UNIVERSIDADE FEDERAL DE SÃO CARLOS - UFSCAR
CIÊNCIA DA COMPUTAÇÃO
PROCESSAMENTO DIGITAL DE IMAGENS

RELATÓRIO: FILTRO PASSA BAIXA E PASSA-ALTA BUTTERWORTH

São Carlos, Outubro de 2020

Discentes:

Agustin Gabriel Amaral Castillo, 592862

Jorge Vinicius Gonçalves, 758594

Thiago de Moraes Teixeira, 760667

Victoria de Martini de Souza, 75937

Docente:
Cesar Henrique Comin, Prof.
Universidade Federal de São Carlos – UFSCar

INTRODUÇÃO

Dentro dos filtros passivos, encontramos o filtro passa-baixa e o filtro passa-alta. Neste experimento, iremos verificar o funcionamento destas.

A técnica “filtro passa-baixa” está apresentada no arquivo “passabaixa.py”, o funcionamento dela permite a passagem de baixas frequências sem dificuldades e atenua, ou reduz, a amplitude das frequências maiores que a frequência de corte. A quantidade de atenuação para cada frequência varia de filtro para filtro.

A técnica “filtro passa-alto” está apresentada no arquivo “passaalta.py”, permitem a passagem de sinais de alta frequência e reduz a intensidade de sinais de baixa frequência. Ou seja, a partir de uma frequência de referência ele permite que frequências mais altas passem livremente e frequências mais baixa sejam atenuadas.

Importante lembrar que sinais digitais são visto como se estivessem repetidos diversas vezes tanto no domínio espacial quanto no da frequência. Portanto, precisamos preencher os sinais com zeros antes de aplicar a filtragem (partes do processo como esta são tratadas no código, porém não abordadas neste documento, que tem como objetivo abordar pontos específicos do método aplicado).

CÓDIGO

1.1 Arquivo passabaixa.py

Temos o filtro passa-baixa sendo aplicado em uma imagem, seguindo a seguinte fórmula que é aplicada pixel a pixel da imagem:

$$H(\mu,\nu) = \frac{1}{1 + \left(\frac{D(\mu,\nu)}{D_0}\right)^{2n}} \qquad D(\mu,\nu) = \sqrt{\mu^2 + \nu^2}$$

que para ser traduzida em código temos:

```
for row in range(num_rows):
        for col in range(num_cols):
            distance = np.sqrt(freq_r[row]**2 + freq_c[col]**2)
            low_pass_butterwoorth_filter[row, col] = 1/(1 + ((distance/d0)**(2*n)))
```

Onde será produzido o filtro passa-baixa que depois será aplicado na imagem pelo código:

```
# Aplicação do filtro na imagem
Fimg_filtered = lowpass_filter*Fimg
```

Fimg é a imagem com a transformada de Fourier aplicada, obtendo o resultado em *Fimg_filtered* multiplicando pixel a pixel os valores do filtro com os da *Fimg.*

Após a aplicação do filtro realizamos a transformação inversa de Fourier e removemos os números complexos que ela pode gerar, ficando apenas com os números reais, que formam agora a imagem com o filtro passa-baixa aplicado.

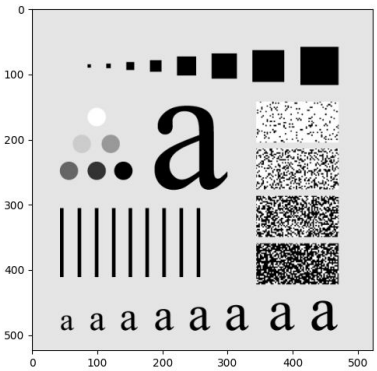
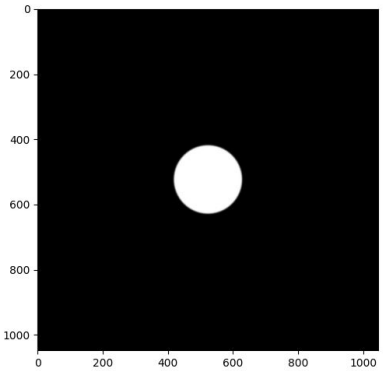
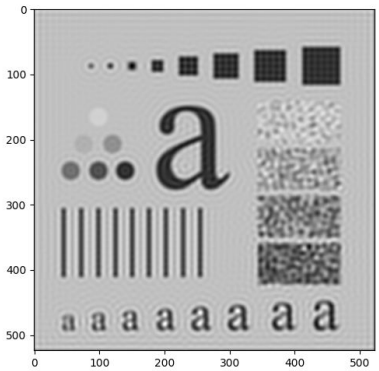
```
# fftshift é utilizado para retornar o espectro para as posições esperadas pela função ifft2
Fimg_filtered = fftshift(Fimg_filtered)
img_filtered = ifft2(Fimg_filtered)

# A transformação inversa de Fourier possui números complexos para niveis de precisão
# Aqui filtramos somente a parte real do resultado
img_filtered = np.real(img_filtered)
```

1.2 Análise dos valores de *d0* e *n* na fórmula

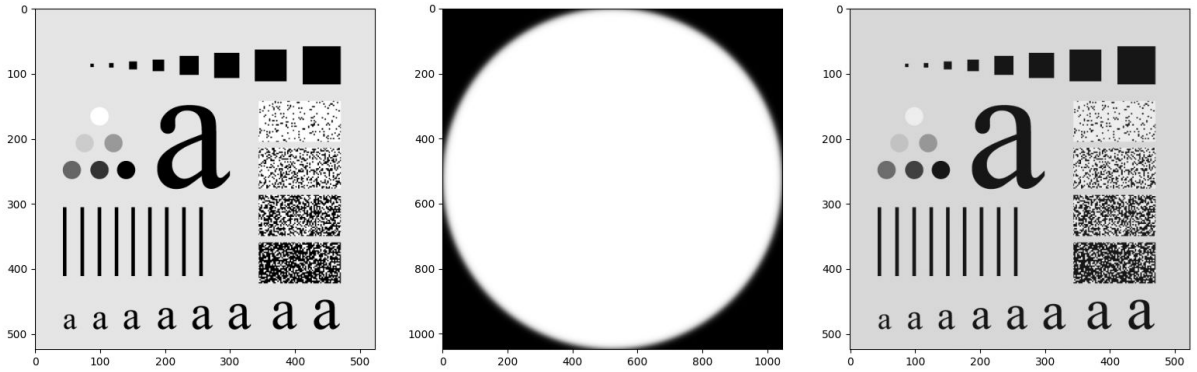
Após alguns testes de valores, chegamos a conclusão que abstratamente podemos associar o valor de *d0* ao raio do filtro enquanto o valor de *n* pode ser associado ao efeito de *Blur* aplicado às bordas do círculo, obtendo o efeito gaussiano dependendo do valor utilizado.

Valores de *d0*: Quanto mais próximo de 1 maior será o raio do filtro, por outro lado quanto mais próximo de 0 o valor, menor será o raio.
Valores de *n*: Quanto maior o valor menor será o efeito de *Blur* no círculo do filtro, por outro lado quanto mais próximo de 0 maior será o efeito.

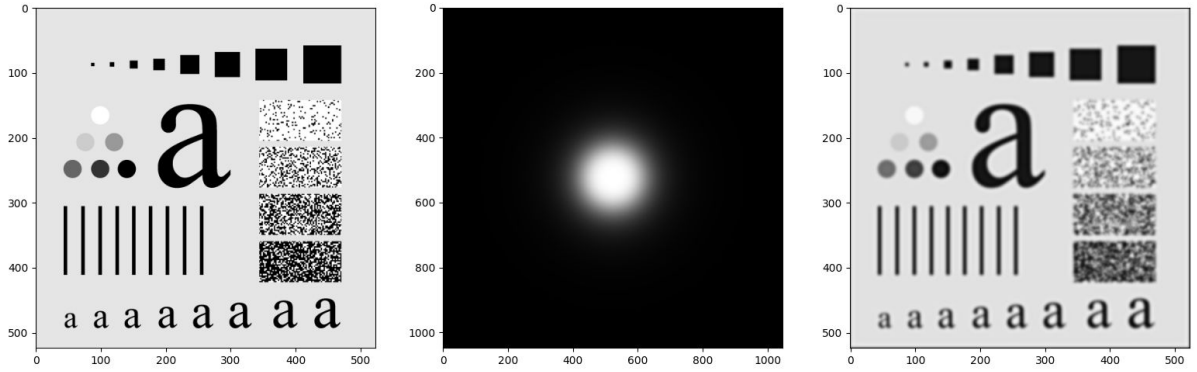
Com *d0=0,1* e *n=50*: vamos ter um filtro próximo ao passa-baixa ideal

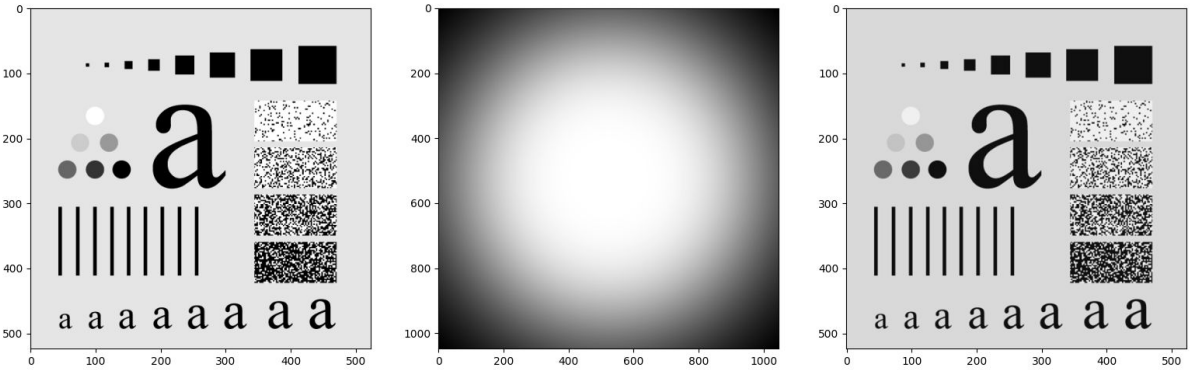
Com *d0=0,5* e *n=50*: teremos o raio extremamente grande e as bordas definidas, sem efeito de *Blur* pois o valor de *n* se manteve alto



Com *d0=0,1* e *n=2*: obtemos um filtro próximo do passa-baixa gaussiano, pois o raio será reduzido com o efeito de *Blur* acentuado em sua extensão, com o seguinte resultado:



Por último com *d0=0,5* e *n=2*: temos o *d0* mais próximo de 1, portanto o raio será grande e com o *n* se aproximando de 0 vamos ter o efeito de *Blur* presente em toda extensão do filtro:



2.1 Arquivo passaalta.py

Temos o filtro passa-alta sendo aplicado em uma imagem, seguindo a seguinte fórmula que é aplicada pixel a pixel da imagem:

$$H(\mu,\nu) = \frac{1}{1 + \left(\frac{D_0}{D(\mu,\nu)}\right)^{2n}}$$

que para ser traduzida em código temos:

```
    for row in range(num_rows):
        for col in range(num_cols):
            distance = np.sqrt(freq_r[row]**2 + freq_c[col]**2)
            high_pass_butterwoorth_filter[row, col] = 1/(1 + (d0/(distance+1e-10)**(2*n)))
```

Onde será produzido o filtro passa-alta que depois será aplicado na imagem pelo código:

```
# Aplicação do filtro na imagem
Fimg_filtered = highpass_filter*Fimg
```

Fimg é a imagem com a transformada de Fourier aplicada, obtendo o resultado em *Fimg_filtered* multiplicando pixel a pixel os valores do filtro com os da *Fimg.*

Após a aplicação do filtro realizamos a transformação inversa de Fourier e removemos os números complexos que ela pode gerar, ficando apenas com os números reais, que formam agora a imagem com o filtro passa-alta aplicado.

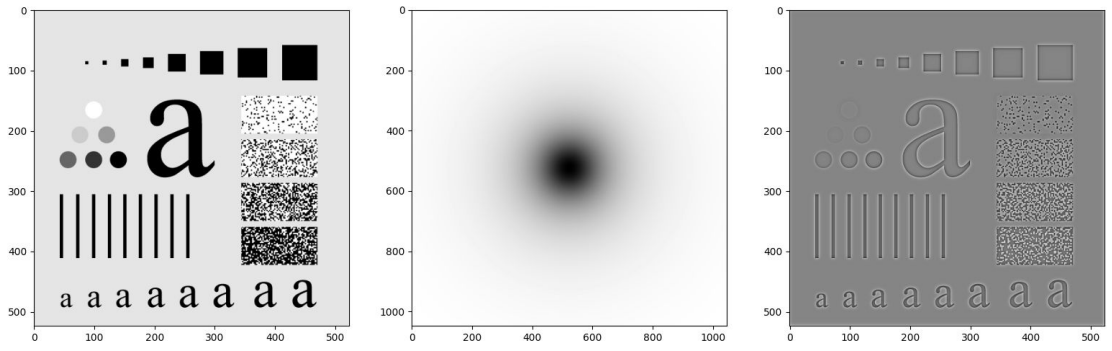
```
# fftshift é utilizado para retornar o espectro para as posições esperadas pela função ifft2
Fimg_filtered = fftshift(Fimg_filtered)
img_filtered = ifft2(Fimg_filtered)

# A transformação inversa de Fourier possui números complexos para niveis de precisão
# Aqui filtramos somente a parte real do resultado
img_filtered = np.real(img_filtered)
```

2.2 Análise dos valores de *d0* e *n* na fórmula

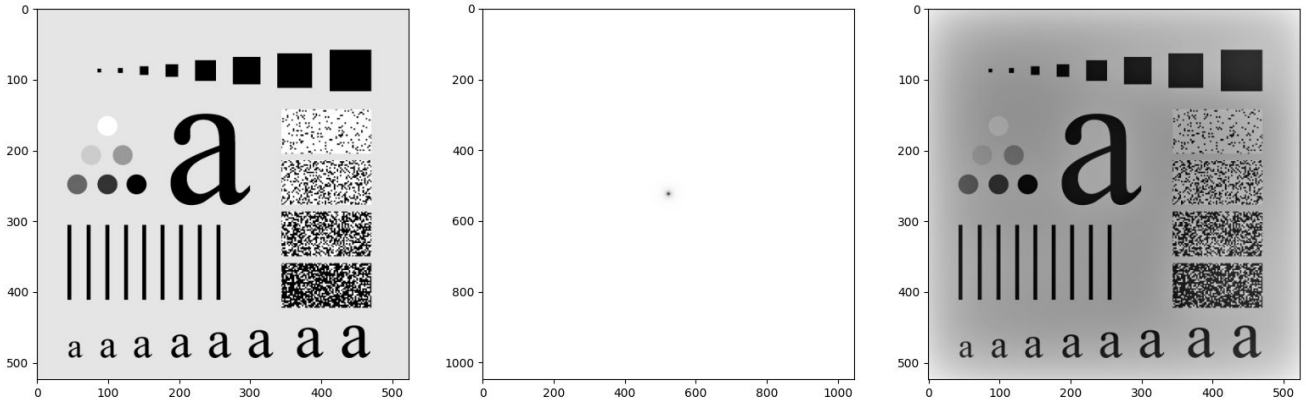
Agora no filtro passa-alta, obtemos uma relação mútua entre *d0* e *n* onde ambos contribuem para o valor do raio que o filtro terá, enquanto para o efeito de *Blur* temos a participação somente do valor de *d0*.

Valores de *d0*: Quanto mais próximo de 0 (i.e. 0,00000001) menor será o efeito de *Blur* (mais definida a borda do círculo fica), quanto mais casas decimais menor será o raio do círculo do filtro, pois a tendência é o efeito de *Blur* sumir, fazendo com que o círculo fique mais suscinto (vamos demonstrar abaixo).
Valores de *n*: Quanto maior o valor maior será o raio do círculo do filtro, por outro lado quanto mais próximo de 0 menor será o raio.

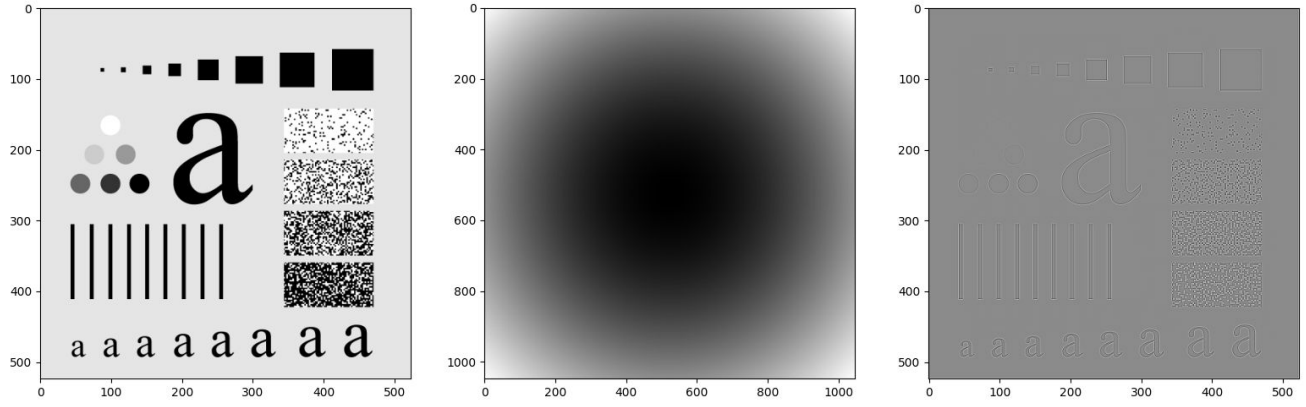
Com *d0=0,01* e *n=1*: Obtemos um filtro passa-alta gaussiano, pois o efeito de *Blur* terá um bom efeito e como as casas decimais de *d0* não são muitas o valor de *n=1* é o bastante para compensar o raio.



Com *d0=0,00001* e *n=1*: Fixamos *n=1* para observarmos, o resultado com 5 casas decimais em *d0*, onde o raio tende a diminuir, junto com o efeito de *Blur*, obtendo o seguinte resultado, onde o filtro é praticamente um ponto, porém com zoom podemos observar uma pequena auréola em torno do ponto, onde o efeito de *Blur* ainda é aplicado:



Com *d0=2* e *n=1*: Obtemos um filtro muito próximo ao filtro passa-alto laplaciano observado em aula, como podemos observar:



Para níveis de comparação fixamos *d0* agora

Com *d0=2* e *n=0.2*: Com *n* se aproximando de zero temos que a raio do filtro diminuirá consideravelmente, porém o efeito de *Blur* continua com um grande alcance como podemos observar.