INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus Birigui	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA** **Campus Birigui** **Bacharelado em Engenharia de Computação**

Disciplina: Processamento Digital de Imagem	**Atividade**
Professor: Prof. Dr. Murilo Varges da Silva	**Data:** 18/09/2023
Nome do Aluno: Henrique Akira Hiraga	**Prontuário:** BI300838X

FILTRAGEM DE FREQUÊNCIA

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

Código da implementação

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw, ImageOps

def zoom():
    # Aplica um zoom na imagem
    fator_zoom = 2.0
    imagem_zoom = imagem.resize((int(largura * fator_zoom),
int(altura * fator_zoom)), Image.BICUBIC)

    # Exibe a imagem com zoom
    plt.subplot(141)
    plt.imshow(imagem_zoom, cmap='gray')
    plt.title('Imagem com Zoom')

    # Converte a imagem com zoom para um array NumPy
    imagem_array_zoom = np.array(imagem_zoom)

    # Calcula a Transformada de Fourier 2D da imagem com
zoom
    espectro_fourier_zoom = np.fft.fft2(imagem_array_zoom)

    # Calcula a amplitude do espectro de Fourier com zoom
    amplitude_zoom = np.abs(espectro_fourier_zoom)

    # Calcula a fase do espectro centralizado
    fase = np.angle(espectro_fourier_zoom)

    # Centraliza o espectro
```

```
    espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier_zoom)

    # Calcula a magnitude do espectro centralizado
    amplitude = np.abs(espectro_fourier_centralizado)

    # Exibe o espectro de amplitude do Fourier com zoom (não
centralizado)
    plt.subplot(142)
    plt.imshow(np.log(amplitude_zoom + 1), cmap='gray')
    plt.title('Espectro de Amplitude (Zoom, log)')
    plt.subplot(143)
    plt.imshow(fase, cmap='gray')
    plt.title('Espectro de Fase')
    plt.subplot(144)
    plt.imshow(np.log(amplitude + 1), cmap='gray')
    plt.title('Espectro de Fourier Centralizado')
    plt.show()

def fourier():
    # Converte a imagem para um array NumPy
    imagem_array = np.array(imagem)

    # Calcula a Transformada de Fourier 2D
    espectro_fourier =
np.fft.fftshift(np.fft.fft2(imagem_array))
    amplitude = np.abs(espectro_fourier)
    fase = np.angle(espectro_fourier)

    # Calcula a fase do espectro
    fase = np.angle(espectro_fourier)

    # Exibe o espectro de amplitude e a fase
    plt.subplot(121)
    plt.imshow(np.log(amplitude + 1), cmap='gray')
    plt.title('Espectro de Amplitude (log)')
    plt.subplot(122)
    plt.imshow(fase, cmap='gray')
    plt.title('Espectro de Fase')
    plt.show()

def fourierCentralizado():
```

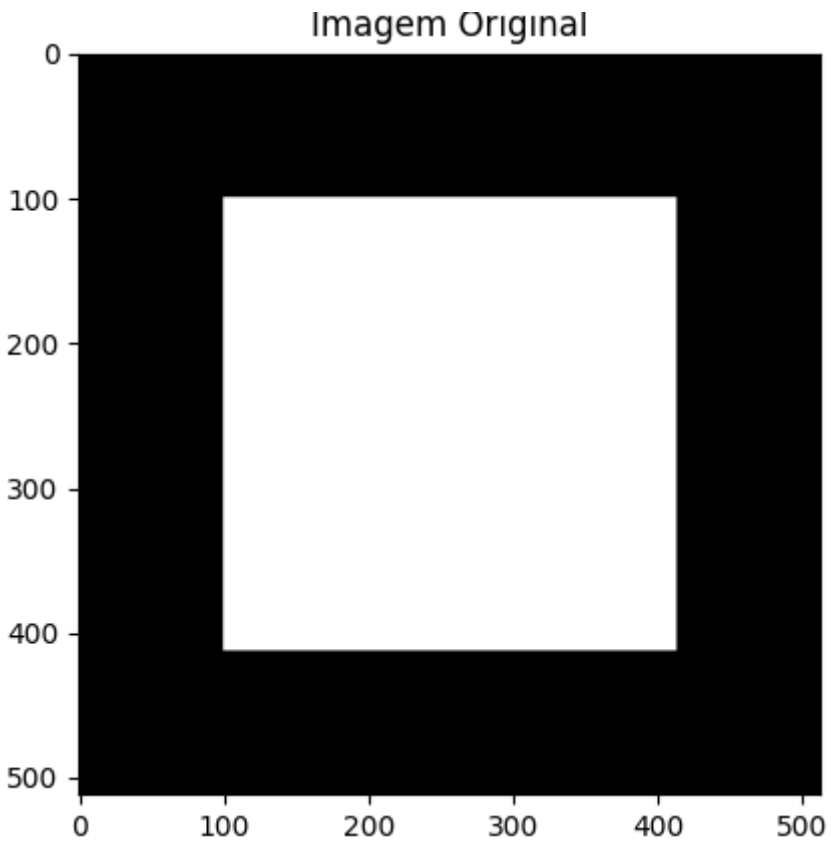
```
    # Converte a imagem para um array NumPy
    imagem_array = np.array(imagem)

    # Calcula a Transformada de Fourier 2D
    espectro_fourier = np.fft.fft2(imagem_array)
    espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)  # Centraliza o espectro

    # Calcula a magnitude do espectro centralizado
    amplitude = np.abs(espectro_fourier_centralizado)

    # Exibe o espectro de amplitude centralizado
    plt.subplot(121)
    plt.imshow(np.log(amplitude + 1), cmap='gray')
    plt.title('Espectro de Amplitude Centralizado (log)')

    # Calcula a fase do espectro centralizado
    fase = np.angle(espectro_fourier_centralizado)

    # Exibe o espectro de fase centralizado
    plt.subplot(122)
    plt.imshow(fase, cmap='gray')
    plt.title('Espectro de Fase Centralizado')
    plt.show()

def angulo40():
    # Aplica uma rotação de 40 graus na imagem
    imagem_rotacionada = imagem.rotate(40,
resample=Image.BICUBIC, center=(largura / 2, altura / 2))

    # Exibe a imagem rotacionada
    plt.subplot(141)
    plt.imshow(imagem_rotacionada, cmap='gray')
    plt.title('Imagem Rotacionada (40 graus)')

    # Converte a imagem rotacionada para um array NumPy
    imagem_array_rotacionada = np.array(imagem_rotacionada)

    # Calcula a Transformada de Fourier 2D da imagem
rotacionada
    espectro_fourier_rotacionado =
np.fft.fft2(imagem_array_rotacionada)
```

```
    # Calcula a amplitude do espectro de Fourier rotacionado
    amplitude_rotacionado =
np.abs(espectro_fourier_rotacionado)

    # Calcula a fase do espectro centralizado
    fase = np.angle(espectro_fourier_rotacionado)

    # Centraliza o espectro
    espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier_rotacionado)

    # Calcula a magnitude do espectro centralizado
    amplitude = np.abs(espectro_fourier_centralizado)

    # Exibe o espectro de amplitude do Fourier rotacionado
(não centralizado)
    plt.subplot(142)
    plt.imshow(np.log(amplitude_rotacionado + 1),
cmap='gray')
    plt.title('Espectro de Amplitude (Rotacionado, log)')
    plt.subplot(143)
    plt.imshow(fase, cmap='gray')
    plt.title('Espectro de Fase')
    plt.subplot(144)
    plt.imshow(np.log(amplitude + 1), cmap='gray')
    plt.title('Espectro de Fourier Centralizado')
    plt.show()

# Cria uma nova imagem preta de 512x512 pixels
largura, altura = 512, 512
imagem = Image.new('L', (largura, altura), 'black')

# Cria um objeto para desenhar na imagem
desenho = ImageDraw.Draw(imagem)

# Define as coordenadas do quadrado branco
x1, y1 = 100, 100  # Canto superior esquerdo
x2, y2 = 412, 412  # Canto inferior direito

# Desenha um quadrado branco na imagem
desenho.rectangle([x1, y1, x2, y2], fill='white')
```

```python
# Exibe a imagem original
plt.figure(figsize=(12, 6))
plt.subplot(131)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

if __name__ == "__main__":
    fourier()
    fourierCentralizado()
    angulo40()
    zoom()
```

Imagem Original

A imagem original é uma imagem preta de 512x512 pixels com um quadrado branco no centro. Abaixo está a representação visual da imagem original:



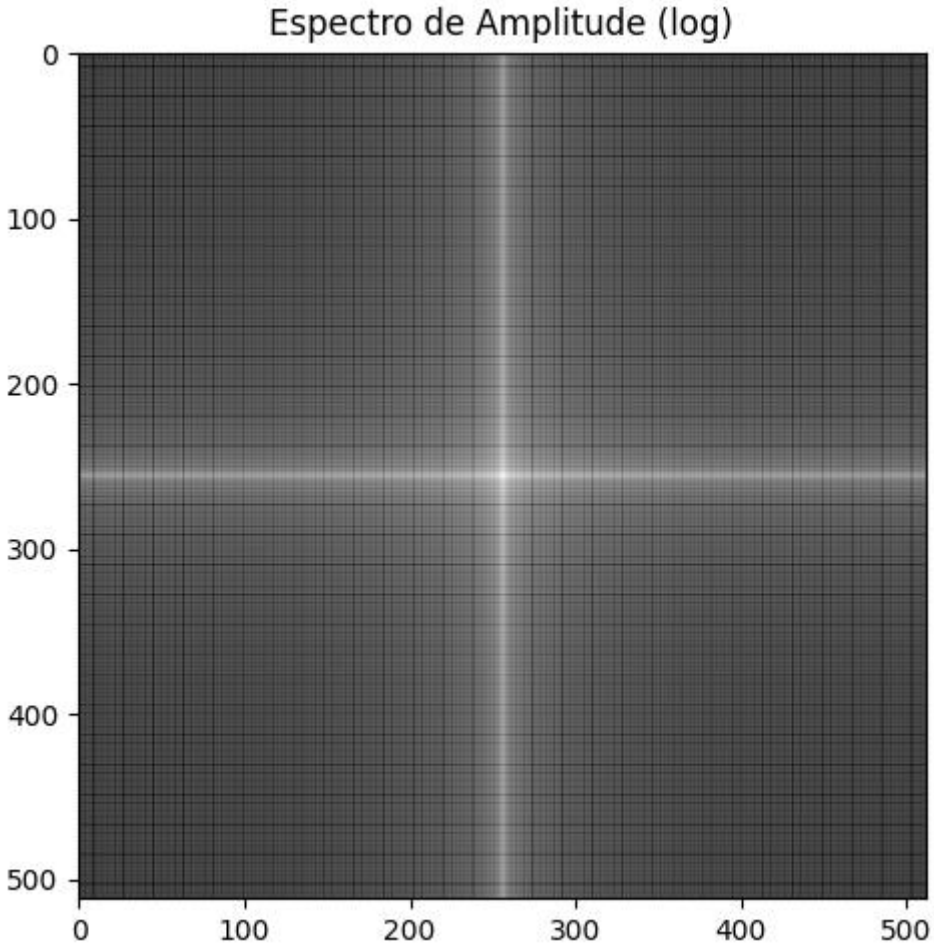
Transformada de Fourier 2D

A Transformada de Fourier 2D foi calculada para a imagem original. A Transformada de Fourier é uma técnica importante para análise de frequência em imagens. Abaixo estão

os resultados da Transformada de Fourier 2D da imagem original:

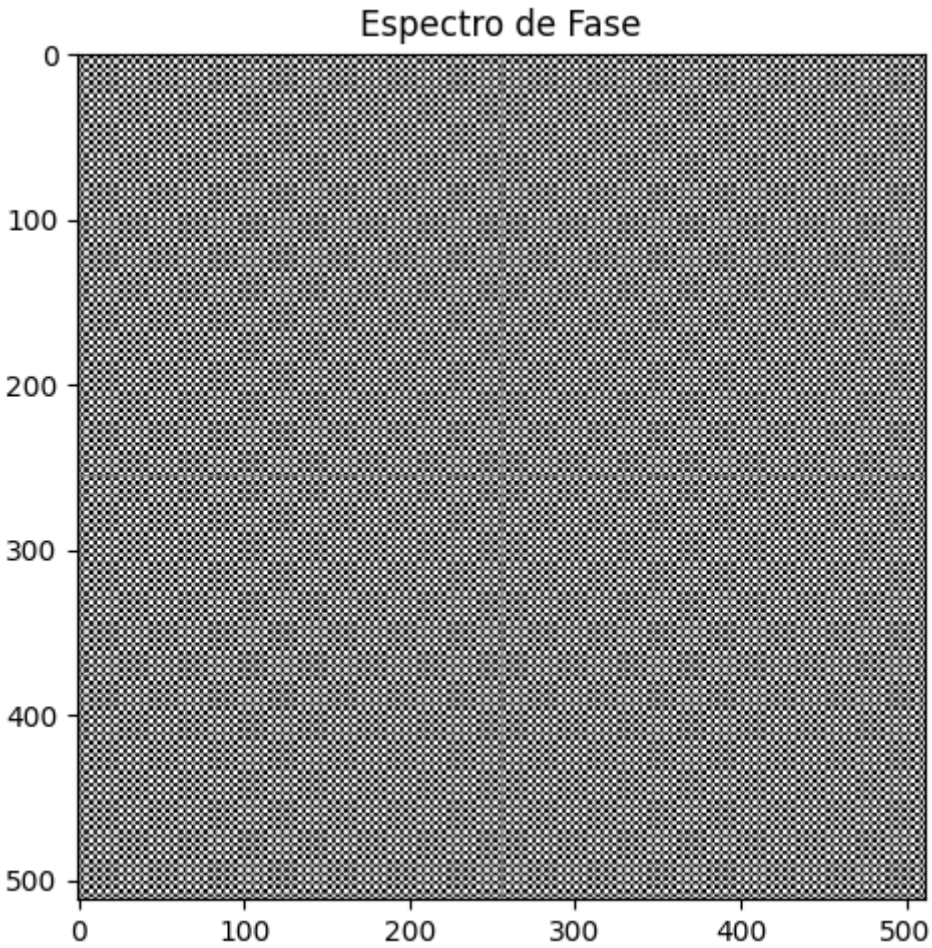
Espectro de Amplitude (log)

A imagem abaixo mostra o espectro de amplitude da Transformada de Fourier 2D da imagem original, em escala logarítmica:



Espectro de Fase

A imagem abaixo mostra o espectro de fase da Transformada de Fourier 2D da imagem original:

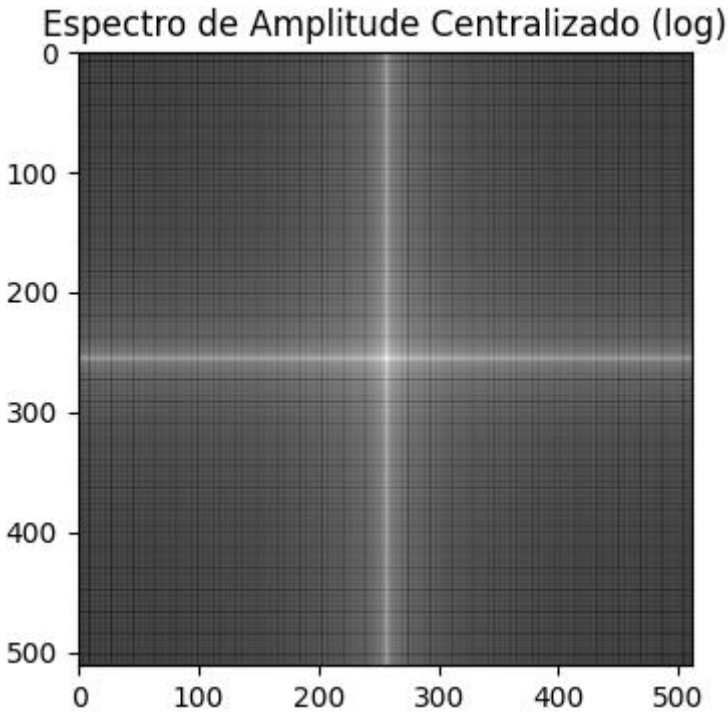


Transformada de Fourier 2D Centralizada

A Transformada de Fourier 2D centralizada foi calculada para a imagem original. A centralização do espectro facilita a análise das frequências. Abaixo estão os resultados da Transformada de Fourier 2D centralizada:

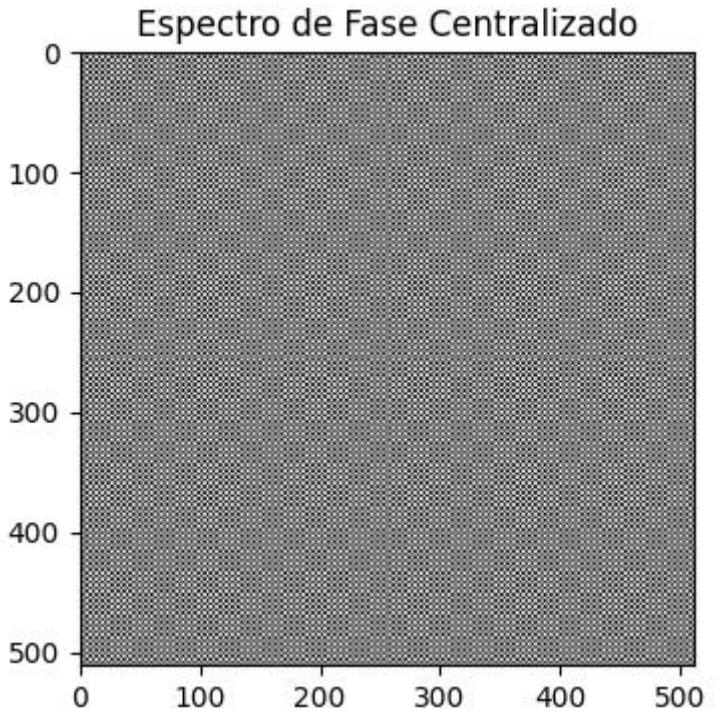
Espectro de Amplitude Centralizado (log)

A imagem abaixo mostra o espectro de amplitude centralizado da Transformada de Fourier 2D da imagem original, em escala logarítmica:



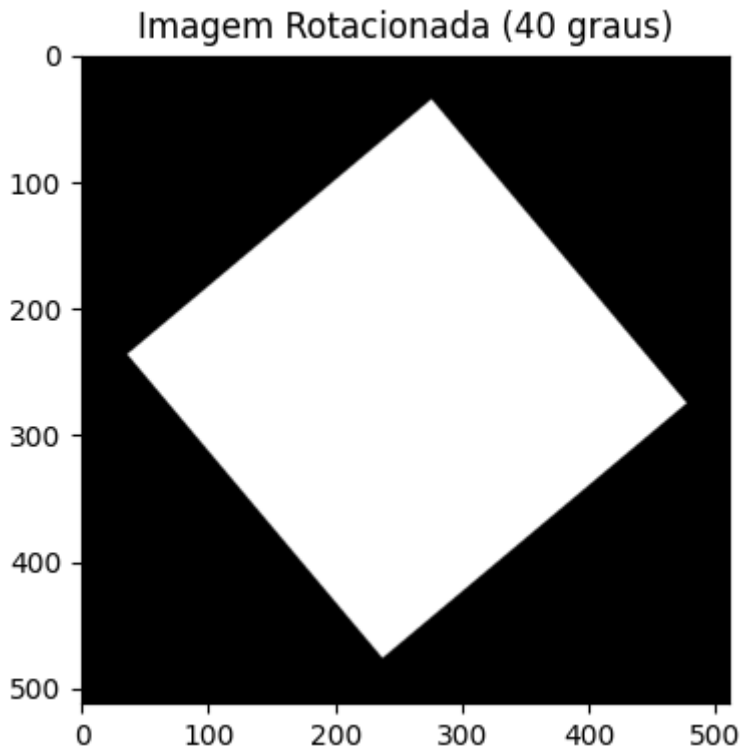
Espectro de Fase Centralizado

A imagem abaixo mostra o espectro de fase centralizado da Transformada de Fourier 2D da imagem original:

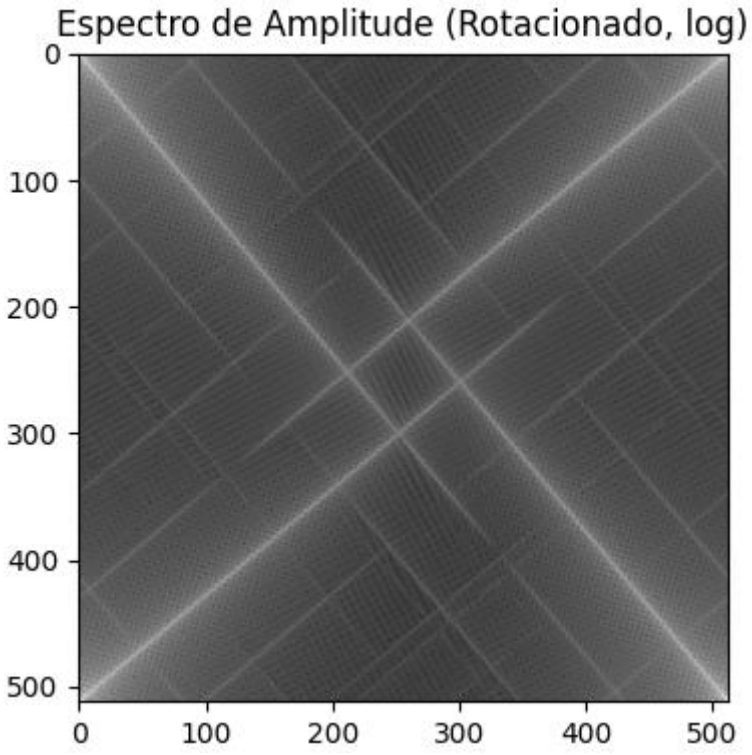


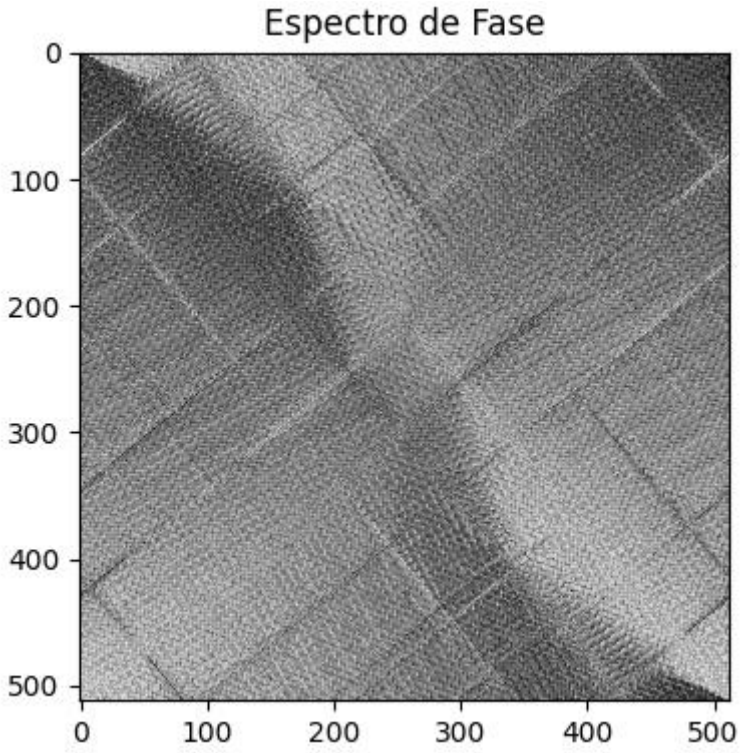
Rotação da Imagem em 40 Graus

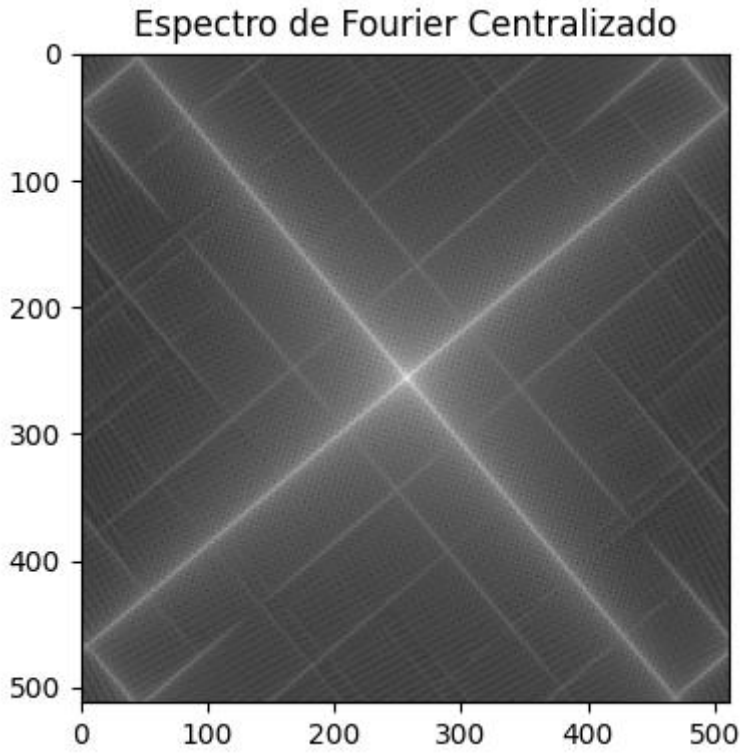
A imagem original foi rotacionada em 40 graus utilizando um método de interpolação bicúbica. Abaixo está a imagem resultante da rotação:



Além disso, a Transformada de Fourier 2D da imagem rotacionada foi calculada, e os espectros de amplitude e fase foram analisados, seguindo o mesmo procedimento descrito anteriormente. As imagens a seguir se referem aos resultados obtidos desses procedimentos:

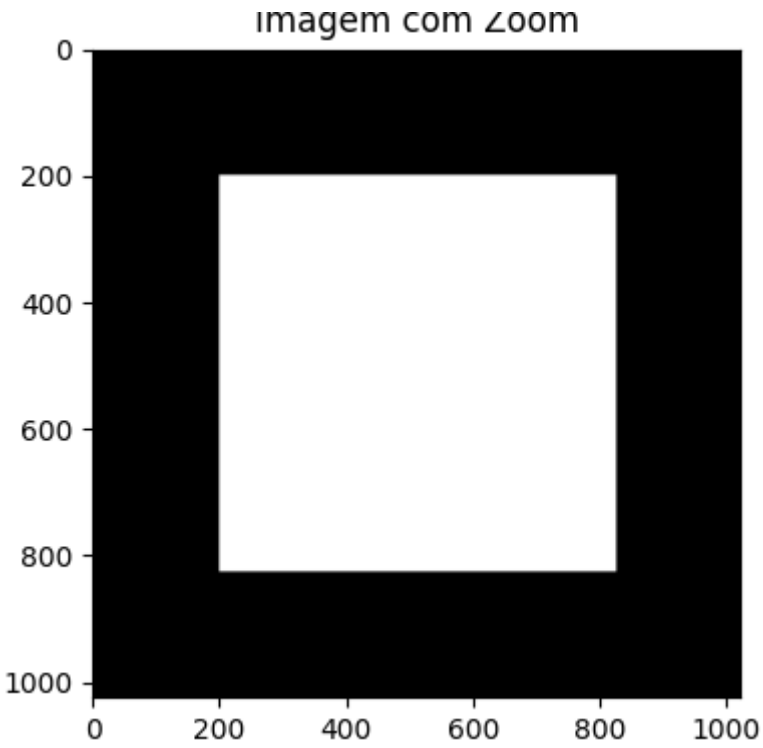




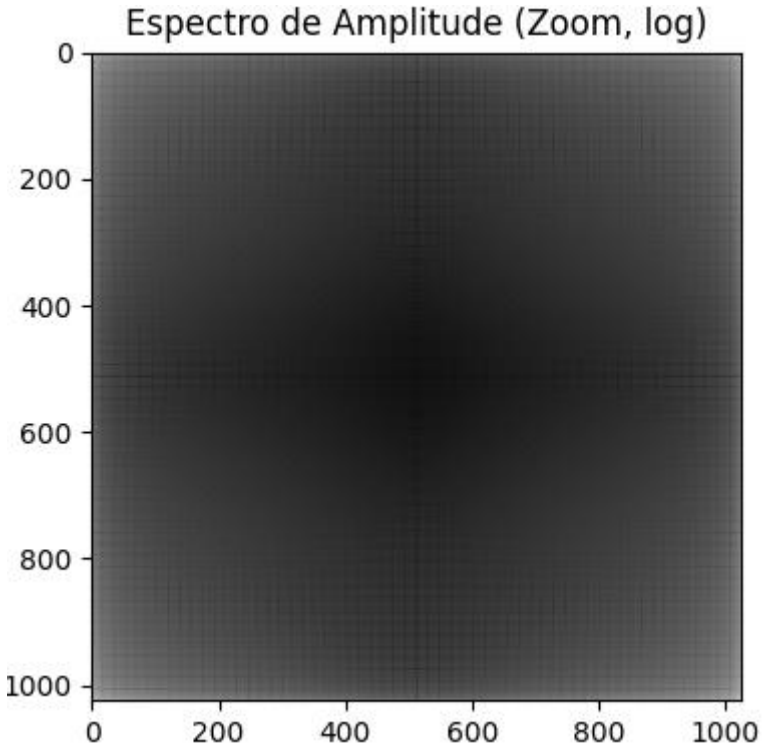


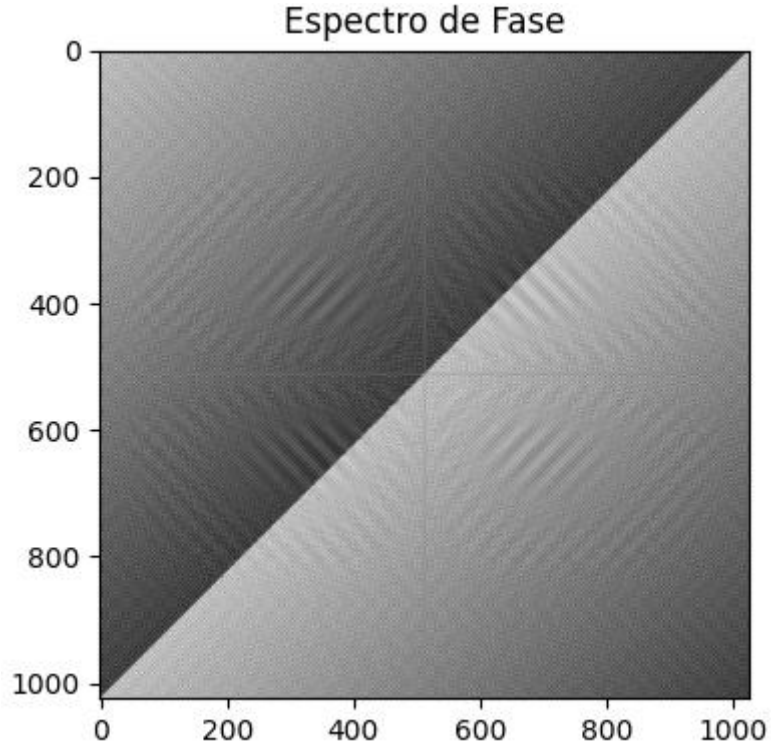
Aplicação de Zoom

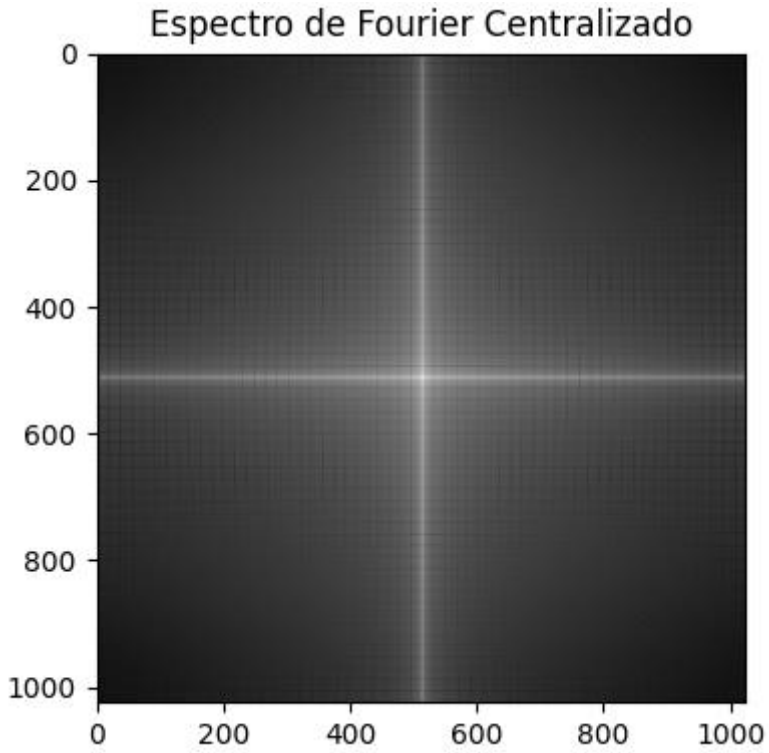
A imagem original foi ampliada por um fator de zoom de 2.0 utilizando o método de interpolação bicúbica. Abaixo está a imagem resultante do zoom:



A Transformada de Fourier 2D da imagem com zoom também foi calculada, e os espectros de amplitude e fase foram analisados, seguindo o mesmo procedimento descrito anteriormente. As imagens a seguir se referem aos resultados obtidos desses procedimentos:







Conclusão

Neste relatório, foram demonstradas várias operações de processamento digital de imagens aplicadas a uma imagem preta e branca de 512x512 pixels. As operações incluíram a Transformada de Fourier 2D, Transformada de Fourier 2D centralizada, rotação da imagem em 40 graus e aplicação de zoom. O uso dessas técnicas permite

analisar as propriedades de frequência das imagens e realizar operações de transformação úteis em processamento de imagens.

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

Código da implementação

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

# Carregar a imagem
imagem = cv2.imread('car.tif', cv2.IMREAD_GRAYSCALE)

# Calcular o espectro de Fourier da imagem
espectro_fourier = np.fft.fft2(imagem)
espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)

# Dimensões da imagem
altura, largura = imagem.shape

# Criar grades de frequência para a aplicação dos filtros
x, y = np.meshgrid(np.arange(-largura/2, largura/2),
np.arange(-altura/2, altura/2))
raio = np.sqrt(x**2 + y**2)

# Definir frequência de corte para os filtros
frequencia_corte = 40

# Filtro passa-baixa ideal
filtro_ideal = (raio <= frequencia_corte).astype(float)

# Filtro Butterworth
ordem_butterworth = 3
filtro_butterworth = 1 / (1 + (raio / frequencia_corte)**(2
* ordem_butterworth))

# Filtro gaussiano
sigma = 40
filtro_gaussiano = np.exp(-raio**2 / (2 * (sigma**2)))
```

```
# Aplicar os filtros ao espectro de Fourier
espectro_filtrado_ideal = espectro_fourier_centralizado *
filtro_ideal
espectro_filtrado_butterworth =
espectro_fourier_centralizado * filtro_butterworth
espectro_filtrado_gaussiano = espectro_fourier_centralizado
* filtro_gaussiano

# Transformada inversa de Fourier para obter as imagens
filtradas
imagem_filtrada_ideal =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_ideal
)))
imagem_filtrada_butterworth =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_butte
rworth)))
imagem_filtrada_gaussiano =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_gauss
iano)))

# Exibir as imagens
plt.figure(figsize=(12, 12))

# Imagem original
plt.subplot(3, 4, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

# Espectro de Fourier
plt.subplot(3, 4, 2)
plt.imshow(np.log(np.abs(espectro_fourier_centralizado) +
1), cmap='gray')
plt.title('Espectro de Fourier')

# Filtro Passa-Baixa Ideal
plt.subplot(3, 4, 3)
plt.imshow(filtro_ideal, cmap='gray')
plt.title('Filtro Passa-Baixa Ideal')

# Imagem Resultante após Filtro Ideal
plt.subplot(3, 4, 4)
```

```
plt.imshow(imagem_filtrada_ideal, cmap='gray')
plt.title('Imagem após Filtro Ideal')

# Filtro Butterworth
plt.subplot(3, 4, 7)
plt.imshow(filtro_butterworth, cmap='gray')
plt.title('Filtro Butterworth')

# Imagem Resultante após Filtro Butterworth
plt.subplot(3, 4, 8)
plt.imshow(imagem_filtrada_butterworth, cmap='gray')
plt.title('Imagem após Filtro Butterworth')

# Filtro Gaussiano
plt.subplot(3, 4, 11)
plt.imshow(filtro_gaussiano, cmap='gray')
plt.title('Filtro Gaussiano')

# Imagem Resultante após Filtro Gaussiano
plt.subplot(3, 4, 12)
plt.imshow(imagem_filtrada_gaussiano, cmap='gray')
plt.title('Imagem após Filtro Gaussiano')

plt.tight_layout()
plt.show()
```

Carregamento da Imagem

A imagem 'car.tif' foi carregada em tons de cinza (grayscale) usando a biblioteca OpenCV (cv2). A imagem resultante foi armazenada na variável imagem.

Cálculo do Espectro de Fourier

O espectro de Fourier 2D da imagem foi calculado utilizando a função np.fft.fft2(). O resultado foi armazenado na variável espectro_fourier.

Centralização do Espectro de Fourier

Para facilitar a análise, o espectro de Fourier calculado foi centralizado utilizando a função np.fft.fftshift(). O resultado foi armazenado na variável espectro_fourier_centralizado.

Dimensões da Imagem

As dimensões da imagem original (altura e largura) foram obtidas utilizando a função .shape e armazenadas nas variáveis altura e largura.

Filtros Aplicados

Filtro Passa-Baixa Ideal

Um filtro passa-baixa ideal foi criado para atenuar frequências acima de uma determinada frequência de corte. Neste caso, a frequência de corte foi definida como 40. O resultado foi armazenado na variável filtro_ideal.

Filtro Butterworth

Um filtro Butterworth foi implementado para realizar uma atenuação gradual das frequências acima da frequência de corte. A ordem do filtro é ajustável; no exemplo, foi definida como 3. O resultado foi armazenado na variável filtro_butterworth.

Filtro Gaussiano

Um filtro gaussiano foi criado para atenuar as frequências com base na distância da frequência de corte. O desvio padrão (sigma) do filtro gaussiano foi definido como 40. O resultado foi armazenado na variável filtro_gaussiano.

Aplicação dos Filtros

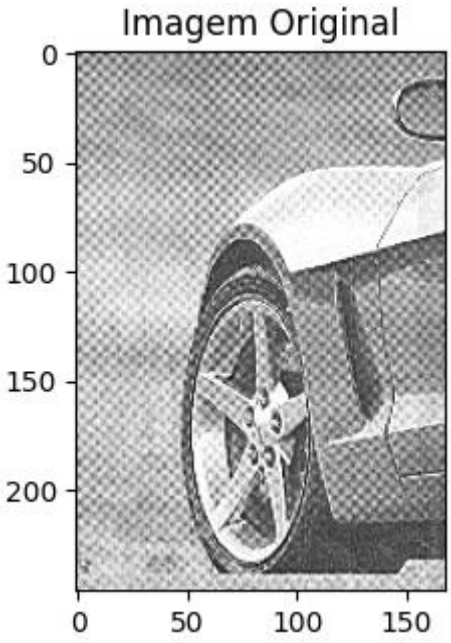
Os filtros passa-baixa ideal, Butterworth e gaussiano foram aplicados ao espectro de Fourier centralizado multiplicando-os ponto a ponto. Os resultados foram armazenados nas variáveis espectro_filtrado_ideal, espectro_filtrado_butterworth e espectro_filtrado_gaussiano.

Transformada Inversa de Fourier

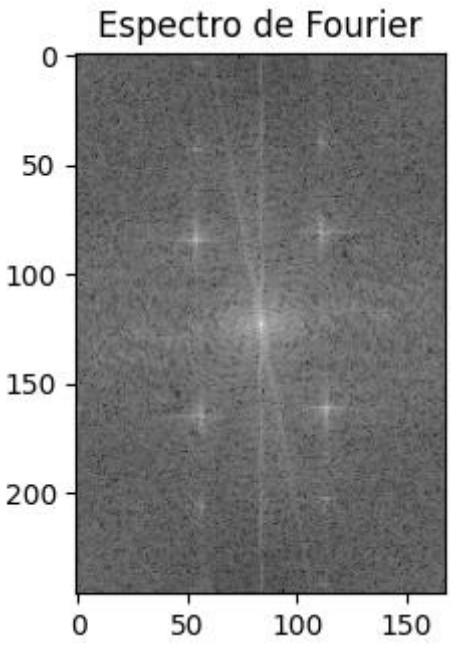
A transformada inversa de Fourier foi aplicada aos espectros filtrados para obter as imagens filtradas. O valor absoluto (np.abs()) foi utilizado para evitar valores complexos. As imagens filtradas resultantes foram armazenadas nas variáveis imagem_filtrada_ideal, imagem_filtrada_butterworth e imagem_filtrada_gaussiano.

Resultados e Visualização

Imagem Original



Espectro de Fourier



Filtro Passa-Baixa Ideal

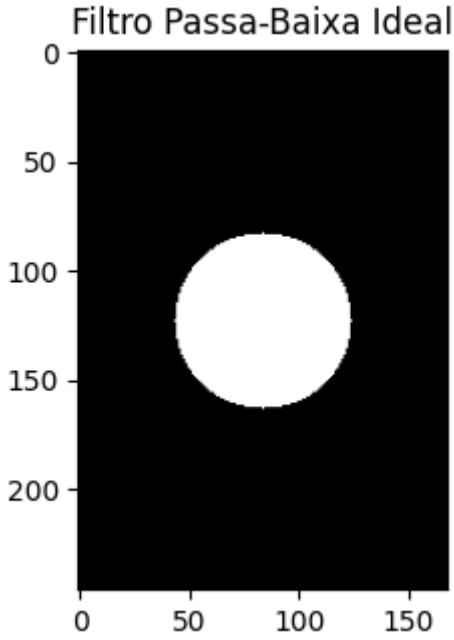
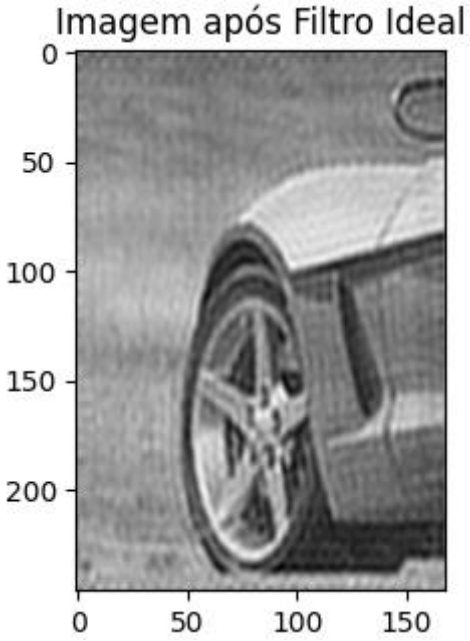


Imagem Resultante após Filtro Ideal



Filtro Butterworth

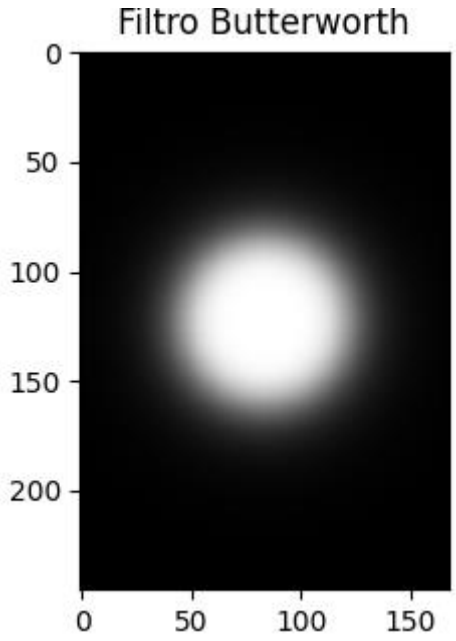
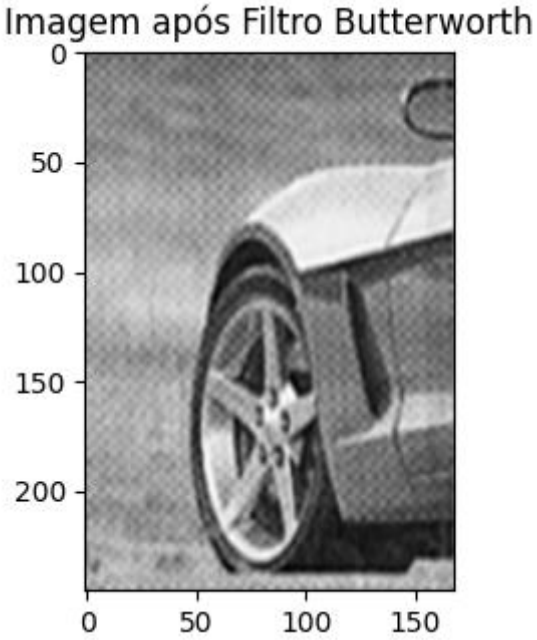


Imagem Resultante após Filtro Butterworth



Filtro Gaussiano

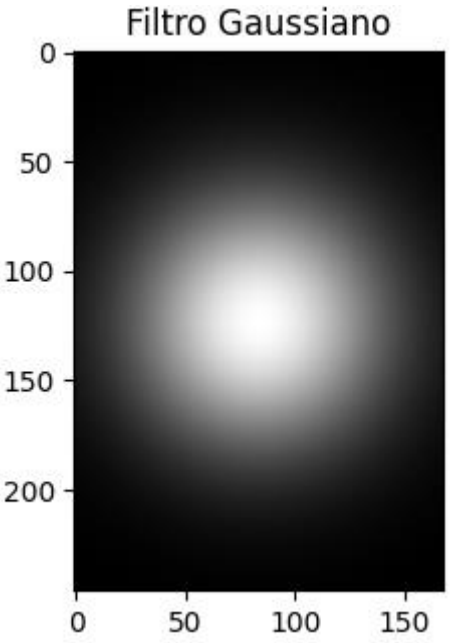
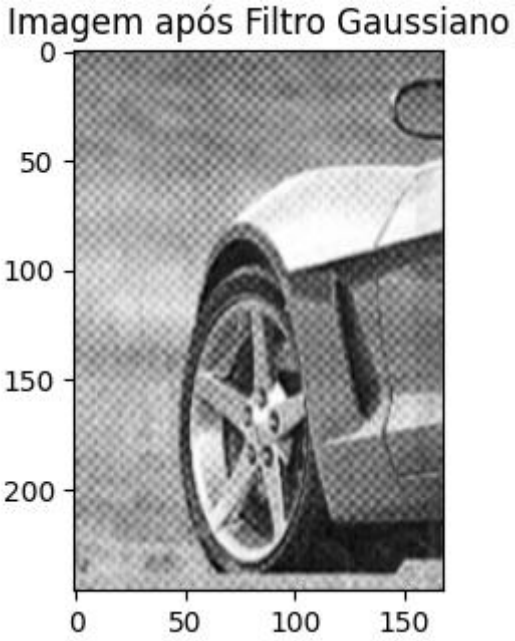


Imagem Resultante após Filtro Gaussiano



3. **Crie um filtro passa-alta do tipo ideal, butterworth e gaussiano e aplique-o às imagens disponibilizadas.**

Código da implementação

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

# Carregar a imagem
```

```
imagem = cv2.imread('car.tif', cv2.IMREAD_GRAYSCALE)

# Calcular o espectro de Fourier da imagem
espectro_fourier = np.fft.fft2(imagem)
espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)

# Dimensões da imagem
altura, largura = imagem.shape

# Criar grades de frequência para a aplicação dos filtros
x, y = np.meshgrid(np.arange(-largura/2, largura/2),
np.arange(-altura/2, altura/2))
raio = np.sqrt(x**2 + y**2)

# Definir frequência de corte para os filtros
frequencia_corte = 40

# Filtro passa-alta ideal
filtro_passa_alta_ideal = (raio >
frequencia_corte).astype(float)

# Filtro passa-alta Butterworth
ordem_butterworth = 3
filtro_passa_alta_butterworth = 1 - 1 / (1 + (raio /
frequencia_corte)**(2 * ordem_butterworth))

# Filtro passa-alta gaussiano
sigma = 40
filtro_passa_alta_gaussiano = 1 - np.exp(-raio**2 / (2 *
(sigma**2)))

# Aplicar os filtros passa-alta ao espectro de Fourier
espectro_filtrado_passa_alta_ideal =
espectro_fourier_centralizado * filtro_passa_alta_ideal
espectro_filtrado_passa_alta_butterworth =
espectro_fourier_centralizado *
filtro_passa_alta_butterworth
espectro_filtrado_passa_alta_gaussiano =
espectro_fourier_centralizado * filtro_passa_alta_gaussiano
```

```
# Transformada inversa de Fourier para obter as imagens filtradas
imagem_filtrada_passa_alta_ideal = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_alta_ideal)))
imagem_filtrada_passa_alta_butterworth = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_alta_butterworth)))
imagem_filtrada_passa_alta_gaussiano = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_alta_gaussiano)))

# Exibir as imagens
plt.figure(figsize=(15, 12))

# Imagem original
plt.subplot(3, 4, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

# Espectro de Fourier
plt.subplot(3, 4, 2)
plt.imshow(np.log(np.abs(espectro_fourier_centralizado) + 1), cmap='gray')
plt.title('Espectro de Fourier')

# Filtro Passa-Alta Ideal
plt.subplot(3, 4, 3)
plt.imshow(filtro_passa_alta_ideal, cmap='gray')
plt.title('Filtro Passa-Alta Ideal')

# Imagem Resultante após Filtro Passa-Alta Ideal
plt.subplot(3, 4, 4)
plt.imshow(imagem_filtrada_passa_alta_ideal, cmap='gray')
plt.title('Imagem após Filtro Passa-Alta Ideal')

# Filtro Passa-Alta Butterworth
plt.subplot(3, 4, 7)
plt.imshow(filtro_passa_alta_butterworth, cmap='gray')
plt.title('Filtro Passa-Alta Butterworth')

# Imagem Resultante após Filtro Passa-Alta Butterworth
```

```
plt.subplot(3, 4, 8)
plt.imshow(imagem_filtrada_passa_alta_butterworth,
cmap='gray')
plt.title('Imagem após Filtro Passa-Alta Butterworth')

# Filtro Passa-Alta Gaussiano
plt.subplot(3, 4, 11)
plt.imshow(filtro_passa_alta_gaussiano, cmap='gray')
plt.title('Filtro Passa-Alta Gaussiano')

# Imagem Resultante após Filtro Passa-Alta Gaussiano
plt.subplot(3, 4, 12)
plt.imshow(imagem_filtrada_passa_alta_gaussiano,
cmap='gray')
plt.title('Imagem após Filtro Passa-Alta Gaussiano')

plt.tight_layout()
plt.show()
```

Carregamento da Imagem

A imagem 'car.tif' foi carregada em tons de cinza (grayscale) utilizando a biblioteca OpenCV (cv2) e armazenada na variável imagem.

Cálculo do Espectro de Fourier

O código calculou o espectro de Fourier 2D da imagem utilizando a função np.fft.fft2(). O resultado foi armazenado na variável espectro_fourier.

Centralização do Espectro de Fourier

Para facilitar a análise, o espectro de Fourier calculado foi centralizado utilizando a função np.fft.fftshift(). O resultado foi armazenado na variável espectro_fourier_centralizado.

Dimensões da Imagem

As dimensões da imagem original (altura e largura) foram obtidas utilizando a função .shape e armazenadas nas variáveis altura e largura.

Filtros Passa-Alta

Filtro Passa-Alta Ideal

Um filtro passa-alta ideal foi criado para atenuar as frequências abaixo de uma frequência de corte especificada. Neste caso, a frequência de corte foi definida como 40. O resultado foi armazenado na variável filtro_passa_alta_ideal.

Filtro Passa-Alta Butterworth

Um filtro passa-alta Butterworth foi implementado para realizar uma atenuação gradual das frequências abaixo da frequência de corte. A ordem do filtro é ajustável, e no exemplo, foi definida como 3. O resultado foi armazenado na variável filtro_passa_alta_butterworth.

Filtro Passa-Alta Gaussiano

Um filtro passa-alta gaussiano foi criado para atenuar as frequências com base na distância da frequência de corte. O desvio padrão (sigma) do filtro gaussiano foi definido como 40. O resultado foi armazenado na variável filtro_passa_alta_gaussiano.

Aplicação dos Filtros Passa-Alta

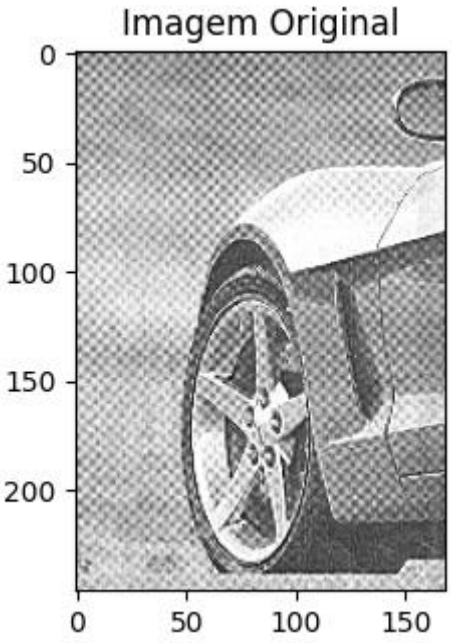
Os filtros passa-alta ideal, Butterworth e gaussiano foram aplicados ao espectro de Fourier centralizado multiplicando-os ponto a ponto. Os resultados foram armazenados nas variáveis espectro_filtrado_passa_alta_ideal, espectro_filtrado_passa_alta_butterworth e espectro_filtrado_passa_alta_gaussiano.

Transformada Inversa de Fourier

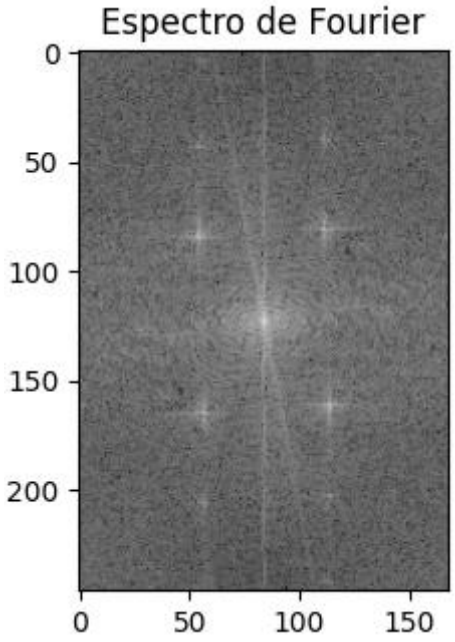
A transformada inversa de Fourier foi aplicada aos espectros filtrados para obter as imagens filtradas. O valor absoluto (np.abs()) foi utilizado para evitar valores complexos. As imagens filtradas resultantes foram armazenadas nas variáveis imagem_filtrada_passa_alta_ideal, imagem_filtrada_passa_alta_butterworth e imagem_filtrada_passa_alta_gaussiano.

Resultados e Visualização

Imagem Original



Espectro de Fourier



Filtro Passa-Alta Ideal

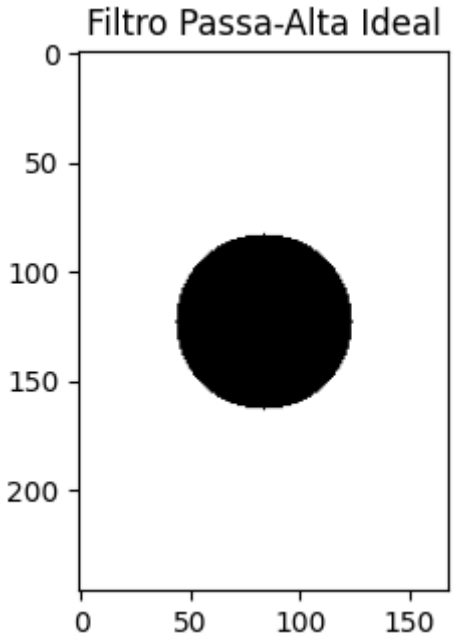
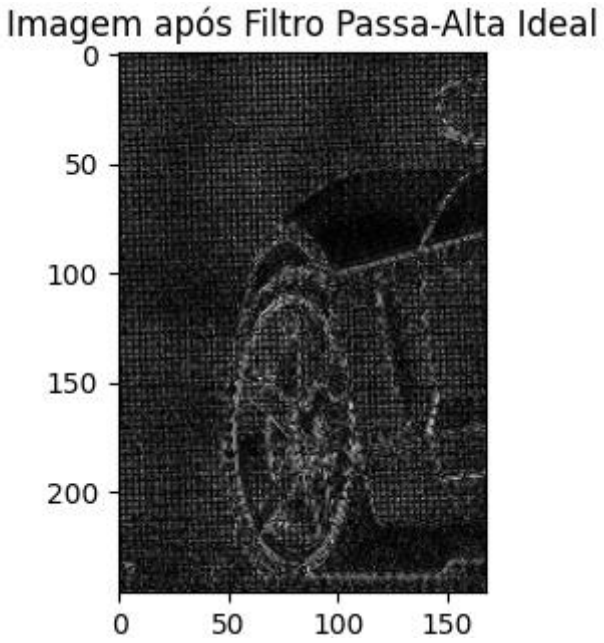


Imagem Resultante após Filtro Passa-Alta Ideal



Filtro Passa-Alta Butterworth

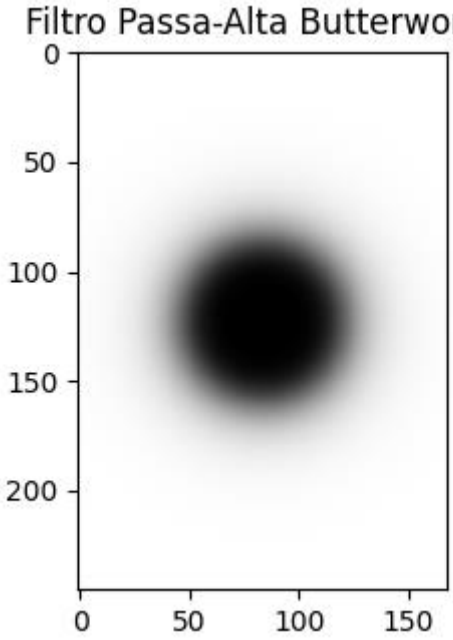
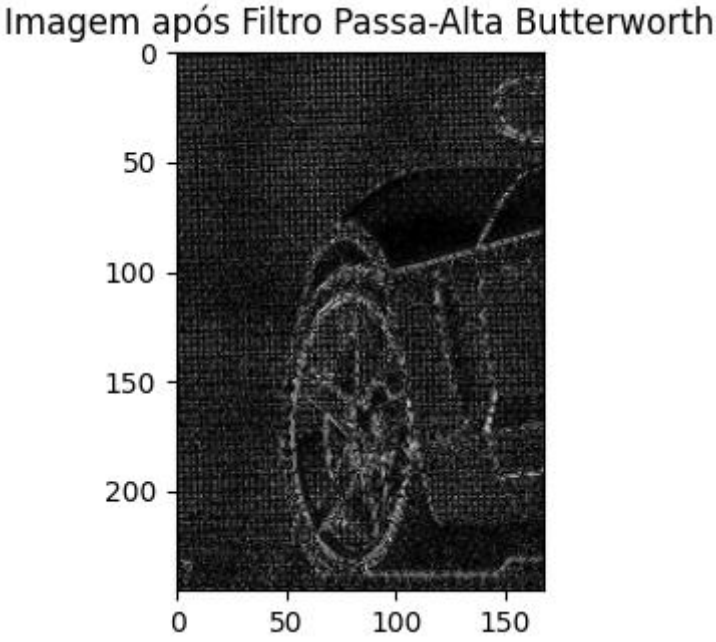


Imagem Resultante após Filtro Passa-Alta Butterworth



Filtro Passa-Alta Gaussiano

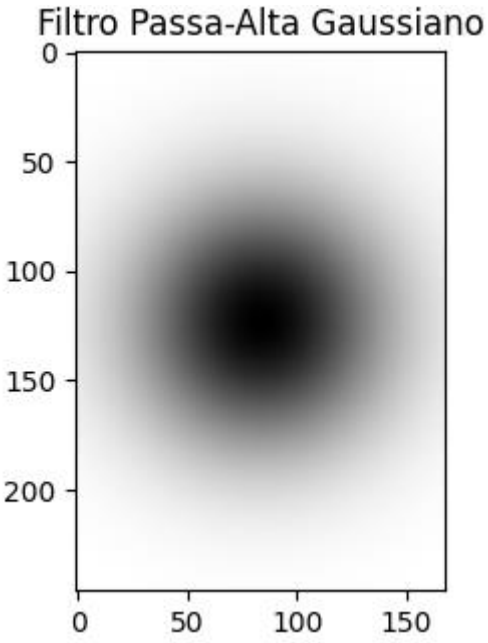
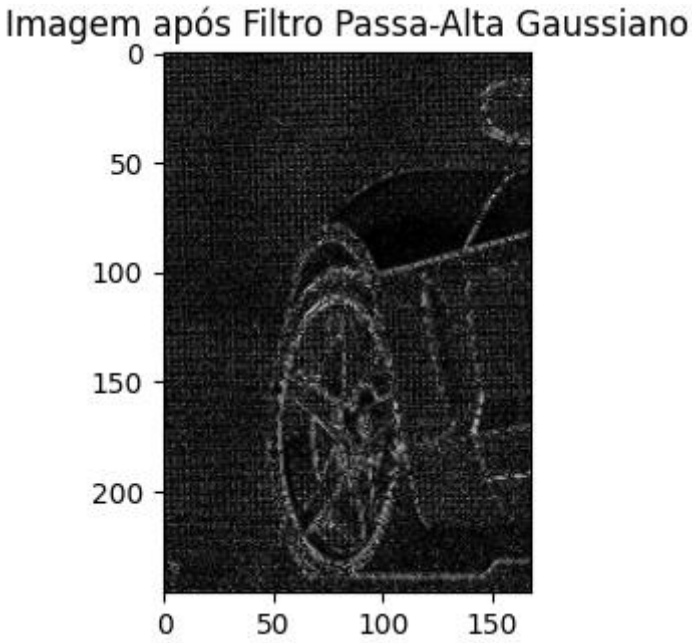


Imagem Resultante após Filtro Passa-Alta Gaussiano



4. **Varie o parâmetro de frequência de corte no filtro passa-baixa**

Código da implementação

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

# Carregar a imagem
imagem = cv2.imread('car.tif', cv2.IMREAD_GRAYSCALE)
```

```
# Calcular o espectro de Fourier da imagem
espectro_fourier = np.fft.fft2(imagem)
espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)

# Dimensões da imagem
altura, largura = imagem.shape

# Criar grades de frequência para a aplicação dos filtros
x, y = np.meshgrid(np.arange(-largura/2, largura/2),
np.arange(-altura/2, altura/2))
raio = np.sqrt(x**2 + y**2)

# Definir valores de D0 para os filtros passa-baixa
valores_d0 = [0.01, 0.05, 0.5]

plt.figure(figsize=(15, 12))

# Loop sobre os diferentes valores de D0
for i, d0 in enumerate(valores_d0):
    # Filtro passa-baixa ideal com o valor de D0 atual
    filtro_passa_baixa_ideal = (raio <= d0).astype(float)

    # Filtro passa-baixa Butterworth com o valor de D0 atual
    ordem_butterworth = 3
    filtro_passa_baixa_butterworth = 1 / (1 + (raio /
d0)**(2 * ordem_butterworth))

    # Filtro passa-baixa Gaussiano com o valor de D0 atual
    filtro_passa_baixa_gaussiano = np.exp(-raio**2 / (2 *
(d0**2)))

    # Aplicar os filtros passa-baixa ao espectro de Fourier
    espectro_filtrado_passa_baixa_ideal =
espectro_fourier_centralizado * filtro_passa_baixa_ideal
    espectro_filtrado_passa_baixa_butterworth =
espectro_fourier_centralizado *
filtro_passa_baixa_butterworth
    espectro_filtrado_passa_baixa_gaussiano =
espectro_fourier_centralizado * filtro_passa_baixa_gaussiano
```

```
    # Transformada inversa de Fourier para obter as imagens filtradas
    imagem_filtrada_passa_baixa_ideal = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_baixa_ideal)))
    imagem_filtrada_passa_baixa_butterworth = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_baixa_butterworth)))
    imagem_filtrada_passa_baixa_gaussiano = np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa_baixa_gaussiano)))

    # Exibir as imagens dos filtros e as imagens resultantes
    plt.subplot(4, 6, i * 6 + 1)
    plt.imshow(filtro_passa_baixa_ideal, cmap='gray')
    plt.title(f'Filtro Ideal (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 2)
    plt.imshow(imagem_filtrada_passa_baixa_ideal, cmap='gray')
    plt.title(f'Imagem após Filtro Ideal (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 3)
    plt.imshow(filtro_passa_baixa_butterworth, cmap='gray')
    plt.title(f'Filtro Butterworth (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 4)
    plt.imshow(imagem_filtrada_passa_baixa_butterworth, cmap='gray')
    plt.title(f'Imagem após Filtro Butterworth (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 5)
    plt.imshow(filtro_passa_baixa_gaussiano, cmap='gray')
    plt.title(f'Filtro Gaussiano (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 6)
    plt.imshow(imagem_filtrada_passa_baixa_gaussiano, cmap='gray')
    plt.title(f'Imagem após Filtro Gaussiano (D0 = {d0})')

# Imagem original
plt.subplot(4, 6, 19)
```

```
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

# Espectro de Fourier
plt.subplot(4, 6, 20)
plt.imshow(np.log(np.abs(espectro_fourier_centralizado) +
1), cmap='gray')
plt.title('Espectro de Fourier')

plt.tight_layout()
plt.show()
```

Carregamento da Imagem

A imagem 'car.tif' foi carregada em tons de cinza (grayscale) utilizando a biblioteca OpenCV (cv2) e armazenada na variável imagem.

Cálculo do Espectro de Fourier

O código calculou o espectro de Fourier 2D da imagem utilizando a função np.fft.fft2(). O resultado foi armazenado na variável espectro_fourier.

Centralização do Espectro de Fourier

Para facilitar a análise, o espectro de Fourier calculado foi centralizado utilizando a função np.fft.fftshift(). O resultado foi armazenado na variável espectro_fourier_centralizado.

Dimensões da Imagem

As dimensões da imagem original (altura e largura) foram obtidas utilizando a função .shape e armazenadas nas variáveis altura e largura.

Filtros Passa-Baixa com Diferentes Valores de D0

O código executou um loop sobre diferentes valores de D0 (0.01, 0.05 e 0.5) para criar filtros passa-baixa com frequências de corte variáveis. Para cada valor de D0, os seguintes filtros foram criados:

Filtro Passa-Baixa Ideal

Um filtro passa-baixa ideal foi criado para atenuar frequências acima da frequência de

corte especificada (D0). O resultado foi armazenado na variável filtro_passa_baixa_ideal.

Filtro Passa-Baixa Butterworth

Um filtro passa-baixa Butterworth foi implementado para realizar uma atenuação gradual das frequências acima da frequência de corte (D0). A ordem do filtro é ajustável, e no exemplo, foi definida como 3. O resultado foi armazenado na variável filtro_passa_baixa_butterworth.

Filtro Passa-Baixa Gaussiano

Um filtro passa-baixa gaussiano foi criado para atenuar as frequências com base na distância da frequência de corte (D0). O resultado foi armazenado na variável filtro_passa_baixa_gaussiano.

Aplicação dos Filtros Passa-Baixa

Para cada valor de D0, os filtros passa-baixa foram aplicados ao espectro de Fourier centralizado multiplicando-os ponto a ponto. Os resultados foram armazenados nas variáveis espectro_filtrado_passa_baixa_ideal, espectro_filtrado_passa_baixa_butterworth e espectro_filtrado_passa_baixa_gaussiano.

Transformada Inversa de Fourier

A transformada inversa de Fourier foi aplicada aos espectros filtrados para obter as imagens filtradas. O valor absoluto (np.abs()) foi utilizado para evitar valores complexos. As imagens filtradas resultantes foram armazenadas nas variáveis imagem_filtrada_passa_baixa_ideal, imagem_filtrada_passa_baixa_butterworth e imagem_filtrada_passa_baixa_gaussiano.

Resultados e Visualização

O resultado do processamento foi visualizado em um conjunto de imagens, incluindo:

- Os filtros passa-baixa ideal, Butterworth e gaussiano para cada valor de D0.
- As imagens resultantes após a aplicação de cada filtro passa-baixa.

As imagens resultantes destacam diferentes características da imagem original, dependendo do valor de D0 escolhido. Isso demonstra como o ajuste da frequência de corte afeta a transformação de frequência em imagens.

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

Código da implementação

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

# Carregar a imagem
imagem = cv2.imread('car.tif', cv2.IMREAD_GRAYSCALE)

# Calcular o espectro de Fourier da imagem
espectro_fourier = np.fft.fft2(imagem)
espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)

# Dimensões da imagem
altura, largura = imagem.shape

# Criar grades de frequência para a aplicação dos filtros
x, y = np.meshgrid(np.arange(-largura/2, largura/2),
np.arange(-altura/2, altura/2))
raio = np.sqrt(x**2 + y**2)

# Definir valores de D0 para os filtros passa-alta
valores_d0 = [0.01, 0.05, 0.5]

plt.figure(figsize=(15, 12))

# Loop sobre os diferentes valores de D0
for i, d0 in enumerate(valores_d0):
    # Filtro passa-alta ideal com o valor de D0 atual
    filtro_passa_alta_ideal = (raio > d0).astype(float)

    # Filtro passa-alta Butterworth com o valor de D0 atual
    ordem_butterworth = 3
    filtro_passa_alta_butterworth = 1 / (1 + (d0 / raio)**(2
* ordem_butterworth))

    # Filtro passa-alta Gaussiano com o valor de D0 atual
```

```
    filtro_passa_alta_gaussiano = 1 - np.exp(-raio**2 / (2 *
(d0**2)))

    # Aplicar os filtros passa-alta ao espectro de Fourier
    espectro_filtrado_passa_alta_ideal =
espectro_fourier_centralizado * filtro_passa_alta_ideal
    espectro_filtrado_passa_alta_butterworth =
espectro_fourier_centralizado *
filtro_passa_alta_butterworth
    espectro_filtrado_passa_alta_gaussiano =
espectro_fourier_centralizado * filtro_passa_alta_gaussiano

    # Transformada inversa de Fourier para obter as imagens
filtradas
    imagem_filtrada_passa_alta_ideal =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa
_alta_ideal)))
    imagem_filtrada_passa_alta_butterworth =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa
_alta_butterworth)))
    imagem_filtrada_passa_alta_gaussiano =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa
_alta_gaussiano)))

    # Exibir as imagens dos filtros e as imagens resultantes
    plt.subplot(4, 6, i * 6 + 1)
    plt.imshow(filtro_passa_alta_ideal, cmap='gray')
    plt.title(f'Filtro Ideal (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 2)
    plt.imshow(imagem_filtrada_passa_alta_ideal,
cmap='gray')
    plt.title(f'Imagem após Filtro Ideal (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 3)
    plt.imshow(filtro_passa_alta_butterworth, cmap='gray')
    plt.title(f'Filtro Butterworth (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 4)
    plt.imshow(imagem_filtrada_passa_alta_butterworth,
cmap='gray')
    plt.title(f'Imagem após Filtro Butterworth (D0 = {d0})')
```

```
    plt.subplot(4, 6, i * 6 + 5)
    plt.imshow(filtro_passa_alta_gaussiano, cmap='gray')
    plt.title(f'Filtro Gaussiano (D0 = {d0})')

    plt.subplot(4, 6, i * 6 + 6)
    plt.imshow(imagem_filtrada_passa_alta_gaussiano,
cmap='gray')
    plt.title(f'Imagem após Filtro Gaussiano (D0 = {d0})')

# Imagem original
plt.subplot(4, 6, 19)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

# Espectro de Fourier
plt.subplot(4, 6, 20)
plt.imshow(np.log(np.abs(espectro_fourier_centralizado) +
1), cmap='gray')
plt.title('Espectro de Fourier')

plt.tight_layout()
plt.show()
```

Carregamento da Imagem

A imagem 'car.tif' foi carregada em tons de cinza (grayscale) utilizando a biblioteca OpenCV (cv2) e armazenada na variável imagem.

Cálculo do Espectro de Fourier

O código calculou o espectro de Fourier 2D da imagem utilizando a função np.fft.fft2(). O resultado foi armazenado na variável espectro_fourier.

Centralização do Espectro de Fourier

Para facilitar a análise, o espectro de Fourier calculado foi centralizado utilizando a função np.fft.fftshift(). O resultado foi armazenado na variável espectro_fourier_centralizado.

Dimensões da Imagem

As dimensões da imagem original (altura e largura) foram obtidas utilizando a função .shape e armazenadas nas variáveis altura e largura.

Filtros Passa-Alta com Diferentes Valores de D0

O código executou um loop sobre diferentes valores de D0 (0.01, 0.05 e 0.5) para criar filtros passa-alta com frequências de corte variáveis. Para cada valor de D0, os seguintes filtros foram criados:

Filtro Passa-Alta Ideal

Um filtro passa-alta ideal foi criado para atenuar frequências abaixo da frequência de corte especificada (D0). O resultado foi armazenado na variável filtro_passa_alta_ideal.

Filtro Passa-Alta Butterworth

Um filtro passa-alta Butterworth foi implementado para realizar uma atenuação gradual das frequências abaixo da frequência de corte (D0). A ordem do filtro é ajustável, e no exemplo, foi definida como 3. O resultado foi armazenado na variável filtro_passa_alta_butterworth.

Filtro Passa-Alta Gaussiano

Um filtro passa-alta gaussiano foi criado para atenuar as frequências com base na distância da frequência de corte (D0). O resultado foi armazenado na variável filtro_passa_alta_gaussiano.

Aplicação dos Filtros Passa-Alta

Para cada valor de D0, os filtros passa-alta foram aplicados ao espectro de Fourier centralizado multiplicando-os ponto a ponto. Os resultados foram armazenados nas variáveis espectro_filtrado_passa_alta_ideal, espectro_filtrado_passa_alta_butterworth e espectro_filtrado_passa_alta_gaussiano.

Transformada Inversa de Fourier

A transformada inversa de Fourier foi aplicada aos espectros filtrados para obter as imagens filtradas. O valor absoluto (np.abs()) foi utilizado para evitar valores complexos. As imagens filtradas resultantes foram armazenadas nas variáveis imagem_filtrada_passa_alta_ideal, imagem_filtrada_passa_alta_butterworth e

imagem_filtrada_passa_alta_gaussiano.

Resultados e Visualização

O resultado do processamento foi visualizado em um conjunto de imagens, incluindo:

- Os filtros passa-alta ideal, Butterworth e gaussiano para cada valor de D0.
- As imagens resultantes após a aplicação de cada filtro passa-alta.

As imagens resultantes destacam diferentes características da imagem original, dependendo do valor de D0 escolhido. Isso demonstra como o ajuste da frequência de corte afeta a transformação de frequência em imagens.

6. **Filtro passa-banda**

Código da implementação

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

# Carregar a imagem
imagem = cv2.imread('car.tif', cv2.IMREAD_GRAYSCALE)

# Calcular o espectro de Fourier da imagem
espectro_fourier = np.fft.fft2(imagem)
espectro_fourier_centralizado =
np.fft.fftshift(espectro_fourier)

# Dimensões da imagem
altura, largura = imagem.shape

# Criar grades de frequência para a aplicação dos filtros
x, y = np.meshgrid(np.arange(-largura/2, largura/2),
np.arange(-altura/2, altura/2))
raio = np.sqrt(x**2 + y**2)

# Definir as frequências de corte para o filtro passa-banda
frequencia_corte_inferior = 0.3
frequencia_corte_superior = 0.9

# Filtro passa-alta para realçar frequências acima da
frequência de corte inferior
```

```
filtro_passa_alta = (raio >
frequencia_corte_inferior).astype(float)

# Filtro passa-baixa para eliminar frequências acima da
frequência de corte superior
filtro_passa_baixa = (raio <=
frequencia_corte_superior).astype(float)

# Combinar os filtros passa-alta e passa-baixa para criar o
filtro passa-banda
filtro_passa_banda = filtro_passa_alta * filtro_passa_baixa

# Aplicar o filtro passa-banda ao espectro de Fourier
espectro_filtrado_passa_banda =
espectro_fourier_centralizado * filtro_passa_banda

# Transformada inversa de Fourier para obter a imagem
filtrada
imagem_filtrada_passa_banda =
np.abs(np.fft.ifft2(np.fft.ifftshift(espectro_filtrado_passa
_banda)))

# Exibir as imagens
plt.figure(figsize=(12, 8))

# Imagem original
plt.subplot(2, 3, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')

# Espectro de Fourier
plt.subplot(2, 3, 2)
plt.imshow(np.log(np.abs(espectro_fourier_centralizado) +
1), cmap='gray')
plt.title('Espectro de Fourier')

# Filtro Passa-Banda
plt.subplot(2, 3, 3)
plt.imshow(filtro_passa_banda, cmap='gray')
plt.title('Filtro Passa-Banda')

# Imagem Resultante após Filtro Passa-Banda
```

```
plt.subplot(2, 3, 4)
plt.imshow(imagem_filtrada_passa_banda, cmap='gray')
plt.title('Imagem após Filtro Passa-Banda')

plt.tight_layout()
plt.show()
```

Carregamento da Imagem

A imagem 'car.tif' foi carregada em tons de cinza (grayscale) utilizando a biblioteca OpenCV (cv2) e armazenada na variável imagem.

Cálculo do Espectro de Fourier

O código calculou o espectro de Fourier 2D da imagem utilizando a função np.fft.fft2(). O resultado foi armazenado na variável espectro_fourier.

Centralização do Espectro de Fourier

Para facilitar a análise, o espectro de Fourier calculado foi centralizado utilizando a função np.fft.fftshift(). O resultado foi armazenado na variável espectro_fourier_centralizado.

Dimensões da Imagem

As dimensões da imagem original (altura e largura) foram obtidas utilizando a função .shape e armazenadas nas variáveis altura e largura.

Criação do Filtro Passa-Banda

Para criar um filtro passa-banda, dois filtros foram combinados:

Filtro Passa-Alta

Um filtro passa-alta foi criado para realçar frequências acima de uma frequência de corte inferior (0.3 no exemplo). Isso foi realizado gerando um filtro que atenua as frequências abaixo da frequência de corte inferior e preserva as frequências mais altas.

Filtro Passa-Baixa

Um filtro passa-baixa foi criado para eliminar frequências acima de uma frequência de corte superior (0.9 no exemplo). Isso foi realizado gerando um filtro que atenua as

frequências acima da frequência de corte superior e preserva as frequências mais baixas.

Combinando os Filtros

Os filtros passa-alta e passa-baixa foram multiplicados ponto a ponto para criar o filtro passa-banda final. O resultado foi armazenado na variável filtro_passa_banda.

Aplicação do Filtro Passa-Banda

O filtro passa-banda foi aplicado ao espectro de Fourier centralizado multiplicando-o ponto a ponto. O resultado foi armazenado na variável espectro_filtrado_passa_banda.

Transformada Inversa de Fourier

A transformada inversa de Fourier foi aplicada ao espectro filtrado para obter a imagem filtrada. O valor absoluto (np.abs()) foi utilizado para evitar valores complexos. A imagem resultante foi armazenada na variável imagem_filtrada_passa_banda.

Resultados e Visualização

O resultado do processamento foi visualizado em um conjunto de imagens, incluindo:

- A imagem original.
- O espectro de Fourier da imagem.
- O filtro passa-banda criado.
- A imagem resultante após a aplicação do filtro passa-banda.

A imagem resultante após a aplicação do filtro passa-banda realça as frequências na faixa especificada (entre as frequências de corte inferior e superior), enquanto atenua as frequências fora dessa faixa. Isso permite que você destaque componentes específicos de frequência na imagem original.