```
PlotlyBackend()
```

Experiência 3

Gabriel Tavares - 10773801

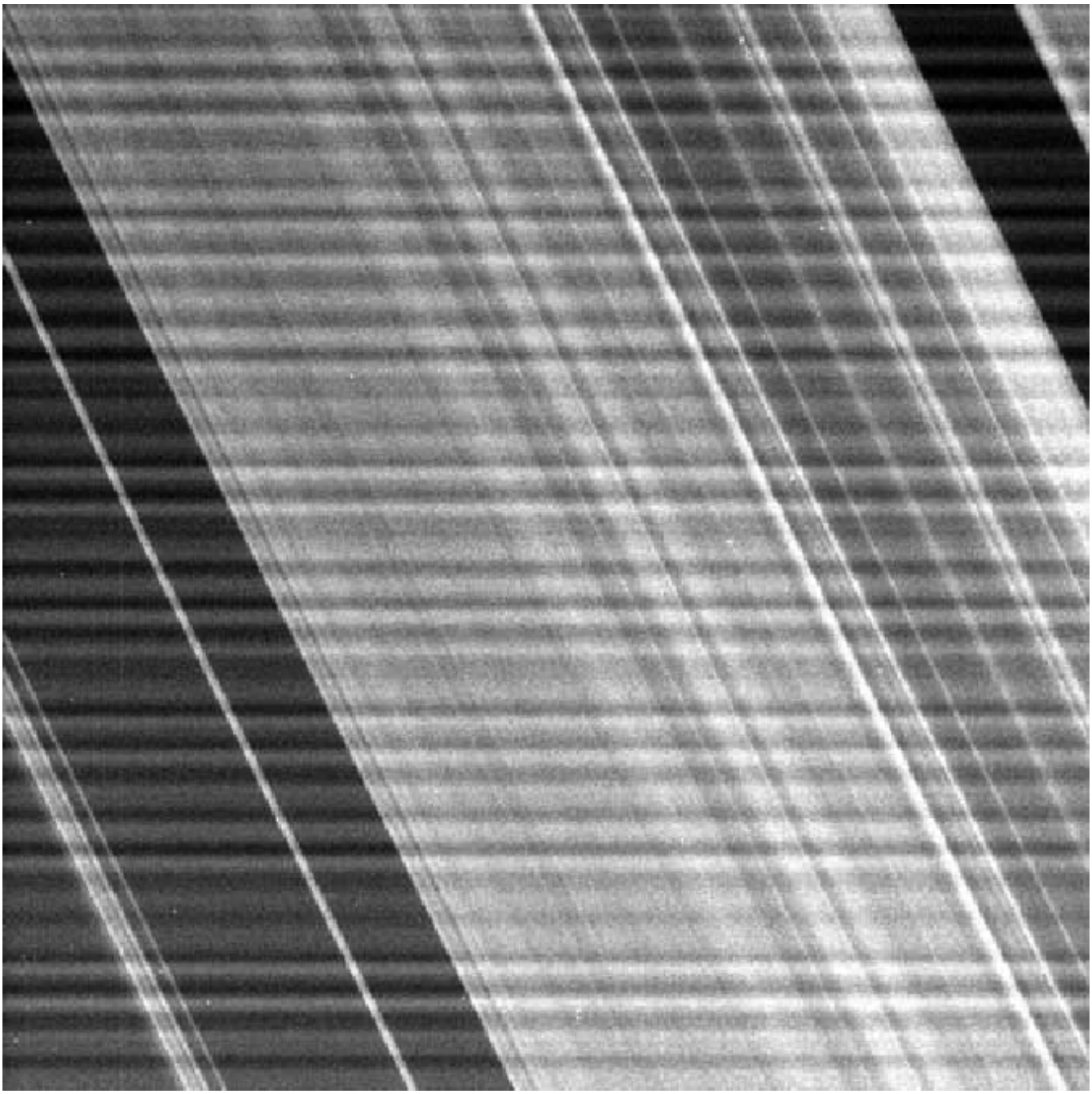
Guilherme Reis - 10773700

Diego Hidek - 10336622

Cassini-Interfence

Análise da imagem

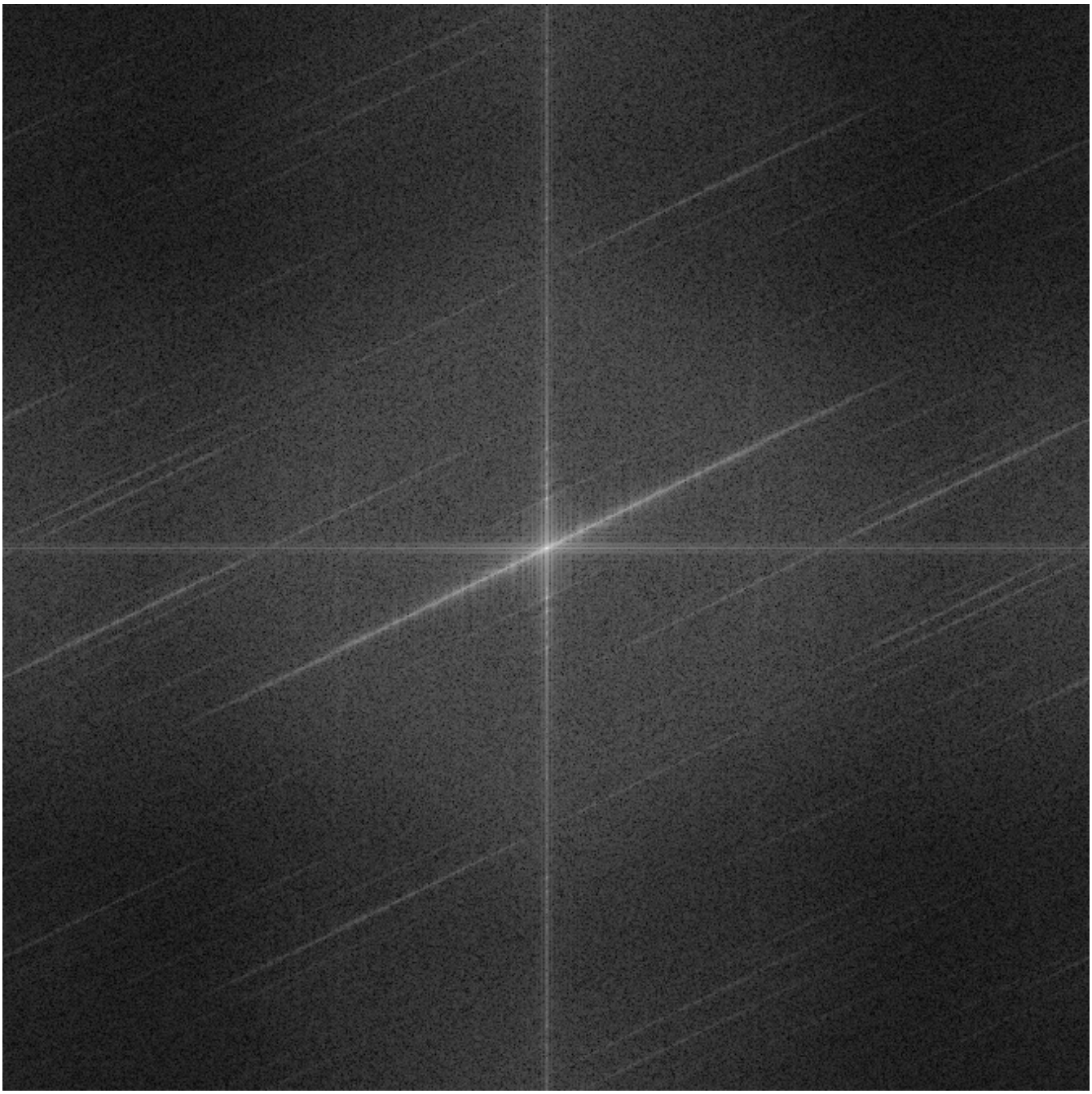
cassini =



```
cassini = load("cassini-interference.tif")
```

```
begin
    cassini_f = Float64.(cassini)
    CASSINI = fftshift(fft(cassini_f))
    md""
end
```

espectro_cassini =



```
espectro_cassini = Gray.(log.(abs.(CASSINI) .+ 1) ./ maximum(log.(abs.(CASSINI) .+1)))
```

Analizando a imagem, vemos que a interferência ocorre em sinais faixas horizonteis na imagem. Portanto podemos concluir que o ruído é gerado por uma interferência na vertical no espectro.

Se olharmos com atenção, também é possível ver que o espectro tem a linha central vertical um pouco mais clara do que a horizontal, o que indica que o ruído pode vir dali

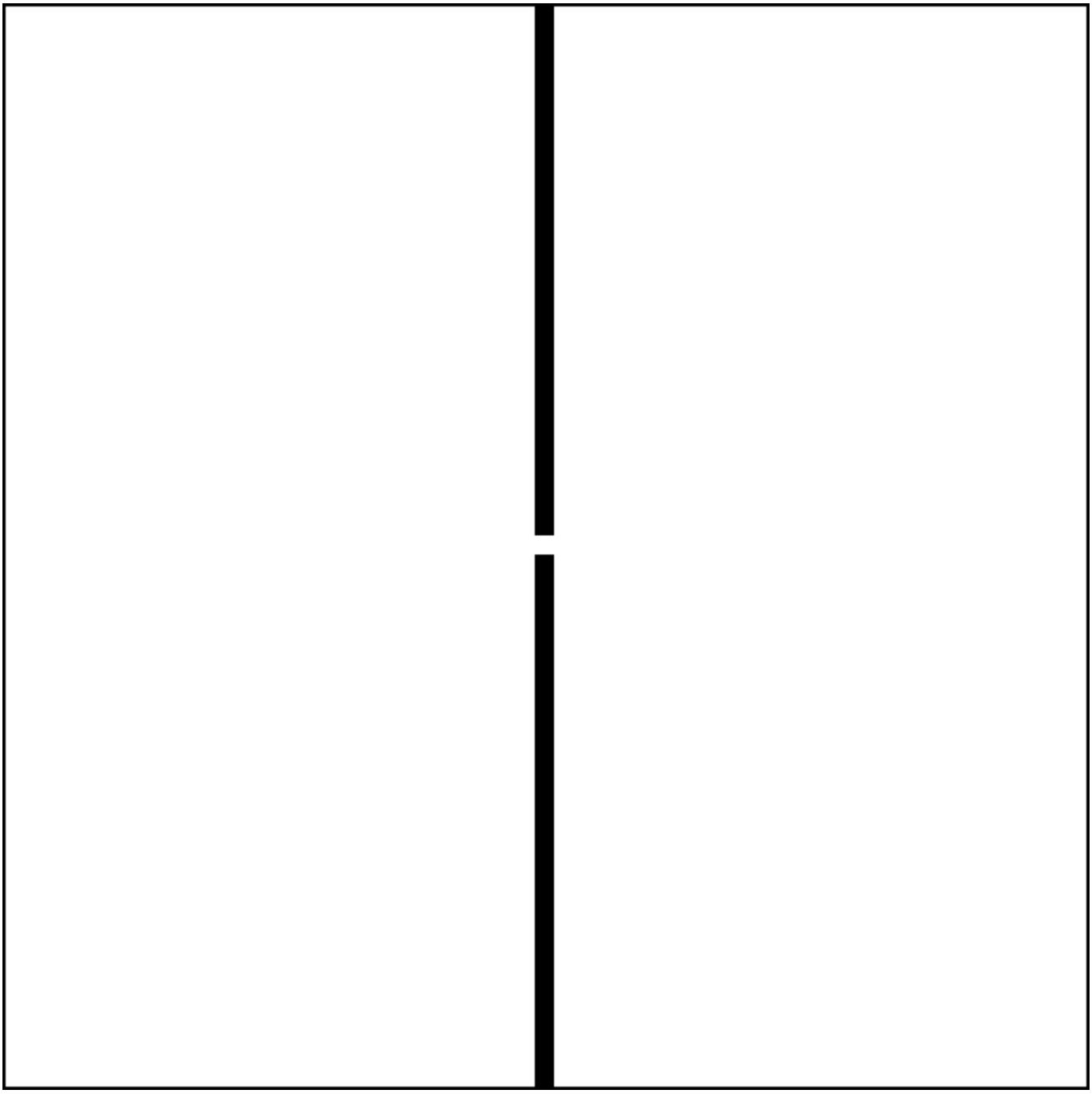
Criação do filtro

Inicialmente iremos criar um filtro que elimine essa faixa central de interferência mas mantenha o nível DC da imagem

```
begin
    nlin = size(CASSINI)[1] #674
    ncol = size(CASSINI)[2] #674

    centro = (Int64)(ncol/2) #337
    larg = 5

    H10 = ones(nlin,ncol)
    H10[:,centro-larg-1:centro+larg].=0 #zera colunas 333:342
    H10[centro-larg-1:centro+larg,:] .=1 #recupera o centro DC da imagem
    md""
end
```

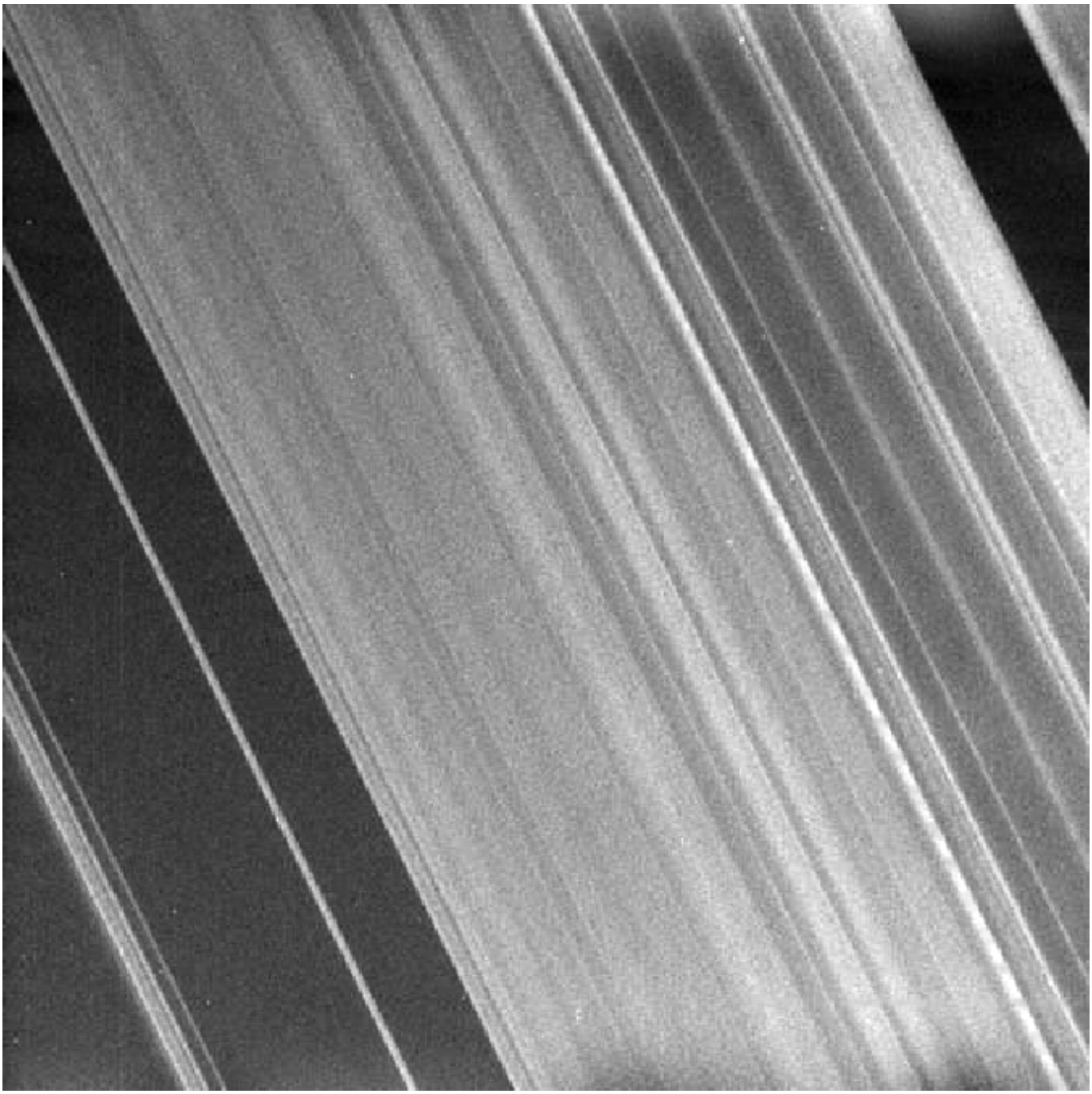


```
make_border(Gray.(real.(H10)),2)
```

Filtrando a imagem

Faremos a filtragem da imagem multiplicando a resposta em frequência do filtro pela transformada da imagem. Dessa forma iremos remover as frequências de interferência.

```
filtrado =
```



```
filtrado = Gray.(real.(ifft(ifftshift(CASSINI.*H10))))
```

Filtragem com outros filtros

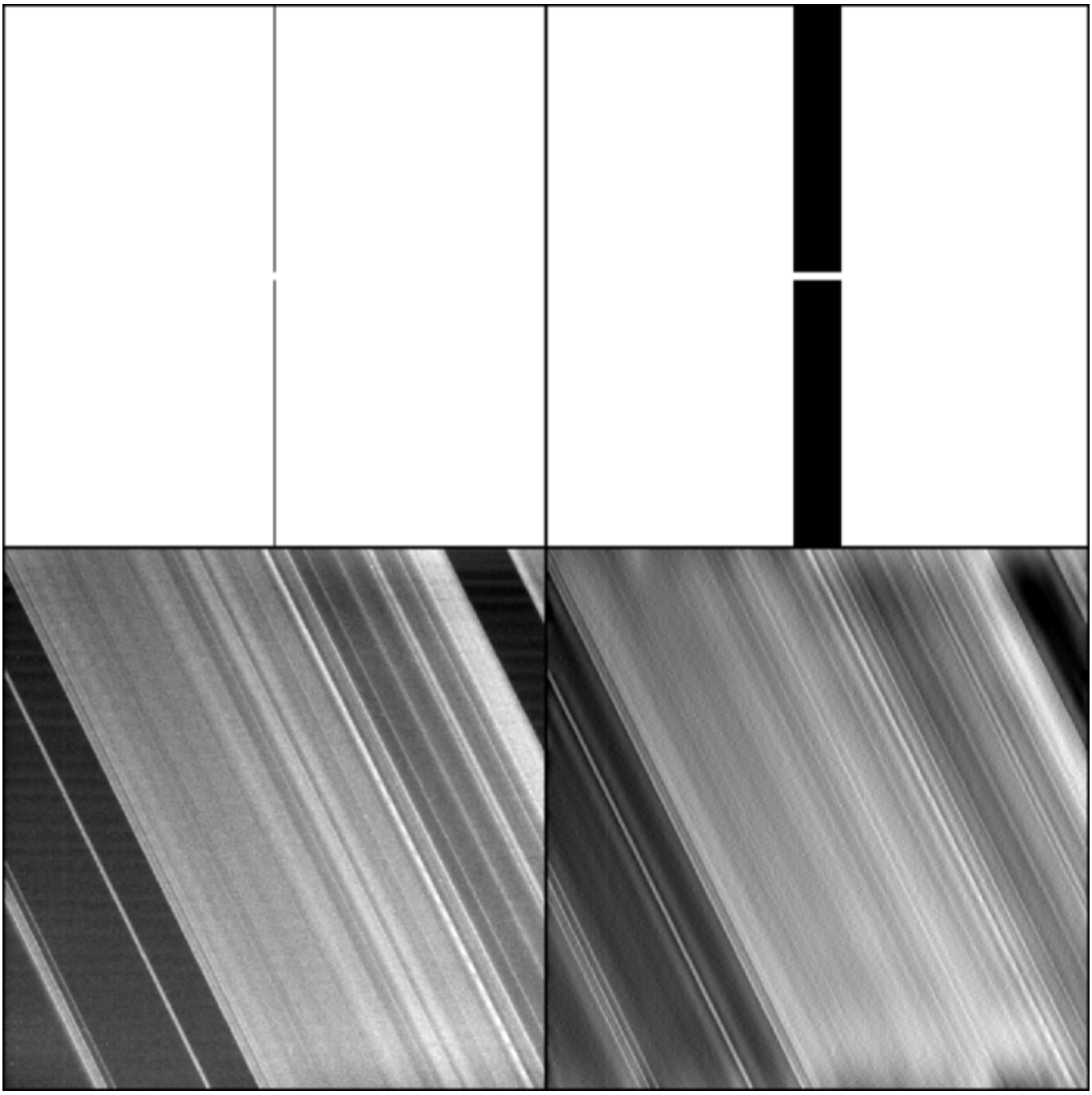
Aqui iremos alargar e achatar o filtro para verificar as alterações na imagem

```
begin
    H1 = ones(nlin,ncol)
    H1[:,centro:centro+1].=0 #zera colunas 333:342
    H1[centro-4:centro+5,:] .=1 #recupera o centro DC da imagem
    imgH1 = Gray.(real.(H1))

    H60 = ones(nlin,ncol)
    H60[:,centro-29:centro+30].=0 #zera colunas 333:342
    H60[centro-4:centro+5,:] .=1 #recupera o centro DC da imagem
    imgH60 = Gray.(real.(H60))

    filtrado1 = Gray.(real.(ifft(ifftshift(CASSINI.*H1))))
    filtrado60 = Gray.(real.(ifft(ifftshift(CASSINI.*H60))))
    md""
end
```

Aqui temos as respostas dos filtro na linha de cima, e na linha de baixo as imagens filtradas.



É possível observar que no filtro mais achatado o ruído não foi completamente removido; já o filtro mais largo distorce muito a imagem.

Isso ocorre no primeiro caso, por dois motivos. Primeiramente porque estamos fazer um filtragem definindo apenas pontos da TDF, ou seja, estamos definindo zero em amostras da transformada deste filtro, o que não significa que a resposta será zero em todo aquele intervalo. Além disso, também há um espalhamento da interferência nas frequÊncias centrais do espectro da imagem.

No segundo caso, a distorção acontece por estamos removendo uma faixa extensiva de frequências importantes para a formação da imagem.

Blurry-Moon

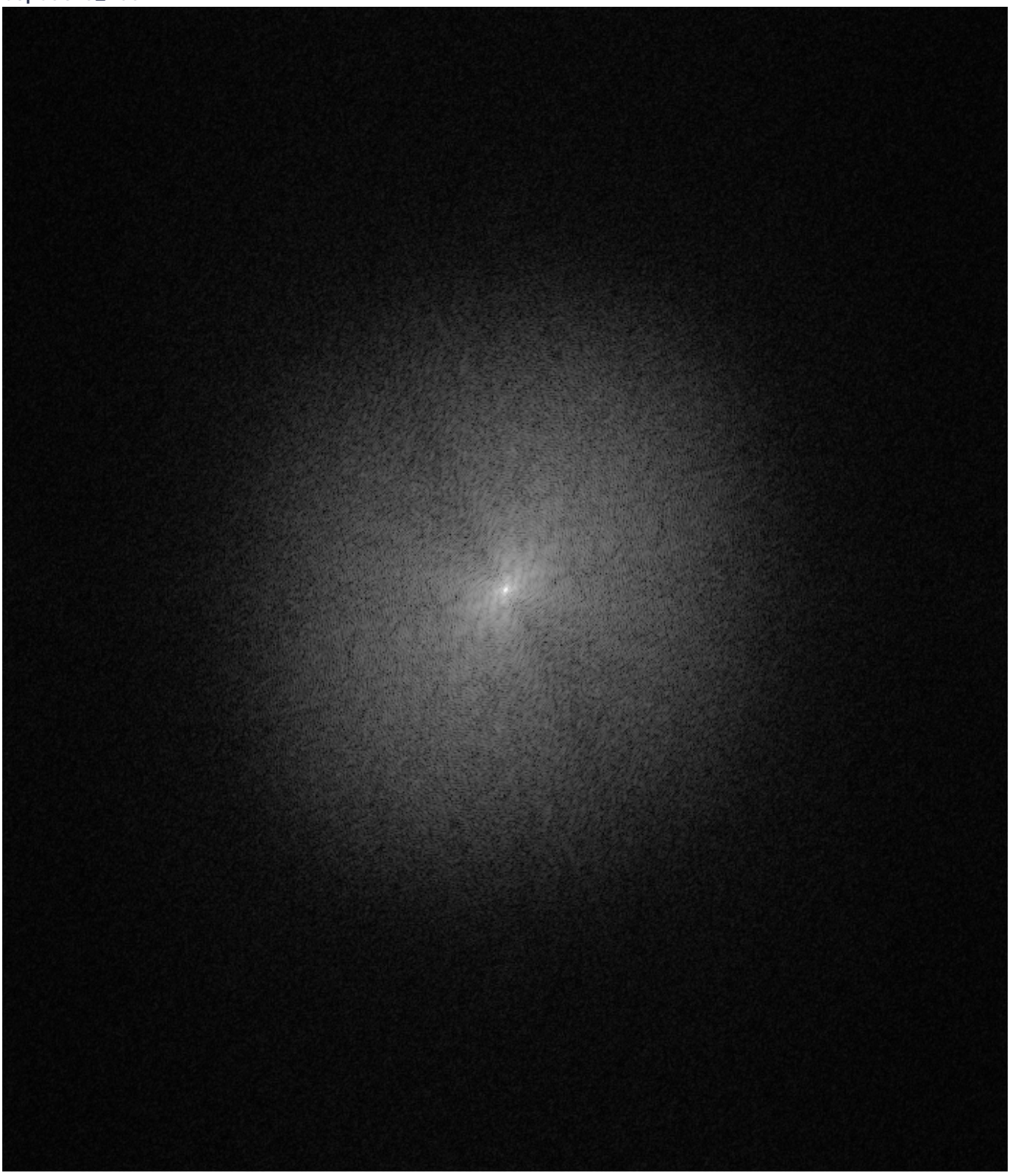
```
begin
    moon = load("blurry-moon.tif")
end
```

Análise da imagem

Veremos aqui as características do espectro da imagem da lua. É possível observar que a maior parte do espectro está concentrada no centro do espectro (próximo ao DC).

```
begin
    moon_f = (Float64).(moon)
    MOON = fftshift(fft(moon_f))
    md""
end
```

espectro_moon =



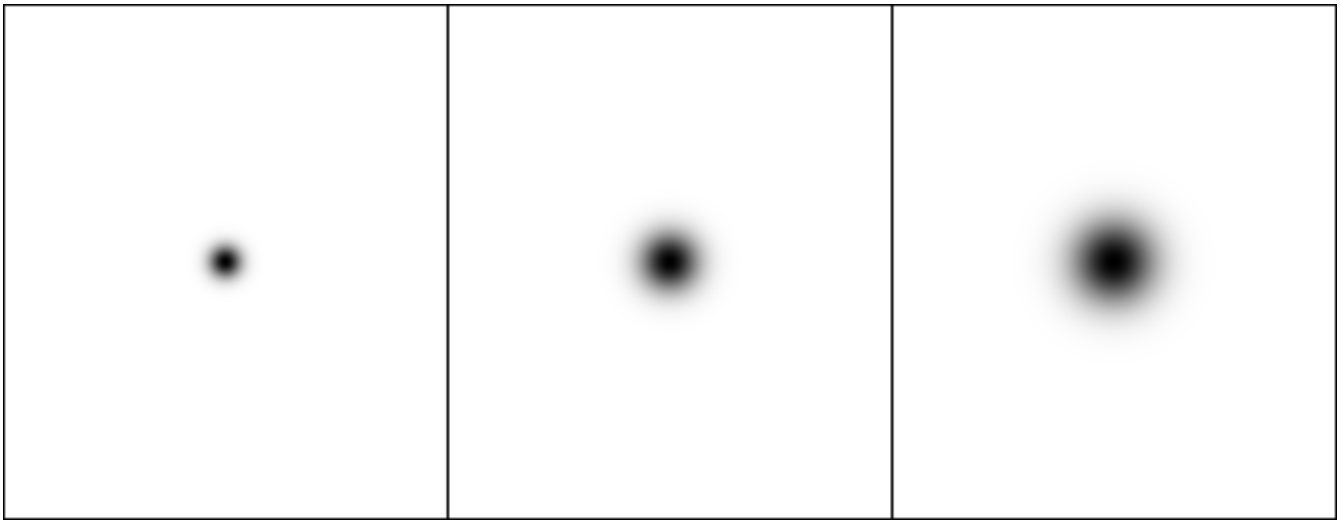
```
espectro_moon= Gray.(log.(abs.(MOON) .+ 1) ./ maximum(log.(abs.(MOON) .+1)))
```

Extração de bordas

Nesta parte iremos extrair as bordas da imagem da lua. Para isso, aplicaremos a imagem em um filtro passa altas que deixará apenas o espectro de alta frequência, e portanto irá deixar apenar as bordas da imagem

```
begin
    H300 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],300)
    H1000 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],1000)
    H2000 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],2000)
    md""
end
```

Filtros com σ = 300, σ = 1000 e σ = 2000

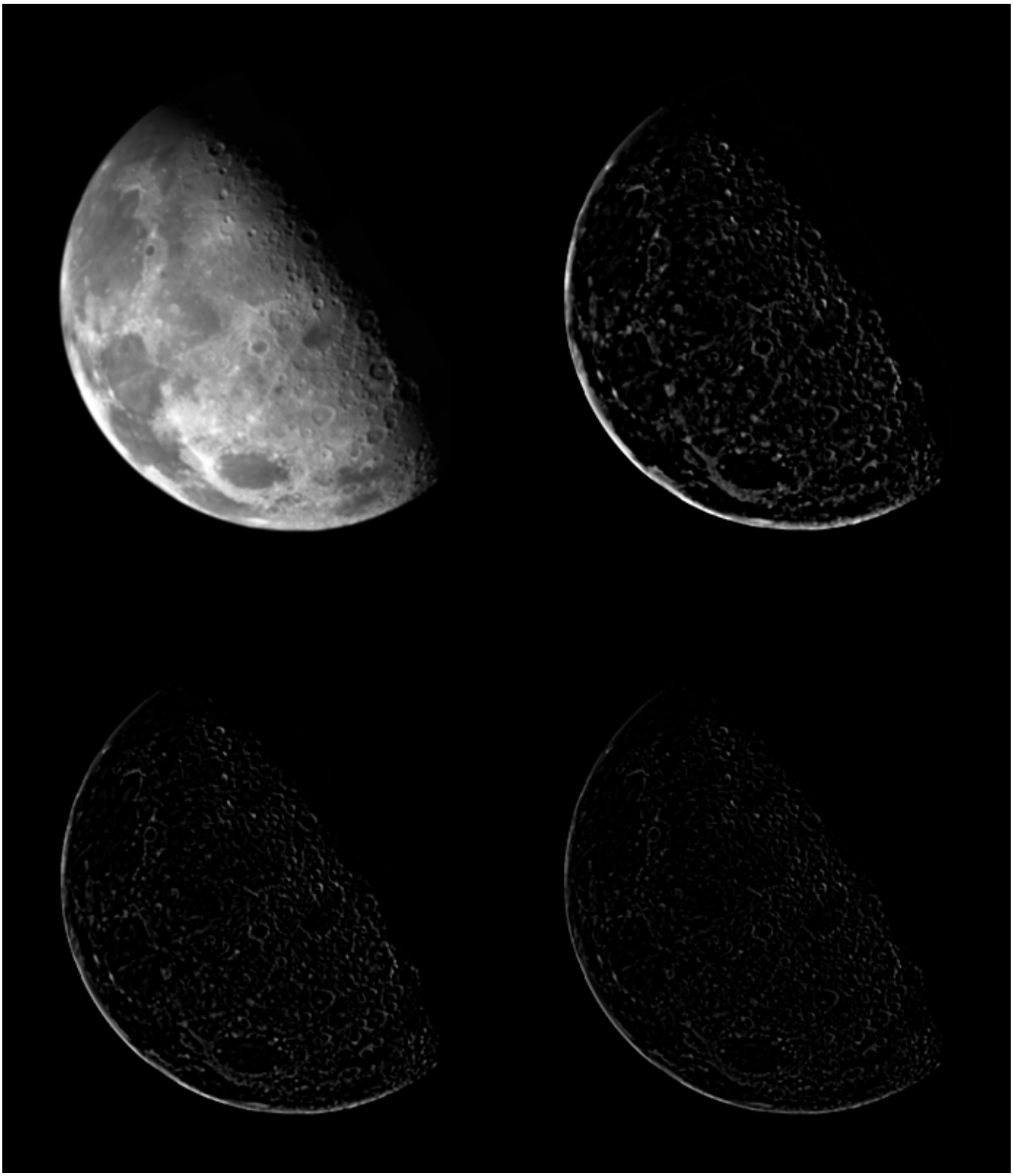


```
begin
    FILTRADO300 = MOON.*H300
    filtrado300 = real.(ifft(ifftshift(FILTRADO300)))

    FILTRADO1000 = MOON.*H1000
    filtrado1000 = real.(ifft(ifftshift(FILTRADO1000)))

    FILTRADO2000 = MOON.*H2000
    filtrado2000 = real.(ifft(ifftshift(FILTRADO2000)))
    md""
end
```

Vemos aqui a imagem original e suas filtragens com diferentes valores de σ.



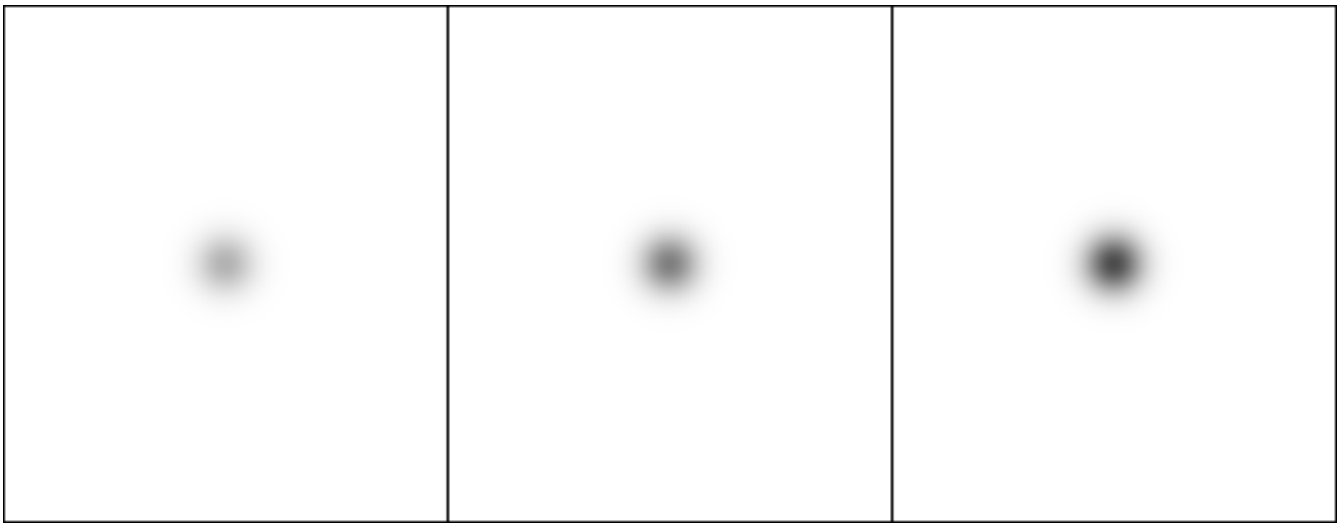
Observa-se que quão maior o valkor de σ, mas finas são as bordas extraídas.

Realce de bordas

Nesta parte iremos fazer apenas um realce de bordas. Para isso, iremos aplicar um filtro gaussiano que não remove completamente o espectro próximo ao DC.

```
begin
    H3 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],1000,0.3)
    H5 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],1000,0.5)
    H7 = passa_alta_gaussiano(size(MOON)[1],size(MOON)[2],1000,0.7)
    md""
end
```

Filtros com : α = 0.3 , α = 0.5, α = 0.7



```
begin
    FILTRADO3 = MOON.*H3
    filtrado3 = real.(ifft(ifftshift(FILTRADO3)))

    FILTRADO5 = MOON.*H5
    filtrado5 = real.(ifft(ifftshift(FILTRADO5)))

    FILTRADO7 = MOON.*H7
    filtrado7 = real.(ifft(ifftshift(FILTRADO7)))
    md""
end
```

Vemos aqui a imagem original e suas filtragens com diferentes valores de α.

Vemos que dessa forma há um realce nas bordas da imagem. O fator α define quanto será o contraste entre as bordas e a imagem original.

Anexos

Funções usadas neste projeto

```
make_border (generic function with 1 method)
```

```
function make_border(imagem, largura)
    nlin = size(imagem)[1] + 2*largura
    ncol = size(imagem)[2] + 2*largura
    saida = Gray.(zeros(nlin,ncol))
    saida[largura+1:nlin-largura,largura+1:ncol-largura] = imagem
    return saida
end
```

```
passa_alta_gaussiano (generic function with 1 method)
```

```
begin
    """
        passa_alta_gaussiano(size_k1,size_k2,σ)
    Retorna um filtro passa altas 2D gaussiano com `size_k1` linhas, `size_k2` colunas e uma variância `σ`
    """

    function passa_alta_gaussiano(size_k1,size_k2,σ)
        lin_center = size_k1/2
        col_center = size_k2/2
        filtro = zeros(size_k1,size_k2)
        for k1 in 1:size_k1
            for k2 in 1:size_k2
                filtro[k1,k2] =  1 - exp(-((k1 - lin_center)^2+(k2 - col_center)^2)/(2σ))
            end
        end
        return filtro
    end
end
```

```
passa_alta_gaussiano (generic function with 2 methods)
```

```
begin
    """
        passa_alta_gaussiano(size_k1,size_k2,σ)
    Retorna um filtro passa altas 2D gaussiano com `size_k1` linhas, `size_k2` colunas e uma variância `σ` e atenuação `α`
    """

    function passa_alta_gaussiano(size_k1,size_k2,σ, α)
        lin_center = size_k1/2
        col_center = size_k2/2
        filtro = zeros(size_k1,size_k2)
        for k1 in 1:size_k1
            for k2 in 1:size_k2
                filtro[k1,k2] =  1 - α * exp(-((k1 - lin_center)^2+(k2 - col_center)^2)/(2σ))
            end
        end
        return filtro
    end
end
```