Load the image

```
% Load the image
img = imread('image.jpg');
imshow(img);
title('Original Image');
```

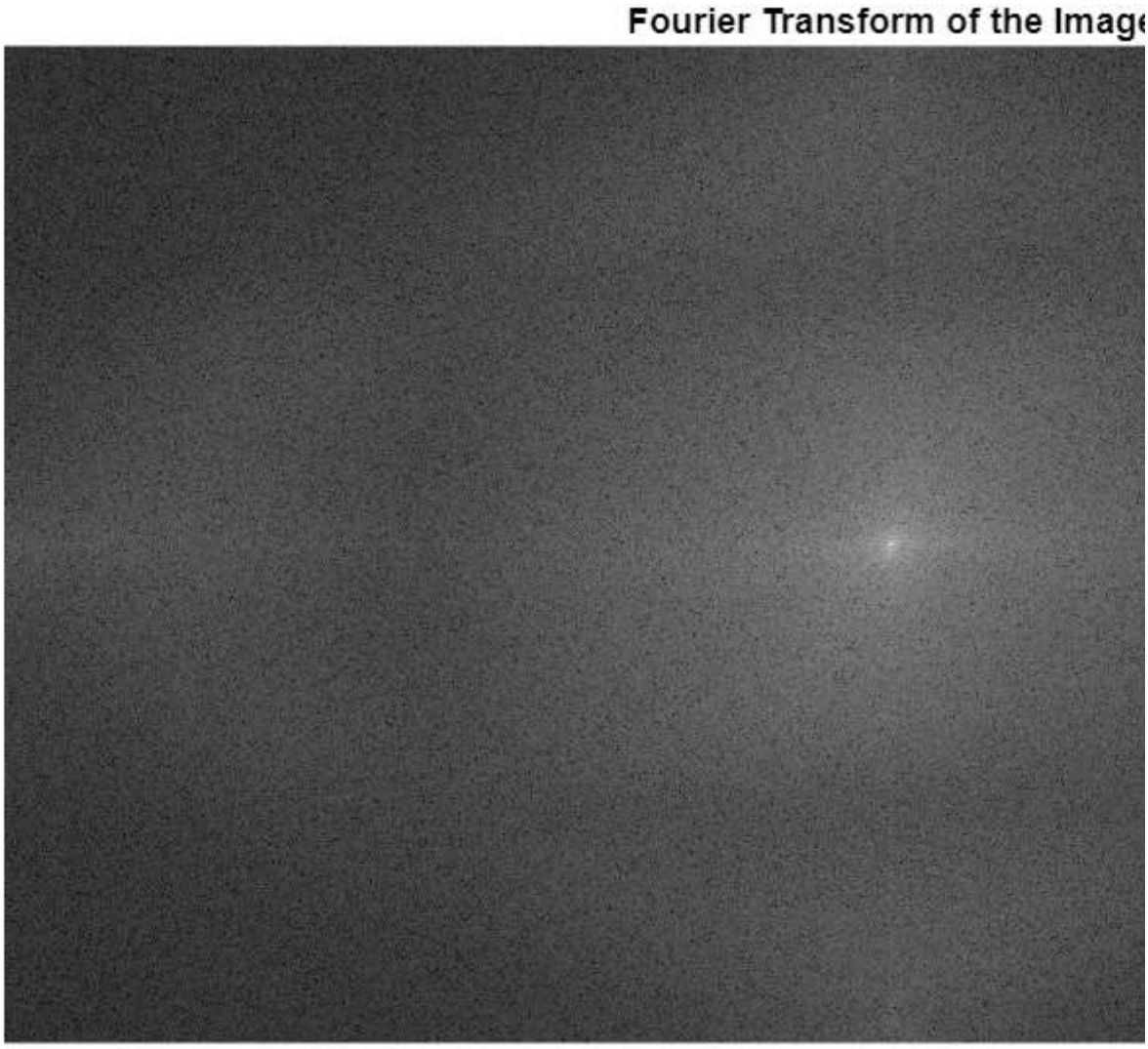
Fourier Transform

```
gray_img = rgb2gray(img); % Convert to grayscale if it's a color image
gray_img = im2double(gray_img); % Convert to double for processing

% Fourier Transform of the imageP
fft_img = fft2(gray_img);          % 2D Fourier Transform
fft_shifted = fftshift(fft_img);   % Shift zero frequency to center

% Display the Fourier Transform
figure;
```

```
imshow(log(1 + abs(fft_shifted)), []); % Log scale for better visualization
title('Fourier Transform of the Image');
```



Apply Butterworth Filter

```
% Butterworth Filter Design
[rows, cols] = size(gray_img);
D0 = 30; % Cutoff frequency
n = 2;  % Order of the filter

% Create a meshgrid for distances
[u, v] = meshgrid((-floor(cols/2):floor(cols/2)-1), (-floor(rows/2):floor(rows/2)-1));

% Calculate the distance D(u,v) from the center
D = sqrt(u.^2 + v.^2);

% Butterworth Low-Pass Filter
H_butterworth = 1 ./ (1 + (D./D0).^(2*n));

% Apply Butterworth filter in frequency domain
```

```
filtered_fft_butterworth = fft_shifted .* H_butterworth;

% Shift back and perform inverse Fourier Transform
filtered_img_butterworth = ifft2(ifftshift(filtered_fft_butterworth));

% Display the Butterworth filtered image
figure;
imshow(real(filtered_img_butterworth), []);
title('Butterworth Low-Pass Filtered Image');
```

Apply Gaussian Filter

```
% Gaussian Filter Design
sigma = 30;  % Standard deviation of the Gaussian (controls blurring)
H_gaussian = exp(-(D.^2) / (2*sigma^2));

% Apply Gaussian filter in frequency domain
filtered_fft_gaussian = fft_shifted .* H_gaussian;

% Shift back and perform inverse Fourier Transform
```

```
filtered_img_gaussian = ifft2(ifftshift(filtered_fft_gaussian));

% Display the Gaussian filtered image
figure;
imshow(real(filtered_img_gaussian), []);
title('Gaussian Low-Pass Filtered Image');
```