Trabalho 02 de Introdução ao Processamento de Imagem Digital

Fabio Fogliarini Brolesi <brolesi@gmail.com> - RA 023718

10 de maio de 2021

Sumário

1 Descrição

A transformada discreta de Fourier (DFT[1]), dada por

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)} \tag{1}$$

E sua inversa (IDFT[2]):

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} F(u,v) e^{j2\pi(ux/M+vy/N)} \tag{2}$$

conforme [1].

Dentro do contexto de filtragem no domínio de frequência, os passos para o processo são os que seguem na figura 1.

Na figura 2 temos:

[1] do inglês *Discrete Fourier transform*
[2] do inglês *Inverse discrete Fourier transform*

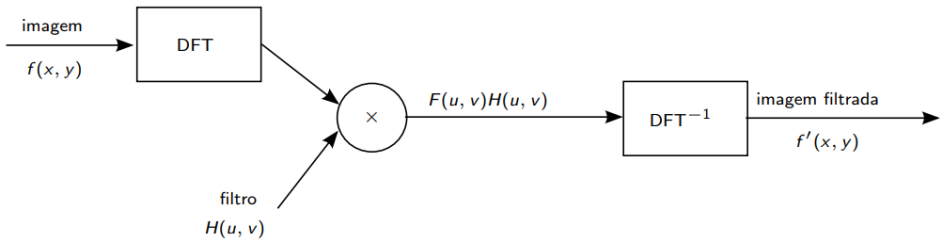

Figura 1: Passos para filtragem, conforme [2]

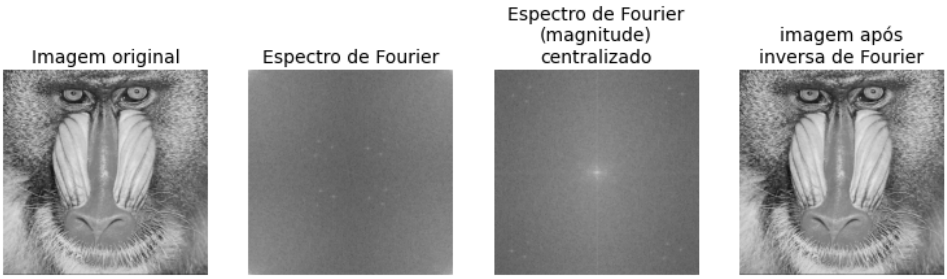

Figura 2: Imagem, espectros de Fourier e imagem filtrada

- a imagem original
- a imagem de entrada $f(x, y)$ transformada para o domínio de frequência por meio da transformada discreta de Fourier, conforme equação 1.
- a imagem no domínio de frequência é prepresentada por $F(u, v)$ e é convoluída com o filtro $H(u, v)$.
- a inversa da transformada de Fourier (conforme equação 2) é aplicada ao produto $F(u, v) \times H(u, v)$ para retornar ao domínio espacial onde se tem a imagem filtrada $f'(x, y)$.

Vamos tratar imagens monocromáticas, realizando transformações e utilizando filtros passa-alta, passa-baixa, passa-faixa, além da remoção a partir de um limiar e realizar uma rotação.

1.1 Filtros passa-baixa

Conforme [1], filtros passa-baixa ou filtros espaciais de suavização (também chamados de média) são usados para reduzir transições com intensidades mais acentuadas. O ruído sal e pimenta, por exemplo, se apresenta na forma de transições nítidas de intensidade. Filtros passa-baixa tem uma aplicação clara de suavização e redução de ruído. A suavização é usada para reduzir detalhes que não tem relevância em uma imagem, onde essa

falta de relevância se traduz em conjuntos de pixel que são pequenos em relação ao tamanho do kernel do filtro. Outra aplicação é para suavizar os contornos que advém de pequenos níveis de intensidade em uma imagem. Filtros passa-baixa são usados em combinação com outras técnicas para aprimoramento de imagem. Além de serem úteis em inúmeras aplicações de processamento de imagens, os filtros passa-baixa são fundamentais, no sentido de que outros filtros importantes, incluindo passa-alta, passa-faixa e rejeita-faixa podem ser derivados de filtros passa-baixa.

Para a atividade, inicialmente identificamos as dimensões da imagem (`height` e `width` para altura e largura, respectivamente), e criamos um retângulo com as mesmas dimensões que será uma máscara com 0 e depois com a bilbioteca `opencv` criamos um círculo na máscara, no centro da imagem, sem borda e preenchido com o valor 1 utilizando o método `circle`. Em seguida, multiplicamos a máscara (que contém valores 0 e 1) pela imagem, habilitando os pixels onde há o valor 1, mais próximos do centro e até a distância do raio `radius`, e desabilitamos os pixels que estão fora da circunferência de raio `radius`, mais distantes da frequência-zero. Um resultado típico obtido a partir das transformações pode ser observado na figura 3.

```
# Criando a máscara
mask = np.zeros((height, width))
cv2.circle(mask, (width // 2, height // 2), radius, (1, 1, 1),
    thickness = -1)

# Aplicando o filtro
masked = np.multiply(image, mask)

# Deslocando o espectro de Fourier para a posição original
masked_original_position = np.fft.ifftshift(masked)
filtered_image = np.fft.ifft2(masked_original_position).real
```

1.2 Filtros passa-alta

Novamente [1] cita filtros agora do tipo passa-alta, ressaltando o aspecto de realce de nitidez. A nitidez destaca as transições em diferenças de intensidade. Os usos de nitidez de imagem variam de impressão eletrônica e imagem médica a industrial inspeção e orientação autônoma em sistemas militares. Como o cálculo da média é análogo à integração, é possível concluir que a nitidez pode ser realizada por diferenciação espacial. A força da resposta de um operador derivativo é proporcional à magnitude da descontinuidade de intensidade no ponto em que o operador é aplicado. Asim, a diferenciação da imagem melhora as bordas e outras descontinuidades (como ruído) e diminui a ênfase nas áreas com variação de intensidade menos abruptas. O ajuste de nitidez costuma ser chamado de filtragem passa-alta. Neste caso, altas frequências, que são responsáveis por detalhes mais finos, são passados,

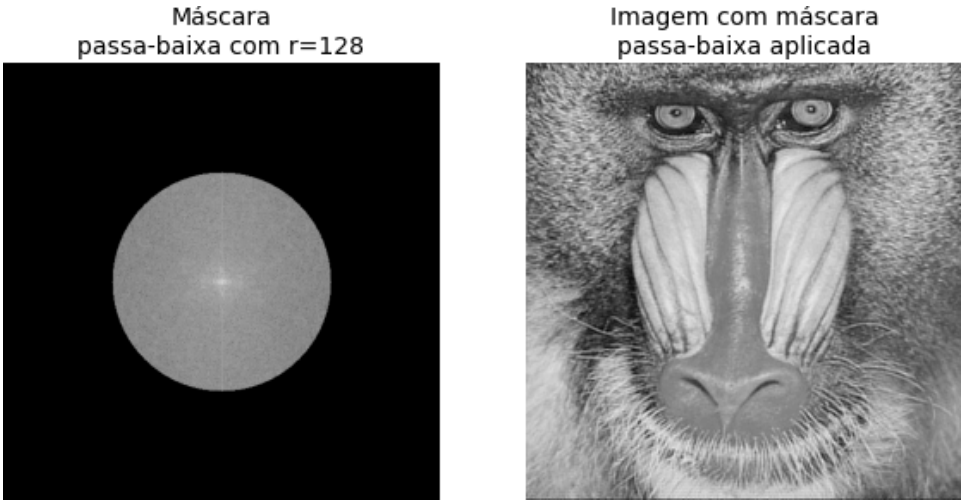

Figura 3: Espectro de Fourier considerando máscara passa-baixa e imagem resultante.

enquanto as frequências baixas são atenuadas ou rejeitadas.

Um resultado típico do filtro passa-alta pode ser visualizado na figura 4.

Para a atividade, fizemos algo semelhante à estrutura para o filtro passa-baixa: identificamos as dimensões da imagem (`height` e `width` para altura e largura, respectivamente), e criamos um retângulo com as mesmas dimensões que será uma máscara com 1 e depois com a bilbioteca `opencv` criamos um círculo na máscara, no centro da imagem, sem borda e preenchido com o valor 0 utilizando o método `circle`. Em seguida, multiplicamos a máscara (que contém valores 0 e 1) pela imagem, habilitando os pixels onde há o valor 1, mais distantes do centro e até a distância do raio `radius`, e desabilitamos os pixels que estão dentro da circunferência de raio `radius`, mais próximos da frequência-zero.

```
# Criando a máscara
mask = np.ones((height, width))
cv2.circle(mask, (width // 2, height // 2), radius, (0, 0, 0),
    thickness = -1)

# Aplicando o filtro
masked = np.multiply(image, mask)

# Deslocando o espectro de Fourier para a posição original
masked_original_position = np.fft.ifftshift(masked)
filtered_image = np.fft.ifft2(masked_original_position).real
```

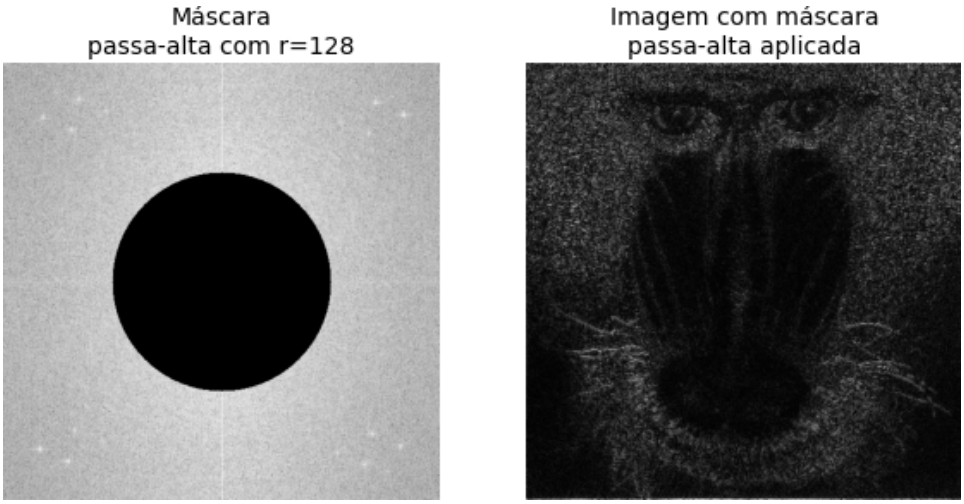

Figura 4: Espectro de Fourier considerando máscara passa-alta e imagem resultante.

1.3 Filtros passa-faixa

Os filtros passa-faixa são utilizados em aplicações em que é de interesse processar bandas específicas de frequências. Segundo [1], os filtros passa-faixa e rejeita-faixa no domínio da frequência são formados por combinações de filtros passa-baixa e passa-alta, sendo que este último também é derivado das funções passa-baixa. Isso significa que as funções de transferência de filtro passa-baixo são a base para formar as funções de filtro passa-alta, rejeita-faixa e passa-faixa.

No nosso caso, do ponto de vista de código, criamos duas estruturas de máscaras: identificamos as dimensões da imagem (`height` e `width` para altura e largura, respectivamente), e criamos dois retângulos com as mesmas dimensões que serão duas máscaras com 0 e depois com a bilbioteca `opencv` criamos um círculo em cada máscara, no centro de cada imagem, uma com o raio maior e outra com o raio menor, sem borda e preenchido com o valor 1 utilizando o método `circle`.

Em seguida, utilizamos o operador XOR para criar a máscara final, conforme a tabela 1.

Ao final, multiplicamos a máscara resultante pela imagem original, tendo então as informações apenas de frequências dentro da faixa gerada entre os raios menor e maior. Como um dos possíveis resultados obtidos, temos a figura 5.

Entrada		Saída
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 1: Tabela-verdade XOR

```
# Criando a máscara menor
mask_small = np.zeros((height, width))
cv2.circle(mask_small, (width // 2, height // 2), small_radius,
    (1, 1, 1), thickness = -1)
mask_small = mask_small.astype(np.int32)

# Criando a máscara maior
mask_big = np.zeros((height, width))
cv2.circle(mask_big, (width // 2, height // 2), large_radius,
    (1, 1, 1), thickness = -1)
mask_big = mask_big.astype(np.int32)

# Operando com XOR para selecionar apenas a área entre as duas
    máscaras
mask = mask_small ^ mask_big

# Aplicando o filtro
masked = np.multiply(image, mask)

# Deslocando o espectro de Fourier para a posição original
masked_original_position = np.fft.ifftshift(masked)
filtered_image = np.fft.ifft2(masked_original_position).real
```

1.4 Compressão

A compressão é a arte de reduzir a quantidade de dados necessários para representar uma imagem. Segundo [1], tem a ver com dados e informação. Para o contexto em que estamos trabalhando, eles tem significado distinto: os dados são os meios pelos quais as informações são transmitidas. Um volume distinto de dados pode ser usado para representar a mesma quantidade de informações

Para o presente trabalho, identificamos a frequência máxima do espectro de Fourier (no domínio dos números reais) e a partir do parâmetro `threshold_percentage`. Então o limiar `threshold = threshold_percentage * max_value`.

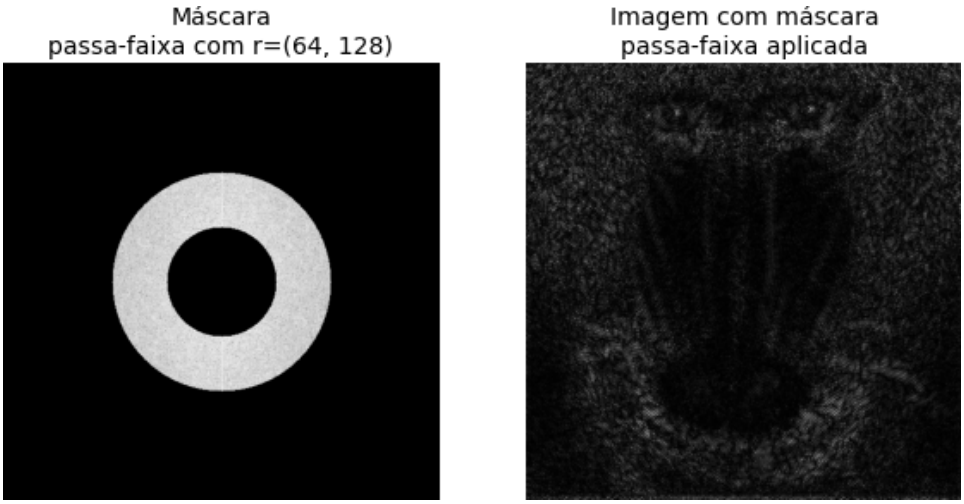

Figura 5: Espectro de Fourier considerando máscara passa-faixa e imagem resultante.

A partir daí, removemos qualquer valor de frequência menor ou igual ao `threshold`.

Capturamos também a porcentagem de valores ignorados dividindo a quantidade abaixo do limiar sobre a multiplicação de altura pela largura da imagem.

Em seguida plotamos a imagem a partir da inversa da transformada de Fourier:

Figura 6: Imagem, espectros de Fourier (em preto, pontos abaixo do limiar) e imagem resultante.

```python
# Realizando a cópia da imagem para manutenção da original
image_copy = image.copy()

# Recuperando as dimensões da imagem
height, width = image_copy.shape

# Recuperando o valor máximo do pixel da imagem
max_value = np.abs(image_copy).max().real

# Definindo o limiar máximo como sendo o parâmetro a
    porcentagem
# do valor threshold em termos do valor máximo
threshold = threshold_percentage * max_value

# Calcula a quantidade de pixels que são abaixo do limiar
    máximo da imagem
qtd = np.abs(image_copy) <= threshold

# Calcula a porcentagem de pixels abaixo do limiar definido
percentage_below_threshold = qtd.sum() / (height * width)

# Remove as frequencias abaixo do limiar
filtered = np.where(np.abs(image_copy) <= threshold, 0,
    image_copy)

# Deslocando o espectro de Fourier para a posição original
filtered_original_position = np.fft.ifftshift(filtered)
filtered_image = np.fft.ifft2(filtered_original_position).real
```

Um exemplo de resutlado da compressão é a figura 6.

1.5 Rotação

O autor [1] falando especificamente do espectro de um retângulo no seu exemplo 4.13, mostra que o espectro de Fourier de um retângulo transla-

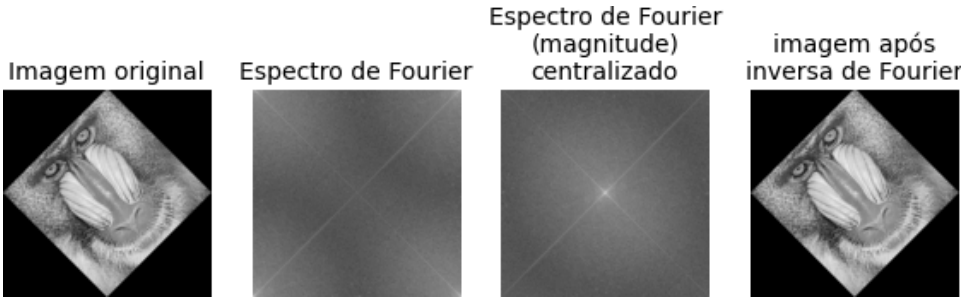

Figura 7: Imagem, espectros de Fourier e imagem filtrada da imagem original rotacionada de 45°.

dado no plano é sempre o mesmo, independente de posição desde que as características da imagem sejam mantidas.

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(x_0u/M + y_0v/N)} \tag{3}$$

Isso significa que, multiplicar $f(x, y)$ pelo exponencial muda a origem da transformada discreta de Fourier para (u_0, v_0) e, por outro lado, multiplicar $F(u, v)$ pelo negativo deste mesmo exponencial muda a origem de $f(x, y)$ para (x_0, y_0). Desta forma, a translação não tem efeito na magnitude (ou espectro) de $F(u, v)$.

Utilizando coordenadas polares:

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \\ u &= \omega\cos\varphi \\ v &= \omega\sin\varphi \end{aligned}$$

Temos a seguinte transformação:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0) \tag{4}$$

Que indica que a rotação de $f(x, y)$ por um ângulo θ_0 rotaciona $F(u, v)$ pelo mesmo ângulo. Por outro lado, rotacionar $F(u, v)$ rotaciona $f(x, y)$ pelo mesmo ângulo.

O resultado equivalente à figura 2 é a figura 7.

2 Código

2.1 Imagens

As imagens utilizadas foram capturadas do endereço https://www.ic.unicamp.br/~helio/imagens_png/ além de outras com dimensões horizontais maiores que verticais e dimensões verticais maiores que dimensões horizontais.

2.2 Espectro de Fourier

Para a utilização do espectro de Fourier, utilizamos os métodos da biblioteca `NumPy`: realizamos a transformação para a transformada discreta de Fourier utilizando o método `fft2`[4] do módulo `ftt`[3], trasladamos para o centro a componente de frequência-zero, a partir do método `fftshift`[6] do mesmo módulo `fft`. Para a inversão das transformações, utilizamos os métodos `ifft2`[5] e em seguida o método `ifftshift`[7] ambos do mesmo módulo `ftt`.

Para os núcleos dos filtros, utilizamos os seguintes valores:

- passa-baixa: $2, 4, 8, 16, 32, 64, 128, 256$. Caso a imagem tenha alguma das dimensões menores (altura ou largura), então o programa não executa a transformação considerando aquela dimensão de filtro passa-baixa.
- passa-alta: $2, 4, 8, 16, 32, 64, 128, 256$. Caso a imagem tenha alguma das dimensões menores (altura ou largura), então o programa não executa a transformação considerando aquela dimensão de filtro passa-alta.
- passa-faixa: 2 a 4, 4 a 8, 8 a 16, 16 a 32, 32 a 64, 64 a 128 e 128 a 256. Caso a imagem tenha alguma das dimensões menores (altura ou largura) do que o raio maior, então o programa não executa a transformação considerando aquelas dimensões de filtro passa-faixa.

No caso da compressão, utilizamos valores $1.00000000e{-}01$, $2.78255940e{-}02$, $7.74263683e-03$, $2.15443469e-03$, $5.99484250e-04$, $1.66810054e-04$, $4.64158883e{-}05$, $1.29154967e{-}05$, $3.59381366e{-}06$ e $1.00000000e{-}06$ para serem multiplicados pelo valor de maior intensidade do espectro de Fourier e assim considerar o limiar como o resultado desta multiplicação.

No caso da rotação, realizamos ela para um ângulo de 45° apenas.

Referências

[1] GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. 4th. ed. USA: Pearson Education Ltd., 2018. ISBN 0-13-335672-8

[2] PEDRINI, H. C.; *Introdução ao Processamento Digital de Imagem MC920 / MO443*, 2021. https://www.ic.unicamp.br/~helio/disciplinas/MO443/aula_dominio_frequencia.pdf

[3] numpy.fft, https://numpy.org/doc/stable/reference/routines.fft.html Acessado em: 2021-05-02

[4] numpy.fft.fft2, https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html Acessado em: 2021-04-29

[5] numpy.fft.ifft2, https://numpy.org/doc/stable/reference/generated/numpy.fft.ifft2.html Acessado em: 2021-05-04

[6] numpy.fft.fftshift, https://numpy.org/doc/stable/reference/generated/numpy.fft.fftshift.html Acessado em: 2021-04-27

[7] numpy.fft.ifftshift, https://numpy.org/doc/stable/reference/generated/numpy.fft.ifftshift.html Acessado em: 2021-05-04

Acessado em 2021-05-08