Introdução ao processamento de imagem digital

MC920

Trabalho #0

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

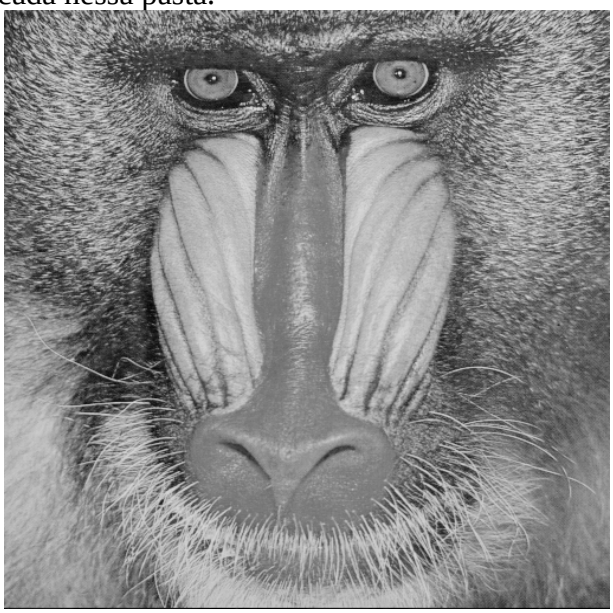
1. Introdução

O objetivo deste trabalho é implementar funções básicas de processamento de imagens, com a finalidade de familiarização com as bibliotecas que serão utilizadas ao longo do curso. Para isso, foi utilizada a linguagem Python 3 e a biblioteca de tratamento de imagens OpenCV.

Junto a este documento está o arquivo comprimido **171928-Trabalho-0.zip**. Nele há todos os scripts que serão citados ao longo deste texto, além das pastas **img**, que contém as imagens utilizadas pelos algorítmos, e **resultados**, que contém os *outputs* de cada script.

2. Os programas

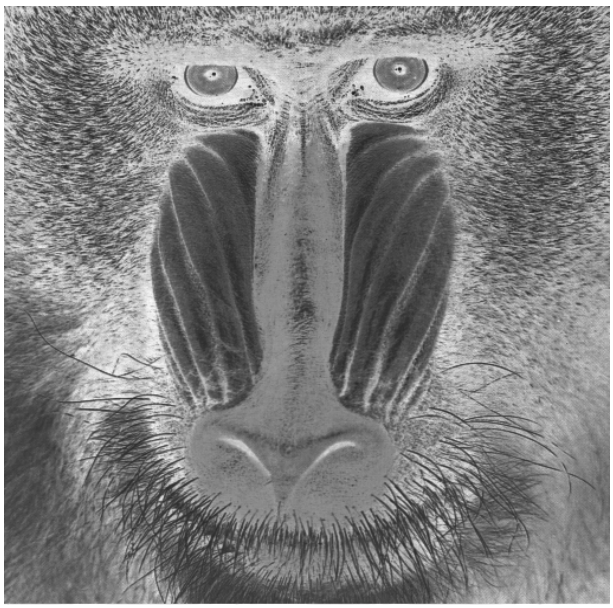
Todos os códigos projetados recebem imagem(s) .png monocromática(s) como parâmetro(s), além da possibilidade de algum outro argumento a depender do código executado. Os programas acessam apenas as imagens presentes na pasta “img”. Assim, para os scripts conseguirem operar uma imagem, ela deve ser colocada nessa pasta.



(a) Original babbon.png

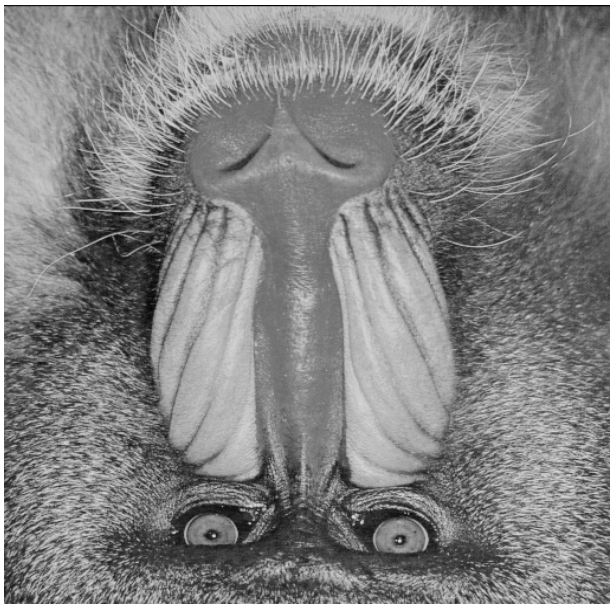
Foram implementados os seguintes scripts:

- **negativo.py:** Recebe como parâmetro uma imagem e mostra seu negativo, além de salvá-lo na pasta “resultados/negativos”. Para isso, é realizada a operação **(255 – img)** de forma vetorizada em uma imagem **img** de entrada. Exemplos de chamada de execução:
  - *python3 negativo.py baboon.png*



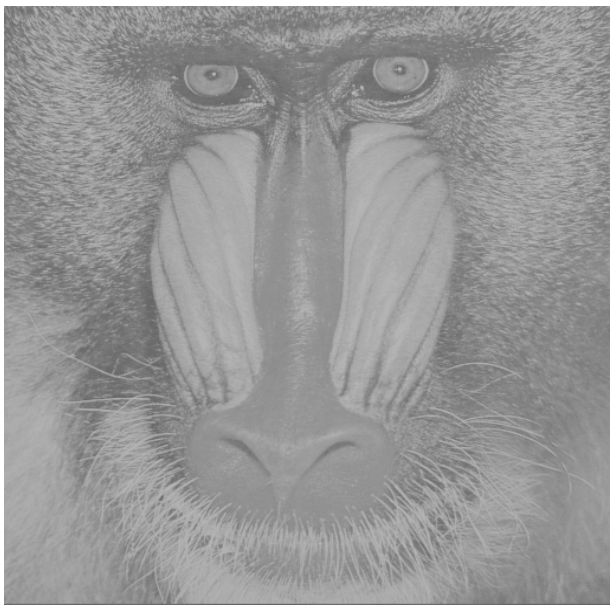
(b) negativo-baboon.png

- **vert-esp.py:** Recebe como parâmetro uma imagem e mostra seu espelhamento vertical, além de salvá-lo na pasta “resultados/espelhadas-verticalmente”. Para isso, é realizada a chamada da função **flip** sobre uma imagem de entrada **img**, com o parâmetro **axis = 0**. Exemplos de chamadas de execução:
  - *python3 vert-esp.py baboon.png*



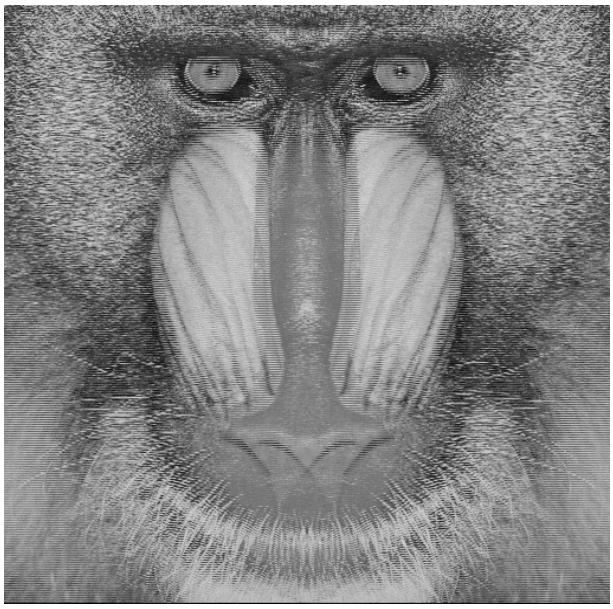
(c) espelhada-baboon.png

- **intervalo-intensidade.py:** Recebe como parâmetro uma imagem e mostra sua versão com o intervalo de intensidade alterado de [0,255] para [100,200], além de salvá-la na pasta “resultados/intervalo-de-intensidades-[100,200]”. Para isso, é realizada de forma vetorizada uma transformação linear através da equação **100 + (img/255)*100** sobre uma imagem **img** de entrada. Exemplos de chamadas de execução:
  - *intervalo-intensidade.py baboon.png*



(d) [100,200]-baboon.png

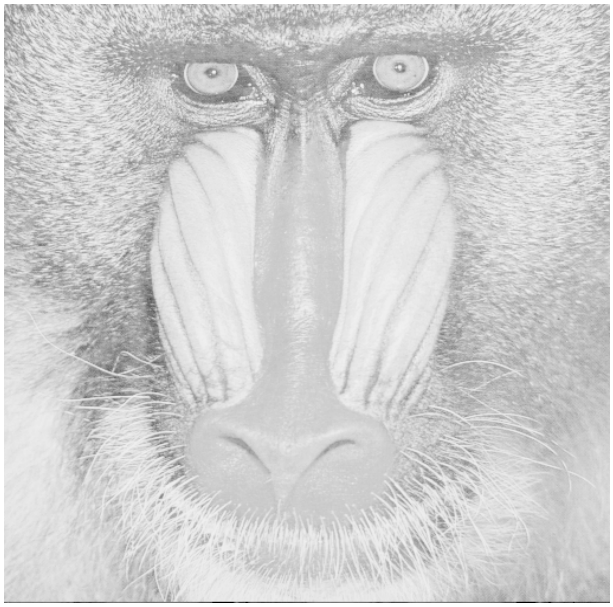
- **linhas-pares.py:** Recebe como parâmetro uma imagem e mostra sua versão com todas as linhas pares invertidas, além de salvá-la na pasta “resultados/pares-invertidas”. Para isso, é realizada a operação vetorizada **np.flip(img[::2], axis=1)** sobre uma imagem **img** de entrada. Exemplos de chamadas de execução:
  - *linhas-pares.py baboon.png*



(e) pares-invertidas-baboon.png

- **ajuste-gamma.py:** Recebe como parâmetros uma imagem e um valor real gamma e mostra sua versão com ajuste de brilho, além de salvá-la na pasta “resultados/correcao-gamma”. Para isso, inicialmente é realizada a transformação linear vetorizada **aux=img/255** sobre uma imagem **img** de entrada para se obter sua versão no intervalo [0,1]. Em seguida, é realizada a operação **aux=aux**(1/gamma)**. Por fim, é realizada outra transformação linear vetorizada **aux*255** para obter-se novamente uma imagem no intervalo [0,255]. Exemplos de chamadas de execução:

- *ajuste-gamma.py baboon.png 3.5*



(f) gamma-3.5-baboon.png

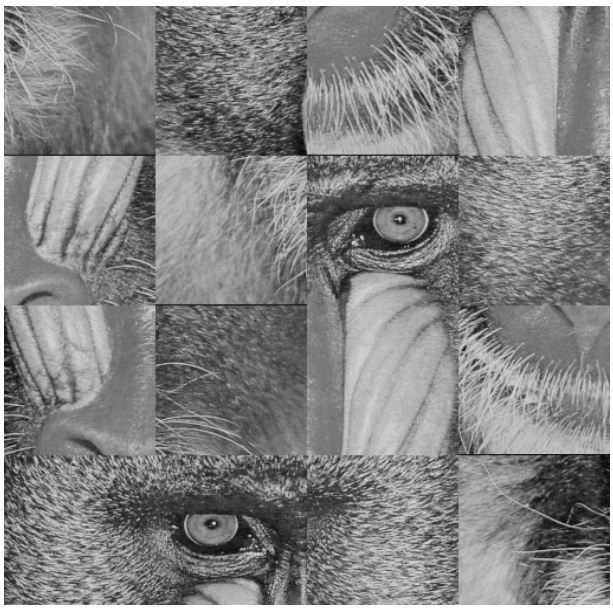
- **planos-de-bits.py:** Recebe como parâmetro uma imagem e mostra seus 8 planos de bits, além de salvá-los na pasta "resultados/planos-de-bits". Para cada um dos 8 planos é realizada a operação vetorizada **((img/2**bits)%2)*255** sobre uma imagem de entrada **img**, que retorna seu plano de ordem **bits**. Exemplos de chamadas de execução:
  - *planos-de-bits.py baboon.png*

(g) bit-7-baboon.png

(h) bit-6-baboon.png

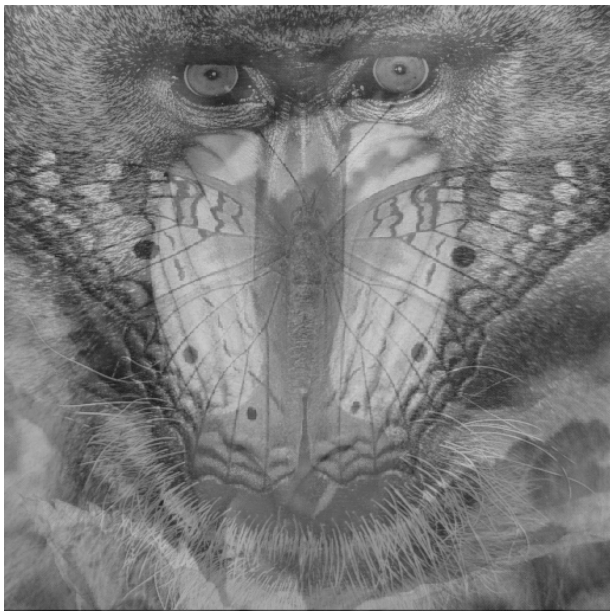
(i) bit-5-baboon.png

- **mosaico.py:** Recebe como parâmetros uma imagem e um valor inteiro n e mostra seu mosaico nxn, além de salvá-lo na pasta "resultados/mosaicos". Para isso, uma imagem **img** de entrada é divida em n^2 imagens menores através da operação vetorizada **img[y*tam:ytam+tam, x*tam:x*tam+tam]**, que são colocadas em um array **mosaico**. Em seguida, é realizada a operação **random.shuffle(mosaico)** para reagrupar de forma aleatória as imagens menores. Por fim, uma nova imagem **newimg** é montada pela operação **newimg[y*tam:y*tam+tam,x*tam:x*tam+tam] = mosaico[y*fracao+x]**. Exemplos de chamadas de execução:
  - *mosaico.py baboon.png 4*



(j) mosaico-4x4-baboon.png

- **combinacao.py:** Recebe como parâmetros duas imagens e um peso (valor real) no intervalo [0,1] e mostra a combinação entre elas, além de salvá-la na pasta “resultados/combinacoes”. Para isso, é realizada a operação vetorizada **(peso*imgA+(1-peso)*imgB)** sobre as imagens de entrada **imgA** e **imgB**. Exemplos de chamadas de execução:
  - combinacao.py baboon.png butterfly.png 0.5



(k) combinacao-0.5-baboon-butterfly.png

Para apresentar os resultados dos algorítmos de forma automatizada, foi desenvolvido um script que executa os programas de forma sequencial, visando atender as exigências do documento de especificações sobre o trabalho 0. Para executá-lo basta navegar através do terminal para a pasta Trabalho-0 e inserir o comando *sh exercicios-propostos.sh*. Ao final das execuções, os arquivos mostrados estarão disponíveis na pasta “resultados”.

3. Considerações

Visando agilizar e simplificar a implementação dos algoritmos, alguns trecho de código utilizam laços **for** para realizar algumas operações. Por exemplo, no script mosaico.py, que se utiliza alguns laços de repetição para fazer a divisão dos blocos de imagens em um *array*, para facilitar o embaralhamento desses blocos. Apesar disso, todos os processamentos relacionados a imagens foram feitos de forma vetorizada, como foi proposto pelo professor para exercitar essa forma de resolução.

Os algorítmos foram testados apenas para as imagens disponibilizadas na página da disciplina. Assim, não há garantia de que imagens de dimensões não quadráticas sejam filtradas com sucesso. A decisão de não realizar esses testes veio da ideia de que esta tarefa tinha a intenção apenas de preparar o aluno para processamentos vetorizados básicos, não havendo necessidade de aumentar a complexidade do trabalho.