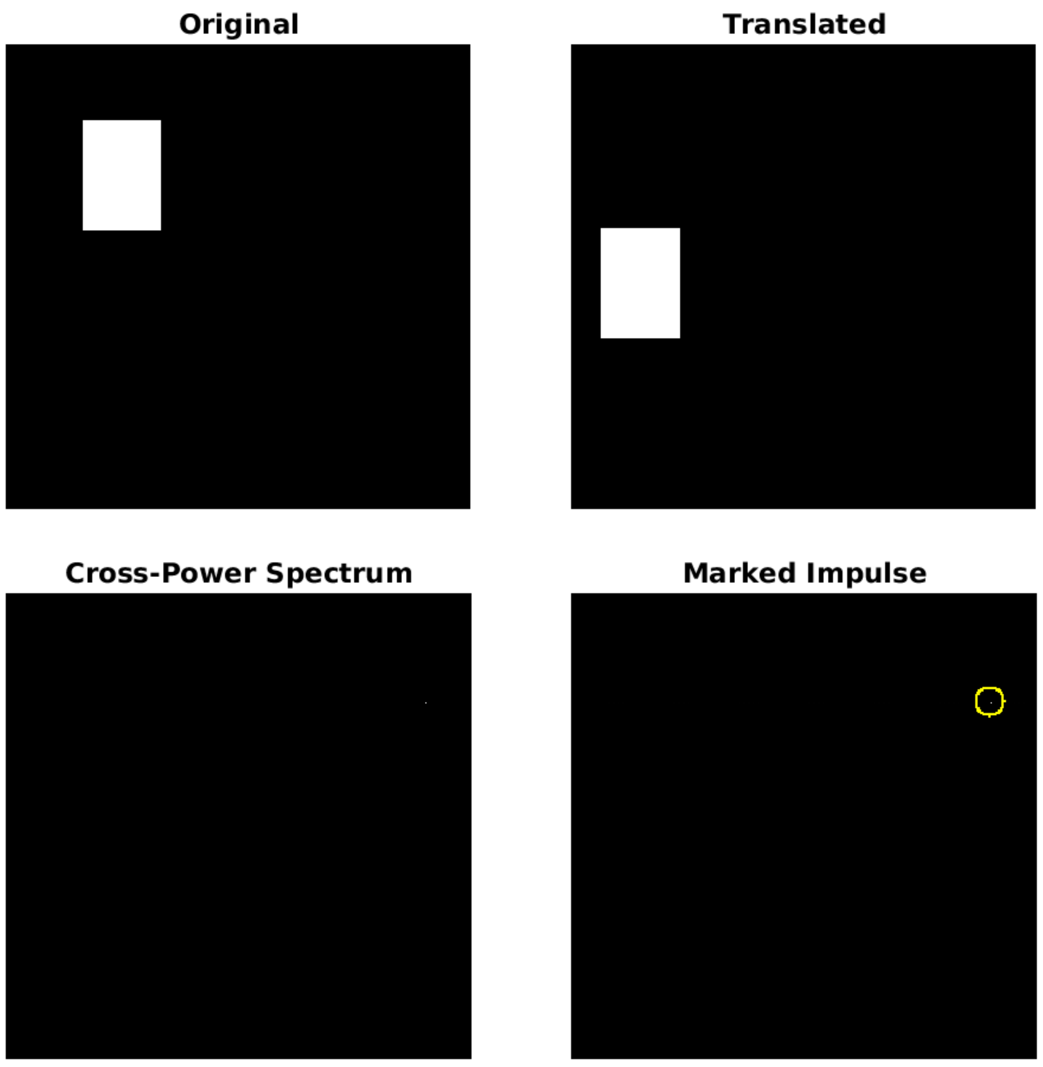


Figure 5.1: Image Registration

3. Image Registration with noise

In the noisy case, the translation was obtained at $t_x = -30$ and $t_y = 70$.
Below are the corresponding plots. The 4th plot is just the 3rd plot with a marking circle added for better visibility. The rectangles appear a bit gray, because after adding noise, all pixels are scaled down between [0,1], and not clipped above 255.

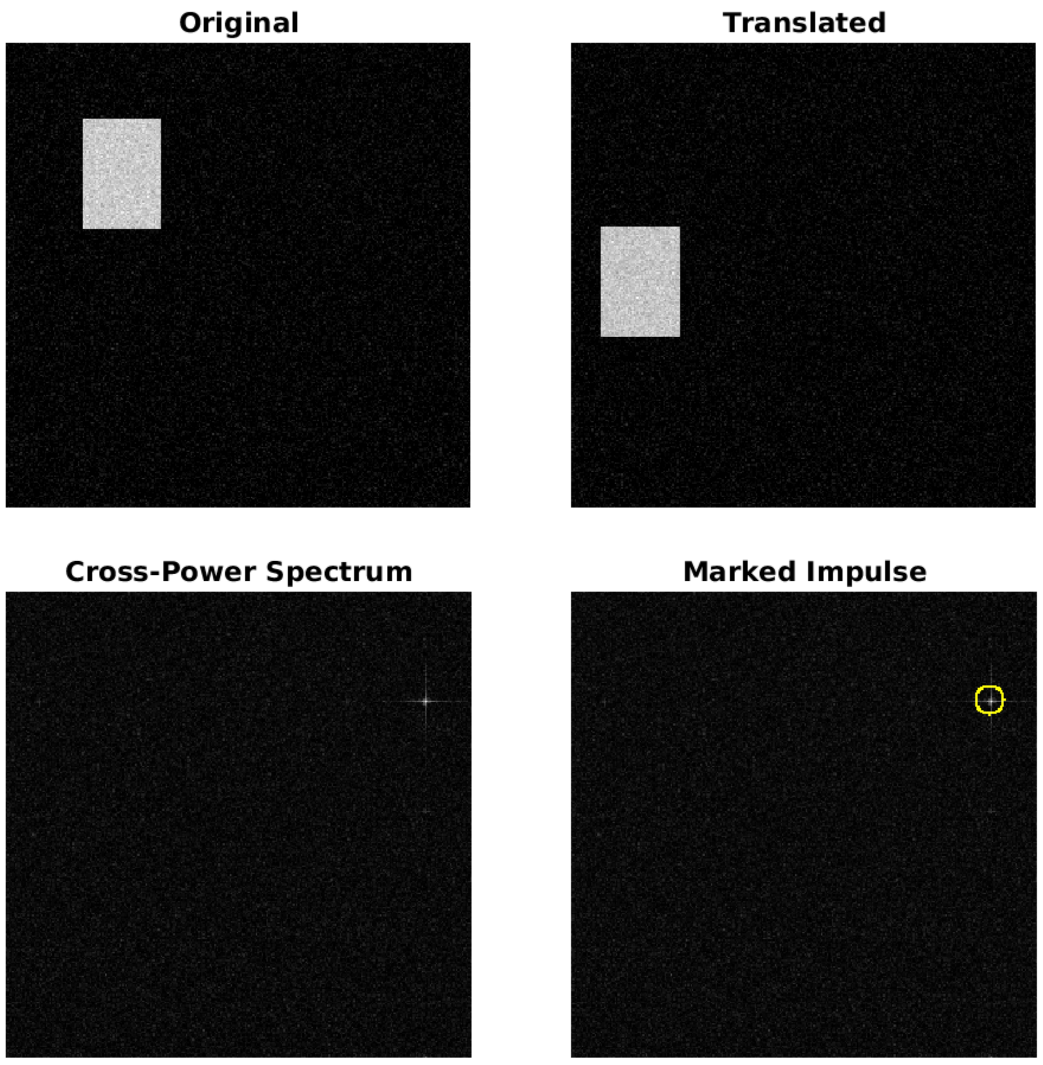


Figure 5.2: Image Registration with Noise

4. Cross Power Spectrum

Here we have attached the images corresponding to the log of the cross power spectrum in both the cases. Precisely, if f is the power spectrum, then we have plotted $log(1+|f|)$. Naturally, this should be a constant signal of value $log(2)$. There are some black spots visible in the power spectrum. These correspond to 0. They arise due to an approximation that we made in the code in section 1, in line 12, where we add e^{-50} to the denominator. On the other hand, due to precision, some points in the numerator are so small, that they reduce the result to 0. Hence, we can see occasional zeros scattered in the plot below.

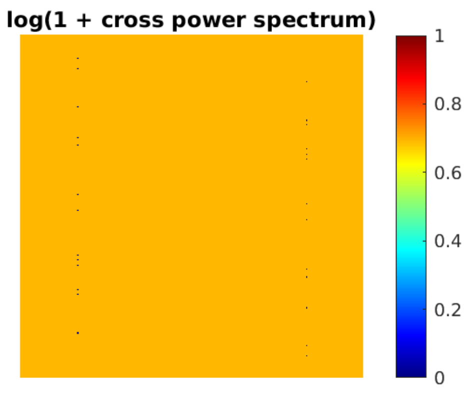


Figure 5.3 (a): Log of cross power spectrum without noise

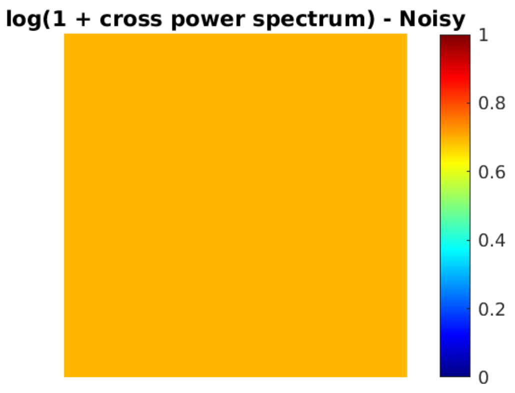


Figure 5.3 (b): Log of cross power spectrum with noise

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

5. Time Complexity Analysis

For an image of size NxN, the FFT algorithm takes $\mathcal{O}(n^2 logn)$ (row-wise/column-wise FFT takes $\mathcal{O}(nlogn)$, and this is done for each of the n rows and columns). The next operation of element wise sum and division take $\mathcal{O}(n^2)$, and the inverse fourier transform again takes $\mathcal{O}(n^2 logn)$.
Hence the overall time complexity is $\mathcal{O}(n^2 logn) + \mathcal{O}(n^2) + \mathcal{O}(n^2 logn) = \mathcal{O}(n^2 logn)$

On the other hand, the pixel wise comparison will take $\mathcal{O}(n^4)$. Basically, we will compute the translated image for every possible translation combination. For a NxN image, there are a total of N^2 translation possibilities. For a given translation, the pixelwise similarity computation will again take $\mathcal{O}(n^2)$.
Hence the overall time complexity of the brute force algorithm is $\mathcal{O}(n^4)$

6. Correcting Rotation

The paper has has broken this into two cases - rotation with and without scaling.
First we consider rotation without scaling. Let $f_2(x, y)$ be the translated and rotated replica of image $f_1(x, y)$, we have

$$f_2(x, y) = f_1(xcos\theta_0 + ysin\theta_0 - x_0, -xsin\theta_0 + ycos\theta_0 - y_0)$$

We know that a translation changes the phase but not the magnitude. Also, the magnitude is rotation invariant. As mentioned in the paper, using rotation property of the Fourier transform, we get,

$$F_2(\xi, \eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} M_1(\xi cos\theta_0 + \eta sin\theta_0, -\xi sin\theta_0 + \eta cos\theta_0)$$

Hence the magnitudes M1, M2 are given as follows:

$$M_2(\xi, \eta) = M_1(\xi cos\theta_0 + \eta sin\theta_0, -\xi sin\theta_0 + \eta cos\theta_0)$$

Since we already know how to compute translation between images, we can convert the above form to polar coordinates, so that the angles are represented in the form of translation.

$$M_2(\rho, \theta) = M_1(\rho, \theta - \theta_0)$$

With the algorithm and code used in section 1, we can easily compute the rotation.

Now, if we want to consider scaling as well, we use logarithmic scale. Following a similar procedure, we get, (from eq 17 in the paper)

$$M_2(\rho, \theta) = M_1(\rho/a, \theta - \theta_0)$$
$$M_2(log\rho, \theta) = M_1(log\rho - loga, \theta - \theta_0)$$
$$M_2(\xi, \theta) = M_1(\xi - d, \theta - \theta_0)$$

where,

$$\xi = log\rho$$
$$d = loga$$

This can be again solved using the formula and algorithm used in section I (for translation).

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

7. Usage of Code

- Simply run the **myMainScript.m**. This will produce and save all the required plots, and print out the translation values in both the pure and noisy case.

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Question 6

Formula:

$$F(u,v) = \sum_{x=0}^{W_1-1}\sum_{y=0}^{W_2-1} f(x,y)exp(-j2\pi(\frac{ux}{W_1}+\frac{vy}{W_2}))$$

$$F_{k1}(u,v) = -6exp(-j2\pi\frac{\lfloor N/2\rfloor u + \lfloor N/2\rfloor v}{N}) + \sum_{x=\lfloor N/2\rfloor-1}^{\lfloor N/2\rfloor+1} exp(-j2\pi(\frac{ux+v(\lfloor N/2\rfloor)}{N})) + \sum_{y=\lfloor N/2\rfloor-1}^{\lfloor N/2\rfloor+1} exp(-j2\pi(\frac{u(\lfloor N/2\rfloor)+vy}{N}))$$

$$F_{k2}(u,v) = 9exp(-j2\pi\frac{\lfloor N/2\rfloor u + \lfloor N/2\rfloor v}{N}) - \sum_{x=\lfloor N/2\rfloor-1}^{\lfloor N/2\rfloor+1}\sum_{y=\lfloor N/2\rfloor-1}^{\lfloor N/2\rfloor+1} exp(-j2\pi(\frac{ux+vy}{N}))$$

Code Snippet:

```
N = 201;
F1 = zeros(N,N);
F2 = zeros(N,N);
for u = 1:201
    for v = 1:201
        for x = 99:101
            for y = 99:101
                if x==y && x==100
                    F1(u,v) = F1(u,v)-4*exp(-1i*2*pi*((u-1-100)*x/N+(v-1-100)*y/N));
                    F2(u,v) = F2(u,v)+8*exp(-1i*2*pi*((u-1-100)*x/N+(v-1-100)*y/N));
                else
                    if x~=y && (x+y)~=(N-1)
                        F1(u,v)= F1(u,v)+exp(-1i*2*pi*((u-1-100)*x/N+(v-1-100)*y/N));
                    end
                    F2(u,v) = F2(u,v)-exp(-1i*2*pi*((u-1-100)*x/N+(v-1-100)*y/N));
                end
            end
        end
    end
end
[U,V]=meshgrid(-100:100,-100:100);
lF1 = log(abs(F1)+1);
figure('Name','k1-2d');imshow(lF1,[min(lF1(:)) max(lF1(:))]);colormap('jet');colorbar
figure('Name','k1-3d'); surf(U,V,lF1); colormap('jet'); colorbar;
lF2 = log(abs(F2)+1);
figure('Name','k2-2d');imshow(lF2,[min(lF2(:)) max(lF2(:))]);colormap('jet');colorbar
figure('Name','k2-3d'); surf(U,V,lF2); colormap('jet'); colorbar;
```

Results:

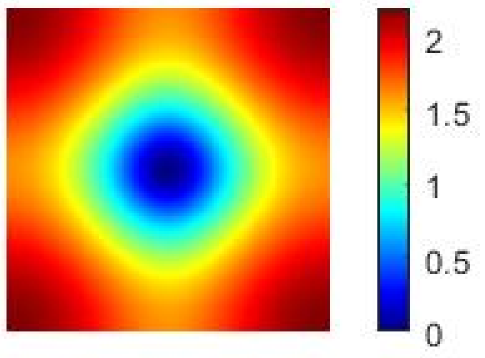


Figure 6.1: N,N-point DFT on **k1** (imshow)

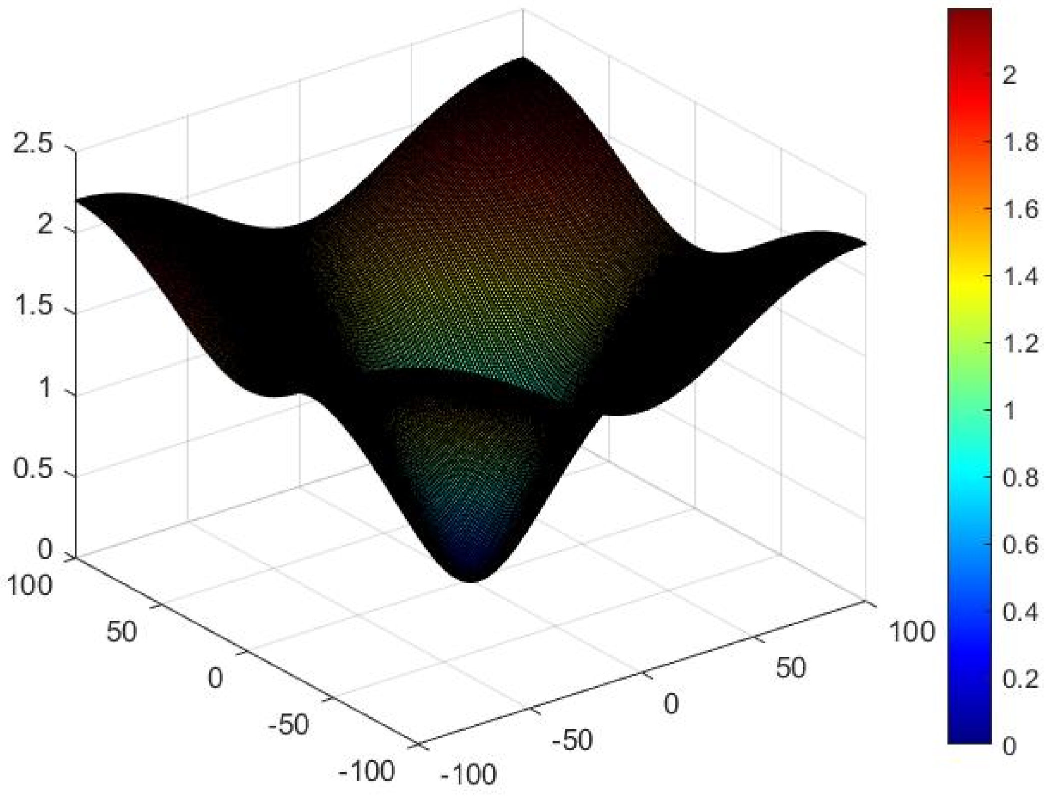


Figure 6.2: N,N-point DFT on **k1** (surf)

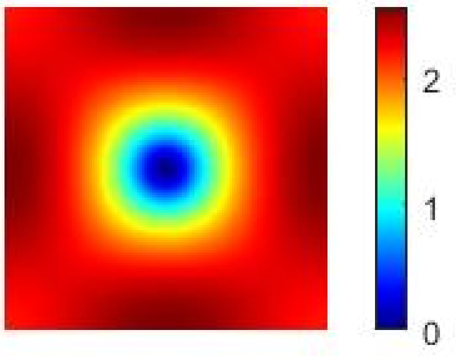


Figure 6.3: N,N-point DFT on **k2** (imshow)

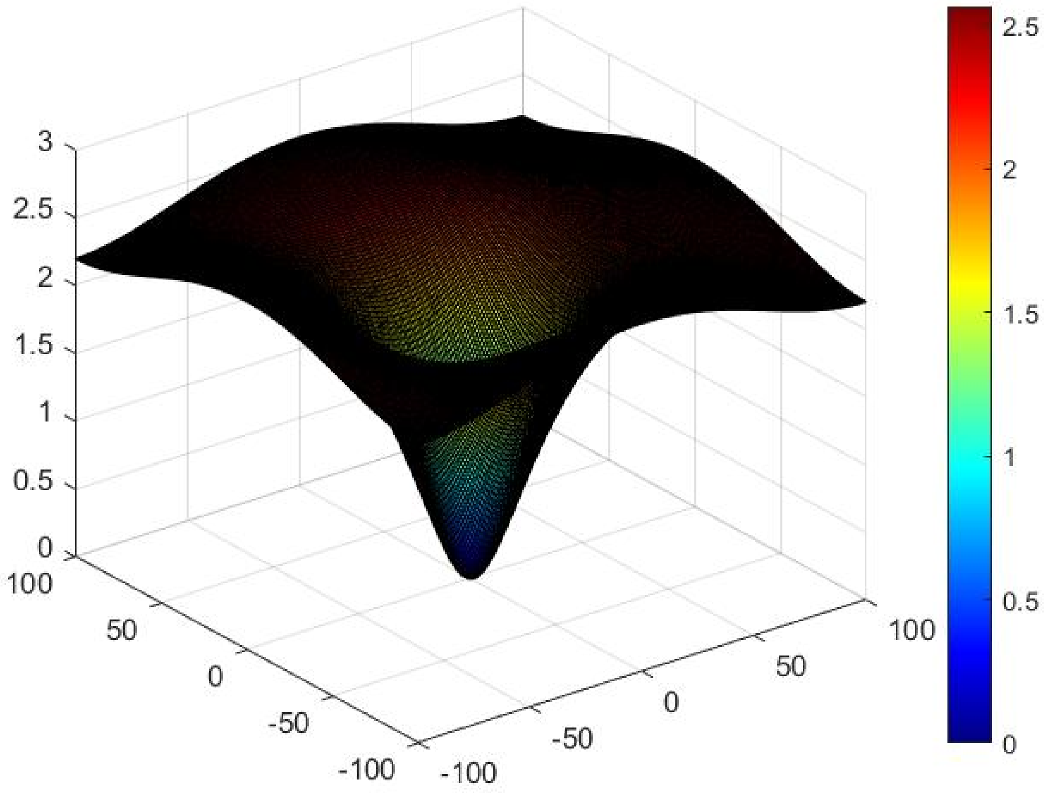


Figure 6.4: N,N-point DFT on **k2** (imshow)

Difference in Fourier Transforms of k1 and k2

Contours for k2 are roughly square (parallel to u and v axes) whereas for k1 they are roughly rotated squares. This can be inferred from the structure of kernels k1 and k2. In k2 we can visualise a square formed by connecting coordinates with value -1. We know that horizontal lines in x-y space result in vertical lines in u-v space on Fourier transformation and similarly vertical lines are translated to horizontal lines. Thus the Fourier transform will result in some square shaped contours parallel to u and v axes. In k1 we can visualise a rotated square formed by connecting coordinates with value 1. Using rotation transformation in DFT we know that the Fourier transform will result in some rotated square shaped contours.