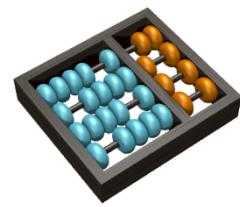




Universidade Estadual de Campinas
Instituto de Computação



Disciplina do 1º Semestre de 2024

IC - UNICAMP

Curso: Bacharelado em Ciência da Computação

MC920 - Introdução ao processamento de imagem digital

Trabalho 2

Alunos: Pedro Barros Bastos **RA:** 204481

Professor: Hélio Pedrini

Campinas – SP
2024

Introdução

Este relatório tem por objetivo descrever o segundo trabalho da disciplina de MC920, que busca aplicar a transformada rápida de Fourier em imagens digitais com o objetivo de realizar o processamento das mesmas no domínio da frequência.

Manipular as imagens no domínio de frequência nos permite modificar seus valores originais para introduzir novas informações. Isso pode ser utilizado para reduzir o ruído, suavizar os dados, aumentar o contraste e destacar detalhes como bordas. Também é possível comprimir imagens removendo coeficientes de Fourier de baixa magnitude.

Neste trabalho foram definidos filtros passa-baixa, passa-alta, passa-faixa e rejeita-faixa, bem como a realização da compressão da imagem removendo-se coeficientes de Fourier menores que um dado limiar.

Arquivos executáveis

- **fftFiltering.py** - script python que irá aplicar os filtros definidos em uma dada imagem.
 - Parâmetros de execução:
 - **--imgFile**: Path da imagem onde os filtros serão aplicados
 - **--r1**: Raio do maior círculo a ser criado nas máscaras
 - **--r2**: Raio do menor círculo a ser criado nas máscaras
 - **--compressThreashold**: Inteiro a ser considerado como limiar para compressão. Todos os coeficientes da transformada rápida de Fourier que estiverem abaixo deste limiar serão zerados.

Obs: Foi deixado um arquivo texto chamado **EXEMPLOS-EXECUCAO.txt** com exemplos de possíveis execuções dos scripts python.

Pacotes utilizados

- **OpenCV** - v4.6.0
- **Numpy** - v1.26.4
- **argparse** - v1.4.0

Obs: Foi utilizada a ferramenta **conda** para construção do ambiente de desenvolvimento para este trabalho, bem como utilização de SO **Ubuntu 22.04.4 LTS** inicializado em uma virtualbox.

Desenvolvimento Filtragem

A realização deste trabalho se deu a partir do uso de funções prontas para obtenção da transformada rápida de Fourier dadas pelo OpenCV.

Primeiro foi efetuada a leitura da imagem em níveis de cinza. Depois foi utilizada a função *cv2.dft* para obtenção da Discrete Fourier Transform. Com o uso da função *fft.fftshift* do pacote numpy foi possível centralizar o espectro de frequência, fazendo com que as menores frequências fossem centralizadas e maiores frequências mais distantes do centro. Depois foi criada uma lista com todos os filtros a serem aplicados na imagem de entrada: filtro passa-baixa, filtro passa-alta, filtro passa-faixa e filtro rejeita-faixa. Tais filtros foram criados a partir da criação de uma máscara de mesmo tamanho da imagem de entrada com um ou mais círculos (a depender do filtro em questão) desenhados no centro da imagem e com fundo correspondente.

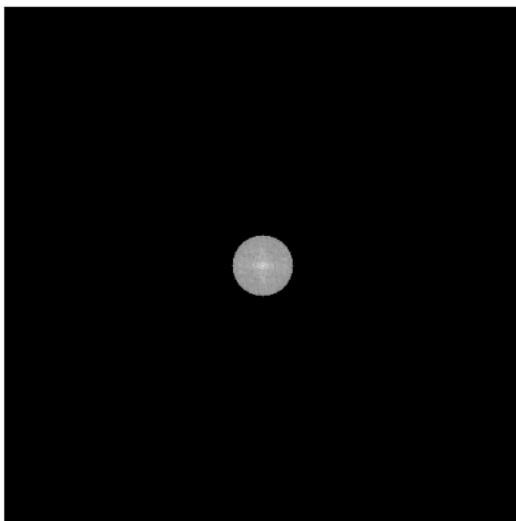
Por exemplo, no filtro passa-baixa, onde queremos manter apenas as baixas frequências e eliminar as maiores frequências, é usada como máscara uma matriz de zeros com um círculo branco desenhado no meio para que na multiplicação da máscara pela FFT da imagem apenas as menores frequências permaneçam (círculo central). De forma análoga, um filtro passa-alta, onde deseja-se manter as maiores frequências, cria-se como máscara uma matriz de pixels brancos com um círculo preto desenhado no meio para que na multiplicação da FFT da imagem com a máscara apenas permaneçam as maiores frequências (conteúdo fora do círculo central).

Dos resultados das multiplicações mencionadas acima, usam-se respectivamente as funções *np.fft.ifftshift* e *cv2.idft* para podermos voltar para o domínio do espaço a partir do domínio de frequência gerado pela FFT. A função *np.fft.ifftshift* realiza a volta da centralização do espectro da frequência para sua localização original e a função *cv2.idft* realiza a operação inversa da Discrete Fourier Transform. Além dessas funções, foi utilizada a função *cv2.normalize* para obter os componentes do módulo da inversa em números de 0 a 255 com o intuito de poder visualizar o resultado das filtragens realizadas em níveis de cinza.

Resultados filtragem

- Filtro Passa-Baixa

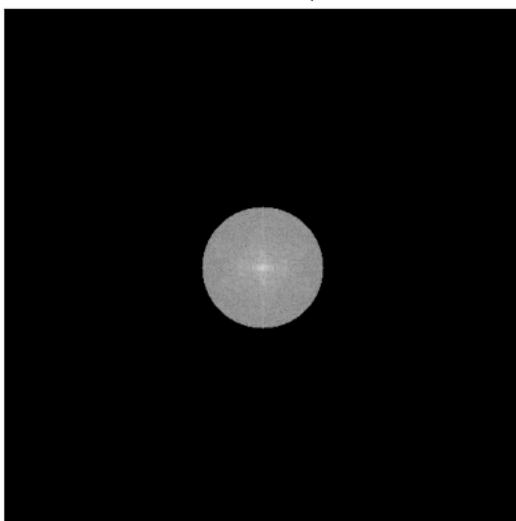
Filtro Passa Baixa, r1 = 30



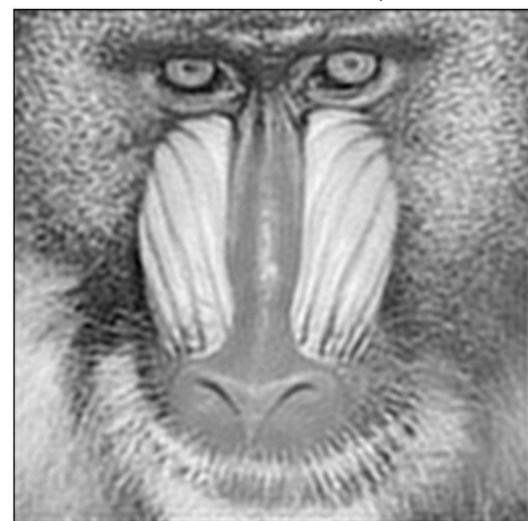
Resultado Filtro Passa Baixa, r1 = 30



Filtro Passa Baixa, r1 = 60



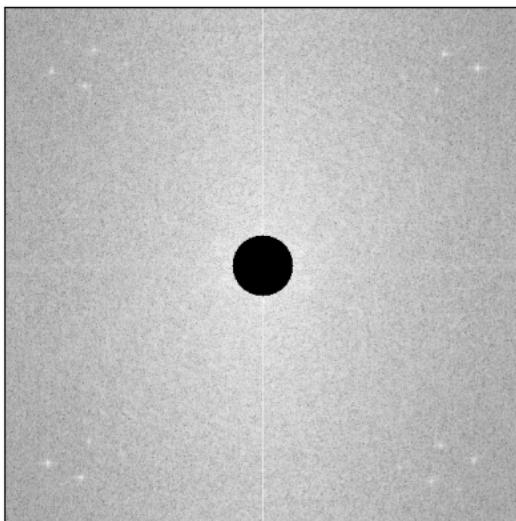
Resultado Filtro Passa Baixa, r1 = 60



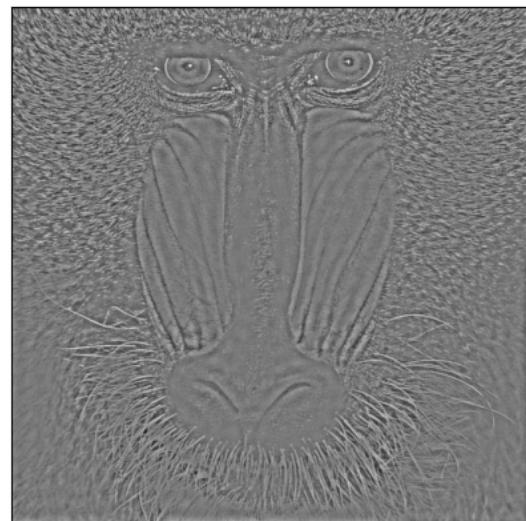
Ao aplicar um filtro passa-baixa na imagem de entrada, podemos perceber um certo borramento da mesma, atenuando alguns detalhes como bordas e ruídos. Percebe-se também que quanto maior o raio do filtro, mais frequências de maior valor acabam sendo aceitas a partir da máscara, atenuando o efeito de borramento.

- Filtro Passa-Alta

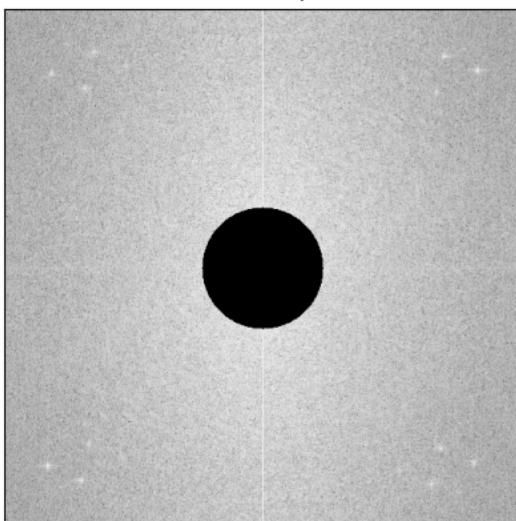
Filtro Passa Alta, r1 = 30



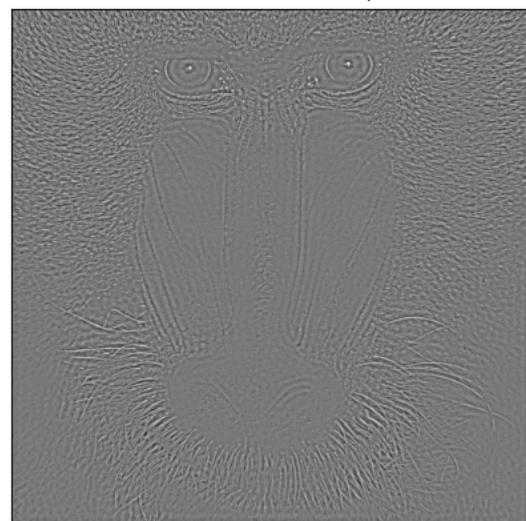
Resultado Filtro Passa Alta, r1 = 30



Filtro Passa Alta, r1 = 60



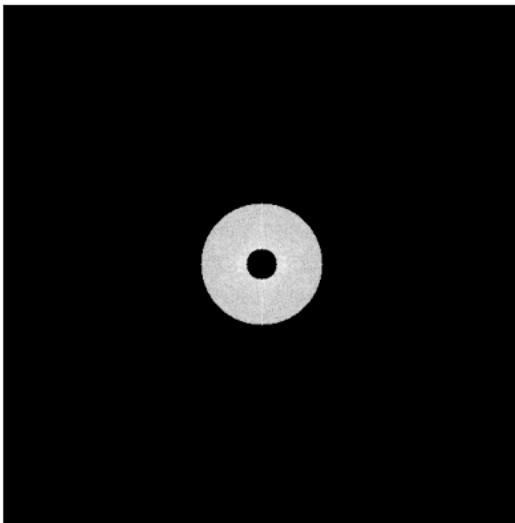
Resultado Filtro Passa Alta, r1 = 60



Já ao aplicar filtros passa-alta, habilitando a passagem de conteúdos de mais alta frequência, vemos nos resultados que componentes como retas, bordas e ruídos foram preservados, enquanto que regiões com características contínuas (texturas, cor de pele, etc) não aparecem. Com o aumento do raio da máscara a fim de manter componentes de ainda maior frequência, vemos que a quantidade de ruídos e bordas é diminuída. Ou seja, o threshold do que seria uma borda ou um ruído acaba sendo elevado e menos componentes de tais características são preservados.

- Filtro Passa-Faixa

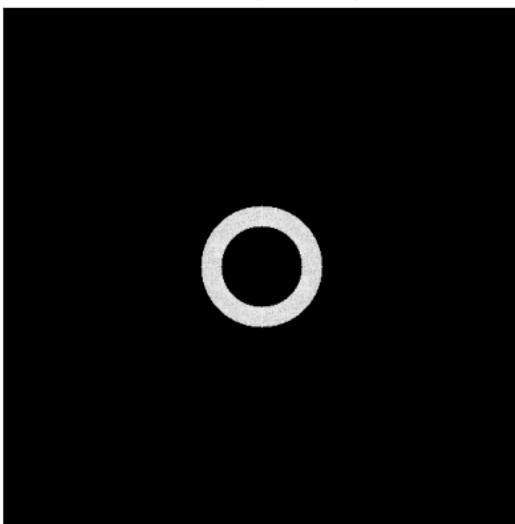
Filtro Passa Faixa, $r1 = 60$, $r2 = 15$



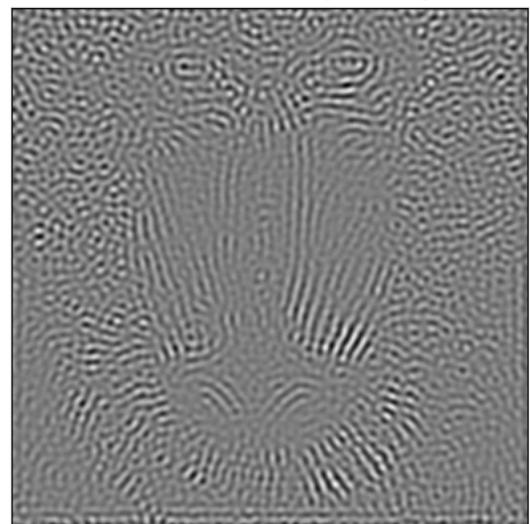
Resultado Filtro Passa Faixa, $r1 = 60$, $r2 = 15$



Filtro Passa Faixa, $r1 = 60$, $r2 = 40$



Resultado Filtro Passa Faixa, $r1 = 60$, $r2 = 40$

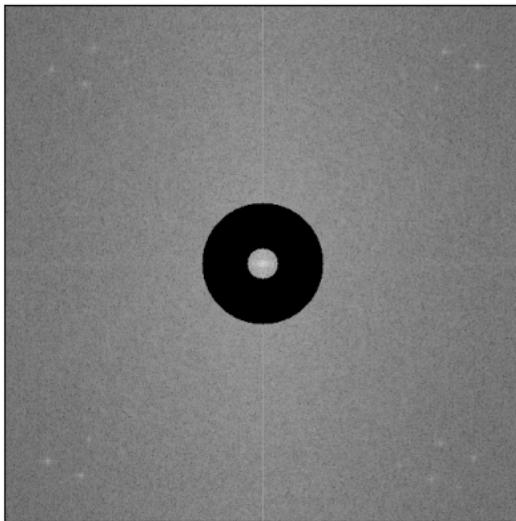


Nos filtros passa-faixa, onde especificamos através da região entre o menor e o maior círculo quais frequências queremos manter, vemos uma espécie de mescla entre os dois filtros anteriores, onde ao mesmo tempo há um borramento da imagem provocado pelo círculo maior, mantendo frequências até determinado ponto e uma determinação de bordas e ruídos determinado pelo círculo menor. Na primeira imagem acima, onde a região de diferença entre os dois círculos é maior, o resultado apresenta maiores detalhes. Já na segunda imagem, onde a diferença entre os dois círculos foi drasticamente reduzida, vê-se que o borramento se manteve (círculo maior)

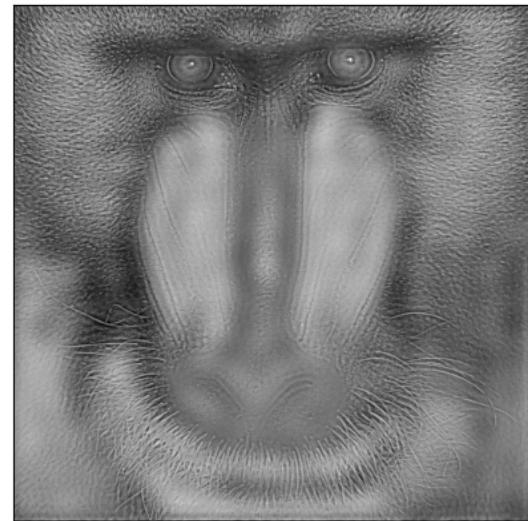
mas o que foi considerado como borda e ruído foi muito reduzido (círculo menor).

- Filtro Rejeita-Faixa

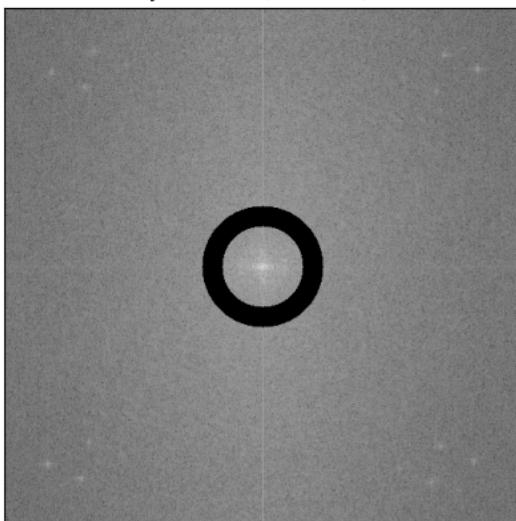
Filtro Rejeita Faixa, $r1 = 60$, $r2 = 15$



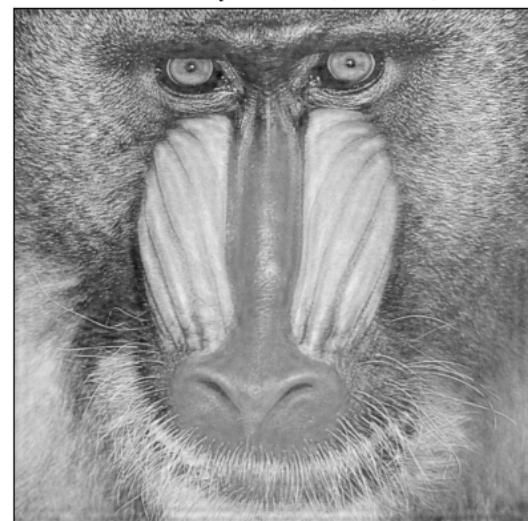
Resultado Filtro Rejeita Faixa, $r1 = 60$, $r2 = 15$



Filtro Rejeita Faixa, $r1 = 60$, $r2 = 40$



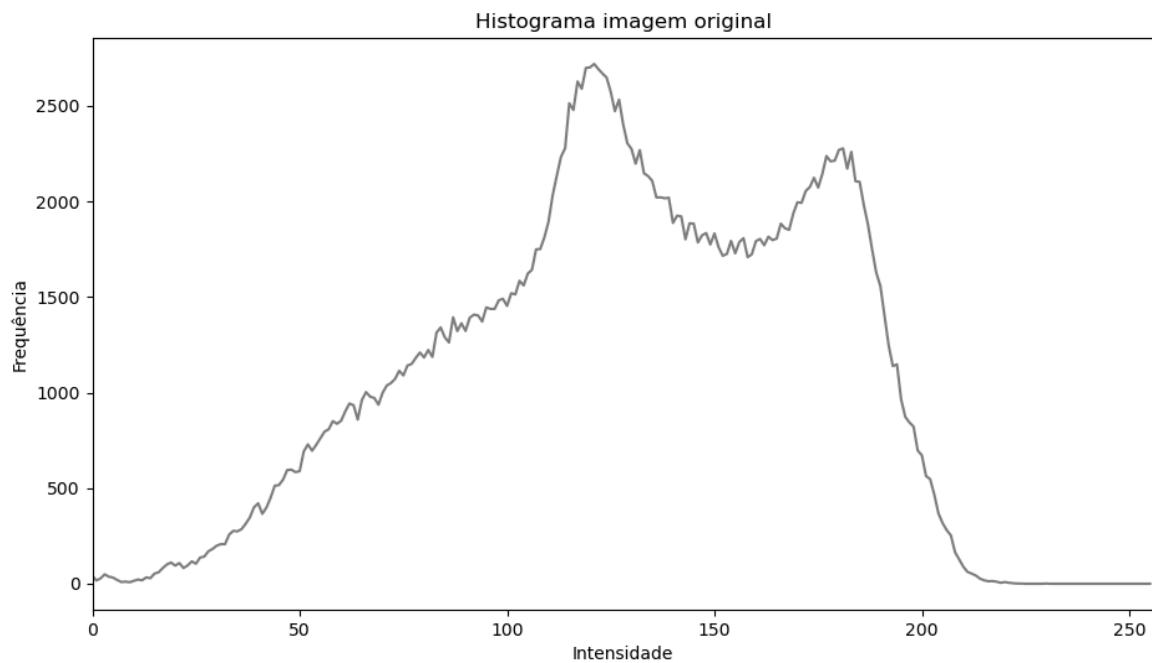
Resultado Filtro Rejeita Faixa, $r1 = 60$, $r2 = 40$



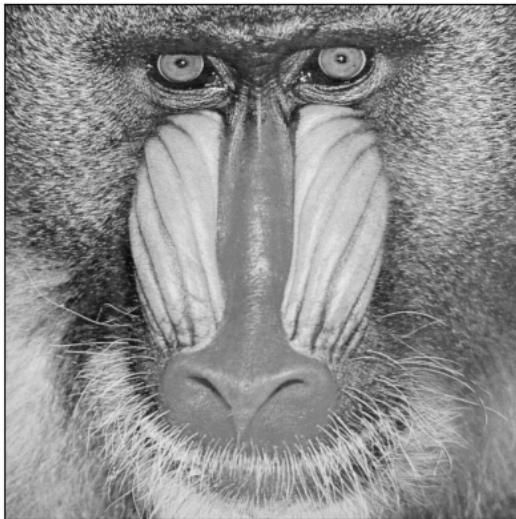
Por último, considerando os filtros rejeita-faixa, excluímos as frequências indesejadas. Na primeira imagem podemos observar um bom nível de borramento juntamente com o evidenciamento de certas bordas. Na segunda imagem, onde a diferença entre os dois círculos foi diminuída bastante, vemos uma imagem praticamente igual à original mas com certos níveis de bordas e ruídos mais evidenciados.

Compressão

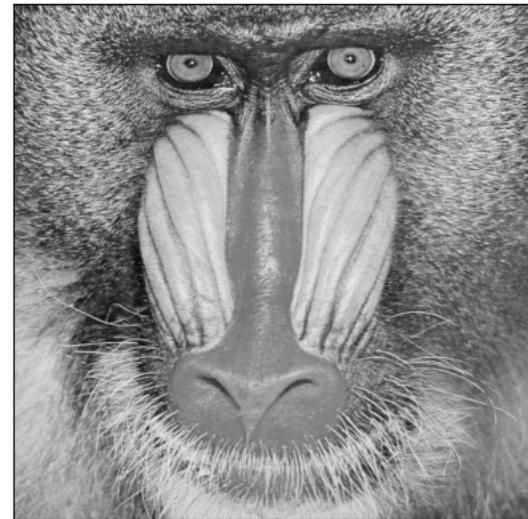
A compressão da imagem foi realizada zerando os coeficientes menores, em valores absolutos, que o limiar passado por parâmetro na execução do script. Tal operação foi realizada tanto nos componentes de módulo quanto nos componentes de fase da FFT. Como podemos observar nos resultados abaixo, apenas a partir de um threshold de 10000 alguma diferença na imagem final começou a ser percebida. Assim, vemos que temos uma boa margem de remoção de dados da FFT com o fim de comprimir uma imagem.



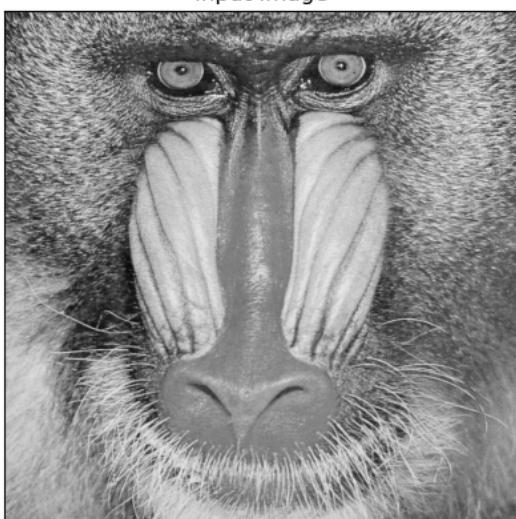
Input Image



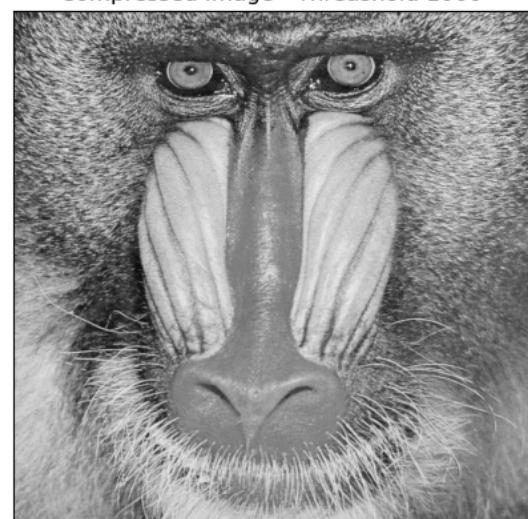
Compressed Image - Threshold 100



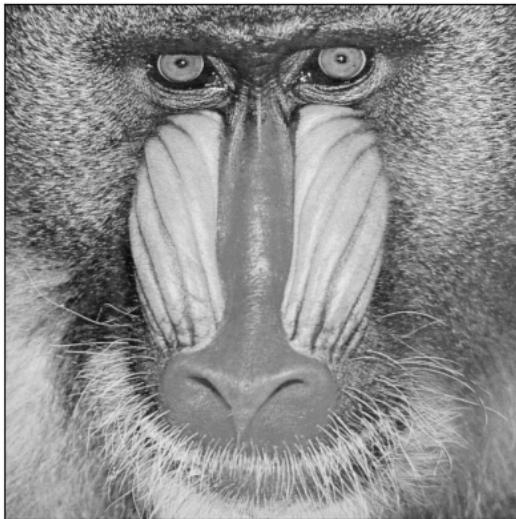
Input Image



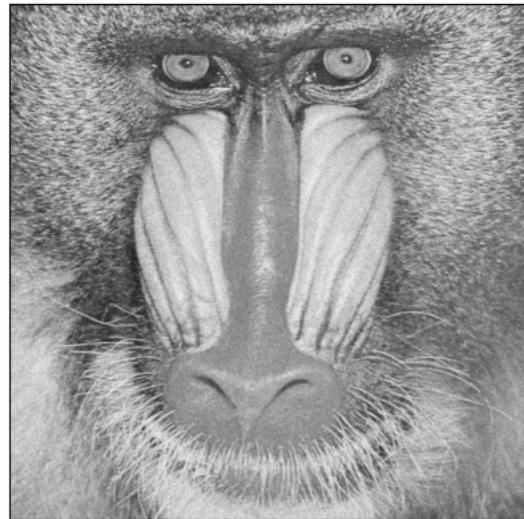
Compressed Image - Threshold 1000



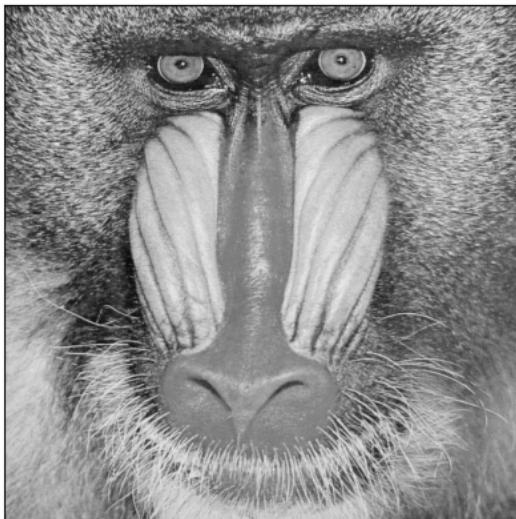
Input Image



Compressed Image - Threshold 10000



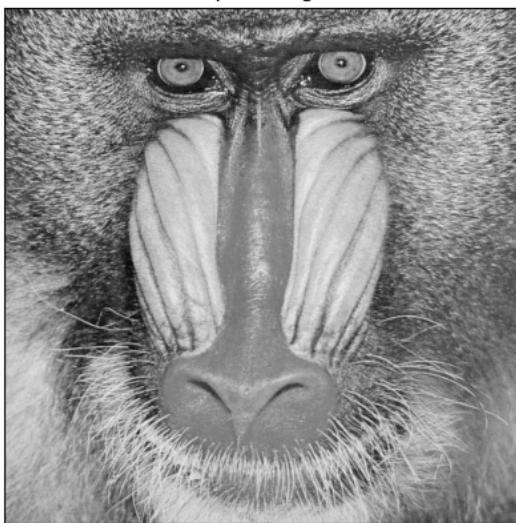
Input Image



Compressed Image - Threshold 50000



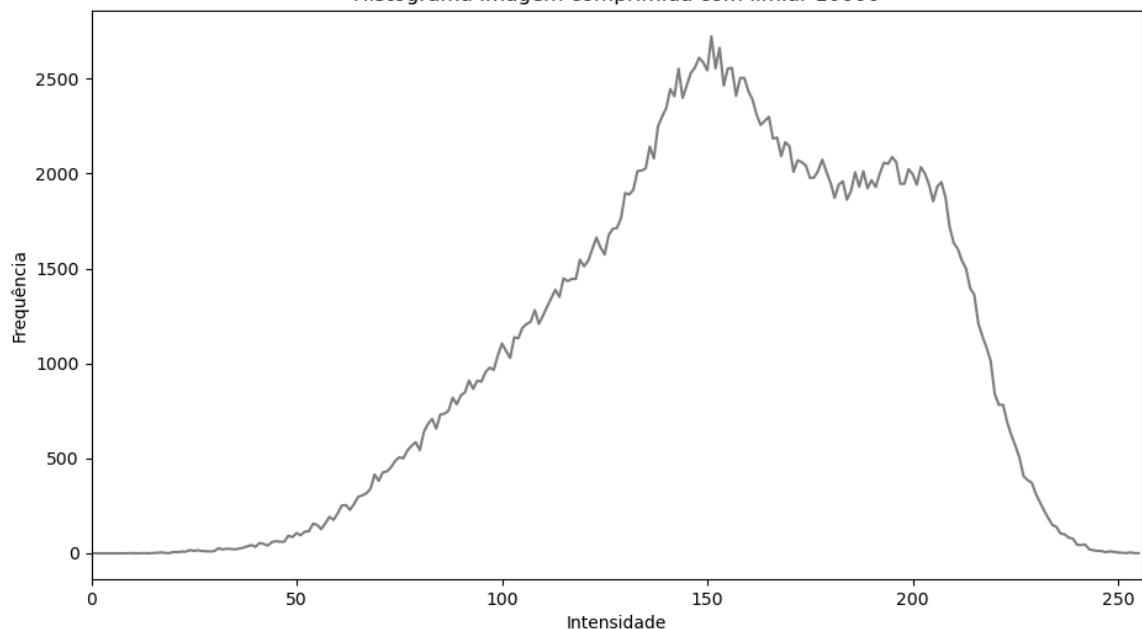
Input Image



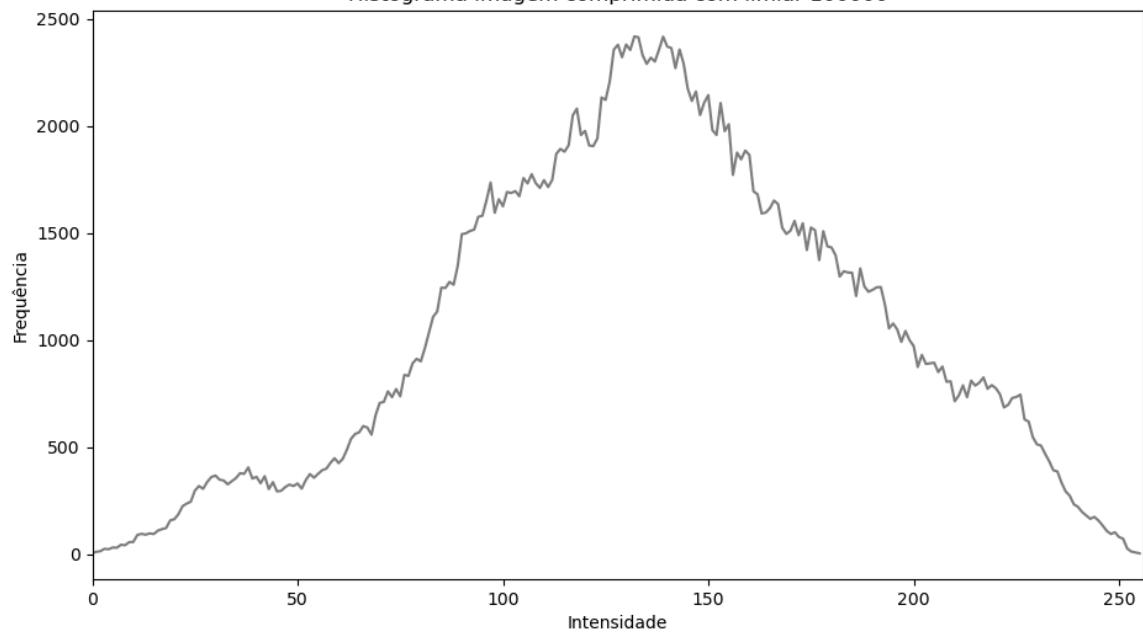
Compressed Image - Threshold 100000



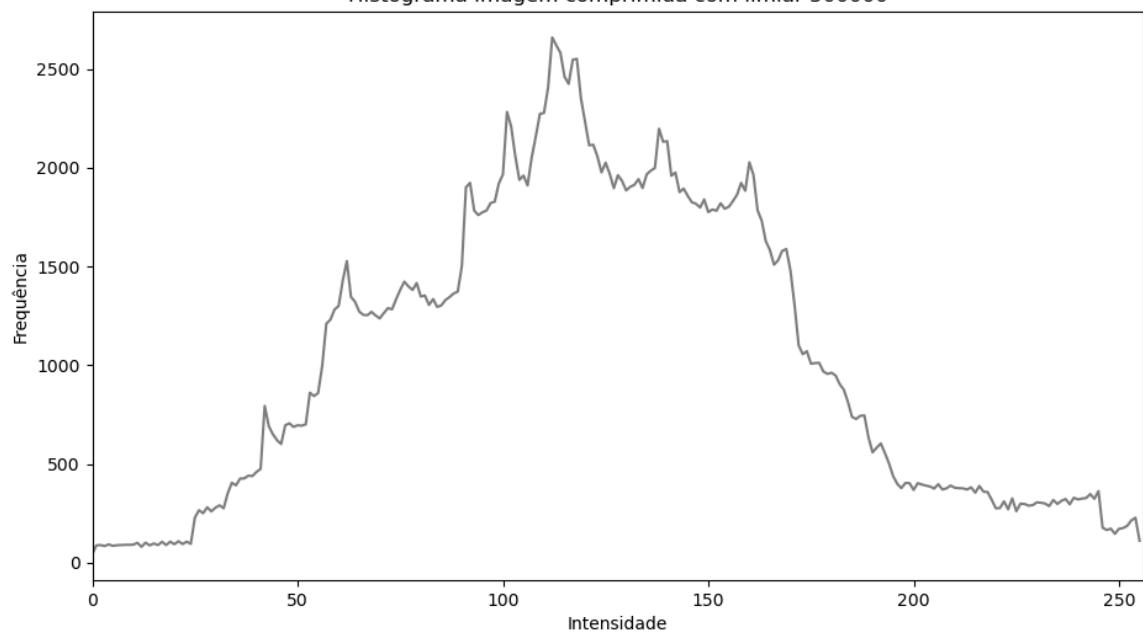
Histograma Imagem comprimida com limiar 10000



Histograma Imagem comprimida com limiar 100000



Histograma Imagem comprimida com limiar 500000



Durante a implementação da compressão foi observado que o mero fato de se aplicar a FFT e logo depois a sua inversa e salvar a imagem resultante da inversa em disco já foi capaz de reduzir o tamanho da imagem para menos de um terço do tamanho da imagem original.

```
(conda-env) (base) pedro.bastos@pb02:mc920$ ls -la images/baboon.png
(conda-env) (base) pedro.bastos@pb02:mc920$ -rw-r--r-- 1 pedro.bastos domain^users 637192 abr 25 13:16 images/baboon.png
(conda-env) (base) pedro.bastos@pb02:mc920$
```

Imagen original. Tamanho de 637,2 kB

```
(conda-env) (base) pedro.bastos@pb02:mc920$ ls -la trabalho2/compressed-threshold_0.png
(conda-env) (base) pedro.bastos@pb02:mc920$ -rw-r--r-- 1 pedro.bastos domain^users 213382 abr 25 17:45 trabalho2/compressed-threshold_0.png
(conda-env) (base) pedro.bastos@pb02:mc920$
```

Imagen “comprimida” com um threshold de 0 (sem remoção de coeficientes). Tamanho de 213,4 kB

Excetuando-se o observado acima, verificou-se que para observar um bom nível de redução do tamanho da imagem precisou-se aumentar muito o threshold, fazendo com que ainda mais coeficientes da FFT fossem removidos.

```
(conda-env) (base) pedro.bastos@pb02:mc920$ ls -la trabalho2/compressed-threshold_1000.png
(conda-env) (base) pedro.bastos@pb02:mc920$ -rw-r--r-- 1 pedro.bastos domain^users 213065 abr 25 17:46 trabalho2/compressed-threshold_1000.png
(conda-env) (base) pedro.bastos@pb02:mc920$
```

Imagen comprimida com um threshold de 1000. Tamanho de 213,1 kB

```
(conda-env) (base) pedro.bastos@pb02:mc920$ ls -la trabalho2/compressed-threshold_10000.png
(conda-env) (base) pedro.bastos@pb02:mc920$ -rw-r--r-- 1 pedro.bastos domain^users 198818 abr 25 17:47 trabalho2/compressed-threshold_10000.png
(conda-env) (base) pedro.bastos@pb02:mc920$
```

Imagen comprimida com um threshold de 10000. Tamanho de 198,8 kB

```
(conda-env) (base) pedro.bastos@pb02:mc920$ ls -la trabalho2/compressed-threshold_500000.png
(conda-env) (base) pedro.bastos@pb02:mc920$ -rw-r--r-- 1 pedro.bastos domain^users 72439 abr 25 17:48 trabalho2/compressed-threshold_500000.png
(conda-env) (base) pedro.bastos@pb02:mc920$
```

Imagen comprimida com um threshold de 500000. Tamanho de 72,4 kB

No caso da compressão utilizando threshold 500000 a imagem de saída praticamente não tem semelhança alguma com a imagem original.

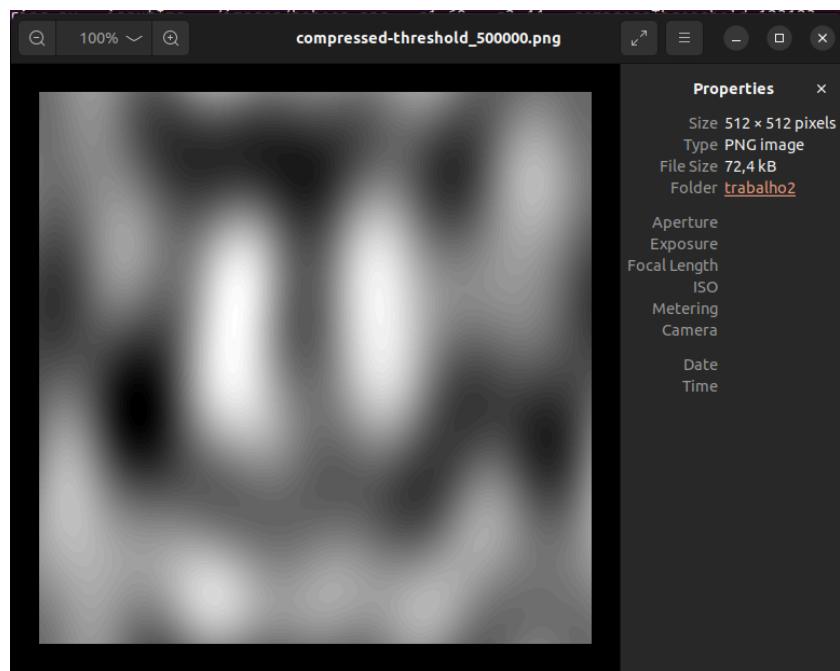


Imagen comprimida com um threshold de 500000. Tamanho de 72,4 kB