Trabalho 3 - Introdução ao Processamento de Imagens Digitais

Nome

Victor Palmerini

RA

178061

Data

29/05/2020

Conteúdos

Introdução

Este é o Trabalho 3 da disciplina ***MC920*** - *Introdução ao Processamento de Imagens Digitais* da **Unicamp** - Universidade Estadual de Campinas.

O principal objetivo deste trabalho é desenvolver uma boa noção sobre o filtros `passa-baixa`, `passa-alta` e `passa-faixa` utilizando a *transformada rápida de Fourier*, que possibilita a conversão de imagens digitais para o domínio de frequência.

O trabalho requer por parte do aluno um conhecimento básico da linguagem de programação `python` e bibliotecas que facilitem a manipulação das imagens digitais bem como seus processamentos. No caso deste trabalho, foram usados os pacotes `numpy`, `pillow`, `opencv` e `jupyter notebook`.

Desenvolvimento

Estrutura do Projeto

1. `/notebooks` - é a pasta que contém os notebooks com os algoritmos para serem executados pelo `jupyter notebook`
2. `requirements.txt` - é um arquivo texto que contém as dependências para executar a aplicação (foi gerado pelo pacote `pip` através do comando `pip freeze > requirements.txt`)

3. `Pipfile` e `Pipfile.lock` - são os arquivos relacionados ao `ambiente virtual` aonde as dependências são instaladas e executadas. Isso permite que as dependências sejam instaladas apenas no ambiente virtual e não no ambiente local.

4. `/images` - pasta com algumas imagens usadas como entrada para execução dos notebooks

Rodando Localmente

Dependências

Para executar os notebooks é necessário instalar as seguintes dependências:

1. `Python 3` - neste projeto foi usada a versão `3.7.3`. Qualquer versão do `python` a partir da `3` é suficiente.

2. `Pipenv` - gerenciador de ambientes virtuais. É equivalente ao `virtualenv`.

Inicialização do Ambiente

Na pasta root do projeto, execute os seguintes comandos em uma `shell`:

1. `pipenv shell` - para iniciar um ambiente virtual localmente

2. `pipenv install -r requirements.txt` - instala no ambiente virtual todas as dependências listadas no arquivo `requirements.txt`

Executando os notebooks

Para executar os notebooks, há 2 caminhos:

1. Executar `jupyter notebook` em uma `shell` para subir um servidor do `jupyterlab` que vai abrir automaticamente o navegador padrão com o ambiente `jupyter` e as pastas e arquivos do projeto.

2. Caso o projeto tenha sido aberto no `Visual Studio Code`, uma opção é instalar a extensão `VS Code Jupyter Notebook`, que permite executar os notebooks no próprio `VS Code`.

Entradas

As entradas dos notebooks serão o `path` da imagem de entrada e o `path` da **pasta** de saída. As imagens de entrada utilizadas são *monocromáticas* e no formato `.png`. Como já foi citado, na pasta `/images` há algumas imagens que podem ser usadas como entrada.

Saídas

As saídas também serão imagens *monocromáticas* no formato `.png` com o respectivo filtro aplicado.

Implementação dos Processamentos

Nesse projeto foram implementados 3 filtros diferentes: `passa-baixa`, `passa-alta` e `passa-faixa`. Estes filtros foram aplicados em imagens que passaram pelo processo da transformada de Fourier. A etapa da transformada é igual para todos os notebooks, o que muda é a aplicação do filtro no meio do processo.

Cada processamento é implementado em um `notebook` diferente. Falando um pouco sobre o algoritmo, pode-se dividí-lo nas seguintes etapas:

1. Importação das bibliotecas necessárias
2. Importação da imagem de entrada e definição do path de saída
3. Aplicação da *transformada de Fourier* na imagem
4. Translação da componente de *frequência-zero* para o centro da imagem
5. Definição da *máscara* que será aplicada (define a região do *núcleo*)
6. Aplica-se a *máscara*
7. Volta-se a componente de *frequência-zero* para a posição original
8. Aplica-se a *transformada inversa de Fourier* obtendo-se então a imagem de saída

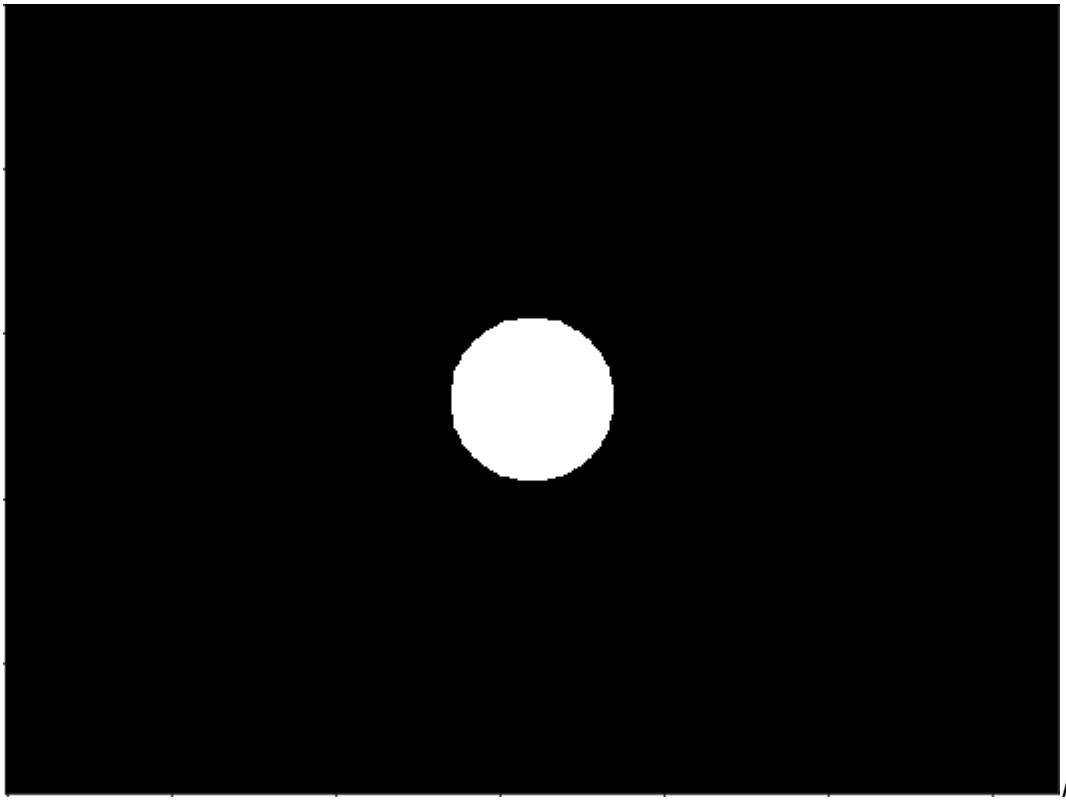
> Em todos os notebooks é mostrado um resultado intermediário da imagem após a transformada de Fourier. Falamos que a imagem foi convertida para o domínio de frequência.

Agora falando um pouco das especifidades de cada filtro:

Passa-Baixa

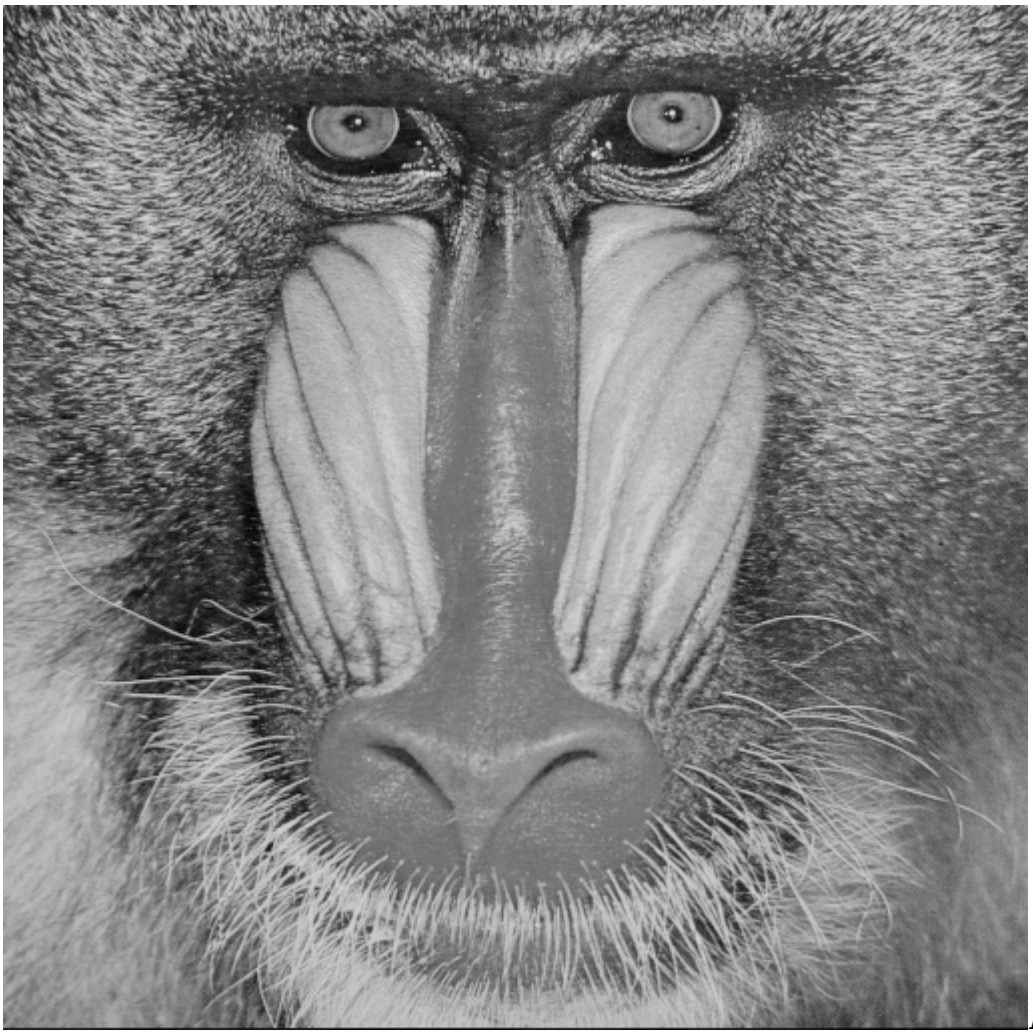
Este é um filtro que atenua sinais com frequências altas e destaca os sinais com frequências baixas. Por sinais de frequência baixa em uma imagem, em geral queremos dizer regiões da imagem com poucas oscilações/variações de cores e contraste, por exemplo.

Neste caso, um exemplo de máscara que pode ser aplicada pra gerar um filtro *passa-baixa* é:



Máscara Passa-Baixa

A partir dessa máscara, aplicada após a transformada de Fourier na imagem de entrada, e a partir da aplicação da transformada inversa de Fourier logo em seguida, obtém-se a imagem de saída com o filtro aplicado. Segue abaixo um exemplo:

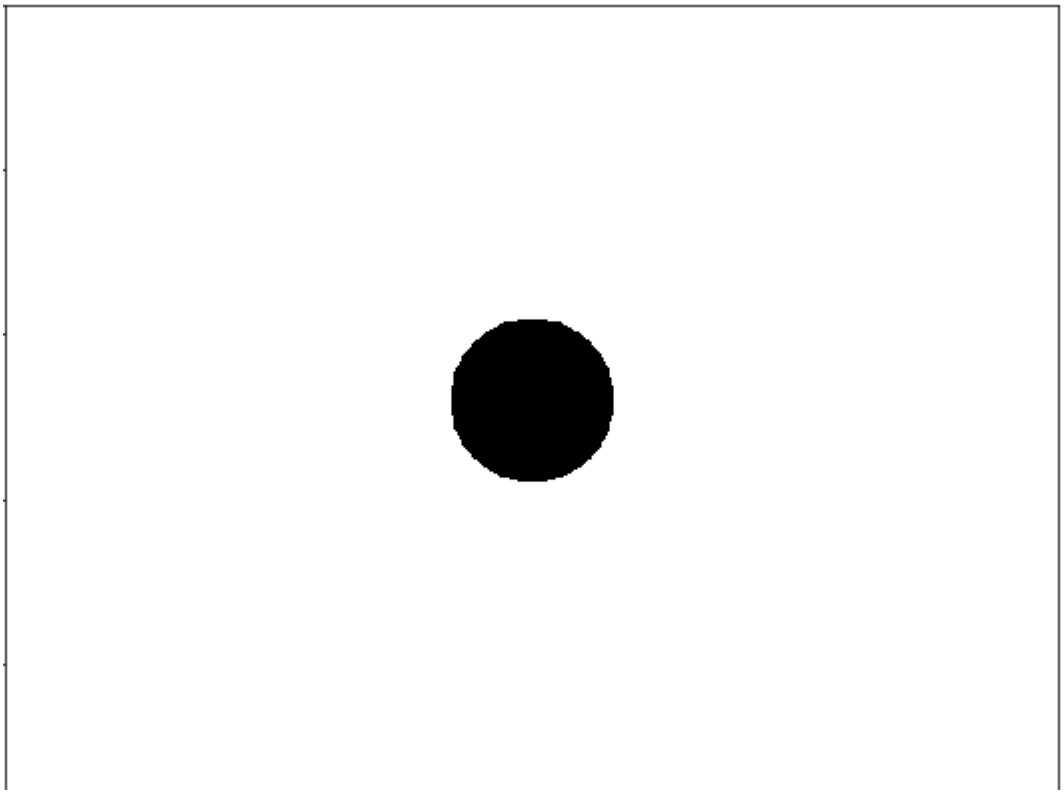
 *Imagem Original*


Filtro Passa-Baixa

Passa-Alta

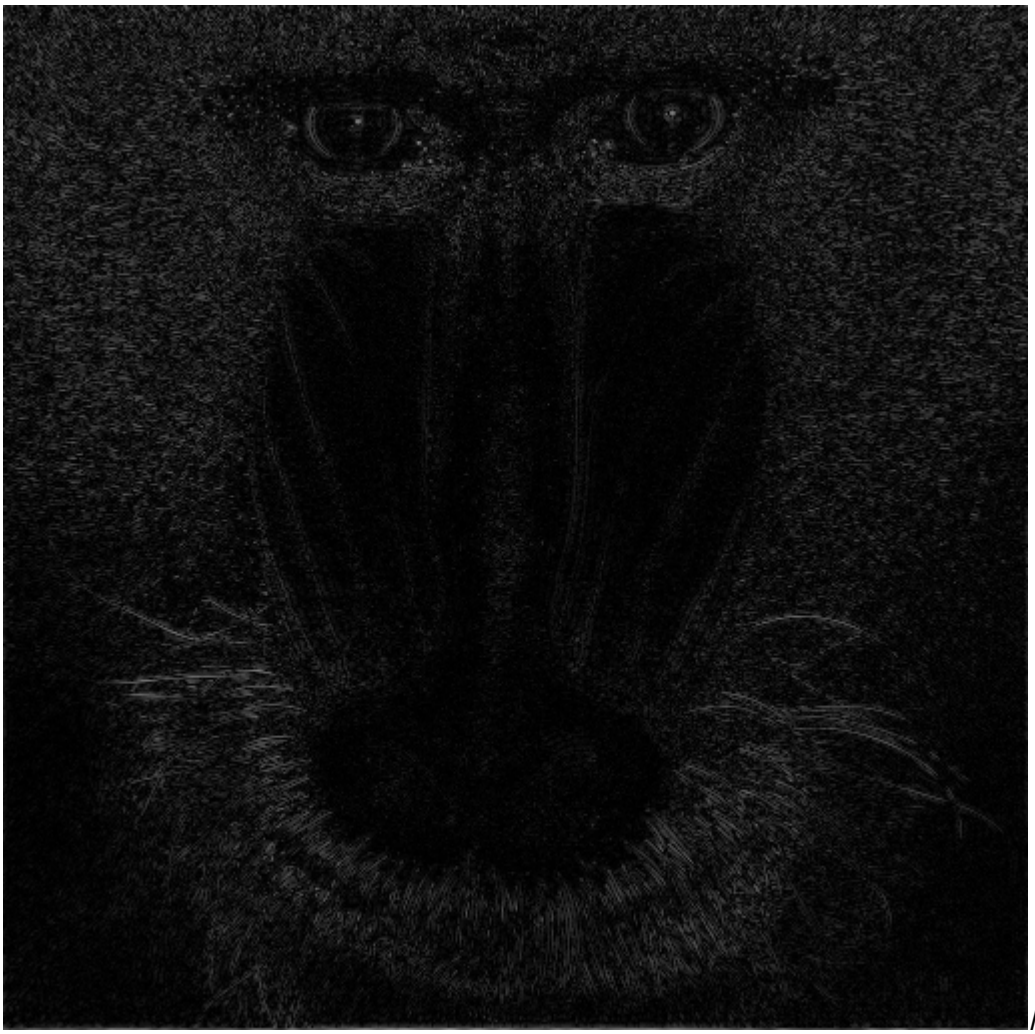
Este é um filtro que atenua sinais com frequências baixas e destaca os sinais com frequências altas. Por sinais de frequência alta em uma imagem, em geral, queremos dizer regiões da imagem com muita oscilação/variação de cores e contraste, por exemplo. A frequência mais alta que se pode obter é quando ocorre uma transição do preto pro branco e vice-versa.

Neste caso, um exemplo de máscara que pode ser aplicada pra gerar um filtro *passa-alta* é:



Filtro Passa-Alta

A partir dessa máscara, aplicada após a transformada de Fourier na imagem de entrada, e a partir da aplicação da transformada inversa de Fourier, obtém-se a imagem de saída com o filtro aplicado. Segue abaixo um exemplo:

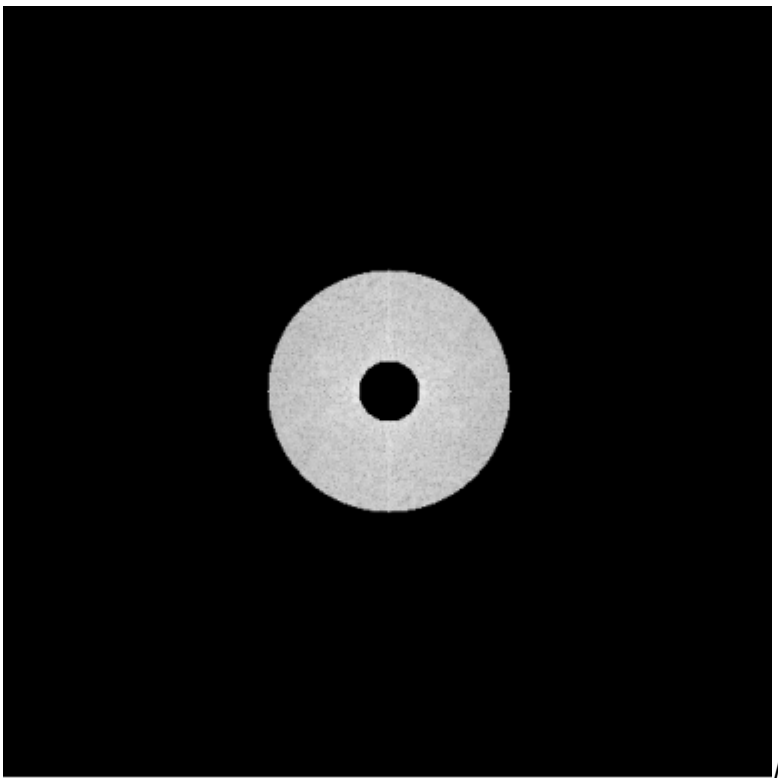


Máscara Passa-Alta

Passa-Faixa

Este é um filtro parecido com o *passa-baixa*, porém a sua máscara é aplicada em uma região um pouco diferente. Enquanto no filtro *passa-baixa* a máscara numa região circular, no filtro *passa-faixa* a máscara é aplicada numa região anelar, isto é, numa região circular mas com um "buraco" no meio (daí o nome `faixa`).

Segue um exemplo de máscara que pode ser aplicada pra gerar um filtro *passa-faixa*:


Máscara Passa-Faixa

A partir dessa máscara, aplicada após a transformada de Fourier na imagem de entrada, e a partir da aplicação da transformada inversa de Fourier, obtém-se a imagem de saída com o filtro aplicado. Segue abaixo um exemplo:


Filtro Passa-Faixa

Análise

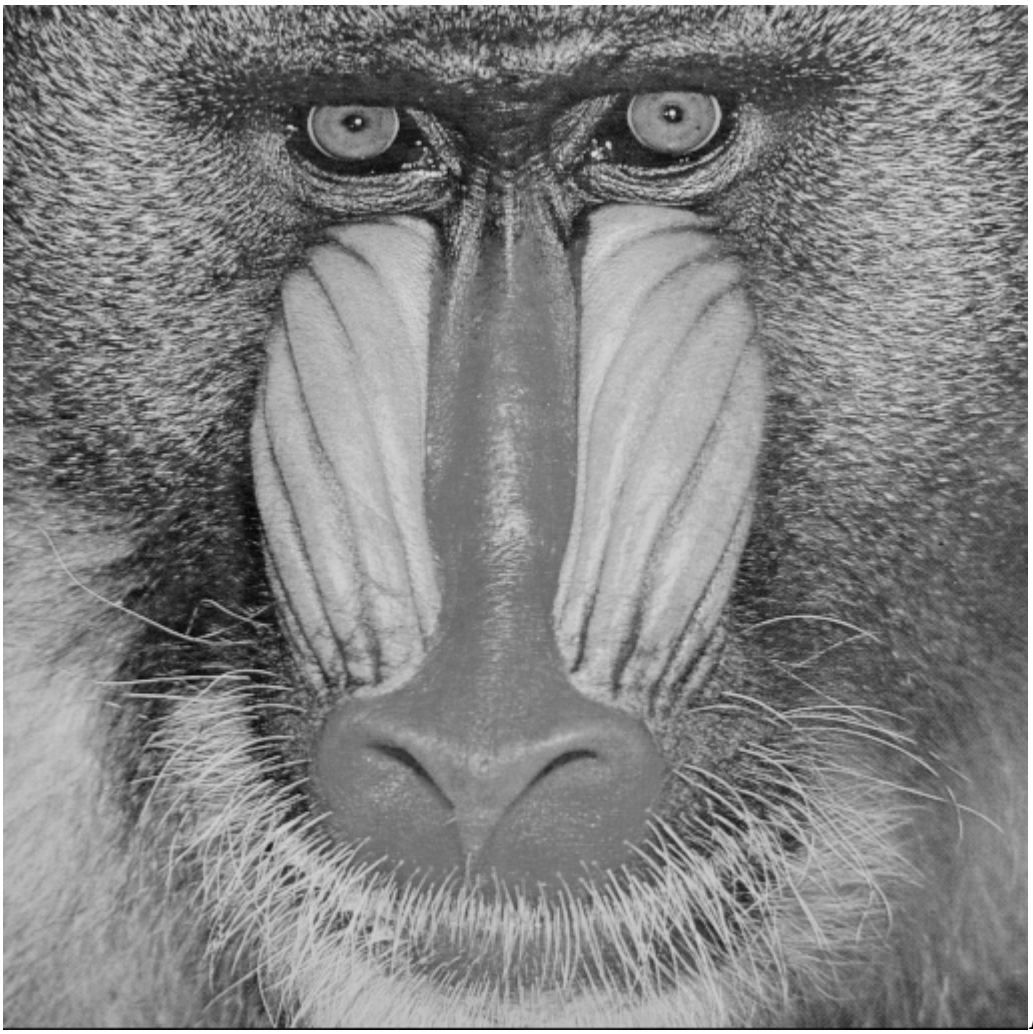
Nesta seção vamos testar algumas variações das máscaras aplicadas na seção anterior e analisar os seus efeitos nas imagens de saída para cada tipo de filtro. Além disso, após a análise será também discutida brevemente uma conclusão sobre as possíveis aplicações do filtro em questão na área de processamento de imagens.

Passa-Baixa

Aqui testamos 3 valores diferentes para `r`, em que `r` é o raio do círculo definido para a máscara do filtro. Quanto maior `r`, maior a área do círculo e vice-versa. No caso deste filtro, a máscara inibe a região externa ao círculo e considera apenas a região interna.

> Lembrando que o "círculo" da máscara é concêntrico em relação à imagem.

Para `r = 10`:

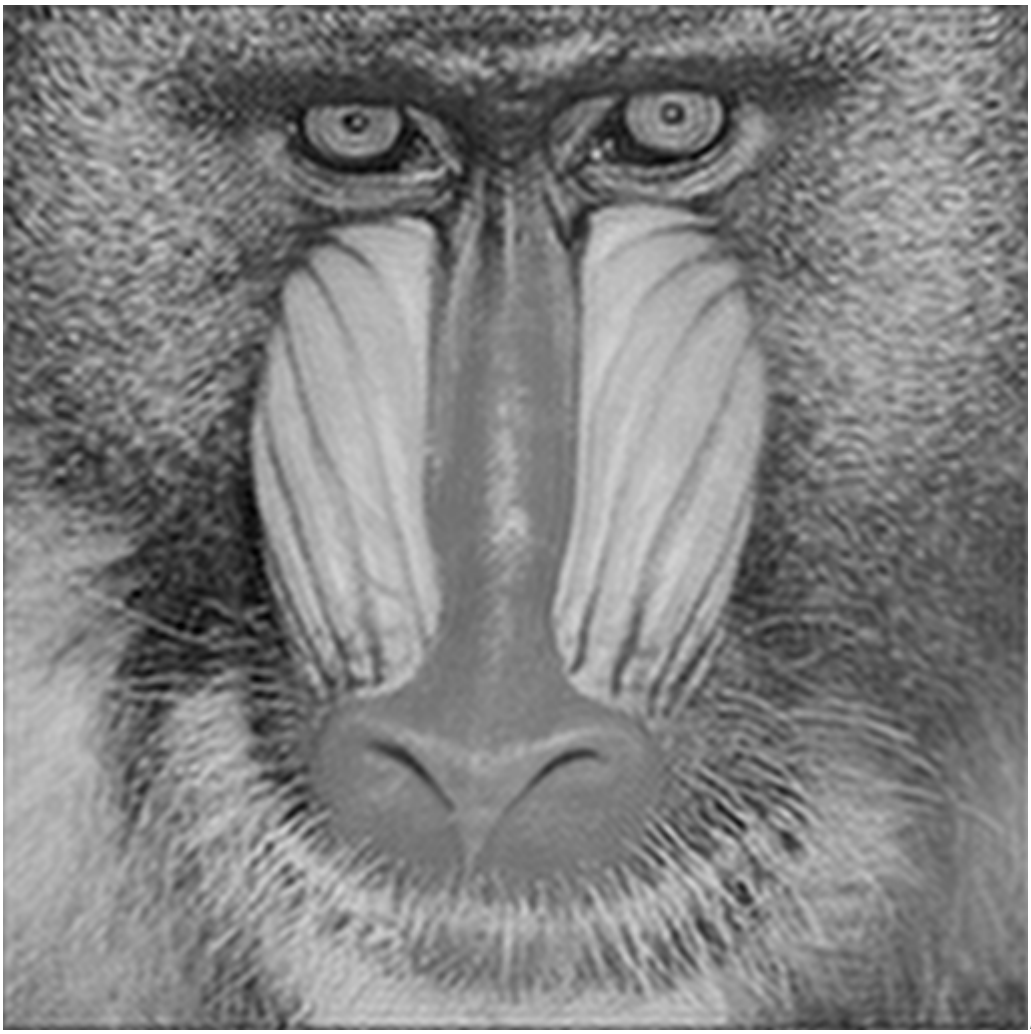
 *Imagem Original*

 *Filtro Passa-Baixa com r = 10*

Para `r = 40`:

 *Filtro Passa-Baixa com r = 40*

Para `r = 80`:

 *Filtro Passa-Baixa com r = 80*

Percebe-se que, conforme o valor de `r` aumenta, a imagem fica cada vez menos "borrada". Isto é intuitivo se pensarmos que o aumento do raio da máscara significa uma região maior de abrangência sobre a imagem original.

Considerando que o filtro `passa-baixa` inibe frequências altas de uma imagem, podemos concluir que este tipo de filtro é interessante para atenuar ruídos/grandes oscilações de cores em imagens digitais.

Passa-Alta

Aqui testamos 3 valores diferentes para `r`, em que `r` é o raio do círculo definido para a máscara do filtro. Quanto maior `r`, maior a área do círculo e vice-versa. No caso deste filtro, a máscara inibe a região interna ao círculo e considera apenas a região externa.

> Foram usados os mesmos valores de `r` para uma possível comparação com o filtro *passa-baixa*

Para `r = 10`:

 *Imagem Original*

 *Filtro Passa-Alta com r = 10*

Para `r = 40`:

 *Filtro Passa-Alta com r = 40*

Para `r = 80`:

 *Filtro Passa-Alta com r = 80*

Pode-se dizer que neste caso quanto maior o valor de `r` mais nítidos ficam os contornos e bordas da imagem. A intensidade do efeito, assim como no filtro anterior, aumenta conforme `r` fica maior. Isso também é esperado já que a área da máscara deste filtro também aumenta conforme se aumenta o valor de `r`.

Considerando que o filtro `passa-alta` inibe frequências baixas de uma imagem e considera apenas frequências altas, podemos concluir que esse filtro é interessante pra destacar regiões em que ocorre grande oscilação de cores como transições entre preto e branco e regiões de borda em imagens digitais.

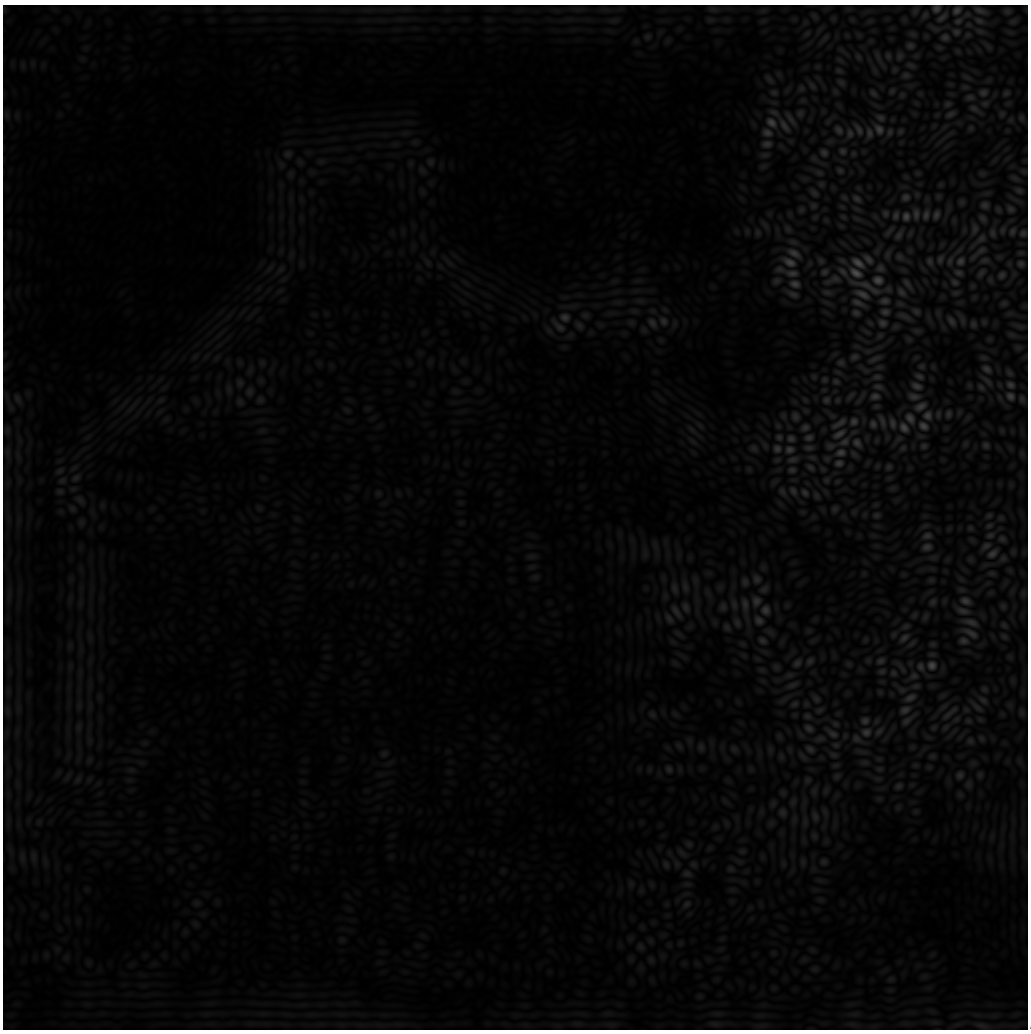
Passa-Faixa

Aqui temos 2 raios, `r1` e `r2`, pois temos 2 círculos. No caso específico deste filtro, a máscara inibe a região externa ao círculo maior e a região interna ao círculo menor. A área considerada então é a que fica entre `r1` e `r2`.

> Lembrando que os "círculos" da máscara são concêntricos entre si e em relação à imagem.

Para `r1 = 60` e `r2 = 40`:

 *Imagem Original*

 *Filtro Passa-Alta com r1 = 60 e r2 = 40*

Para `r1 = 80` e `r2 = 30`:

 *Filtro Passa-Alta com r1 = 80 e r2 = 30*

Para `r1 = 100` e `r2 = 20`:

 *Filtro Passa-Alta com r1 = 100 e r2 = 20*

Olhando para a implementação do filtro de `passa-faixa`, percebe-se que ele parece ser um misto dos 2 filtros anteriores. E na prática é isso que acontece. O filtro `passa-faixa` inibe frequências muito baixas e frequências muito altas e considera faixas intermediárias de frequência.

Portanto, este filtro pode ser usado tanto para atenuar ruídos quanto pra destacar bordas e regiões com grandes transições de cores.

Referências

1. Documentação das bibliotecas usadas
    - OpenCV
    - Pillow
    - NumPy
    - Jupyter Notebook
2. *R.C. Gonzalez, R.E. Woods. Digital Image Processing. Prentice Hall, 2007.*
3. Material de aula fornecido pelo Professor