Universidade Federal do Maranhão
Coordenação do Curso de Engenharia da Computação

Processamento de Imagens - Atividade 3^a Nota

Aluno

Claudio Henrique Velozo Alexandre

Engenharia da Computação

Professor

Prof. Dr. MARCUS VINICIUS DE SOUSA LOPES

Coordenação do Curso de Engenharia da Computação

São Luís, 2023

1 Tarefa 2 e Tarefa 3

A implementação do filtro laplaciano pode ser vista na figura 2. Com esse filtro é possível criar um filtro de realce(figura 2 da linha 23 à 32) e com isso aplicar na imagem com blur(figura 3).

Figura 1: Imagem original

```
# laplaciano
P, Q = F.shape
H = np.ones((P, Q), dtype=np.float32)
for u in range(P):
    for v in range(Q):
        H[u, v] = -4*np.pi*np.pi*((u-P/2)**2 + (v-Q/2)**2)

# Laplace
Lap = H * F
Lap = np.fft.ifftshift(Lap)
Lap = np.real(np.fft.ifft2(Lap))
oldRange = np.max(Lap) - np.min(Lap)
newRange = 1 - -1
LapScaled = (((Lap - np.min(Lap)) * newRange) / oldRange) + -1
c = -1
g = f + c*LapScaled
g = np.clip(g, 0, 1)
```

Figura 2: Filtro laplaciano e realce

O espectro de frequencia da imagem original pode é mostrado na figura **??** abaixo:

A figura 4 exibe o resultado da filtragem:

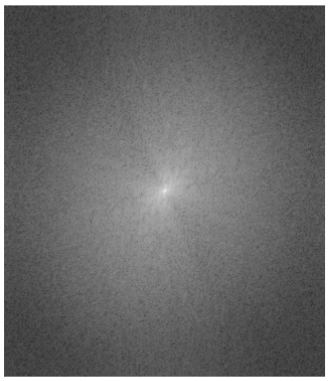

Figura 3: Espectro de frequência da imagem original

Figura 4: Resultado da filtragem e realce

2 Tarefa 4

Para a tarefa 4 usou-se o filtro Notch, que são filtros capazes de rejeitar uma faixa bastante estreita de frequências, atuando quase que exclusivamente na frequência selecionada e bem pouco nas outras ao redor. Sua utilização é recomendada quando o sinal a ser atenuado é bem definido. Pelo fato de atuar em faixas reduzidas de frequências, filtros notch interferem pouco na qualidade do sinal.

A figura 5 que é a imagem a ser filtrada e a figura 6 mostra o espectro de frequência obtido ao aplicar a transformado de fourirer na imagem.

Ao perceber e localizar os picos de frequência, implementou-se uma função para rejeitar essa faixa de frequência que pode ser vista na figura 7 a seguir:

O espctro de frequência com o filtro notch pode ser visto na figura 8 abaixo. Note que tentou-se esconder os picos de frequência utilizando circulos pretos:

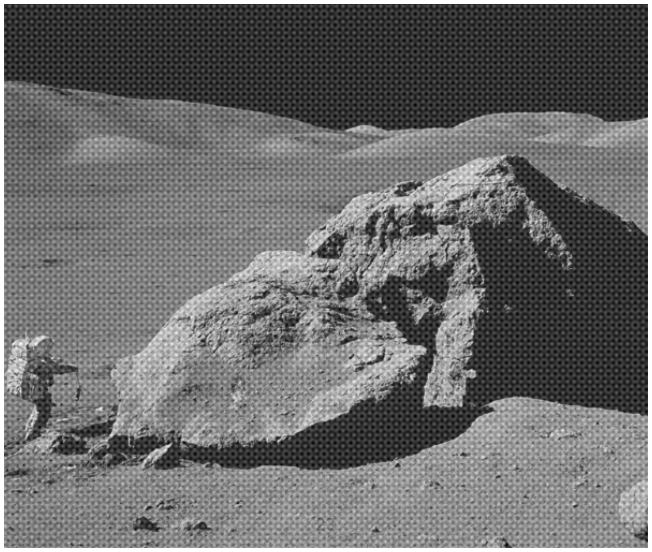

Figura 5: Imagem original

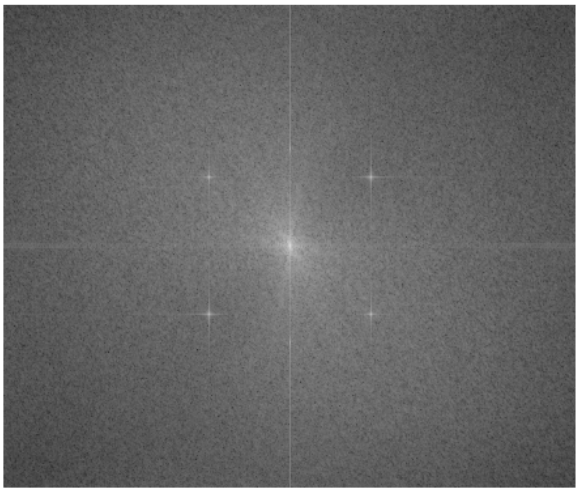

Figura 6: Espectro de Frequência da imagem original

A figura 9 mostra o resultado da filtragem. A filtragem não ficou 100% mas é possível perceber a redução do ruído em comparação a imagem original.

```python
import numpy as np


def filtro_rejeita_notch(shape, d0=9, u_k=0, v_k=0):
    (M, N) = shape

    H_0_u = np.repeat(np.arange(M), N).reshape((M, N))
    H_0_v = np.repeat(np.arange(N), M).reshape((N, M)).transpose()

    D_uv = np.sqrt((H_0_u - M / 2 + u_k) ** 2 + (H_0_v - N / 2 + v_k) ** 2)
    D_muv = np.sqrt((H_0_u - M / 2 - u_k) ** 2 + (H_0_v - N / 2 - v_k) ** 2)

    seletor_1 = D_uv <= d0
    seletor_2 = D_muv <= d0

    seletor = np.logical_or(seletor_1, seletor_2)

    H = np.ones((M, N))
    H[seletor] = 0

    return H

```

Figura 7: Filtro Notch

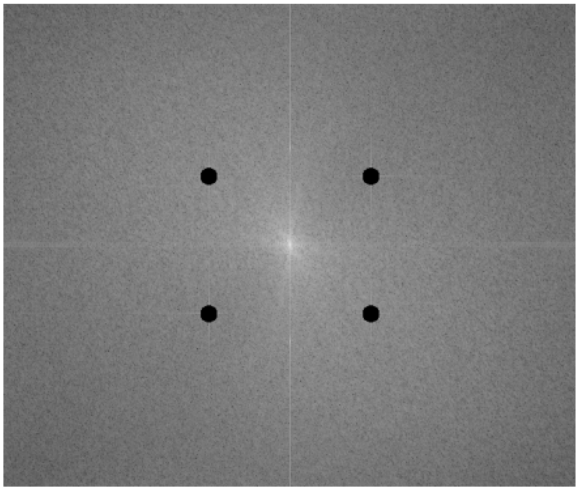

Figura 8: Espectro do Filtro Notch

3 Tarefa 5

Para a tarefa 5 o filtro Notch também foi utilizado.

Podemos observar os ruídos da imagem original e seu espectro de frequência nas figuras 10 e 11 respectivamente.

A mesma função da figura 7 foi utilizada, mudando apenas as coordenadas dos pixels para os picos de frequência.

O filtro notch pode ser visto na figura 12 e o resultado está na figura 13.

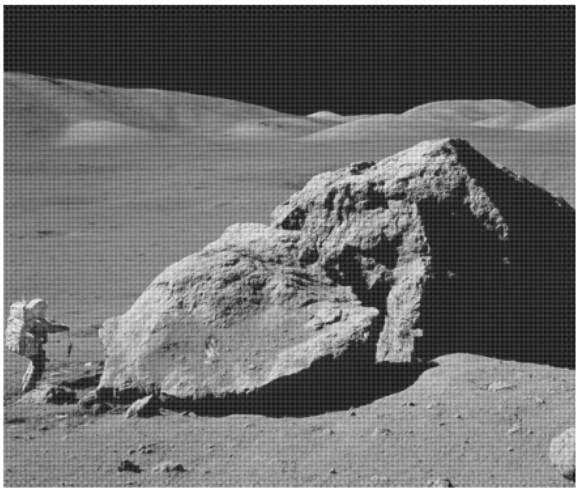

Figura 9: Imagem resultante da filtragem

Figura 10: Imagem original

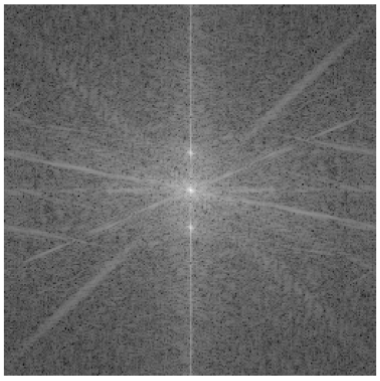

Figura 11: Espectro de Frequência da imagem original

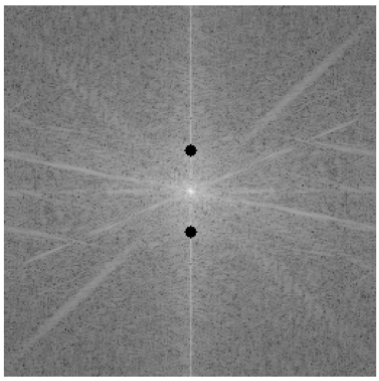

Figura 12: Espectro do Filtro Notch

Figura 13: Imagem resultante da filtragem

4 Implementação

Os código completos podem ser obtidos no meu repositório no github.

link: <https://github.com/cHenrique0/processamento-imagens-ufma/tree/main/provas/P3>