Universidade Estadual de Campinas

Instituto de computação

Introdução ao processamento de imagem digital

MC920

Trabalho #5

Nome: Leonardo de Alencar Lopes **RA:** 171928

1. Introdução

O objetivo deste trabalho é realizar transformações geométricas de escala e rotação em imagens, de modo que o valor do ângulo e o fator de escala sejam números reais. Utilizando diferentes métodos de interpolação para determinar os valores dos pixels na imagem de saída será possível identificar suas diferenças. Para isso, foi utilizada a linguagem Python 3 e a biblioteca OpenCV (tratamento das imagens).

Junto a este documento está o arquivo comprimido **171928-Trabalho-5.zip**. Nele, há todos os scripts que foram implementados e serão citados ao longo deste texto, além das pastas **img**, que contém as imagens utilizadas como entrada pelos algoritmos, e **res**, que contêm as imagens de output.

2. O programa

Para realização das transformações nas imagens, foi implementado o programa *transgeo.py*, que utiliza a biblioteca implementada *transgeo_lib.py*. O programa recebe como parâmetros obrigatórios uma imagem que será realizada a transformação desejada (deve estar dentro da pasta **img**) e um tipo de transformação. Para realizar uma rotação no sentido anti-horário, deve-se passar o parâmetro no ângulo em graus que se deseja rotacionar. Transformações de escala podem ser feitas tanto passando como parâmetro um fator de escala quanto as dimensões (em pixels) desejadas para a imagem de saída. Além disso, o programa pode receber parâmetros que indicam o método de interpolação desejado e o nome da imagem de saída, apesar de haver valores *default*. Caso não seja passado o método de interpolação, será utilizado o Vizinho Mais Próximo.

Tabela 1: Parâmetros de execução do programa

Símbolo	Parâmetro	Valores possíveis
-a	Ângulo de rotação	Qualquer número real
-e	Fator de escala	Qualquer número real
-d	Dimensão da imagem de saída	Dois valores naturais para largura e altura
-m	Método de interpolação	String de caracteres: Vizinho mais próximo = "vmp" Bilinear = "bl" Bicúbica = "bc" Polinômios de Lagrange = "pl"
-i	Imagem de entrada	String de caracteres
-o	Nome da imagem de saída	String de caracteres

Exemplos de execução:

transgeo.py -i baboon.png -a 45.5 -m bl -o baboon_rotacionado.png

transgeo.py -d 1000 1000 -m bc -i city.png -o city_zoom.png

transgeo.py -e 1.5 -i baboon.png

A transformação é realizada através do mapeamento indireto dos pixels da imagem de saída na imagem de entrada. Dessa forma é garantido que todos os pontos da imagem transformada terão um pixel associado na imagem original, apesar de diferentes pixels poderem ser mapeados no mesmo ponto. Isso é feito através da operação

$$P = T^{-1}P'$$

em que P é um ponto na imagem original, T^{-1} é a inversa da matriz de transformação e P' é um ponto na imagem transformada.

As inversas das matrizes de transformação utilizadas em rotação e escala para este programa foram:

$$T_{rot}^{-1} = \begin{bmatrix} cos\theta & -sen\theta & x_1(1-cos\theta) + y_1 sen\theta \\ sen\theta & cos\theta & y_1(1-cos\theta) - x_1 cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{esc}^{-1} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Em quem x_1 e y_1 são as coordenadas do centro da imagem de saída, θ é o ângulo de rotação desejado e Sx e Sy são fatores de escala nos eixos x e y, respectivamente.

Após aplicada a transformação, é necessário aplicar uma técnica de interpolação para determinar o valor que o pixel no ponto P' deverá assumir. Na interpolação pelo vizinho mais próximo, essa intensidade será determinada pelo valor do vizinho mais próximo do ponto P encontrado na imagem original.

A transformação Bilinear utiliza uma média ponderada de distância dos quatro pixels vizinhos mais próximos para determinar a intensidade no ponto P'. Assim, sendo o ponto $P = (x', y')$ e $f(x, y)$ a intensidade do pixel no ponto (x, y) da imagem original, a equação deste método é dada por:

$$f(x', y') = (1 - dx)(1 - dy)f(x, y) + dx(1 - dy)f(x + 1, y) +$$
$$(1 - dx)dyf(x, y + 1) + dxdyf(x + 1, y + 1)$$

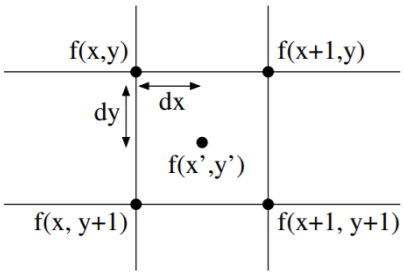

Figura 1: Transformação Bilinear

A interpolação Bicúbica utiliza uma vizinhança de 4x4 pontos ao redor do ponto em questão para calcular seu valor de intensidade. A equação deste método é dada por:

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

Sendo

$$R(s) = \frac{1}{6}[P(s + 2)^3 - 4P(s + 1)^3 + 6P(s)^3 - 4P(s - 1)^3]$$
$$P(t) = \begin{cases} t, & t > 0 \\ 0, & t \leq 0 \end{cases}$$

Assim como a Bicúbica, a interpolação por Polinômios de Lagrange utiliza uma vizinhança de 4x4 para calcular a intensidade de um pixel (x', y'). Sua equação é dada por:

$$f(x', y') = \frac{-dy(dy - 1)(dy - 2)L(1)}{6} + \frac{(dy + 1)(dy - 1)(dy - 2)L(2)}{2} +$$
$$\frac{-dy(dy + 1)(dy - 2)L(3)}{2} + \frac{dy(dy + 1)(dy - 1)L(4)}{6}$$

Sendo

$$L(n) = \frac{-dx(dx - 1)(dx - 2)f(x - 1, y + n - 2)}{6} + \frac{(dx + 1)(dx - 1)(dx - 2)f(x, y + n - 2)}{2} +$$
$$\frac{-dx(dx + 1)(dx - 2)f(x + 1, y + n - 2)}{2} + \frac{dx(dx + 1)(dx - 1)f(x + 2, y + n - 2)}{6}$$

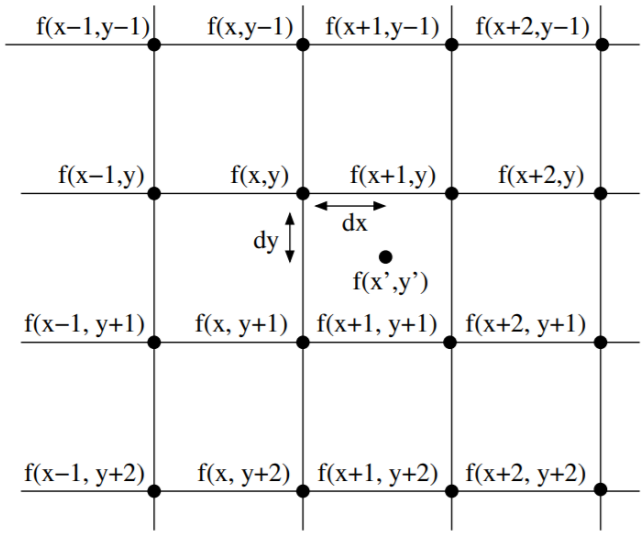

Figura 2: Transformação Bicúbica e Polinômios de Lagrange

3. Resultados

Na seção de apêndice ao final deste documento estão as imagens obtidas como saída do programa utilizando os diferentes métodos de interpolação. Foram realizadas transformações de rotação em 45º e de escala com fator de 1,5.

Para exemplificar os resultados das rotações foi utilizada a imagem *city.png* enquanto para a escala foi utilizada a imagem *baboon.png*. Observando as janelas das imagens de saída, é possível notar certas distorções geradas pela transformação. Pelo método do Vizinho Mais Próximo as janelas tiveram uma quantidade muito alta de serrilhados, borrando alguns trechos da imagem. Isso pode ser notado também nos olhos do mandril, que perderam um pouco sua aparência circular. Na interpolação Bilinear esse problema não aparece, apesar de ainda haver distorções em comparação com a imagem original. Os métodos por Polinômios de Lagrange e Bicúbica tiveram resultados muito próximos à imagem de entrada, não sendo possível notar grandes distorções.

Durante os testes do programa notou-se uma ordem de velocidade de execução para cada um dos métodos de interpolação. Do mais rápido para o mais lento a ordem de velocidade foi: Vizinho Mais Próximo, Bilinear, Polinômios de Lagrange e Bicúbica. Dependendo dos parâmetros utilizados para a execução, interpolação Bicúbica poderia levar alguns minutos para rodar, enquanto todos os outros não passaram de um minuto.

4. Conclusão

Transformações geométricas utilizando os métodos de interpolação exemplificados neste documento podem ser bastante úteis em programas que precisam modificar imagens. Porém, não é possível definir um método de interpolação absoluto, sendo que cada um pode ter aplicações diferentes. Vizinho mais próximo é o que gerou mais distorções, apesar disso, seu tempo de execução é centésimos de segundo, podendo ser útil para imagens prévias. Mesmo assim, o método por Polinômios de Lagrange se mostrou obtendo um bom resultado na imagem de saída, não demorando excessivamente para executar.

Apêndices

Figura 3: city.png original

Figura 4: city.png rotacionada 45° utilizando interpolação pelo Vizinho Mais Próximo

Figura 5: city.png rotacionada 45° utilizando interpolação Bilinear

Figura 6: city.png rotacionada 45º utilizando interpolação Bicúbica

Figura 7: city.png rotacionada 45° utilizando interpolação por Polinômios de Lagrange

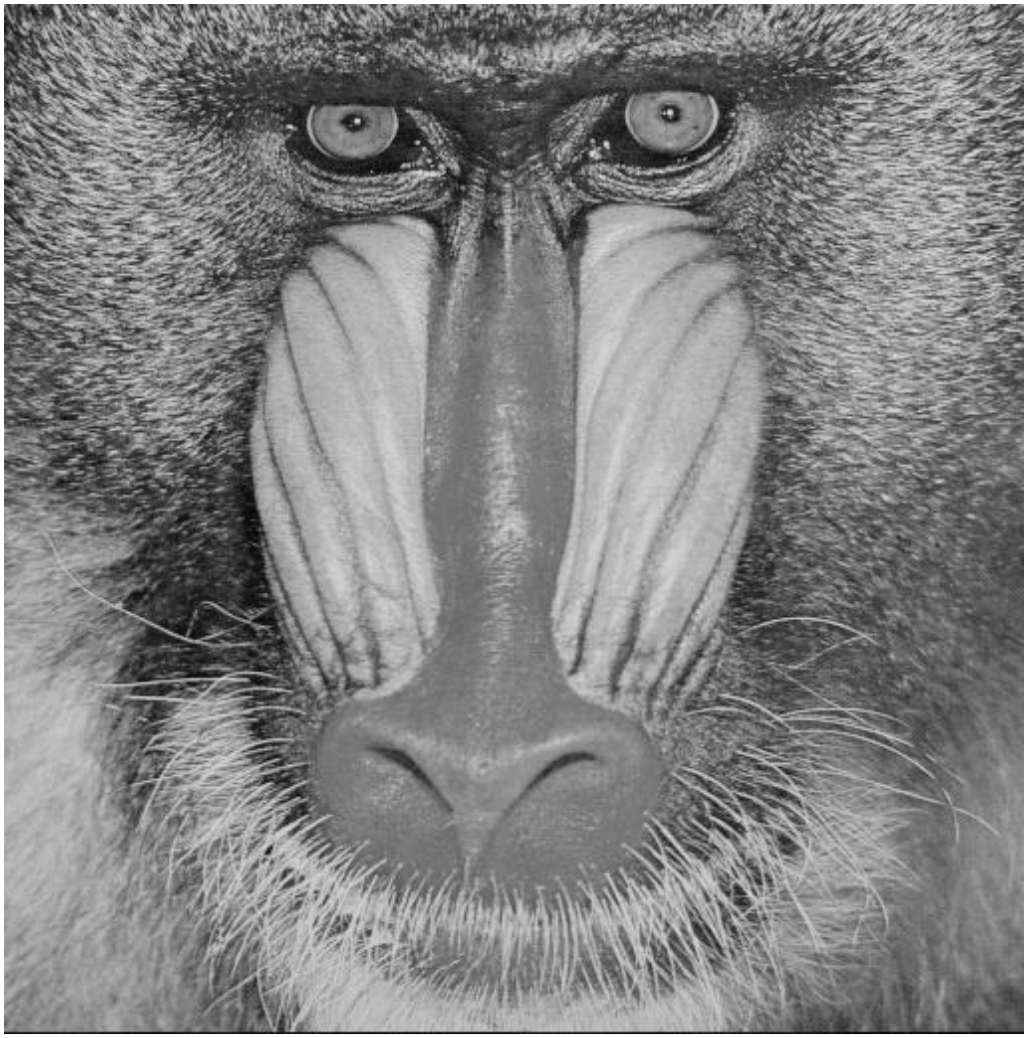

Figura 8: baboon.png original

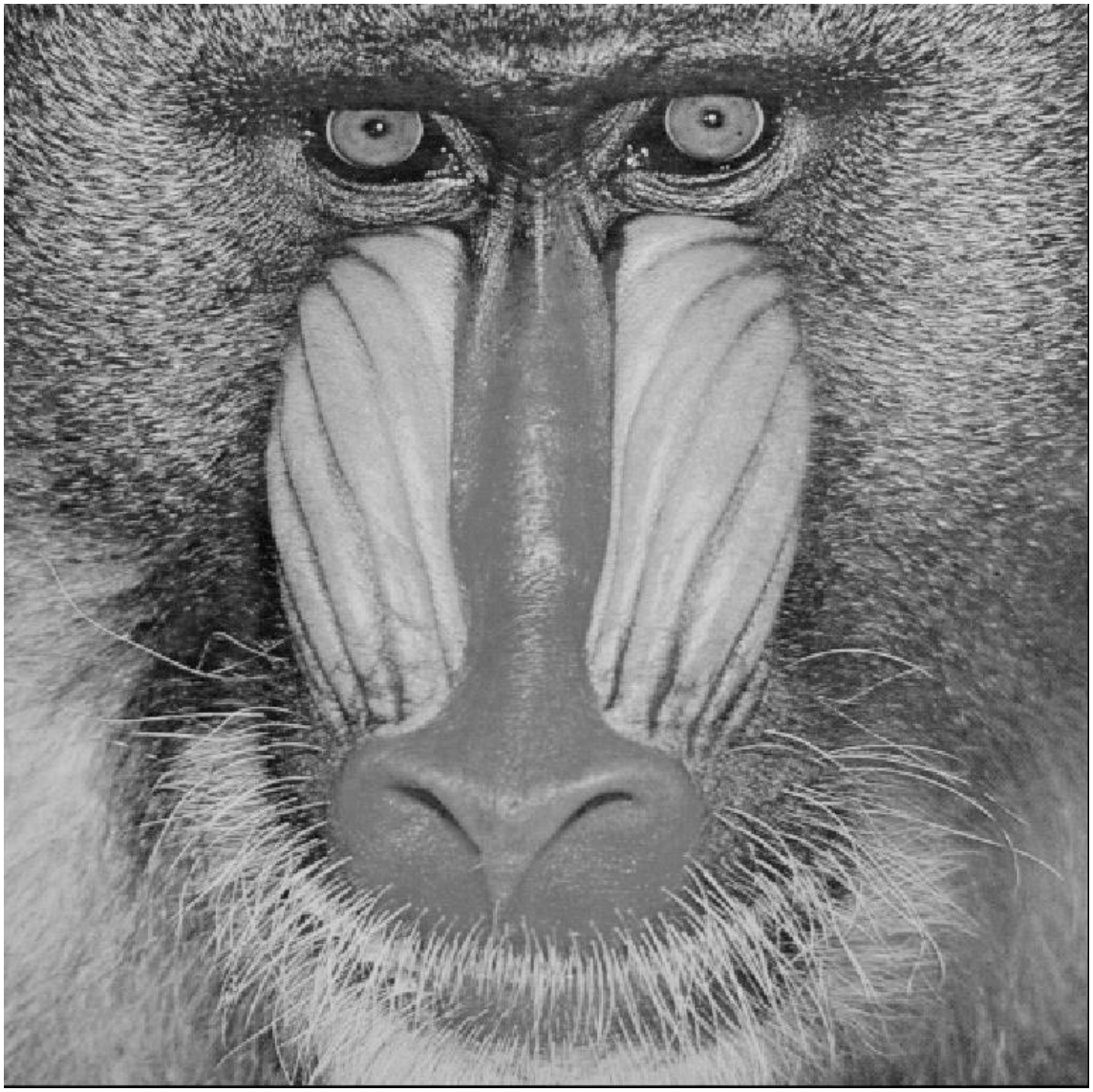

Figura 9: baboon.png aumentada por fator de 1,5 pela interpolação Vizinho Mais Próximo

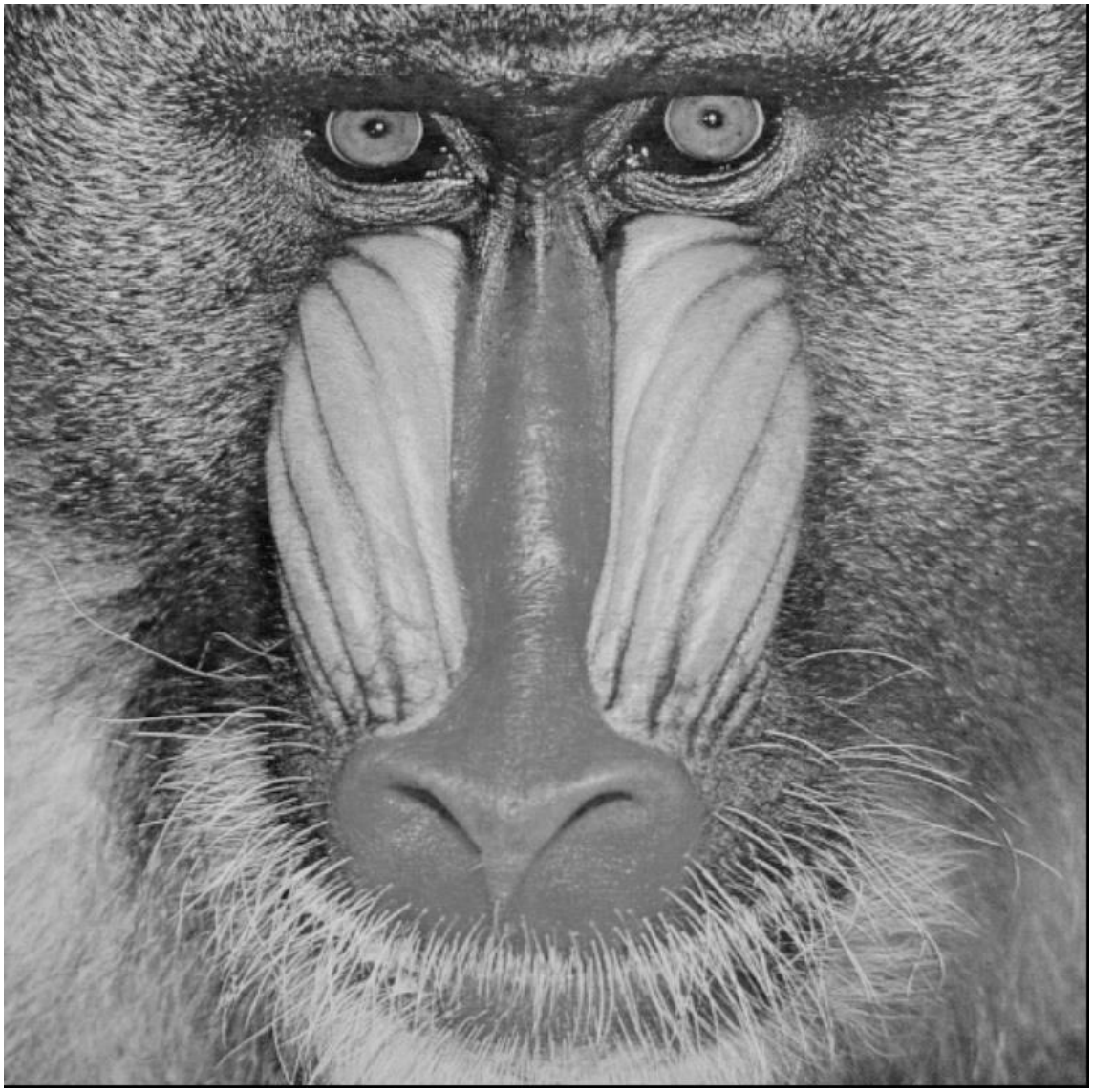

Figura 10: baboon.png aumentada por fator de 1,5 pela interpolação Bilinear

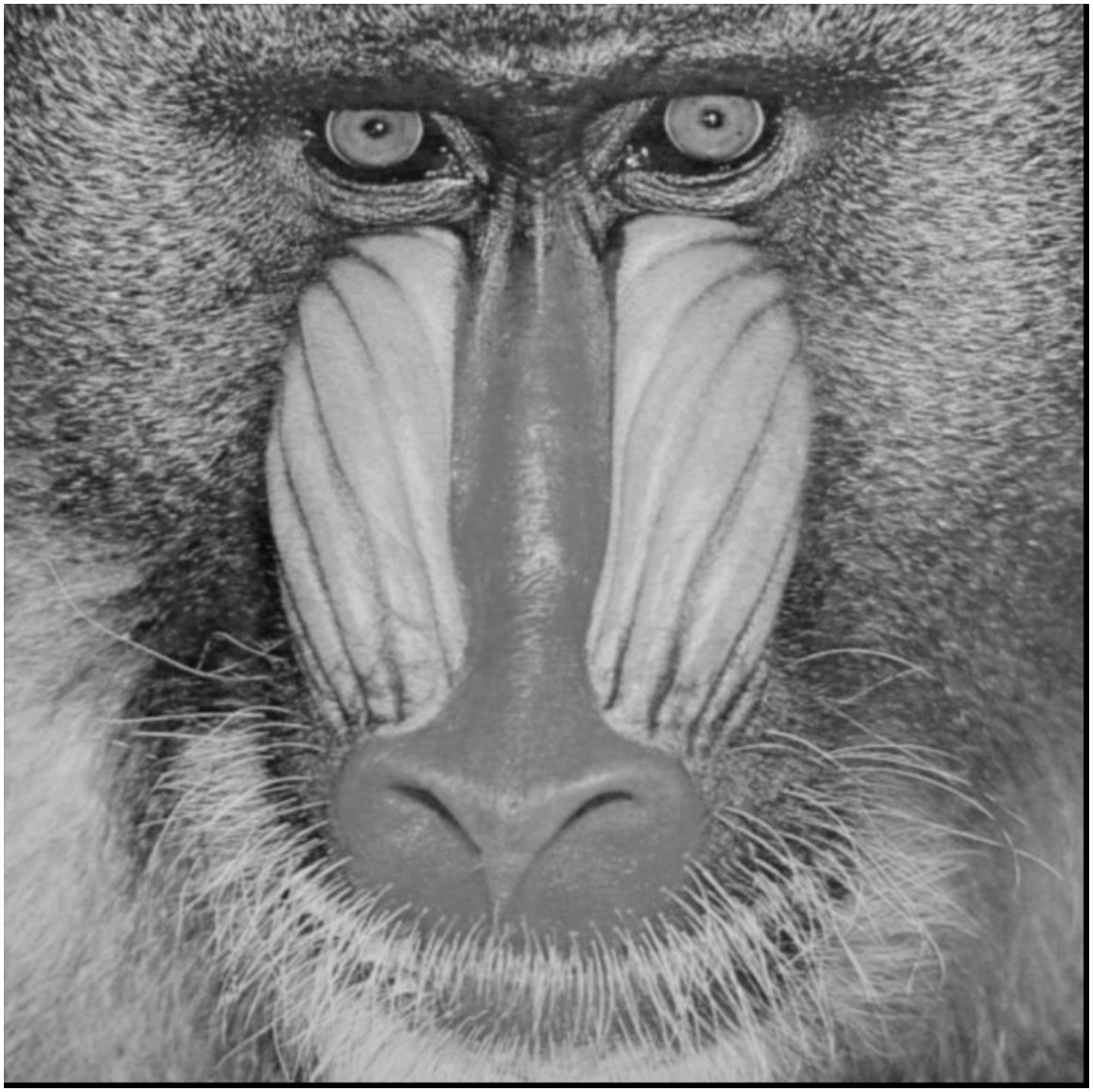

Figura 11: baboon.png aumentada por fator de 1,5 pela interpolação Bicúbica

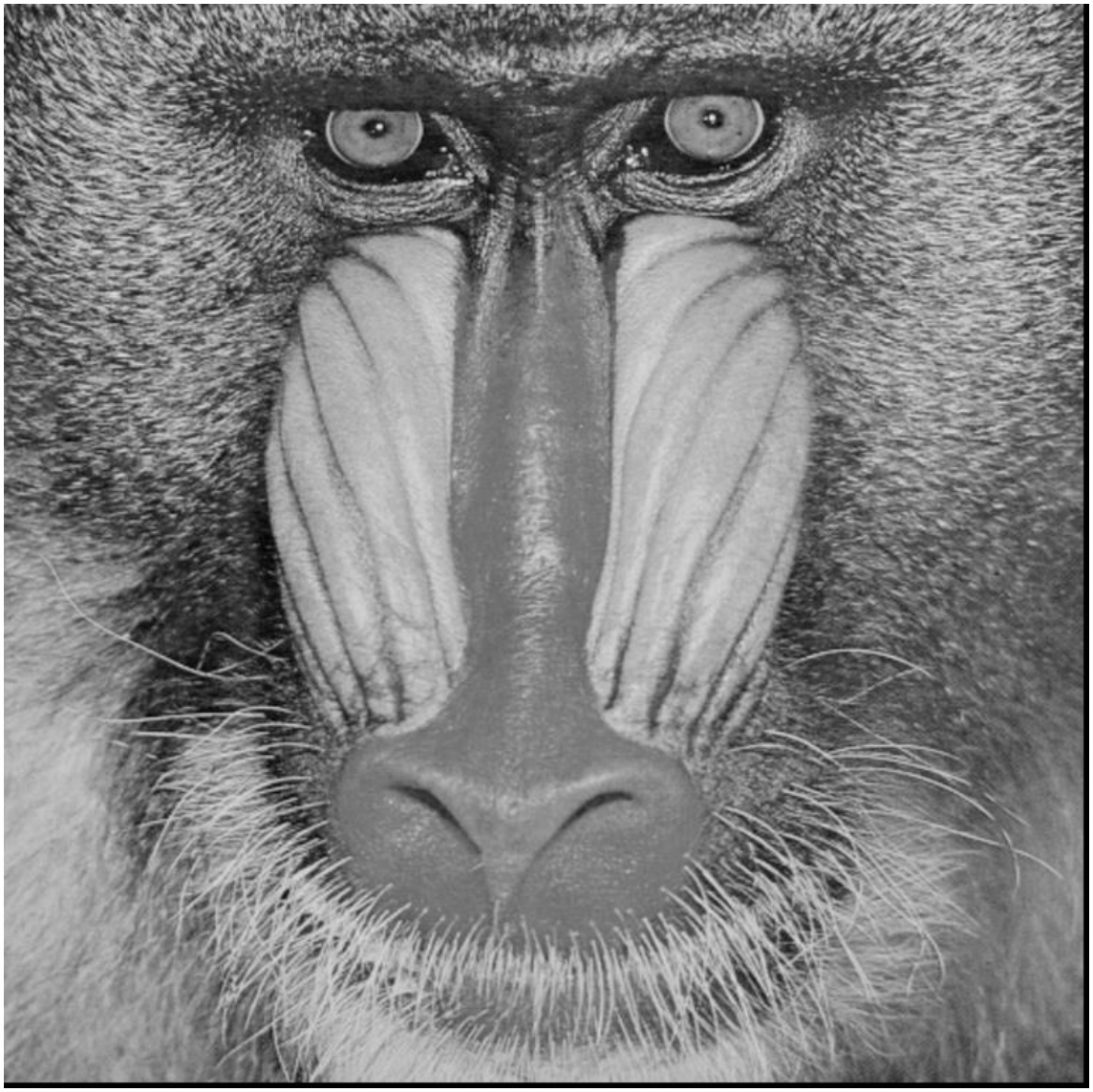

Figura 12: baboon.png aumentada por fator de 1,5 pela interpolação por Polinômios de Lagrange