

Transformada de Fourier

Transformada de Fourier aplicada as imagens disponibilizadas

```
import numpy as np
import matplotlib.pyplot as plt

# Carregue a imagem (você precisa de uma imagem para isso)
imagem = plt.imread('periodic_noise.png')

# Converta a imagem para tons de cinza se necessário
if imagem.ndim == 3:
    imagem = np.mean(imagem, axis=2)

# Calcule a Transformada de Fourier 2D da imagem
transformada_fourier = np.fft.fft2(imagem)

# Desloque a frequência zero para o centro da imagem
transformada_fourier_deslocada = np.fft.fftshift(transformada_fourier)

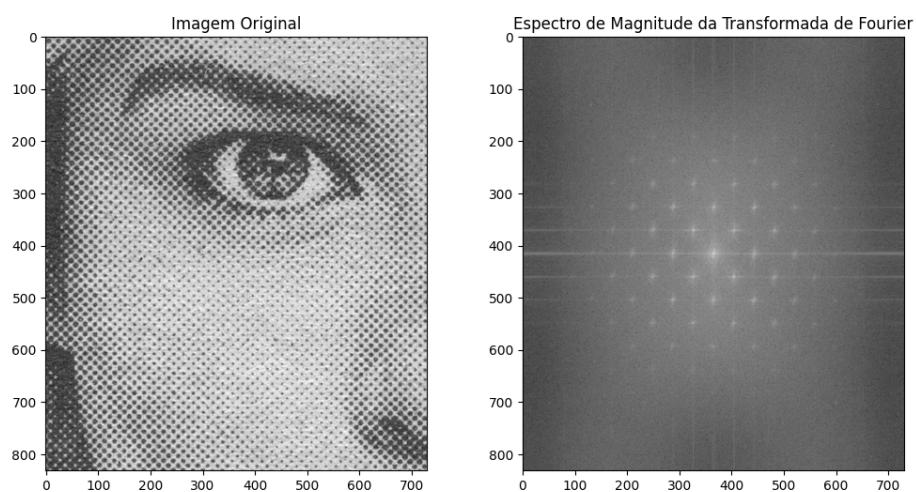
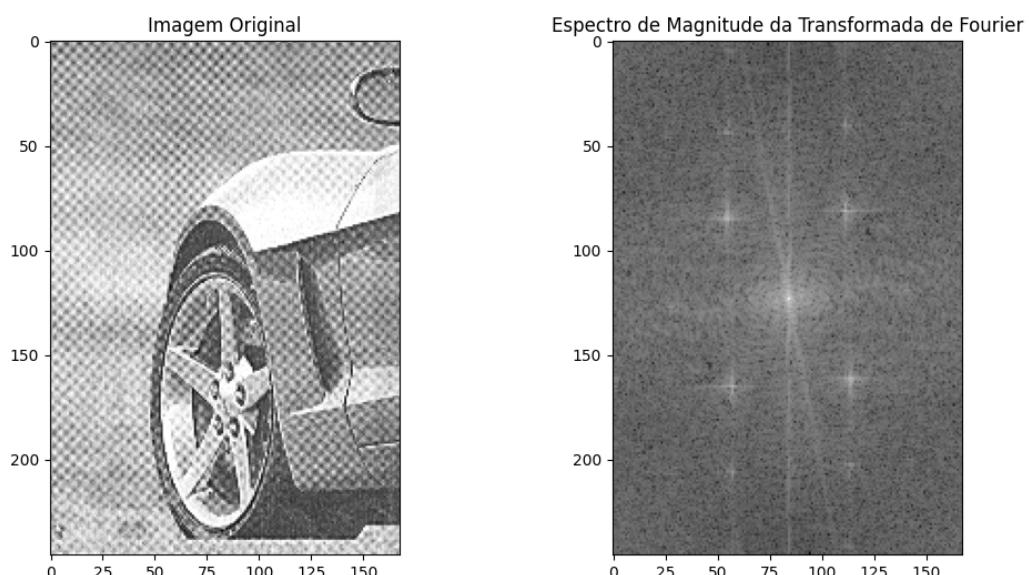
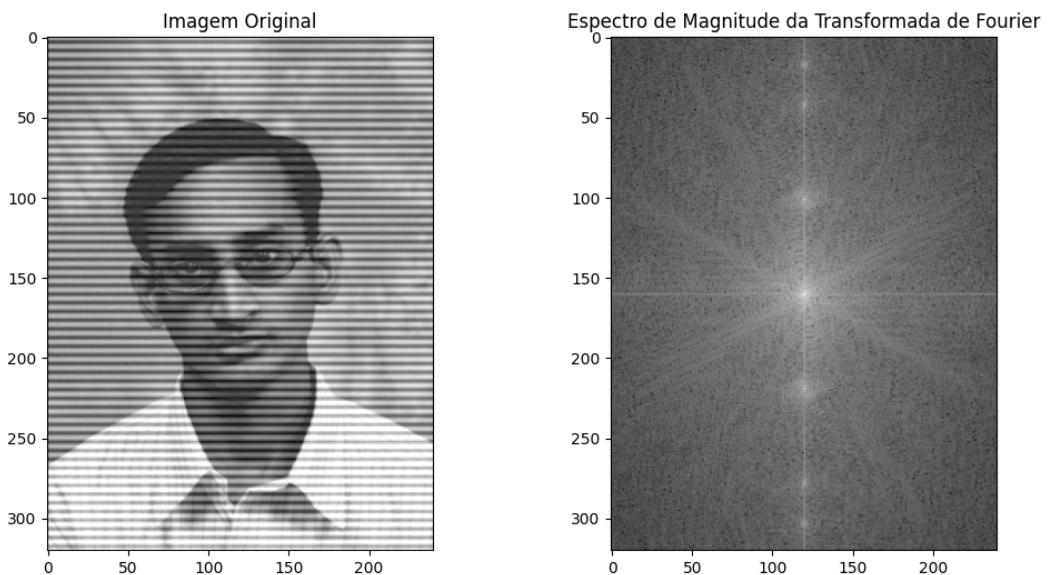
# Calcule o espectro de magnitude (magnitude da Transformada de Fourier)
espectro_magnitude = np.abs(transformada_fourier_deslocada)

# Crie uma figura com duas subtramas (uma para a imagem original e
# outra para o espectro)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

# Mostre a imagem original na primeira subtrama
ax1.imshow(imagem, cmap='gray')
ax1.set_title('Imagen Original')

# Mostre o espectro de magnitude na segunda subtrama
ax2.imshow(np.log(espectro_magnitude), cmap='gray')
ax2.set_title('Espectro de Magnitude da Transformada de Fourier')

plt.show()
```



Transformada inversa de Fourier aplicada as imagens disponibilizadas

```
import numpy as np
import matplotlib.pyplot as plt

# Carregue a imagem (você precisa de uma imagem para isso)
imagem = plt.imread('periodic_noise.png')

# Converta a imagem para tons de cinza se necessário
if imagem.ndim == 3:
    imagem = np.mean(imagem, axis=2)

# Calcule a Transformada de Fourier 2D da imagem
transformada_fourier = np.fft.fft2(imagem)

# Desloque a frequência zero para o centro da imagem
transformada_fourier_deslocada = np.fft.fftshift(transformada_fourier)

# Calcule o espectro de magnitude (magnitude da Transformada de
Fourier)
espectro_magnitude = np.abs(transformada_fourier_deslocada)

# Calcule a transformada inversa de Fourier
imagem_reconstruida =
np.fft.ifft2(np.fft.ifftshift(transformada_fourier_deslocada))

# Normalize a imagem reconstruída para exibi-la corretamente
imagem_reconstruida = np.abs(imagem_reconstruida)

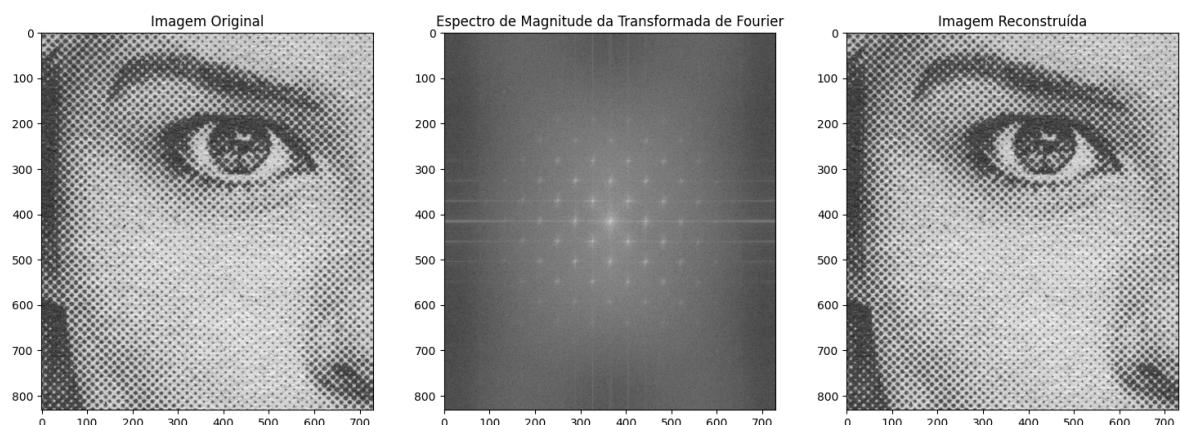
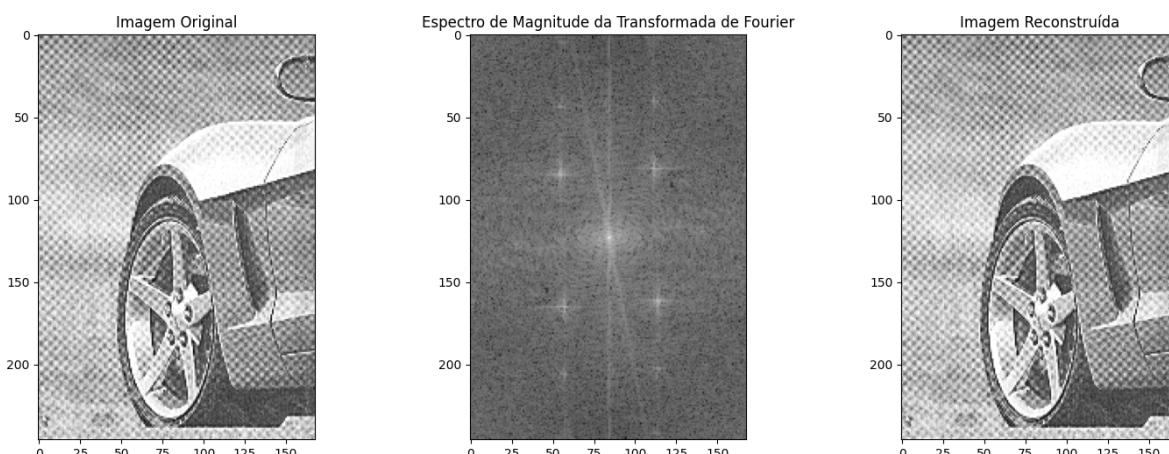
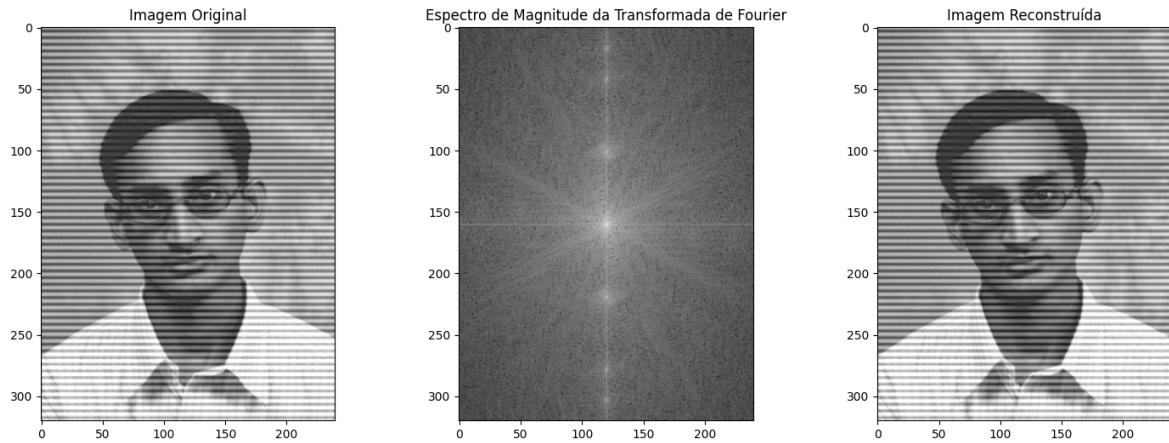
# Crie uma figura com três subtramas (imagem original, espectro e
imagem reconstruída)
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 6))

# Mostre a imagem original na primeira subtrama
ax1.imshow(imagem, cmap='gray')
ax1.set_title('Imagen Original')

# Mostre o espectro de magnitude na segunda subtrama
ax2.imshow(np.log(espectro_magnitude), cmap='gray')
ax2.set_title('Espectro de Magnitude da Transformada de Fourier')

# Mostre a imagem reconstruída na terceira subtrama
ax3.imshow(imagem_reconstruida, cmap='gray')
```

```
ax3.set_title('Imagen Reconstruída')  
plt.show()
```



Transformada de Fourier aplicada as imagens disponibilizadas mostrando o espectro de magnitude e de fase

```
import numpy as np
import matplotlib.pyplot as plt

# Carregue a imagem (você precisa de uma imagem para isso)
imagem = plt.imread('periodic_noise.png')

# Converta a imagem para tons de cinza se necessário
if imagem.ndim == 3:
    imagem = np.mean(imagem, axis=2)

# Calcule a Transformada de Fourier 2D da imagem
transformada_fourier = np.fft.fft2(imagem)

# Desloque a frequência zero para o centro da imagem
transformada_fourier_deslocada =
np.fft.fftshift(transformada_fourier)

# Calcule o espectro de magnitude (magnitude da Transformada de Fourier)
espectro_magnitude = np.abs(transformada_fourier_deslocada)

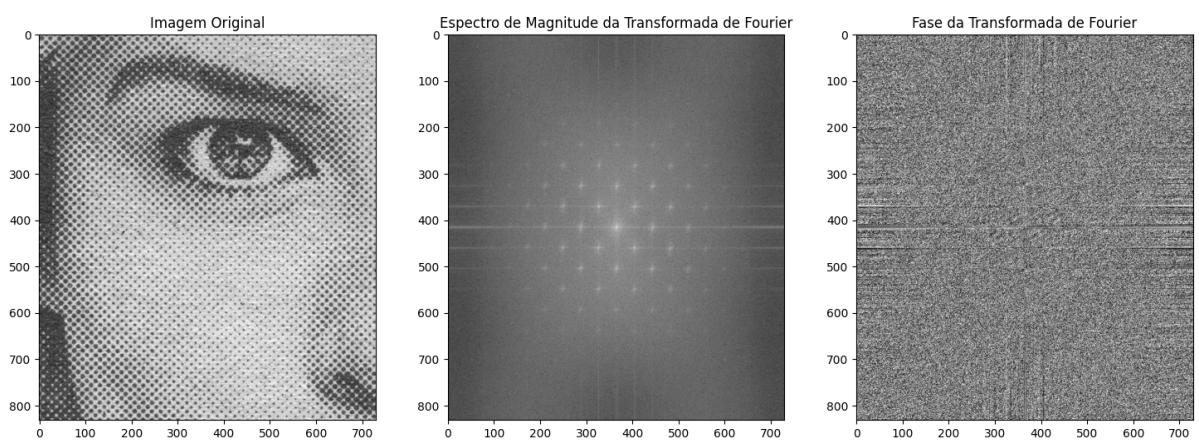
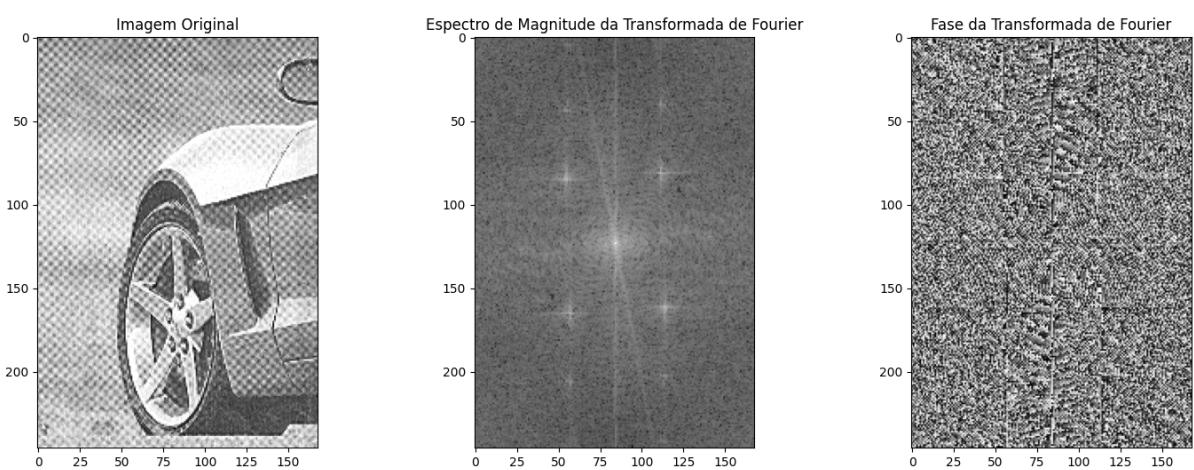
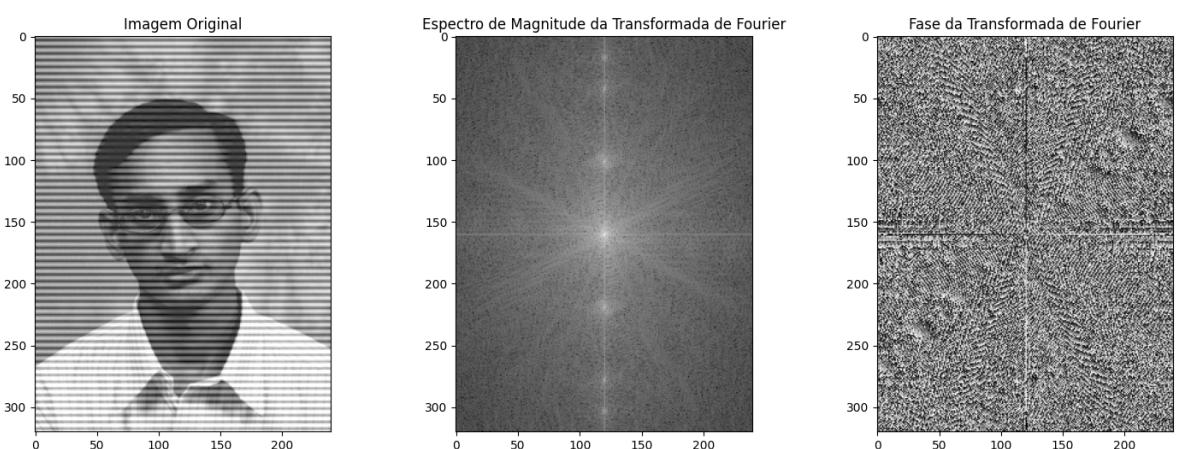
# Calcule a fase da Transformada de Fourier
fase_fourier = np.angle(transformada_fourier_deslocada)

# Crie uma figura com três subtramas (imagem original, espectro e fase)
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 6))

# Mostre a imagem original na primeira subtrama
ax1.imshow(imagem, cmap='gray')
ax1.set_title('Imagen Original')

# Mostre o espectro de magnitude na segunda subtrama
ax2.imshow(np.log(espectro_magnitude), cmap='gray')
ax2.set_title('Espectro de Magnitude da Transformada de Fourier')

# Mostre a fase na terceira subtrama
ax3.imshow(fase_fourier, cmap='gray')
ax3.set_title('Fase da Transformada de Fourier')
plt.show()
```



Plot 3D da transformada de Fourier aplicada as imagens disponibilizadas

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Carregue a imagem (você precisa de uma imagem para isso)
imagem = plt.imread('periodic_noise.png')

# Converta a imagem para tons de cinza se necessário
if imagem.ndim == 3:
    imagem = np.mean(imagem, axis=2)

# Calcule a Transformada de Fourier 2D da imagem
transformada_fourier = np.fft.fft2(imagem)

# Desloque a frequência zero para o centro da imagem
transformada_fourier_deslocada =
np.fft.fftshift(transformada_fourier)

# Calcule o espectro de magnitude (magnitude da Transformada de
Fourier)
espectro_magnitude = np.abs(transformada_fourier_deslocada)

# Crie uma grade de coordenadas para o espectro 3D
coord_x = np.arange(-imagem.shape[1] // 2, imagem.shape[1] // 2)
coord_y = np.arange(-imagem.shape[0] // 2, imagem.shape[0] // 2)
coord_x, coord_y = np.meshgrid(coord_x, coord_y)

# Crie uma figura com duas subtramas (uma para a imagem e outra
para o espectro 3D)
fig = plt.figure(figsize=(15, 6))

# Subtrama para a imagem original
ax1 = fig.add_subplot(121)
ax1.imshow(imagem, cmap='gray')
ax1.set_title('Imagen Original')

# Subtrama para o espectro 3D
ax2 = fig.add_subplot(122, projection='3d')
ax2.plot_surface(coord_x, coord_y, np.log(espectro_magnitude),
cmap='viridis')
```

```

# Defina os rótulos dos eixos para a subtrama do espectro 3D
ax2.set_xlabel('Frequência Horizontal')
ax2.set_ylabel('Frequência Vertical')
ax2.set_zlabel('Log(Espectro de Magnitude)')

# Gire a visualização para uma melhor perspectiva (opcional)
ax2.view_init(elev=20, azim=30)

plt.show()

```

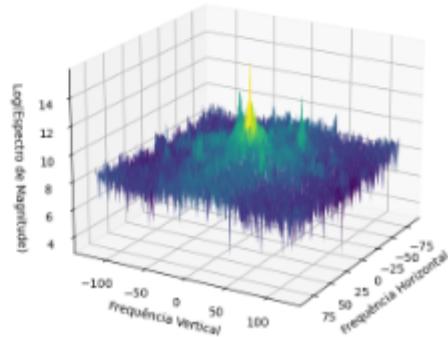
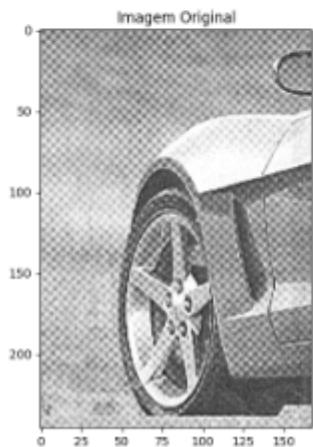
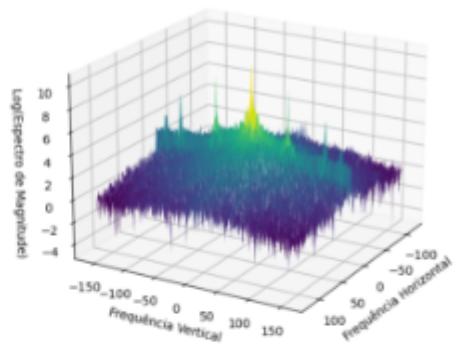


Imagen utilizando a função sinc

```
import numpy as np
import matplotlib.pyplot as plt

# Tamanho da imagem
largura = 512
altura = 512

# Crie uma matriz de zeros para representar o fundo branco
imagem = np.zeros((altura, largura))

# Defina as coordenadas do quadrado
x_min = largura // 4
x_max = 3 * largura // 4
y_min = altura // 4
y_max = 3 * altura // 4

# Desenhe o quadrado na imagem com altura dada pela função sinc
for x in range(x_min, x_max):
    for y in range(y_min, y_max):
        distancia_x = (x - largura // 2) / (largura // 2)
        distancia_y = (y - altura // 2) / (altura // 2)
        sinc_value = np.sinc(distancia_x) * np.sinc(distancia_y)
        imagem[y, x] = sinc_value

# Crie a figura e mostre a imagem
plt.imshow(imagem, cmap='gray')
plt.title('Quadrado Simulando Função Sinc')
plt.axis('off') # Oculta os eixos
plt.show()
```

