Activity 2 (Part 2 of 2) - Properties and Applications of the 2D Fourier Transform

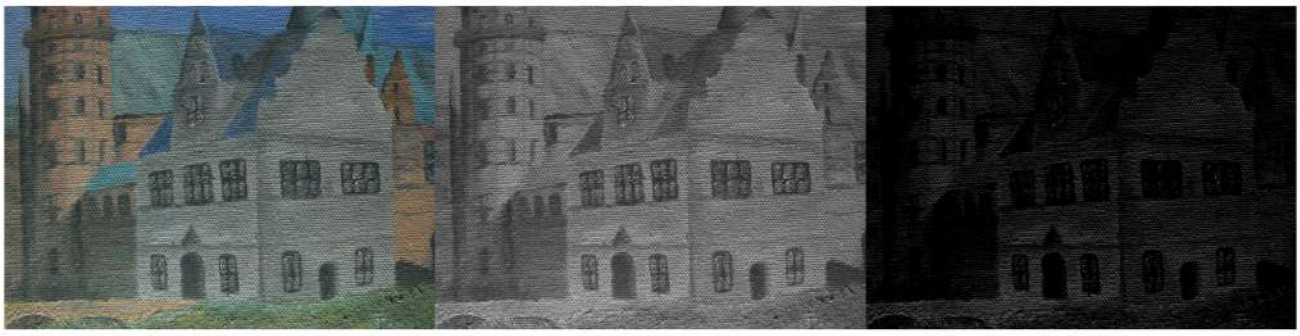
Jonabel Eleanor B. Baldres

2.2.1. Rotation Property of the FT

Due to unforeseen errors, activity 2.2.1 is found on a separate editor.

2.2.2 Application: Canvas Weave Modeling and Removal

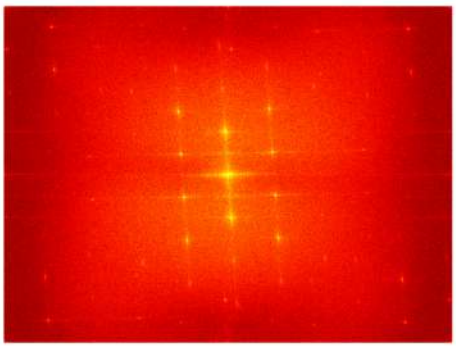
```
%read the image, convert the image into gray, subtract the mean of gray
%image to the gray image, and display the images of the threee
image_used = im2double(imread('185-8526_IMG.jpg'));
gray_image = rgb2gray(image_used);
mean_subtracted_image = gray_image - mean2(gray_image);
montage({image_used, gray_image, mean_subtracted_image}, 'size', [1 NaN])
```



```
%taking the FFT of the mean-subtracted image
FFT_meansub = fft2(mean_subtracted_image);
FFT_absmeansub = abs(FFT_meansub);
FFT_shifted = fftshift(FFT_meansub);
FFT_meansub_shifted = abs(FFT_shifted);
FFT_logarithmic = rescale(log(FFT_meansub_shifted + 1), 0,256);
```

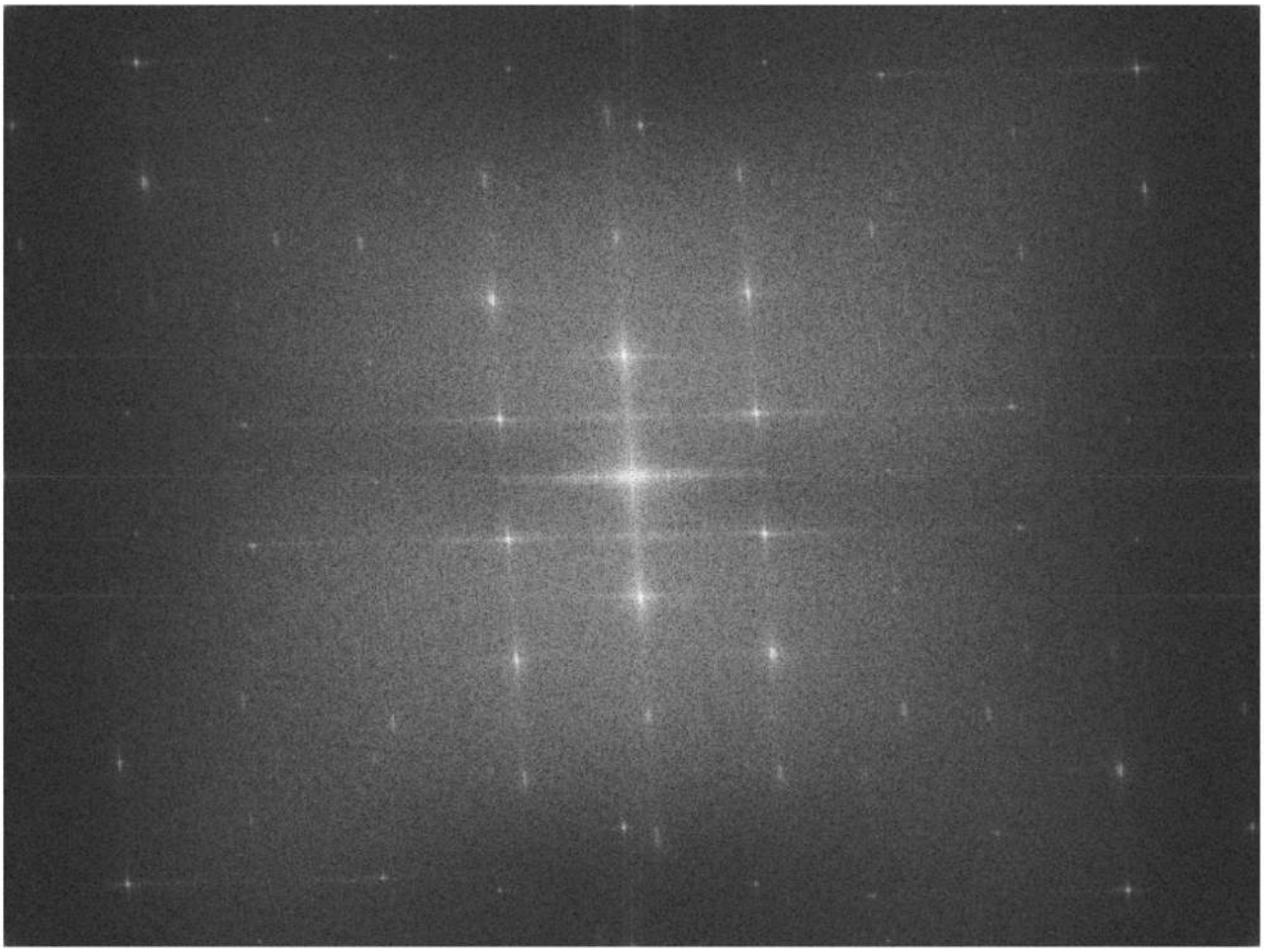
I was having a hard time using imagesc(FFT_logarithmic) directly for the succeeding parts so, I decided to first download the image and made sure that it has the same dimension and resolution as the painting image.

```
imagesc(FFT_logarithmic);
colormap('hot')
axis off;
axis image;
```



```
% Set the figure size
%set(gcf, 'Units', 'pixels');
%set(gcf, 'Position', [0, 0, 1280, 960]);

% Save the figure
%exportgraphics(gcf, '~/Desktop/FFT_logarithmic_hotmap.png', 'Resolution',
180);
```

```
% using the colormap image and making sure that its class is double and its
% image type is gray
colormap_image = im2double(imread('FFT_logarithmic_hotmap.png'));
resized_colormap_image = imresize(colormap_image, [960, 1280]);
colormap_image_gray = rgb2gray(resized_colormap_image);
imshow(colormap_image_gray)
```

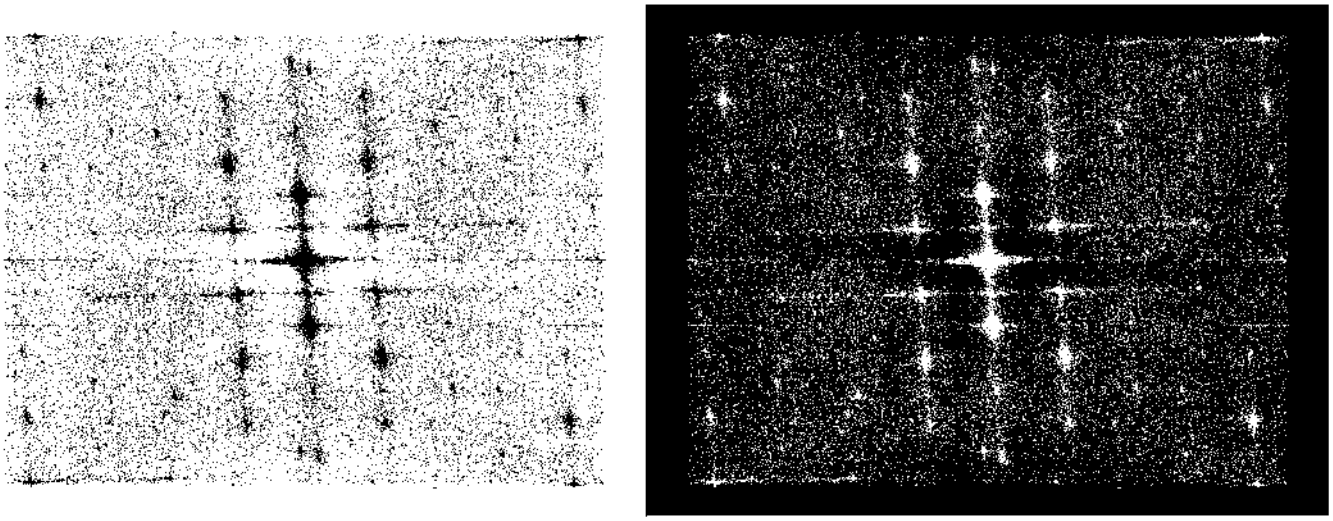


```
% imtool(colormap_image) : used to show the information of the image
```

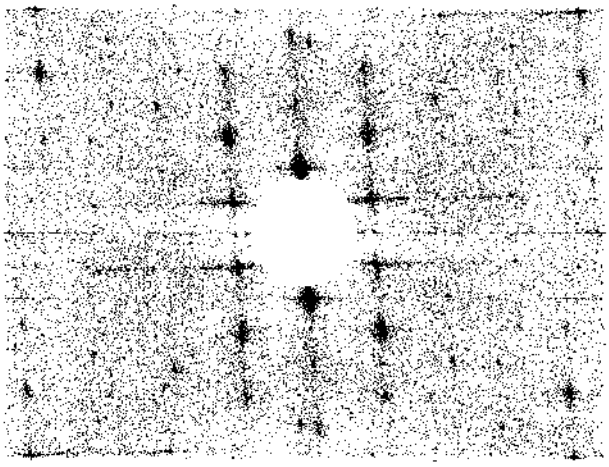
Now, the task is to create a filter mask such that the filter has a value of 0 at locations of the sinusoidal peaks. To do that, I first binarized the gray image of the peaks and applied adaptive thresholding with a bright foreground polarity. The bright foreground polarity is considered since the foreground objects are brighter than the background. The imcomplement function is used to invert the black and white pixels in the binary image. The initial 0 values will be changed to 1 and vice versa.

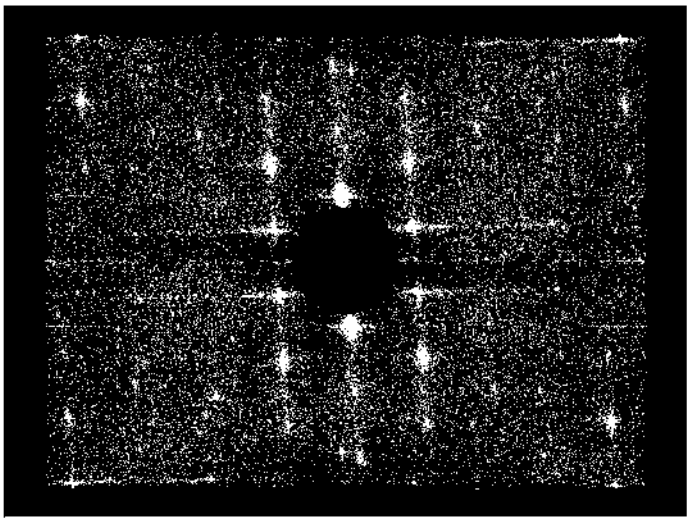
```
% the bright objects are 1 and the dark objects are 0
% this will be used later on to obtain the pattern of the canvas weave
masked =
im2double(imbinarize(colormap_image_gray,"adaptive","ForegroundPolarity","br
ight"));

%the bright objects are 0 and the dark objects are 1
%will be used to obtain the image of the painting with little to no
%presence of weave patterns
masked_edited = im2double(imcomplement(masked));
```

```
%comparing and contrast the masks produced
imshowpair(masked_edited, masked, 'montage')
```

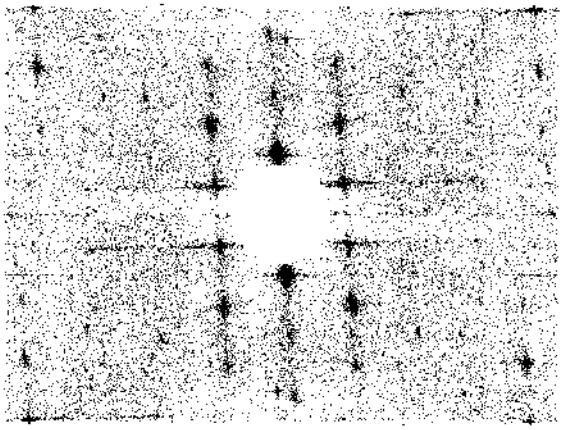


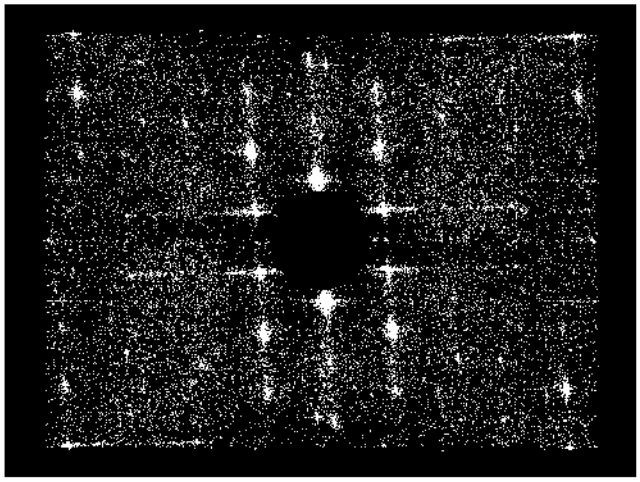
```
% Set the center and radius of the circle
cx = size(masked, 2)/2;   % x coordinate of the center of the circle
cy = size(masked, 1)/2;   % y coordinate of the center of the circle
radius = 100;  % radius of the circle in pixels

% Create a binary mask of the same size as the image
[x, y] = meshgrid(1:size(masked,2), 1:size(masked,1));
mask = hypot(x-cx, y-cy) <= radius;

% Apply the mask to the image
masked(mask) = 0;

masked_edited = im2double(imcomplement(masked));

%comparing and contrast the masks produced
imshowpair(masked_edited, masked, 'montage')
```





```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_fpol_daria.png', 'Resolution', 90);
```





The lines below show how the original image was divided into image with lesser weave patterns and image focusing on the weave pattern.

```
% obtaining the RGB values of the original image
[R,G,B] = imsplit(image_used);


filter_shift = fftshift(abs(masked_edited));

%multiplying the fftshift of the mask to the fft2 of the RGB values
filter_R = fft2(R) .* filter_shift ;
filter_G = fft2(G) .* filter_shift ;
filter_B = fft2(B) .* filter_shift ;

%obtaining the absolute value of the iift2
invR = abs(ifft2(filter_R));
invG = abs(ifft2(filter_G));
```

```
invB = abs(ifft2(filter_B));

Inew(:,:,1)= invR;
Inew(:,:,2)= invG;
Inew(:,:,3)= invB;

% the image produced is the image with clearer view of the painting


% to obtain only the weave patterns, we fft shift the masked variable
filter_shift_weave = fftshift(abs(masked));

filter_R_weave = fft2(R) .* filter_shift_weave ;
filter_G_weave= fft2(G) .* filter_shift_weave ;
filter_B_weave = fft2(B) .* filter_shift_weave ;


invR_weave = abs(ifft2(filter_R_weave ));
invG_weave = abs(ifft2(filter_G_weave ));
invB_weave= abs(ifft2(filter_B_weave ));

Inew_weave(:,:,1)= invR_weave;
Inew_weave(:,:,2)= invG_weave;
Inew_weave(:,:,3)= invB_weave;

montage({image_used, Inew,Inew_weave}, 'size', [1 NaN])
```

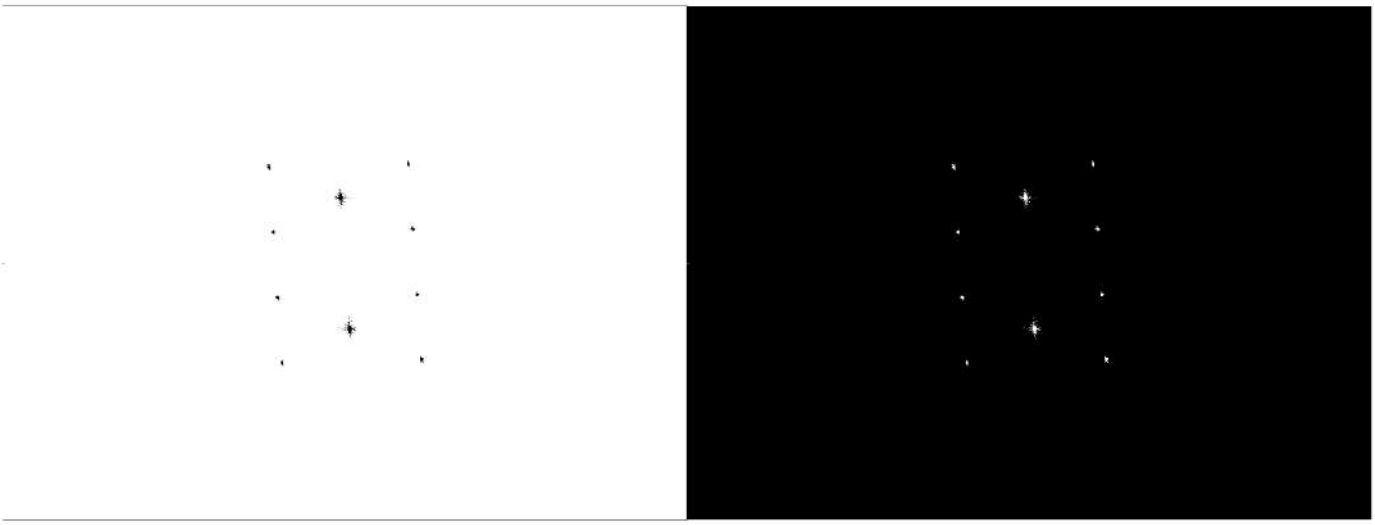
```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_fpol_daria_results.png',
'Resolution', 90);
```

I decided to find for more ways to accomplish the task. In the succeeding codes below, I thresholded the gray colormap created such that when the value is less than or greater than 0.75, the value becomes 0. This is to remove any unneccesary points caused by the noise.

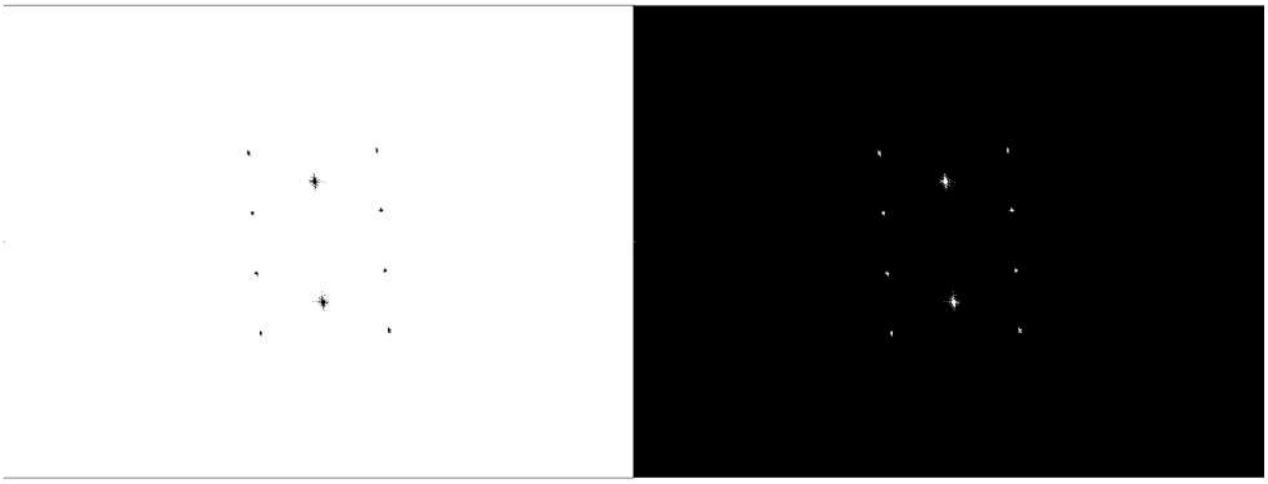
```
colormap_image_thresholded =
im2double(imread('FFT_logarithmic_hotmap.png'));
resized_colormap_image_thresholded = imresize(colormap_image_thresholded ,
[960, 1280]);
colormap_image_gray_thresholded =
rgb2gray(resized_colormap_image_thresholded);
%thresholding the colormap_image_gray such that when its value is <= 0.75
%it becomes 0
colormap_image_gray_thresholded(colormap_image_gray_thresholded <= 0.75) =
0;
```

```
% Set the center and radius of the circle
```

```
cx = size(colormap_image_gray_thresholded, 2)/2;   % x coordinate of the
center of the circle
cy = size(colormap_image_gray_thresholded, 1)/2;   % y coordinate of the
center of the circle
radius = 100;  % radius of the circle in pixels

% Create a binary mask of the same size as the image
[x, y] = meshgrid(1:size(colormap_image_gray_thresholded,2),
1:size(colormap_image_gray_thresholded,1));
mask = hypot(x-cx, y-cy) <= radius;

% Apply the mask to the image
colormap_image_gray_thresholded(mask) = 0;


masked_thresholded =
imbinarize(colormap_image_gray_thresholded ,"adaptive","ForegroundPolarity",
"bright");
masked_edited_thresholded  = imcomplement(masked_thresholded );
imshowpair(masked_edited_thresholded , masked_thresholded , 'montage')
```



```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_thresholding_daria.png',
'Resolution', 90);
```



```
[R,G,B] = imsplit(image_used);

%image of the painting without the weave patterns

filter_shift_thresholded  = fftshift(abs(masked_edited_thresholded ));

filter_R_thresholded  = fft2(R) .* filter_shift_thresholded  ;
filter_G_thresholded  = fft2(G) .* filter_shift_thresholded  ;
filter_B_thresholded  = fft2(B) .* filter_shift_thresholded  ;


invR_thresholded  = real(ifft2(filter_R_thresholded ));
invG_thresholded  = real(ifft2(filter_G_thresholded ));
invB_thresholded  = real(ifft2(filter_B_thresholded ));

Inew_thresholded (:,:,1)= invR_thresholded ;
Inew_thresholded (:,:,2)= invG_thresholded ;
Inew_thresholded (:,:,3)= invB_thresholded ;
```

```
%to create the image of the weave patterns

filter_shift_weave_thresholded  = fftshift(abs(masked_thresholded ));

filter_R_weave_thresholded = fft2(R) .* filter_shift_weave_thresholded ;
filter_G_weave_thresholded = fft2(G) .* filter_shift_weave_thresholded ;
filter_B_weave_thresholded = fft2(B) .* filter_shift_weave_thresholded ;


invR_weave_thresholded = real(ifft2(filter_R_weave_thresholded));
invG_weave_thresholded = real(ifft2(filter_G_weave_thresholded));
invB_weave_thresholded = real(ifft2(filter_B_weave_thresholded));

Inew_weave_thresholded(:,:,1)= invR_weave_thresholded;
Inew_weave_thresholded(:,:,2)= invG_weave_thresholded;
Inew_weave_thresholded(:,:,3)= invB_weave_thresholded;

montage({image_used, Inew_thresholded,Inew_weave_thresholded}, 'size', [1
NaN])
```

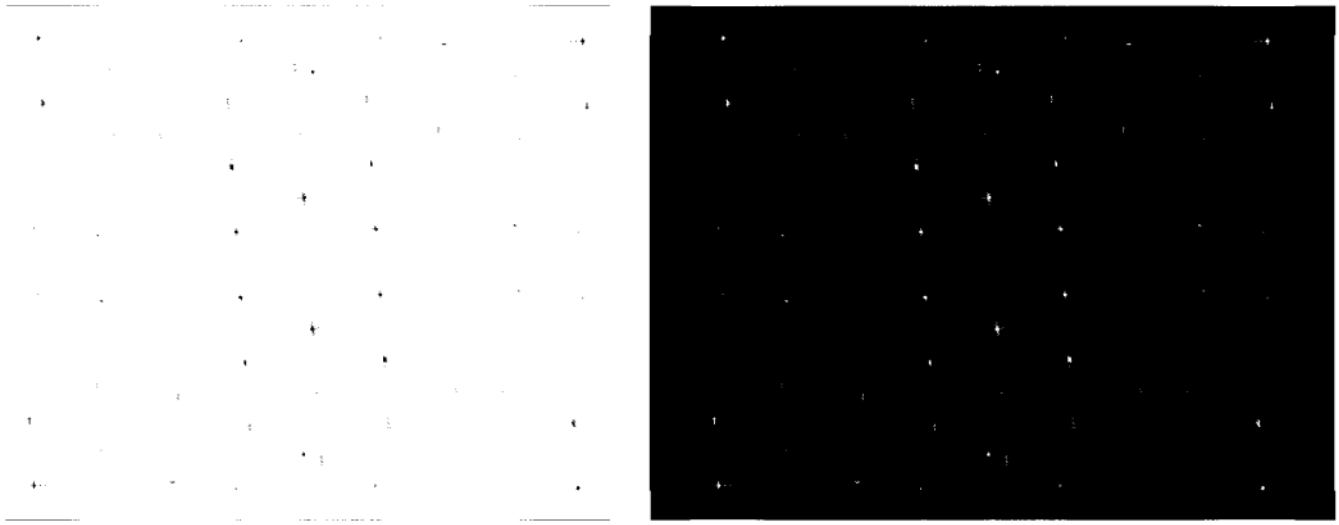
```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_thresholding_daria_results.png',
'Resolution', 90);
```

I personally think that the results are better than the previous one. Still, I decided to add some other ways. Here, I used the sensitivity property of imbinarized. I made the sensitivity 0 to produce a binary image with fewer foreground objects. Greater sensitiy value means greater foreground objects.

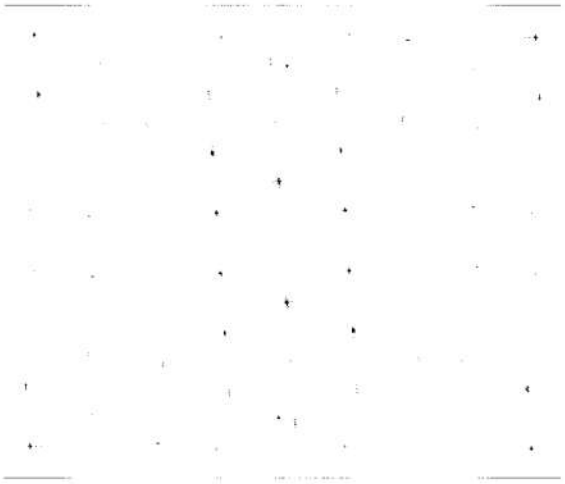
```
colormap_image_thresholded =
im2double(imread('FFT_logarithmic_hotmap.png'));
resized_colormap_image = imresize(colormap_image_thresholded , [960, 1280]);

masked_sensitive =
imbinarize(rgb2gray(resized_colormap_image),"adaptive","Sensitivity",0);
```

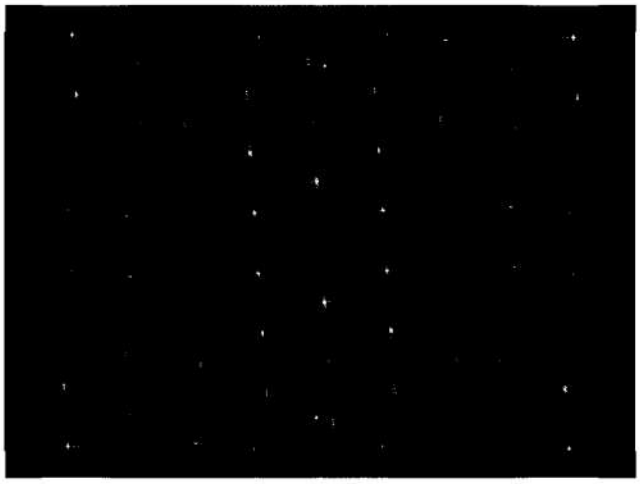
```
cx = size(masked_sensitive, 2)/2;   % x coordinate of the center of the
circle
cy = size(masked_sensitive, 1)/2;   % y coordinate of the center of the
circle
radius = 20;  % radius of the circle in pixels

% Create a binary mask of the same size as the image
```

```
[x, y] = meshgrid(1:size(masked_sensitive,2), 1:size(masked_sensitive,1));
mask = hypot(x-cx, y-cy) <= radius;

% Apply the mask to the image
masked_sensitive(mask) = 0;

masked_edited_sensitive = imcomplement(masked_sensitive);
imshowpair(masked_edited_sensitive, masked_sensitive, 'montage')
```



```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_sensitive_daria.png', 'Resolution',
90);
```





```
[R,G,B] = imsplit(image_used);

% to create the image of the painting with less patterns
filter_shift_sensitive = fftshift(abs(masked_edited_sensitive));

filter_R_sensitive = fft2(R) .* filter_shift_sensitive ;
filter_G_sensitive = fft2(G) .* filter_shift_sensitive ;
filter_B_sensitive = fft2(B) .* filter_shift_sensitive ;


invR_sensitive = real(ifft2(filter_R_sensitive));
invG_sensitive = real(ifft2(filter_G_sensitive));
invB_sensitive = real(ifft2(filter_B_sensitive));

Inew_sensitive(:,:,1)= invR_sensitive;
Inew_sensitive(:,:,2)= invG_sensitive;
Inew_sensitive(:,:,3)= invB_sensitive;

% to create the image of the patterns
```

```
filter_shift_weave_sensitive = fftshift(abs(masked_sensitive));

filter_R_weave_sensitive = fft2(R) .* filter_shift_weave_sensitive ;
filter_G_weave_sensitive = fft2(G) .* filter_shift_weave_sensitive ;
filter_B_weave_sensitive = fft2(B) .* filter_shift_weave_sensitive ;


invR_weave_sensitive = real(ifft2(filter_R_weave_sensitive));
invG_weave_sensitive = real(ifft2(filter_G_weave_sensitive));
invB_weave_sensitive = real(ifft2(filter_B_weave_sensitive));

Inew_weave_sensitive(:,:,1)= invR_weave_sensitive;
Inew_weave_sensitive(:,:,2)= invG_weave_sensitive;
Inew_weave_sensitive(:,:,3)= invB_weave_sensitive;

montage({image_used, Inew_sensitive,Inew_weave_sensitive}, 'size', [1 NaN])
```

```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_sensitive_daria_results.png',
'Resolution', 90);
```

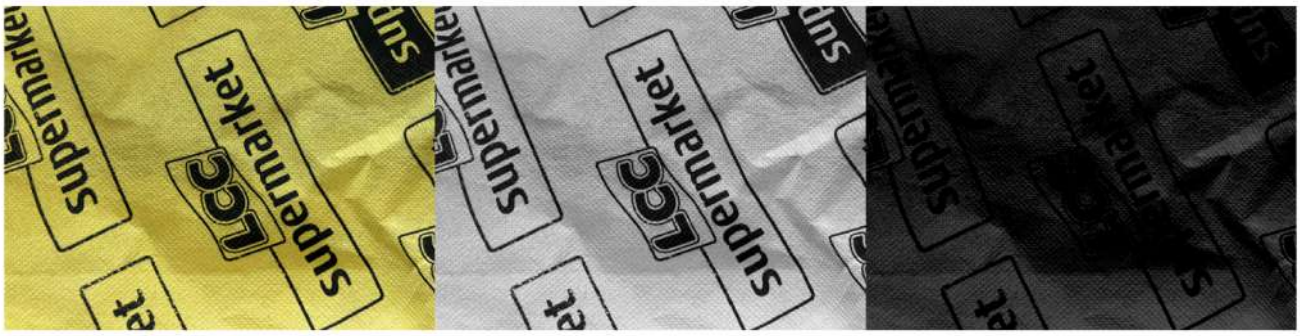
The montage below shows the results.

```
montage({image_used,Inew, Inew_thresholded,Inew_sensitive}, 'size', [1 NaN])
```

```

```

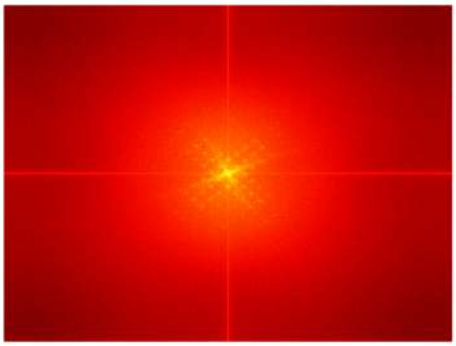
Extra Challenge 1

```
ecobag = im2double(imread('IMG_7624.jpeg'));
gray_ecobag = rgb2gray(ecobag);
mean_subtracted_ecobag = gray_ecobag - mean2(gray_ecobag);
montage({ecobag, gray_ecobag, mean_subtracted_ecobag}, 'size', [1 NaN])
```




```
FFT_meansub_ecobag = fft2(mean_subtracted_ecobag);
FFT_absmeansub_ecobag = abs(FFT_meansub_ecobag);
FFT_shifted_ecobag = fftshift(FFT_meansub_ecobag);
FFT_meansub_shifted_ecobag = abs(FFT_shifted_ecobag);
FFT_logarithmic_ecobag = rescale(log(FFT_meansub_shifted_ecobag + 1),
0,256);
```

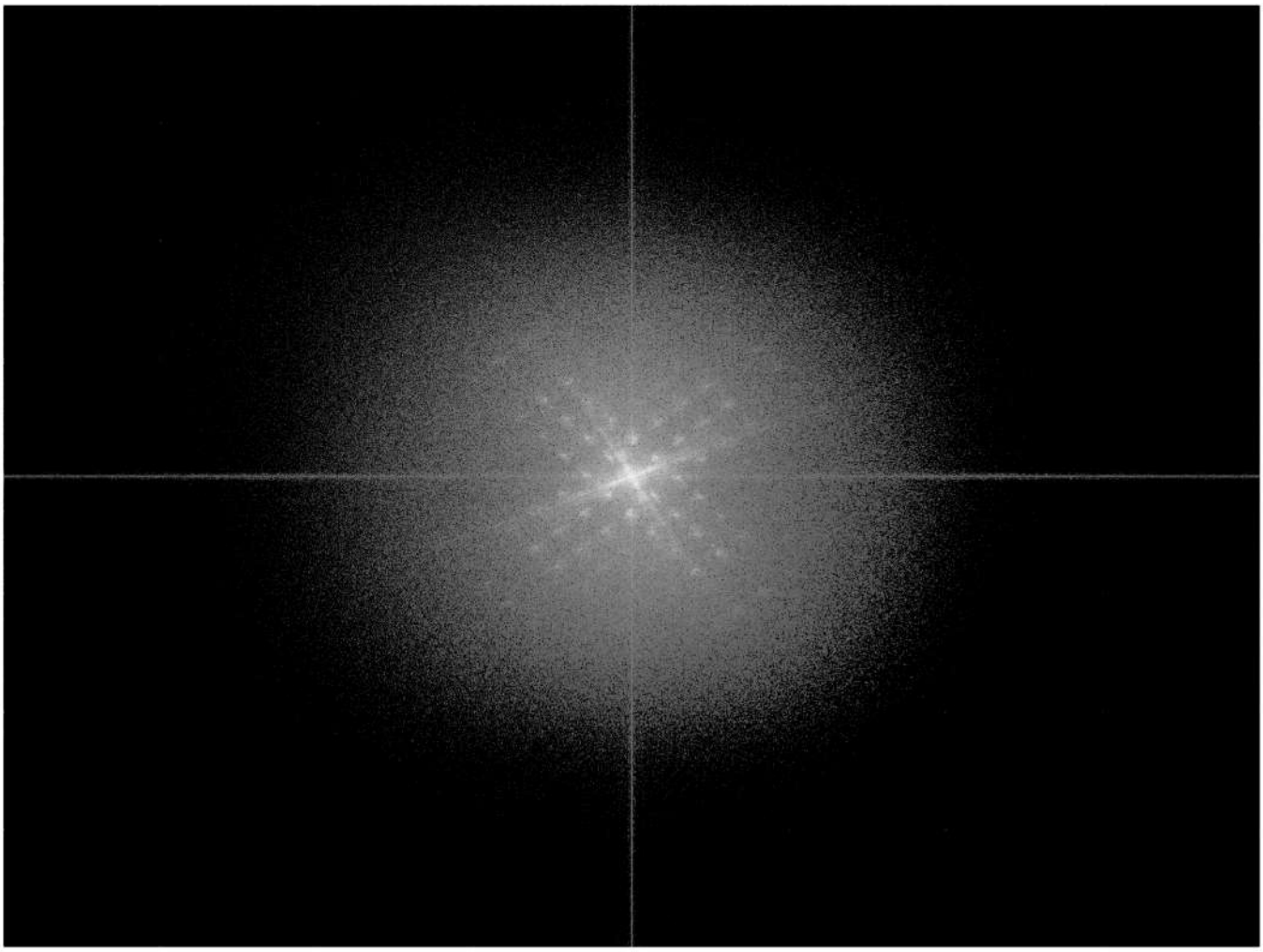
```
imagesc(FFT_logarithmic_ecobag);
colormap('hot')
axis off;
axis image;
```



```
% Set the figure size
%set(gcf, 'Units', 'pixels');
%set(gcf, 'Position', [0, 0, 4032, 3024]);

% Save the figure
```

```
%exportgraphics(gcf, '~/Desktop/FFT_logarithmic_ecobag.png', 'Resolution',
180);
```

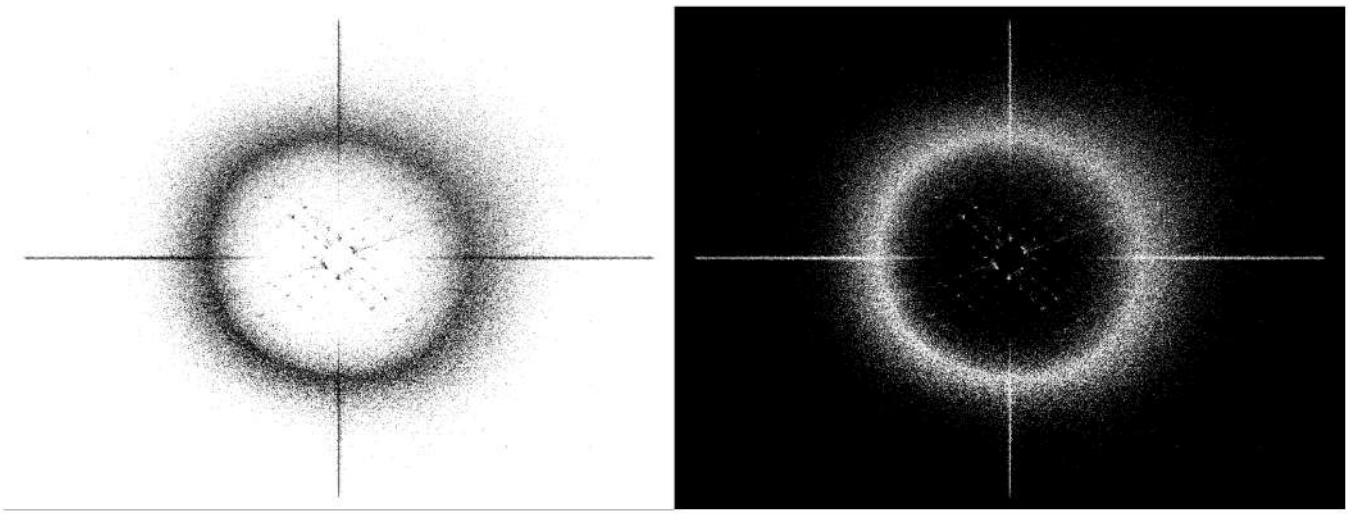
```
colormap_ecobag = im2double(imread('FFT_logarithmic_ecobag.png'));
resized_colormap_ecobag = imresize(colormap_ecobag, [3024, 4032]);
colormap_image_ecobag = rgb2gray(resized_colormap_ecobag);
colormap_image_ecobag(colormap_image_ecobag <= 0.30) = 0;
imshow(colormap_image_ecobag)
```



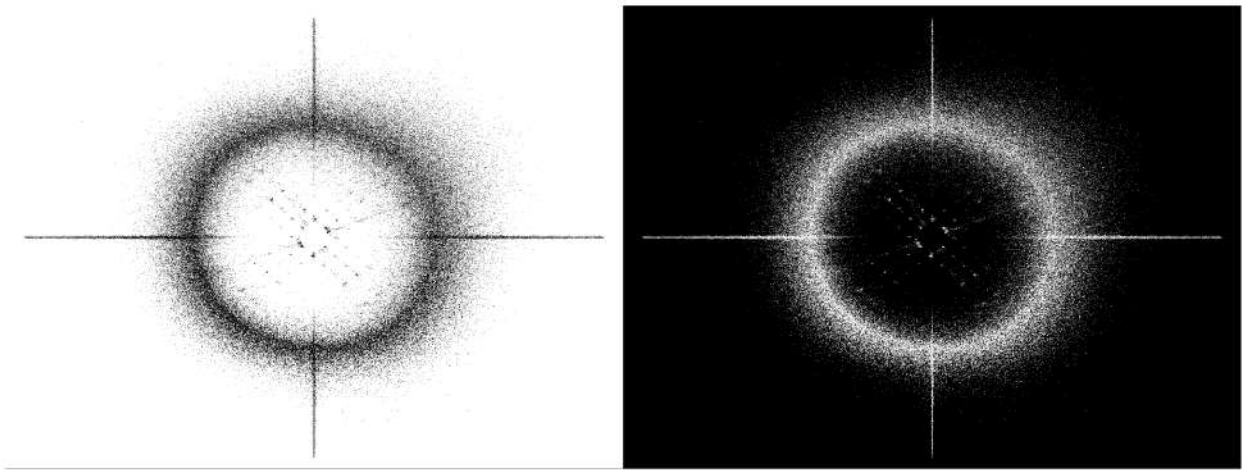
```
%imtool(colormap_image_ecobag)
```

```
masked_ecobag =
imbinarize(im2gray(colormap_image_ecobag),"adaptive","Sensitivity",0.3);
%0.3 sensitivity
cx = size(masked_ecobag, 2)/2;   % x coordinate of the center of the circle
cy = size(masked_ecobag, 1)/2;   % y coordinate of the center of the circle
radius = 90;  % radius of the circle in pixels
```

```
% Create a binary mask of the same size as the image
[x, y] = meshgrid(1:size(masked_ecobag,2), 1:size(masked_ecobag,1));
mask = hypot(x-cx, y-cy) <= radius;

% Apply the mask to the image
masked_ecobag(mask) = 0;

masked_edited_ecobag = imcomplement(masked_ecobag);
imshowpair(masked_edited_ecobag, masked_ecobag, 'montage')
```



```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_ecobag.png', 'Resolution', 90);
```



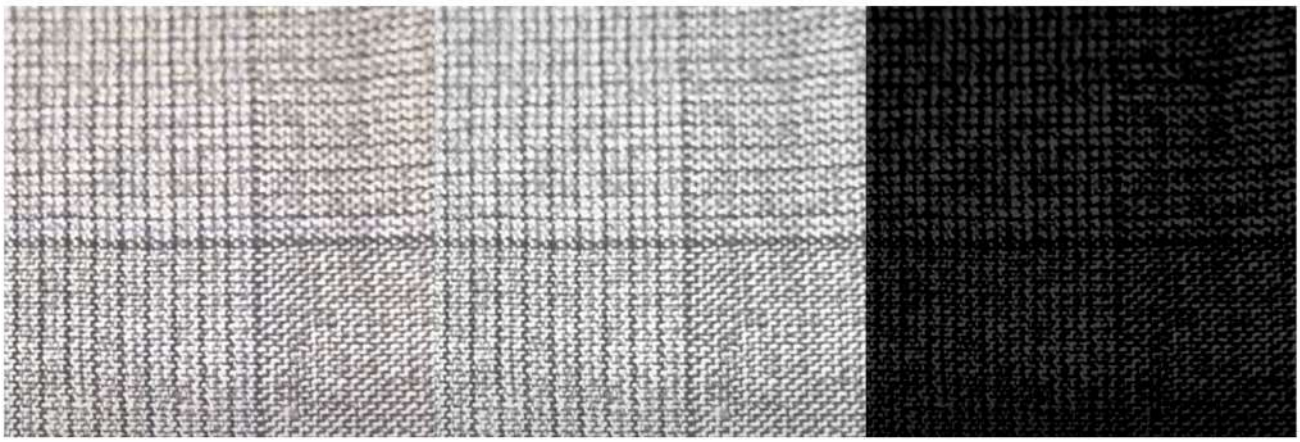
```
[R,G,B] = imsplit(ecobag);

% to create the image of the painting with less patterns
filter_shift_ecobag = fftshift(abs(masked_edited_ecobag));

filter_R_ecobag = fft2(R) .* filter_shift_ecobag ;
filter_G_ecobag = fft2(G) .* filter_shift_ecobag ;
filter_B_ecobag = fft2(B) .* filter_shift_ecobag ;


invR_ecobag = real(ifft2(filter_R_ecobag));
invG_ecobag = real(ifft2(filter_G_ecobag));
invB_ecobag = real(ifft2(filter_B_ecobag));

Inew_ecobag(:,:,1)= invR_ecobag;
Inew_ecobag(:,:,2)= invG_ecobag;
Inew_ecobag(:,:,3)= invB_ecobag;

% to create the image of the patterns
filter_shift_weave_ecobag = fftshift(abs(masked_ecobag));
```

```
filter_R_weave_ecobag = fft2(R) .* filter_shift_weave_ecobag ;
filter_G_weave_ecobag = fft2(G) .* filter_shift_weave_ecobag ;
filter_B_weave_ecobag = fft2(B) .* filter_shift_weave_ecobag ;


invR_weave_ecobag = real(ifft2(filter_R_weave_ecobag));
invG_weave_ecobag = real(ifft2(filter_G_weave_ecobag));
invB_weave_ecobag = real(ifft2(filter_B_weave_ecobag));

Inew_weave_ecobag(:,:,1)= invR_weave_ecobag;
Inew_weave_ecobag(:,:,2)= invG_weave_ecobag;
Inew_weave_ecobag(:,:,3)= invB_weave_ecobag;

montage({ecobag, Inew_ecobag,Inew_weave_ecobag}, 'size', [1 NaN])
```

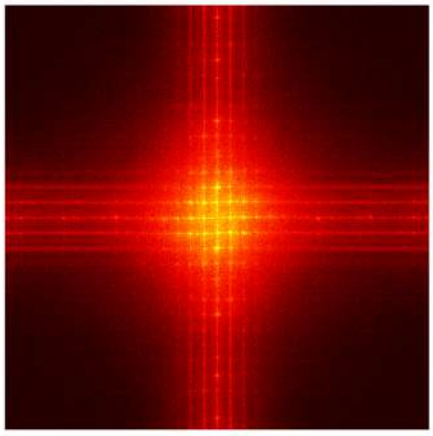

```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_ecobag_results.png', 'Resolution',
90);
```

Extra Challenge 2

```
yt_pic = im2double(imread('weaving_yt.png'));
gray_ytpic = rgb2gray(yt_pic);
mean_subtracted_ytpic = gray_ytpic - mean2(gray_ytpic);
montage({yt_pic, gray_ytpic, mean_subtracted_ytpic}, 'size', [1 NaN])
```

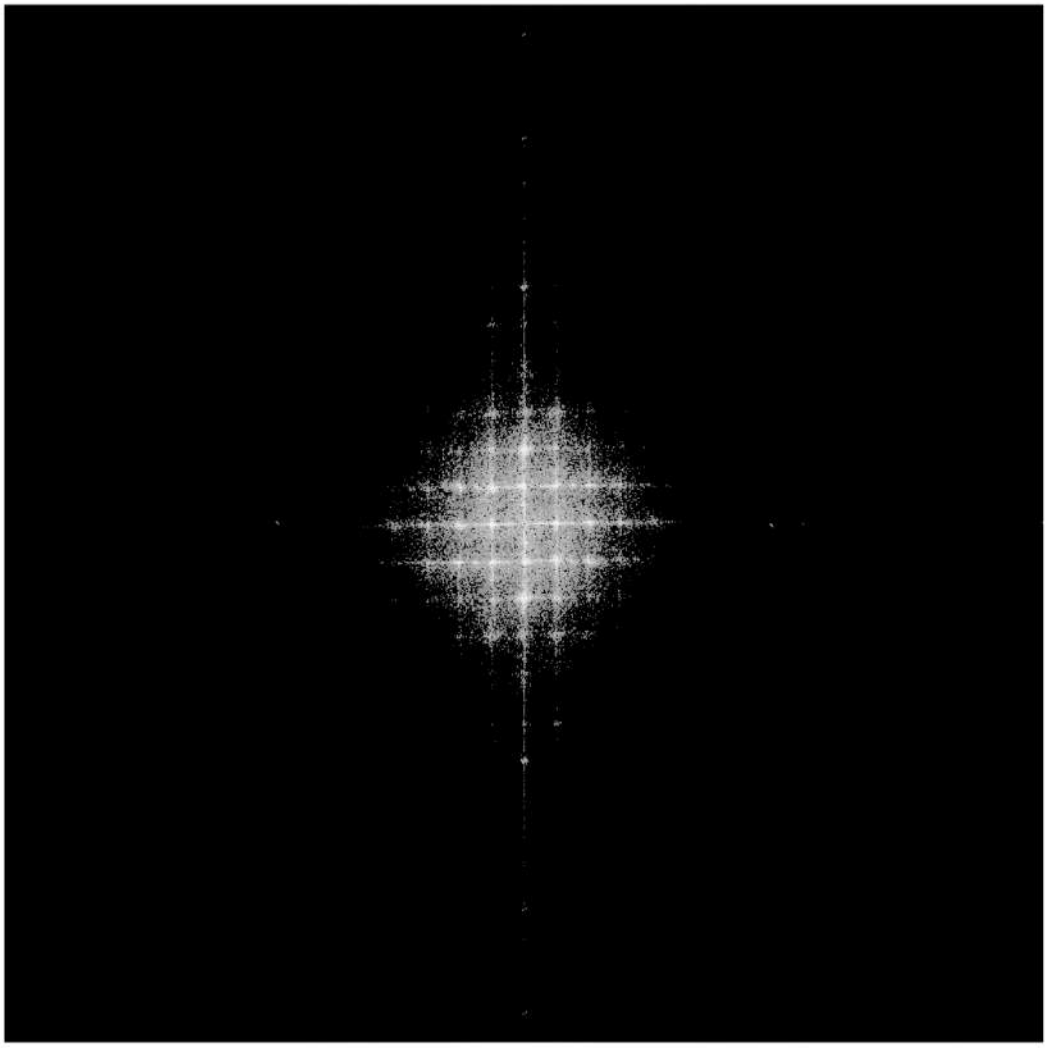


```
% imshow(yt_pic);
ytpic_fft = fft2(mean_subtracted_ytpic);
ytpic_fft_shifted= fftshift(abs(ytpic_fft));
ytpic_log = rescale(log(ytpic_fft_shifted +1), 0, 256);
imagesc(ytpic_log);
colormap('hot')
axis off;
axis image;
```

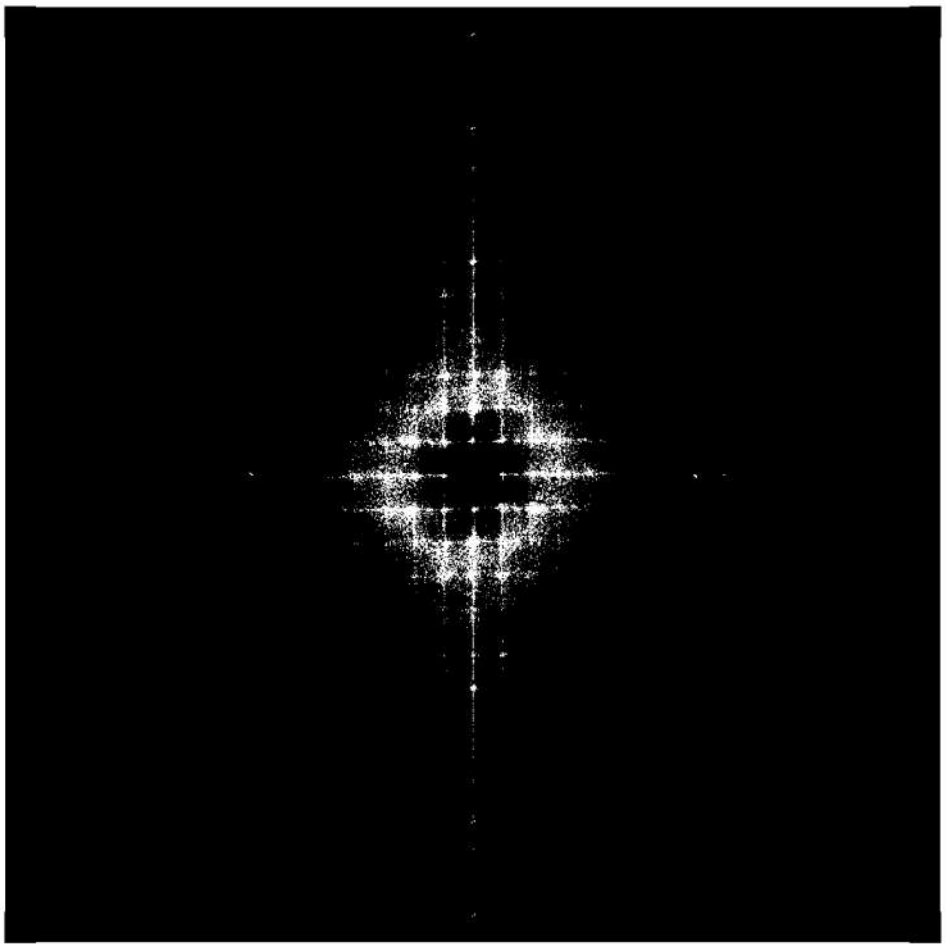


```
% Set the figure size
%set(gcf, 'Units', 'pixels');
%set(gcf, 'Position', [0, 0, 1040, 1040]);

% Save the figure
%exportgraphics(gcf, '~/Desktop/FFT_logarithmic_ytpic.png', 'Resolution',
144);
```

```
colormap_ytpic = im2double(imread('FFT_logarithmic_ytpic.png'));
resized_colormap_ytpic = imresize(colormap_ytpic, [1040, 1040]);
colormap_image_ytpic = rgb2gray(resized_colormap_ytpic);
colormap_image_ytpic(colormap_image_ytpic <= 0.5) = 0;
imshow(colormap_image_ytpic)
```

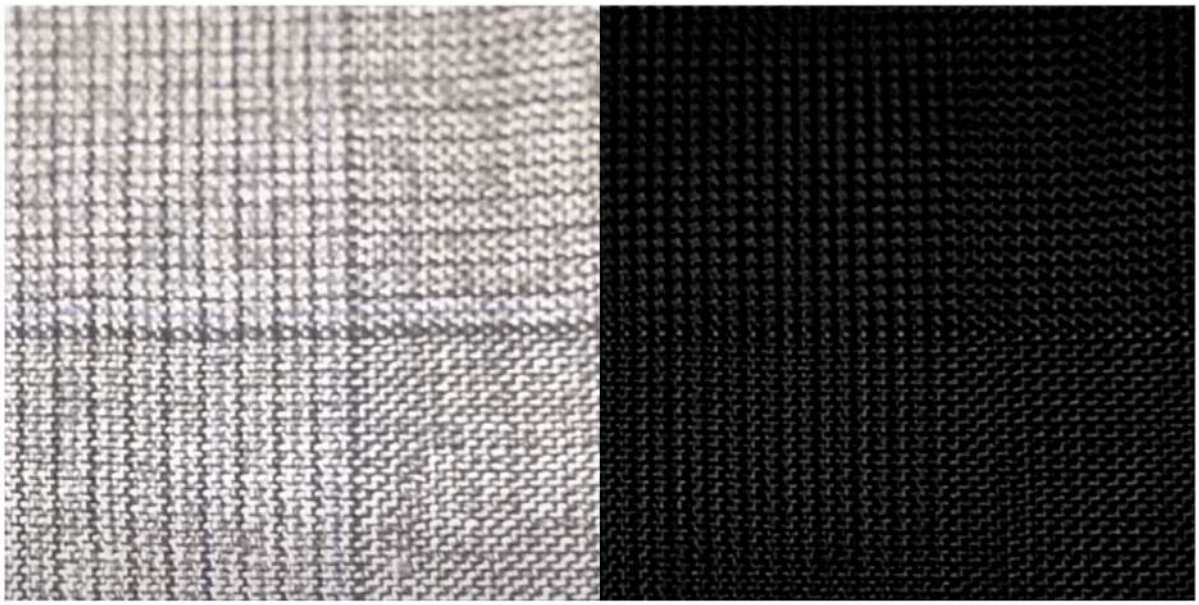


```
masked_ytpic =
imbinarize(im2gray(colormap_image_ytpic),"adaptive","Sensitivity",0);
imshow(masked_ytpic);
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_weave.png', 'Resolution', 90);
```



```
[R,G,B] = imsplit(yt_pic);

% to create the image of the painting with less patterns
filter_shift_ytpic = fftshift(abs(masked_ytpic));

filter_R_ytpic = fft2(R) .* filter_shift_ytpic ;
filter_G_ytpic = fft2(G) .* filter_shift_ytpic ;
filter_B_ytpic = fft2(B) .* filter_shift_ytpic ;


invR_ytpic = real(ifft2(filter_R_ytpic));
invG_ytpic = real(ifft2(filter_G_ytpic));
invB_ytpic = real(ifft2(filter_B_ytpic));

Inew_ytpic(:,:,1)= invR_ytpic;
Inew_ytpic(:,:,2)= invG_ytpic;
Inew_ytpic(:,:,3)= invB_ytpic;
```

```
montage({yt_pic, Inew_ytpic}, 'size', [1 NaN])
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 683 ,512 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/masked_weave_results.png', 'Resolution', 90);
```



2.2.3 Convolution Theorem Redux

```
img = zeros(200,200);
img(100, 99) = 1;
img(100, 101) = 1;

ft = abs(fftshift(fft2(img)));

log_ft = zeros(size(ft)); % initialize log_ft array with same size as ft
log_ft = rescale(log(ft + 1), 0, 256);

imshow(img)
```

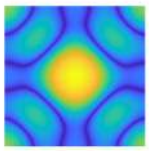
```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros.png', 'Resolution', 90);
```

```
imagesc(log_ft);
colormap('parula')
axis off;
axis image;

% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_fft.png', 'Resolution', 90);
```

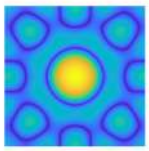
```

```

```
% create binary image with circles
[x, y] = meshgrid(1:200, 1:200);
circles = (x-100).^2 + (y-100).^2 <= 2^2;
img = circles;

% Compute Fourier transform
ft = fftshift(fft2(img));

% Compute modulus
log_ft = rescale(log(abs(ft) + 1), 0, 256);

% Display the result
imagesc(log_ft);
colormap('parula')
axis off;
```

```
axis image;
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/circles.png', 'Resolution', 90);
```

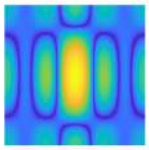


```
% create binary image with circles
[x, y] = meshgrid(1:200, 1:200);
circles = (x-100).^2 + (y-100).^2 <= 3^2;
img = circles;

% Compute Fourier transform
ft = fftshift(fft2(img));

% Compute modulus
log_ft = rescale(log(abs(ft) + 1), 0, 256);

% Display the result
imagesc(log_ft);
colormap('parula')
axis off;
axis image;
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/circles_bigger_rad.png', 'Resolution', 90);
```



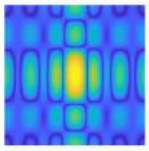
```
% Create the image with two dots
img = zeros(200, 200);
img(100, 99) = 1;
img(100, 101) = 1;
```

```
% Replace the dots with squares of width 3
width = 3;
squares = ones(width, width);
img(99:101, 98:100) = squares;
img(99:101, 102:104) = squares;

% Compute Fourier transform
ft = abs(fftshift(fft2(img)));

% Compute logarithmic scaling
log_ft = rescale(log(ft + 1), 0, 256);

% Display results
imagesc(log_ft);
colormap('parula')
axis off;
axis image;
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/two_dots_circle.png', 'Resolution', 90);
```

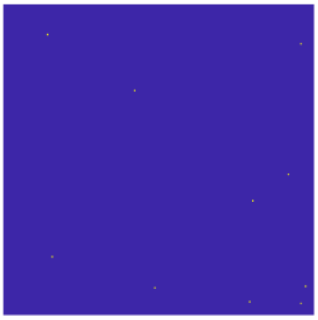


```
% Create the image with two dots
img = zeros(200, 200);
img(100, 99) = 1;
img(100, 101) = 1;

% Replace the dots with squares of width 5
width = 5;
squares = ones(width, width);
img(98:102, 96:100) = squares;
img(98:102, 102:106) = squares;

% Compute Fourier transform
ft = abs(fftshift(fft2(img)));

% Compute logarithmic scaling
log_ft = rescale(log(ft + 1), 0, 256);

% Display results
```

```
imagesc(log_ft);
colormap('parula')
axis off;
axis image;
hold off
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/two_dots_square.png', 'Resolution', 90);
```



```
% Create an array of zeros
A = zeros(200, 200);

% Generate 10 random locations
locs = randi([1 200], 10, 2);

% Place ones at the random locations
for i = 1:size(locs, 1)
    A(locs(i, 1), locs(i, 2)) = 1;
end

% Display the resulting array
imagesc(A);
axis off;
axis image;
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 200 ,200 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_conv.png', 'Resolution', 90);
```



```
pattern_star = im2double(imread('star.png'));
pattern_gray_star = rgb2gray(pattern_star);
pattern_gray_resize_star = imresize(pattern_gray_star, [9, 9]);
convolved_star = ifft2(fft2(A) .* fft2(pattern_gray_resize_star, 200, 200),
'symmetric');

% Display the result
imshow(convolved_star, []);
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 200 ,200 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/convolved_star.png', 'Resolution', 90);
```

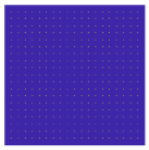
```
%trying another way
conv2(A, pattern_gray_star);
imshowpair(convolved_star, conv2(A, pattern_gray_resize_star), 'montage')
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 200 ,200 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_conv_both.png', 'Resolution', 90);
```

```
pattern_ribbon= im2double(imread('ribbon.png'));
pattern_gray_ribbon = rgb2gray(pattern_ribbon);
pattern_gray_resize_ribbon = imresize(pattern_gray_ribbon, [9, 9]);
convolved_ribbon = ifft2(fft2(A) .* fft2(pattern_gray_resize_ribbon, 200,
200), 'symmetric');

% Display the result
imshow(convolved_ribbon, []);
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 200 ,200 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_conv_heart.png', 'Resolution', 90);
```

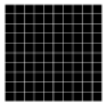
```
%trying another way
conv2(A, pattern_gray_ribbon);
imshow(conv2(A, pattern_gray_ribbon))
```

```
imshowpair(convolved_ribbon, conv2(A, pattern_gray_resize_ribbon),
'montage')
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 200 ,200 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_conv_heart_both.png',
'Resolution', 90);
```

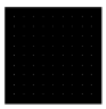
```
A = zeros(200,200);
for i = 10:10:190
    for j = 10:10:190
        A(i,j) = 1;
    end
end

imagesc(A);
axis off;
axis image;
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_equal.png', 'Resolution', 90);
```



```
fft_A = fft2(A);
fft_shifted_A = fftshift(abs(fft_A));
log_fft_A = rescale(log(fft_shifted_A + 1), 0, 256);
imshow(log_fft_A)
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_equal_fft.png', 'Resolution', 90);
```

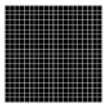


```
B = zeros(200,200);
for i = 20:20:190
    for j = 20:20:190
        B(i,j) = 1;
    end
end

imshow(B);
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_equal_bigger.png', 'Resolution',
90);
```



```
fft_B = fft2(B);
```

```
fft_shifted_B = fftshift(abs(fft_B));
log_fft_B = rescale(log(fft_shifted_B + 1), 0, 256);
imshow(log_fft_B)
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 100 ,100 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/zeros_200_equal_bigger_fft.png',
'Resolution', 90);
```



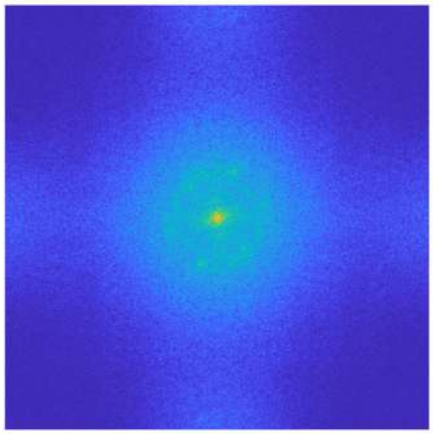
2.2.4 Fingerprints: Ridge Enhancement

```
finger = im2double(imread('fingerprint.png'));
gray_finger = rgb2gray(finger);
mean_subtracted_finger= gray_finger - mean2(gray_finger);
montage({finger, gray_finger, mean_subtracted_finger}, 'size', [1 NaN])
```

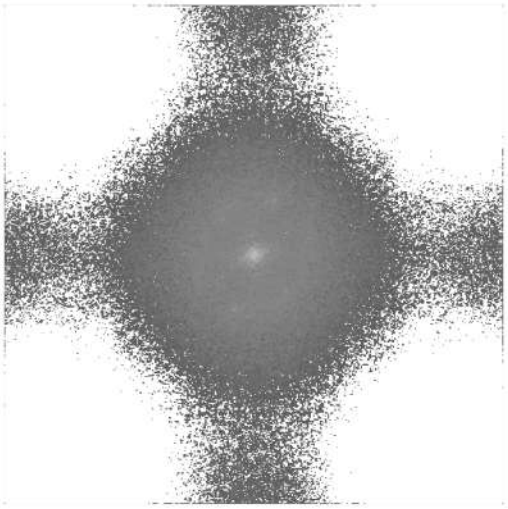
```
%taking the FFT of the mean-subtracted FFT
FFT_meansub_finger = fft2(mean_subtracted_finger);
FFT_absmeansub_finger = abs(FFT_meansub_finger);
FFT_shifted_finger = fftshift(FFT_meansub_finger);
FFT_meansub_shifted_finger = abs(FFT_shifted_finger);
```

```
FFT_logarithmic_finger = rescale(log(FFT_meansub_shifted_finger + 1),
0,256);
```

```
imagesc(FFT_logarithmic_finger);
colormap('parula')
axis off;
axis image;
```

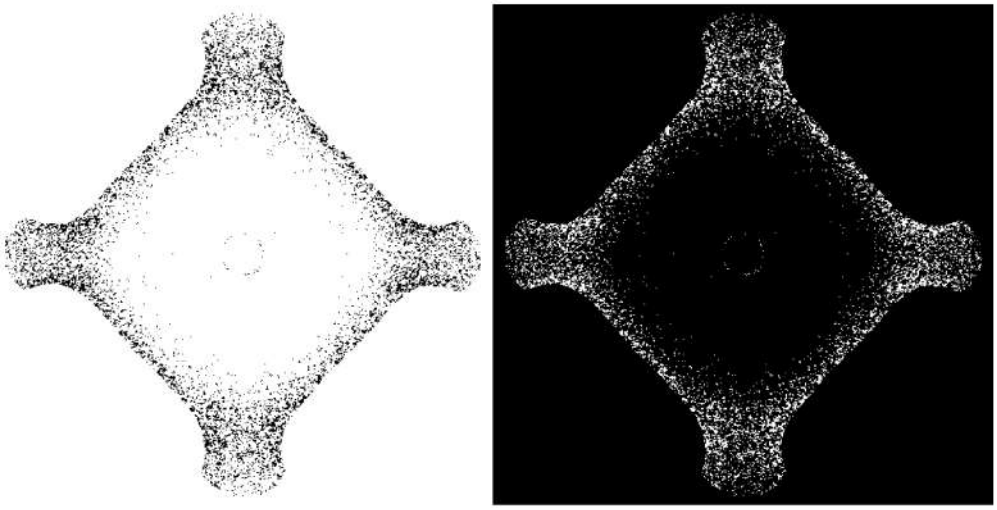


```
 %Set the figure size
%set(gcf, 'Units', 'pixels');
%set(gcf, 'Position', [0, 0, 500, 500]);

% Save the figure
%exportgraphics(gcf, '~/Desktop/FFT_logarithmic_finger.png', 'Resolution',
180);
colormap_finger = im2double(imread('FFT_logarithmic_finger.png'));
resized_colormap_finger = imresize(colormap_finger, [500, 500]);
colormap_image_finger = rgb2gray(resized_colormap_finger);

colormap_image_finger(colormap_image_finger <= 0.35) = 1;
imshow(colormap_image_finger)
```

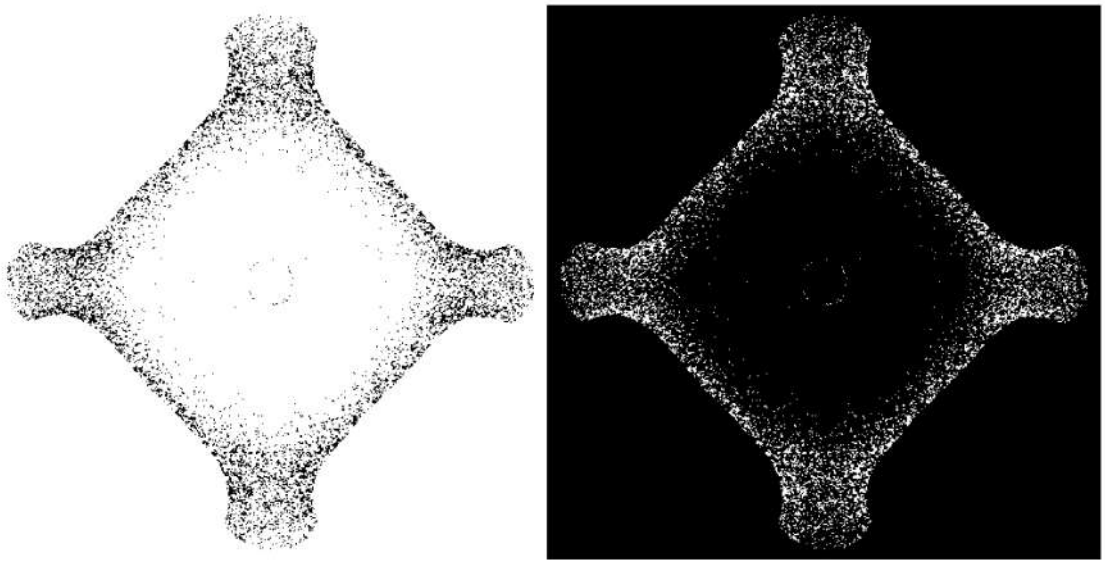


```
%imtool(colormap_image_finger)
```

```
% Set the center and radius of the circle
cx = size(colormap_image_finger, 2)/2;   % x coordinate of the center of
the circle
cy = size(colormap_image_finger, 1)/2;   % y coordinate of the center of
the circle
radius = 20;  % radius of the circle in pixels

% Create a binary mask of the same size as the image
[x, y] = meshgrid(1:size(colormap_image_finger,2),
1:size(colormap_image_finger,1));
mask = hypot(x-cx, y-cy) <= radius;

% Apply the mask to the image
colormap_image_finger(mask) = 0;


masked_thresholded =
imbinarize(colormap_image_finger ,"adaptive","Sensitivity",0.1);
masked_edited_thresholded  = imcomplement(masked_thresholded );
imshowpair(masked_edited_thresholded , masked_thresholded , 'montage')
```



```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1054 ,471 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_finger.png', 'Resolution', 90);
```



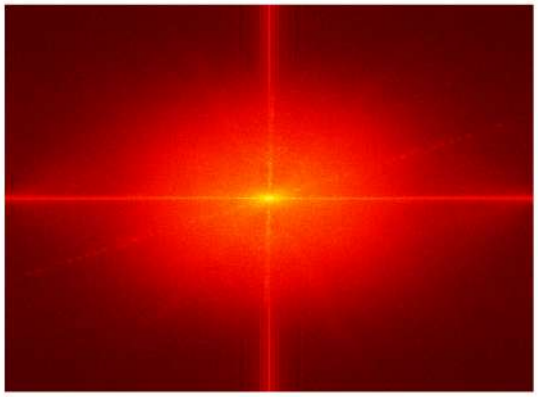
```
% no lines
filter_shift_finger = fftshift(abs(masked_thresholded));
```

```
filter_finger = fft2(gray_finger) .* filter_shift_finger;

inv_finger = real(ifft2(filter_finger));

Inew_finger = mat2gray(inv_finger);

%lines

filter_shift_weave_finger = fftshift(abs(masked_edited_thresholded));

filter_weave_finger = fft2(gray_finger) .* filter_shift_weave_finger;

inv_weave_finger = real(ifft2(filter_weave_finger));

Inew_weave_finger = mat2gray(inv_weave_finger);


montage({gray_finger,Inew_weave_finger, Inew_finger}, 'size', [1 NaN])
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1500 ,480 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_finger.png', 'Resolution', 90);
```

2.2.5 Lunar Landing Scanned Pictures: Line Removal

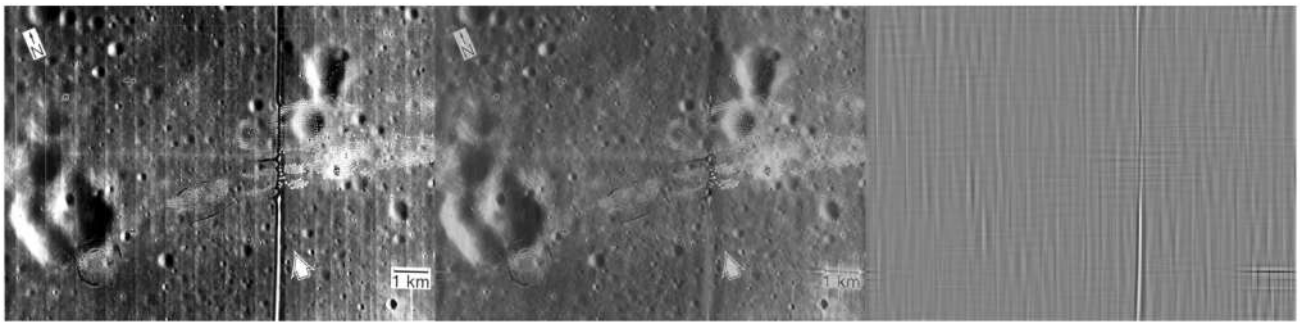
```
moon = im2double(imread('buwan.tif'));
mean_subtracted_image_moon = moon - mean2(moon);

FFT_moon = fft2(mean_subtracted_image_moon);
FFT_absmoon = abs(FFT_moon);
FFT_shifted_moon = fftshift(FFT_absmoon);
```

```
FFT_meansub_shifted_moon = abs(FFT_shifted_moon);
FFT_logarithmic_moon = rescale(log(FFT_meansub_shifted_moon + 1), 0, 256);

imagesc(FFT_logarithmic_moon);
colormap('hot')
axis off;
axis image;
```



```
% Set the figure size
%set(gcf, 'Units', 'pixels');
%set(gcf, 'Position', [0, 0, 1892, 1383]);

% Save the figure
%exportgraphics(gcf, '~/Desktop/FFT_logarithmic_hotmap_moon.png',
'Resolution', 300);
colormap_moon = im2double(imread('FFT_logarithmic_hotmap_moon_lines.png'));
resized_colormap_moon = imresize(colormap_moon, [1383, 1892]);
colormap_image_gray_moon_lines = rgb2gray(resized_colormap_moon);
colormap_image_gray_moon_reversed_lines =
imcomplement(colormap_image_gray_moon_lines);
imshowpair(colormap_image_gray_moon_reversed_lines,colormap_image_gray_moon_
lines, 'montage')
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1054 ,471 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_moon.png', 'Resolution', 90);
```

```
% no lines
filter_shift_moon_lines = fftshift(abs(colormap_image_gray_moon_lines));

filter_moon_lines = fft2(moon) .* filter_shift_moon_lines;

inv_moon_lines = real(ifft2(filter_moon_lines));

Inew_moon_lines = mat2gray(inv_moon_lines);

%lines

filter_shift_weave_moon_lines =
fftshift(abs(colormap_image_gray_moon_reversed_lines));

filter_weave_moon_lines = fft2(moon) .* filter_shift_weave_moon_lines;

inv_weave_moon_lines = real(ifft2(filter_weave_moon_lines));

Inew_weave_moon_lines = mat2gray(inv_weave_moon_lines);


montage({moon, Inew_moon_lines, Inew_weave_moon_lines}, 'size', [1 NaN])
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1500 ,470 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_moon_first.png', 'Resolution', 90);
```

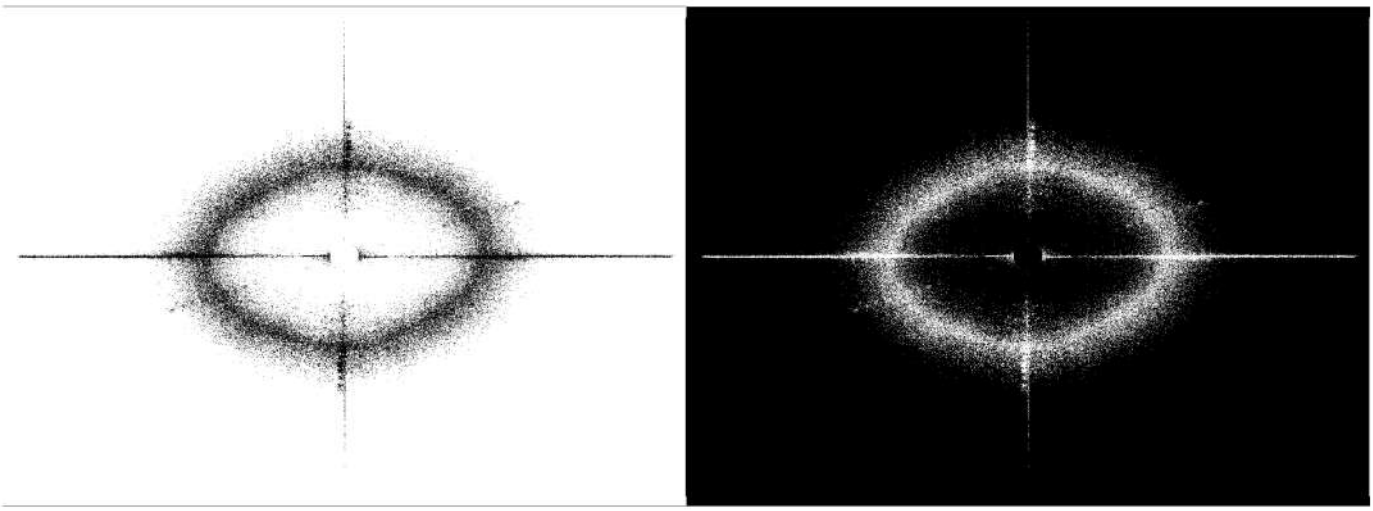


Another Method but this time the mask is not manually created but is based on thresholding

```
%imtool(colormap_image_gray_moon)

%thresholding the colormap_image_gray such that when its value is <= 0.75
%it becomes 0
%colormap_image_gray_thresholded(colormap_image_gray_thresholded <= 0.40) =
0;
colormap_moon_moon = im2double(imread('FFT_logarithmic_hotmap_moon.png'));
resized_colormap_moon_moon = imresize(colormap_moon_moon, [1383, 1892]);
colormap_image_gray_moon = rgb2gray(resized_colormap_moon_moon);
colormap_image_gray_moon(colormap_image_gray_moon <= 0.3) = 0;
masked_moon = imbinarize(colormap_image_gray_moon,
"adaptive","Sensitivity",0.30);
%masked_moon = imbinarize(,'adaptive', 'Sensitivity', 0.5,
'ForegroundPolarity', 'bright', 'Method', 'gaussian');
colormap_image_gray_moon_reversed = imcomplement(colormap_image_gray_moon);
%imshowpair(colormap_image_gray_moon,colormap_image_gray_moon_reversed,
'montage')
```

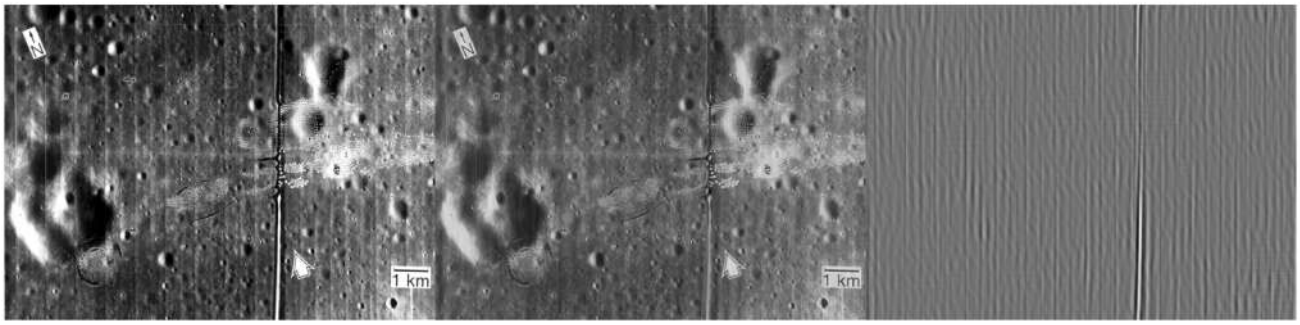
```
cx = size(masked_moon, 2)/2;   % x coordinate of the center of the circle
cy = size(masked_moon, 1)/2;   % y coordinate of the center of the circle
radius = 40;  % radius of the circle in pixels

% Create a binary mask of the same size as the image
[x, y] = meshgrid(1:size(masked_moon,2), 1:size(masked_moon,1));
mask = hypot(x-cx, y-cy) <= radius;
masked_moon(mask) = 0;


masked_moon_edited = imcomplement(masked_moon);
imshowpair(masked_moon_edited, masked_moon, 'montage')
```

```
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1054 ,471 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_moon_2.png', 'Resolution', 90);
```



```
% no lines
filter_shift_moon = fftshift(abs(masked_moon_edited));

filter_moon = fft2(moon) .* filter_shift_moon;

inv_moon = real(ifft2(filter_moon));

Inew_moon = mat2gray(inv_moon);

%lines

filter_shift_weave_moon = fftshift(abs(masked_moon));

filter_weave_moon = fft2(moon) .* filter_shift_weave_moon;

inv_weave_moon = real(ifft2(filter_weave_moon));

Inew_weave_moon = mat2gray(inv_weave_moon);

montage({moon, Inew_moon, Inew_weave_moon}, 'size', [1 NaN])
% Set the figure size
set(gcf, 'Units', 'pixels');
```

```
set(gcf, 'Position', [0, 0, 1500 ,470 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_moon_2_results.png', 'Resolution', 90);
```




```
montage({moon,Inew_moon_lines, Inew_moon}, 'size', [1 NaN])
% Set the figure size
set(gcf, 'Units', 'pixels');
set(gcf, 'Position', [0, 0, 1500 ,470 ]);

% Save the figure
exportgraphics(gcf, '~/Desktop/mask_moon_1_2_results.png', 'Resolution',
90);
```