

Relatório - Trabalho 01

MO443 - Introdução ao Processamento de Imagem Digital

VINICIUS TEIXEIRA DE MELO - RA: 230223

Universidade Estadual de Campinas
viniciusteixeira@liv.ic.unicamp.br

16 de Abril de 2019

I. ESPECIFICAÇÃO DO PROBLEMA

O objetivo deste trabalho é implementar alguns filtros de imagens no domínio espacial e de frequências. A filtragem aplicada a uma imagem digital é uma operação local que altera os valores de intensidade dos pixels da imagem levando-se em conta tanto o valor do pixel em questão quanto valores de pixels vizinhos.

No processo de filtragem, utiliza-se uma operação de convolução de uma máscara pela imagem. Este processo equivale a percorrer toda a imagem alterando seus valores conforme os pesos da máscara e as intensidades da imagem.

O trabalho está dividido em duas seções principais:

- Filtragem no Domínio Espacial
- Filtragem no Domínio de Frequências

Na seção de filtragem no domínio espacial, são dados 4 seguintes tipos de filtros:

$$(a) h_1 = \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & -1 & 0 & 0 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline -1 & -2 & 16 & -2 & -1 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline 0 & 0 & -1 & 0 & 0 \\ \hline \end{array}$$

$$(b) h_2 = \frac{1}{256} \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

$$(c) h_3 = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$(d) h_4 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

As aplicações dos filtros devem ser feitas de forma individual, porém, os filtros h_3 e h_4 devem ser aplicados de forma combinada, somando-se as respostas de cada um dos filtros por meio da seguinte expressão: $\sqrt{(h_3)^2 + (h_4)^2}$.

Na seção de filtragem no domínio de frequências, é necessário aplicar um filtro Gaussiano em uma imagem representada por seu espectro de Fourier, e a componente de frequência zero deve ser transladada para o centro do espectro. Nesse experimento, é necessário testar diferentes graus de suavização, de forma a borrar a imagem com mais ou menos intensidade.

II. ENTRADA DE DADOS

O código fonte criado para a execução de todas as tarefas está no notebook **Trabalho 01.ipynb**. O código foi criado para aceitar imagens em tons de cinza no formato RGB (*Red, Green and Blue*) do tipo PNG (*Portable Network Graphics*).

Para executar o notebook, basta iniciar o ambiente *Jupyter Notebook*, abrir o notebook **Trabalho 01.ipynb** e executar as células em ordem. Todo o algoritmo foi implementado na linguagem Python na versão 3.6.

As imagens de entrada utilizadas nos testes do algoritmo foram retiradas da página do prof. Hélio Pedrini: [Imagens](#). Na pasta **imgs/** estão as duas imagens monocromáticas utilizadas nos testes: **baboon.png** e **butterfly.png**. As

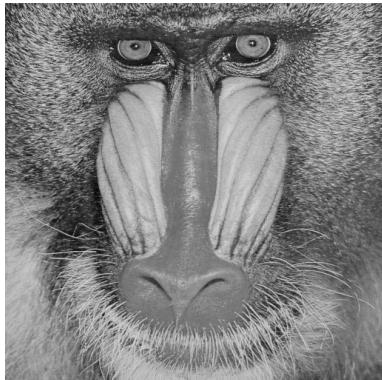


Figura 1: baboon.png

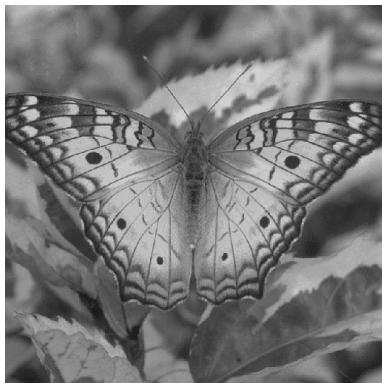


Figura 2: butterfly.png

dimensões das imagens de entrada utilizadas são 512x512.

III. DEPENDÊNCIAS E CÓDIGOS

As bibliotecas utilizadas neste trabalho foram:

Biblioteca	Versão
numpy	1.16.2
cv2	3.4.2
matplotlib	3.0.3
skimage	0.15.0
scipy	1.2.1
warnings	2.1

A leitura das imagens foi realizada utilizando uma função do **opencv** [1] chamada **imread()**, a qual necessitou de uma cons-

tante do próprio **opencv** para que a imagem ficassem apenas com o canal de escala de cinza, essa constante é denominada **IMREAD_GRAYSCALE**.

Os filtros foram gerados manualmente através da função **numpy.array()**. A função para gerar o preenchimento de borda presente no código utiliza uma função pronta do **opencv** denominada **copyMakeBorder()**, a qual preenche na largura a na altura a imagem com valor de **pixel 0**.

A função de convolução 2D foi criada manualmente, mas antes de acontecer as multiplicações e somas, o filtro sofre uma rotação de 180° utilizando a função **numpy.rot90()** duas vezes. Para plotar os resultados utilizamos a biblioteca **matplotlib**.

Para o cálculo da última questão da **seção 1.1**, foi necessário transformar a saída da aplicação do filtro h_3 e h_4 em inteiros de 32 bits, pois na expressão $\sqrt{(h_3)^2 + (h_4)^2}$ pode haver um estouro de inteiro, por conta de estar representado, inicialmente, por inteiros de 8 bits.

Na **seção 1.2** são utilizadas as funções **cv2.dft()** e **cv2.idft()** para transformação de Fourier e transformação inversa de Fourier, respectivamente. Após levar a imagem para o domínio de frequência é necessário transladar a componente de frequência zero para o centro da imagem, para isso é utilizado a função **numppy.fft.fftshift()** e, para ter essa operação inversa, é utilizado a função **numppy.fft.ifftshift()**.

IV. FUNDAMENTAÇÃO

i. Preenchimento de borda

As imagens de entrada foram carregadas em memória utilizando uma função do OpenCv [1] que lê a imagem em escala de cinza e salva em uma estrutura I_{mxn} , na qual m e n representa a largura e a altura da imagem, respectivamente.

Para a aplicação da operação de convolução na imagem, foi necessário realizar um pré-processamento chamado de *padding*. A operação de *padding* é usada para gerar espaços em

torno do conteúdo de um elemento, nesse caso, a imagem. Nesse trabalho, o *padding* gerado nas imagens foi de tamanho $\lfloor \frac{x}{2} \rfloor$, onde x é a dimensão do filtro que será convoluído com a imagem. Todo o *padding* foi preenchido com o valor 0, ou seja, com a cor preta.

ii. Convolução 2D

Convolução 2D é uma operação que, a partir de imagem e um filtro, resulta em uma terceira imagem que mede a soma do produto da imagem e o filtro inicial ao longo da região subentendida pela superposição delas em função do deslocamento existente entre elas.

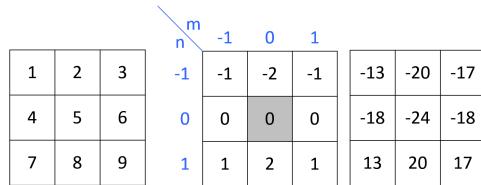


Figura 3: Exemplo de imagem, filtro e resultado da convolução, respectivamente.

A operação de convolução, primeiramente, rotaciona o filtro em 180° e depois multiplica pixel a pixel, da seguinte forma:

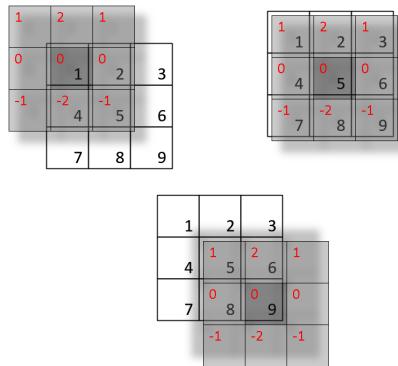


Figura 4: Exemplo de aplicação de convolução 2D.

Nesse trabalho, foram utilizados 2 tipos de filtros:

1. Filtro passa-alta: filtros utilizados para realçar certas características presentes nas

imagens, tais como bordas, linhas ou regiões de interesse [2].

2. Filtro passa-baixa: filtros utilizados para suavizar as imagens, uma vez que as frequências altas correspondem às transições abruptas são atenuadas [2].

Os filtro h_1 , h_3 e h_4 são exemplos de filtros passa-alta. Já o filtro h_2 é um exemplo de filtro passa-baixa. A seguir temos a exemplificação gráfica desses filtros.

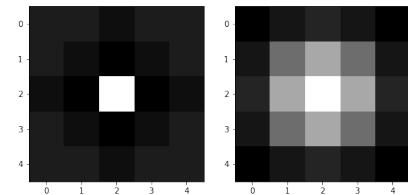


Figura 5: Filtros h_1 e h_2 .

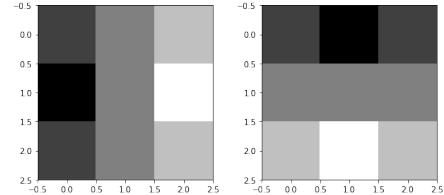


Figura 6: Filtros h_3 e h_4 .

iii. Redimensionamento

A operação de redimensionamento foi utilizada para transformar as saídas da operação de convolução 2D, que não necessariamente estavam no intervalo de escalas de cinza de $[0, 255]$, de volta ao intervalo de escalas de cinza representado por inteiros de 8 bits.

iv. Transformada Discreta de Fourier

A Transformada Discreta de Fourier (DFT) é uma transformação de coordenadas em componentes pertencentes ao conjunto dos números complexos, em que cada coeficiente é obtido

pela combinação linear dos elementos da entrada com o núcleo da transformada [2].

A Transformada de Fourier para duas variáveis pode ser escrita da seguinte forma:

$$F(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

e a partir de $F(u, v)$, pode-se obter $f(x, y)$ através da Transformada Inversa de Fourier:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux + vy)] du dv.$$

Na filtragem no domínio de frequência, o principal objetivo, é que as operações de convolução possuem o custo computacional menor do que no domínio espacial. Portanto, normalmente, o processamento de imagens possuem a seguinte sequência.



Figura 7: Sequência usualmente utilizada para as operações de processamento de imagens.

v. Filtro Gaussiano

Os filtros Gaussianos são um tipo de filtro passa-baixa, os quais suavizam a imagem filtrada. Nos filtros Gaussianos, os coeficientes da máscara são derivados a partir de uma função Gaussiana bidimensional [2]. A função Gaussiana discreta com média zero e desvio padrão σ é definida como

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right).$$

As variações de sigma utilizados neste trabalho foram: $\sigma = [2, 4, 8, 16, 32, 64]$.

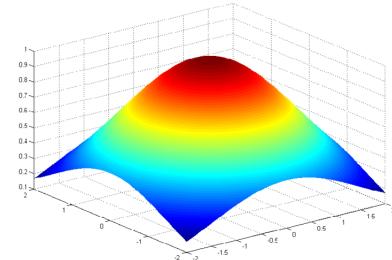


Figura 8: Exemplificação gráfica de um filtro Gaussiano.

V. SAÍDA DE DADOS

Os imagens resultantes da seção 1.1 e 1.2 foram salvas dentro da pasta **resultados/** utilizando uma função da biblioteca **matplotlib** chamada **matplotlib.pyplot.imsave()** [3].

O formato dos nomes de saída estão da seguinte forma: para a seção 1.1 o padrão é o nome da imagem de entrada (*baboon* ou *butterfly*) concatenado com o nome do filtro aplicado sobre ela; na seção 1.2 são geradas 4 imagens principais resultantes para cada valor de sigma atribuído na função de criar um filtro gaussiano.

VI. RESULTADOS E DISCUÇÕES

Os resultados obtidos na execução dos códigos da **seção 1.1** estão a seguir:

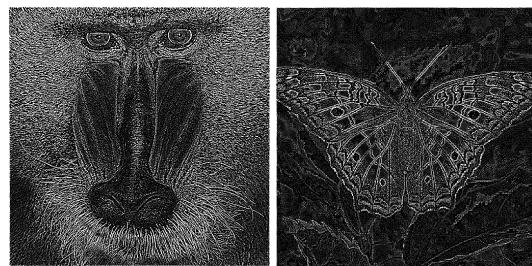


Figura 9: Resultado da aplicação do filtro h_1 .

Como já dito, o filtro h_1 é do tipo passa-alta, o qual atenua o valor central da máscara considerada na imagem. Como resultado da passagem desse filtro nas imagens de entrada,

podemos observar que os *pixels* de maior intensidade na imagem original foram intensificados, e os *pixels* de menor intensidade tiveram seus valores reduzidos.



Figura 10: Resultado da aplicação do filtro h_2 .

O filtro h_2 é do tipo passa-baixa, pois é baseado em uma distribuição estatística, nesse caso, a média. O resultado obtido pela convolução do filtro h_2 com as imagens de entrada nos dá como saída imagens suavizadas (com um leve borramento), pois nesse tipo de filtro, os valores de *pixels* da vizinhança são levados em consideração no cálculo do *pixel* central da posição corrente.

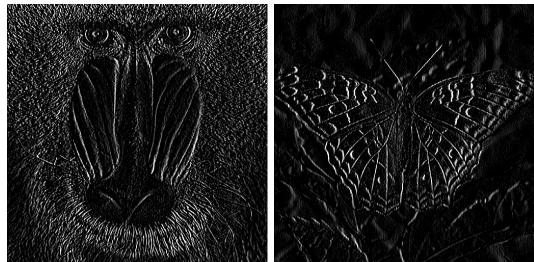


Figura 11: Resultado da aplicação do filtro h_3 .

O filtro h_3 é do tipo passa-alta, pois a soma acumulada das células é 0. Esse é um filtro muito conhecido na área de processamento de imagens, denominado *Filtro de Sobel*. A aplicação desse filtro consiste num operador que calcula diferenças finitas, dando uma aproximação do gradiente da intensidade dos *pixels* da imagem. Como resultado da aplicação do filtro h_3 temos a detecção de bordas na vertical nas imagens de entrada.

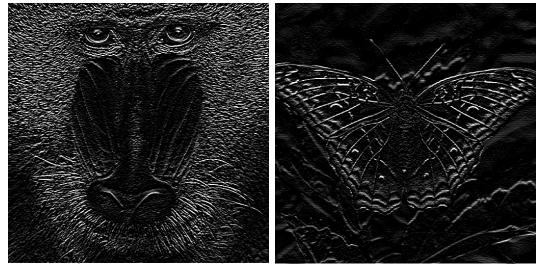


Figura 12: Resultado da aplicação do filtro h_4 .

O filtro h_4 também é do tipo passa-alta e também é denominado como *Filtro de Sobel*. Porém, esse filtro detecta bordas na horizontal em imagens, atenuando esse valores para que passem a ficar mais visíveis na imagem de saída.

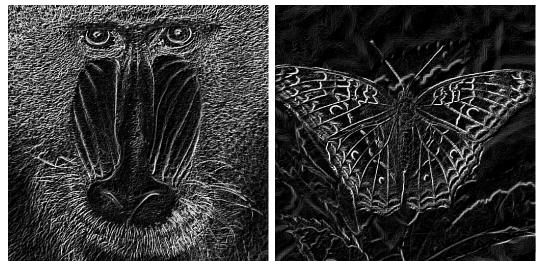


Figura 13: Resultado da expressão $\sqrt{(h_3)^2 + (h_4)^2}$.

Como resultado da combinação, somando-se as respostas de cada um filtros por meio da expressão $\sqrt{(h_3)^2 + (h_4)^2}$ temos uma imagem que equilibra a detecção de bordas na vertical, obtidas com a aplicação do filtro h_3 , e a detecção de bordas na horizontal, obtidas com a aplicação do filtro h_4 . Os componentes que estão elevados ao quadrado tem como objetivo a atenuação das componentes de alta intensidade, assim, com a operação de radiciação, obtemos um grau de combinação entre as linhas verticais e horizontais.

Os resultados obtidos na execução dos códigos da **seção 1.2** estão a seguir:

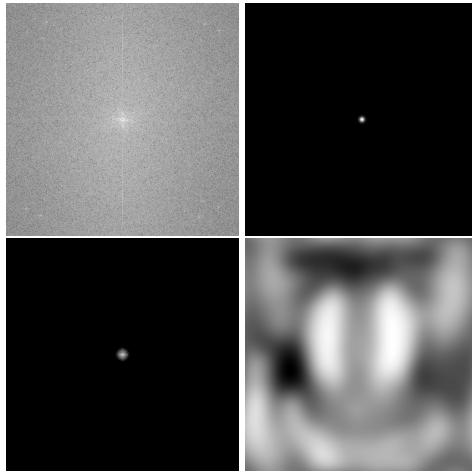


Figura 14: Resultados obtidos com o sigma valendo 4.

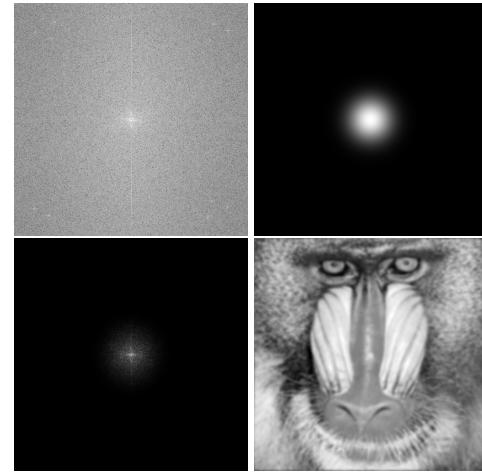


Figura 15: Resultados obtidos com o sigma valendo 32.

Na Figura 14 temos, na primeira imagem, o espectro de magnitude gerado após a aplicação da *Transformada Discreta de Fourier* na imagem e a componente de frequência zero ser transladada para o centro da imagem. Na segunda imagem da primeira linha temos a visualização gráfica de um filtro gaussiano com média 0 e desvio padrão de valor 4, como o filtro é do tamanho da imagem, os intensidades visíveis estão agrupadas em um círculo bem pequeno no centro da imagem, por conta do baixo valor de desvio padrão. Na primeira imagem da segunda linha temos o resultado da aplicação do filtro gaussiano no espectro de frequência da imagem *baboon.png*. Como o filtro em questão possui um baixo valor de desvio padrão, então ao ser aplicado pelo espectro de frequência, ele deixa passar somente valores correspondentes a sua área. Nesse caso, o espectro de frequência resultante perdeu informações sobre as frequências mais altas (que são as componentes presentes em regiões mais escuras). Por fim, a segunda imagem da segunda linha é o resultado da aplicação da *Transformada Inversa de Fourier*, que transforma a imagem de volta para o domínio espacial. Podemos observar que, como houve a aplicação de um filtro gaussiano com baixo desvio padrão, a imagem resultante possui um borramento alto em comparação com a imagem original.

Na Figura 15 temos a mesma ordem da figura anterior, porém, nesse caso, o valor de desvio padrão aplicado foi de 32. Podemos ver que o filtro possui uma circunferência bem maior e que, assim, após a aplicação desse filtro no espectro de frequência da imagem original, o espectro de frequência resultante possui mais componentes de frequências mais altas do que a anterior. Portanto, após a aplicação da função inversa, vemos que a imagem resultante possui somente um leve borramento, pois seu espectro tinha mais informações.

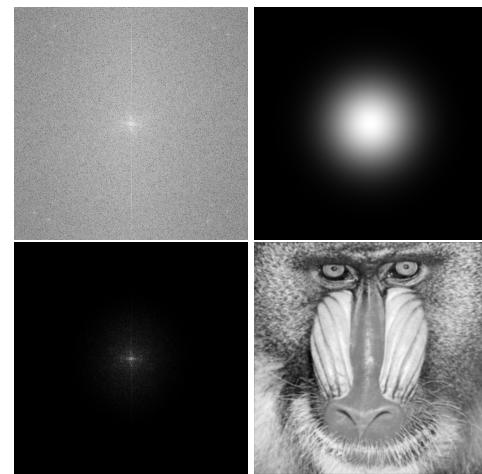


Figura 16: Resultados obtidos com o sigma valendo 64.

Por fim, na Figura 22 temos um filtro gaussiano com desvio padrão valendo 64. Podemos observar que o filtro ficou bem mais abrangente, em relação aos anteriores, e que sua aplicação no espectro de frequência deixou passar componentes de frequências bem mais altas, que resultam em mais informações para a reconstrução da imagem inicial. Como resultado final, vemos que a imagem possui um grau de detalhes bem maior que os resultados anteriores e que já é bem mais próxima da imagem original.

Portanto, com esses resultados podemos observar que quanto maior o filtro gaussiano, mais informações do espectro de frequência vão ser liberados e, assim, na aplicação da *Transformação Inversa de Fourier*, a função terá bem mais informações para a reconstrução da imagem original. Quanto mais informações no espectro de frequência resultante, mais próximo da imagem original é o resultante e, analogamente, quanto menos informações no espectro de frequência resultante, mais distante (mais borrado) da imagem original é o resultado.

VII. CONCLUSÃO

Podemos concluir que os resultados obtidos na **seção 1.1** e na **seção 1.2** foram satisfatórios quanto a especificação do trabalho e que a visualização prática dos tipos de filtros (passa-alta e passa-baixa), aplicação de filtros no domínio de frequência, mudanças de domínios das imagens e combinação de resultados, consolida ainda mais os conceitos vistos em sala de aula.

REFERÊNCIAS

- [1] Welcome to opencv documentation! <https://docs.opencv.org/2.4/index.html>
Acesso em: 15/04/2019.
- [2] Pedrini, Hélio, and William Robson Schwartz. Análise de imagens digitais: princípios, algoritmos e aplicações. Thomson Learning, 2008.

[3] Matplotlib Version 3.0.3
<https://matplotlib.org/contents.html>
Acesso em: 15/04/2019.

ANEXOS

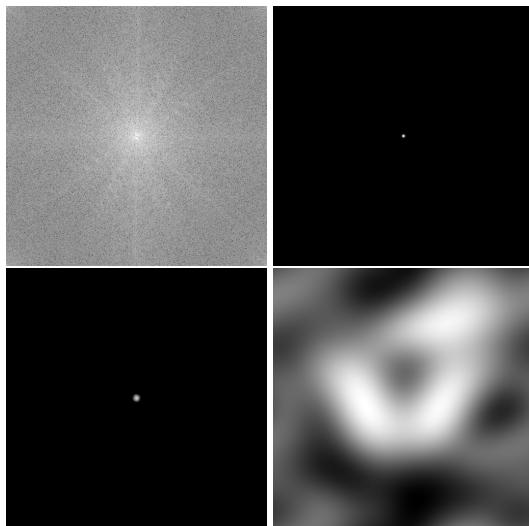


Figura 17: Resultados obtidos com o sigma valendo 2.

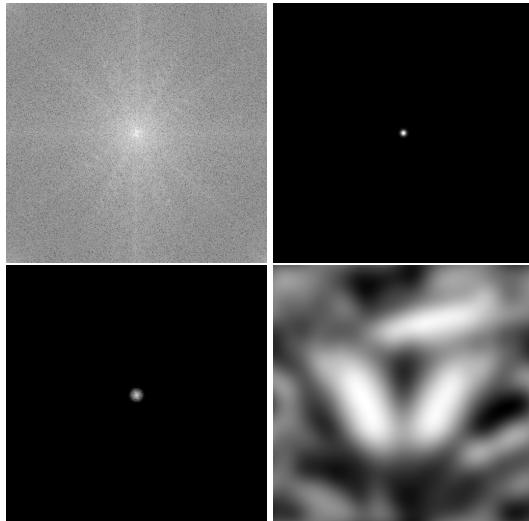


Figura 18: Resultados obtidos com o sigma valendo 4.

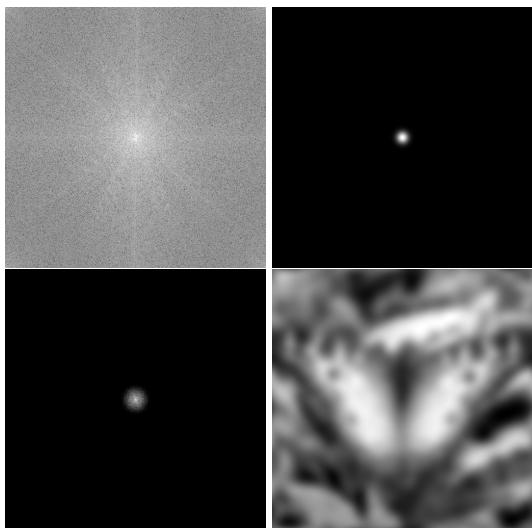


Figura 19: Resultados obtidos com o sigma valendo 8.

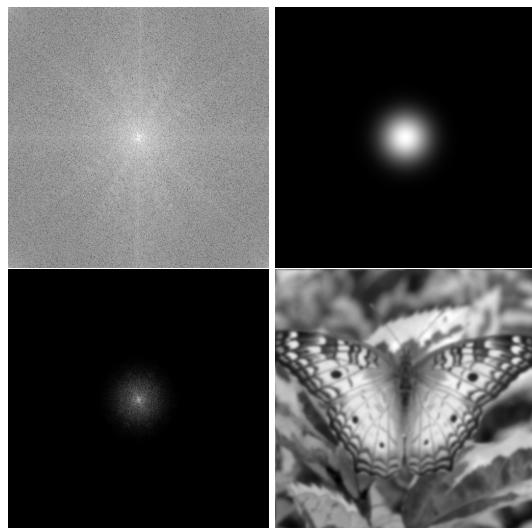


Figura 21: Resultados obtidos com o sigma valendo 32.

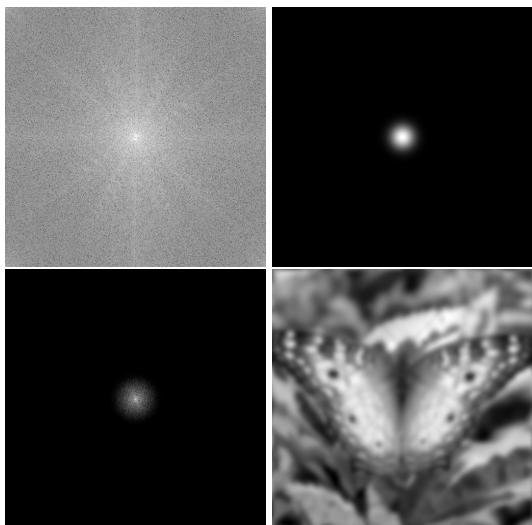


Figura 20: Resultados obtidos com o sigma valendo 16.

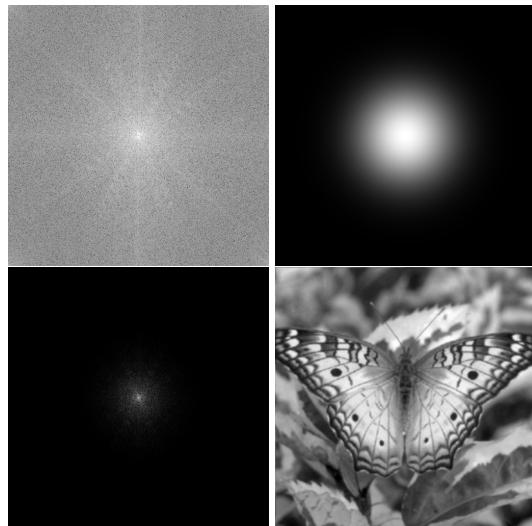


Figura 22: Resultados obtidos com o sigma valendo 64.