

Universidade Estadual de Campinas
Instituto de Computação

Introdução ao Processamento Digital de Imagem – MO443
Professor Hélio Pedrini

Trabalho 01 – 1º Semestre de 2023

Maurício Pereira Lopes – RA 225242

I – Introdução

Este relatório traz uma análise dos resultados obtidos no desenvolvimento das atividades propostas no Trabalho 01 da disciplina MO443.

O objetivo do trabalho é realizar alguns processamentos básicos em imagens digitais. O trabalho apresentou oito questões que foram realizadas através de códigos em Python.

Por se tratar de oito atividades não diretamente relacionadas, optei por implementar um programa para cada questão. A próxima sessão descreve como é a utilização dos programas bem como as dependências para o funcionamento deles.

Também optei por não utilizar nenhum pacote especializado em manipulação de imagens. Busquei aplicar os conceitos estudados e implementar os algoritmos somente com manipulação de numpy arrays e comandos básicos do Python.

II – Sobre os Programas Desenvolvidos

Foram entregues um arquivo *.zip* contendo um arquivo *.py* para cada uma das oito questões do trabalho. Os nomes dos arquivos seguem o padrão abaixo:

trabalho01-questao1.x.py

(Onde x representa o número da questão)

Cada um dos programas, ao ser executado, além de apresentar o resultado em tela, também o salva num arquivo *.png* cujo nome segue o padrão indicado abaixo:

trabalho01-questao1.x.png

(Onde x representa o número da questão)

Também faz parte da entrega, no mesmo arquivo *.zip*, um arquivo *jupyter notebook* de nome *trabalho01.ipynb* que contém o código desenvolvido para cada uma das questões. São os mesmos

códigos dos arquivos *.py* fornecidos de forma individual para cada questão. Se trata apenas de uma segunda forma de execução dos programas.

A. Dependências

Todos os programas funcionam de forma independente e não levam nenhum parâmetro de entrada.

É necessária conexão com a internet pois os arquivos de imagens utilizados são lidos através de url's de endereços na Internet.

Para execução de qualquer um deles basta fazer a chamada na linha de comando (*e.g., python3 trabalho01-questao1.1.py*).

As bibliotecas que foram carregadas nos programas são as listadas abaixo:

- 1) *imageio*: oferece uma interface de leitura e escrita de dados em forma de imagens.
- 2) *matplotlib.pyplot*: criação de gráficos e visualizações incluindo dados de imagens.
- 3) *numpy*: manipulação de arrays.
- 4) *random*: foi utilizada para ordenar sub-imagens de forma aleatória na questão 01.

Cada programa irá executar ao ser chamado e irá gerar duas saídas, uma em tela e outra em arquivo de imagem *.png* com igual conteúdo e com o mesmo nome do programa.

III – Análise dos Resultados das Questões

1 Desenvolvimento das questões propostas no trabalho

1.1 Mosaico

A questão solicita construir um mosaico de 4x4 blocos a partir de uma imagem monocromática. Em aula o professor instruiu a classe de que poder-se-ia seguir uma ordem aleatória para a reordenação dos blocos de imagens no mosaico e foi essa a estratégia que adotei.

A princípio, minha abordagem foi de buscar uma maneira de quebrar a imagem original de dimensões (512, 512) utilizando o comando *numpy.reshape*. Não encontrei nenhuma combinação ou alternativa que me permitisse gerar os blocos de imagem com este comando.

A opção adotada foi de usar *slicing* do array dentro de *for loops*. A imagem foi quebrada em 16 blocos de dimensões (128, 128) e adicionadas a uma lista através de uma linha de comando com loops no formato de *list comprehension*, gerando assim uma lista de dimensões (16, 128, 128).

Em seguida, os elementos desta lista foram reordenados através do comando *random.shuffle()* e, depois, a lista foi convertida em um array ainda de dimensões (16, 128, 128).

O próximo passo foi reagrupar os blocos de imagem numa única imagem de dimensões (512, 512). Novamente não encontrei solução para implementar esta tarefa apenas fazendo o *reshape* do array.

A opção foi o uso de *for loops*, novamente. Para isso, criei uma imagem vazia de (512, 512) pixels e a percorri em seus índices de forma a poder escrever cada bloco de imagem numa posição sequencial da imagem final, fazendo com que os blocos ordenados de forma aleatória no array, viessem para a imagem final também nesta ordem aleatória.

Uma das possíveis imagens finais, bem como a imagem original desta questão, são apresentadas na *figura 1*.

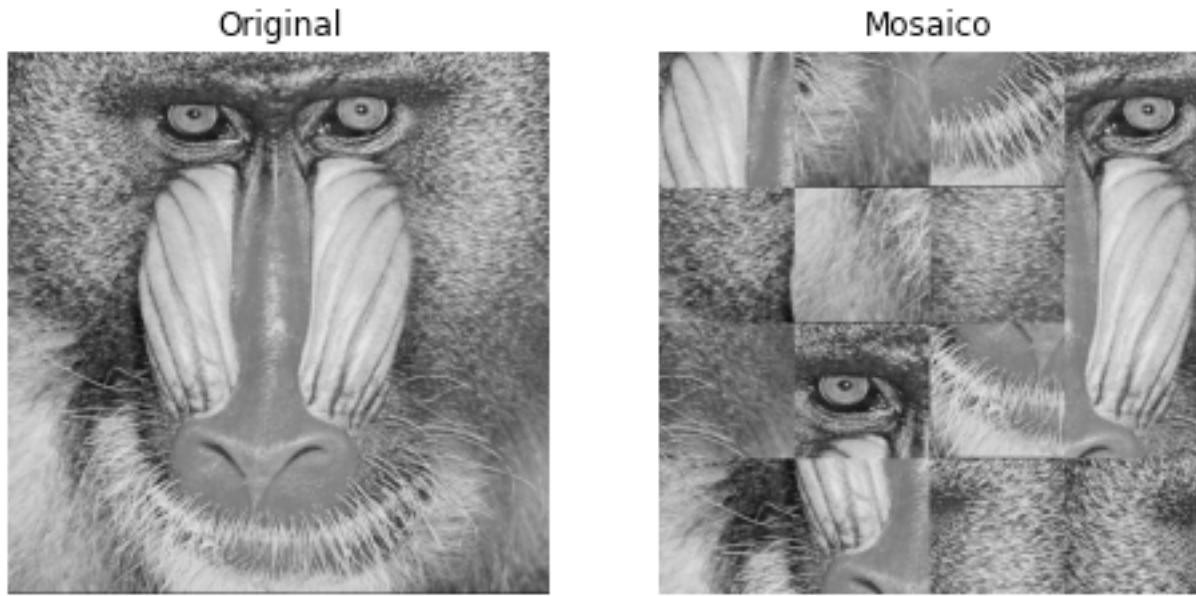


Figura 1 - Questão 1 - Comparação da imagem original com seu mosaico

1.2 Combinação de Imagens

A questão pede para combinar duas imagens monocromáticas de mesmo tamanho por meio da média ponderada de seus níveis de cinza.

Dadas duas imagens A e B, três novas imagens deverão ser criadas:

$$\text{Imagem1} = 0.2 * A + 0.8 * B$$

$$\text{Imagem2} = 0.5 * A + 0.5 * B$$

$$\text{Imagem 3} = 0.8 * A + 0.2 * B$$

As imagens originais utilizadas nesta questão estão indicadas na *figura 2*.

A abordagem para esta questão foi de criar uma lista com os pesos a serem utilizados e varrer-los em um *for loop* para, a cada passagem, gerar uma nova imagem que fosse a combinação das duas originais aplicando-se os devidos pesos a cada uma.

O resultado é claramente a sobreposição das duas imagens com seus valores de pixels atenuados pelos pesos aplicados a cada uma delas, de forma que o peso menor faz a imagem ficar mais “apagada” em comparação com a de maior peso.

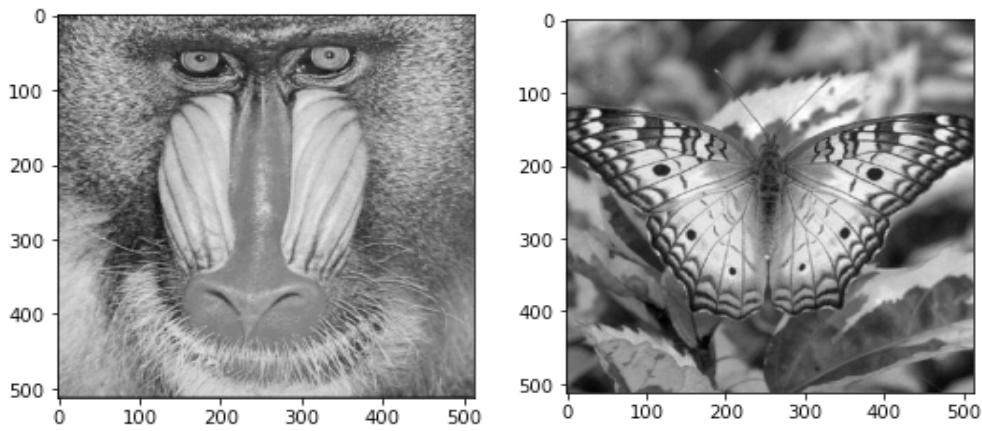


Figura 2 - Questão 2 - Imagens originais utilizadas no problema

Como os pesos somados resultam em 1, isso significa que a soma não passará dos limites de 0 e 255 para os valores dos pixels.

As imagens combinadas resultantes são apresentadas na *figura 3*.

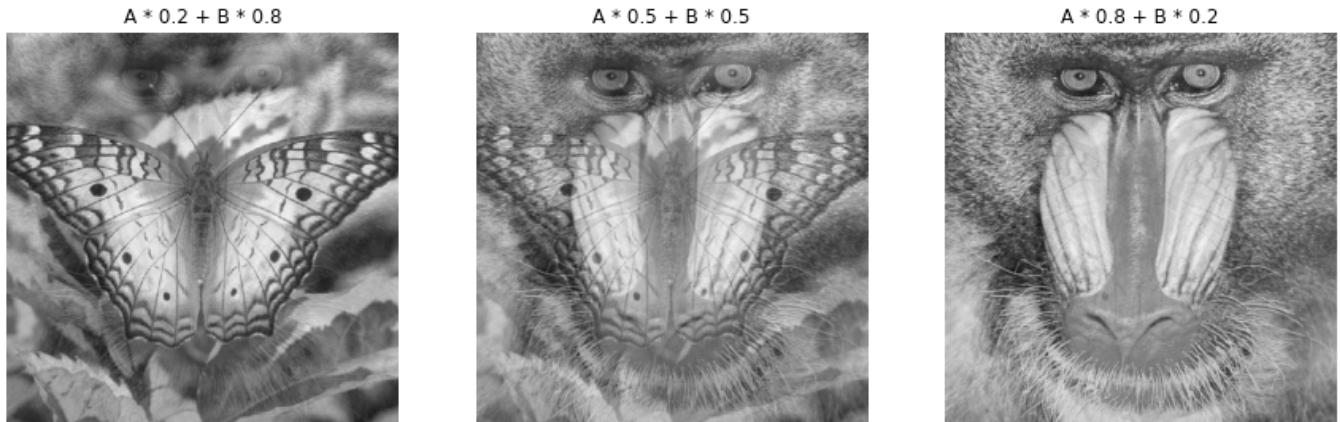


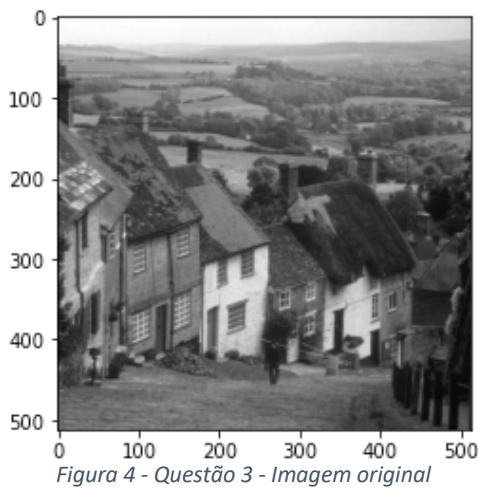
Figura 3 - Questão 2 - Imagens combinadas com pesos diferentes

1.3 Transformação de Intensidade

A questão 3 pede a transformação do espaço de intensidades (níveis de cinza) de uma imagem monocromática de forma a obter as seguintes transformações:

- Imagen original*
- Imagen em negativo*
- Intervalos de intensidades de 0 a 200*
- Linhas pares com valores invertidos da esquerda para a direita*
- Espelhamento da metade superior da imagem*
- Espelhamento vertical de toda a imagem*

- a) A imagem original utilizada nesta questão é mostrada na *figura 4*.



- b) A imagem em negativo foi obtida pela seguinte expressão:

$$\text{negativo} = 255 - \text{original}$$

Com essa simples operação, que subtrai o valor de cada pixel do valor máximo de 255, é feita a inversão dos valores dos níveis de cinza de forma a transformar valores 0 em 255, 255 em zero, e o mesmo em toda a escala de 256 tons de cinza.

O resultado desta operação sobre a imagem original é mostrado na *figura 5*.



Figura 5 - Questão 3 - Imagem em negativo

- c) A conversão do intervalo de intensidades de tons de cinza de $[0, 255]$ para $[100, 200]$, foi feita utilizando-se o mesmo conceito do *Teorema de Tales* e à definição do *teorema fundamental da proporcionalidade* que é mostrado na *figura 6*.

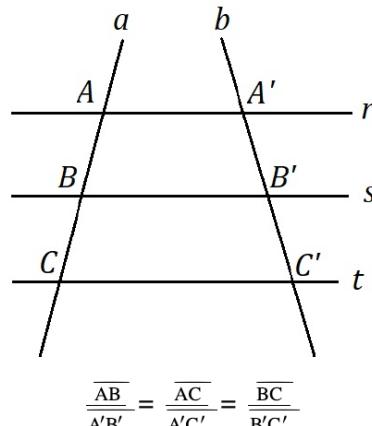


Figura 6 - Questão 3 - Teorema Fundamental da Proporcionalidade

Aplicando-se esse teorema ao caso das imagens cheguei à fórmula para converter os intervalos de intensidade de tons de cinza. A fórmula final é mostrada abaixo, onde i representa a intensidade do pixel:

$$i_{novo} = i_{atual} \frac{100}{255} + 100$$

Essa redução do intervalo de intensidades de tons de cinza diminui a diferença entre os valores dos pixels e fazem com que tons mais próximos do preto (valor 0) e do branco (255) sejam removidos da imagem. Isso equivale a uma redução no contraste da imagem, diminuindo o nível de detalhes dela.

O resultado desta mudança na imagem original é mostrado na *figura 7*. Foi necessário utilizar os parâmetros $vmin = 0$ e $vmax = 255$ no comando *imshow* do pacote *Pyplotlib* para garantir a apresentação correta da imagem resultante. Não fazer isso faz com que o comando faça uma correção automática dos valores dos pixels buscando apresentar a imagem com todos os tons de cinza possíveis com os 8 bits de profundidade.



Figura 7 - Questão 3 - Imagem com redução do intervalo de intensidades

- d) No item (d) foi solicitado fazer a inversão dos valores dos pixels das linhas pares da imagem original de forma que os valores do final da linha substituam os valores no início da linha, e vice-versa.

Executei esta tarefa criando uma imagem, cópia da original, e fazendo um *slice* da imagem que pegasse cada linha par do fim para o início, e substituindo esta mesma linha na imagem original. O *for loop* foi usado para varrer as linhas pares, mas não foi preciso deste recurso para fazer a inversão dos valores.

A imagem com as linhas pares invertidas é mostrada na *figura 8*.



Figura 8 - Questão 3 - Linhas pares com valores invertidos

- e) Este item pedia que se fizesse o espelhamento da metade superior da imagem original para a parte inferior da imagem.

Esta tarefa foi feita substituindo de todas as linhas da imagem a partir da linha 256 pela metade superior da imagem, mas iniciando pela linha 255 e indo até a linha 0. Esta operação pode ser feita fazendo-se o *slicing* adequado da imagem. Tomei a metade superior da linha 0 até a 255, e as inverti com os recursos de indexação do Python.

Este foi o comando usado para gerar a imagem final:

```
new_im[256:, :] = new_im[0:256, :][::-1, ...]
```

A imagem resultante está na *figura 9*.



Figura 9 - Questão 3 - Espelhamento da metade superior da imagem

- f) Aplicar um espelhamento vertical na imagem levando-se em conta todas as linhas da mesma.

Aqui, utilizei o mesmo recurso do item anterior para apenas gerar uma nova imagem onde a primeira linha é a última da imagem original, e assim por diante. Novamente, isso foi feito utilizando-se os recursos de indexação dos elementos do array disponíveis no Python e na biblioteca Numpy. A indexação das linhas de traz para frente foi feita com `[::-1, ...]`.

O resultado desta transformação está na *figura 10*.



Figura 10 - Questão 3 - Espelhamento vertical completo

1.4 Imagens Coloridas

- a) Nesta atividade foi solicitado alterar uma imagem colorida RGB através de três operações aplicadas a cada uma das camadas de cor:

$$R' = 0.393R + 0.769G + 0.189B$$

$$G' = 0.349R + 0.686G + 0.168B$$

$$B' = 0.272R + 0.534G + 0.131B$$

Estas transformações vão alterar cada uma das camadas de cor, mas levando-se em conta todas as camadas multiplicadas por diferentes pesos.

A imagem utilizada nesta transformação tem as dimensões (512, 512, 3) e é apresentada na *figura 11*.



Figura 11 - Questão 4 - Imagem original - Fonte: <https://webpages.tuni.fi/imaging/tampere17/t095.png>

Nesta atividade criei uma nova imagem com as mesmas dimensões da imagem original e alterei cada uma das camadas de cor segundo as expressões fornecidas.

Depois disso, apliquei a mesma normalização da imagem para garantir que todos os pixels tivessem valores entre 0 e 255. Além disso, garanti que todos os valores dos pixels fossem inteiros.

O resultado foi uma imagem também com três camadas de cores, mas agora as cores combinadas dão um tom de “sépia” à imagem final.

O resultado é visto na *figura 12*.



Figura 12 - Questão 4 - Imagem em tons de sépia

- b) Alterar a imagem original RGB de forma que ela contenha apenas uma banda de cor, cujos valores devem ser calculados pela média ponderada abaixo:

$$I = 0.2989R + 0.5870G + 0.1140B$$

Para resolver esta tarefa, cada canal de cor foi selecionado usando os recursos de indexação do Numpy e, então, foram multiplicados pelos pesos da expressão acima. A imagem resultante não tinha mais 3 camadas de cores, passando a ter dimensão (512, 512).

O resultado desta transformação gerou uma imagem em tons de cinza com todos os detalhes da imagem original, exceto as cores. Nota-se assim, que a transformação de uma imagem RGB em uma imagem em tons de cinza não é obtida fazendo-se somente a soma das informações de cada canal. Portanto, cada canal de cores tem peso diferente na formação da mesma imagem em tons de cinza.

Isso pode ser visto na *figura 13* que mostra a imagem com os canais de cores somados com os pesos e sem os pesos.

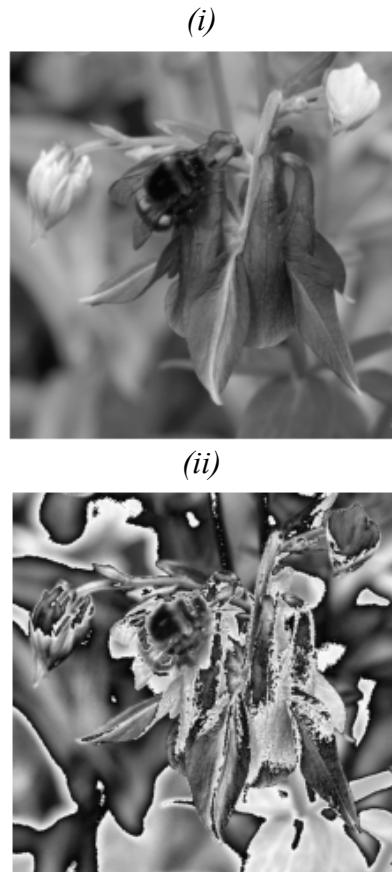


Figura 13- Questão 4 - (i) Imagem gerada pela soma dos canais de cores com os pesos solicitados, (ii) Imagem com os canais de cores somados sem pesos

1.5 Ajuste de Brilho

Esta atividade solicita aplicar a correção $gama$ para ajustar o brilho de uma imagem monocromática A de entrada e gerar uma imagem monocromática B de saída.

São indicadas três etapas para se fazer essa transformação:

- (i) Converter as intensidades dos pixels para o intervalo $[0, 1]$
- (ii) Aplicar a equação $B = A^{1/\gamma}$
- (iii) Converter os valores resultantes de volta para o intervalo $[0, 255]$

Foram definidos valores de $gama = [1, 1.5, 2.5, 3.5]$. sendo que $gama = 1$, representa a imagem original.

A conversão do intervalo de intensidades para $[0, 1]$ foi feita seguindo o mesmo teorema fundamental da proporcionalidade usando anteriormente.

A aplicação da equação (ii) foi feita à imagem original e os resultados foram salvos em uma lista para serem plotados em seguida.

O resultado pode ser visto na *figura 14*.

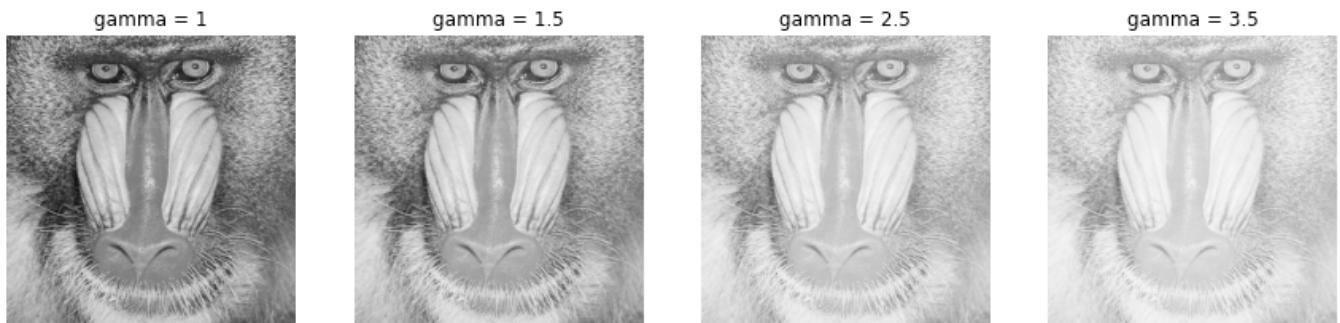


Figura 14 - Questão 5 - Ajuste de Brilho

1.6 Quantização de Imagens

A quantização se refere ao número de níveis de cinza usados para representar uma imagem monocromática. Ela está relacionada à profundidade de uma imagem e corresponde ao número de bits necessários para armazenar a imagem.

Esta tarefa solicita representar uma imagem monocromática com diferentes níveis de quantização.

Realizei esta atividade convertendo o intervalo de intensidades de cinza da imagem original para a faixa $[0, 1]$ e multipliquei esta imagem pelo valor do número de bits desejado. Isso fará com que o valor resultante tenha o número de bits da quantização.

As imagens resultantes são mostradas na *figura 15*.

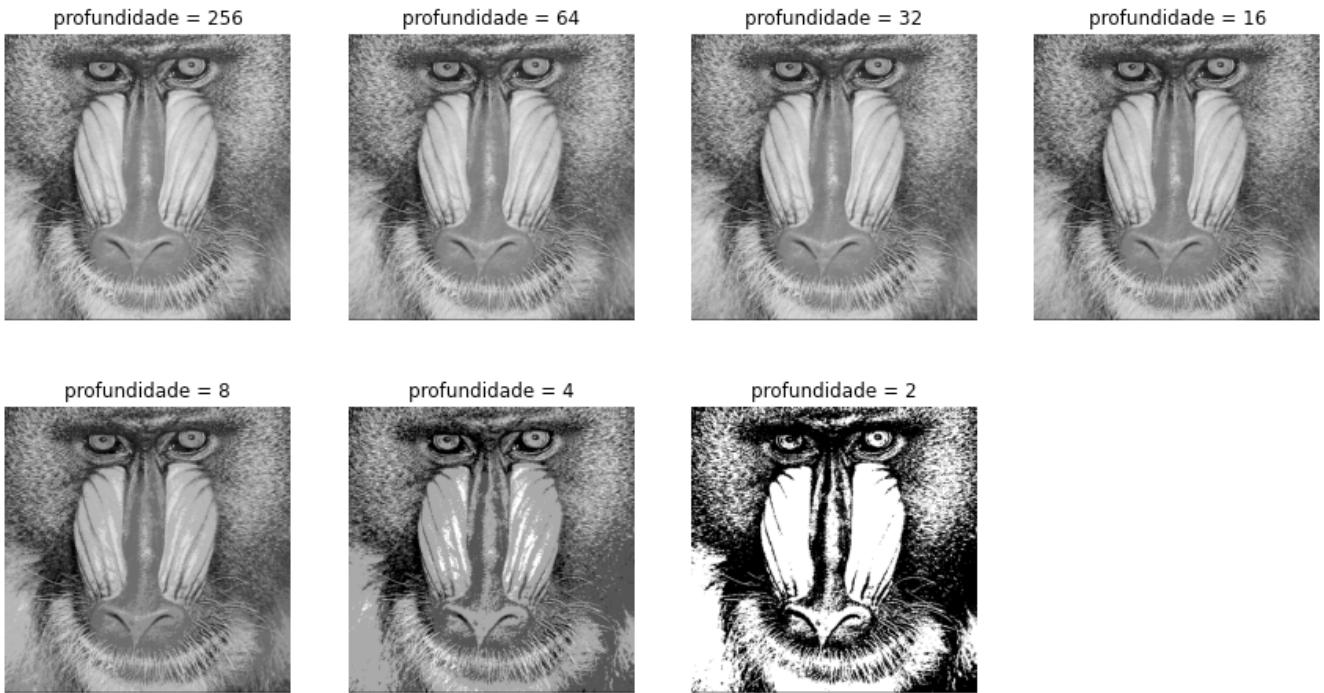


Figura 15 - Questão 5 - Quantização de imagens

1.7 Planos de Bits

Esta atividade pede para se extrair os planos de bits de uma imagem monocromática.

Os níveis de cinza de uma imagem monocromática com m bits podem ser representados na forma de um polinômio de base 2:

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \dots + a_12^1 + a_02^0$$

O plano de bits de ordem 0 é formado pelos coeficientes a_0 de cada pixel, enquanto o plano de bits de ordem $m-1$ é formado pelos coeficientes a_{m-1} .

Implementei a solução gerando uma imagem para cada plano de bits e salvando como elementos de uma lista para serem plotadas em seguida.

Cada imagem foi gerada através da operação AND aplicada bit a bit entre o valor de cada pixel e o valor correspondente ao a cada plano de bits ($2^0, 2^1, 2^2, 2^3, \dots$).

Os planos correspondentes aos bits menos significativos são extremamente ruidosos e não permitem reconhecer um padrão visual. Os bits mais significativos já permitem o reconhecimento visual da imagem, mas sem o nível de detalhes armazenados nos bits menos significativos.

As imagens resultantes para cada plano de bits estão mostradas na figura 16.

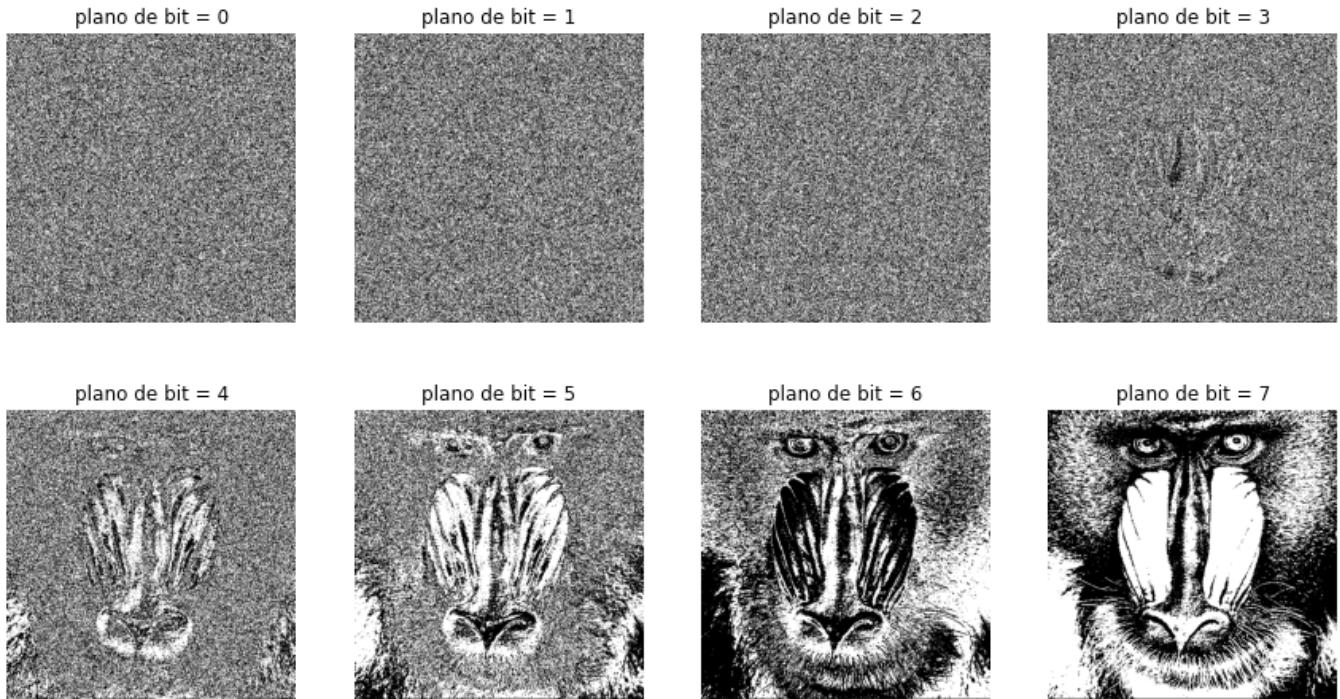


Figura 16 - Questão 7 - Imagens em cada plano de bits

1.8 Filtragem de Imagens

Esta última tarefa solicita a aplicação de alguns filtros de imagens a uma imagem monocromática e a avaliação do resultado de forma a qualificar cada um dos filtros e que tipo de filtragem cada um deles faz. Foram fornecidas 11 diferentes máscaras de filtros, h_1 até h_{11} , para serem aplicados à imagem sendo que para os filtros h_3 e h_4 , deve-se, além de aplicar os filtros individualmente, aplicá-los de forma combinada segundo a expressão: $\sqrt{(h3)^2 + (h4)^2}$

A imagem original tem dimensões (512, 512) e desenvolvi minha solução para uso específico com imagens dessa dimensão.

Optei por não utilizar nenhum pacote pronto para a aplicação dos filtros e fiz minha implementação de forma manual somente com comandos básicos de Python e Numpy.

Para isso, criei uma função Python chamada *filtrar* que recebe dois parâmetros: uma imagem de dimensões (512, 512) e um *numpy array* de duas dimensões representando a máscara do filtro a ser aplicado. Esta função define o tamanho do *pad*, cria o *pad* na imagem, aplica o filtro à imagem através de um *for loop* e redimensiona a imagem final para garantir o intervalo de intensidades de 0 a 255.

Para a operação de convolução do filtro com a imagem, apliquei um *padding* com o comando *numpy.pad()* fornecendo como parâmetros a imagem, o tamanho do *pad* e o modo *reflect*.

Para definir o tamanho do *pad* eu dividi a dimensão 0 do filtro por 2 para que fossem adicionadas essa quantidade de camadas externas à imagem. O preenchimento dos valores do *pad* foi feito com o

parâmetro *reflect* que reflete os valores internos da imagem correspondentes à quantidade de camadas adicionadas pelo *pad*. Com isso, evita-se preencher o *pad* com zeros e usam-se valores existentes na imagem, mantendo o mesmo padrão das bordas da própria imagem.

A aplicação do filtro é feita posicionando o primeiro elemento do filtro sobre o primeiro elemento da imagem com o *pad*, fazendo a multiplicação de cada elemento do filtro com o elemento correspondente da imagem, somando o resultado destas multiplicações e colocando o valor resultante na primeira posição de uma nova imagem. O filtro é, então, movimentado uma posição para a direita e o processo é repetido até se chegar à última coluna da imagem. Então o filtro é reposicionado na primeira coluna novamente, porém na segunda linha. Isso é feito até que toda a imagem original com o *pad* seja varrida e uma nova imagem de $[512, 512]$ seja criada com os resultados de cada etapa da operação de convolução.

A nova imagem, pode conter valores negativos ou fora dos limites da faixa de 0 a 255. Para corrigir isso, minha função *filtrar()* faz a normalização da imagem para intensidades na faixa de 0 a 255.

Por fim, a função retorna a nova imagem filtrada.

1.8.1 Aplicação do filtro *h1*

O filtro *h1* é um filtro passa altas que faz detecção de pontos que possuem nível diferente do que os pontos ao seu redor. Isso dá destaque a ruídos e a regiões de maior variância nos valores dos pixels da imagem.

O filtro *h1* é mostrado na *figura 17*.

$$h_1 = \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & -1 & 0 & 0 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline -1 & -2 & 16 & -2 & -1 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline 0 & 0 & -1 & 0 & 0 \\ \hline \end{array}$$

*Figura 17 - Questão 8 - Filtro *h1* - Passa altas*

A imagem original e a resultante da aplicação do filtro h_1 é vista na *figura 18*.

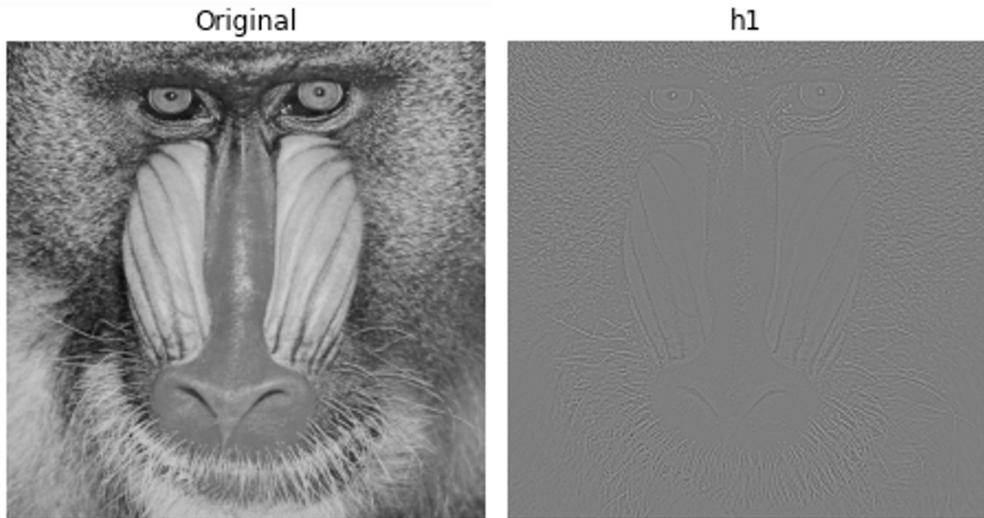


Figura 18 - Questão 8 – Aplicação do filtro h_1

Percebe-se que a imagem filtrada apresenta detalhes mais ruidosos da imagem, apesar da imagem não ter tanta nitidez. Mas é visível que o filtro realçou detalhes mais ruidosos da imagem.

Eu não consegui encontrar uma maneira de melhorar o contraste da imagem a tempo para a finalização do trabalho, mas esse seria um próximo passo de melhoria para o algoritmo que desenvolvi.

1.8.2 Aplicação do filtro h_2

O filtro h_2 se trata de um filtro passa baixas Gaussiano que suaviza a imagem eliminando pontos de maior variância em relação aos seus vizinhos, mas sem perder muito da nitidez da imagem.

O formato do filtro h_2 é visto na *figura 19*.

$$h_2 = \frac{1}{256} \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

Figura 19 - Questão 19 - Filtro h_2

As imagens original e filtrada pelo filtro h2 são mostradas na *figura 20*.

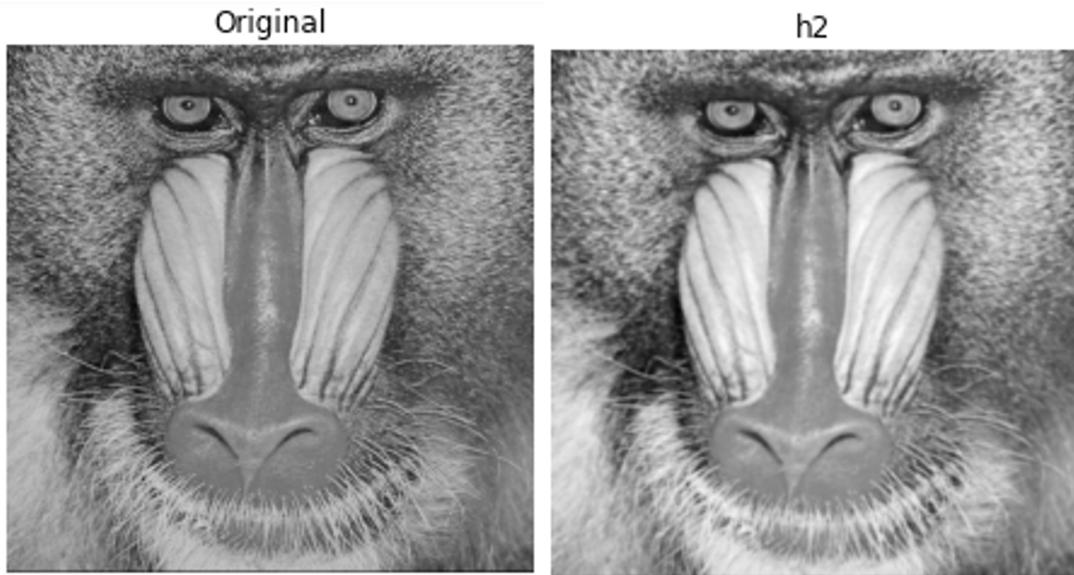


Figura 20 - Questão 20 - Aplicação do filtro Gaussiano h2

Nota-se na imagem filtrada, um discreto borramento em comparação com a imagem original. Há uma suavização de detalhes mais finos da imagem, mas ainda assim, há bastante nitidez na imagem.

1.8.3 Aplicação do filtro h3, h4 e a combinação de h3 com h4

Os filtros h3 e h4 fazem parte da técnica dos *Filtros de Sobel* que são capazes de encontrar variações nas intensidades das imagens bem como o ângulo correspondente à direção de maior variação de intensidade. Portanto se tratam de filtros passa altas.

Especificamente o filtro h3 realça bordas na direção do eixo *x* da imagem. Já o filtro h4, faz esse realce de bordas na direção do eixo *y*.

Os filtros h3 e h4 são mostrados na *figura 21*.

$$h_3 = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad h_4 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Figura 21 - Questão 8 - Filtros h3 e h4

As imagens filtradas por h3 e h4 são mostradas na *figura 22*.

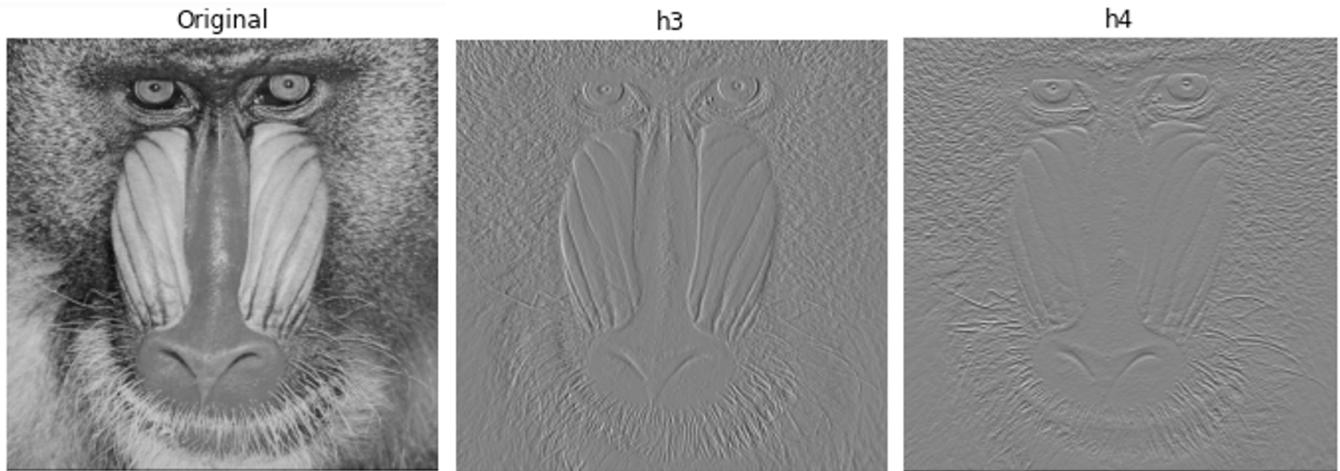


Figura 22 - Questão 8 - Aplicação dos filtros de Sobel h3 e h4

Nota-se que em h3 há um sombreamento ou realce maior em bordas que estão mais na vertical. Estas são detectadas pelo movimento da máscara h3 na direção do eixo x.

Já na imagem filtrada por h4, este realce é maior nas linhas ou bordas que estão mais na horizontal. Esse realce é gerado pelo movimento da máscara no eixo y.

Um ponto de interesse para se notar essa diferença no realce são os olhos do animal na imagem. Há uma sombra mais destacada na vertical em h3 e mais destacada na horizontal em h4.

Os filtros de Sobel h3 e h4 permitem se chegar à magnitude da variação de intensidade assim como o ângulo da maior variação. A tarefa pediu para se aplicar o cálculo de magnitude desses filtros, o que é feito através da expressão abaixo:

$$G = \sqrt{(h3 * I)^2 + (h4 * I)^2}$$

Onde G é a magnitude e I é a imagem original.

A imagem resultante é mostrada na *figura 23*.

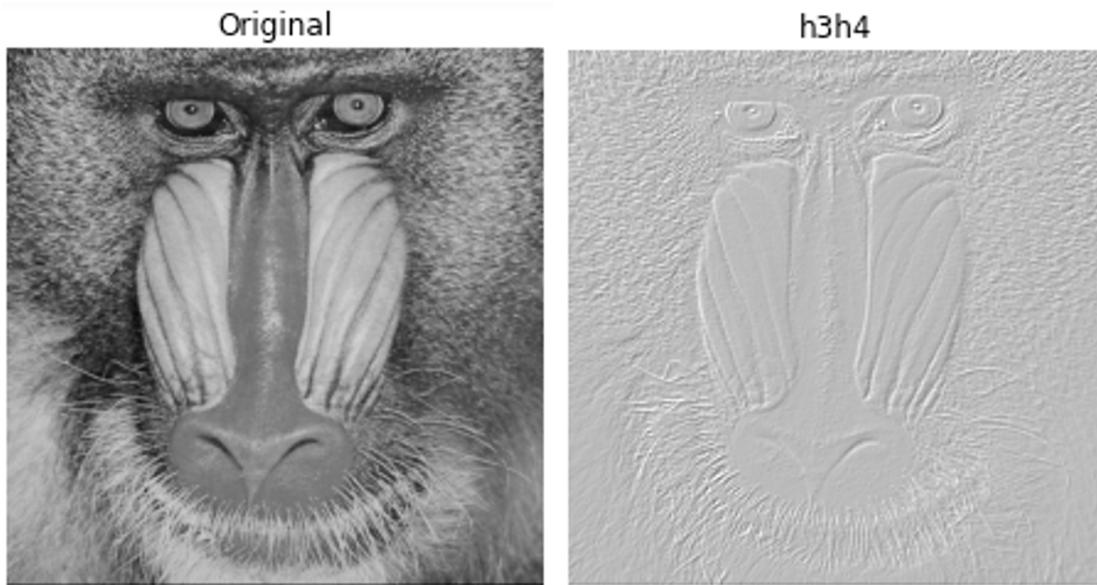


Figura 23 - Questão 8 - Aplicação dos Filtros de Sobel para se obter a magnitude das variações de intensidade na imagem através de h_3 e h_4

Percebe-se um realce mais nítido e não concentrado em somente uma das direções das bordas da imagem, apesar do contraste ainda ser baixo.

1.8.4 Aplicação do filtro h_5

O filtro h_5 é um filtro passa altas capaz de realçar pontos de maior variação de intensidade em relação aos pontos vizinhos. Em vez do ponto ser suavizado, ele é realçado.

O formato do filtro h_5 é mostrado na *figura 24*.

$$h_5 = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Figura 24 - Questão 8 - Aplicação do filtro h_5

A imagem filtrada por h_5 é mostrada na *figura 25*.

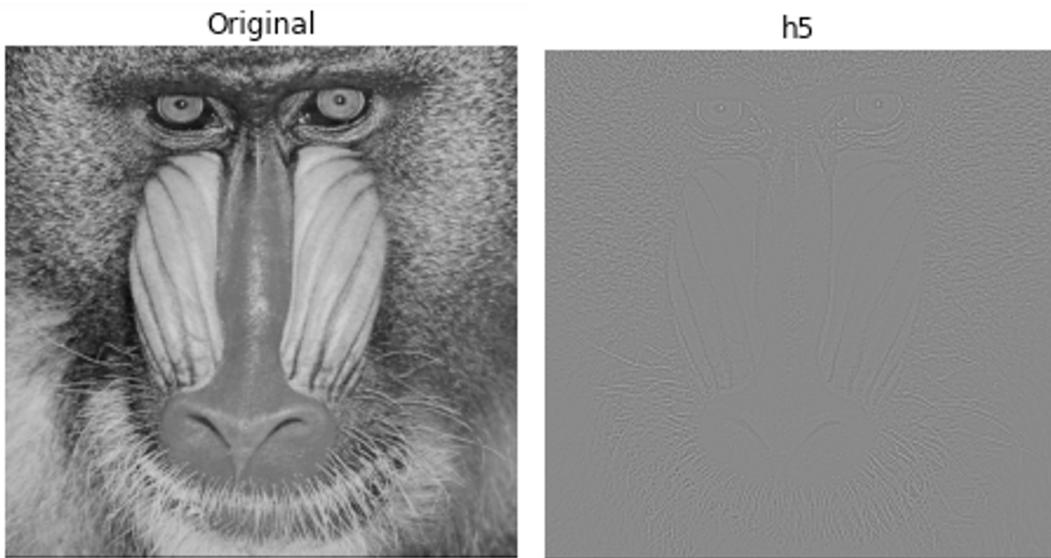


Figura 25 - Questão 8 - Aplicação do filtro h5

Por se tratar de um filtro passa altas que realça pontos de maior variação de intensidade em relação aos seus vizinhos, a imagem resultante mostra os contornos e pontos mais ruidosos da imagem. Apesar do baixo contraste, é possível visualizar o destaque dado a estes pontos, muito parecido com o resultado do filtro h1.

1.8.5 Aplicação do filtro h6

O filtro h6 é um passa baixas que faz a filtragem calculando a média dos pontos ao redor do ponto central do filtro, pois ele soma o valor de todos os pontos cobertos pela máscara e divide pela quantidade de pontos da máscara, caracterizando a média das intensidades dos pixels.

Essa característica resulta na suavização ou um certo borramento da imagem. Como se trata de uma máscara de dimensões 3, 3, essa suavização é mais leve pois leva em conta somente os pontos adjacentes ao pixel do centro do filtro.

O filtro h6 é mostrado na *figura 26*.

$$h_6 = \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figura 26 - Questão 8 - Filtro de média h6, passa baixas

A imagem resultante da aplicação do filtro h6 é vista na *figura 27*.

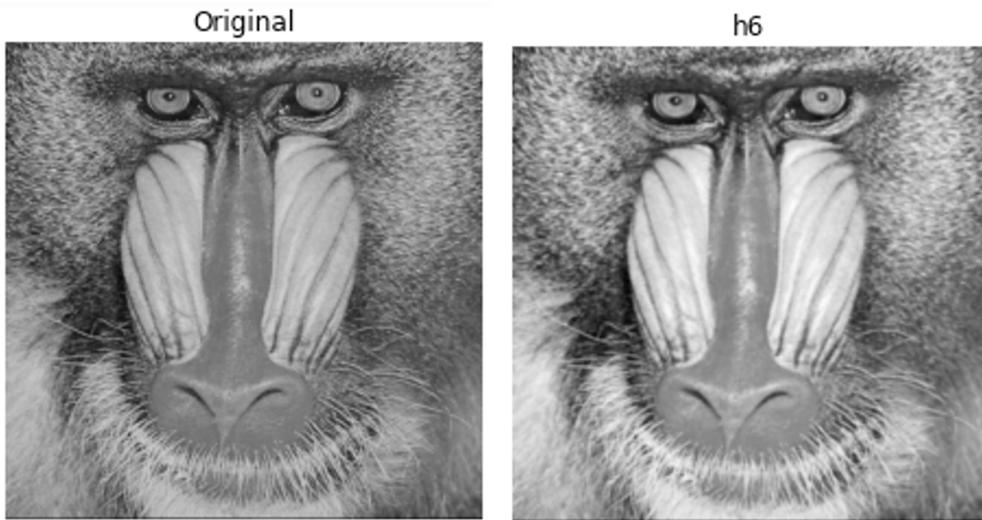


Figura 27 - Questão 8 - Aplicação do filtro h6

Como no caso do filtro h2, a aplicação do filtro h6 trouxe uma sutil suavização da imagem original onde se percebe um leve borramento geral da imagem, mas sem grande perda de nitidez da imagem. Uma possibilidade para se aumentar essa suavização seria utilizar um filtro de maiores dimensões.

1.8.6 Aplicação dos filtros h7 e h8

Os filtros h7 e h8 são mostrados na *figura 28*.

Se trata de um filtros passa altas que fazem o realce de bordas inclinadas e cantos de elementos da imagem.

$$h_7 = \begin{array}{|c|c|c|} \hline -1 & -1 & 2 \\ \hline -1 & 2 & -1 \\ \hline 2 & -1 & -1 \\ \hline \end{array} \quad h_8 = \begin{array}{|c|c|c|} \hline 2 & -1 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & -1 & 2 \\ \hline \end{array}$$

Figura 28 - Questão 8 - Filtros h7 e h8

A imagem final filtrada por h7 apresenta pequenos serrilhamentos das bordas mais inclinadas da imagem e praticamente elimina bordas que era verticais ou horizontais. Isso pode ser notado nas linhas quase verticais que há sobre o focinho no animal da imagem. Elas praticamente inexistem na imagem filtrada. O mesmo ocorre para bordas

quase horizontais sobre os olhos do animal na imagem original. Elas também são muito atenuadas na imagem filtrada. O realce máximo deste filtro ocorreria em bordas a ângulos de 45°.

Já o resultado do filtro h8 é muito semelhante, porém ele tem um realce maior em bordas com inclinação de 135°.

As imagens originais e filtradas por h7 e h8 são vistas na *figura 29*.

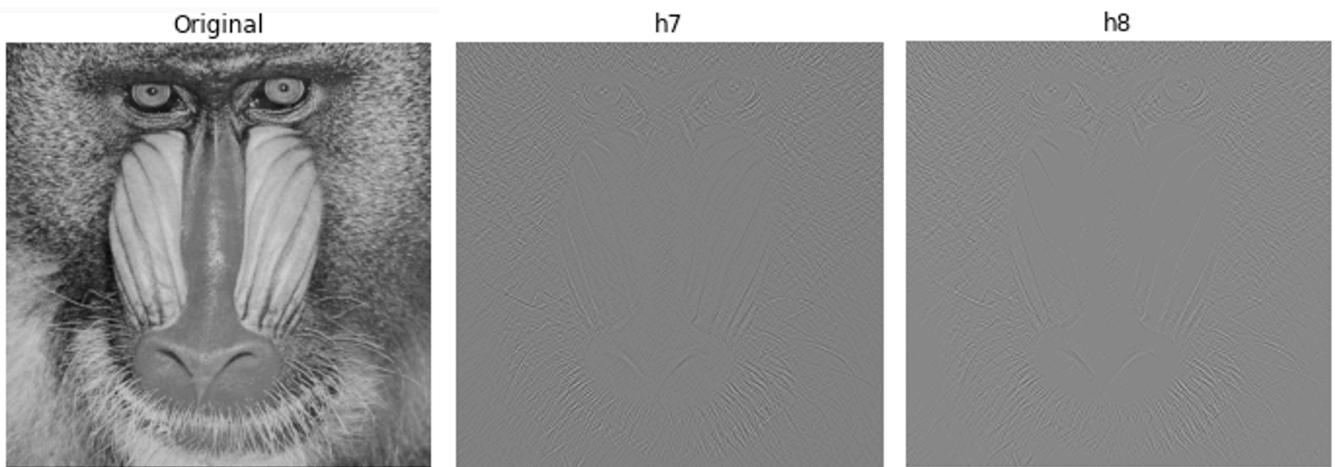


Figura 29 - Questão 8 - Aplicação dos filtros h7 e h8

1.8.7 Aplicação do filtro h9

O filtro h9 tem dimensões (9, 9) com todos os valores da diagonal principal iguais a 1 e o restante são zeros. Ou seja, é uma matriz identidade 9 por 9.

Este filtro resultou num borramento bem acentuado da imagem original que não parece com uma suavização. A perda de nitidez parece ser devida a um aparente deslocamento de parte dos pixels fazendo com que a imagem parecesse fora de foco.

O filtro h9 é mostrado na *figura 30*.

$$h_9 = \frac{1}{9} \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Figura 30 - Questão 9 - Filtro h9 - Matriz identidade

A figura 31 mostra a imagem resultante do filtro h9.

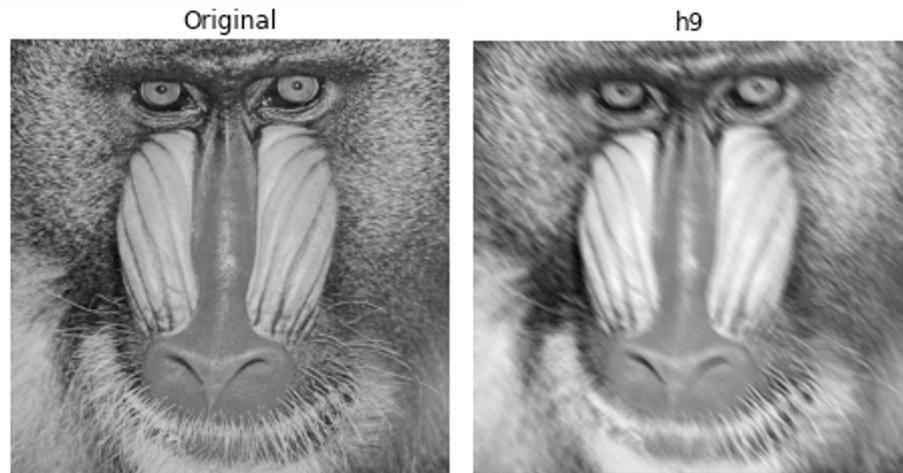


Figura 31 - Questão 8 - Resultado do filtro h9

1.8.8 Aplicação do filtro h10

O filtro h10 é apresentado na figura 32.

$$h_{10} = \frac{1}{8} \begin{array}{|c|c|c|c|c|} \hline -1 & -1 & -1 & -1 & -1 \\ \hline -1 & 2 & 2 & 2 & -1 \\ \hline -1 & 2 & 8 & 2 & -1 \\ \hline -1 & 2 & 2 & 2 & -1 \\ \hline -1 & -1 & -1 & -1 & -1 \\ \hline \end{array}$$

Figura 32 - Questão 8 - Filtro h10

O filtro h10 aplicado à imagem original teve um efeito de redução do contraste da imagem. Portanto foi um efeito de filtro passa baixas que parece ter reduzido a profundidade de níveis de cinza da imagem.

A imagem final é vista na figura 33.

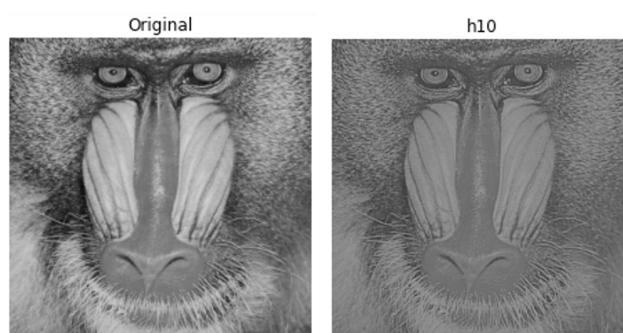


Figura 33 - Questão 8 - Resultado do filtro h10

1.8.9 Aplicação do filtro h11

O filtro h11 se trata de um passa altas que detecta variação na intensidade de pixels na porção inferior direita do filtro, realçando bordas que tenham cantos nesta posição.

O filtro h11 tem dimensões 3x3 e é mostrado na *figura 34*.

$$h_{11} = \begin{array}{|c|c|c|} \hline -1 & -1 & 0 \\ \hline -1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline \end{array}$$

Figura 34 - Questão 8 - Filtro h11

O resultado foi uma imagem com realce das bordas de forma que os tons mais escuros estão nas partes de inclinação próxima a 45º e na parte inferior direita da borda, formando um sombreado pelo canto inferior direito da imagem.

A imagem resultante é mostrada na *figura 35*.

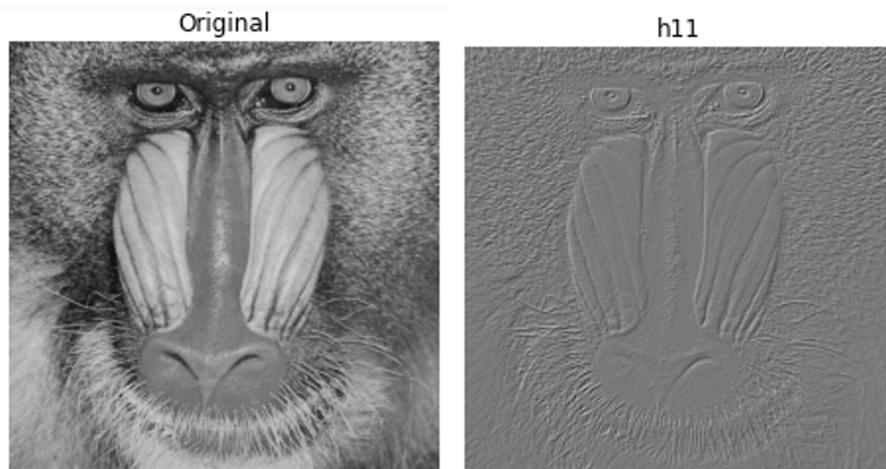


Figura 35 - Questão 8 - Resultado do filtro h11

Conclusão: As atividades propostas foram de grande valor para colocar em prática conceitos e operações básicas de processamento digital de imagens e a tentativa de desenvolver as soluções sem o uso de bibliotecas especializadas foi um desafio adicional. Principalmente na questão 8, os resultados dos filtros ainda me demandam uma necessidade de aprofundamento no entendimento dos mesmos. Isso me faz ter a percepção de que minha implementação da convolução pode ter alguns problemas ou pontos a serem melhorados para que os resultados sejam mais nítidos, principalmente nos casos de filtros passa alta, onde o contraste ficou muito baixo, dificultando a visualização dos efeitos dos filtros.