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

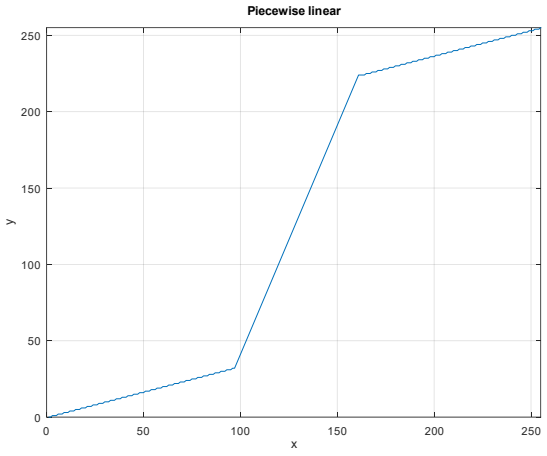
Atividade #04
Vale nota, individual, observar prazo e instruções de entrega no moodle

Arquivos necessários

1. vpfig.png [adaptada de Volnei A. Pedroni, Eletrônica Digital Moderna e VHDL, 2020, Fig. 4.6]
2. 42049_20-200.png [adaptada de https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/images/gray/42049.html]
3. gDSC04422m16.png

4.1) Funções de transformação dos níveis de cinza (piecewise-linear usando intlut)

```
%Piecewise Linear
%Aloca uint8
%para depopis usar funcao intlut (y1 é a
LUT)
y1 = uint8(zeros([1 256]));
%Equação da reta inferior y = (1/3)*x
y1(1:97) = (1/3)*(0:96);
%Equação da reta intermediária y = 3*x -256
y1(98:161) = 3*(97:160) - 256;
%Equação da reta superior y = (1/3)*x + 170
y1(162:256) = (1/3)*(161:255) + 170;
%Display
figure, plot(y1)
xlim([0 255]), ylim([0 255])
grid on
title('Piecewise linear')
xlabel('x'), ylabel('y')
```



O contrast streching (alongamento do contraste) pode ser realizado utilizando-se funções pecewise-linear (linear por partes). Nos plot anteriores pode ser observada uma função piecewise-linear para o contrast streching.

A função para contrast stretching usando o recurso piecewise-linear mostrada no código mapeia:

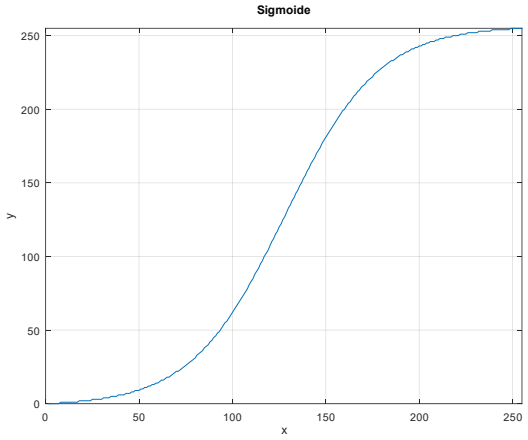
- Reta inferior: os pixels escuros da faixa [0 96] em pixels mais escuros ainda na faixa [0 32], realizando assim uma compressão dos pixels escuros.
- Reta superior: os pixels claros da faixa [161 255] em pixels mais claros ainda na faixa [224 255], realizando assim uma compressão dos pixels claros.
- Reta intermediária: os pixels de nível médio da faixa [97 160] em pixel de nível médio na faixa [35 224], realizando assim um expansão dos pixels de nível de cinza médios.

Aplique a função piecewise-linear para contrast streching especificadas anteriormente (código exemplo) na imagem *vpfig.png*. Use a função `intlut` do Octave. Mostre a imagem original e a processada.

Nome do .m: atv04_01.m

4.2) Funções de transformação dos níveis de cinza (sigmoide usando intlut)

```
%Sigmoid
%Aloca uint8
%para depois usar funcao intlut (y1 é a LUT)
%Equação da sigmoide
slope = 0.05;
inflec = 127;
x = 0:1:255;
y1 = 1./(1 + exp(-slope*(x - inflec)));
y1n = mat2gray(y1);
y1n = uint8(y1n.*255);
%Display
figure, plot(y1n)
xlim([0 255]), ylim([0 255])
grid on
title('Sigmoide')
xlabel('x'), ylabel('y')
```



A função sigmoide também pode ser utilizada para o contrast streching.

Aplique a função sigmoide para contrast streching especificada anteriormente (código exemplo) na imagem *vpfig.png*. Use a função `intlut` do Octave. Teste e mostre os resultados com diferentes valores para a variável 'slope', que determina a inclinação da sigmoide ('slope' maior: sigmoide mais 'ingreme'). Explique, no próprio arquivo .m, para que foi utilizada a função `mat2gray` no código.

Nome do .m: atv04_02.m

4.3) Cálculo do histograma na unha

Computar e plotar o histograma de uma imagem sem usar a função `imhist` ou `hist` ou .... Pode usar laços de repetição à vontade, mas tem que ser na unha. Mostrar uma figure com a imagem *42049_20-200.png*, outra com o histograma que vc gerou e outra com o histograma do `imhist`, com o objetivo de comparar os dois.

Nome do .m: atv04_03.m

4.4) Equalização do histograma usando a função histeq

Fazer a equalização do histograma da imagem *gDSC04422m16.png* usando a função `histeq(I,256)` do Octave.

Nome do .m: atv04_04.m

4.5) Equalização do histograma na unha

Fazer a equalização do histograma de uma imagem na unha. Pode usar a função `cumsum`. Mostrar a imagem *gDSC04422m16.png* processada com o seu programa e com o `histeq(I,256)` do Octave e o histograma de cada uma. Os passos para a equalização do histograma estão descritos abaixo.

Nome do .m: atv04_05.m

1. Obter o histograma da imagem original.
2. Normalizar este histograma [dividir por M*N (número de pixels da imagem)].
3. Obter a *cumulative distribution function* (cdf) [acumular o histograma do passo 2, isto é, *p(i) = Σ(j=0 até i) de p(j)*].
4. Transformar a cdf em *níveis de cinza arredondados* [multiplicar por 255 e transformar em `uint8`].
5. Aplicar o vetor do passo 4 como uma *função de transformação* sobre a imagem original, usando a função `intlut` do Octave.