
PROCESSAMENTO DIGITAL DE IMAGENS

Carlos Vinicius Baggio Savian, BI3002217

# Atividades - Filtragem Frequência

1. Calcule e visualize o espectro de uma imagem 512x512 pixels:

a) crie e visualize uma imagem simples – quadrado branco sobre fundo preto;

b) calcular e visualizar seu espectro de Fourier (amplitudes);

c) calcular e visualizar seu espectro de Fourier (fases);

d) obter e visualizar seu espectro de Fourier centralizado;

e) Aplique uma rotação de 40º no quadrado e repita os passo b-d;

f) Aplique uma translação nos eixos x e y no quadrado e repita os passo b-d;

g) Aplique um zoom na imagem e repita os passo b-d;

h) Explique o que acontece com a transformada de Fourier quando é aplicado a rotação, translação e zoom.

2. Crie filtros passa-baixa do tipo ideal, butterworth e gaussiano e aplique-o às imagens disponibilizadas. Visualize o seguinte:

a) a imagem inicial;

b) a imagem do spectro de fourier;

c) a imagem de cada filtro;

d) a imagem resultante após aplicação de cada filtro.

3. Crie um filtro passa-alta do tipo ideal, butterworth e gaussiano e aplique-o às imagens disponibilizadas. Visualize os mesmos dados da tarefa anterior:

a) a imagem inicial;

b) a imagem do spectro de fourier;

c) a imagem de cada filtro;

d) a imagem resultante após aplicação de cada filtro.

4. Varie o parâmetro de frequência de corte no filtro passa-baixa criado na tarefa 2. Por exemplo, tome valores de D0 iguais a 0,01, 0,05, 0,5. A imagem inicial é igual à anterior. Visualize as imagens dos filtros e as imagens resultantes. Explique os resultados.

5. Efetue o mesmo que se pede no item 4, mas use o filtro passa-alta em vez do filtro passa-baixa.

6. Além dos filtros passa-baixa e passa-alta também existe o filtro passa-banda? Explique seu funcionamento e aplique um filtro passa-banda na imagem.

1)
a – g) Os códigos e imagens estão no final do arquivo.

h) A transformada de Fourier é afetada pela rotação, translação e zoom, resultando em mudanças nas frequências e fases.

2)
a-d) Os códigos e imagens estão no final do arquivo.

3)
a-d) Os códigos e imagens estão no final do arquivo.

4) Os códigos e imagens estão no final do arquivo.
Variando a frequência de corte do filtro, o tamanho do filtro também variou, alterando assim as frequências que passarão por ele, fazendo com que a imagem resultante varie de acordo com isso.//A variação do parâmetro de frequência de corte em filtros passa-baixa permite controlar quais frequências são retidas ou atenuadas na imagem. Valores maiores de D0 permitem passar frequências mais altas, preservando detalhes.

5) Os códigos e imagens estão no final do arquivo.
Assim como na pergunta 4, variando a frequência de corte do filtro, o tamanho do filtro também variou, alterando assim as frequências que passarão por ele, fazendo com que a imagem resultante varie de acordo com isso, mas dessa vez o efeito é o oposto.//A variação do parâmetro de frequência de corte em filtros passa-alta controla a ênfase nas frequências de alta magnitude. Valores maiores de D0 em filtros passa-alta destacam detalhes de alta frequência.

6) Os códigos e imagens estão no final do arquivo.
Sim, o filtro passa-banda também existe e é usado para permitir somente a passagem de determinadas frequências.//Um filtro passa-banda é usado para selecionar uma faixa específica de frequências na imagem. Ele permite a passagem apenas das frequências dentro dessa faixa, bloqueando as outras. Isso é útil para destacar características específicas da imagem que estão dentro de um intervalo de frequência desejado.

codigo ex1)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# A)

# Tamanho da matriz da imagem a ser criada
height = 512
width = 512

# Encontrando o centro da matriz da imagem a ser criada
x = height//2
y = width//2

# Definindo o tamanho do quadrado que ficará no centro da imagem (quadrado terá tamanho de 200x200 pixels)
# (será dividido por 2 porque o quadrado será criado a partir do pixel central da matriz)
size = 200//2
```

```
# Imprime posição do pixel central da imagem
print(x,y)

# Cria matriz de zeros no tamanho 512x512 pixels
black_whiteSquare = np.zeros(shape=[512, 512, 3], dtype=np.uint8)

# Cria um quadrado dentro da matriz a partir da posição do pixel central da imagem
# cv2.rectangle(image, start_point, end_point, color, thickness)
# Obs: thickness = -1 -> Imagem preenchida)
cv2.rectangle(black_whiteSquare, pt1=(x-size,y-size), pt2=(x+size,y+size), color=(255,255,255),
thickness=-1)

# Converte imagem BGR (RGB no formato cv2) para Escala de Cinza
# cv2.cvtColor(image, color_space_conversion_code)
whiteblankimage = cv2.cvtColor(black_whiteSquare, cv2.COLOR_BGR2GRAY)

# Imprime imagem
plt.imshow(black_whiteSquare)
plt.show()

# Salva imagem
cv2.imwrite("black_whiteSquare.tif", black_whiteSquare)

# Cria figura
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

# Carrega Imagem em formato de escala de cinza
img = cv2.imread("black_whiteSquare.tif", 0)

# Plota a imagem carregada como Imagem Original
# plt.subplot(linhas|colunas|posição do plot) -> Ex: plt.subplot(211) = linha 2, coluna 1, posição
1
plt.subplot(151), plt.imshow(img,'gray'), plt.title('Imagem Original')

# B)
img_fft_amplitude = np.fft.fft2(img)
plt.subplot(152), plt.imshow(np.log(1+np.abs(img_fft_amplitude)), "gray"), plt.title("Espectro
Amplitude")

# C)
img_fft_fase = np.angle(img_fft_amplitude)
plt.subplot(153), plt.imshow(img_fft_fase, "gray"), plt.title("Espectro Fase")

# D)
img_centralizado = np.fft.fftshift(img_fft_amplitude)
```

```
plt.subplot(154), plt.imshow(np.log(1+np.abs(img_centralizado)), "gray"), plt.title("Espectro Centralizado")

plt.savefig('ex1_a_d.tif')
plt.show()

# E)
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

imagem = cv2.imread('black_whiteSquare.tif')
altura, largura = imagem.shape[:2]
cv2.waitKey(0)

#rotacao
ponto = (largura / 2, altura / 2) #ponto no centro da figura
rotacao = cv2.getRotationMatrix2D(ponto, 40, 1.0)
rotacionado = cv2.warpAffine(imagem, rotacao, (largura, altura))
cv2.imshow("Rotacionado 40 graus", rotacionado)
cv2.waitKey(0)

cv2.imwrite('black_whiteSquare_40.tif', rotacionado)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img_rotacionada40 = cv2.imread('black_whiteSquare_40.tif', 0)
plt.subplot(151), plt.imshow(img_rotacionada40, "gray"), plt.title("Rotação 40º")

img_fft_amplitude = np.fft.fft2(img_rotacionada40)
plt.subplot(152), plt.imshow(np.log(1+np.abs(img_fft_amplitude)), "gray"), plt.title("Espectro Amplitude")

img_fft_fase = np.angle(img_fft_amplitude)
plt.subplot(153), plt.imshow(img_fft_fase, "gray"), plt.title("Espectro Fase")

img_centralizado = np.fft.fftshift(img_fft_amplitude)
plt.subplot(154), plt.imshow(np.log(1+np.abs(img_centralizado)), "gray"), plt.title("Espectro Centralizado")
plt.savefig('ex1_e.tif')
plt.show()

# F)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

# translacao (deslocamento)
deslocamento = np.float32([[1, 0, -50], [0, 1, -90]])
```

```
deslocado = cv2.warpAffine(imagem, deslocamento, (largura, altura))
cv2.imshow("Cima e esquerda", deslocado)
cv2.waitKey(0)

cv2.imwrite('black_whiteSquare_transladada.tif', deslocado)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img_deslocada = cv2.imread('black_whiteSquare_transladada.tif', 0)
plt.subplot(151), plt.imshow(img_deslocada, "gray"), plt.title("Imagem Transladada")

img_fft_amplitude = np.fft.fft2(img_deslocada)
plt.subplot(152), plt.imshow(np.log(1+np.abs(img_fft_amplitude)), "gray"), plt.title("Espectro
Amplitude")

img_fft_fase = np.angle(img_fft_amplitude)
plt.subplot(153), plt.imshow(img_fft_fase, "gray"), plt.title("Espectro Fase")

img_centralizado = np.fft.fftshift(img_fft_amplitude)
plt.subplot(154), plt.imshow(np.log(1+np.abs(img_centralizado)), "gray"), plt.title("Espectro
Centralizado")
plt.savefig('ex1_f.tif')
plt.show()
```

---

## Ex2)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, exp


def distance(point1, point2):
    return sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)


def idealFilterLP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            if distance((y, x), center) < D0:
                base[y, x] = 1
    return base
```

```
def butterworthLP(D0, imgShape, n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 / (1 + (distance((y, x), center) / D0) ** (2 * n))
    return base


def gaussianLP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = exp(((-distance((y, x), center) ** 2) / (2 * (D0 ** 2))))
    return base


plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img = cv2.imread("lena.jpg", 0)

# A
plt.subplot(151), plt.imshow(img, "gray"), plt.title("Imagem Original")

img_fft = np.fft.fft2(img)
img_fft_centralizada = np.fft.fftshift(img_fft)

#B
passaBaixaIdeal = idealFilterLP(50, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaBaixaIdeal), "gray"), plt.title("Filtro Ideal")

passaBaixaButterworth = butterworthLP(50, img.shape, 20)
plt.subplot(153), plt.imshow(np.abs(passaBaixaButterworth), "gray"), plt.title("Filtro
Butterworth")

passaBaixaGaussiano = gaussianLP(50, img.shape)
plt.subplot(154), plt.imshow(np.abs(passaBaixaGaussiano), "gray"), plt.title("Filtro Gaussiano")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

#C
```

```
img_centralizada_ideal = img_fft_centralizada * passaBaixaIdeal
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(161), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro Ideal")

img_centralizada_butterworth = img_fft_centralizada * passaBaixaButterworth
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(162), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth")

img_centralizada_gaussiano = img_fft_centralizada * passaBaixaGaussiano
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(163), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem Filtro Gaussiano")

plt.show()
```

---

## Ex3)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, exp


def distance(point1, point2):
    return sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)


def idealFilterHP(D0, imgShape):
    base = np.ones(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            if distance((y, x), center) < D0:
                base[y, x] = 0
    return base


def butterworthHP(D0, imgShape, n):
```

```
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 - 1 / (1 + (distance((y, x), center) / D0) ** (2 * n))
    return base


def gaussianHP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 - exp(((-distance((y, x), center) ** 2) / (2 * (D0 ** 2))))
    return base


plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img = cv2.imread("lena.jpg", 0)

# A
plt.subplot(151), plt.imshow(img, "gray"), plt.title("Imagem Original")

img_fft = np.fft.fft2(img)
img_fft_centralizada = np.fft.fftshift(img_fft)

#B
passaAltaIdeal = idealFilterHP(50, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaAltaIdeal), "gray"), plt.title("Filtro Ideal")

passaAltaButterworth = butterworthHP(50, img.shape, 20)
plt.subplot(153), plt.imshow(np.abs(passaAltaButterworth), "gray"), plt.title("Filtro
Butterworth")

passaAltaGaussiano = gaussianHP(50, img.shape)
plt.subplot(154), plt.imshow(np.abs(passaAltaGaussiano), "gray"), plt.title("Filtro Gaussiano")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

#C
img_centralizada_ideal = img_fft_centralizada * passaAltaIdeal
```

```
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(161), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro
Ideal")

img_centralizada_butterworth = img_fft_centralizada * passaAltaButterworth
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(162), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem
Filtro Butterworth")

img_centralizada_gaussiano = img_fft_centralizada * passaAltaGaussiano
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(163), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem
Filtro Gaussiano")

plt.show()
```

---

## Ex4

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, exp


def distance(point1, point2):
    return sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)


def idealFilterLP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            if distance((y, x), center) < D0:
                base[y, x] = 1
    return base


def butterworthLP(D0, imgShape, n):
    base = np.zeros(imgShape[:2])
```

```
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 / (1 + (distance((y, x), center) / D0) ** (2 * n))
    return base


def gaussianLP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = exp(((-distance((y, x), center) ** 2) / (2 * (D0 ** 2))))
    return base


plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img = cv2.imread("lena.jpg", 0)

plt.subplot(151), plt.imshow(img, "gray"), plt.title("Imagem Original")

img_fft = np.fft.fft2(img)
img_fft_centralizada = np.fft.fftshift(img_fft)

passaBaixaIdeal1 = idealFilterLP(25, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaBaixaIdeal1), "gray"), plt.title("Filtro Ideal 25")

passaBaixaIdeal2 = idealFilterLP(1, img.shape)
plt.subplot(153), plt.imshow(np.abs(passaBaixaIdeal2), "gray"), plt.title("Filtro Ideal 1")

passaBaixaIdeal3 = idealFilterLP(0.5, img.shape)
plt.subplot(154), plt.imshow(np.abs(passaBaixaIdeal3), "gray"), plt.title("Filtro Ideal 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_ideal = img_fft_centralizada * passaBaixaIdeal1
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(161), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro
Ideal 25")
```

```
img_centralizada_ideal = img_fft_centralizada * passaBaixaIdeal2
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(162), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro Ideal 1")

img_centralizada_ideal = img_fft_centralizada * passaBaixaIdeal3
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(163), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro Ideal 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

passaBaixaButterworth1 = butterworthLP(25, img.shape, 20)
plt.subplot(151), plt.imshow(np.abs(passaBaixaButterworth1), "gray"), plt.title("Filtro Butterworth 25")

passaBaixaButterworth2 = butterworthLP(1, img.shape, 20)
plt.subplot(152), plt.imshow(np.abs(passaBaixaButterworth2), "gray"), plt.title("Filtro Butterworth 1")

passaBaixaButterworth3 = butterworthLP(0.5, img.shape, 20)
plt.subplot(153), plt.imshow(np.abs(passaBaixaButterworth3), "gray"), plt.title("Filtro Butterworth 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_butterworth = img_fft_centralizada * passaBaixaButterworth1
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(161), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 25")

img_centralizada_butterworth = img_fft_centralizada * passaBaixaButterworth2
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(162), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 1")

img_centralizada_butterworth = img_fft_centralizada * passaBaixaButterworth3
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
```

```
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(163), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

passaBaixaGaussiano1 = gaussianLP(25, img.shape)
plt.subplot(151), plt.imshow(np.abs(passaBaixaGaussiano1), "gray"), plt.title("Filtro Gaussiano 25")

passaBaixaGaussiano2 = gaussianLP(1, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaBaixaGaussiano2), "gray"), plt.title("Filtro Gaussiano 1")

passaBaixaGaussiano3 = gaussianLP(0.5, img.shape)
plt.subplot(153), plt.imshow(np.abs(passaBaixaGaussiano3), "gray"), plt.title("Filtro Gaussiano 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_gaussiano = img_fft_centralizada * passaBaixaGaussiano1
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(161), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem Filtro Gaussiano 25")

img_centralizada_gaussiano = img_fft_centralizada * passaBaixaGaussiano2
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(162), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem Filtro Gaussiano 1")

img_centralizada_gaussiano = img_fft_centralizada * passaBaixaGaussiano3
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(163), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem Filtro Gaussiano 0,5")

plt.show()
```

---

## Ex5

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, exp


def distance(point1, point2):
    return sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)


def idealFilterHP(D0, imgShape):
    base = np.ones(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            if distance((y, x), center) < D0:
                base[y, x] = 0
    return base


def butterworthHP(D0, imgShape, n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 - 1 / (1 + (distance((y, x), center) / D0) ** (2 * n))
    return base


def gaussianHP(D0, imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows / 2, cols / 2)
    for x in range(cols):
        for y in range(rows):
            base[y, x] = 1 - exp(((-distance((y, x), center) ** 2) / (2 * (D0 ** 2))))
    return base


plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img = cv2.imread("lena.jpg", 0)
```

```
plt.subplot(151), plt.imshow(img, "gray"), plt.title("Imagem Original")

img_fft = np.fft.fft2(img)
img_fft_centralizada = np.fft.fftshift(img_fft)

passaAltaIdeal1 = idealFilterHP(25, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaAltaIdeal1), "gray"), plt.title("Filtro Ideal 25")

passaAltaIdeal2 = idealFilterHP(1, img.shape)
plt.subplot(153), plt.imshow(np.abs(passaAltaIdeal2), "gray"), plt.title("Filtro Ideal 1")

passaAltaIdeal3 = idealFilterHP(0.5, img.shape)
plt.subplot(154), plt.imshow(np.abs(passaAltaIdeal3), "gray"), plt.title("Filtro Ideal 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_ideal = img_fft_centralizada * passaAltaIdeal1
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(161), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro
Ideal 25")

img_centralizada_ideal = img_fft_centralizada * passaAltaIdeal2
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(162), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro
Ideal 1")

img_centralizada_ideal = img_fft_centralizada * passaAltaIdeal3
img_ideal = np.fft.ifftshift(img_centralizada_ideal)
img_processada_ideal = np.fft.ifft2(img_ideal)
plt.subplot(163), plt.imshow(np.abs(img_processada_ideal), "gray"), plt.title("Imagem Filtro
Ideal 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

passaAltaButterworth1 = butterworthHP(25, img.shape, 20)
plt.subplot(151), plt.imshow(np.abs(passaAltaButterworth1), "gray"), plt.title("Filtro
Butterworth 25")

passaAltaButterworth2 = butterworthHP(1, img.shape, 20)
```

```
plt.subplot(152), plt.imshow(np.abs(passaAltaButterworth2), "gray"), plt.title("Filtro Butterworth 1")

passaAltaButterworth3 = butterworthHP(0.5, img.shape, 20)
plt.subplot(153), plt.imshow(np.abs(passaAltaButterworth3), "gray"), plt.title("Filtro Butterworth 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_butterworth = img_fft_centralizada * passaAltaButterworth1
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(161), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 25")

img_centralizada_butterworth = img_fft_centralizada * passaAltaButterworth2
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(162), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 1")

img_centralizada_butterworth = img_fft_centralizada * passaAltaButterworth3
img_butterworth = np.fft.ifftshift(img_centralizada_butterworth)
img_processada_butterworth = np.fft.ifft2(img_butterworth)
plt.subplot(163), plt.imshow(np.abs(img_processada_butterworth), "gray"), plt.title("Imagem Filtro Butterworth 0,5")

plt.show()

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

passaAltaGaussiano1 = gaussianHP(25, img.shape)
plt.subplot(151), plt.imshow(np.abs(passaAltaGaussiano1), "gray"), plt.title("Filtro Gaussiano 25")

passaAltaGaussiano2 = gaussianHP(1, img.shape)
plt.subplot(152), plt.imshow(np.abs(passaAltaGaussiano2), "gray"), plt.title("Filtro Gaussiano 1")

passaAltaGaussiano3 = gaussianHP(0.5, img.shape)
plt.subplot(153), plt.imshow(np.abs(passaAltaGaussiano3), "gray"), plt.title("Filtro Gaussiano 0,5")

plt.show()
```

```
plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

img_centralizada_gaussiano = img_fft_centralizada * passaAltaGaussiano1
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(161), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem
Filtro Gaussiano 25")

img_centralizada_gaussiano = img_fft_centralizada * passaAltaGaussiano2
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(162), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem
Filtro Gaussiano 1")

img_centralizada_gaussiano = img_fft_centralizada * passaAltaGaussiano3
img_gaussiano = np.fft.ifftshift(img_centralizada_gaussiano)
img_processada_gaussiano = np.fft.ifft2(img_gaussiano)
plt.subplot(163), plt.imshow(np.abs(img_processada_gaussiano), "gray"), plt.title("Imagem
Filtro Gaussiano 0,5")

plt.show()
```

---

## Ex6

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from skimage.filters import difference_of_gaussians, window
from scipy.fftpack import fftn, fftshift

img = cv2.imread("lena.jpg", 0)
wimage = img * window('hann', img.shape)
img_filtrada = difference_of_gaussians(img, 1, 20)
wimage_filtrada = img_filtrada * window('hann', img.shape)
im_fft_mag = fftshift(np.abs(fftn(wimage)))
fim_fft_mag = fftshift(np.abs(fftn(wimage_filtrada)))

plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)

plt.subplot(161), plt.imshow(img, "gray"), plt.title("Imagem Original")
plt.subplot(162), plt.imshow(np.log(im_fft_mag), "gray"), plt.title("Fourier Amplitude")
plt.subplot(163), plt.imshow(img_filtrada, "gray"), plt.title("Imagem Filtrada")
```

plt.subplot(164), plt.imshow(np.log(fim_fft_mag), "gray"), plt.title("Fourier Amplitude Imagem Filtrada")

plt.show()

1. **Carregamento da Imagem**:
   ```python
   img = cv2.imread("lena.jpg", 0)
   ```
   - A imagem "lena.jpg" é carregada em tons de cinza (0 indica que a imagem deve ser carregada em tons de cinza).

2. **Aplicação da Janela de Hann**:
   ```python
   wimage = img * window('hann', img.shape)
   ```
   - Uma janela de Hann é aplicada à imagem original para reduzir artefatos de borda na Transformada de Fourier.

3. **Filtragem usando o Diferença de Gaussiana**:
   ```python
   img_filtrada = difference_of_gaussians(img, 1, 20)
   ```
   - O filtro de diferença de gaussianas é aplicado à imagem original. Esse filtro realça as bordas e detalhes na imagem.

4. **Aplicação da Janela de Hann à Imagem Filtrada**:
   ```python
   wimage_filtrada = img_filtrada * window('hann', img.shape)
   ```
   - Uma janela de Hann também é aplicada à imagem filtrada para reduzir artefatos de borda na Transformada de Fourier.

5. **Cálculo da Transformada de Fourier e Amplitude do Espectro**:
   ```python
   im_fft_mag = fftshift(np.abs(fftn(wimage)))
   ```
   - É calculada a Transformada de Fourier da imagem original com a janela de Hann aplicada, e a amplitude do espectro é obtida. O `np.abs` é usado para calcular a magnitude.

6. **Cálculo da Transformada de Fourier e Amplitude do Espectro da Imagem Filtrada**:
   ```python
   fim_fft_mag = fftshift(np.abs(fftn(wimage_filtrada)))
   ```

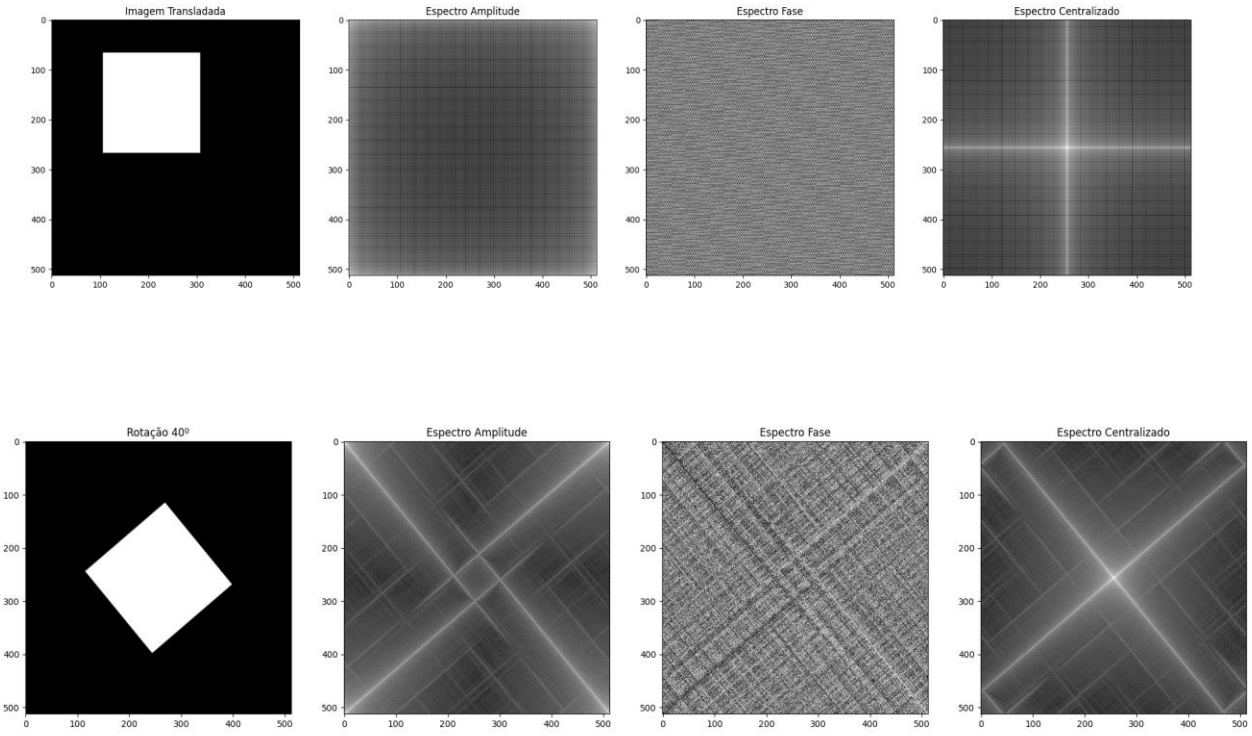
- Similarmente, a Transformada de Fourier da imagem filtrada com a janela de Hann é calculada, e a amplitude do espectro é obtida.

7. **Exibição das Imagens e Espectros**:
   ```python
   plt.figure(figsize=(6.4 * 5, 4.8 * 5), constrained_layout=False)
   ...
   ```
   - As imagens originais, seus espectros de Fourier e as imagens filtradas com seus respectivos espectros de Fourier são exibidos em uma figura grande.

O código mostra a imagem original, o espectro de Fourier da imagem original, a imagem filtrada e o espectro de Fourier da imagem filtrada. A aplicação da janela de Hann ajuda a melhorar a qualidade das imagens nos domínios espacial e de frequência. O filtro de diferença de gaussianas realça características da imagem.

---

# Imagens



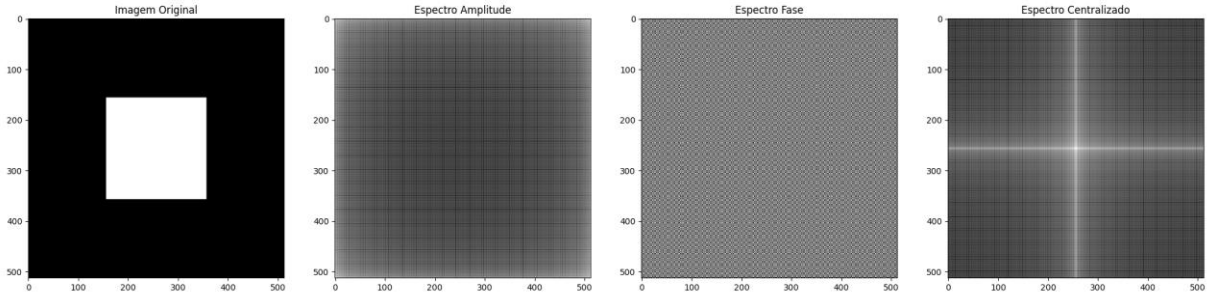
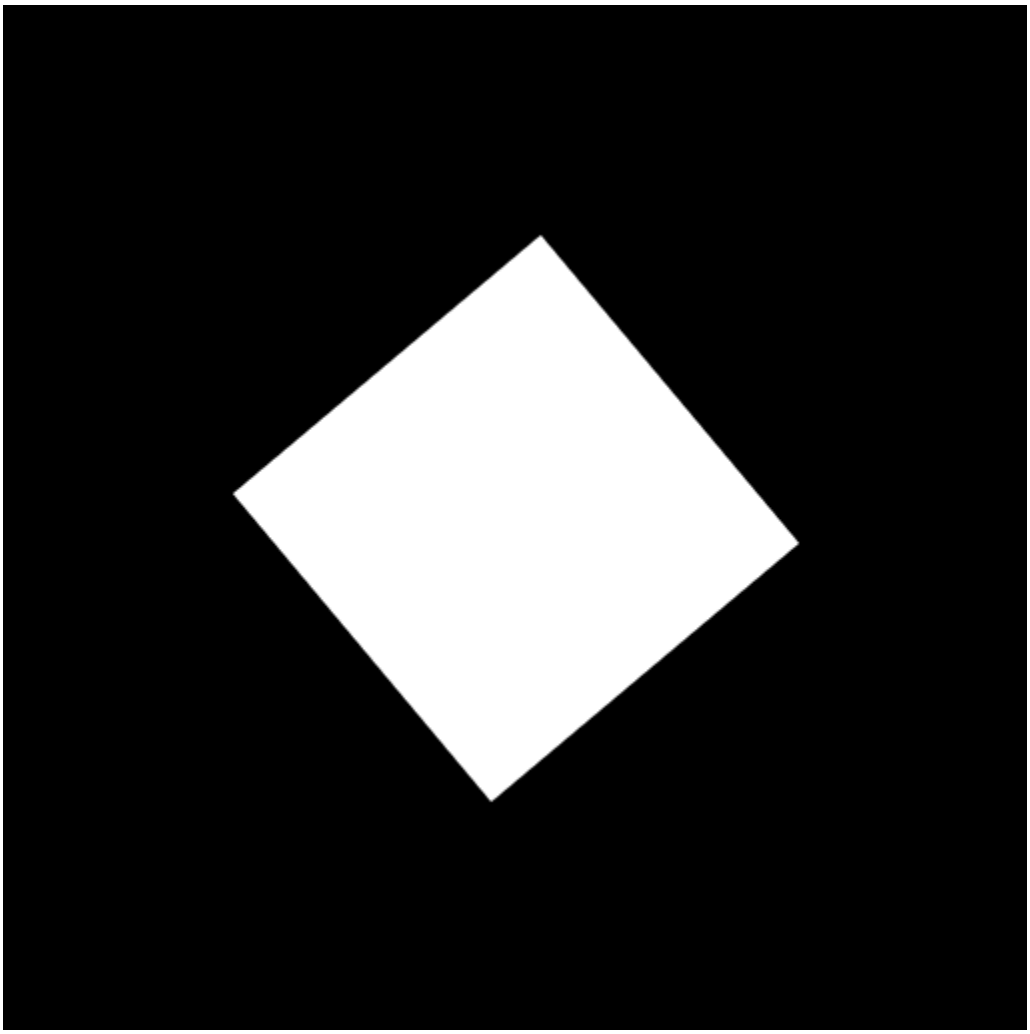
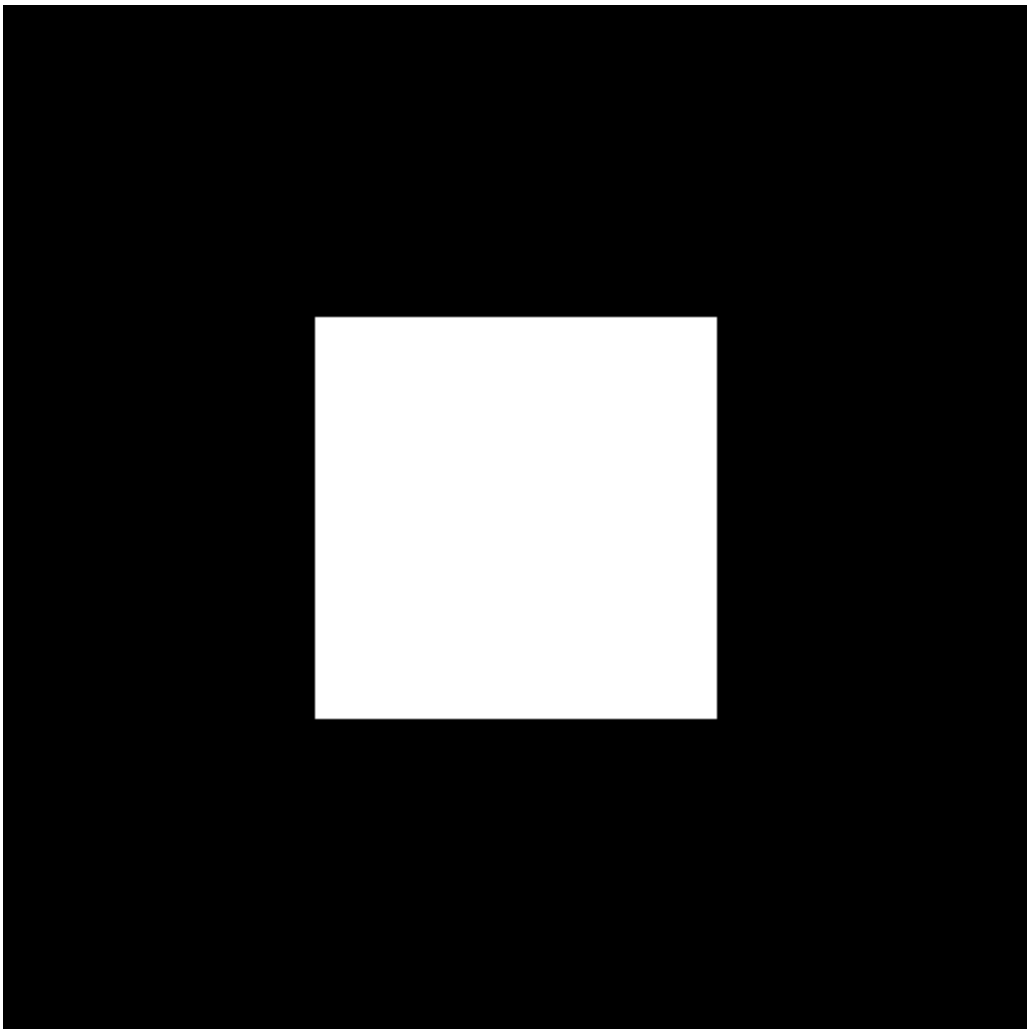
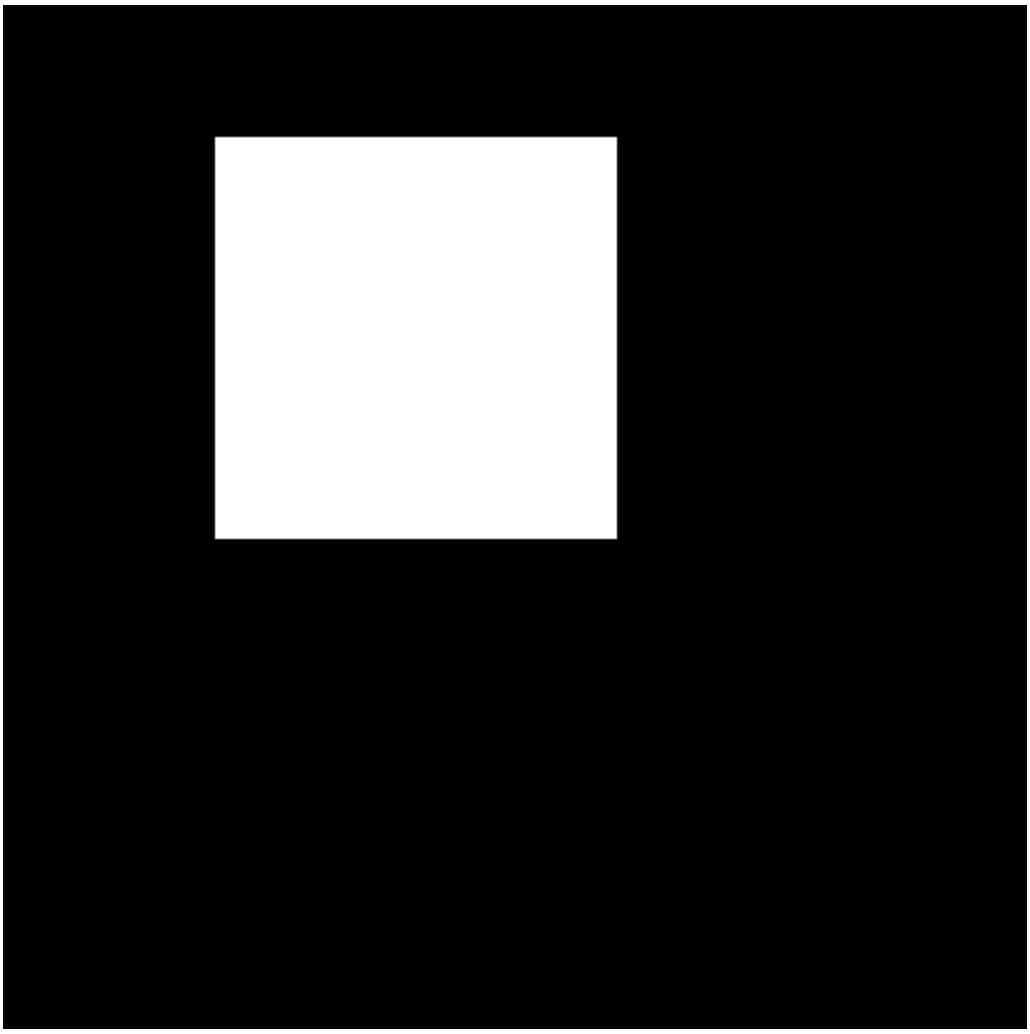


Imagem Original
Espectro Amplitude
Espectro Fase
Espectro Centralizado
0
100
200
300
400
500







# ex1

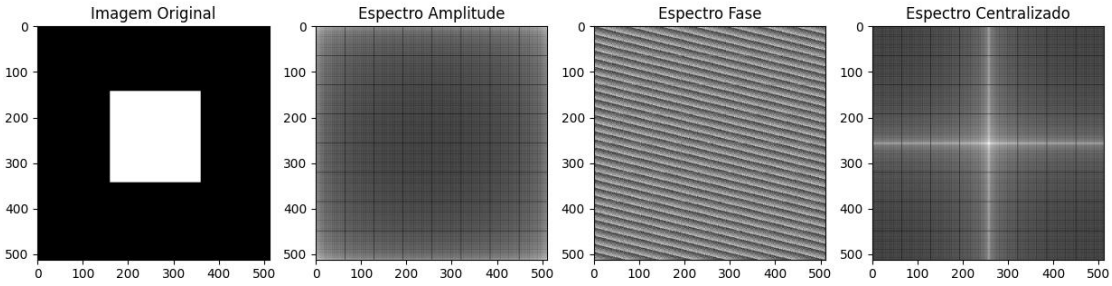


Imagem Original
Espectro Amplitude
Espectro Fase
Espectro Centralizado
0
100
200
300
400
500

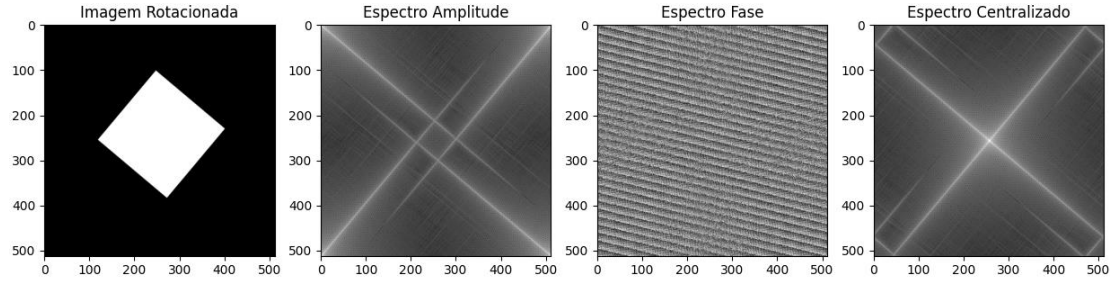


Imagem Rotacionada
Espectro Amplitude
Espectro Fase
Espectro Centralizado

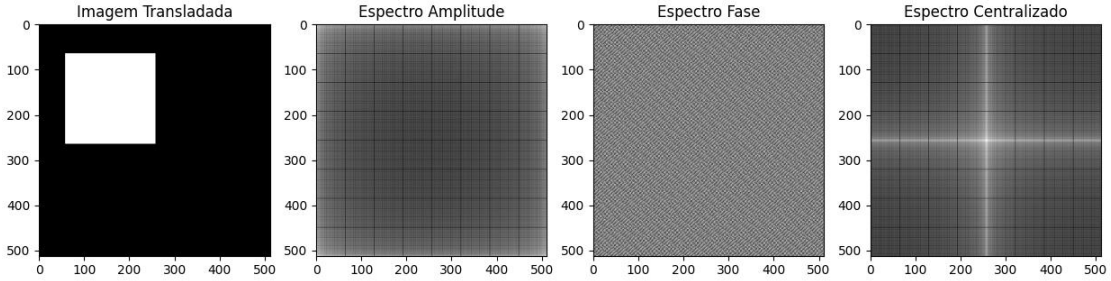


Imagem Transladada
Espectro Amplitude
Espectro Fase
Espectro Centralizado

# Ex2

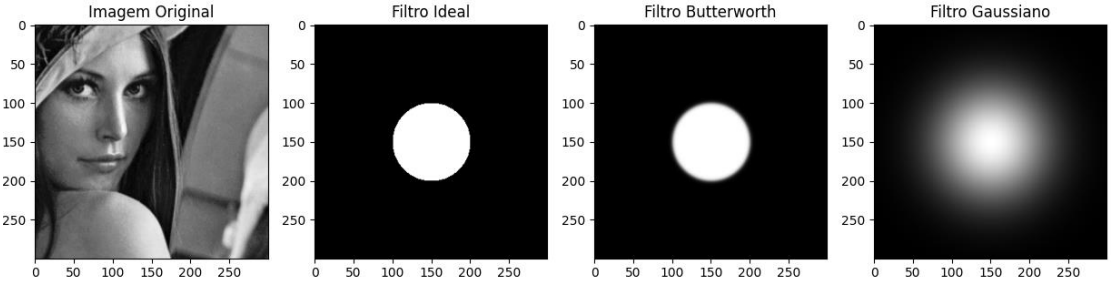


Imagem Original
Filtro Ideal
Filtro Butterworth
Filtro Gaussiano

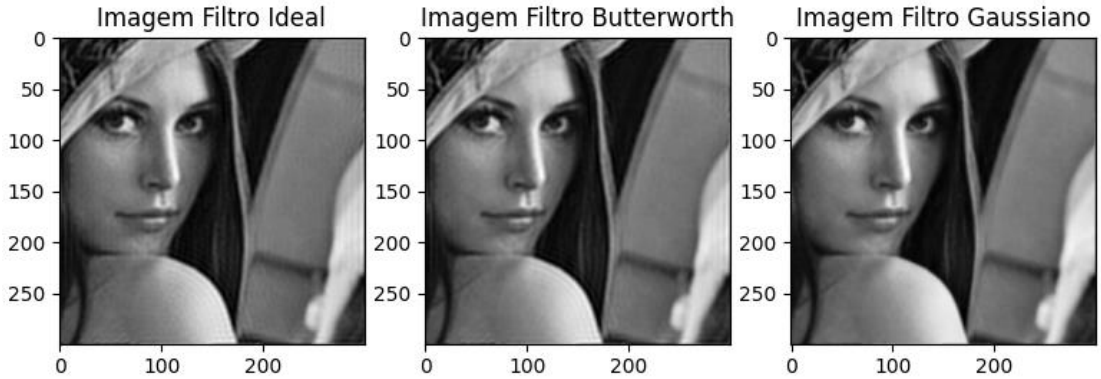


Imagem Filtro Ideal
Imagem Filtro Butterworth
Imagem Filtro Gaussiano

# Ex3

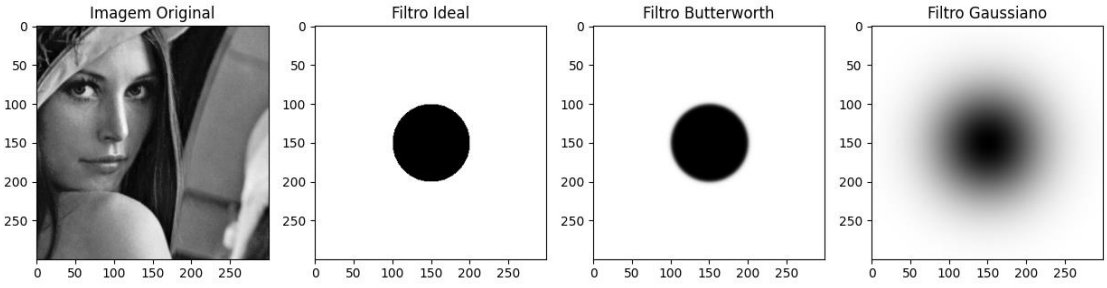


Imagem Original
Filtro Ideal
Filtro Butterworth
Filtro Gaussiano

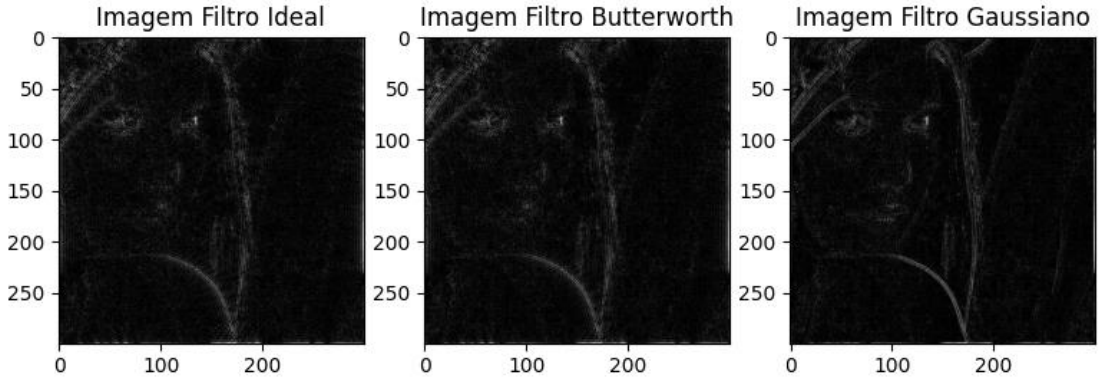


Imagem Filtro Ideal
Imagem Filtro Butterworth
Imagem Filtro Gaussiano

# Ex4

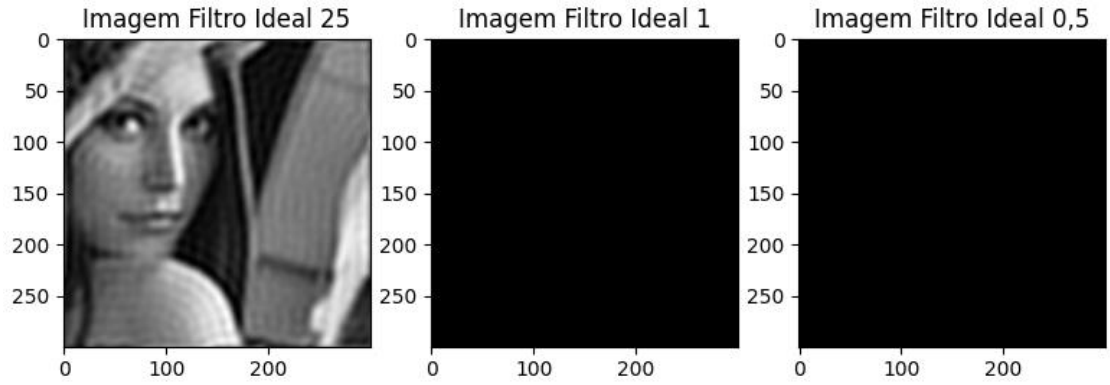


Imagem Filtro Ideal 25
Imagem Filtro Ideal 1
Imagem Filtro Ideal 0,5

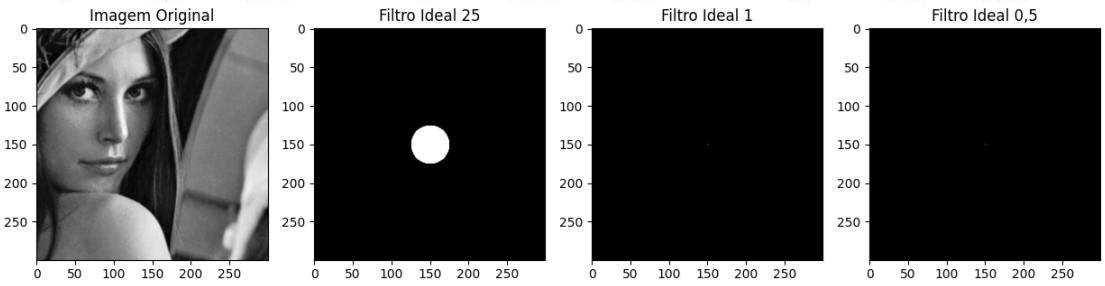


Imagem Original
Filtro Ideal 25
Filtro Ideal 1
Filtro Ideal 0,5

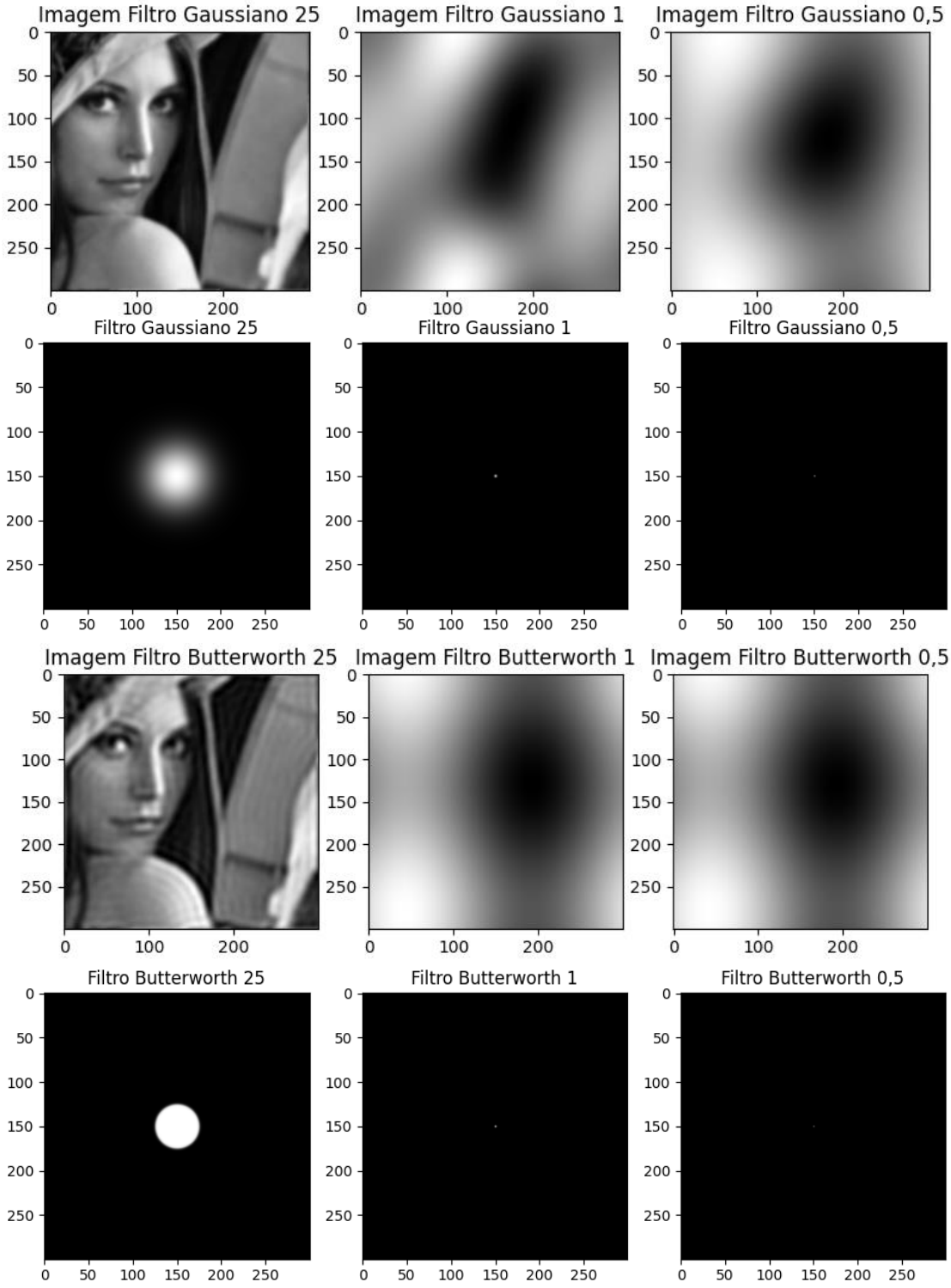


Imagem Filtro Gaussiano 25
Imagem Filtro Gaussiano 1
Imagem Filtro Gaussiano 0,5
Filtro Gaussiano 25
Filtro Gaussiano 1
Filtro Gaussiano 0,5
Imagem Filtro Butterworth 25
Imagem Filtro Butterworth 1
Imagem Filtro Butterworth 0,5
Filtro Butterworth 25
Filtro Butterworth 1
Filtro Butterworth 0,5

# Ex5

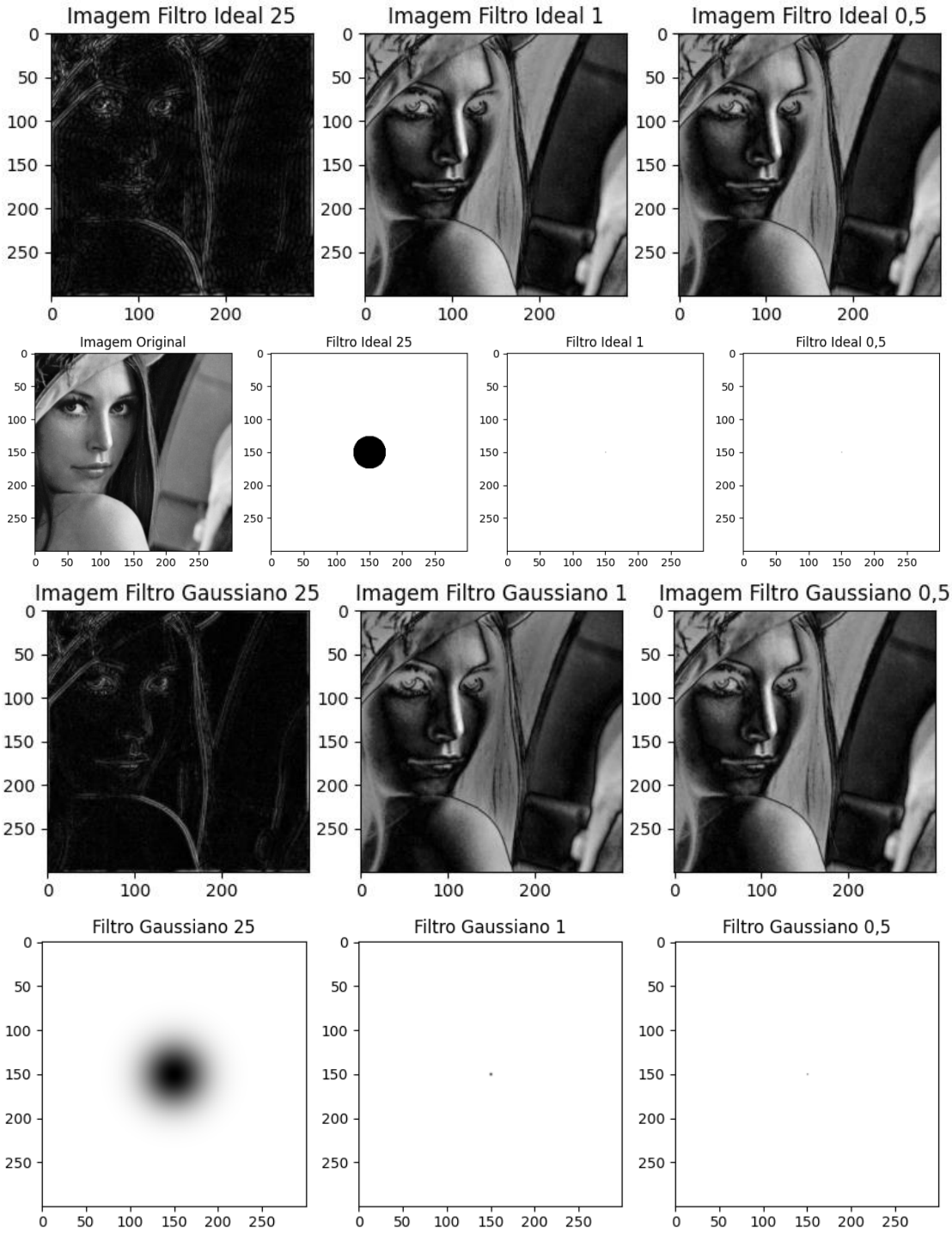


Imagem Filtro Ideal 25
Imagem Filtro Ideal 1
Imagem Filtro Ideal 0,5
Imagem Original
Filtro Ideal 25
Filtro Ideal 1
Filtro Ideal 0,5
Imagem Filtro Gaussiano 25
Imagem Filtro Gaussiano 1
Imagem Filtro Gaussiano 0,5
Filtro Gaussiano 25
Filtro Gaussiano 1
Filtro Gaussiano 0,5

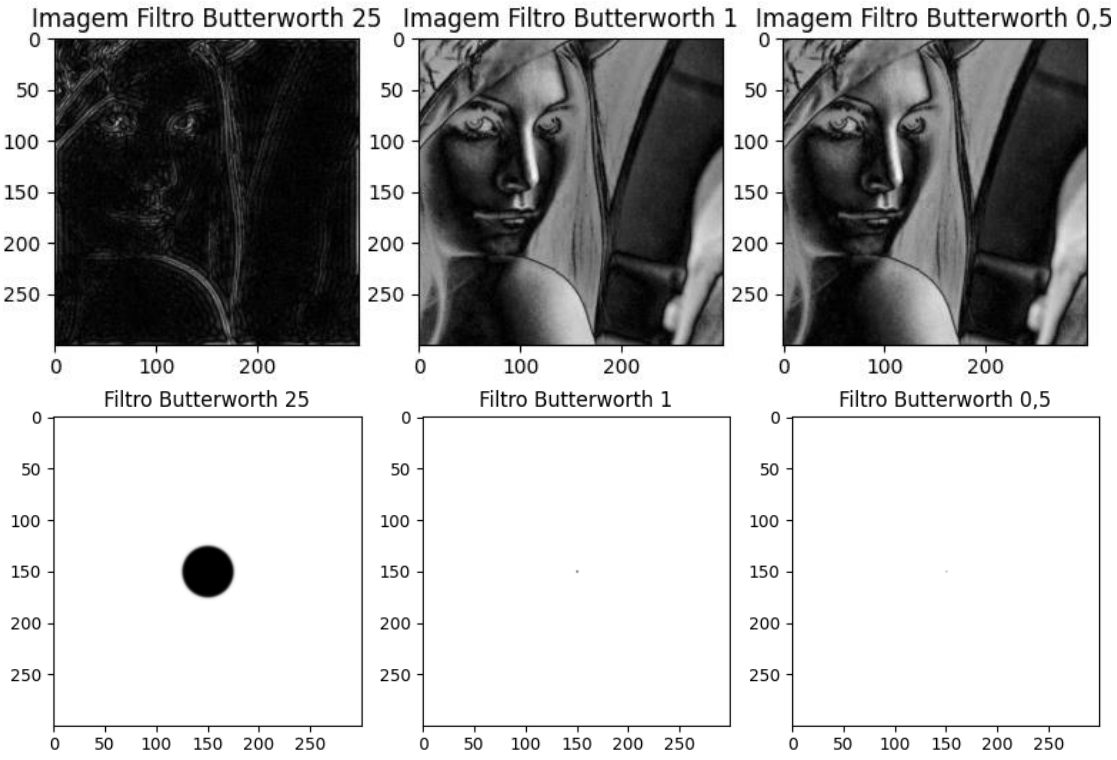


Imagem Filtro Butterworth 25
Imagem Filtro Butterworth 1
Imagem Filtro Butterworth 0,5
Filtro Butterworth 25
Filtro Butterworth 1
Filtro Butterworth 0,5

# Ex6

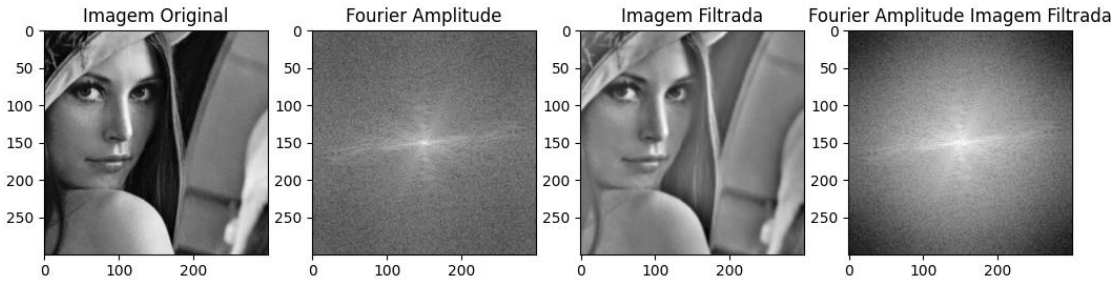


Imagem Original
Fourier Amplitude
Imagem Filtrada
Fourier Amplitude Imagem Filtrada