# Introdução ao Processamento Digital de Imagem (MC920) - Trabalho 4

Heitor Boschirolli - 169477

2018
June

## 1 Executando o código

O tar Heitor_Boschirolli_169477.tar contém o código transform.py - responsável pelo processamento das imagens -, cinco imagens para teste e este PDF.

Para executar o arquivo transform.py, é preciso digitar "python transform.py" em um terminal linux com python 3.6, seguido dos argumentos opcionais e seus respectivos valores. Existem seis argumentos opcionais e cada um deles é explicado abaixo:

- imagem de entrada (flag -i ou --input): nome da imagem de entrada.
- imagem de saída (flag -o ou --output): nome com o qual a imagem transformada deve ser salva, se não definido, a imagem será salva como out.png.
- ângulo (flag -a ou --angle): ângulo em graus com o qual a imagem de entrada será rotacionada no sentido anti-horário.
- fator de escala (flag -e ou --scale): fator com o qual a imagem de saída será redimensionada.
- dimensão da imagem de saída (flag -d ou --dimensions): dimensões da imagem de saída
- método de interpolação (flag -m ou --interpolation): corresponde ao método de interpolação utilizado, os métodos disponíveis são: Nearest neighbor, Bilinear, Bicubic e Lagrange interpolating polynomial que podem ser invocados por esse argumento opcional como "Nearest", "Bilinear", "Bicubic" e "Lagrange" respectivamente (sem as aspas).

Como os argumentos que definem a escala e a dimensão da imagem de saída podem ser redundantes entre si (isto é, um pode ser obtido a partir do outro), apenas um deve ser usado (na tentativa de usar os dois, o programa gerará uma mensagem de erro e encerrará sua execução). Além disso, uma imagem não pode ser rotacionada e redimensionada em uma mesma execução do programa, portanto se o argumento de ângulo for definido junto com algum argumento que define as dimensões da imagem de saída, o programa também irá gerar uma mensagem de erro e terminar a execução.

Abaixo seguem alguns exemplos de chamadas corretas do programa:

```
python3 transform.py -i inp.png -m Bicubic -a 45 -o
    out.png
python3 transform.py -i inp.png -m Nearest -e 1.3
python3 transform.py -i inp.png -m Lagrange -d 521
    521
```

Nesse exemplo, inp.png é a imagem a ser processada.

## 2 Solução

### 2.1 Transformação de escala

No método utilizado, a transformação de escala precisa do fator de escala para ser feita, portanto, quando esse não era um argumento do programa, era calculado dividindo uma dimensão da imagem de saída pela mesma dimensão da imagem de entrada, obtendo dois fatores de escala, um para a dimesão x e outro para a y.

Para mapear os pixels da imgem redimensionada para um equivalente da imagem original, fez-se (sendo $s_x$ e $s_y$ os fatores de escala, $P_i = (x_i, y_i)$ um ponto da imagem de entrada e $P_o = (x_o, y_o)$ o ponto da imagem de saída a ser mapeado):

$$x_i = x_o / s_x$$

$$y_i = y_o / s_y$$

Como isso pode resultar em um ponto com coordenadas não inteiras, é preciso aproximá-lo utilizando os valores dos seus vizinhos, isso é feito com os métodos de interpolação. Existem diversas formas de realizar a interpolação, neste projeto foram usados os métodos do vizinho mais próximo (nearest neighbor), bilinear, bicúbico e por polinômios de Lagrange.

Como os métodos de interpolação usam os vizinhos de um píxel para determinar a intensidade, existe um problema quando eles são usados nos píxels de borda (pois estes não têm todos os vizinhos). Para os píxels da borda, portanto, decidiu-se não realizar a interpolação, para evitar que seus valores permanecam como zero (gerando uma borda preta no contorno da imagem), copiou-se os valores dos píxels adjacentes aos não interpolados.

Na imagem da figura 2 tem-se o resultado do aumento da imagem da figura 1 em 0.75 vez com os píxels de borda ignorados (a interpolação usada foi a bilinear), na imagem da figura 3 tem-se o resultado após copiar os píxels vizinhos à borda para a borda.

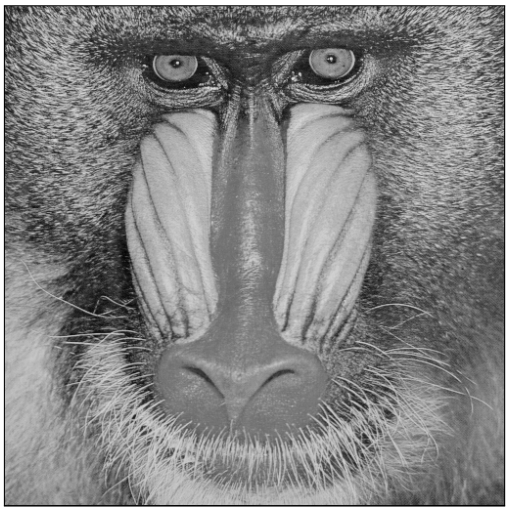

Figure 1: imagem original

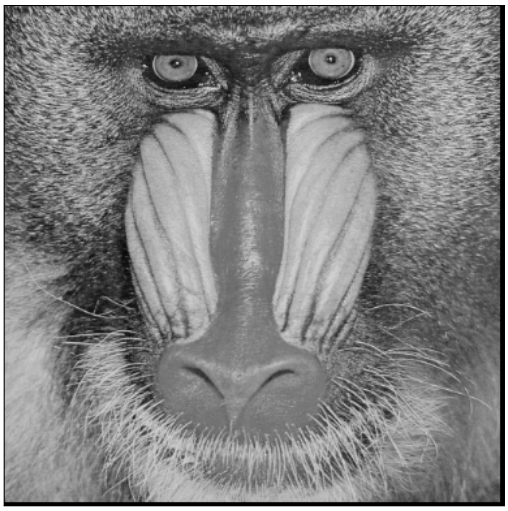

Figure 2: imagem após um aumento em 0.75 vezes com bordas ignoradas (redimensionada para caber no pdf)

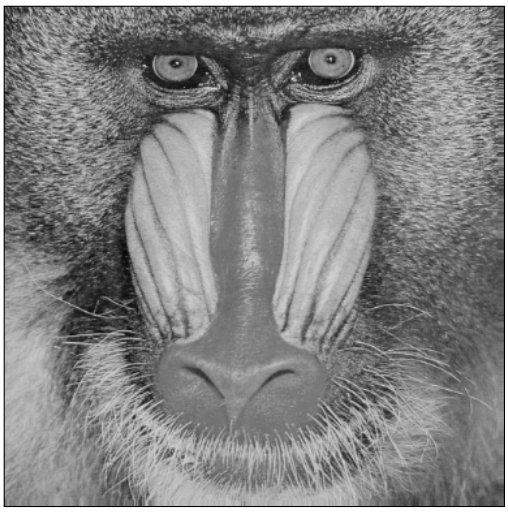

Figure 3: imagem após um aumento em 0.75 vezes com bordas copiadas dos vizinhos (redimensionada para caber no pdf)

## 2.2 Transformação de rotação

Seja $P_o = (x, y)$ o ponto da imagem de saída cujo valor quer-se calcular para definir uma rotação e $P_i = (x', y')$ a posição do píxel da imagem de entrada para o qual $P_o$ deve ser mapeado, dado um ângulo theta em radianos, podemos calcular $P_i$ em função dos valores de $P_o$ com:

$$x' = xcos(theta) - ysin(theta)$$

$$y' = xsin(theta) + ycos(theta)$$

Que representa uma rotação dos pontos no plano xy tendo a origem como eixo de rotação. Como ter a origem da imagem (canto superior esquerdo) como centro de rotação fazia com que algumas rotações gerassem perdas muito grandes do conteúdo da imagem (uma rotação de 90 graus, por exemplo, faria com que toda imagem saísse da região orignal, fazendo com que todo o conteúdo desaparecesse), preferiu-se realizar uma rotação com eixo de rotação no centro da imagem. Para isso, basta alterar a fórmula original para (sendo ($c_x$, $c_y$) o centro da imagem):

$$x' = c_x + (x - c_x)cos(theta) - (y - c_y)sin(theta)$$

$$y' = c_y + (x - c_x)sin(theta) + (y - c_y)cos(theta)$$

Por fim, como o ângulo era dado em graus e a rotação desejada era no sentido anti-horário (sendo que a fórmula apresentada rotaciona no sentido horário para um ângulo em radianos), foi realizada a seguinte transformação em theta antes de usá-lo na fórmula apresentada:

$$theta := -theta' \times \pi/180$$

onde theta' é o valor do ângulo em graus indicando uma rotação no sentido anti-horário.

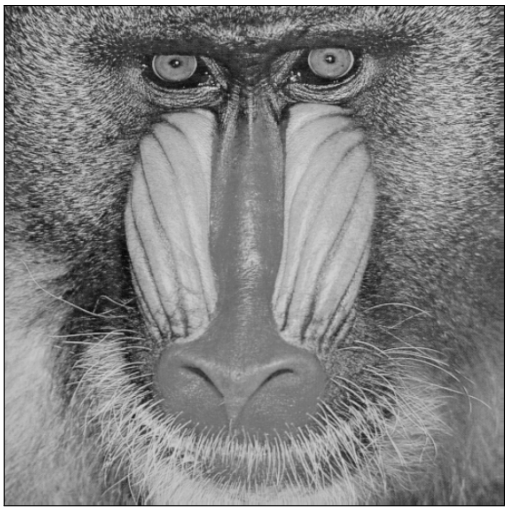

Figure 4: Resultado da ampliação em 1.1 vez da imagem da figura 1 usando interpolação bilinear

Para o caso em que a imagem de entrada é input.png (figura 1), o método de interpolação usado é a do vizinho mais próximo e as dimensões da imagem de saída são 512x256, abaixo encontra-se a chamada do programa e na figura 5 a imagem resultante.

# 3 Exemplo de entrada e saída

Para o caso em que a imagem de entrada é input.png (figura 1), o método de interpolação usado é o bilinear e a imagem de saída deve ser aumentada em 1.1 vez, abaixo encontra-se a chamada do programa e na figura 4 a imagem resultante.

```
python transform.py -i input.png -m
    Bilinear -e 1.1
```

```
python transform.py -i input.png -m
    Nearest -d 512 256
```

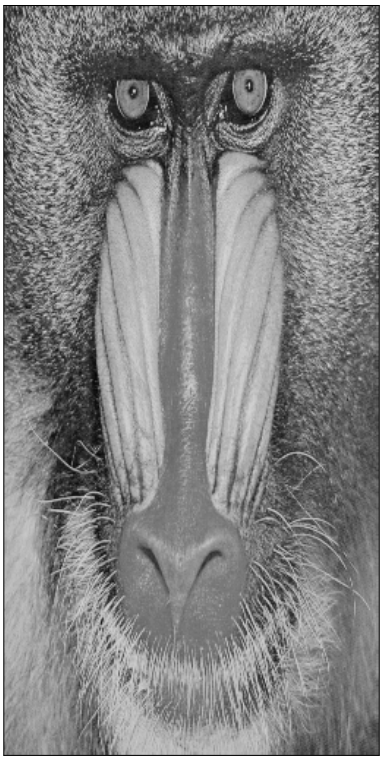

Figure 5: Resultado da alteração das dimensões da imagem da figura 1 com dimenões 512x512 para dimensões 512x256

Para o caso em que a imagem de entrada é input.png (figura 1), o método de interpolação usado é o de polinômios de Lagrange e o ângulo de rotação é de 10 graus, abaixo encontra-se a chamada do programa e na figura 6 a imagem resultante.

```
python transform.py -i input.png -m
    Lagrange -a 10
```

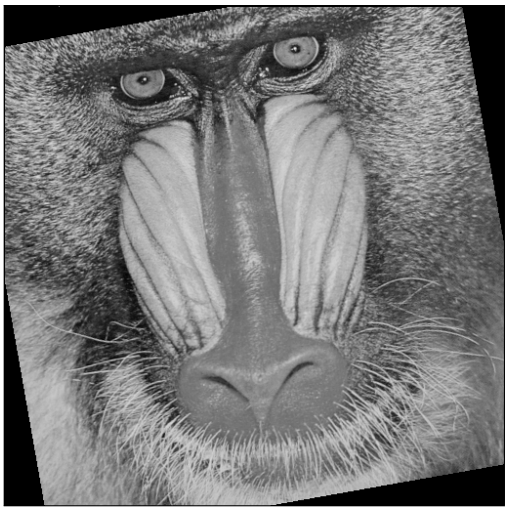

Figure 6: Resultado da rotação da imagem da figura 1 em 10 graus no sentido anti-horário

# 4 Limitações

O programa é limitado a trabalhar com imagens png em escalas de cinza; como o objetivo do trabalho era o estudo das técnicas de interpolação em operações de transformações de imagens, essas limitações não são sérias quanto ao que foi estudado no trabalho. Se as funções implementadas fossem usadas em aplicações práticas, essas limitações tornam-se bastante sérias (especialmente o fato de aceitar apenas imagens em escalas de cinza).

Para o caso da transformação de escala, o tratamento escolhido para as bordas (de copiar os píxels adjacentes) é uma solução possível, porém o fato de outras opções não serem oferecidas (como preencher as bordas com um píxels de valor determinado) como estão em algumas bibliotecas como open-cv e scikit-image é um fator limitante.