MO443 - Introdução ao Processamento de Imagem Digital - Trabalho 1

Mauro Roberto Costa da Silva - 192800
maurorcsc@gmail.com

I. Introdução

O objetivo deste trabalho é implementar algumas operações em imagens, tanto monocromáticas quanto coloridas, no domínio espacial. Nas imagens coloridas, devem ser aplicadas operações que atribuem pesos para cada banda de cor das imagens. Já nas imagens monocromáticas, devem ser aplicados filtros. A filtragem é uma técnica que consiste na transformação da imagem *pixel* a *pixel*, levando em consideração não somente a intensidade do *pixel* atual como os valores dos *pixels* vizinhos. No processo de filtragem são utilizadas matrizes quadradas, denominadas máscaras, que são aplicadas sobre a imagem obtendo algum efeito específico. Este processo consiste em percorrer toda a imagem alterando seus valores conforme os pesos da mascara e as intensidades da imagem.

II. Execução

Juntamente a este relatório, foi entregue o código fonte e os demais arquivos citados no texto. O programa foi desenvolvido em *Python* 3.6.9, utilizando as bibliotecas *numpy* 1.19.5, *OpenCv* 3.2.0 e *Scikit-Image* 0.17.2, e pode ser executado através do arquivo ***trabalho1.py***.

A. Parâmetros

Para a execução correta do programa, é necessário a utilização de alguns parâmetros, sendo eles:

- arquivo de entrada;
- tipo da imagem;
- opção de filtro a ser aplicado;
- e arquivo de saída. (opcional)

Após o nome do arquivo *python*, é necessário inserir o **nome do arquivo da imagem de entrada**, logo após deve ser inserido o valor que corresponde ao tipo da imagem, um valor que corresponde ao filtro que deve ser aplicado e, por fim, o nome da imagem de saída. O **tipo da imagem** deve ser 0 caso a imagem seja colorida e 1 caso seja monocromática. Em caso de imagem colorida, o **valor correspondente ao filtro** deve ser 1 para aplicar o filtro do Item A, 2 para o filtro do Item B ou 0 para aplicar ambos os filtros (gerando duas imagens de saída). Já em caso de imagem monocromática, o **valor correspondente ao filtro** deve ser i para aplicar o filtro h_i (com i no intervalo $[1, 9]$), 10 para a combinação dos filtros h_1 e h_2 através da expressão $\sqrt{{h_1}^2 + {h_2}^2}$ ou 0 para aplicar todos os filtros, incluindo a combinação de h_1 e h_2 (gerando uma imagem de saída para cada filtro). O **nome do arquivo de saída** é opcional, o valor padrão é "*./output.png*".

O programa deve ser executado como no seguinte exemplo:

```
python3 trabalho1.py ./imagens_coloridas/baboon.png
                                        0 1 baboon.png
```

Nesse exemplo, é recebido como imagem de entrada o arquivo *./imagens_coloridas/baboon.png*, é aplicado o filtro do Item A e é devolvido como saída o arquivo *baboon.png*. A entrada de dados foi testada com imagens cedidas pelo professor. Todas as imagens testadas estão nos diretórios "*./imagens_coloridas*" e "*./imagens_monocromaticas*".

III. Implementação

Nesta seção, serão descritas as decisões e motivações que levaram para o estado final da implementação do programa.

A imagem de entrada é lida através do método **imread** da biblioteca *OpenCV* e armazenada em uma matriz do tipo *numpy.array*. Os arquivos de saída foram gerados utilizando também a biblioteca *OpenCV*, com o método **imwrite**.

Nas subseções a seguir, está descrito como o programa foi implementado para imagens coloridas e monocromáticas.

A. Imagens Coloridas

No filtro do Item A, a imagem de entrada é dividida em três bandas (R, G e B), usando o método **split** da biblioteca *OpenCV*. Cada uma das bandas corresponde a uma matriz do tipo *numpy.array* e são usadas para calcular as novas bandas R', G' e B'. Essas novas bandas são calculadas multiplicando a linha correspondente da matriz do filtro pela transposta da imagem e somando o resultado no eixo 0, usando o método **sum**. A transposta dessa soma é armazenada na matriz da banda correspondente e, em cada uma das bandas, é aplicada uma operação do tipo $R[R > 255] = 255$ para limitar os valores em 255. É necessário usar a transposta na multiplicação para que a dimensão das bandas de cores apareça primeiro e cada banda seja multiplicada por um dos valores correspondente no filtro e é necessário fazer a transposta da soma para que a imagem volte ao *shape* original. Por fim, é usado o método **merge** para juntar as bandas geradas e formar uma nova imagem, que é o resultado da aplicação do filtro.

No Item B, o filtro foi aplicado de forma semelhante ao Item A, multiplicando a matriz do filtro pela transposta da imagem e somando o resultado no eixo 0 com o método **sum**. A transposta dessa soma é armazenada em uma matriz auxiliar I. Em seguida é aplicada uma operação do tipo $I[I > 255] = 255$ para limitar os valores em 255. A função do filtro então devolve a matriz I como resultado.

B. Imagens Monocromáticas

Neste parte do trabalho, foi aplicada a operação de correlação. Caso o filtro fosse rotacionada $180°$, seria possível obter a operação de convolução. Foi criada uma matriz h_i do tipo *numpy.array* para cada filtro i descrito no enunciado do trabalho. Para aplicar o filtro, foi criada uma função que recebe a imagem e a matriz do filtro. Nessa função, as imagens são convertidas em $2D$, caso tenham 3 dimensões e é criada uma imagem auxiliar a partir da imagem de entrada com uma moldura equivalente à metade do filtro, usando o método *numpy.pad*. A moldura adicionada na matriz auxiliar é preenchida com 0's e facilita o processo de aplicação do filtro, pois não altera o valor dos *pixels* e permite percorrer a imagem sem verificar os limites explicitamente.

A função então cria uma matriz de saída com o mesmo tamanho da imagem de entrada, e cada posição (r, c) da matriz de saída recebe o valor da soma de $padded[r : r + rows_k, c : c + cols_k] \cdot h_i$, em que *padded* é a matriz com bordas, $rows_k$ e $cols_k$ são, respectivamente, o número de linhas e colunas do filtro e h_i é o filtro passado.

Inicialmente essa função foi desenvolvida com *loops*, usando o método **sum** e o operador *, mas em seguida uma nova implementação foi feita, vetorizando todos os passos. Para isso, foi utilizado o método **view_as_windows** da biblioteca *Scikit-Image*, que permite dividir uma matriz em submatrizes no formato passado como parâmetro. Ele foi utilizado para dividir a imagem com bordas em submatrizes do tamanho do filtro passado e foi utilizado *step=1* (valor padrão) para indicar que cada *pixel* da imagem deve gerar uma submatriz (sem estourar os limites da imagem).

Em seguida, o método **numpy.einsum** foi utilizado para multiplicar cada uma das matrizes pelo filtro. Esse método permite fazer operações com matrizes de acordo com a *string* passada no parâmetro *subscripts*. Nesse caso, a *string* utilizada em *subscripts* foi "ij,rcij->rc", que indica que cada posição (i, j) do filtro deve ser multiplicada pela posição (r, c) da submatriz e o somatório dessas multiplicações deve ser armazenado na posição (r, c) da matriz de saída. Antes de ser retornada, a matriz de saída tem todos os valores negativos substituídos por 0 e todos os valores maiores que 255 substituídos por 255.

No caso da combinação dos filtros h_1 e h_2 através da expressão $\sqrt{{h_1}^2 + {h_2}^2}$ (opção 10), primeiramente os filtros h_1 e h_2 são aplicados separadamente na imagem de entrada. As matrizes resultantes são convertidas para o tipo *numpy.float32*, através do método **numpy.astype**. Essa conversão é necessária pois os valores resultantes da potenciação podem ser muito grandes e os resultantes da raiz quadrada não necessariamente são inteiros. Em seguida, cada uma das matrizes é elevada ao quadrado, somadas e a raiz da soma é tirada. A matriz resultante ao final de todas as operações é arredondada usando o método **numpy.around** e normalizada para o tipo *numpy.uint8* com o método **cv2.normalize**.

IV. Resultados e discussão

A. Imagens Coloridas

Para apresentações do resultados em imagens coloridas, será utilizado a imagem *baboon.png* colorida (Figura 1), disponibilizada pelo professor. Em seguida, serão apresentados os resultados das operações e o que foi observado.

Figure 1: Imagem original colorida

Após a aplicação das operações do Item A, a imagem ficou com aparência de envelhecida, como no filtro sépia. O resultado pode ser visto na Figura 2.

Figure 2: Imagem colorida após as operações do Item A

Já após a aplicação das operações do Item B, a imagem ficou monocromática, como podemos ver na Figura 3. Isso se deve ao fato de que a imagem resultante possui apenas uma banda de cor, que é constituída por uma média ponderada das bandas de cores da imagem original.

B. Imagens Monocromáticas

Para apresentações do resultados em imagens monocromáticas, será utilizada a imagem *baboon.png*

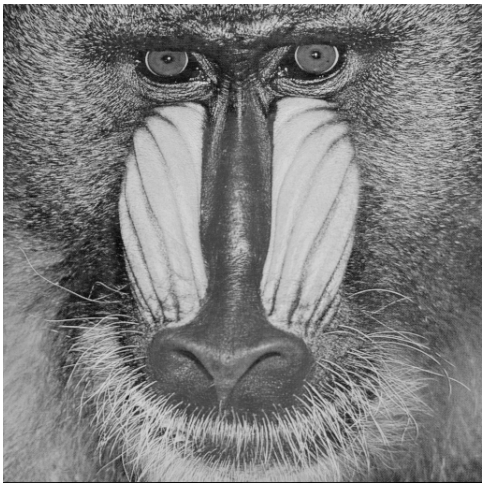

Figure 3: Imagem colorida após as operações do Item B

monocromática (Figura 4), também disponibilizada pelo professor. Em seguida, serão apresentados os resultados das aplicações dos filtros e o que foi observado.

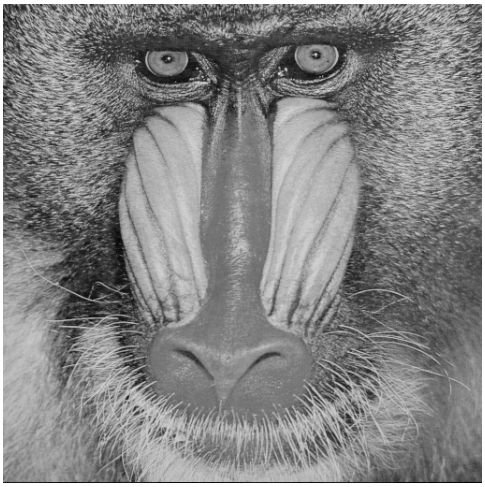

Figure 4: Imagem original monocromática

Os filtros h_1 e h_2 são conhecidos como *Sobel*. Usando esse tipo de filtro é possível estimar a presença de uma transição claro-escuro e de qual a orientação desta. Com isso, é possível fazer a detecção de contornos, já que as variações claro-escuro intensas correspondem a fronteiras bem definidas entre objetos. O filtro h_1 permite destacar os contornos à esquerda dos elementos. Já o filtro h_2 permite destacar os contornos acima dos elementos.

Após a aplicação do filtro h_1, foi obtida uma imagem escura com realce nos contornos verticais à esquerda dos elementos. Essas bordas ficaram destacadas enquanto as bordas horizontais e verticais à direita dos elementos ficaram mais difíceis de serem visualizadas. Podemos ver o resultado desse filtro na Figura 5.

Após a aplicação do filtro h_2, foi obtida uma imagem escura com realce nos contornos horizontais acima dos elementos.

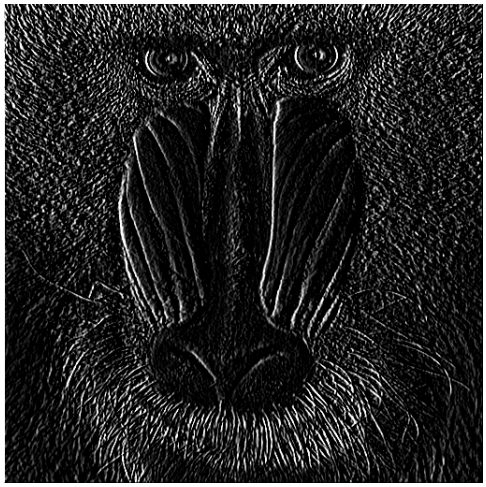

Figure 5: Imagem monocromática com o filtro h_1 aplicado

Essas bordas ficaram destacadas enquanto as bordas verticais e horizontais abaixo dos elementos ficaram mais difíceis de serem visualizadas. Podemos ver o resultado desse filtro na Figura 6.

Figure 6: Imagem monocromática com o filtro h_2 aplicado

O h_3 é um tipo de filtro utilizado para detecção de arestas. O principal objetivo desse tipo de filtro é identificar pontos em uma imagem digital em que o brilho muda acentuadamente ou, mais formalmente, apresenta descontinuidades. Os pontos nos quais o brilho da imagem muda nitidamente são normalmente organizados em um conjunto de segmentos de linha curvos denominados arestas. Podemos ver o resultado desse filtro na Figura 7.

O filtro h_4 é conhecido como *Box Blur*. Com ele, cada *pixel* tem o valor da média aritmética dos seus vizinhos, o que cria esse efeito de desfoque da imagem e permite que seja utilizado para remover ruídos. O h_4 é um *Box Blur* 3×3. Podemos ver o resultado desse filtro na Figura 8.

Os filtros h_5 e h_6 dão pesos maiores às diagonais, respectivamente, secundárias e primárias, o que faz com que as bordas

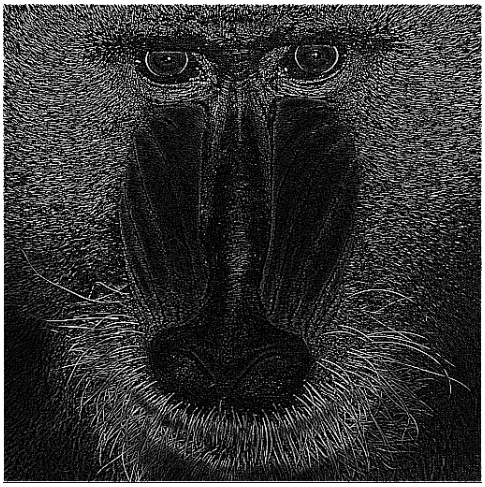

Figure 7: Imagem monocromática com o filtro h_3 aplicado

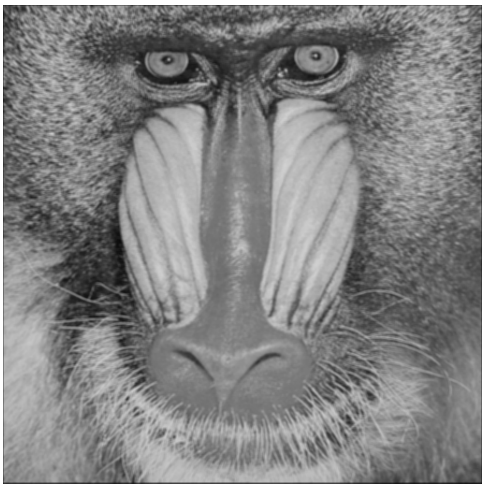

Figure 8: Imagem monocromática com o filtro h_4 aplicado

paralelas à diagonal secundária sejam realçadas no filtro h_5 e as paralelas à primária sejam realçadas no filtro h_6. Podemos ver o resultado da aplicação desses filtros nas Figuras 9 e 10.

Com a aplicação do filtro h_7, a imagem de saída possui uma aparência de alto-relevo, destacando as principais bordas dos elementos da imagem. Podemos ver o resultado desse filtro na Figura 11.

Após a aplicação do filtro h_8, foi obtida uma imagem escura com realce em praticamente todas as bordas. Podemos ver o resultado desse filtro na Figura 12.

O filtro h_9 é conhecido como *Gaussian Blur*. Esse é um filtro comumente usado em softwares gráficos para criar o efeito de desfoque e perda de detalhes. Nele cada *pixel* tem o valor da média ponderada dos seus vizinhos. O h_9 é um *Gaussian Blur* 5×5. Podemos ver o resultado desse filtro na Figura 13.

Ao combinarmos os filtros h_1 e h_2, usando a expressão $\sqrt{(h_1)^2 + (h_2)^2}$, o resultado foi uma imagem com realce nas bordas horizontais e verticais. Os filtros h_1 e h_2 destacam, respectivamente, bordas verticais à esquerda e horizontais acima dos elementos, o que gerou um realce das bordas verticais e horizontais quando combinados. Podemos ver o

Figure 9: Imagem monocromática com o filtro h_5 aplicado

Figure 10: Imagem monocromática com o filtro h_6 aplicado

Figure 11: Imagem monocromática com o filtro h_7 aplicado

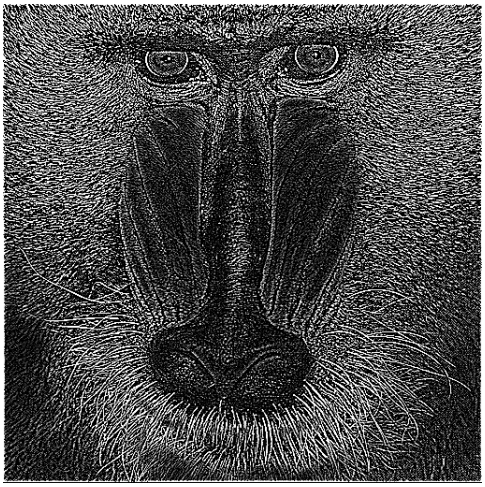

Figure 12: Imagem monocromática com o filtro h_8 aplicado

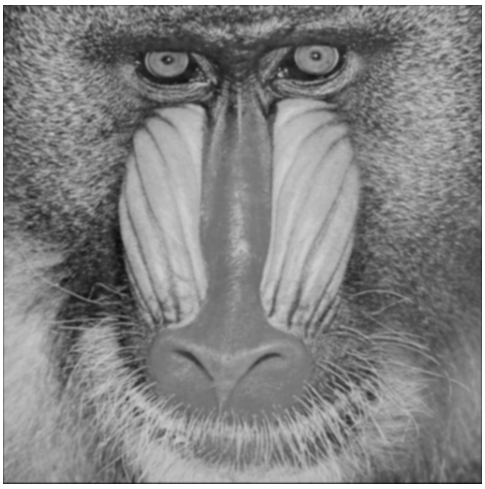

Figure 13: Imagem monocromática com o filtro h_9 aplicado

resultado dessa combinação de filtros na Figura 14.

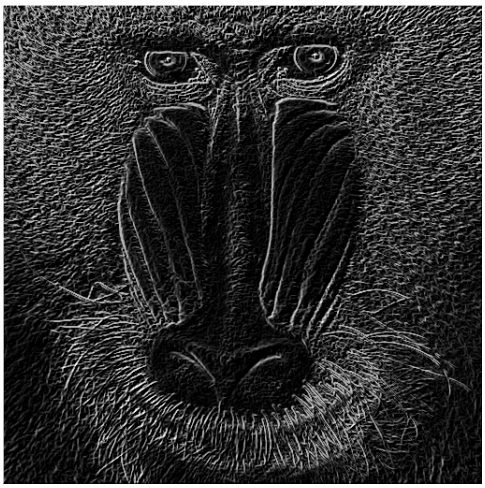

Figure 14: Imagem monocromática com o filtro h_1 e h_2 combinados

V. Conclusão

Pudemos perceber os resultados das operações aplicadas tanto nas imagens coloridas quanto nas monocromáticas. A implementação para a aplicação dos filtros obteve resultados semelhantes aos das funções já existentes em bibliotecas, como a **filter2D** da biblioteca *OpenCV*. Também foi possível aprender mais sobre vetorização, já que foi necessário aplicar funções vetorizadas durante o processo para obter um bom desempenho de tempo.