

Trabalho 0 de Introdução ao Processamento de Imagem Digital

Nome: Paulo Junio Reis Rodrigues - RA: 265674

05/10/2020

1 Introdução

O objetivo deste trabalho é realizar as 5 questões do trabalho 0, para verificar se o ambiente construído pelos alunos, estão devidamente bem estruturados.

Junto com esse relatório, está sendo enviado o arquivo trabalho_0_265674.zip que possui dentro dele todos os arquivos que irão ser citados nesse relatório.

2 Programa

Todos os programas foram implementados usando Python 3.8.5, com as bibliotecas Numpy 1.19.2 e OpenCV 4.4.0.42.

3 Como executar

Há 5 scripts dentro do arquivo enviado. O primeiro script trabalho_0_exec_1.py, deve ser executado enviando como argumento uma imagem e a letra do exercício que deseja executar:

```
python trabalho\0\_exec\1.py imagens/baboon.png d
```

O segundo script trabalho_0_exec_2.py, deve ser executado enviando como argumento uma imagem e o gamma que deseja aplicar na imagem:

```
python trabalho\0\_exec\2.py imagens/baboon.png 1.5
```

O terceiro script trabalho_0_exec_3.py, deve ser executado enviando como argumento uma imagem e o plano de bit que será mostrado:

```
python trabalho\0\_exec\3.py imagens/baboon.png 5
```

O quarto script trabalho_0_exec_4.py, deve ser executado enviando como argumento uma imagem.

```
python trabalho\0\_exec\4.py imagens/baboon.png
```

O quinto script trabalho_0_exec_5.py, deve ser executado enviando como argumento uma imagem, a primeira escala, a segunda imagem e a segunda escala. Lembrando que as escalas somadas devem ser igual 1.

```
python trabalho\0\_exec\5.py imagens/baboon.png 0.4 imagens/butterfly.png 0.6
```

4 Entrada

As entradas dependem do tipo de exercício que será executado, porém cada um necessita de pelo menos 1 imagem e outros argumentos que foram citados na seção acima.

5 Saída

As saídas dependem do tipo de exercício que será executado, as saídas de cada exercício estão nos itens abaixo:

1. Primeira questão
 - (a) Imagem original.
 - (b) Retorna uma imagem negativada.
 - (c) Retorna uma imagem com espalhamento vertical.
 - (d) Retorna uma imagem com normalização de cores [100,200].
 - (e) Retorna uma imagem com linhas verticais pares invertidas.
 - (f) Retorna uma imagem com espelhamento horizontal.
2. Segunda questão: Retorna uma imagem com ajuste de brilho.
3. Terceira questão: Retorna uma imagem com o plano de bit escolhido.
4. Quarta questão: Retorna uma imagem em mosaico.
5. Quinta questão: Retorna uma imagem com a combinação de duas imagens.

6 Parâmetros Utilizados

Todas as imagens utilizadas para execução dos programas estão presentes no diretório `imagens/`, todas elas podem ser encontradas no http://www.ic.unicamp.br/helio/imagens_inclinadas_png/.

Todas as saídas de qualquer script executado, terá o nome do exercício exemplo: `exercicio_1.png`, e estarão no diretório `outputs/`.

7 Solução do exercício

Todas as soluções do trabalho utilizam a função `imread` do **OpenCV** para a leitura das imagens a partir do diretório informado.

7.1 Primeira questão

7.1.1 Letra B

Nessa parte o primeiro script chama o método **ConverteCorNegativo**, que utiliza o operador `BitWise` da biblioteca **OpenCV** que converte todos os valores sejam invertidos, deixando a imagem com aspecto negativo.

7.1.2 Letra C

Para realizá-lo o primeiro script chama o método **EspelharVerticalmente**, onde dentro dele é utilizado o operador `Slice` da biblioteca **Numpy** para inverter a imagem verticalmente.

7.1.3 Letra D

Nesse método o script chama a função **TransformarImagem**, onde é pego todos os valores de bits da imagem e é feita uma normalização de [100:200], para isso é feito um cálculo para cada valor da matriz para gerar o novo número $(f - f_{min}) / (f_{max} - f_{min}) = (g - 100) / (200 - 100)$, sendo f a imagem que será normalizada e g o novo valor da imagem.

Nessa mesma letra foi feito outro método chamado **TransformarImagemOpenCv**, que recebe os mesmos argumentos da função anterior, porém para realizar a transformação foi utilizado o método `Normalize` da biblioteca **OpenCV**.

7.1.4 Letra E

Nessa letra foi feita uma função chamada **InverteLinhas**, que utiliza os operador Slice da biblioteca Numpy que pega todas as linhas pares horizontais e inverte elas.

7.1.5 Letra F

E por último esse método foi chamado de **RefletirImagem**, onde é pego os valores iniciais da imagem até a metade do eixo horizontal ou metade mais um, dependendo respectivamente se o eixo horizontal da imagem é par ou ímpar, depois de guarda esse valores, é pego novamente os mesmo valores e com o uso da função Flip do Numpy, foi feita uma inversão horizontal dos valores, porém também também necessitava-se inverte os valores verticais também, para isso foi usada uma solução anterior da letra E.

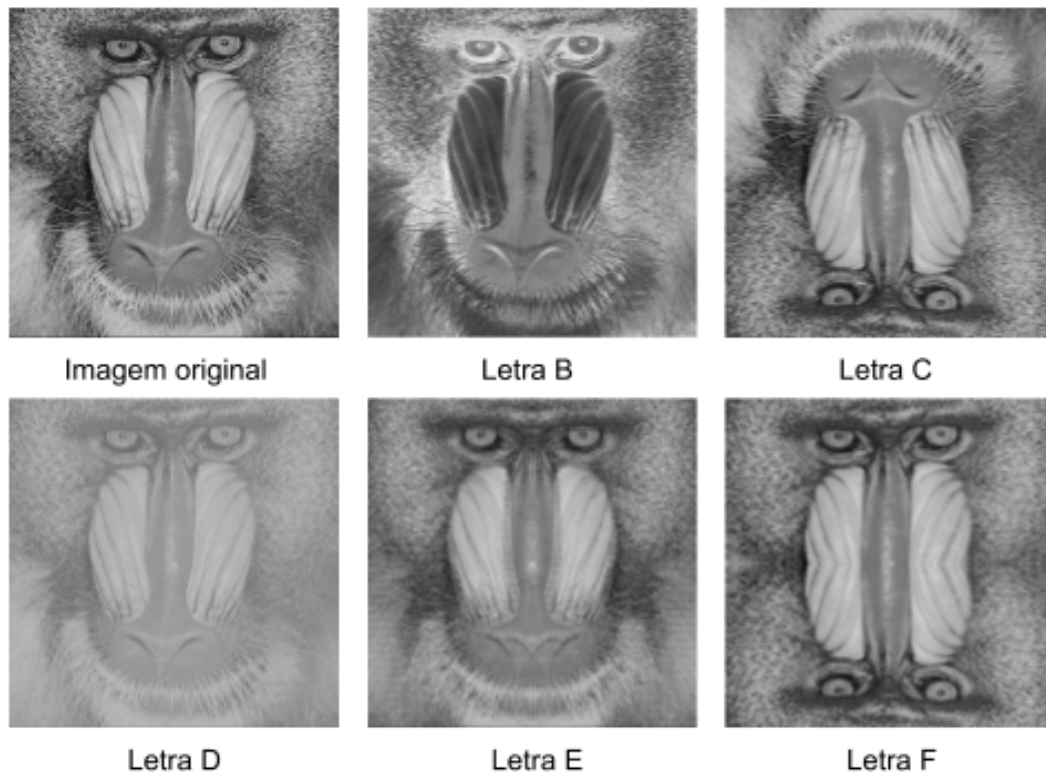


Figura 1: Imagem mostra todos os resultados da questão 1

7.2 Segunda questão

Nessa questão precisava-se de criar um método para ajustar o brilho de uma imagem, para isso foi usado o mesmo método de normalização da questão 1 letra D, porém com os valores $[0:1]$, depois todos os valores da matriz foram elevados com 1 sobre a escala escolhida, e logo depois desse passo é feita a multiplicação por 255 para cada valor, para voltar a escala de $[0:255]$.

7.3 Terceira questão

A terceira questão necessita-se encontrar todos os planos de bits da imagem escolhida, para isso é criada uma matriz onde todos os seus valores são iguais a 2 elevado ao plano de bit escolhido, a imagem criada é da mesma altura e largura da imagem original. Agora, para conseguir encontrar o plano de bit desejado, é feita uma operação **BitWise_AND** das duas imagens para encontrar qual é o bit semelhante nas duas imagens e assim encontrar formar o plano de bit escolhido, porém, para uma melhor visualização é feita uma multiplicação de 255 para cada bit.

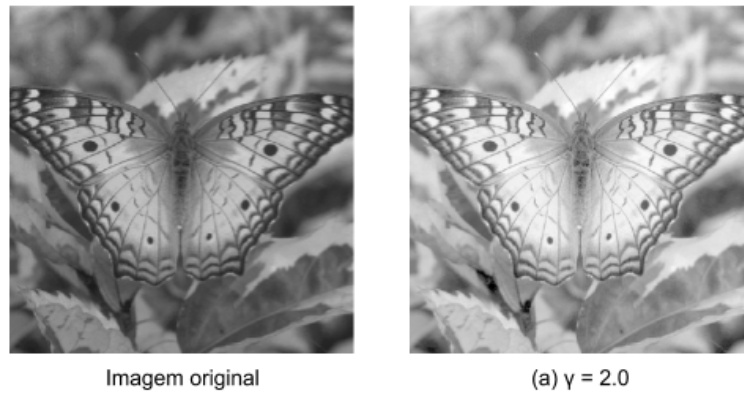


Figura 2: Imagem mostra o resultado da questão 2 com a escala 2.0



Figura 3: Imagem mostra o resultado da questão 3 do plano de bit 6

7.4 Quarta questão

Nessa questão é realizado um mosaico com a imagem original, para isso foi usado somente operadores Slices da biblioteca **Numpy**, onde é feito 16 operações, uma para cada parte do mosaico. Entretanto como o eixo horizontal e vertical do mosaico é feito de 4 linhas e 4 colunas, caso a imagem de entrada não tiver um tamanho par por par, isso significa, a imagem tiver uma largura ou altura de tamanho ímpar, pode ocasionar imperfeição na hora da criação do mosaico.

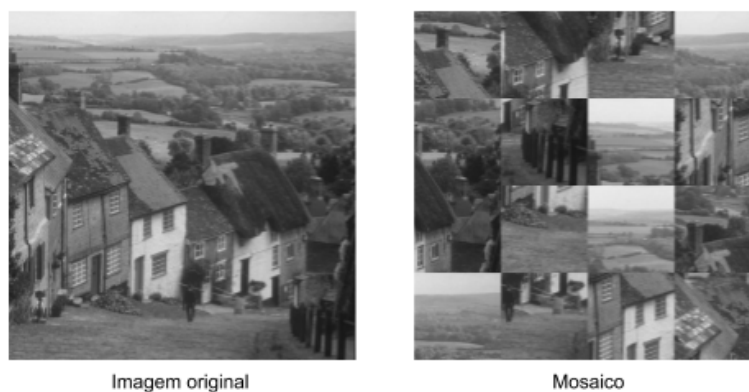


Figura 4: Imagem mostra o resultado da questão 4

7.5 Quinta questão

Essa quinta e última questão é feita uma combinação de duas imagens, para isso foi utilizada uma função chamada **addWeighted** da biblioteca **OpenCV**, esse método faz uma soma ponderada de duas matrizes, levando em consideração dois pesos, que também são informados no começo da execução do script. Uma única limitação é de que os dois pesos somados tem que ser igual a 1, caso o contrário o resultado pode não ser satisfatório.



Figura 5: Imagem mostra o resultado da questão 5