Universidade Tecnológica Federal do Paraná - UTFPR
Campus Curitiba
Departamento Acadêmico de Eletrônica - DAELN
Curso de Engenharia Eletrônica
Prof. Gustavo B. Borba

Notas de aula e prática Octave #04

Arquivos necessários

1. 5.1.09.tiff [http://sipi.usc.edu/database/database.php]
2. xray01.tif [adaptada de https://wexnermedical.osu.edu/blog/facts-about-dense-breast-tissue]
3. radio.tif [http://www.ogemarques.com/wp-content/uploads/2014/11/Tutorial_Images.zip]
4. vpfig.png [adaptada de Volnei A. Pedroni, Eletrônica Digital Moderna e VHDL, 2020, Fig. 4.6]
5. 42049_20-200.png [adaptada de https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/images/gray/42049.html]
6. gDSC04422m16.png

4.1) Funções de transformação dos níveis de cinza (identidade e negativo)

O código a seguir cria uma *função de transformação* (ajuste) do tipo *identidade*.

Para o warm up deste exercício, entenda o código e execute-o.

```
clear all; close all; clc

g = checkerboard(1, [1 1]);
figure; imshow(g)

gu = im2uint8(g);
gu2 = uint8(g.*255); %igual im2uint8(g)

%Função de transformação
y = uint8(255:-1:0);
gu1 = double(gu)+1;

gt = y(gu1);
gtu = uint8(gt);
figure; imshow(gtu);
```

O *negativo* de uma imagem é equivalente ao negativo fotográfico. É obtido fazendo-se $n = P - p$ para todos os pixels da imagem, onde P é o máximo valor de nível de cinza possível (255 para imagens uint8) e p é o valor do pixel.

O negativo pode ser utilizado para facilitar a visualização de objetos em determinados tipos de imagens. Para o caso de imagens de exames de mamografia, por exemplo, há especialistas que preferem visualizar o negativo da imagem original. O argumento é que, na imagem original, os detalhes são mais difíceis de visualizar pois ficam envoltos por grandes áreas escuras.

Crie uma função de transformação para obter o negativo da imagem *xray01.png*. Só pra saber: o Octave tem a função `imcomplement [https://octave.sourceforge.io/image/function/imcomplement.html]`, mas nesse exercício não é pra usá-la. Use o código do exemplo como template. Resposta disponível em a04_01.m.

4.2) Funções de transformação dos níveis de cinza (negativo usando a função intlut)

Outra maneira de implementar funções de transformação no Octave é utilizando-se a função `intlut` [https://octave.sourceforge.io/image/function/intlut.html], que implementa uma *lookup table* (LUT). O código a seguir cria uma *função de transformação* (ajuste) do tipo *identidade* usando a função `intlut`.

```
clear all; close all; clc

g = imread('5.1.09.tiff');
figure; imshow(g)

%Função de transformação
y = uint8(0:255);
figure;
plot(y); xlim([0 255]); ylim([0 255]);
%Usando intlut
gt = intlut(g,y);
figure, imshow(gt)

%Se verifica == 0, imagens iguais
verifica = double(g) - double(gt);
verifica = sum(verifica(:));
```

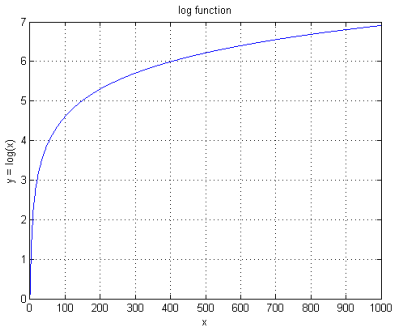
Crie uma função de transformação para obter o negativo da imagem *5.1.09.tiff*, utilizando a função `intlut`. Só pra saber: o Octave tem a função `imcomplement`, mas nesse exercício não é pra usá-la. Use o código do exemplo como template, no mesmo espírito do exercício anterior. Resposta disponível em a04_02.m.

4.3) Funções de transformação dos níveis de cinza ($\log_e$)

```
%Funcao log

%Logaritmo neperiano (base e)
y = log(1:1000);

%Display
figure
plot(y)
grid on
title('log function')
xlabel('x')
ylabel('y = log(x)')
```



A função de transformação do tipo logarítmica é $y = c \cdot log(1+x)$. x é o pixel da imagem e c uma constante geralmente igual a 1. Também pode ser menor que 1, para manter a saída dentro da faixa dinâmica (255 para imagens `uint8`). O *1+* serve para evitar *o log(0) = indeterminado* ($\rightarrow -\infty$).

Lembrando que, geralmente, *log* é o log_{10} (a base pode ser suprimida), log_e ou *ln* é o *logarítmo natural* ou *logarítmo neperiano* [http://en.wikipedia.org/wiki/Logarithm, Tópico Particular bases]. Atenção, pois no Octave: função `log10` é o *log* e função `log` é o log_e.

Observando a curva da função logarítmica plotada anteriormente, conclui-se que ela mapeia os níveis de cinza da entrada para a saída da seguinte forma:

- Uma faixa de poucos valores baixos (faixa estreita) de níveis de cinza na entrada → uma faixa de muitos valores (faixa ampla) de níveis de cinza na saída
- A faixa restante de valores mais altos de níveis de cinza na entrada → uma faixa de poucos (faixa estreita) valores de níveis de cinza maiores na saída.

O efeito disto é a expanção dos pixels escuros, para deixá-los mais claros e visíveis, e por consequência a compressão dos pixels claros, para que todos os níveis de cinza caibam na faixa dinâmica da imagem.

A função *gamma* tem a mesma finalidade da *log*, é mais versátil e portanto mais utilizada. A função *log* é útil para comprimir os valores dos pixels de imagens com variações muito grandes nos valores dos pixels. A aplicação típica está na visualização da transformada de Fourier de uma imagem, que pode assumir valores de 0 até 10^6, por exemplo.

Aplicar o *log* (log_e) na imagem *radio.tif*. Não use auto-contraste (mat2gray). Resposta disponível em a04_03.m.

4.4) Auto-contraste (também chamado de normalização) (na unha, usando mat2gray e imadjust)

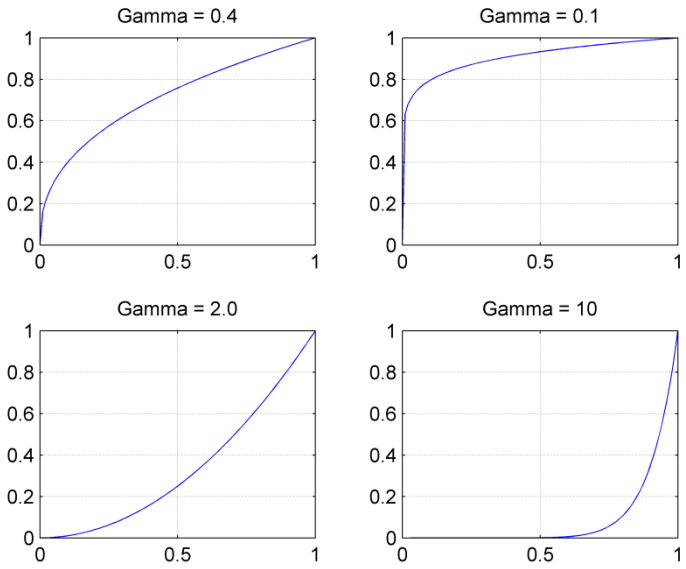
Outros nomes utilizados para esta operação são *normalização*, *contrast stretching* ou *histogram stretching*. Apenas 'normalização' não deixa claro se está realizando-se a operação *min(img)* $\rightarrow 0$, *max(img)*$\rightarrow 1$, ou apenas uma divisão por 255 (para imagens de entrada `uint8`).

Fazer o auto-contraste da imagem *42049_20-200.png* de três formas diferentes: na unha, usando a função `mat2gray` e usando a função `imadjust` [https://octave.sourceforge.io/image/function/imadjust.html]. Os resultadoz devem ser visualmente iguais. Resposta disponível em a04_04.m.

4.5) Funções de transformação dos níveis de cinza (gamma usando função imadjust)

```
%Funcao Gamma
x = 0:0.01:1;
y1 = x.^0.4;
y2 = x.^0.1;
y3 = x.^2.0;
y4 = x.^10;

%Display
figure
subplot(2,2,1), plot(x,y1)
grid on
title('Gamma = 0.4')
subplot(2,2,2), plot(x,y2)
grid on
title('Gamma = 0.1')
subplot(2,2,3), plot(x,y3)
grid on
title('Gamma = 2.0')
subplot(2,2,4), plot(x,y4)
grid on
title('Gamma = 10')
```



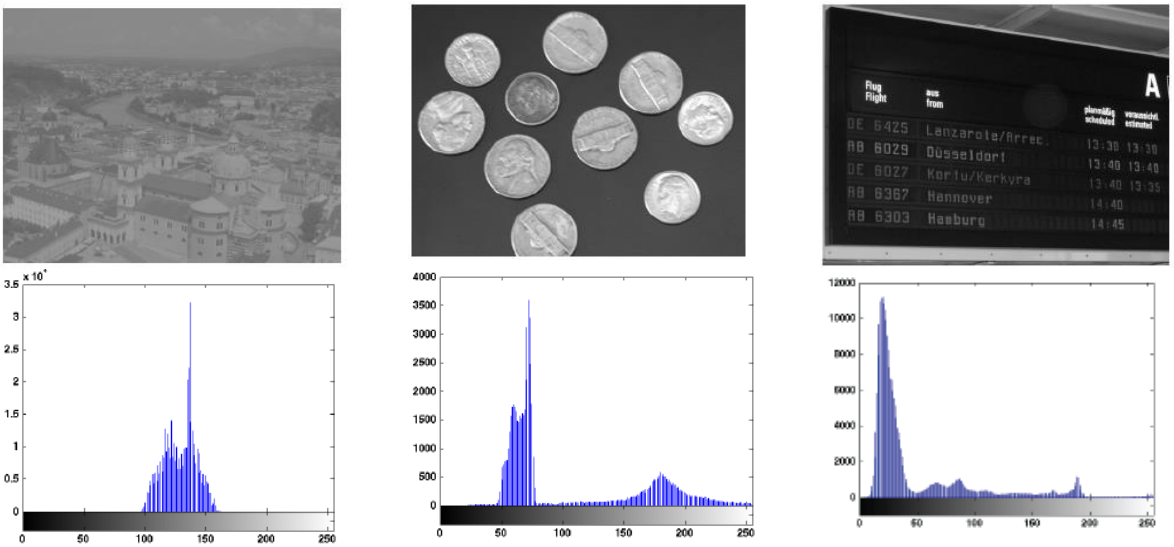
A função de transformação do tipo *gamma* é $y = c \cdot x^{\gamma}$. x é o pixel da imagem e c uma constante geralmente igual a 1. Também pode ser menor que 1, para manter a saída dentro da faixa dinâmica (255 para imagens `uint8`). γ é a constante que determina o formato da curva.
Observando a curva da função gamma plotada anteriormente, conclui-se que ela mapeia os níveis de cinza da entrada para a saída como descrito a seguir.

- Para $\gamma < 1$: A imagem 'fica mais clara'.
- Para $\gamma > 1$: A imagem 'fica mais escura'.
- Para $\gamma = 1$: Função identidade, isto é, a saída é igual à entrada (desde que c=1)

Usar a função `imadjust` [http://www.mathworks.com/help/images/ref/imadjust.html] para aplicar a função de transformação *gamma* na imagem *radio.tif* . Mostrar cada imagem em uma figure: original e as processadas com γ = 0,4, γ = 0,1 e γ = 2,0. Resposta disponível em a04_05.m.

4.6) Histograma usando a função imhist

Exemplos de imagens e respectivos histogramas [[OM], Tópico 9.3, Figura 9.2]:



Obter os histogramas das imagens *gDSC04422m16.png*, *42049_20-200.png* e *vpfig.png*, utilizando a função `imhist` do Octave. Resposta disponível em a04_06.m.

Referências

[OM] Oge Marques, Practical image and video processing using MATLAB, Wiley, 2011.