Relatório do Trabalho 1

Raissa Cavalcante Correia - RA: 150619
Disciplina: MC920 - 1s2019 - Turma A
18 de Abril de 2019

Introdução

Sendo o trabalho sobre filtragem no domínio espacial e da frequência, foram utilizadas diversas bibliotecas para que o desenvolvimento fosse de melhor compreensão.

```
from scipy import ndimage, misc
import numpy.fft
import numpy
import matplotlib.pyplot as plt
import cv2
```

Para executar o código em python 3 que foi anexado a este .zip, é necessário a instalação dos seguintes módulos: Scipy, Numpy, Matplotlib e cv2.
Ao compilar como qualquer programa em python 3 "`python3 trabalho1.py`", o usuário deve digitar o nome de um dos arquivos que deseja dar como entrada, este arquivo será usado para os 5 resultados da primeira parte e do resultado da segunda parte.

```
print("Escolha uma das seguintes imagens digitando as seguintes strings:")
print("city.png, baboon.png, butterfly.png,house.png,seagull.png")
entrada = input()
```

Em seguida pede-se o sigma para a segunda parte.

```
print("Escolha o sigma")
fator_sigma = float(input())
```

Com isso o código irá plotar na tela o resultado da segunda parte, com o matplotlib. E o resultado da primeira parte serão 5 fotos na pasta "filtered".

Execução da Primeira Parte

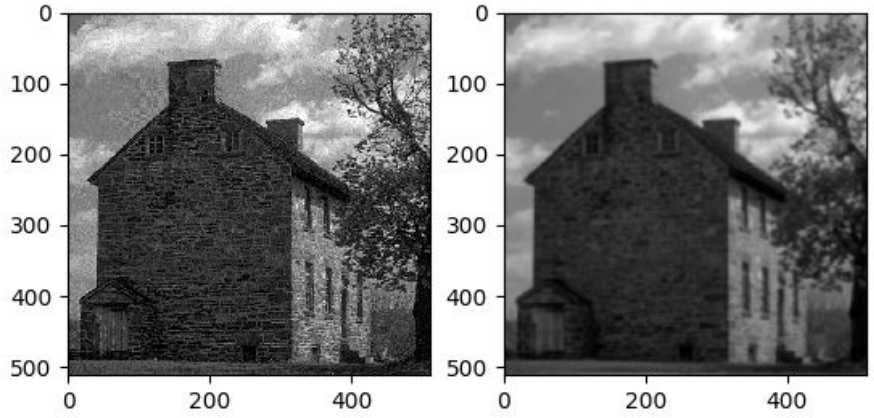
Em primeiro lugar foi necessário deixar especificado os 5 kernels de filtragem. Onde os 4 primeiros eram matrizes muito bem especificadas e o 5º uma matriz calculadas a partir da terceira e da quarta dada a especificação. $h5 = \sqrt{(h3)^2 + (h4)^2}$
A biblioteca numpy cumpriu o propósito. Após isso, houve um arredondamento da matriz h5 para que não houvessem casas decimais uma vez que o resultado do filtro era uma imagem quase totalmente branca e a filtragem em um comentario abaixo não resolveu a questão.
Na sequência usa-se o cv2 e sua função imread para converter a imagem png de entrada em um numpy array, e tornar possível a convolução pelo cv2.

Então aplica-se a convolução filter2D do cv2[4], usando como parâmetro a imagem recebida pelo imread, a matriz h1 a h5 e o parâmetro anchor como -1, o padrão, que é posicionar o kernel centralizado tanto no eixo x quanto y da imagem.
Por fim usa-se o imwrite do cv2 para que a imagem seja salva nos arquivos dentro da pasta, como uma forma prática de visualizar os 5 de uma vez, e reescreva a cada teste.

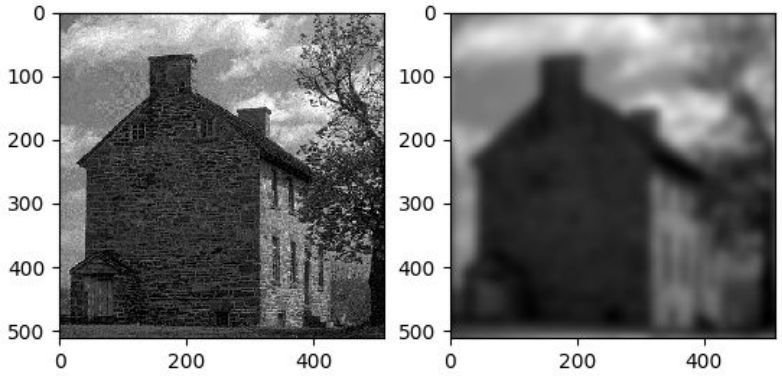
Execução da Segunda Parte

As primeiras 3 linhas são dedicadas ao plot do matplotlib e da leitura da imagem png como um numpy array pelo imread do cv2.
Em seguida aplicamos a função de fourier do numpy[2]. Levando em conta o pedido do enunciado do trabalho, não foi usado nenhum parâmetro opcional por não haver necessidade, tanto para a aplicação quanto para a inversa de fourier.[3]
Depois usa-se a função fourier_gaussian da ndimage do scipy[1], nesta função foi necessário apenas passar a imagem e o sigma que foi passado como input. Pois a questão da frequência-zero ser transladada para o centro da imagem é realizada com o parâmetro axis que por default é igual a -1.
Após tudo isso nas 3 últimas linhas temos o plot da imagem original e da parte real da imagem modificada.

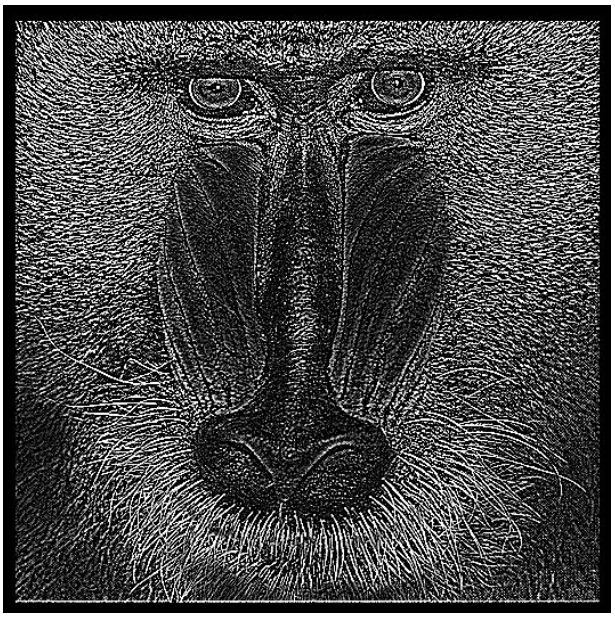
Resultados



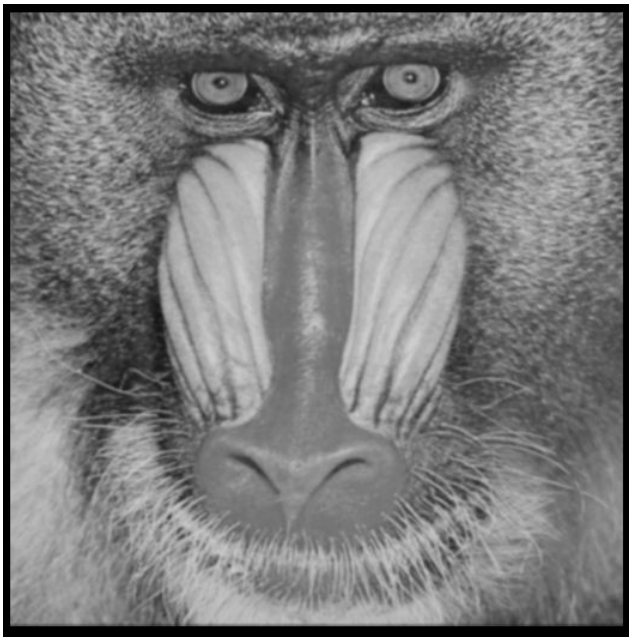
house.png aplicando-se sigma=2



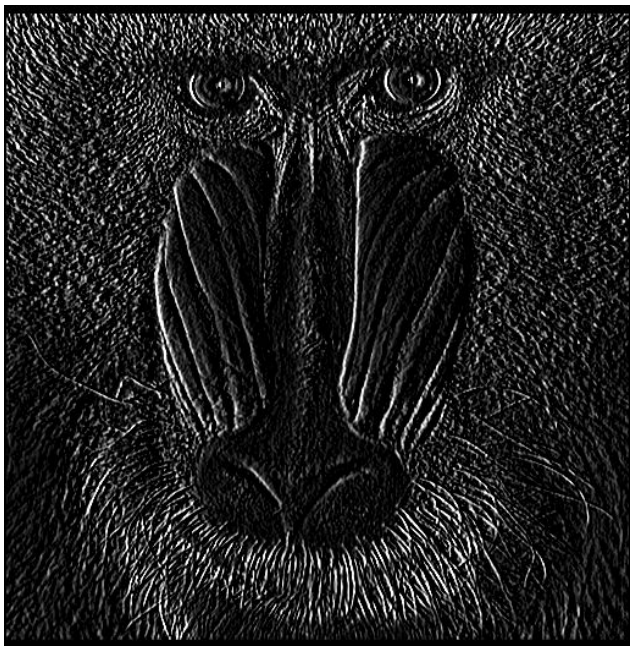
house.png aplicando-se sigma=8



baboon.png após filtro 1

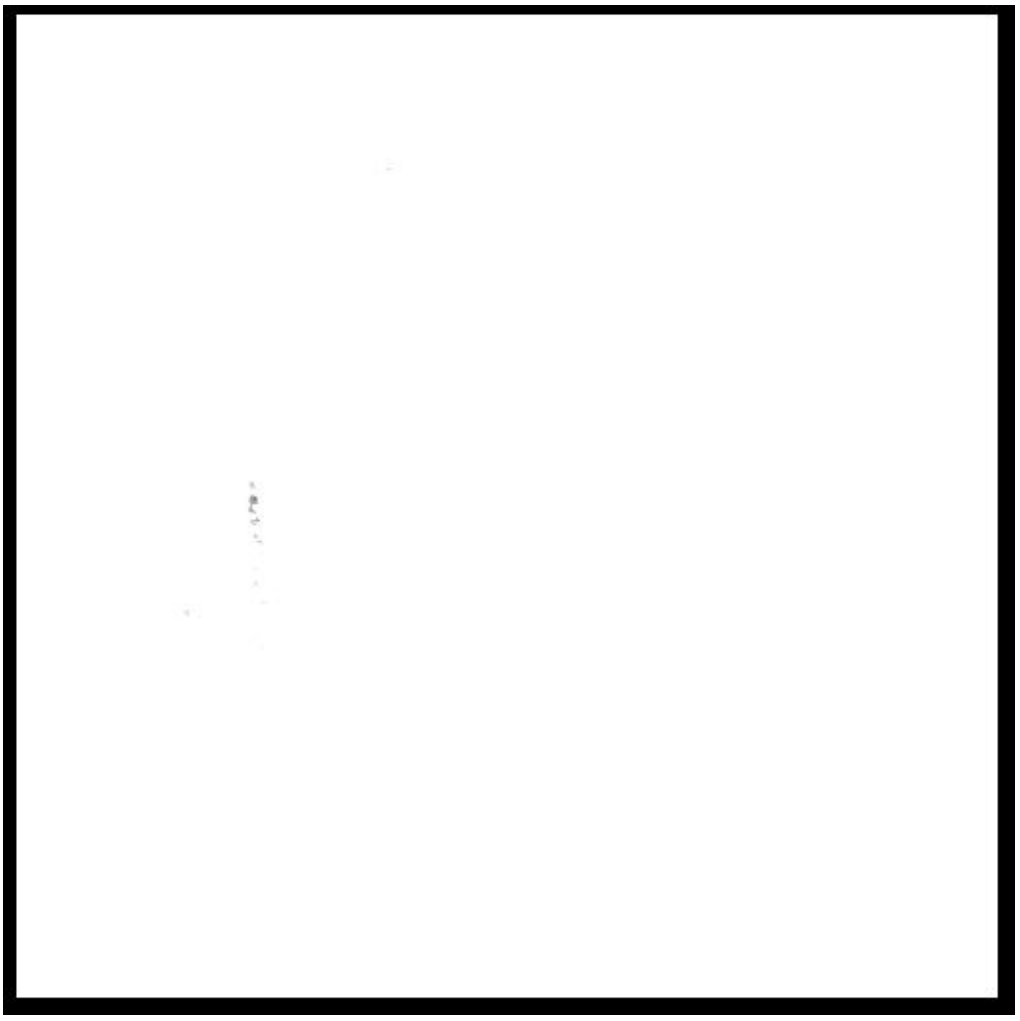


Após o filtro 2



Após o filtro 3

Após o filtro 4



E após o filtro 5, e esta está sendo exibida maior para mostrar uma mancha do lado esquerdo, mostrando que a imagem tem níveis muito altos, mas não é 100% branca.

Conclusões

As aplicações do filtro gaussiano foram de acordo com o esperado com a teoria.
As aplicações dos filtros de domínio espacial foram bem sucedidas e de acordo com o esperado, com exceção da última imagem, por nem ter sido possível a normalização, nem a operação diretamente sobre as imagens resultantes do terceiro e o quarto filtro. Por isso foi feita a operação diretamente sobre os kernel, e provavelmente esse deve ser o problema quando aplicada a convolução do cv2.

Referências

[1]https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.ndimage.fourier_gaussian.html
[2]https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft2.html
[3]https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.ifft2.html
[4]https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html