



Curso: Ciência da Computação / Engenharia da Computação

Disciplina: Computação Gráfica

Professor: Thiago Raposo Milhomem de Carvalho

Alunos:

- Mariana de Freitas Cruz - 1712082002
- Rafael da Rocha Henrard - 1812022028
- Victor Ribeiro da Silva Dantas - 171230034
- Wili Ragner Gasparetto - 2012082060

**ATIVIDADE DE LABORATÓRIO nº 01**

Data da atividade	Prazo para envio
02/04/2022	09/04/2022

Este trabalho extra-classe – a ser realizado em MATLAB / Octave – poderá ser feito em grupos de até 5 alunos, e sua nota fará parte da pontuação relativa aos trabalhos. Deverá ser elaborado um breve relatório (postar em arquivo PDF) abordando os seguintes itens:

- breve introdução teórica relativa aos assuntos da atividade e do problema a ser resolvido (descrição dos conceitos e parâmetros estudados na atividade);
- a explicação do experimento e procedimentos realizados;
- os resultados (figuras, gráficos, etc.), explicações e comentários correspondentes;
- conclusões gerais;
- no final do relatório: os comandos utilizados na atividade e, quando for o caso, os textos dos códigos fonte dos algoritmos '.m' (copie o texto do(s) script(s) e cole-o(s) aqui).

**ATENÇÃO:**

- Os comentários, interpretação dos resultados e o entendimento do que se observa são o principal fator considerado na atividade de laboratório. É o que dá sentido ao seu relatório. Relatórios sem comentários ou somente com algoritmos e/ou figuras serão desconsiderados.
- A organização do relatório e dos algoritmos escritos também é considerada na avaliação. Algoritmos desorganizados – ou escritos de forma que só possam ser compreendidos por quem os escreveu – de nada servem no ambiente acadêmico e devem ser evitados.
- Não há problemas com o diálogo e cooperação entre colegas, mas cópias não serão aceitas.

Em caso de cópias, a todos os envolvidos será aplicado o previsto no Código de Ética Discente do IESB. Seja honesto, faça seu próprio trabalho.

- Em caso de trabalhos feitos em grupo, somente um dos membros deve submetê-lo ao professor, sendo todos os membros do grupo igualmente responsáveis pelo relatório enviado, devendo certificar-se da realização da submissão e da correta identificação (nome/matricula) no relatório.

## ATIVIDADE Nº 1

### Parte 1 – Algoritmo simples para alteração de saturação de uma imagem

Nesta parte da atividade, deve-se escrever um algoritmo em Matlab / Octave que realize o seguinte procedimento: ao receber uma imagem (em cores, representada no espaço RGB) como entrada, realizar, sucessivamente alterações em sua saturação, mostrando-a para o usuário na tela a cada alteração.

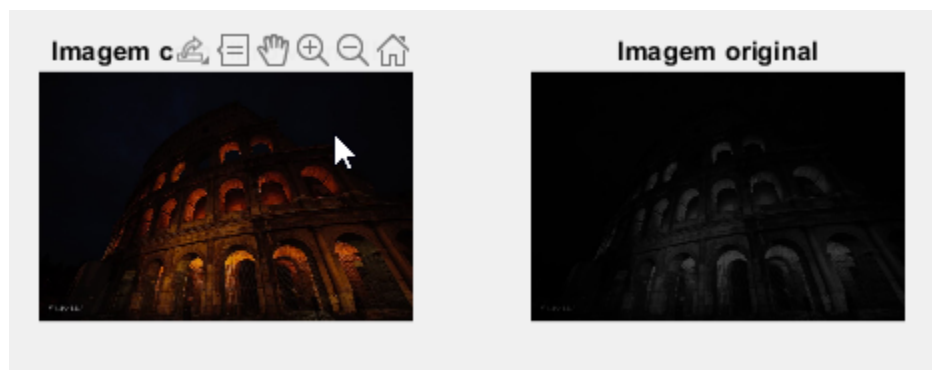
Assim, seu script deve receber um único parâmetro de entrada (a imagem, escolhida pelo usuário, a ser sucessivamente alterada e mostrada na tela). Não há necessidade de nenhum parâmetro de saída para este script.

As alterações sobre a saturação da imagem devem ser feitas em um laço de repetição. Inicialmente, deve-se mostrar a imagem com o mínimo de saturação (componente de saturação anulada em todos os pixels da imagem), sendo a saturação incrementada (de 10% em 10%, por exemplo), até seu valor máximo. As demais características da imagem (tonalidade e intensidade) devem permanecer inalteradas ao longo do procedimento.

Lembre-se, em MATLAB, a conversão do sistema RGB para o HSI, e vice-versa, pode ser realizada com os comandos `rgb2hsv` e `hsv2rgb`, respectivamente.

Aplique o procedimento a imagens “naturais” (fotografias contendo paisagens ou pessoas) e “artificiais” (como desenhos etc.) e comente os resultados acerca de suas componentes de cores. Insira, em seu relatório, exemplos de imagens resultantes que você obteve.

Aproveite a atividade para sentir-se mais à vontade para manipular imagens em Matlab / Octave em seus aspectos básicos.





```
function img_original = filtra_cor(path_img_original, cor)

img_original = imread(path_img_original);

img_original = double(img_original)/255;

tam = size(img_original);
[lin, col] = size(img_original);
fprintf('Linhas: %d | Colunas %d\n', lin, col);

if length(tam) ~= 3
    display('A variavel de entrada não é uma imagem no espaço RGB');
end

cor_escolhida = tolower(cor)

if cor_escolhida == 'r'
    display('Filtro vermelho escolhido!')

img_filtrada = img_original;
```

```
img_filtrada(:, :, 2) = 0;
img_filtrada(:, :, 3) = 0;

img_filtrada = rgb2gray(img_filtrada);

figure;
subplot(1, 2, 1); imshow(img_original);
title('Imagem com filtro vermelho');
subplot(1, 2, 2); imshow(img_filtrada);
title('Imagem original');

elseif cor_escolhida == 'g'
    display('Filtro verde escolhido!')
    img_filtrada = img_original;

    img_filtrada(:, :, 1) = 0;
    img_filtrada(:, :, 3) = 0;

    img_filtrada = rgb2gray(img_filtrada);

    figure;
    subplot(1, 2, 1); imshow(img_original);
    title('Imagem com filtro verde');
    subplot(1, 2, 2); imshow(img_filtrada);
    title('Imagem original');

elseif cor_escolhida == 'b'
    display('Filtro azul escolhido!')
    img_filtrada = img_original;

    img_filtrada(:, :, 1) = 0;
    img_filtrada(:, :, 2) = 0;

    img_filtrada = rgb2gray(img_filtrada);

    figure;
    subplot(1, 2, 1); imshow(img_original);
    title('Imagem com filtro azul');
    subplot(1, 2, 2); imshow(img_filtrada);
    title('Imagem original');

else
    display('Opção inválida! \n Escolha R, G ou B!')
```

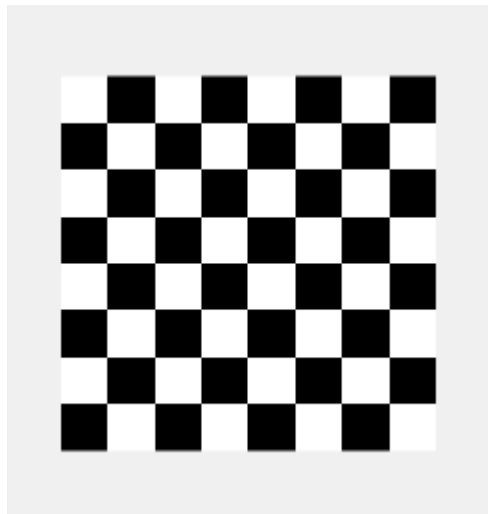
end

## Parte 2 – Tabuleiro de xadrez

Nesta parte da atividade, você deverá escrever um algoritmo simples em Matlab / Octave para criar uma imagem de tabuleiro de xadrez.

A imagem deve ser binária (somente pixels pretos e pixels brancos) e cada casa do tabuleiro deve conter 20x20 pixels. Lembre-se que o tabuleiro de xadrez tem 8x8 casas, alternando entre casas pretas e casas brancas nas linhas e colunas do tabuleiro.

Desta maneira, como se trata de um procedimento fixo, não é necessário que seu script receba nenhum parâmetro de entrada, nem que forneça nenhum parâmetro de saída. Insira em seu relatório – além do código fonte do algoritmo escrito e explicação de sua ideia – a imagem resultante que você obteve.



```
function xadrez = tabuleiro_xadrez()
    tabuleiro = [ 1 0 1 0 1 0 1 0;
                  0 1 0 1 0 1 0 1;
                  1 0 1 0 1 0 1 0;
                  0 1 0 1 0 1 0 1;
                  1 0 1 0 1 0 1 0;
                  0 1 0 1 0 1 0 1;
                  1 0 1 0 1 0 1 0;
                  0 1 0 1 0 1 0 1];

    imshow(tabuleiro)
end
```

### Parte 3 – Algoritmo simples de modificação de contraste

Nesta parte da atividade, você deverá escrever um script em Matlab / Octave para modificar o contraste de uma imagem, em escala de cinza, recebida como entrada.

Ao receber a imagem como parâmetro de entrada (matriz  $M \times N$ ), seu algoritmo deve determinar, de modo automático, se a imagem apresenta, de modo geral, brilho em falta ou em excesso (isto é, se a imagem recebida está muito escura ou muito clara). Essa decisão pode ser feita com critério à sua escolha (por exemplo, pela média do brilho dos pixels; utilizando o histograma da imagem etc.).

Caso seja uma imagem muito escura, deve ser aplicada a correção *gamma* para deixá-la mais clara e, caso seja muito clara, deve ser aplicada a correção *gamma* para escurecê-la. O valor do parâmetro  $\gamma$  deve ser escolhido adequadamente ( $\gamma > 1$  ou  $\gamma < 1$ ), a depender da característica da imagem recebida como entrada (clara ou escura). Justifique os valores escolhidos e seu critério de escolha.



```
function img_original = gama(path_img_original)

img_original = imread(path_img_original);

img_original = double(img_original)/255;

tam = size(img_original);
[lin, col] = size(img_original);
fprintf('Linhas: %d | Colunas %d\n', lin, col);

if length(tam) ~= 3
    display('A variavel de entrada não é uma imagem no espaço RGB');
end
```

```

img_cinza = rgb2gray(img_original);

mean_intensity_cinza = mean(img_cinza(:));
fprintf('Média de intensidade de pixels: %d\n', mean_intensity_cinza);

if mean_intensity_cinza < 0.5

    img_claro = img_cinza.^(0.6);

    display('Imagem escura, vamos clarear!')
    mean_intensity_claro = mean(img_claro(:));
    fprintf('Média de intensidade de pixels enclarear: %d\n', mean_intensity_claro);

    figure;
    subplot(1, 2, 1); imshow(img_cinza);
    title('Imagem binária');
    subplot(1, 2, 2); imshow(img_claro);
    title('Imagem enclarecida');

else

    img_escura = img_cinza.^(1.5);

    display('Imagem clara, vamos escurecer')
    mean_intensity_escuro = mean(img_escura(:));
    fprintf('Média de intensidade de pixels após escurecer: %d\n', mean_intensity_escuro);

    figure;
    subplot(1, 2, 1); imshow(img_cinza);
    title('Imagem binária');
    subplot(1, 2, 2); imshow(img_escura);
    title('Imagem escurecida');
end



```

Assim, o único parâmetro de entrada do algoritmo deve ser:

- Imagem em escala de cinza (matriz de pixels) para o ajuste de contraste;

O único parâmetro de saída deve ser:

- Matriz de pixels da imagem resultante da aplicação da correção *gamma*.

Teste seu algoritmo com imagens de fotografias subexpostas e superexpostas, por exemplo, comentando os resultados e os efeitos da escolha de  .

Lembre-se das recomendações para os relatórios e aproveite a atividade para compreender melhor o uso da técnica, seus efeitos e suas limitações.