MC920/MO443 - Trabalho 2

Arthur Rezende 166003

1 Introdução

Este documento é um relatório contendo descrições e resultados do programa feito para o trabalho 4. Nele foram exploradas tecnicas de projeção perspectiva, rotação e escala. Para o último foram explorados quatro diferentes metódos de interpolação sendo eles: vizinho mais próximo, bilinear, bicúbica e polinômios de lagrange.

Todos foram implementados *from scratch*, isto é, do "zero". Utilizando somente a biblioteca *NumPy*. O resultado da projeção perspectiva foi comparado com o da biblioteca *OpenCV*. Quanto as técnicas de interpolação, foram comparadas entre si.

Este documento está divido em 2 áreas, Implementação e Discussão. Na primeira, serão discutidas detalhes algorítmicos das implementações feitas tal como vetorização. Na segunda, discussão dos efeitos visuais causados.

2 Implementação

2.1 Projeção Perspectiva

Para aplicar a transformação de projeção perspectiva em uma imagem é necessário encontrar a matriz de transformação correspondente e multiplicar com a imagem em questão. Transpondo os pontos originais conforme a matriz encontrada. Que pode ser mapeado em resolver a seguinte equação:

$$\begin{bmatrix} WX' \\ WY' \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ i & j & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{1}$$

Onde os termos correspondem a imagem transformada, matriz de correspondência e imagem original, respectivamente.

Por possuir 8 incógnitas, o sistema linear da forma $Ax = B$, terá 8 entradas, sendo os componentes X e Y dos oito pontos. Na forma:

$$\begin{vmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0u_0 & -y_0u_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0v_0 & -y_0v_0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 \end{vmatrix} \cdot \begin{vmatrix} a \\ b \\ c \\ d \\ e \\ f \\ i \\ j \end{vmatrix} = \begin{vmatrix} u_0 \\ u_1 \\ u_2 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{vmatrix} \tag{2}$$

Onde os pontos $(x_i,\, y_i)$ correspondem aos de entrada e $(u_i,\, v_i)$ aos de saída. Note que esta matriz é a versão 8x8 da multiplicação de um ponto da equação 1.

$$X' = \frac{aX + bY + c}{iX + jY + 1} \tag{3}$$

O que a torna computacionalmente de mais fácil resolução, após encontrada a mesma é necessário dividir pela última linha tendo em vista a eliminação do coeficiente W. Então basta resolver após a correção das dimensões, via adição do eixo Z.

Para o passo final, foi necessário utilizar a inversa da matriz encontrada, uma vez que com ela o resultado era a adição de perspectiva da imagem e não o retorno do baboon a sua glória, forma e formato original.

Como todo o procesos foi feito com vetorização de matrizes via *NumPy* a solução funciona para imagens com múltiplos canais também. Exemplos de imagens são as figuras 1 e 3.

3 Transformações

3.1 Escala

A operação de escala foi implementada vetorialmente com o numpy. Primeiro, cria-se uma nova imagem com o tamanho correto feito pela escala (seja ela de resolução ou de fator único). Após isso a manipulação dos indicies será feita por alguma das funções de interpolação. Resultando numa imagem maior ou menor preenchida por interpolação.

3.2 Rotação

Para a operação de rotação, foi implementado vetorialmente a equação:

$$R(\theta) = \begin{vmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{vmatrix}$$

$$\mathrm{T}(\mathrm{X}_1, Y_1)R(\theta)T(-X_1, -Y_1)(4)$$

Porém com a observação de que as coordenadas X e Y devem ser invertidas por estarmos tratando como Y o primeiro elemento do par ordenado. Entretanto para aplicar a rotação é necessário primeiro levar o ponto ao zero, efetuar a rotação e depois retornar as coordenadas originais.

Feito isso os elementos são clipados para assegurar o valor dentre as dimensões da imagem. Exemplos de rotação são as imagens 4 e 5.

3.3 Interpolação

3.3.1 Vizinho mais próximo

Para esta interpolação foi utilizado somente os indicies aumentados da função base de escala. Dividindo-os e utilizando de um type cast para inteiro obtemos todos os valores interpolados para a nova imagem. Assim escalando seu tamanho.

3.3.2 Bilinear

Para implementar a interpolação bilinear, utiliza dos mesmos princípios iniciais. Entretanto após a criação da nova matriz com indicies é calculado uma recriação com as imagens originais, afim de calcular duas novas matrizes (dx, dy) para o calculdo das novas posições. Então os valores originais são clipados para garantir compatibilidade dimensional considerando que para o dx e dy os valores devem ter um espaço antes da borda da imagem.

Assim com os vetores, combina-se as novas posições com um com o tamanho dos respectivos intervalos, neste caso 2, gerando uma imagem nova com tamanho $(Y, X, 4)$ onde Y e X correspondem ao tamanho da nova imagem já escalada e assim aplicando a equação 5, gerando a imagem interpolada pelo método bilinear.

$$f(x', y') = (1dx)(1dy)f(x, y) + dx(1dy)f(x+1, y) + (1dx)dyf(x, y+1) + dxdyf(x+1, y+1) \tag{5}$$

Este último passo foi feito utilizando um laço for. Pois seriam somente 4 iterações com operações constantes, o que não impactaria a performance do algoritmo.

3.3.3 Bicúbica

Para implementação da interpolação bicúbica, um procedimento similar a bilinear foi utilizado. Após assegurar a compatibilidade dimensional, se divide a matriz em nove blocos , $(Y, X, 9)$ e assim com auxílio das funções R e P, se calcula a soma dos indicies das mesmas conforme a sua posição. Considerando as equações:

$$f(x', y') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} f(x+m, y+n) R(m-dx) R(dy-n) \tag{6}$$

$$R(s) = \frac{1}{6}[P(s+2)^3 - 4P(s+1)^3 + 6P(s)^3 - 4P(s-1)^3]$$

$$P(t) = max(t, 0)$$

Assim como no último método, o último passo foi aplicado iterativamente.

3.3.4 Polinômios de Lagrange

Analisando a implementação deste método, ela se torna muito similar a da bicúbica, bastando apenas mudar a função de interpolação numérica utilizada. Entretanto é possível perceber que sua formula requer menos iterações, uma vez que consegue ser vetorizada com mais eficiência na parte de cálculo dos indicies correspondentes, facilitando a portabilidade da implementação.

4 Discussão

Os resultados para transformação perspectiva foram satisfatórios. Considerando a diferença entre a matrizes na tabela 1 percebe-se que só existe erros de arredondamento em torno da 13 casa decimal.

Tabela 1: Diferenças entre as matrizes do OpenCV e Caseira

0	0	5.7e-14
1.0e-15	-1.0e-15	4.3e-14
0	0	0

Agora, analisando a imagem 1 existe pouca diferença visual entre os baboons. Na versão caseira, existe uma linha preta mais visível no final da imagem, ainda assim a diferença é pouco perceptível visualmente. Na figura 2, é possível visualizar a diferença entre a imagem caseira e a da biblioteca. Também foram contabilizados os pixeis não nulos desta diferença, contabilizando cerca de 2% de diferença.

Também foi testado o código para uma imagem com três canais, em 3, é possível visualizar o efeito na imagem.

Assim pode-se considerar a transformação caseira da perspectiva eficaz em comparação a da biblioteca.

4.1 Interpolação

Para a interpolação foram testadas os 4 métodos em diferentes cenários. Sendo eles:

1. Escala 1.5
2. Escala 1.5 com zoom no nariz do baboon
3. Escala fixa de resolução (150,120)
4. Escala 0.1
5. Escala 0.155 para três canais
6. Escala 0.455 para imagem de três canais com zoom

Analisando primeira imagem referente a escala do baboon, figura 6, é possível perceber uma nitida diferença na cor das imagens. Enquanto a do vizinho, bilinear e lagrange preservam os tons originais. A bicubica acaba deixando a imagem levemente mais escura e levemente mais suave, como se fosse suavizada.

Podemos com ainda mais facilidade perceber esta suavização, na imagem 7, com o zoom no nariz do nosso mandril favorito. Em suas narinas existem um serrilhado mais profundo nas outras interpolações. Mas a suavização também tem um preço grande nos detalhes do nariz dele, é facilmente identificável a perda de detalhes.

Na imagem de escala fixa por resolução, 8, é possível notar uma grande diferença na "juba"do mandril (região superior esquerda), entre os demais metódos e a bicúbica. Na última, a região está muito mais próxima ao baboon original, vide 12. Além disso, os detalhes do bigode também são notavelmente mais precisos. Generalizando, para reduções de escala, a bicúbica preserva mais o detalhes que as demais técnicas.

Em cenários exageradamente reduzidos, como na figura 9, onde houve uma redução para 10% da imagem original. A imagem está completamente degenerada para todos os métodos, ainda assim a performance do bicubico está "melhor"pois é possível perceber ainda alguns detalhes do símio mas estranhamente está com uma coluna quase preta. Provavelmente, os valores daquela região estavam fora do intervalo $[0, 255]$ e acabaram por serem clipadas.

Por fim, as figuras 10 e 11, retratam a portabilidade e flexibilidade do código para uma imagem colorida de três canais. Diferentemente da projeção perspectiva, foi necessário uma adaptação do código para trabalhar com os múltiplos canais, tratando os canais isoladamente e unindo em uma só imagem também foi preciso normalizar a saída dos canais para executar o plot, uma vez que a sáida das imagens estavam apresentando valores muito altos.

Visualmente os resultados não são diferentes do já analisados. Provando que o código feito consegue trabalhar com imagens coloridas.

Por fim, chega-se a conclusão que existem poucas diferenças entre as operações de vizinho mais próximo, bilinear e lagrange, em termos de impacto visual, tanto para diminuir e aumentar imagens. Já a operação bicúbica, acabou por suavizar mais a imagem do que as demais, perdendo alguns detalhes ao se analisar com zoom porém preservando os aspectos gerais da imagem.

5 Rotação

Na rotação, figuras 4 e 5, é possível perceber algumas diferenças entre a imagem colorida e grayscale. A primeira sofreu um pad, e região limítrofe da imagem foi substituída por preto, eliminando o "efeito quina"que podemos ver na imagem do Tom Platz, próximo ao seus pés.

Infelizmente, o código não foi capaz de tratar tais casos para imagens coloridas. Assim como não foi feita a expansão da imagem, assim ao rodar em ângulos não múltiplos a 90 alguns pedaços da imagem não são visualizados.

6 Imagens

Figura 1: Transformação de perspectiva projetiva baboon

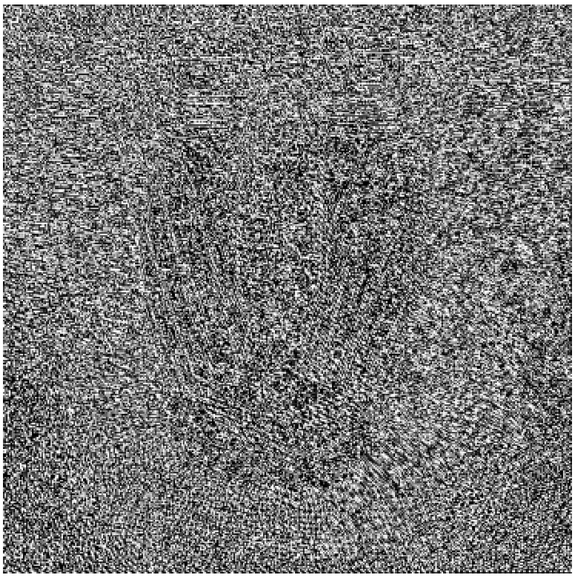

Figura 2: Diferença das imagens de perspectiva do baboon

Figura 3: Transformação de perspectiva projetiva fisiculturista

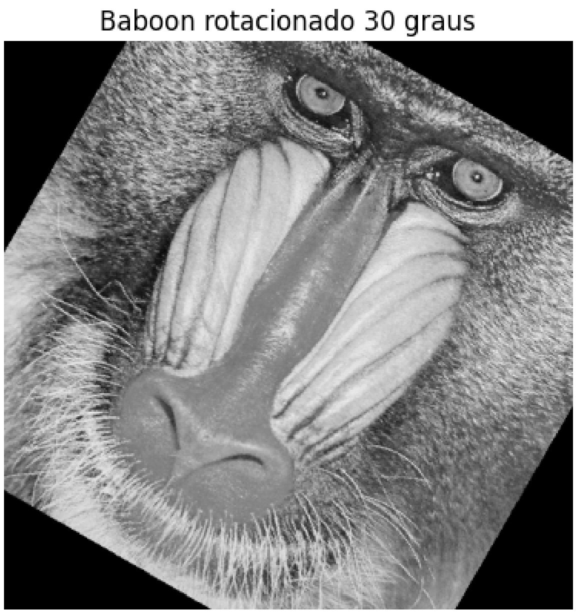

Figura 4: Rotação Baboon

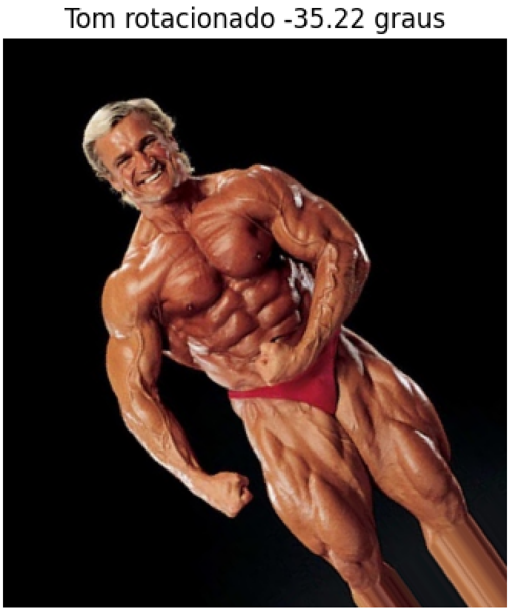

Figura 5: Rotação fisiculturista

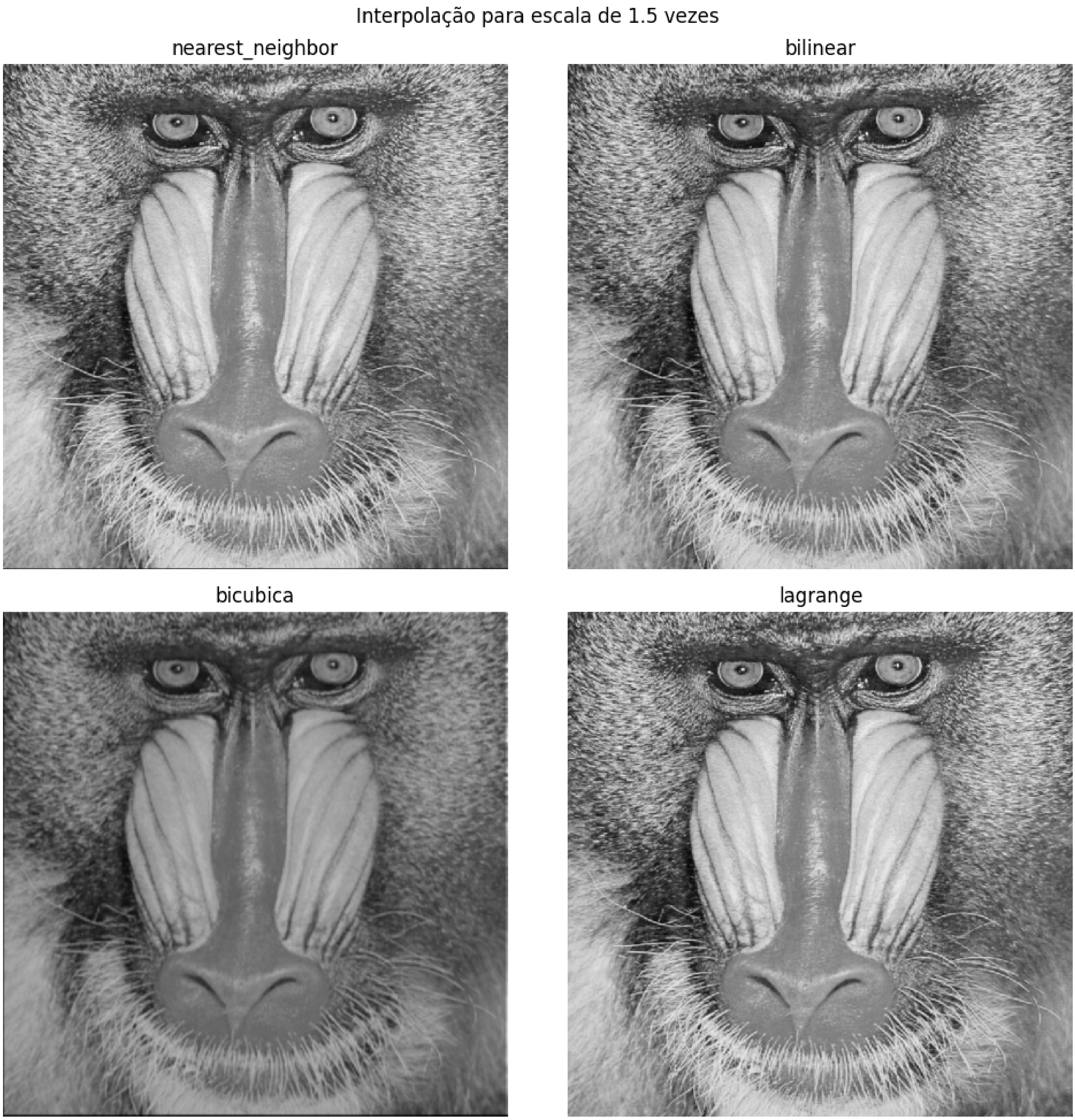

Figura 6: Escala do baboon

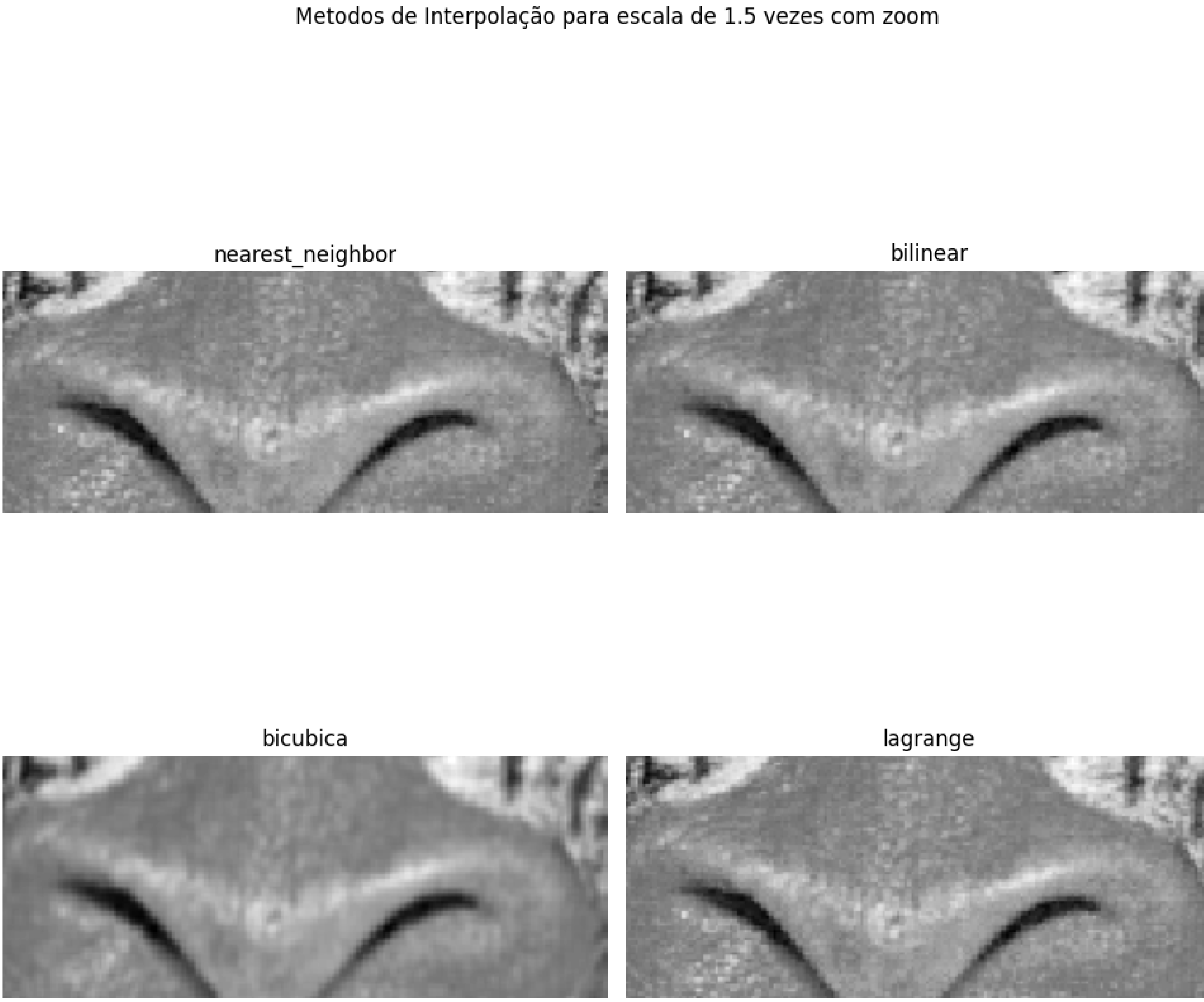

Figura 7: Escala do baboon com foco no nariz

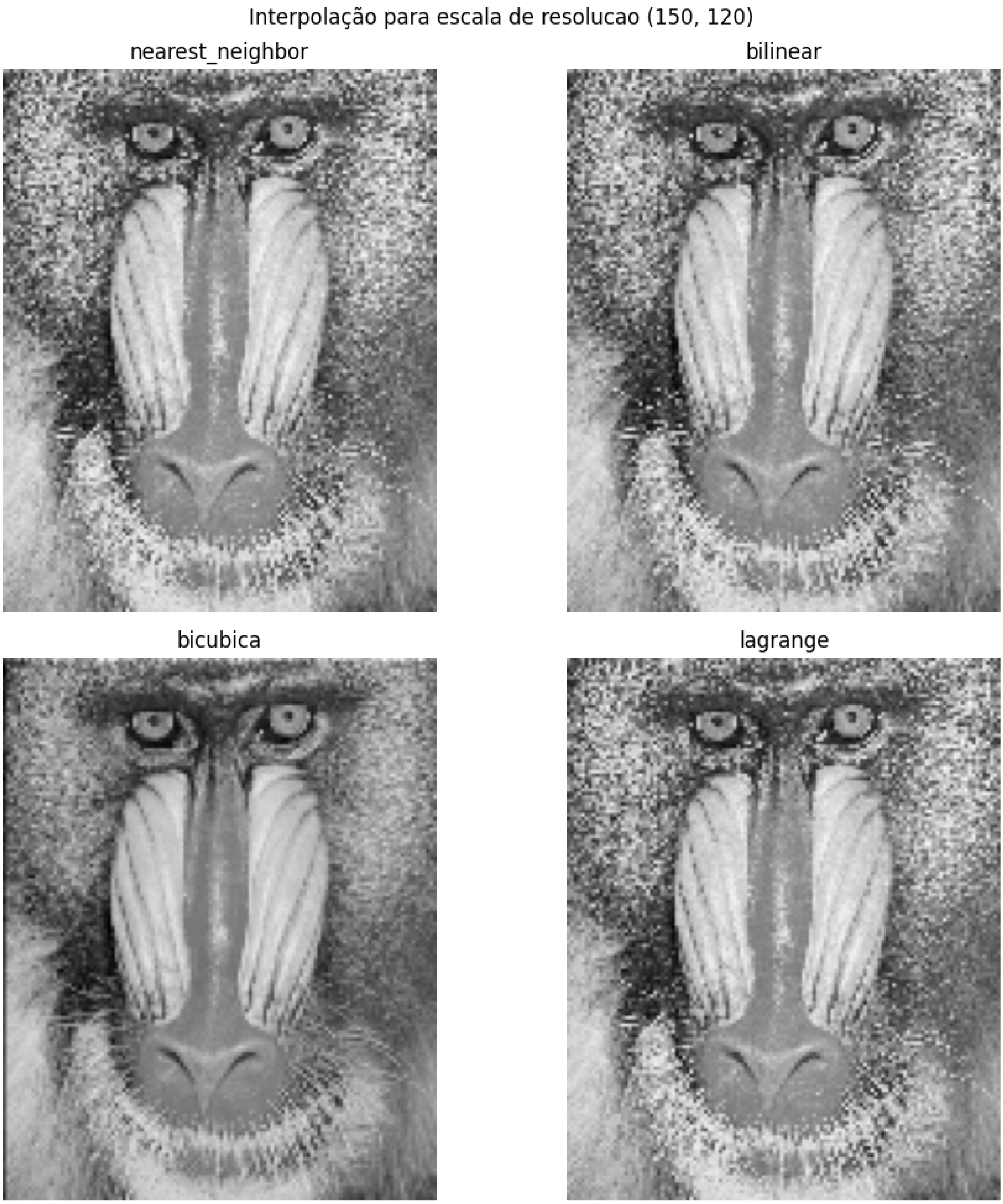

Figura 8: Escala utilizando dimensão de saída

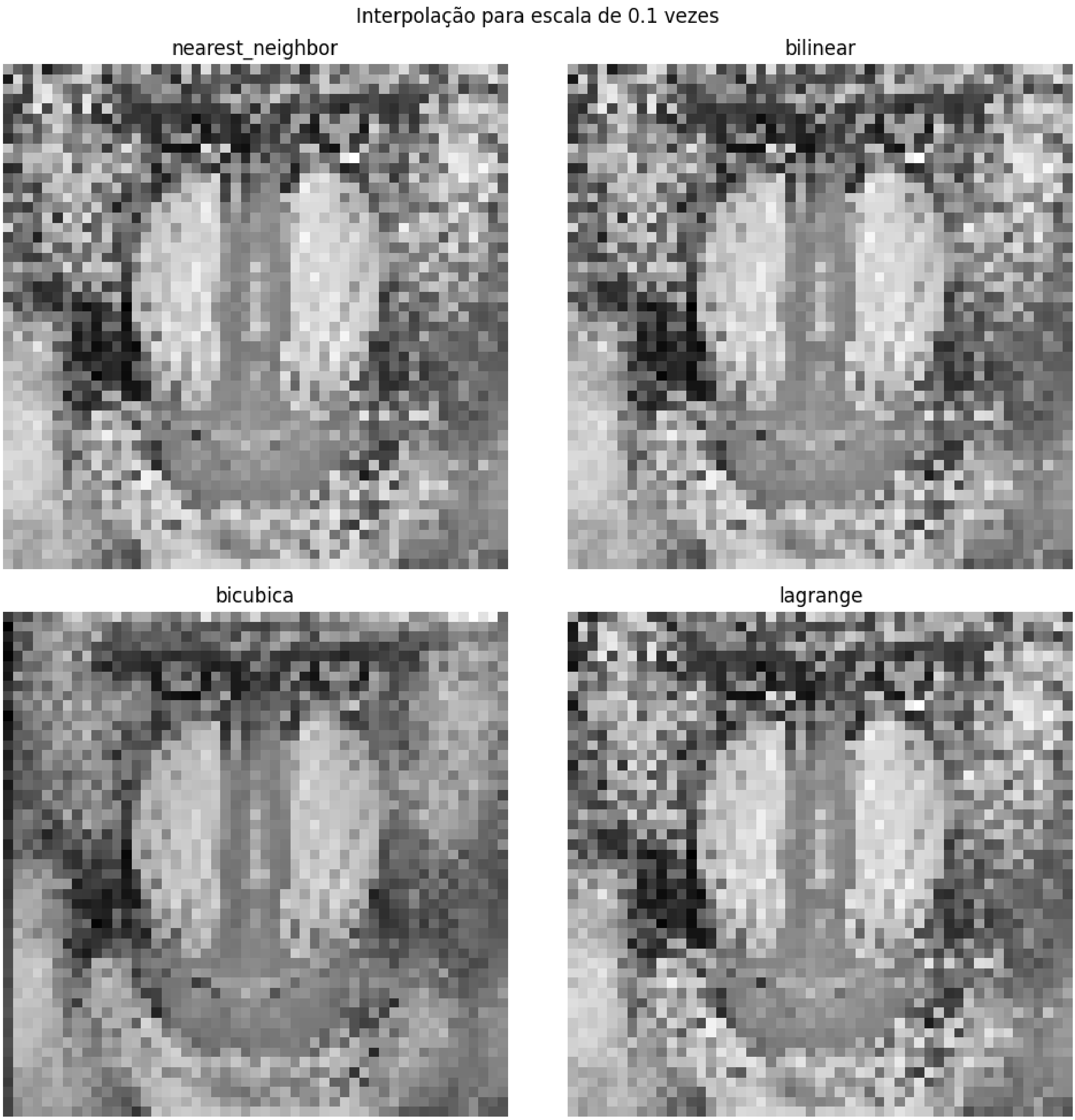

Figura 9: Escala com alta redução

Imagem com três canais interpolada para 0.155

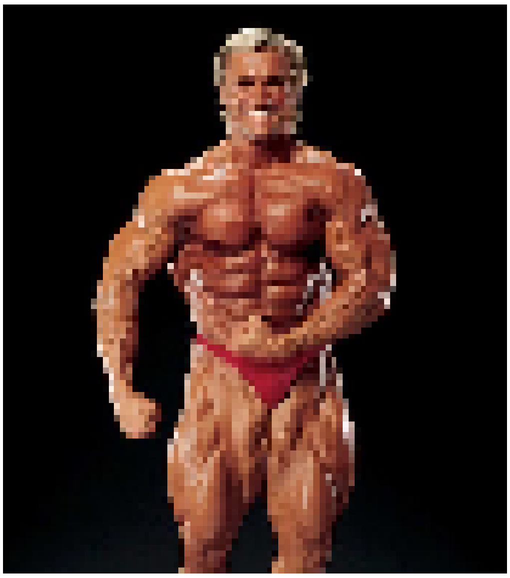

Figura 10: Escala com fisiculturista - Bilinear - 3 canais

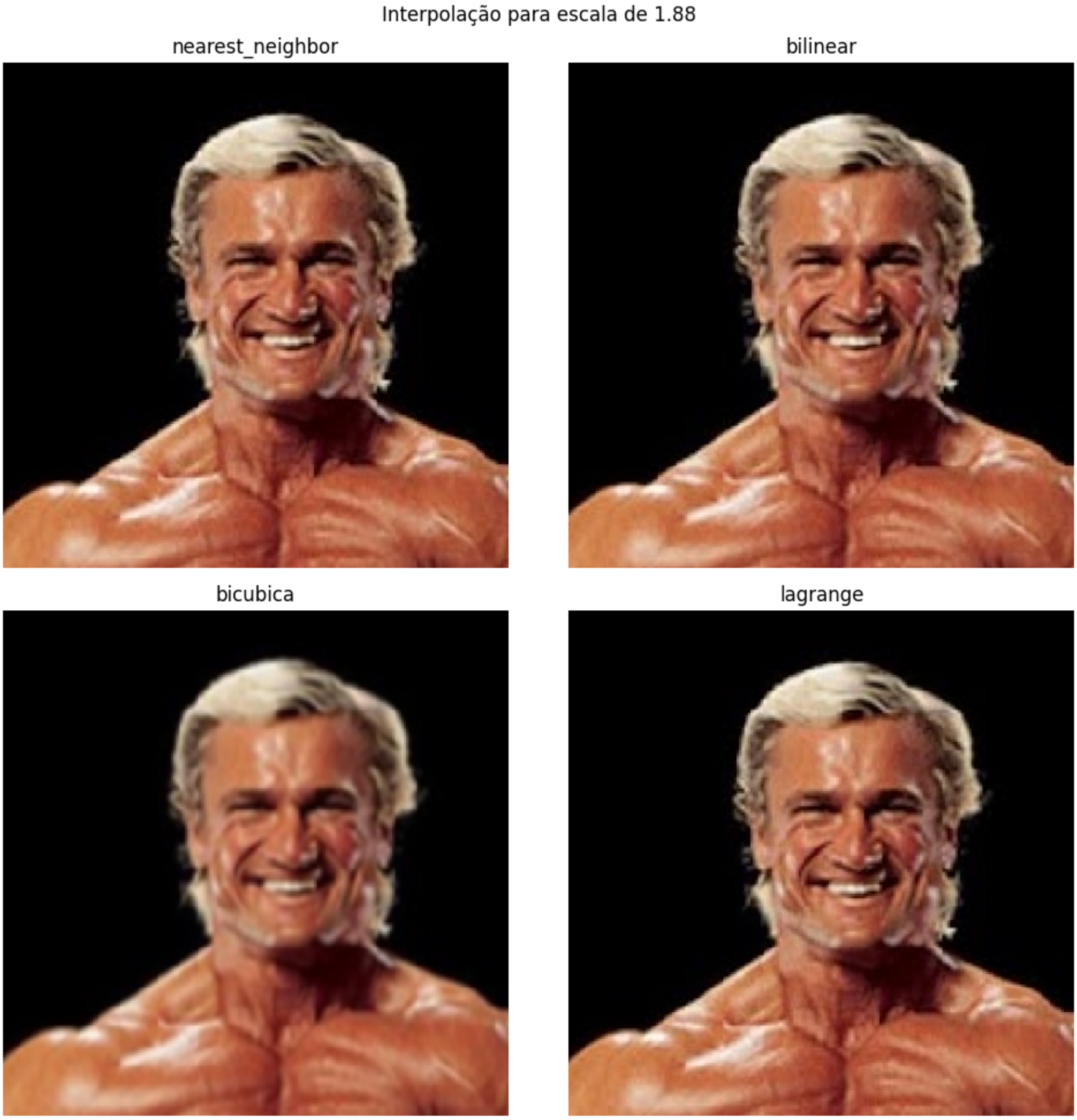

Figura 11: Escala com fisiculturista - Zoom - 3 canais

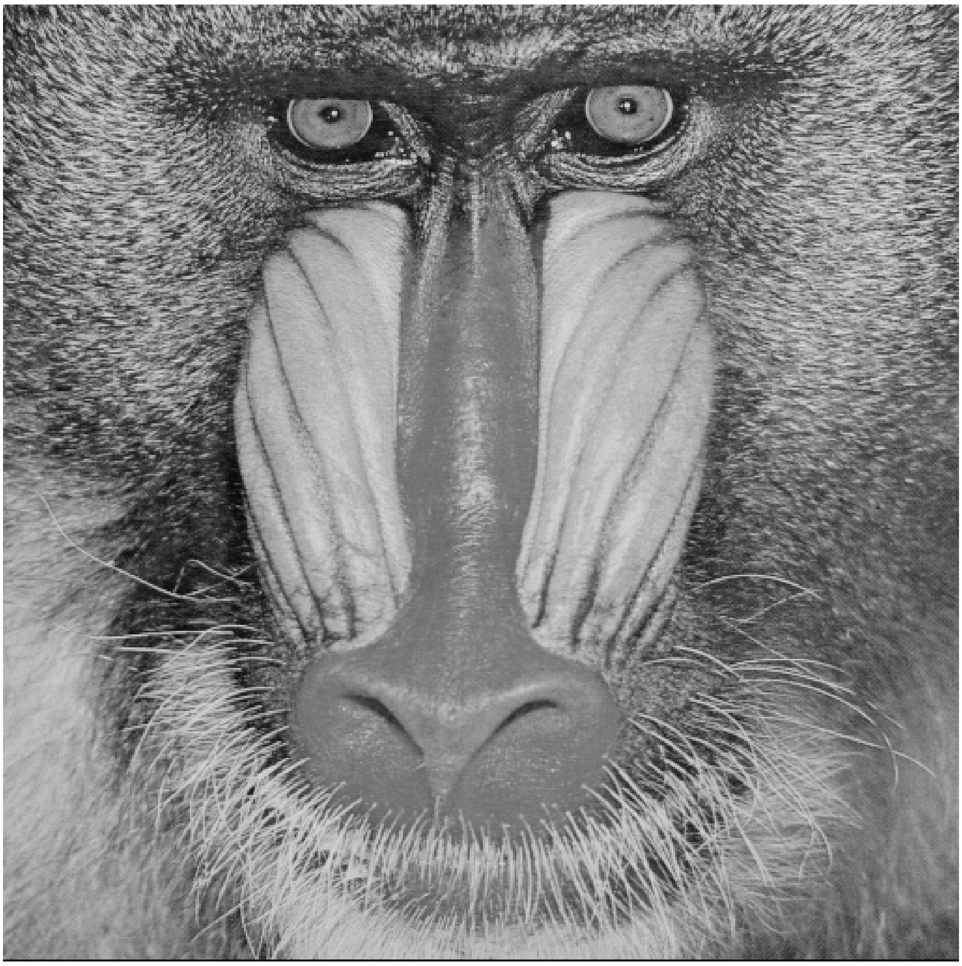

Figura 12: Baboon - O Mandril