Practical 5 Part 2 Fourier filter applications

March 2, 2021

Jaya Parmar, MATLAB filenames catstripes.m, findtarget.m, removerain2.m

1 Introduction

Our aim is to apply Fourier Filter on three different images. We will divide the report in three sections to show these three applications

2 Method

2.1 Cat stripes

We start by reading in the images and converting it to double and normalize it to 0-1 range.

$$i = double(imread('catStripes.jpg'))/255 \tag{1}$$

See Figure 1 in Results section for the original image.At this moment we donot see the cat in the image. This is due to the vertical lines which is the noise in the image.

Thereafter we convert it to grayscale image and

$$i = rgb2gray(i) \tag{2}$$

transform it to fourier domain.

$$ft = fft2(i) \tag{3}$$

We also shift zero frequency component to the center of spectrum.

$$ft = fftshift(ft) \tag{4}$$

See Figure 2 in Results section for the resulting image.

$$figure(2), imshow(log(1 + abs(ft)), []), impixelinfo \tag{5}$$

This image has two strongest lines vertically on both sides of the center. It also has a horizontal strong line passing across the center. These are the periodic noise lines. We need to remove them.

We will create a mask using 'roipoly' function in Matlab. This function creates a polygon mask on the image, setting pixels inside the polygon region of interest (roi) to 1 and pixels outside the polygon as zero.

Since there are three strong lines we need three 'roipoly' masks.

Let us start by masking the first vertical line with the help of its coordinates on the original image i. c1 and r1 are its column and row coordinates as shown below.

$$c1 = [172\ 179\ 179\ 172]; r1 = [1\ 1\ 458\ 458]; BW1 = roipoly(i, c1, r1); \tag{6}$$

Similarly c2 and r2 are the coordinates of the second vertical line on the original image as below

$$c2 = [284\ 290\ 290\ 284]; r2 = [1\ 1\ 458\ 458]; BW2 = roipoly(i, c2, r2); \tag{7}$$

and c3 and r3 are the coordinates of the horizontal line on the original image

$$c3 = [1\ 462\ 462\ 1]; r3 = [226\ 226\ 232\ 232]; BW3 = roipoly(i, c3, r3); \tag{8}$$

These coordinates are picked with the help of impixelinfo on the Figure 2.

Figure 5 in the Results section shows the resulting regions BW1, BW2 and BW3.

At this point, we need to make two changes in Figure 5.

1. First we want to let the bright spot periodic noise at the central zero frequency pass through.

2. Second,we want to block the BW1, BW2 and BW3 region since these are noises.

For the first part, we open up the center by creating another roipoly mask with the center periodic noise co ordinates as below

$$midpolyx = [226\ 237\ 239\ 224]; midpolyy = [223\ 223\ 236\ 236]; polymid = roipoly(i, midpolyx, midpolyy) \tag{9}$$

(Use impixelinfo over Figure 2 to see these coordinates).

Since these coordinates lie on the horizontal line, we will make BW3 region zero on the polymid region.

$$BW3(polymid) = 0 \tag{10}$$

Now for the second part, we use the Matlab not operator ( ) as below.

$$BW1 = \ BW1; BW2 = \ BW2; BW3 = \ BW3 \tag{11}$$

This will make the pixel value of BW regions zero (black) and and rest of the regions pixel value 1 (white).

It is time to combine the three BW regions to make the final mask for the original image. We use and operator [&] to achieve this as below

$$BW = and(and(BW1, BW2), BW3) \tag{12}$$

Multiply the mask BW with the image in Fourirer domain and put in variable cat.

$$cat = BW. * ft; \tag{13}$$

This is our cat in frequency domain shown in Figure 4.

$$figure(4), imshow(log(1 + abs(cat)), []), impixelinfo \tag{14}$$

Using ifftshift and ifft2 functions from Matlab, we inverse the fftshift and fft2 to get the image in spatial domain.

These frequencies from frequency domain have real and complex part. We will use the real part and save it in variable klar.

$$klar = real(ifft2(ifftshift(cat))) \tag{15}$$

Also, the last step is to normalize klar so that we have the final image normalized. This is needed to compare to the original image which is also normalized.

$$klar = klar - min(klar(:)); \tag{16}$$

$$klar = klar./max(klar(:)); \tag{17}$$

The cat in spatial domain is shown in Figure 3 in the results section.

$$figure(3), imshow(log(1 + abs(klar)), []), impixelinfo \tag{18}$$

2.2 Finding target

Read in the search and target images below

$$bw = imread('final_search_image.tif'); m = imread('final_target_image.tif'); \tag{19}$$

To find the target coordinates in the search image, use impixelinfo on image variable bw.

For this we see the original and target image in Figure 1 and Figure 2 respectively.

$$figure(1), imshow(bw), title('Search\ image'), impixelinfo \tag{20}$$

$$figure(2), imshow(m), title('Target'), impixelinfo \tag{21}$$

The target has below coordinates in search image.

$$target = bw(122 : 131, 235 : 244) \tag{22}$$

This target is a 10X10 image and bw is 467X477 image. Before we start finding correlation between images, we need change the size of target to 467X477 so that it matches the search image in frequency domain.

Correlation is convolution kernel rotated by 180 degrees. So first we rotate the target twice by 90 degrees.

Also, we need to do this in fourier domain.

The target operation is *fft2(rot90(target,2),467,477))*

Thereafter we multiply the two images to get their correlation.

*fft2(bw).*fft2(rot90(target,2),467,477)*

To see the results in spatial domain we need to do inverse fft2 and take the real part.

$$C = real(ifft2(fft2(bw). * fft2(rot90(target, 2), 467, 477))); \tag{23}$$

Resulting correlation image is shown in Figure 3.

$$figure(3), imshow(C, []), title('Correlated\ images'), impixelinfo \tag{24}$$

To view the location of the target in the search image, we find the maximum pixel value and then define a threshold value that is less than maximum.

$$maxC = max(max(C)); \tag{25}$$

Value of maxC in our case is 1775894. We choose a threshold value = 1775800.

A thresholded image is created for pixel values greater than the threshold value as below.

$$D = C > thresh \tag{26}$$

Thresholded image is shown in Figure 4. Observe the tiny white spot in the black ground.

$$figure(4), imshow(D), title('Thresholded\ image'), impixelinfo \tag{27}$$

We dilate the white spot to make it bigger.
A structuring element will be needed. Since the target is square we choose square shaped structuring element. Let us keep the size of structuring element equal to the size of target image m i.e. 24.

$$se = strel('square', 24) \tag{28}$$

and dilate

$$E = imdilate(D, se) \tag{29}$$

See the dilated target in Figure 5.

$$figure(5), imshow(E), title('Dilated\ target'), impixelinfo \tag{30}$$

Final step is to see the overlay the dilated target over the search image. We use imfuse function from Matlab

$$final = imfuse(bw, E) \tag{31}$$

See final image in Figure 6

$$figure(6), imshow(final), title('Fused\ images'), impixelinfo \tag{32}$$

2.3 Remove rain2

Read the regn2 image and convert it to double to normalize it.

$$i = double(imread('regn2.jpg'))/255 \tag{33}$$

The image in spatial domain is shown in Figure 1.

$$figure(1), imshow(i), impixelinfo \tag{34}$$

Convert the image to frequency domain and shift zero frequency component to the center of spectrum

$$ft = fft2(i) \tag{35}$$

The image in frequency domain is shown in Figure 2.

$$figure(2), imshow(log(1 + abs(ft)), []), impixelinfo \tag{36}$$

To hide the rain, we need to mask the image using 'roipoly'. Let us define the x and y coordinates of the masking polygon using impixelinfo from Figure 2.

$$c = [1\ 793\ 793\ 1]; r = [248\ 315\ 333\ 271]; BW = roipoly(i, c, r) \tag{37}$$

BW is the rain masking region from the frequency domain image.

We want to let the center bright spot periodic noise at the central zero frequency pass through and block the BW region since these are noises.

Choosing the center bright spot coordinates from Figure 2 and making them zero.

$$midpolyx = [387\ 412\ 412\ 384]; midpolyy = [284\ 286\ 300\ 296]; polymid = roipoly(BW, midpolyx, midpolyy) \tag{38}$$

$$BW(polymid) = 0 \tag{39}$$

Finally inverting the BW region to use it for masking the original image.

$$BW = \ BW \tag{40}$$

Multiplying this mask BW with the image in frequency domain ft and storing in variable 'norain',

$$norain = BW. * ft \tag{41}$$

will show the image with no rain. See Figure 4 for results in frequency domain.

$$figure(4), imshow(log(1 + abs(norain)), []), impixelinfo \tag{42}$$

To see the image in spatial domain, inverse the fftshift and fft2 operations using ifftshift and ifft2 respectively.

$$klar = real(ifft2(ifftshift(norain))) \tag{43}$$

Finally normalize the spatial image 'klar' as below

$$klar = klar - min(klar(:)) \tag{44}$$

$$klar = klar./max(klar(:)) \tag{45}$$

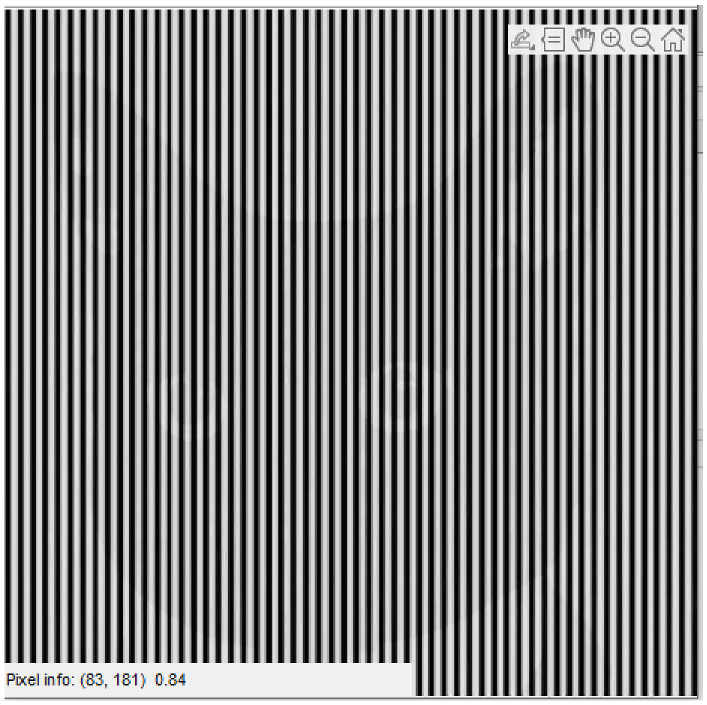
See the final image in Figure 3 in results section.

$$figure(3), imshow(log(1 + abs(klar)), []), impixelinfo \tag{46}$$

3 Results

3.1 Cat stripes

Figure 1



Pixel info: (83, 181) 0.84

Figure 2

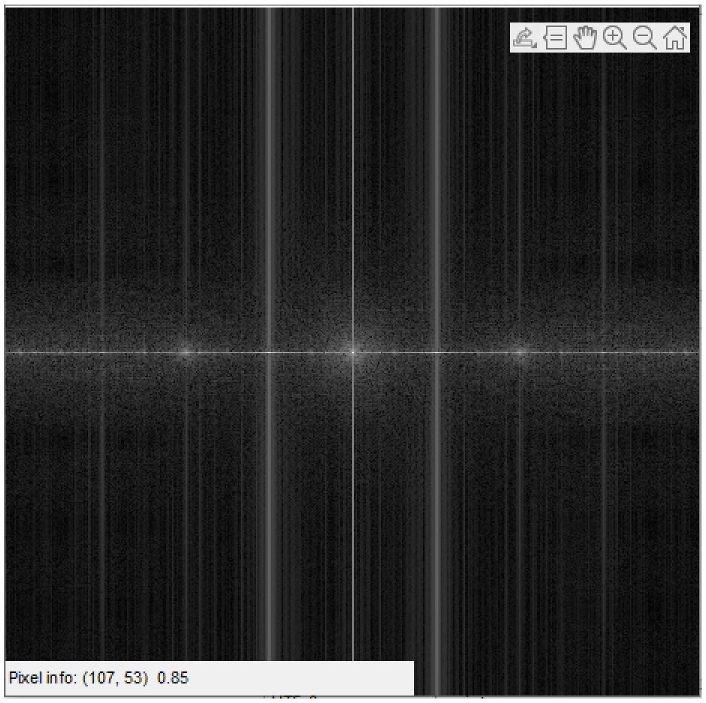

Pixel info: (107, 53) 0.85

Figure 3

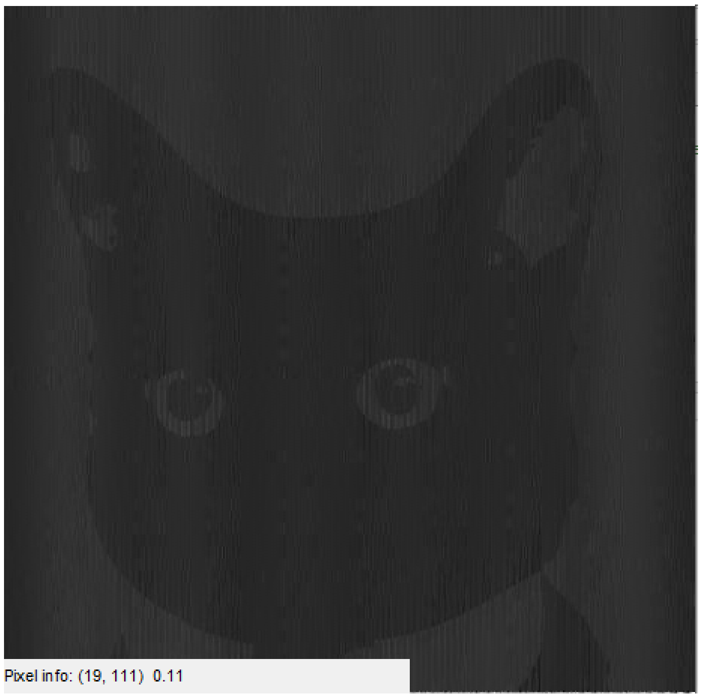


Figure 4

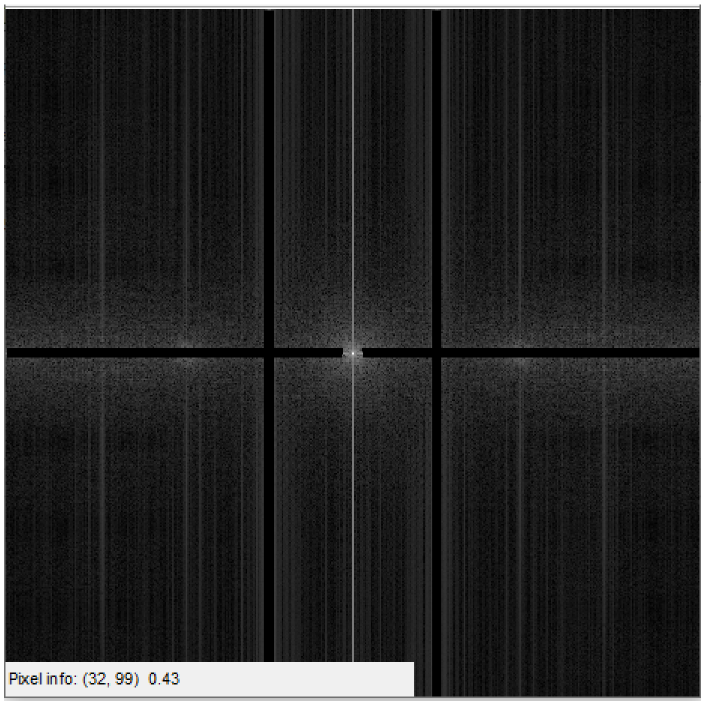
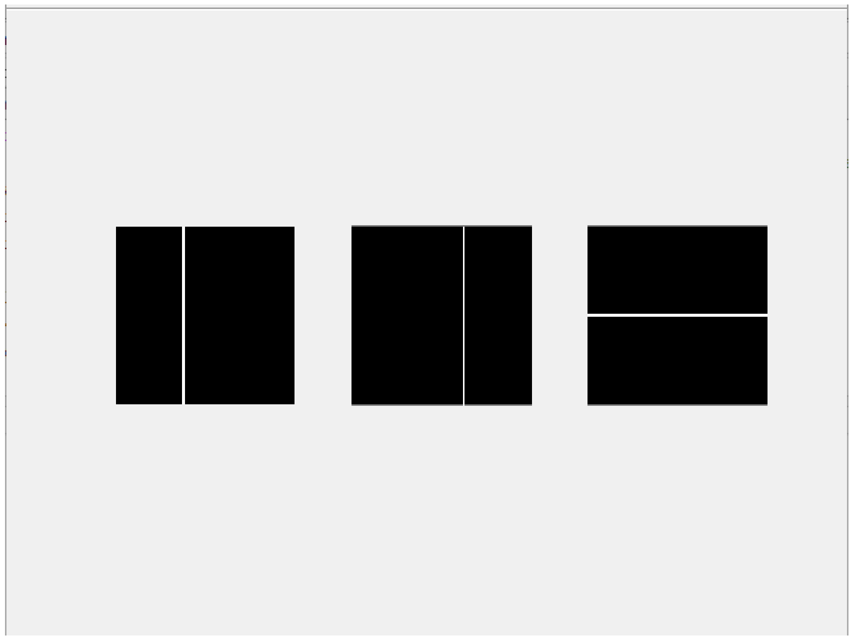

Pixel info: (32, 99) 0.43

Figure 5



3.2 Finding target

Figure 1

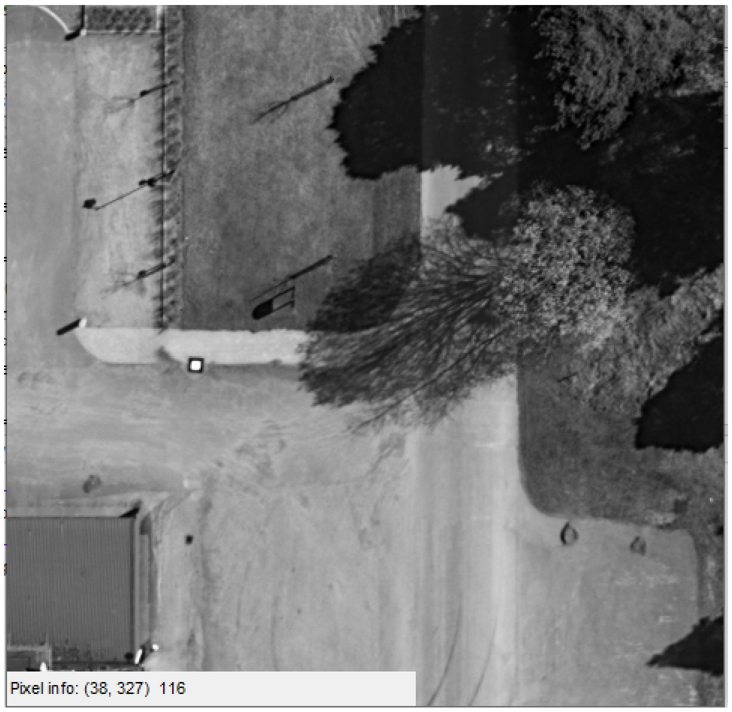


Figure 2

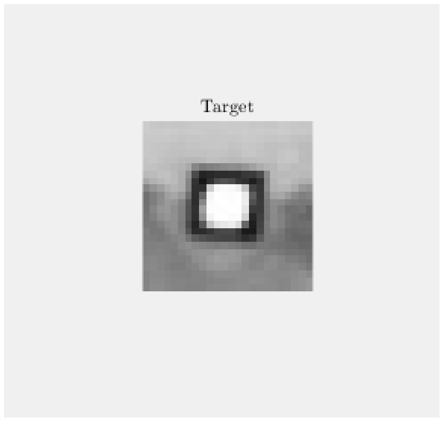


Figure 3

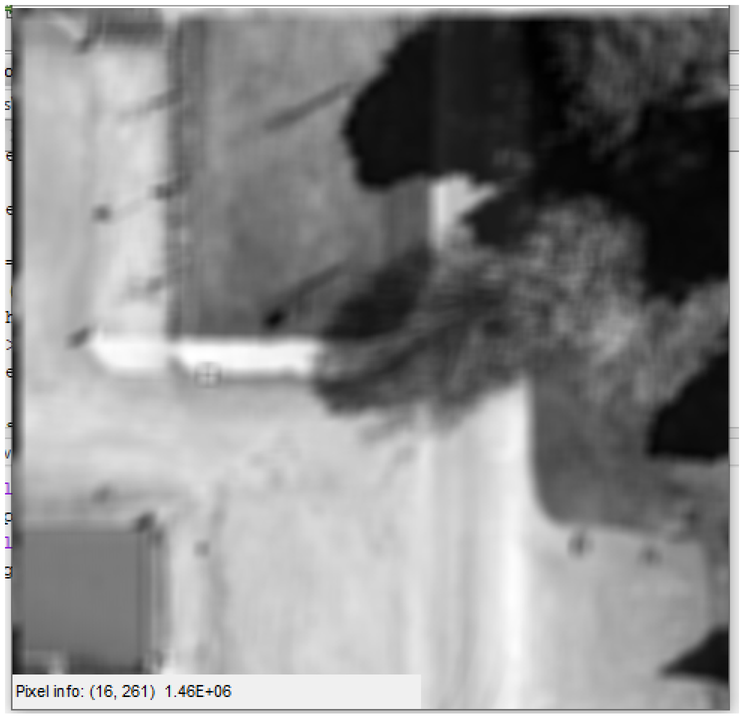


Figure 4

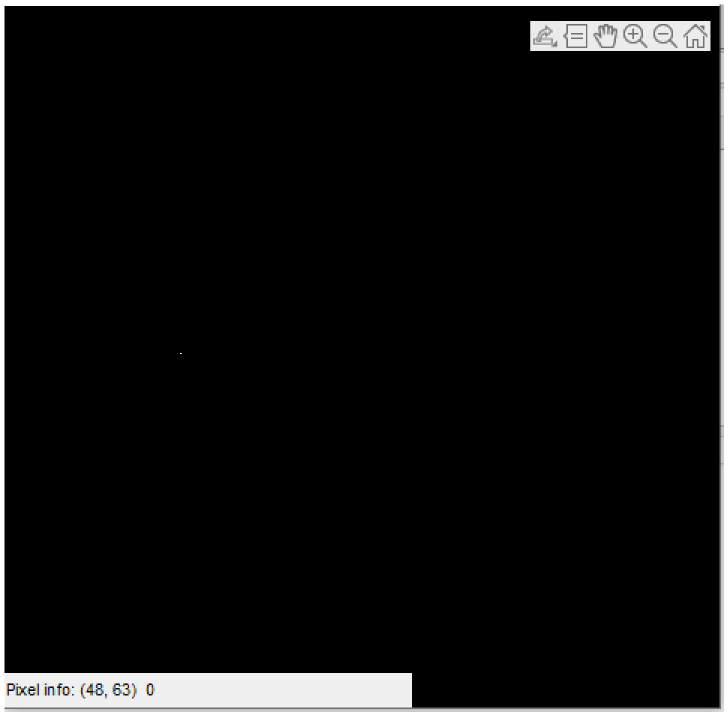

Pixel info: (48, 63) 0

Figure 5

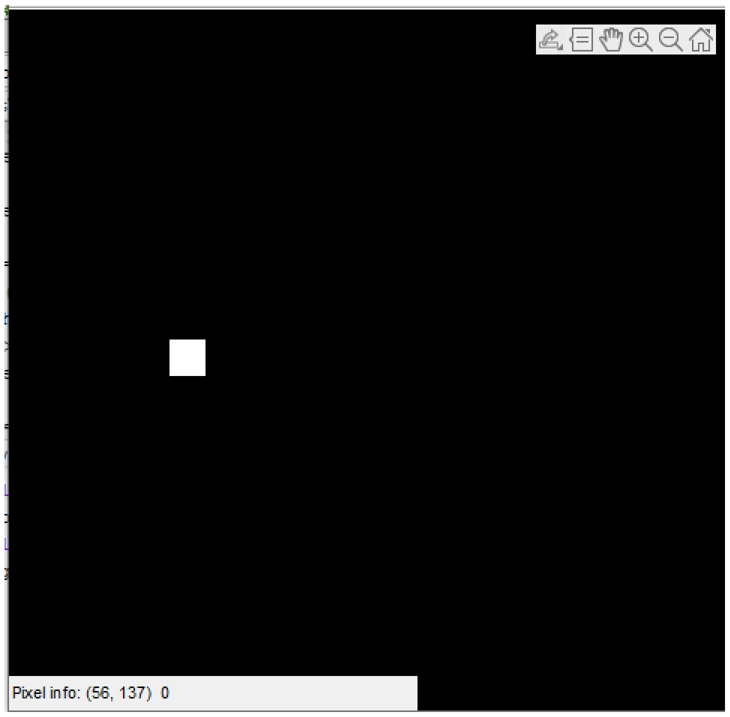
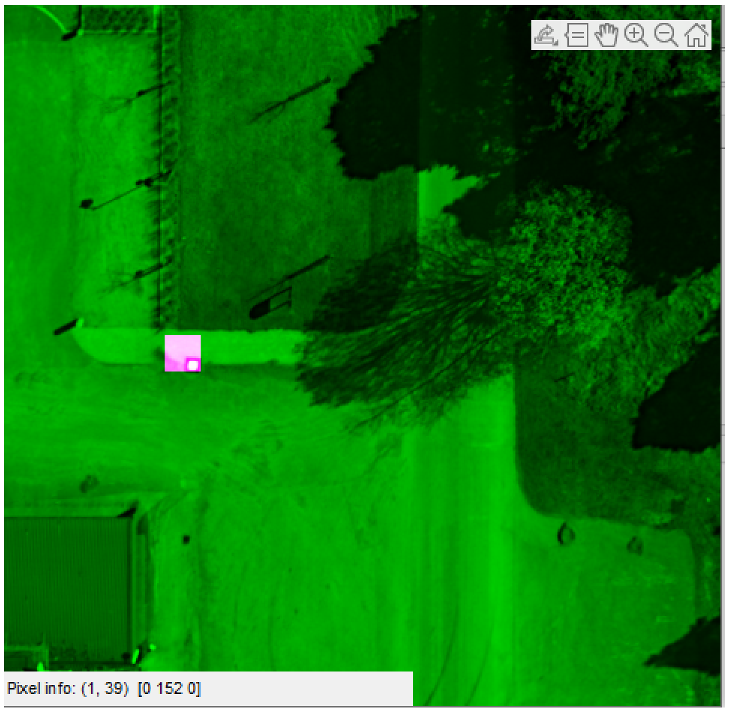

Pixel info: (56, 137) 0

Figure 6


Pixel info: (1, 39) [0 152 0]

3.3 Remove rain2

Figure 1

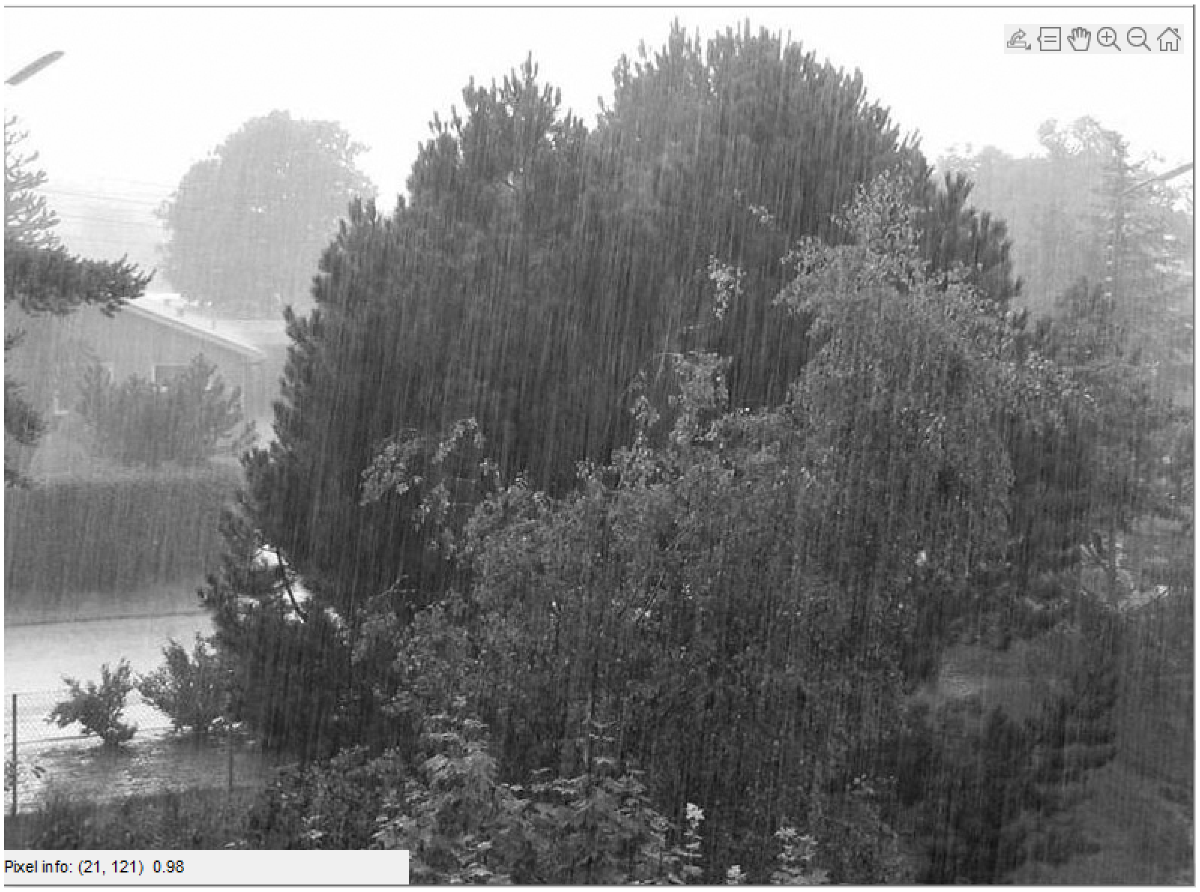


Figure 2

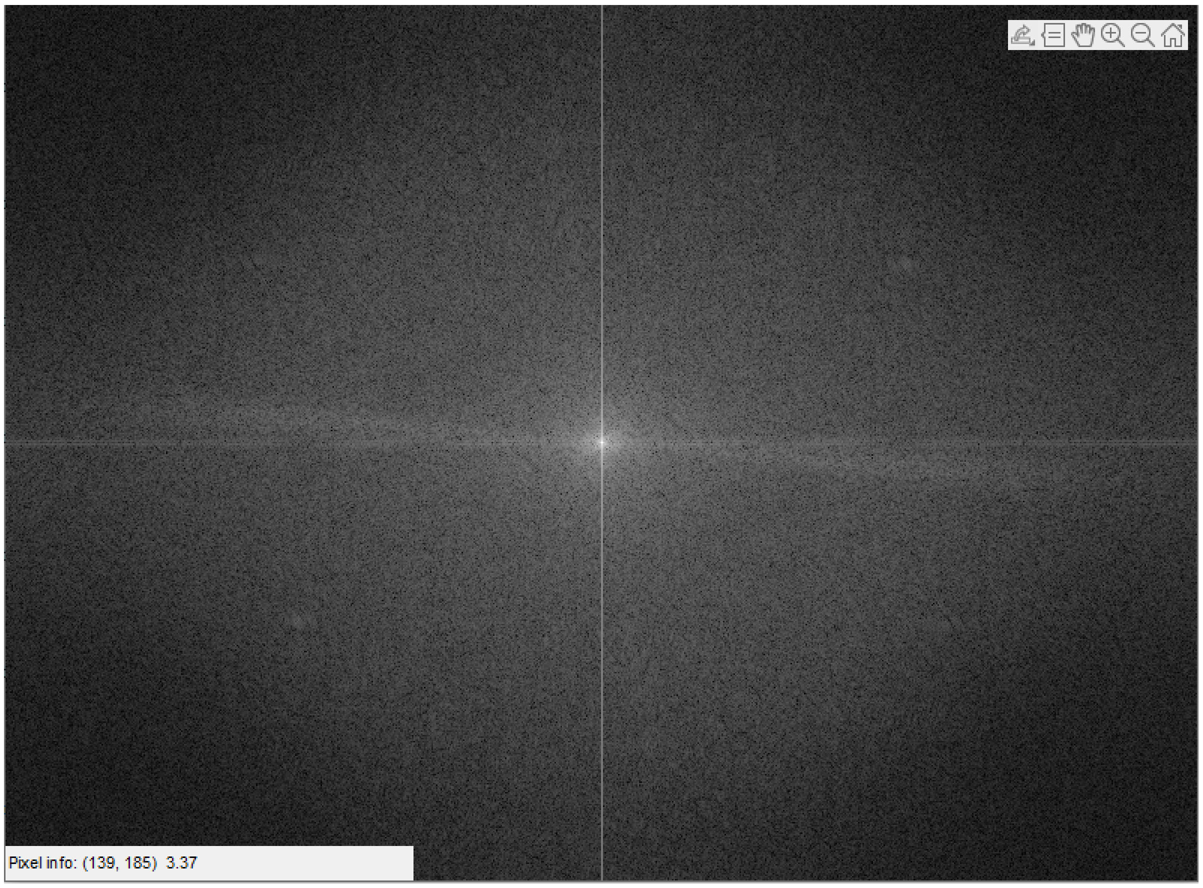


Figure 3

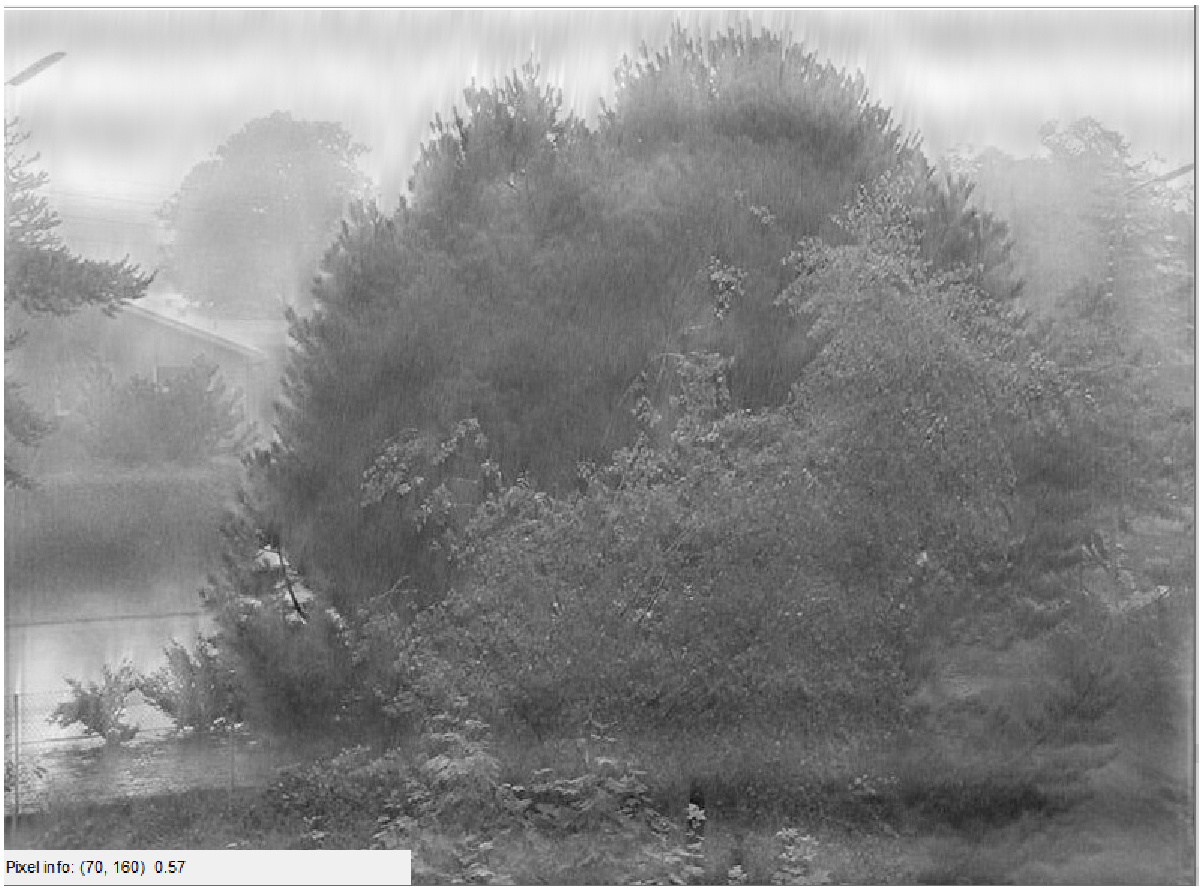
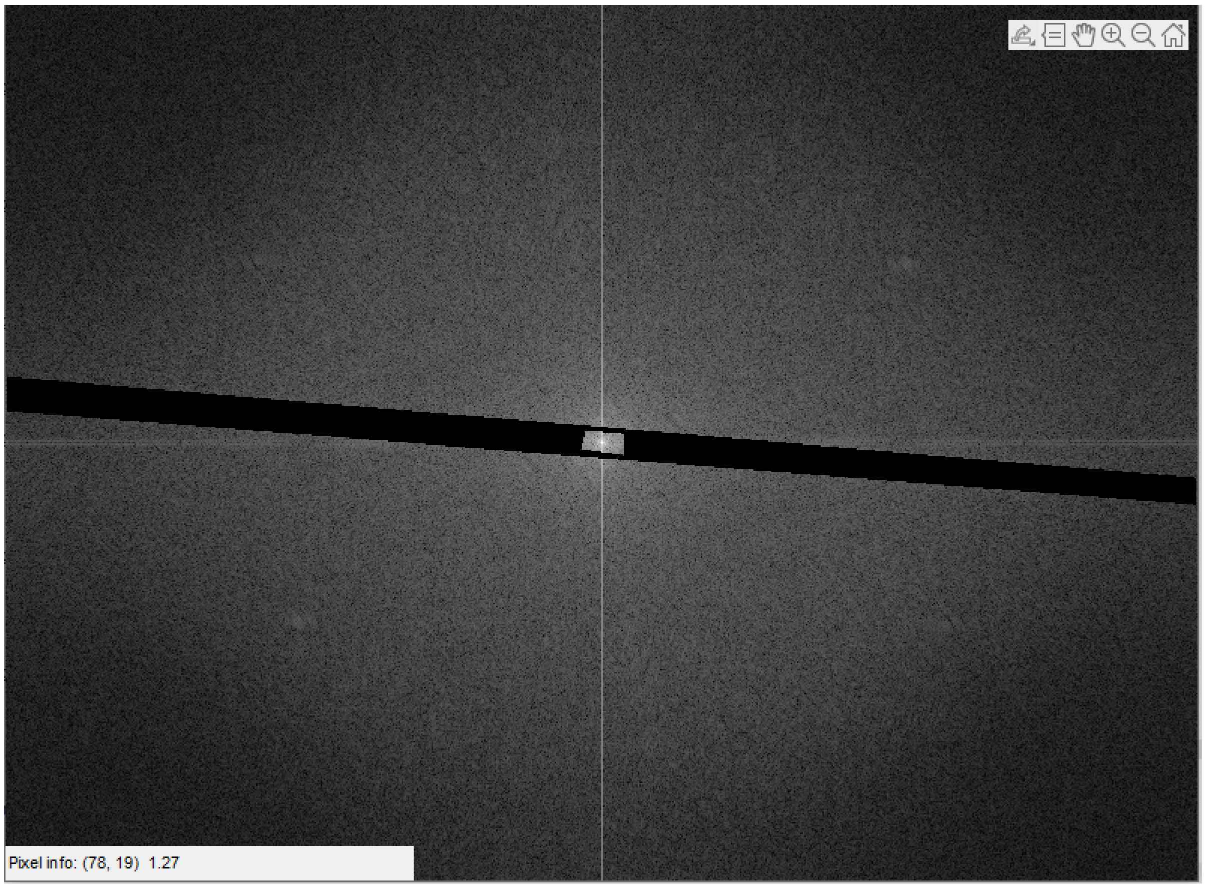


Figure 4



4 Discussion

Noise removal is relatively easier in the fourier domain as frequencies to be masked are represented by their pixel coordinates in the image. Also, the filter size is same as the image size which minimizes the number of computations compared to filtering in spatial domain.

5 Appendix

5.1 Cat stripes

```
set(0, 'defaulttextinterpreter','Latex');
i = double(imread('catStripes.jpg'))/255;
i = rgb2gray(i);
```

```
ft = fft2(i);
ft = fftshift(ft);

c1 = [172 179 179 172];
r1 = [1 1 458 458];
BW1 = roipoly(i,c1,r1);

c2 = [284 290 290 284];
r2 = [1 1 458 458];
BW2 = roipoly(i,c2,r2);

c3 = [1 462 462 1];
r3 = [226 226 232 232];
BW3 = roipoly(i,c3,r3);

figure(5),subplot(1,3,1),imshow(BW1); subplot(1,3,2),
   imshow(BW2),subplot(1,3,3),imshow(BW3);

midpolyx = [226 237 239 224];
midpolyy = [223 223 236 236];
polymid = roipoly(i,midpolyx,midpolyy);
BW3(polymid)=0;
BW1=~BW1;BW2=~BW2;BW3=~BW3;

%figure(6),subplot(1,3,1),imshow(BW1),subplot(1,3,2),
   imshow(BW2),subplot(1,3,3),imshow(BW3),impixelinfo;

BW = and(and(BW1,BW2),BW3);
%figure(7),imshow(BW);

cat = BW.*ft;
klar=real(ifft2(ifftshift(cat)));
klar=klar-min(klar(:));
klar=klar./max(klar(:));

figure(1),imshow(i),impixelinfo,
figure(2),imshow(log(1+abs(ft)),[]),impixelinfo,
figure(3),imshow(log(1+abs(klar)),[]),impixelinfo,
figure(4),imshow(log(1+abs(cat)),[]),impixelinfo;
```

5.2 Finding target

```
set(0, 'defaulttextinterpreter','Latex');
bw = imread('final_search_image.tif');
m = imread('final_target_image.tif');
```

```
figure(1),imshow(bw),title('Search image'),impixelinfo
    ;
figure(2),imshow(m),title('Target'),impixelinfo;

target = bw(122:131,235:244);

C = real(ifft2(fft2(bw).*fft2(rot90(target,2),467,477)
    ));

figure(3),imshow(C,[]),title('Correlated images'),
    impixelinfo;

maxC = max(max(C));
%disp(num2str(maxC,'%.2f'));
thresh = 1775800;
D = C>thresh;
figure(4),imshow(D),title('Thresholded image'),
    impixelinfo;

se = strel('square',24);
E = imdilate(D,se);
figure(5),imshow(E),title('Dilated target'),
    impixelinfo;

final=imfuse(bw,E);
figure(6), imshow(final),title('Fused images'),
    impixelinfo;
```

5.3 Remove rain2

```
set(0, 'defaulttextinterpreter','Latex');
i = double(imread('regn2.jpg'))/255;

ft = fft2(i);
ft = fftshift(ft);

c = [1 793 793 1];
r = [248 315 333 271]
BW = roipoly(i,c,r);
%figure(5),imshow(BW),impixelinfo;

midpolyx = [387 412 412 384];
midpolyy = [284 286 300 296];
polymid = roipoly(BW,midpolyx,midpolyy);
BW(polymid)=0;
BW=~BW;
```

```
%figure(6),imshow(BW),impixelinfo;

norain = BW.*ft;

klar=real(ifft2(ifftshift(norain)));
klar=klar-min(klar(:));
klar=klar./max(klar(:));

figure(1),imshow(i),impixelinfo,
figure(2),imshow(log(1+abs(ft)),[]),impixelinfo,
figure(3),imshow(log(1+abs(klar)),[]),impixelinfo,
figure(4),imshow(log(1+abs(norain)),[]),impixelinfo;
```