Projeto Final - Imagens Médicas 2

October 19, 2021

Universidade Federal de Uberlândia

Faculdade de Engenharia Elétrica

Imagens Médicas 2

Discente: Levy Gabriel da S. G.

---

1 Projeto final

O projeto final será dividido em duas seções principais. A primeira consiste no processamento de uma imagem contaminada com diversos tipos de ruídos e, estes deverão ser filtrados por diferentes filtros e no final serão comparados os resultados.

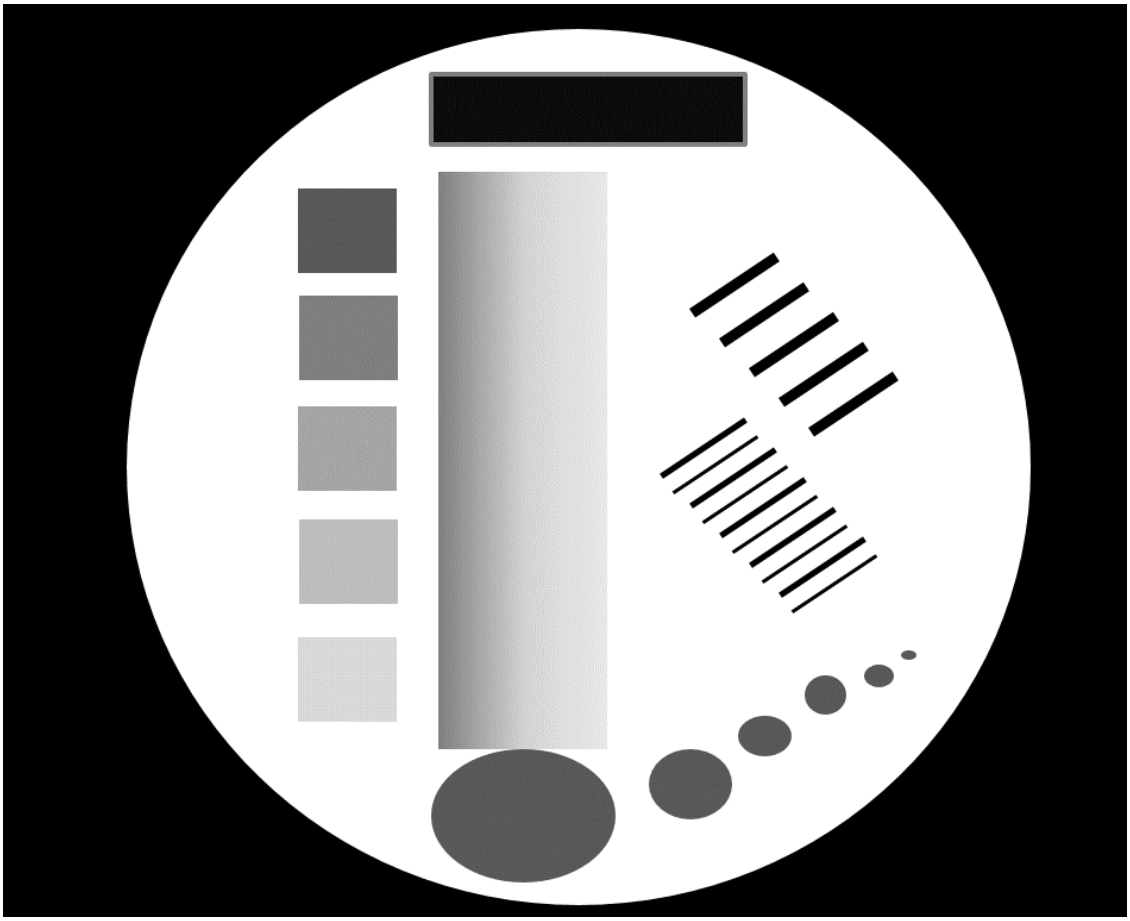
Na segunda e última seção serão comparados os efeitos de filtros no domínio da frequência e do espaço aplicados a diferentes imagens.

1.1 Remoção de ruído

Nesta seção será utilizada uma imagem que será contaminada por 4 diferentes ruídos, estes são: ruído branco, ruído sal e pimenta, ruído Gaussiano e ruído exponencial. Para cada imagem ruidosa serão experimentados três diferentes filtros: filtro de média, filtro de mediana e um filtro passa-baixas Butterworth.

De início, carrega-se a imagem que será utilizada no processamento:

[ ]:
```
pkg load image
img = double(imread('imagem_projeto_item1.png'));
imshow(img)
```



1.1.1 Contaminação da imagem por ruídos

Obtém-se as dimensões da imagem para que sejam criadas as máscaras de ruído que serão adicionadas à imagem original:

[ ]:
```
size_ = size(img)
```

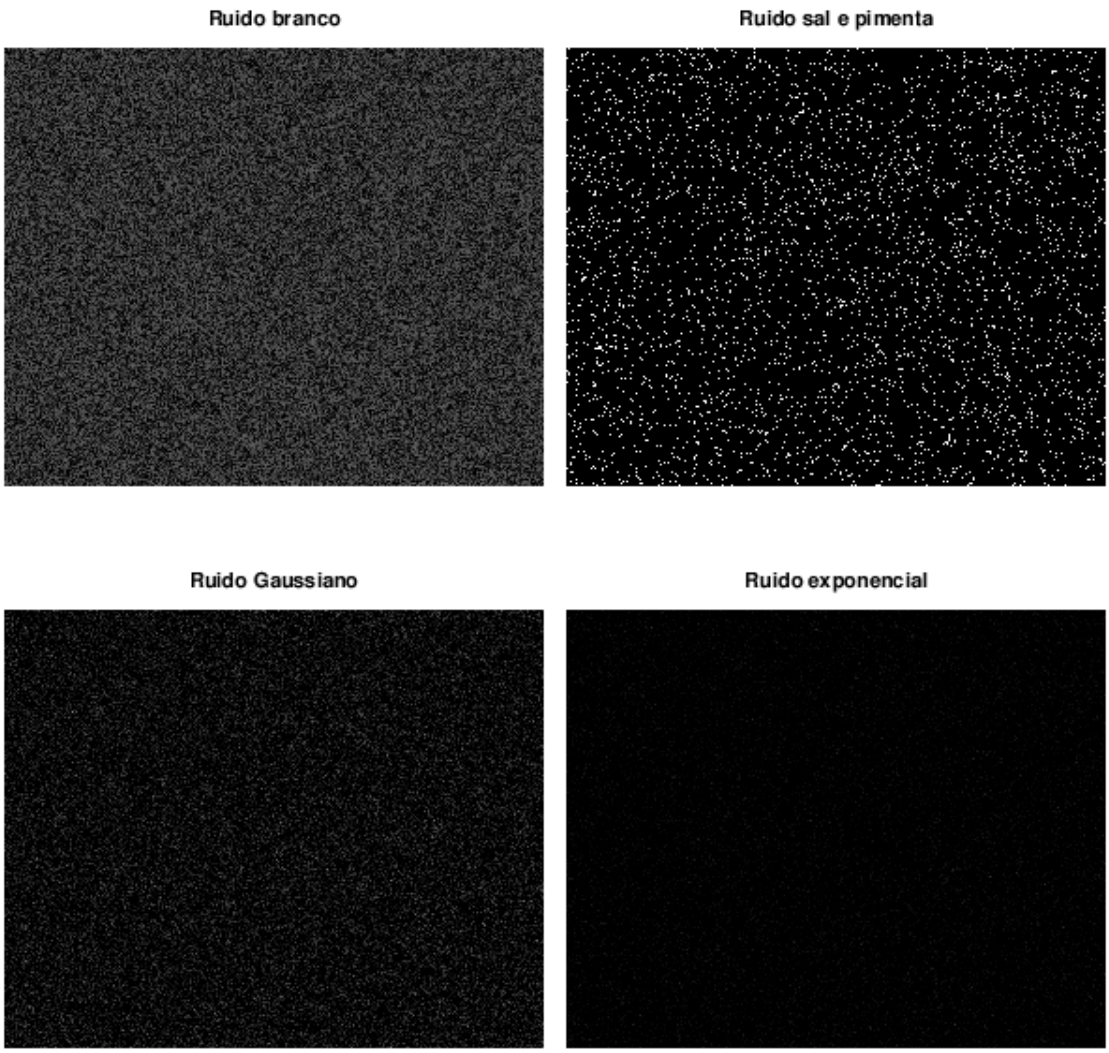
```
size_ =

   767   945
```

O pacote de estatística também é carregado para que as distribuições de probabilidade sejam utilizadas para gerar o ruído:

[ ]:
```
pkg load statistics
```

Em seguida são definidas as máscaras de ruído. Primeiramente o ruído branco de característica constante, em seguida o ruído sal e pimenta, o ruído Gaussiano e, por fim, o ruído exponencial.

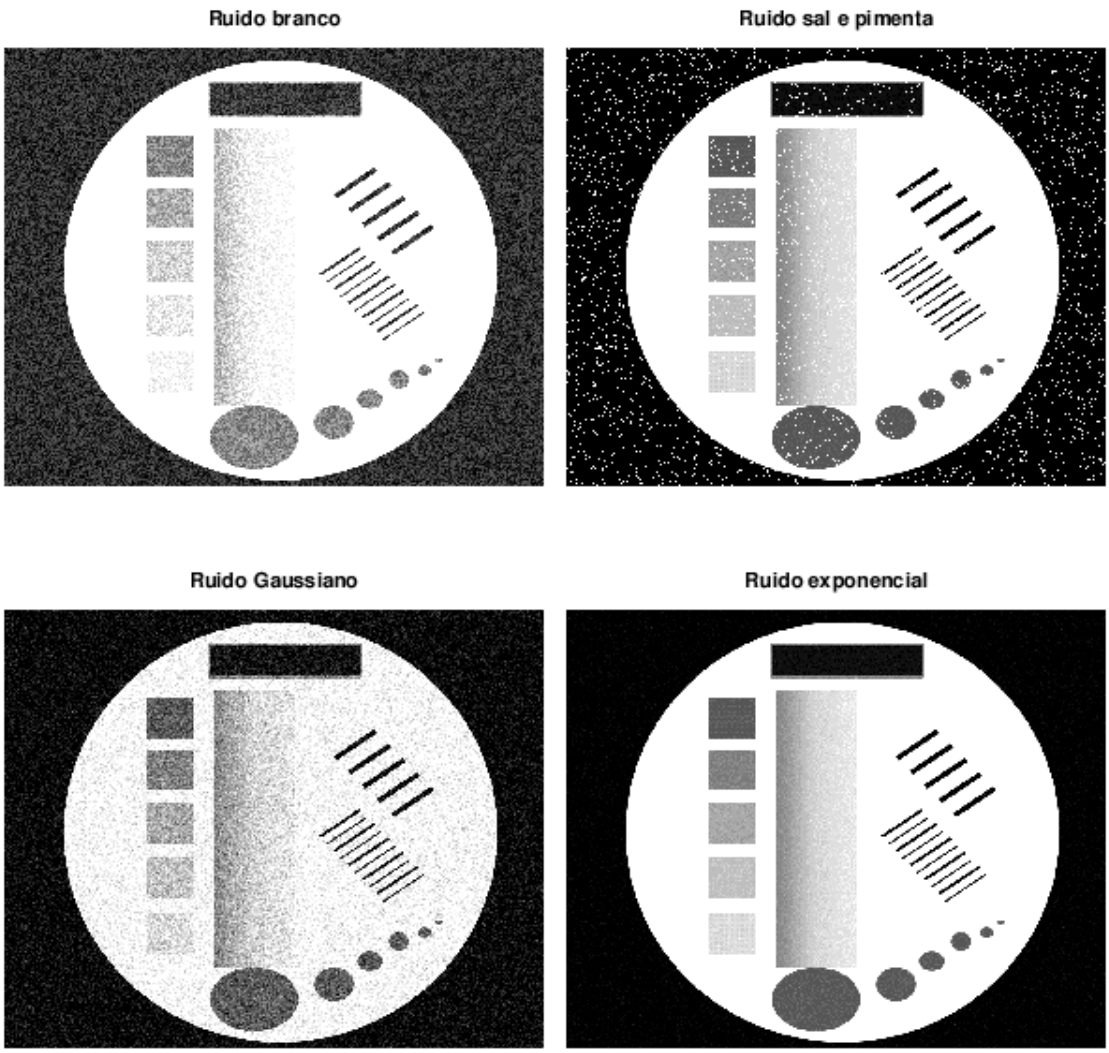
[ ]:
```
gap = 0.01;

% white noise
start_ = 0;
end_ = 100;
whiteNoise = unifrnd(start_, end_, size_);
h = subplot(2,2,1);
set(h,'Position', [0+gap .5 .5-2*gap .5])
imshow(uint8(whiteNoise));
title('Ruido branco');

% salt n' pepper noise
pepperThreshold = 0.05;
saltThreshold = 1-pepperThreshold;
saltnpepperNoise = rand(size_);
saltnpepperNoise(saltnpepperNoise <= pepperThreshold) = 0;
saltnpepperNoise(saltnpepperNoise >= saltThreshold) = 255;
h = subplot(2,2,2);
set(h,'Position', [.5+gap .5 .5-2*gap .5])
imshow(uint8(saltnpepperNoise));
title('Ruido sal e pimenta');

% Gaussian noise
mean = 5;
std = 30;
gaussianNoise = normrnd(mean, std, size_);
h = subplot(2,2,3);
set(h,'Position', [0+gap 0 .5-2*gap .5])
imshow(uint8(gaussianNoise));
title('Ruido Gaussiano');

% exponencial noise
mean = 5;
exponencialNoise = exprnd(mean, size_);
h = subplot(2,2,4);
set(h,'Position', [.5+gap 0 .5-2*gap .5])
imshow(uint8(exponencialNoise));
title('Ruido exponencial');

set(gcf,'Position',[0 0 600 600])
```



Por fim os ruídos são incoporados à imagem de interesse:

[ ]:
```
simulada_whiteNoise = img + whiteNoise;
simulada_saltnpepperNoise = img + saltnpepperNoise;
simulada_gaussianNoise = img + gaussianNoise;
simulada_exponencialNoise = img + exponencialNoise;

% white noise
h = subplot(2,2,1);
set(h,'Position', [0+gap .5 .5-2*gap .5])
imshow(uint8(simulada_whiteNoise));
title('Ruido branco');
```

```
% salt n' pepper noise
h = subplot(2,2,2);
set(h,'Position', [.5+gap .5 .5-2*gap .5])
imshow(uint8(simulada_saltnpepperNoise));
title('Ruido sal e pimenta');

% Gaussian noise
h = subplot(2,2,3);
set(h,'Position', [0+gap 0 .5-2*gap .5])
imshow(uint8(simulada_gaussianNoise));
title('Ruido Gaussiano');

% exponencial noise
h = subplot(2,2,4);
set(h,'Position', [.5+gap 0 .5-2*gap .5])
imshow(uint8(simulada_exponencialNoise));
title('Ruido exponencial');

set(gcf,'Position',[0 0 600 600])
```



1.1.2 Inicialização dos filtros

Os filtros podem ser inicializados da seguinte maneira, para o filtro espacial de média será utilizado um *kernel* de dimensões 5x5:

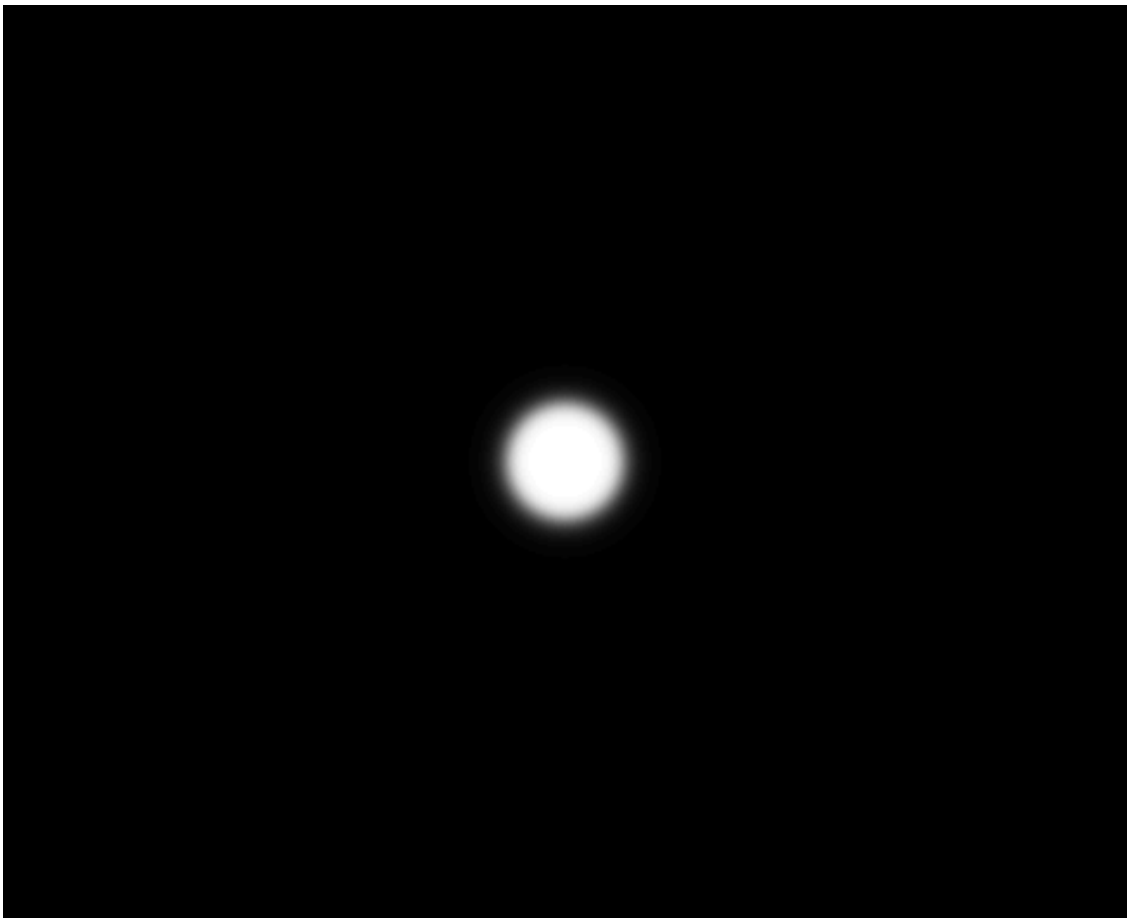
[ ]:
```
size_ = [5,5];
meanFilter = ones(size_)/(size_(1)*size_(2))
```

```
meanFilter =

   0.040000   0.040000   0.040000   0.040000   0.040000
   0.040000   0.040000   0.040000   0.040000   0.040000
   0.040000   0.040000   0.040000   0.040000   0.040000
   0.040000   0.040000   0.040000   0.040000   0.040000
```

```
    0.040000    0.040000    0.040000    0.040000    0.040000
```

E o filtro Butterworth passa-baixas na frequênciade frequência de corte de 50 pixels e ordem 5:

[ ]:
```
f_cutoff = 50;
order = 5;
[M,N] = size(img);
butterworthFilter = zeros(M,N);
cx = floor(M/2)+1;
cy = floor(N/2)+1;
for x= 1:M
    for y= 1:N
        D=sqrt(power((x-cx),2)+power((y-cy),2));
        butterworthFilter(x,y)=1/(1+power(D/f_cutoff,2*order));
    end
end
subplot(1,1,1);
imshow(imresize(butterworthFilter,1/2))
```



No caso do filtro de mediana, este, em si, não possui um *kernel*, portanto ele será aplicado diretamente na imagem por meio da função `medfilt2`.

Por fim, os filtros são aplicados a cada uma das imagens ruidosas. Os resultados serão apresentados somente na próxima seção.

[ ]:
```
function filtered = freq_filt(img, filter)
    img_freq = fft2(img); % Fourier transform
    img_freq_cent = fftshift(img_freq); % image frequency spectrum centralized
    img_complex = ifftshift(img_freq_cent.*filter);
    filtered = real(ifft2(img_complex));
end
```

[ ]:
```
% applying mean filter
simulada_whiteNoise_mean       = imfilter(simulada_whiteNoise, meanFilter);
simulada_saltnpepperNoise_mean = imfilter(simulada_saltnpepperNoise,␣
 ↪meanFilter);
simulada_gaussianNoise_mean    = imfilter(simulada_gaussianNoise, meanFilter);
simulada_exponencialNoise_mean = imfilter(simulada_exponencialNoise,␣
 ↪meanFilter);

% applying butterworth filter
simulada_whiteNoise_butterworth       = freq_filt(simulada_whiteNoise,␣
 ↪butterworthFilter);
simulada_saltnpepperNoise_butterworth = freq_filt(simulada_saltnpepperNoise,␣
 ↪butterworthFilter);
simulada_gaussianNoise_butterworth    = freq_filt(simulada_gaussianNoise,␣
 ↪butterworthFilter);
simulada_exponencialNoise_butterworth = freq_filt(simulada_exponencialNoise,␣
 ↪butterworthFilter);

% applying median filter
simulada_whiteNoise_median       = medfilt2(simulada_whiteNoise);
simulada_saltnpepperNoise_median = medfilt2(simulada_saltnpepperNoise);
simulada_gaussianNoise_median    = medfilt2(simulada_gaussianNoise);
simulada_exponencialNoise_median = medfilt2(simulada_exponencialNoise);
```

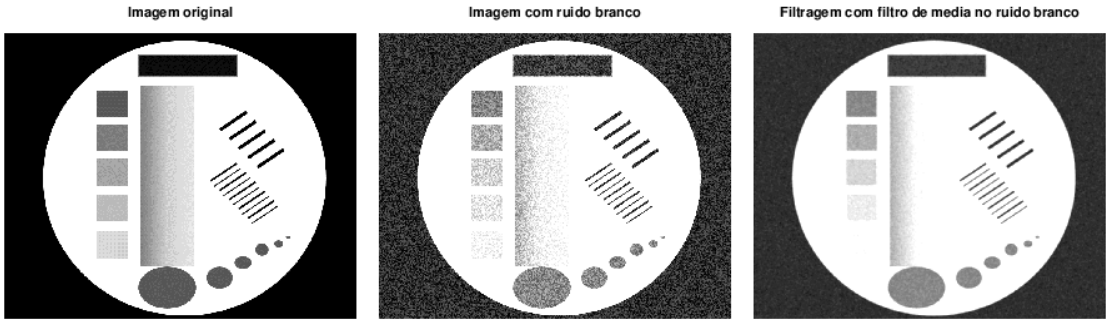
1.1.3 Resultados da remoção de ruído

[ ]:
```
function tripleImages(img1, img2, img3, str2, str3)
    gap = 0.01;

    h = subplot(1,3,1);
    set(h,'Position', [0+gap 0 1/3-2*gap 1])
    imshow(uint8(img1));
    title('Imagem original');

    h = subplot(1,3,2);
```

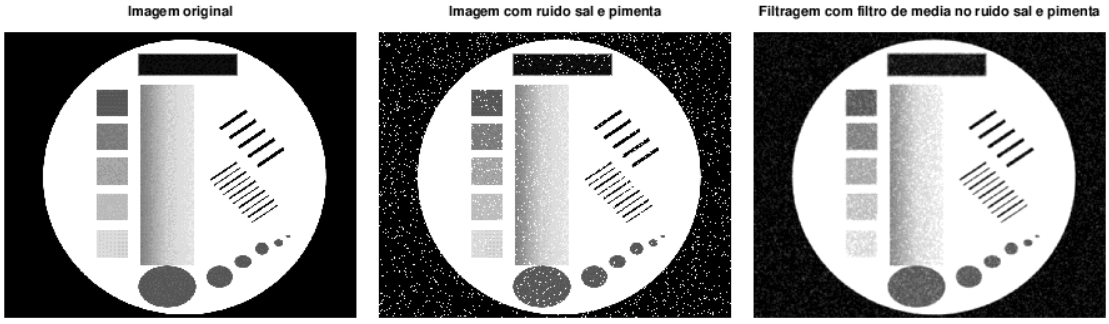
```
    set(h,'Position', [1/3+gap 0 1/3-2*gap 1])
    imshow(uint8(img2));
    title(str2);

    h = subplot(1,3,3);
    set(h,'Position', [2/3+gap 0 1/3-2*gap 1])
    imshow(uint8(img3));
    title(str3);


    set(gcf,'Position',[0 0 900 300])
end
```

Filtro de média Nesta subseção serão analisados os resultados para o filtro de média.

[ ]:
```
tripleImages(img, simulada_whiteNoise, simulada_whiteNoise_mean, ...
             'Imagem com ruido branco', ...
             'Filtragem com filtro de media no ruido branco');
```

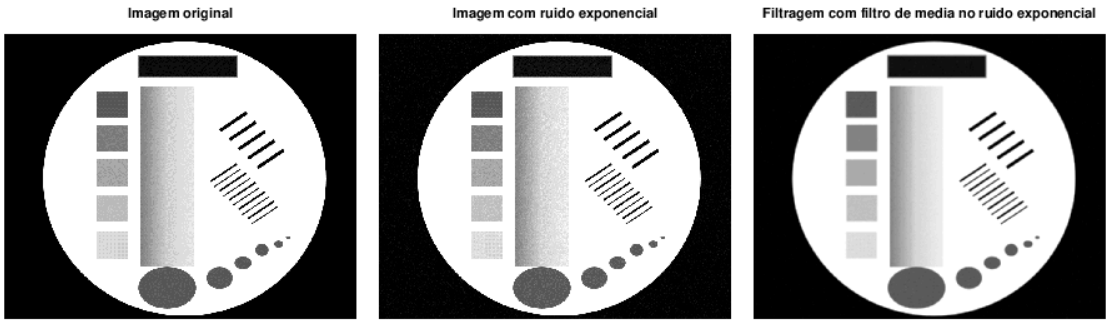


[ ]:
```
tripleImages(img, simulada_saltnpepperNoise, simulada_saltnpepperNoise_mean, ...
             'Imagem com ruido sal e pimenta', ...
             'Filtragem com filtro de media no ruido sal e pimenta');
```



```
[ ]: tripleImages(img, simulada_gaussianNoise, simulada_gaussianNoise_mean, ...
                  'Imagem com ruido Gaussiano', ...
                  'Filtragem com filtro de media no ruido Gaussiano');
```


```
[ ]: tripleImages(img, simulada_exponencialNoise, simulada_exponencialNoise_mean, ...
                  'Imagem com ruido exponencial', ...
                  'Filtragem com filtro de media no ruido exponencial');
```



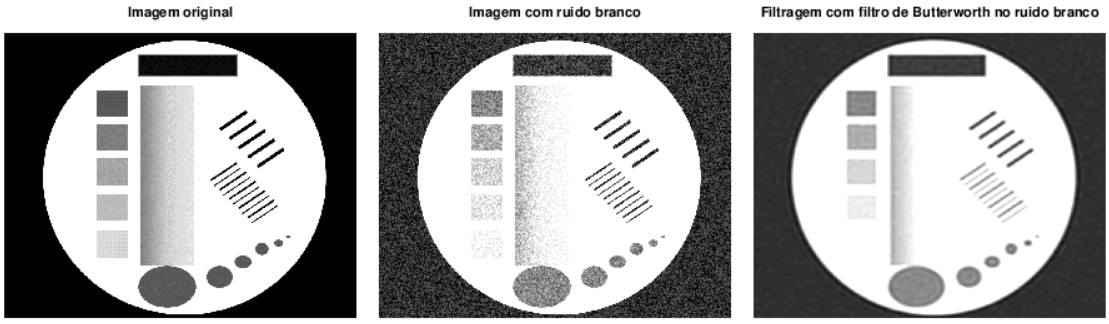
Dos resultados com o filtro de média 5x5, observa-se o melhor resultado ao lidar com o ruído exponencial, porém restando pouca distorção em objetos menores.

Em relação ao ruído Gaussiano e branco, o filtro foi capaz de restaurar a imagem próximo à sua condição original. No caso do ruído branco, apesar da imagem estar altamente contaminada, fazendo com que o fundo preto assuma tons de cinza, o filtro foi capaz de proporcionar o borramento necessário para eliminar o ruído de alta frequência, porém sem restaurar a cor original.

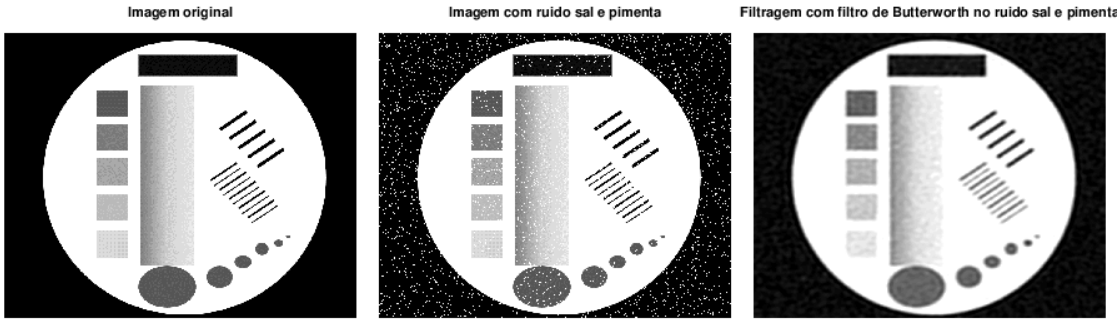
Por fim, em relação ao ruído sal e pimenta, o filtro não se mostrou tão eficaz e fez com que o ruído se espalhasse mais pela imagem devido ao borramento.

Filtro Butterworth passa-baixas Nesta subseção serão analisados os resultados para o filtro Butterworth passa-baixas.

```
[ ]: tripleImages(img, simulada_whiteNoise, simulada_whiteNoise_butterworth, ...
                  'Imagem com ruido branco', ...
                  'Filtragem com filtro de Butterworth no ruido branco');
```

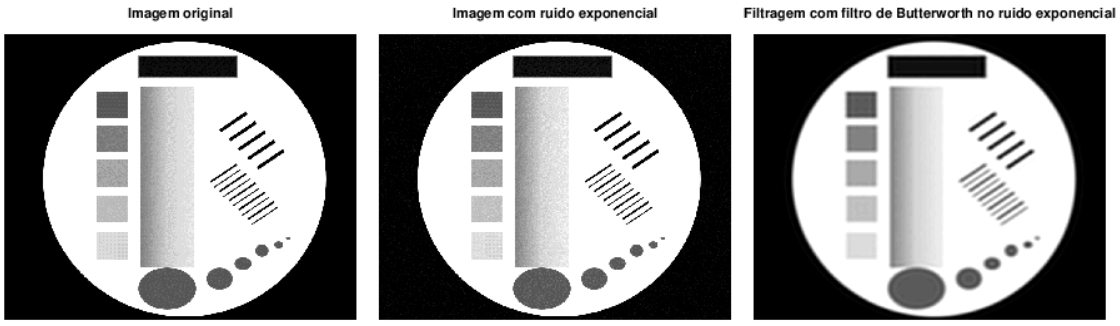



```
[ ]: tripleImages(img, simulada_saltnpepperNoise,␣
     ↪simulada_saltnpepperNoise_butterworth, ...
                  'Imagem com ruido sal e pimenta', ...
                  'Filtragem com filtro de Butterworth no ruido sal e pimenta');
```




```
[ ]: tripleImages(img, simulada_gaussianNoise, simulada_gaussianNoise_butterworth, ..
     ↪.
                  'Imagem com ruido Gaussiano', ...
                  'Filtragem com filtro de Butterworth no ruido Gaussiano');
```

[ ]:
```
tripleImages(img, simulada_exponencialNoise,␣
↪simulada_exponencialNoise_butterworth, ...
            'Imagem com ruido exponencial', ...
            'Filtragem com filtro de Butterworth no ruido exponencial');
```



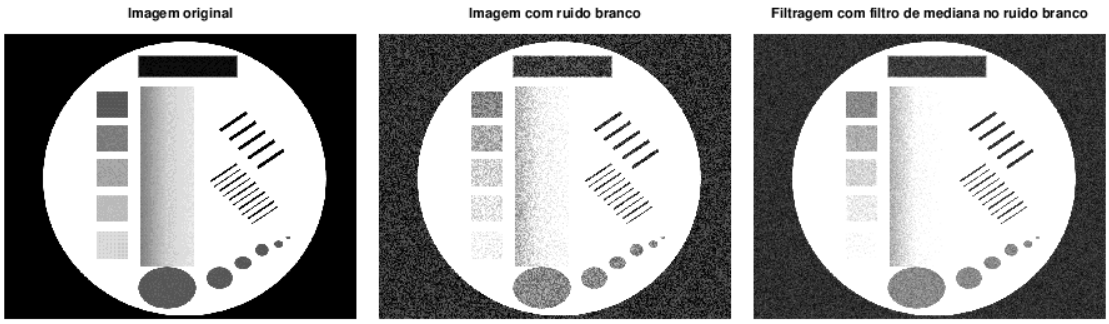
Assim como para o filtro anterior, o filtro Butterworth se mostrou bastante eficaz em relação ao ruído exponencial. No ruído Gaussiano a sua performance foi um pouco degradada, considerando que seu resultado na transição entre branco e cinza apresenta-se um pouco ruidoso, mas no geral apresenta um bom borramento que eliminou maior parte do ruído.

Em relação ao ruído branco, este também possui comportamento semelhante ao filtro de média, porém seu borramento foi ainda maior e apresenta um leve efeito de *ringing* na transição abruta entre branco e preto.

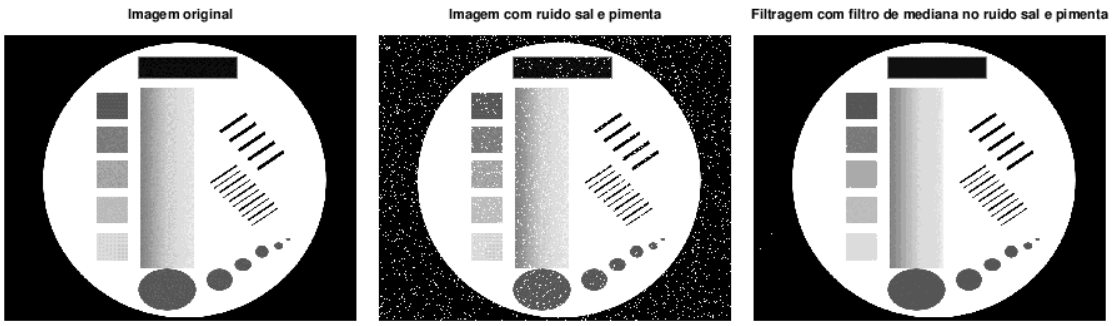
Por fim, em relação ao ruído sal e pimenta, o filtro Butterworth apesar de não possuir grande eficácia, esse ainda se mostrou melhor que o filtro de média, lidando melhor com a eliminação do ruído com o borramento, porém não suficiente.

Filtro de mediana Nesta subseção serão analisados os resultados para o filtro de mediana

[ ]:
```
tripleImages(img, simulada_whiteNoise, simulada_whiteNoise_median, ...
             'Imagem com ruido branco', ...
             'Filtragem com filtro de mediana no ruido branco');
```

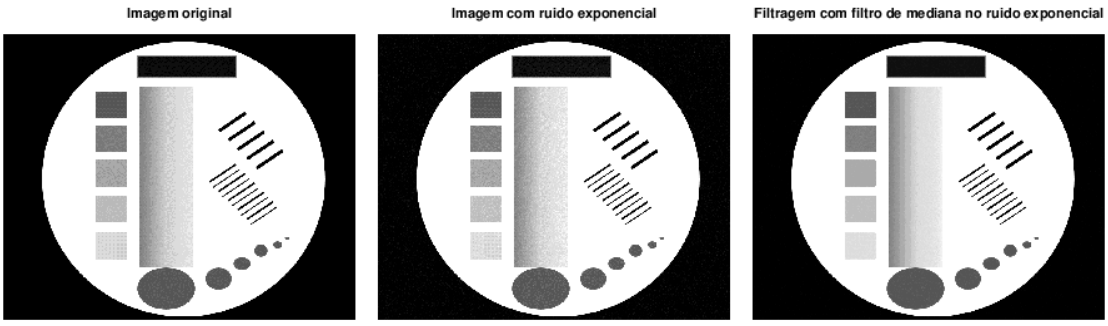


[ ]:
```
tripleImages(img, simulada_saltnpepperNoise, simulada_saltnpepperNoise_median, .
 ↪..
             'Imagem com ruido sal e pimenta', ...
             'Filtragem com filtro de mediana no ruido sal e pimenta');
```



[ ]:
```
tripleImages(img, simulada_gaussianNoise, simulada_gaussianNoise_median, ...
             'Imagem com ruido Gaussiano', ...
             'Filtragem com filtro de mediana no ruido Gaussiano');
```

[ ]:
```
tripleImages(img, simulada_exponencialNoise, simulada_exponencialNoise_median, ...
             'Imagem com ruido exponencial', ...
             'Filtragem com filtro de mediana no ruido exponencial');
```



Como esperado do filtro de mediana, este foi o que lidou melhor o ruído sal e pimenta. Seu desempenho também foi excelente para o ruído exponencial. No caso do ruído Gaussiano a performance começou a diminuir e não conseguiu eliminar completamente o ruído, ainda persistindo aqueles de maiores frequências.

O pior desempenho desse filtro foi em relação ao ruído branco que quase não surtiu efeito para eliminar o ruído, permanecendo, ainda, uma imagem altamente contaminada.

1.2 Filtro no domínio do espaço vs. domínio da frequência

Esta seção validará quatro diferentes filtros aplicados a três diferentes imagens que serão carregadas de acordo com a necessidade de processamento. Desses quatro, dois serão no domínio espacial e os demais no domínio da frequência.

A escolha para os filtros no domínio do espaço será um filtro de média e um filtro de mediana. Para os filtros no domínio da frequência, será usado um filtro passa baixas ideal de frequência de corte de e um filtro passa baixas Butterworth.

As dimensões, frequência de corte e ordem dos filtros serão definidas de acordo com as imagens de análise ao decorrer do processamento. A menos do filtro de mediana que será implementado por meio da função `medfilt2` do Octave, os demais filtros podem ser inicializados com o conjunto de funções definidas abaixo.

[ ]:
```
% filtro de media
function filter = mean_filter(order)
    filter = ones(order,order)/(order*order)
end
```

[ ]:
```
% filtro passa baixas ideal
function filter = ideal_LP_filter(img, f_cutoff)
    [M,N] = size(img);
    filter = zeros(M,N);
    cx = floor(M/2)+1;
    cy = floor(N/2)+1;
    for x= 1:M
        for y= 1:N
            D=sqrt(power((x-cx),2)+power((y-cy),2));
            if (D < f_cutoff)
                filter(x,y)=1;
            end
        end
    end
    imshow(filter)
end
```

[ ]:
```
% filtro passa baixas Butterworth
function filter = butterworth_LP_filter(img, f_cutoff, order)
    [M,N] = size(img);
    filter = zeros(M,N);
    cx = floor(M/2)+1;
    cy = floor(N/2)+1;
    for x= 1:M
        for y= 1:N
            D=sqrt(power((x-cx),2)+power((y-cy),2));
            filter(x,y)=1/(1+power(D/f_cutoff,2*order));
        end
    end
    imshow(filter)
end
```

A função abaixo auxiliará na visualização:

[ ]:
```
function doubleImages(img1, img2, str, size)
    gap = 0.01;

    h = subplot(1,2,1);
```

```
    set(h,'Position', [0+gap 0 1/2-2*gap 1])
    imshow(uint8(img1));
    title('Imagem original');

    h = subplot(1,2,2);
    set(h,'Position', [1/2+gap 0 1/2-2*gap 1])
    imshow(uint8(img2));
    title(str);

    set(gcf,'Position',[0 0 size(1) size(2)])
end
```

E abaixo auxiliará na filtragem no domínio da frequência:

[ ]:
```
function result = imfilter_freq(img, filter)
    img = fft2(img); % Fourier transform
    img = fftshift(img); % image frequency spectrum centralized
    img = ifftshift(img.*filter); % apply filter
    result = uint8(real(ifft2(img))); % covnert imag eback to space domain
end
```

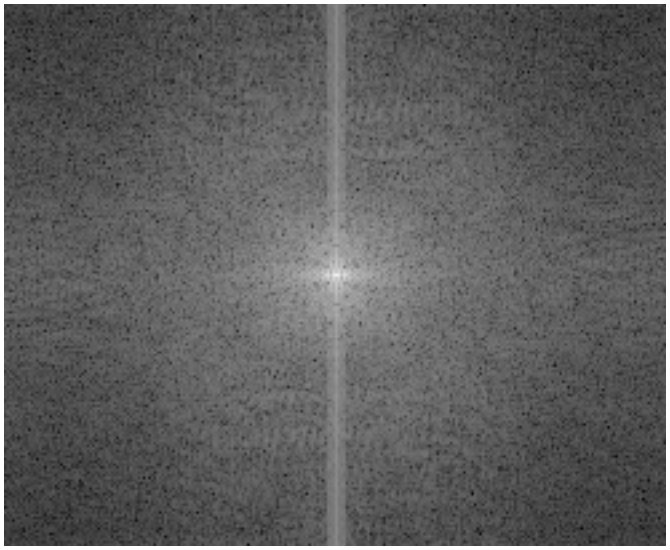
1.2.1 Imagem 1

A primeira imagem é carregada abaixo:

[ ]:
```
img1 = double(imread('Raios X 6_2.jpg'));
imshow(img1)
```

No domínio da frequência:

[ ]:
```
img1f = fftshift(fft2(img1));
imshow(log(1+abs(img1f)))
```



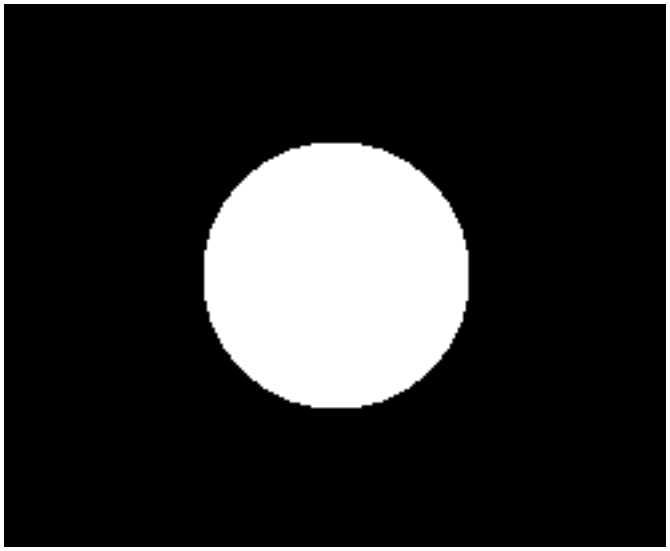
Para essa imagem serão implementados os filtros com as seugintes características:

- Filtro de média 3x3;
- Filtro de mediana 4x4;
- Filtro passa baixas ideal com frequência de corte de 50 pixels;
- Filtro passa baixas ideal com frequência de corte de 50 pixels e ordem 5;

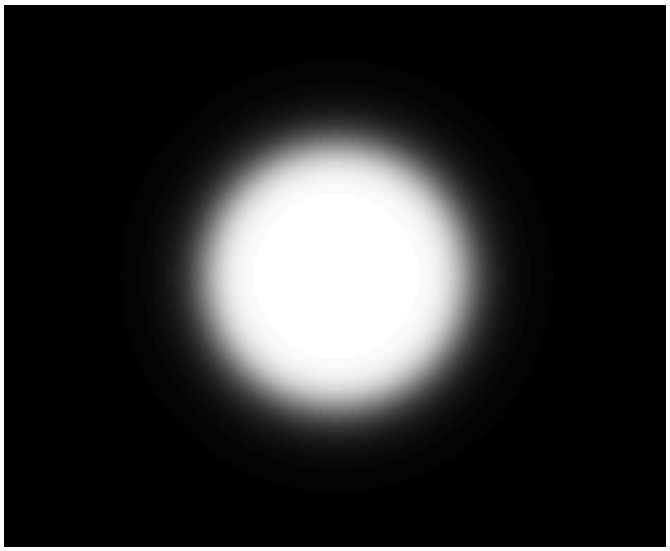
[ ]:
```
filt_mean = mean_filter(3);
```

```
filter =

   0.11111   0.11111   0.11111
   0.11111   0.11111   0.11111
   0.11111   0.11111   0.11111
```

[ ]:
```
filt_lp = ideal_LP_filter(img1, 50);
```

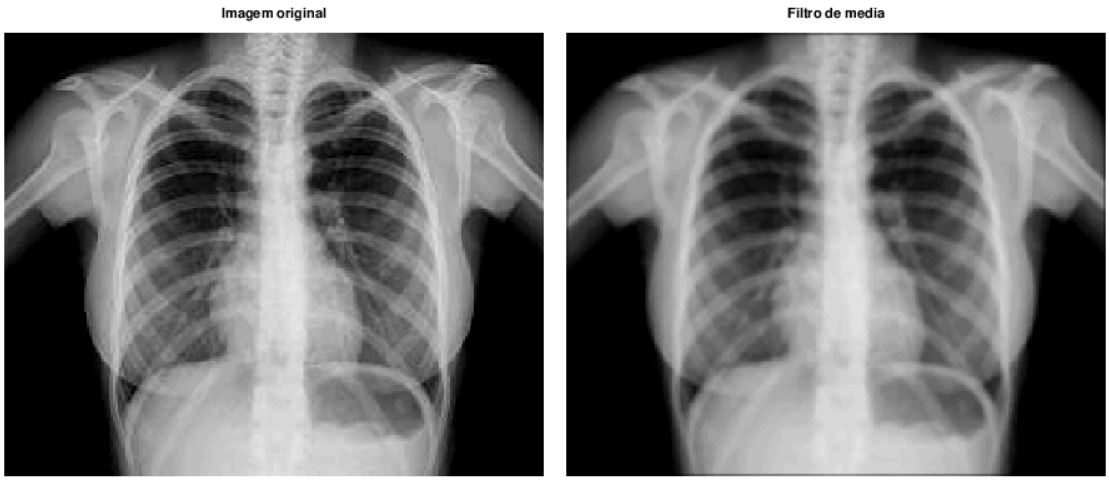


[ ]:
```
filt_butter = butterworth_LP_filter(img1, 50, 5);
```

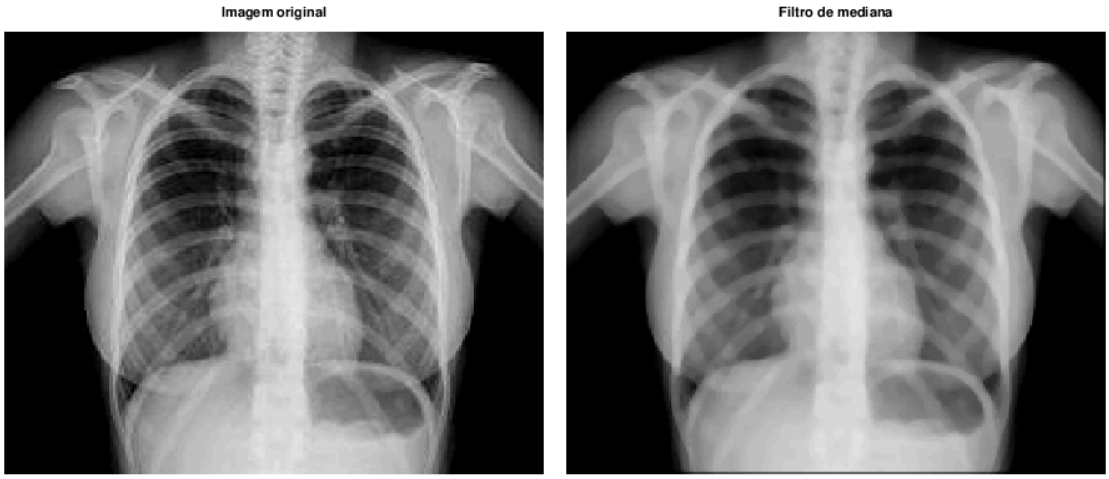


Ao implementar, obtém-se as figuras abaixo.

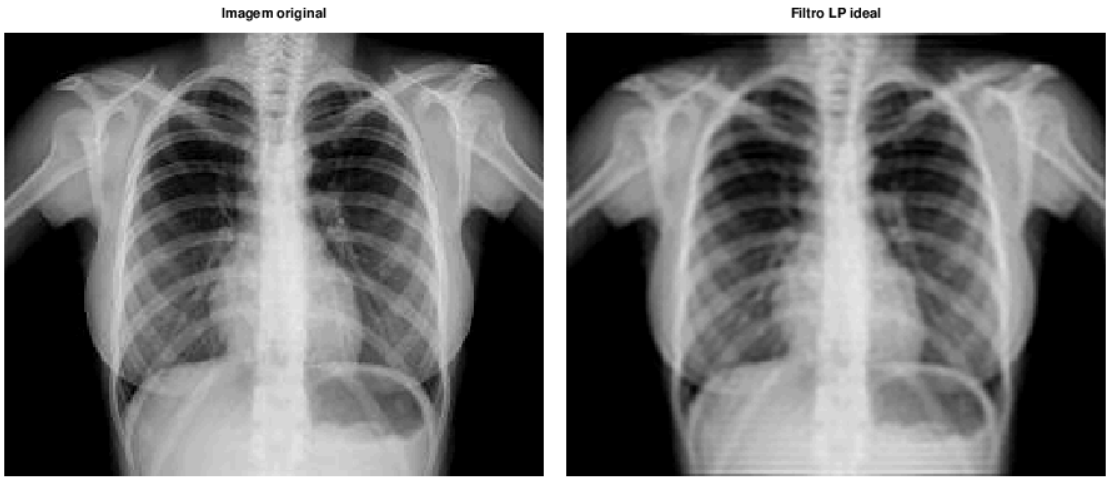
[ ]:
```
doubleImages(img1, imfilter(img1, filt_mean), 'Filtro de media', [900 400])
```



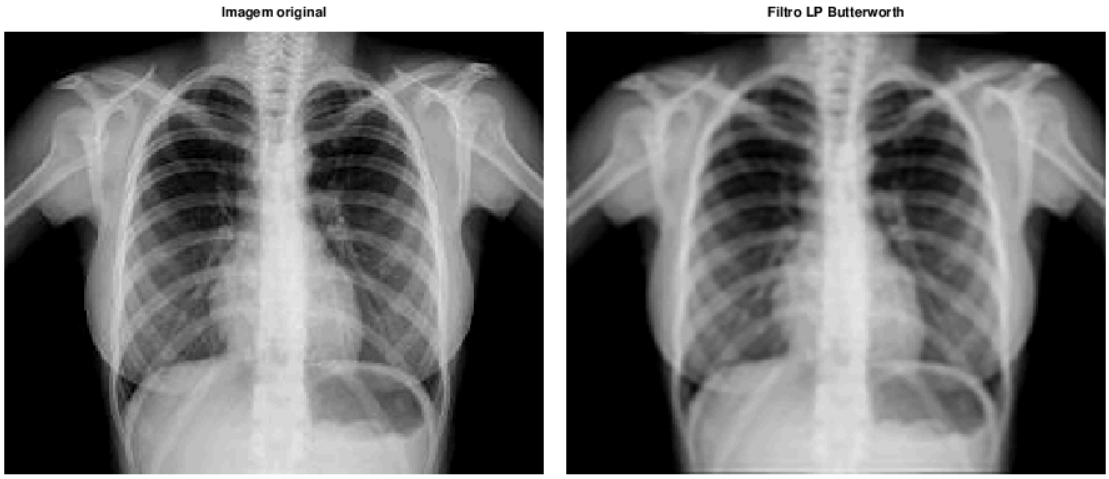
[ ]:
```
doubleImages(img1, medfilt2(img1, [4 4]), 'Filtro de mediana', [900 400])
```



[ ]:
```
doubleImages(img1, imfilter_freq(img1, filt_lp), 'Filtro LP ideal', [900 400])
```




```
[ ]: doubleImages(img1, imfilter_freq(img1, filt_butter), 'Filtro LP Butterworth',␣
     ↪[900 400])
```



Comentando sobre os resultados, observa-se que o melhor resultado foi aquele do filtro Butterworth.

O resultado do filtro Butterworth mostrou-se até semelhante com o filtro de média se comaprar o nível de borramento, porém mostrou um resultado melhor em termos de manter o contraste da imagem. Isso pode ser observado ao notar os ossos nos ombros.

O filtro de mediana causou um borramento aquém do necessário, tornando algumas áreas da imagem opaca, assim impossibilitando a identificação de diferentes estruturas no raio X.

Por outro lado o filtro passa baixas ideal mostrou um nível de boarramento razoável, porém o efeito de *ringing* prejudicou uma melhor visualização da imagem.

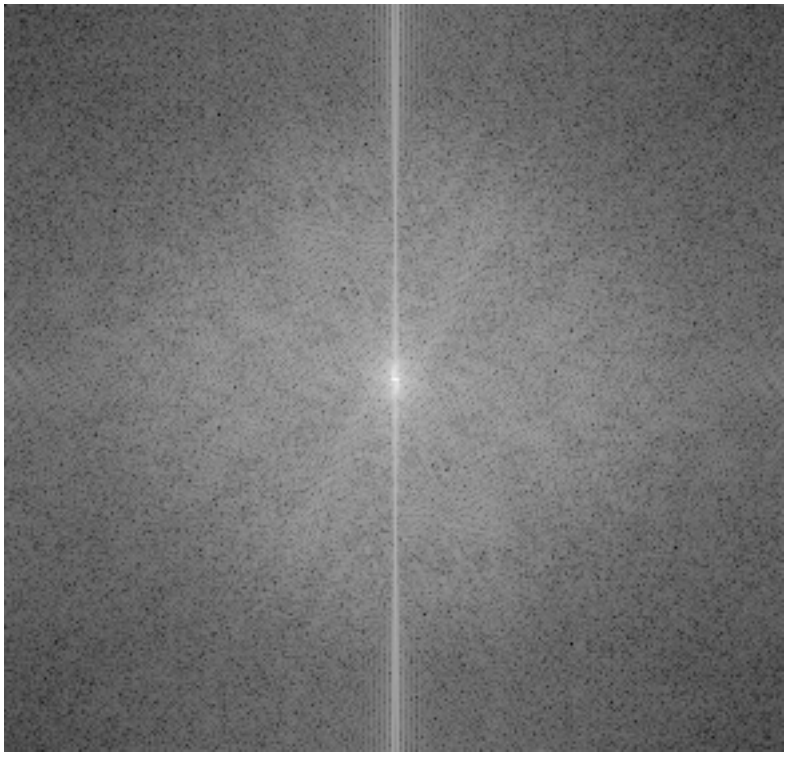
1.2.2 Imagem 2

A segunda imagem é carregada abaixo:

[ ]:
```
img2 = double(imread('raios X 7_1.jpg'));
imshow(img2)
```

No domínio da frequência:

[ ]:
```
img2f = fftshift(fft2(img2));
imshow(log(1+abs(img2f)))
```



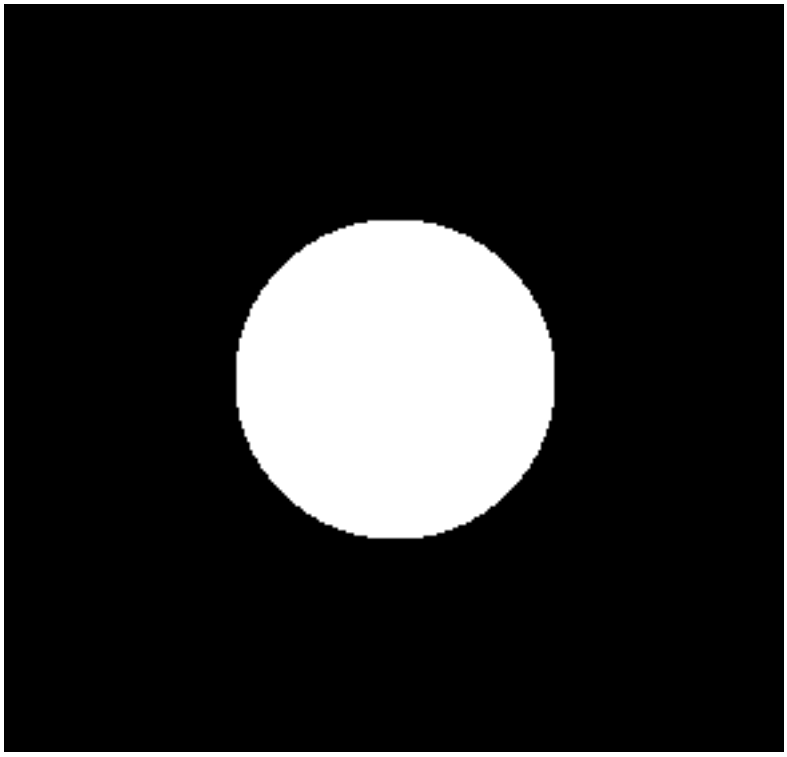
Para essa imagem serão implementados os filtros com as seugintes características:

- Filtro de média 3x3;
- Filtro de mediana 3x3;
- Filtro passa baixas ideal com frequência de corte de 60 pixels;
- Filtro passa baixas ideal com frequência de corte de 60 pixels e ordem 5;

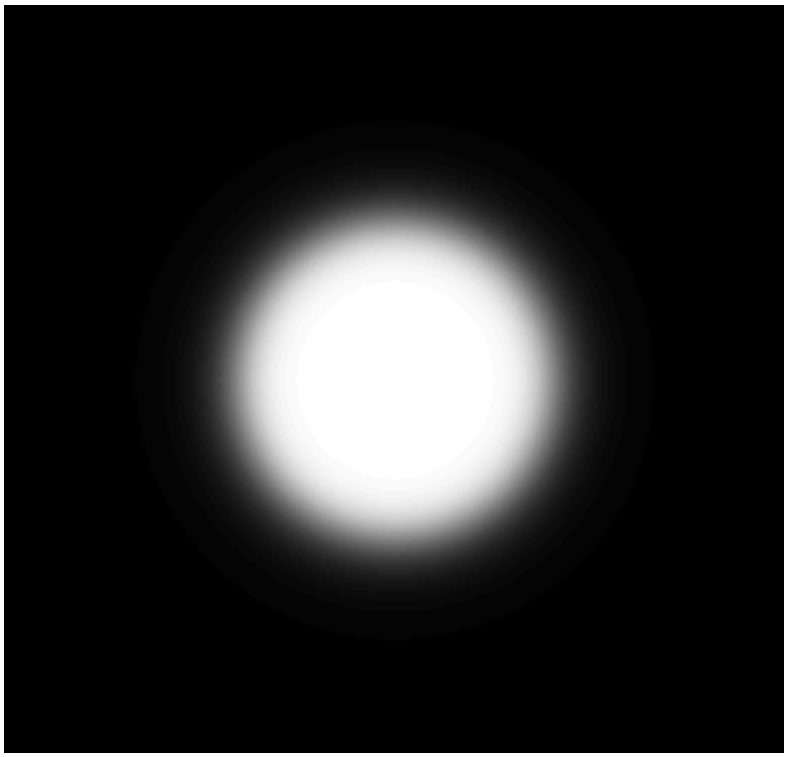
[ ]:
```
filt_mean = mean_filter(3);
```

```
filter =

   0.11111   0.11111   0.11111
   0.11111   0.11111   0.11111
   0.11111   0.11111   0.11111
```

[ ]:
```
filt_lp = ideal_LP_filter(img2, 60);
```

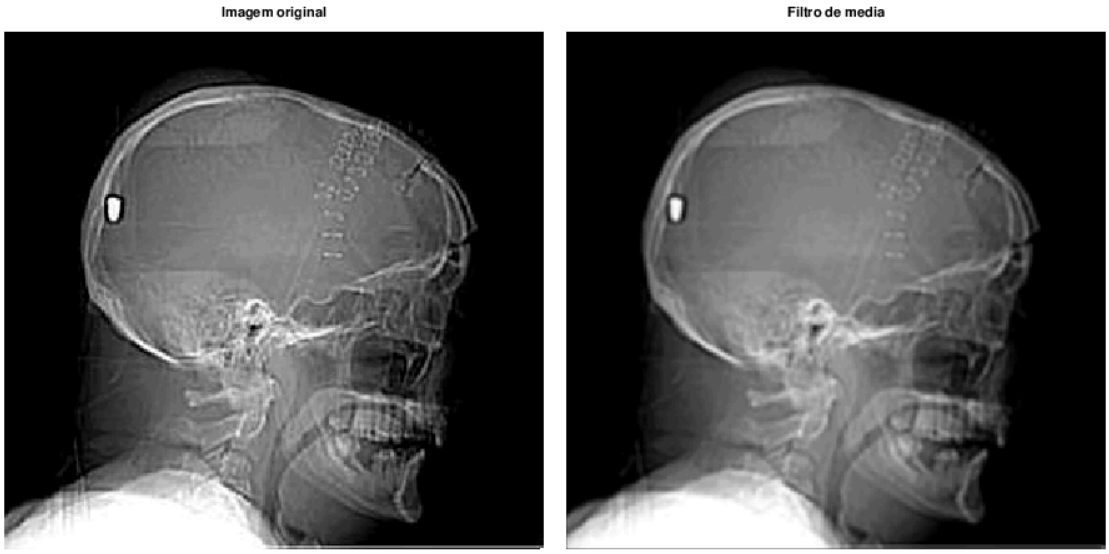


[ ]:
```
filt_butter = butterworth_LP_filter(img2, 60, 5);
```

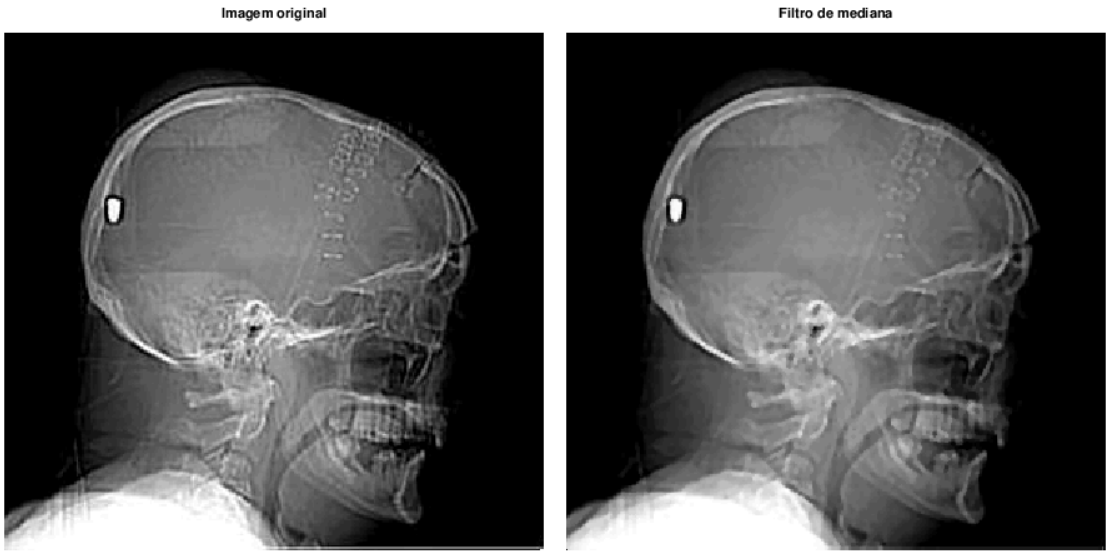


Ao implementar, obtém-se as figuras abaixo.

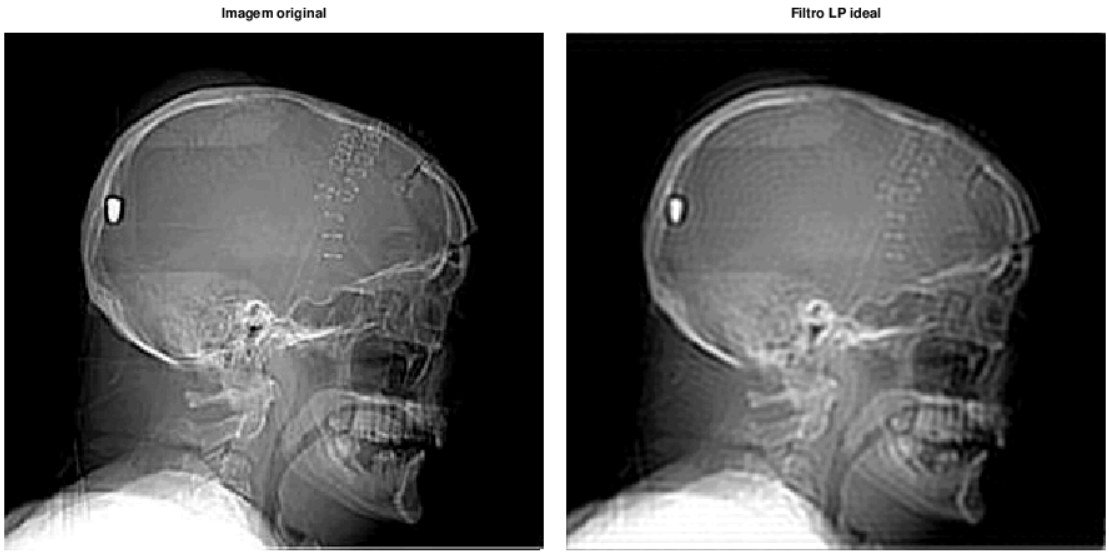
```
[ ]: doubleImages(img2, imfilter(img2, filt_mean), 'Filtro de media', [900 500])
```



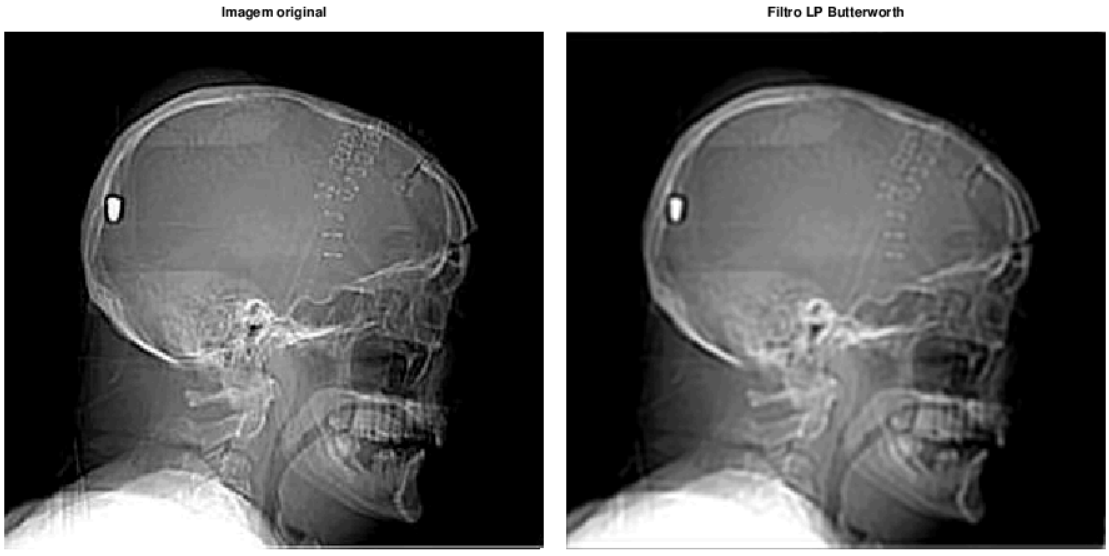

```
[ ]: doubleImages(img2, medfilt2(img2, [3 3]), 'Filtro de mediana', [900 500])
```



[ ]:
```
doubleImages(img2, imfilter_freq(img2, filt_lp), 'Filtro LP ideal', [900 500])
```



[ ]:
```
doubleImages(img2, imfilter_freq(img2, filt_butter), 'Filtro LP Butterworth',␣
↪[900 500])
```



Desta vez o filtro de mediana mostrou o melhor resultado. Diferente da outra imagem, esta possui maiores detalhes, fazendo com que as transiões abruptas gerem um espectro de frequência mais largo.

À respeito do filtro de mediana, neste caso seu efeito não foi tão destrutivo e ele até ajudou a eliminar o ruído impulsivo contido na região auricular. Ele ocasionou um leve aumento na opacidade devido ao borramento, fazendo com que regiões mais improtantes se destaquem ao olho.

O filtro de média e o Butterworth ainda apresentaram bons resultados. Suas imagens filtradas possuem bordas borradas idealmente, mas na região da face o efeito de borramento não foi tão desejado, eliminando leves características que o filtro de mediana manteve.

O filtro passa baixas ideal se mostrou o pior filtro. Além do borramento insuficiente, este ainda apresentou um alto nível de *ringing* que pode prejudicar a identificação das regiões intracranianas.

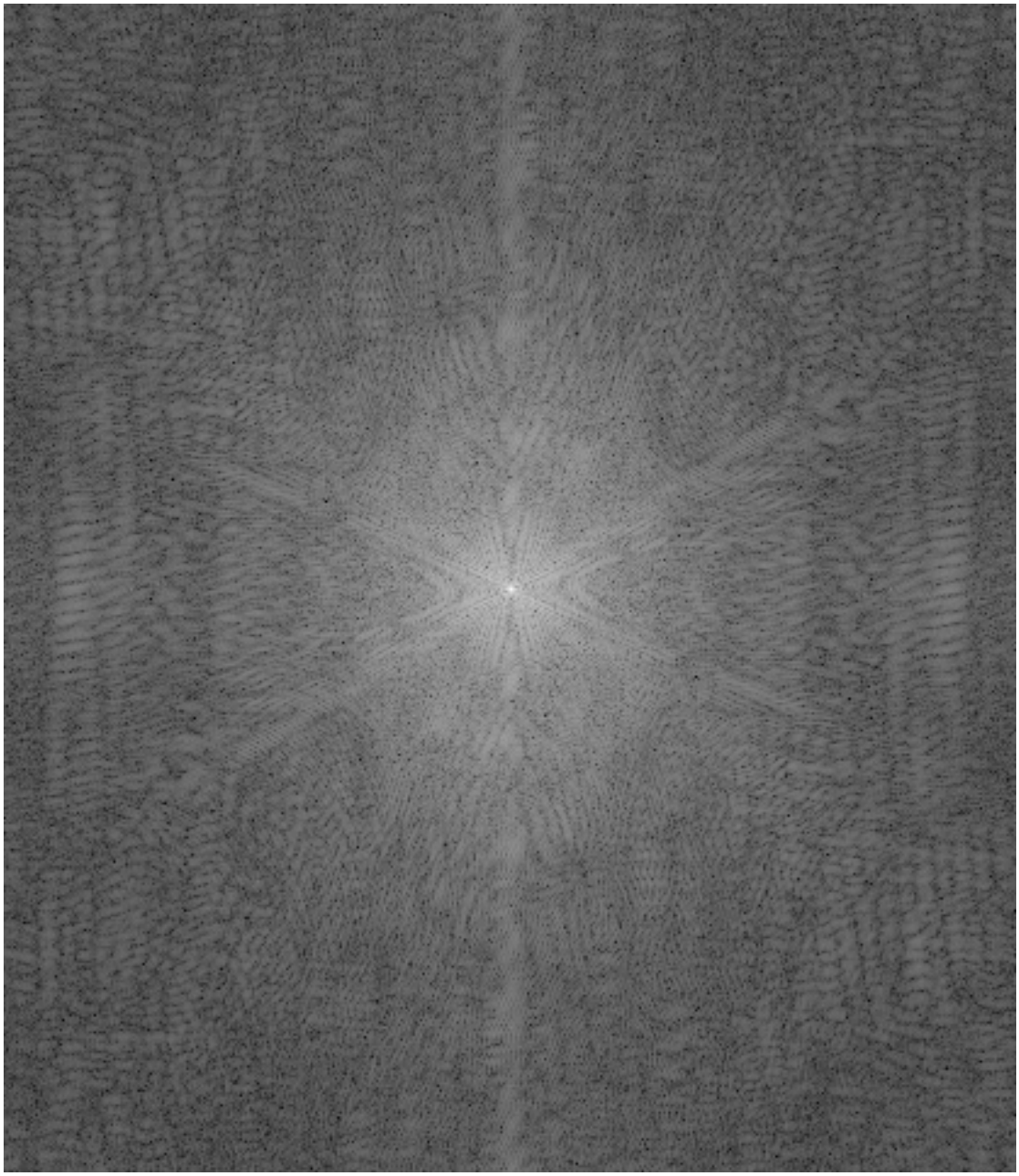
1.2.3 Imagem 3

A terceira e última imagem é carregada abaixo:

[ ]:
```
img3 = double(imread('Ressonancia 6_2.jpg'));
imshow(img3)
```

No domínio da frequência:

[ ]:
```
img3f = fftshift(fft2(img3));
imshow(log(1+abs(img3f)))
```



Para essa imagem serão implementados os filtros com as seugintes características:

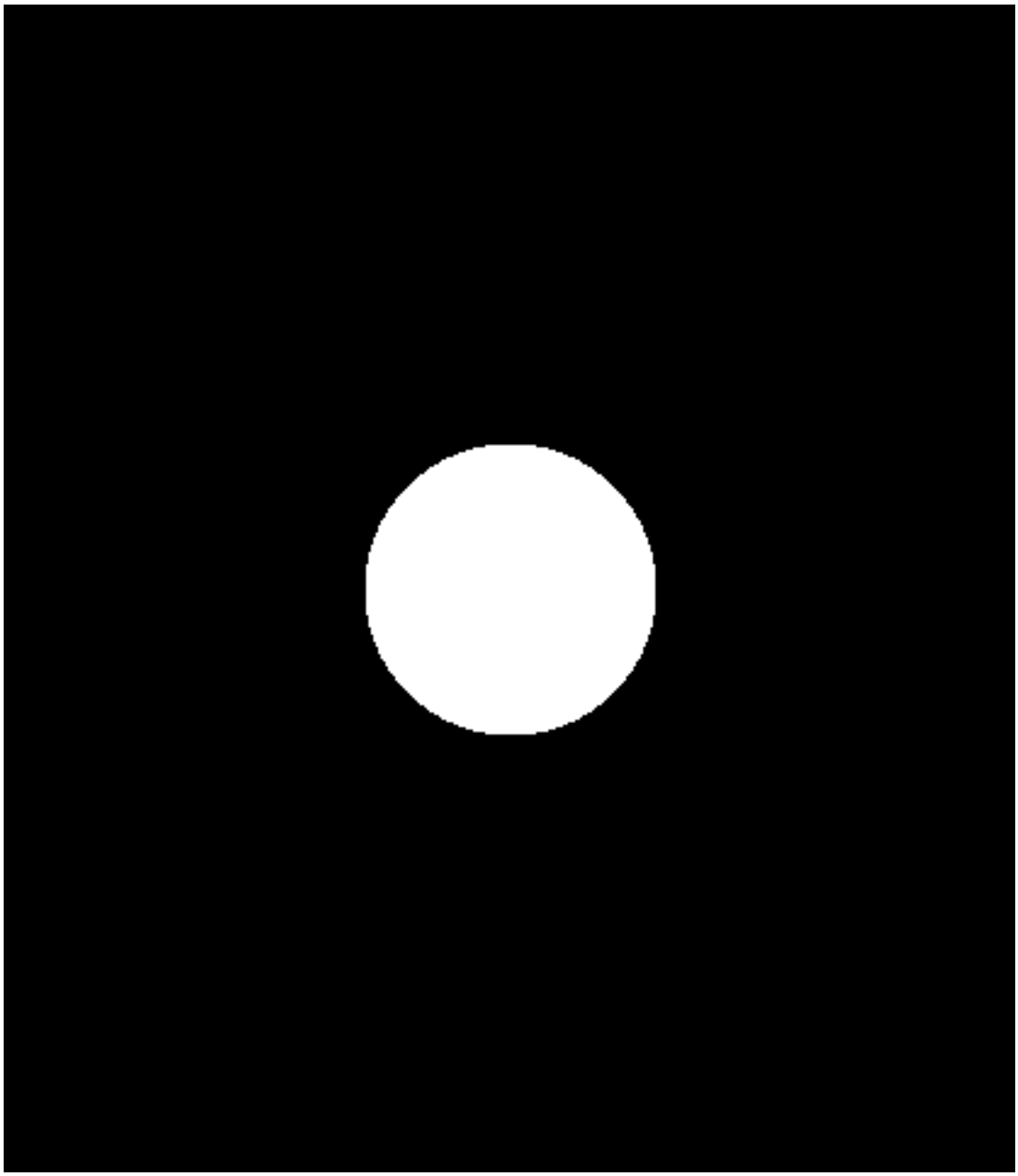
- Filtro de média 3x3;
- Filtro de mediana 3x3;
- Filtro passa baixas ideal com frequência de corte de 60 pixels;
- Filtro passa baixas ideal com frequência de corte de 65 pixels e ordem 10;

[ ]:
```
filt_mean = mean_filter(3);
```

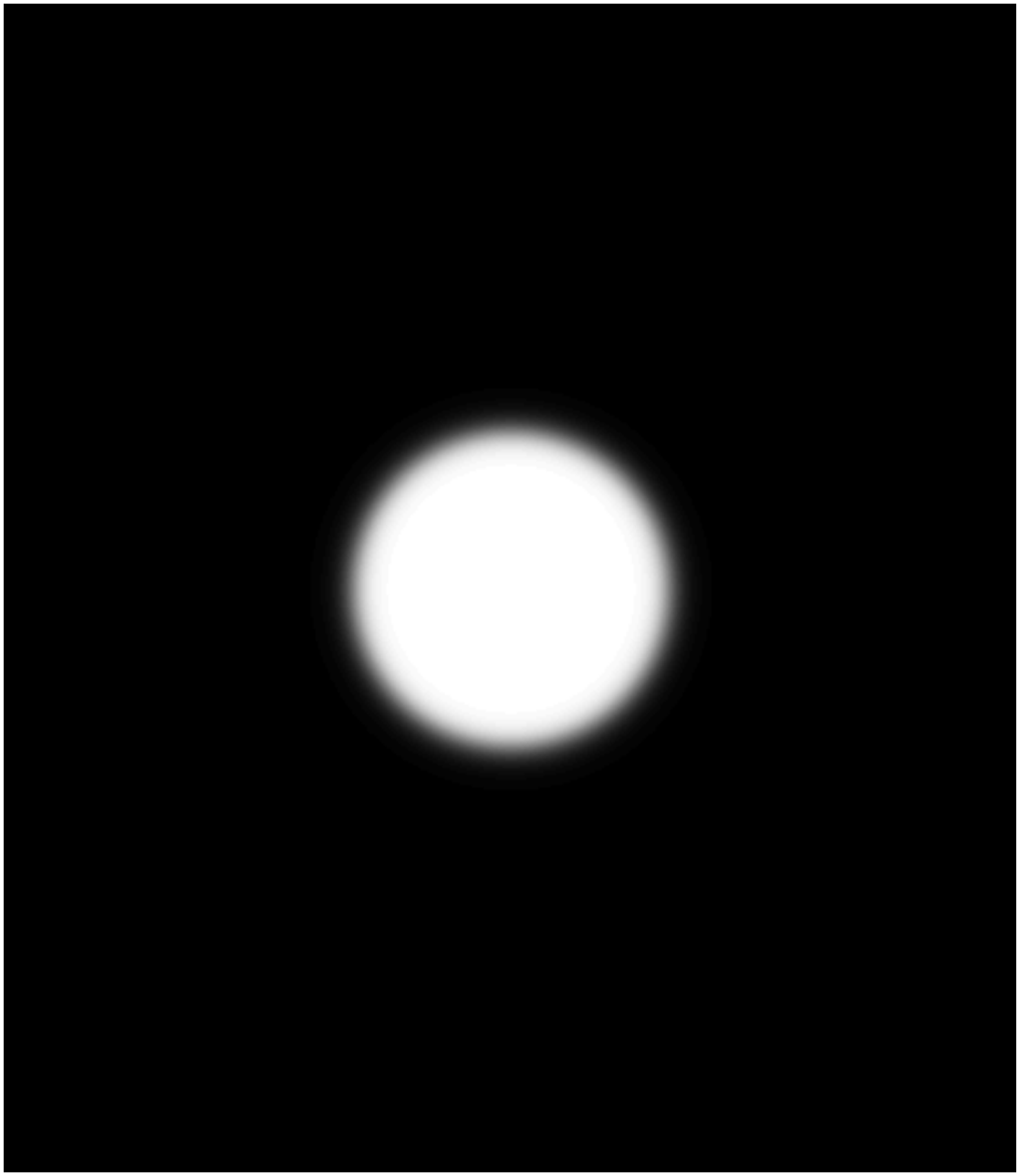
```
filter =
```

```
0.11111   0.11111   0.11111
0.11111   0.11111   0.11111
0.11111   0.11111   0.11111
```

[ ]:
```
filt_lp = ideal_LP_filter(img3, 60);
```

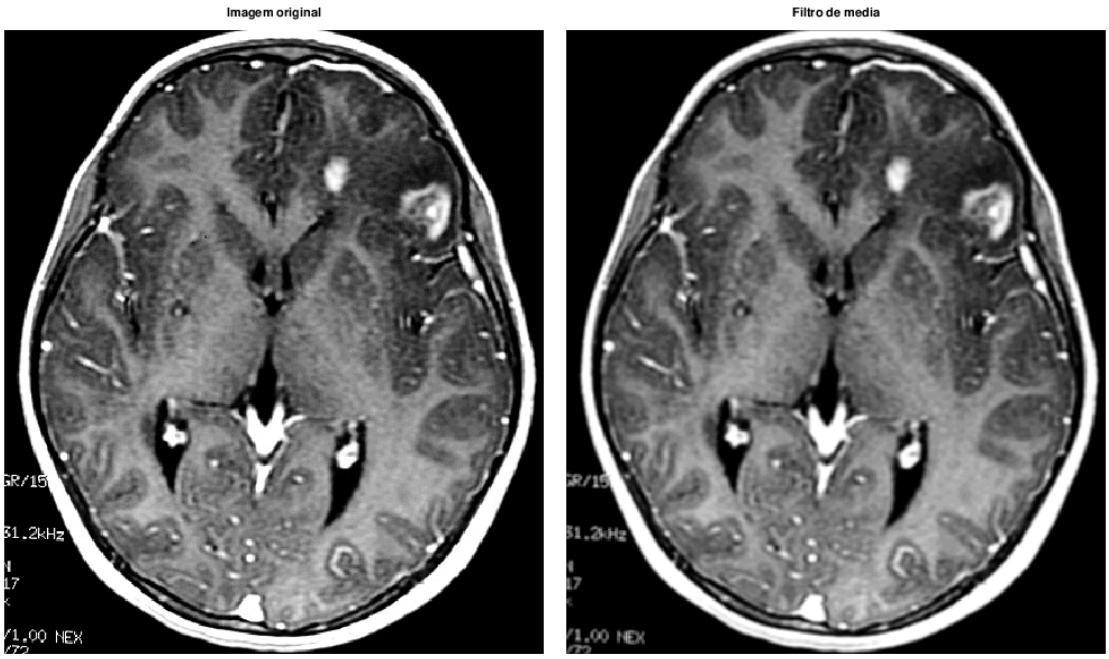


[ ]:
```
filt_butter = butterworth_LP_filter(img3, 65, 10);
```

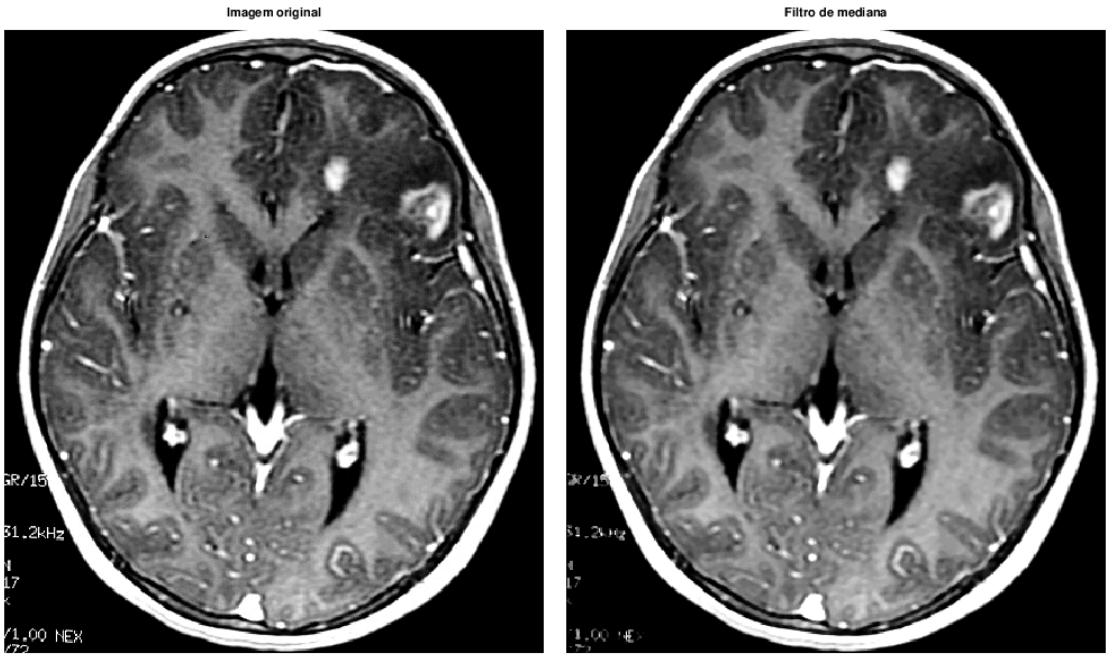


Ao implementar, obtém-se as figuras abaixo:

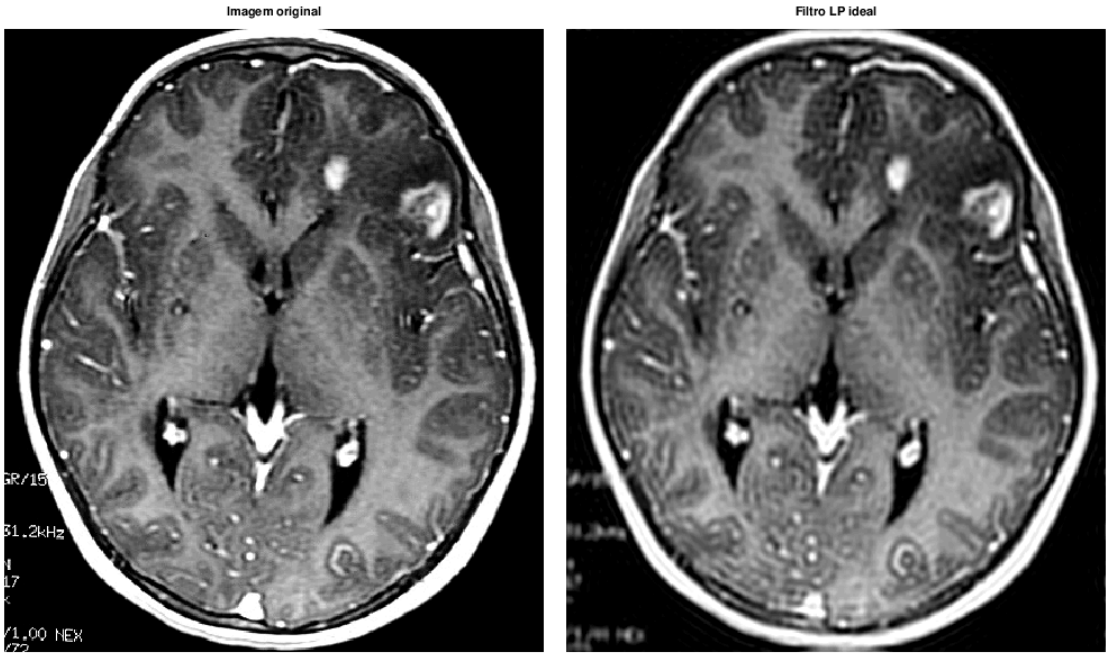
[ ]:
```
doubleImages(img3, imfilter(img3, filt_mean), 'Filtro de media', [1000 650])
```



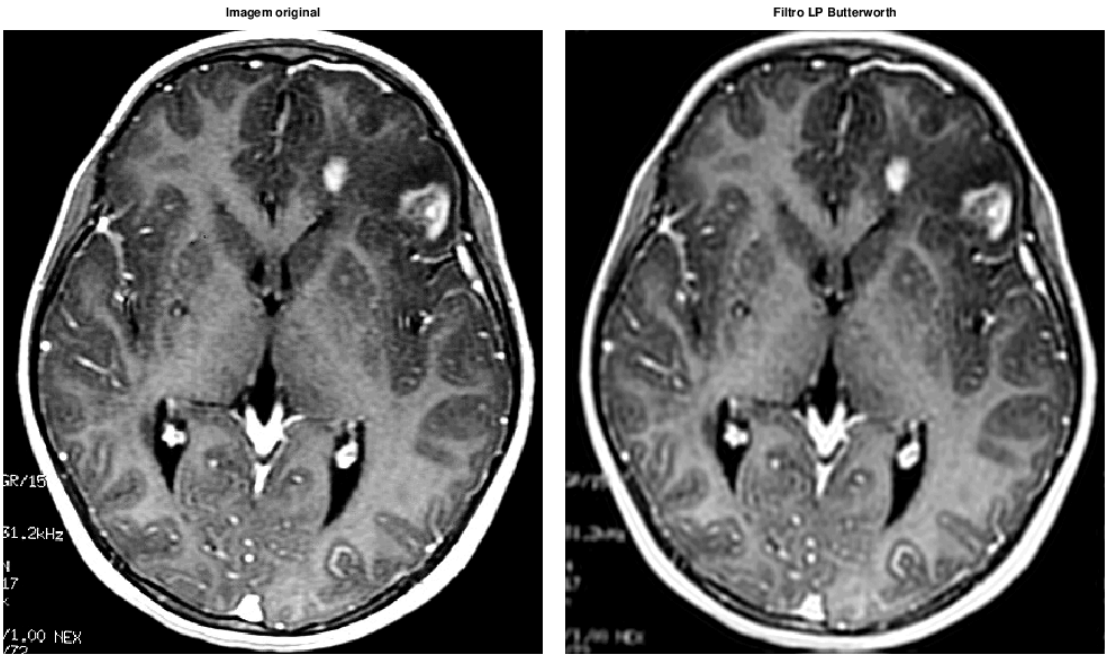
[ ]:
```
doubleImages(img3, medfilt2(img3, [3 3]), 'Filtro de mediana', [1000 650])
```



[ ]:
```
doubleImages(img3, imfilter_freq(img3, filt_lp), 'Filtro LP ideal', [1000 650])
```



[ ]:
```
doubleImages(img3, imfilter_freq(img3, filt_butter), 'Filtro LP Butterworth',␣
↪[1000 650])
```



Esta imagem possui um conteúdo ainda mais rico de detalhes, pois é uma ressonância do cérebro. Todo processamento deve ser cuidadoso para não perder reigões como as leves manchas brancas na parte inferior da imagem.

Um primeiro *disclaimer* que se estendeu ao longo das três imagens e que foi sendo provado é que o filtro passa baixas ideal possui o pior dos resultados. Seu efeito de *ringing* prejudica muito o produto final filtrado e no caso de imagens circulares, como essa, pode confundir as informações presentes na imagem.

Apesar dos esforços para sintonizar a frequência de corte e a ordem, o filtro Butterworth não obteve um bom resultado. Este apenas contribuiu para um borramento insatisfatório e que não eliminou maiores ruídos na região mais clara da imagem. Este também prejudicou a compreensão por completo do texto ao lado esquerdo da imagem.

O filtro de média possuiu resultados semelhantes ao Butterworth, mas não tão ruins quanto. O texto ao lado da imagem manteve-se levemente compreensível e o borramento proporcionado eliminou levemente o ruído contido na imagem, mas privilegiou eliminá-lo em regiões mais claras.

Por fim, o filtro de mediana foi o que possuiu o melhor resultado. Além de manter intacto os detalhes da imagem, este propocionou um bom borramento nas regiões livres da imagem, independente da intensidade.

Porém caso o foco seja observar as áreas livres da imagem, o filtro de média pode ser uma melhor opção.