Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC 326 - Processamento de Imagem
Professor: Guillermo Camara Chavez

Lista 02 - PDI

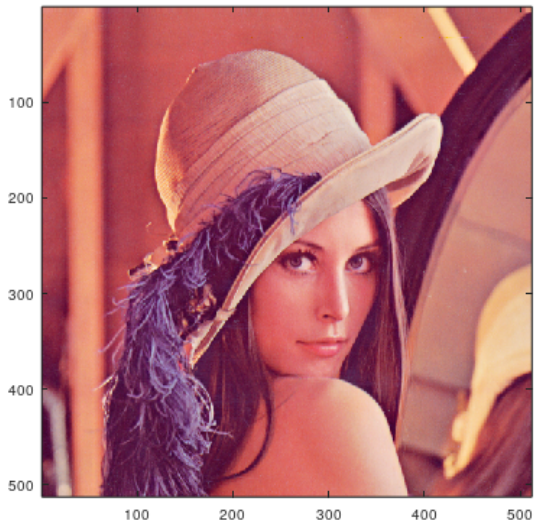
Aluno: Felipe Augusto Vasconcelos e Silva
Matrícula: 16.2.4358

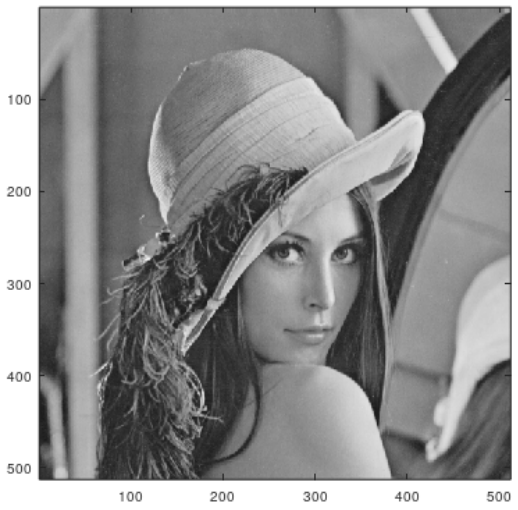
Questão 01) Converta uma imagem colorida para tons de cinza (luminância). Uma imagem em tons de cinza pode ser obtida a partir de uma imagem colorida aplicando-se a seguinte fórmula para cada um dos pixels da imagem original: $L = 0.299 * R + 0.587 * G + 0.114 * B$, onde R, G e B são as componentes de cor do pixel original. Ao criar uma imagem a ser exibida em tons de cinza, para cada pixel pi , faça: Ri = Gi = Bi = Li ; O seu programa deve permitir que a aplicação do cálculo de luminância um número arbitrário de vezes durante sua execução. Pergunta: O que acontecerá com uma imagem em tons de cinza (Ri = Gi = Bi = Li) caso o cálculo de luminância seja aplicado repetidas vezes (e.g., recursivamente) a imagem?

```
function new_img = convert_to_gray(img_rgb)
  img_rgb = double(img_rgb);

  R = img_rgb(:,:,1) * 0.299;
  G = img_rgb(:,:,2) * 0.587;
  B = img_rgb(:,:,3) * 0.114;
  L = R + G + B;

  gray_img = cat(3, L, L, L);
  new_img = uint8(gray_img);
```



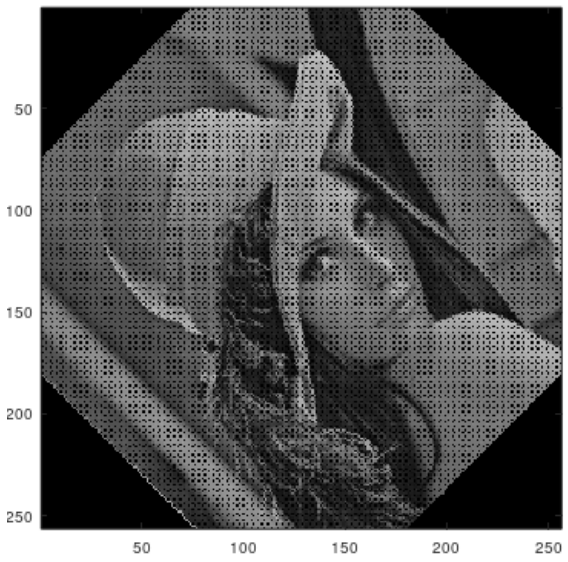


Questão 02) Implementar as funções de transformação geométrica (rotação, escalamento, translação, cisalhamento) em imagens.

Rotação:

```
function nimg = rotacao(img, ang)
  [lin, col, ~] = size(img);
  nimg = zeros(lin, col);
  matT1 = [1 0 -lin/2 ; 0 1 -col/2 ;  0 0 1];
  matR = [sind(ang) -cosd(ang) 0; cosd(ang) sind(ang)  0 ; 0 0 1];
  matT2 = [1 0 lin/2 ; 0 1 col/2 ;  0 0 1];
  mat = matT2 * matR * matT1;

  for i = 1 : lin
    for j = 1 : col
       ncoord = mat * [i; j; 1];
       ni = floor(ncoord(1));
       nj = floor(ncoord(2));
       if  ni > 0 && ni <= lin && nj > 0 && nj <=col
         nimg(ni, nj) = img(i, j);
       endif
    endfor
  endfor
  nimg = uint8(nimg);
```

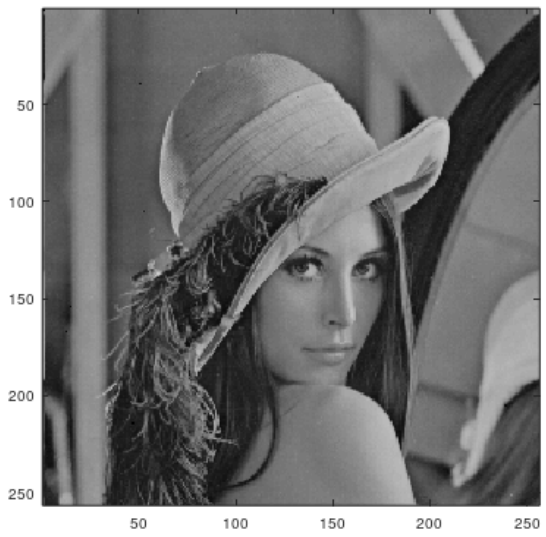


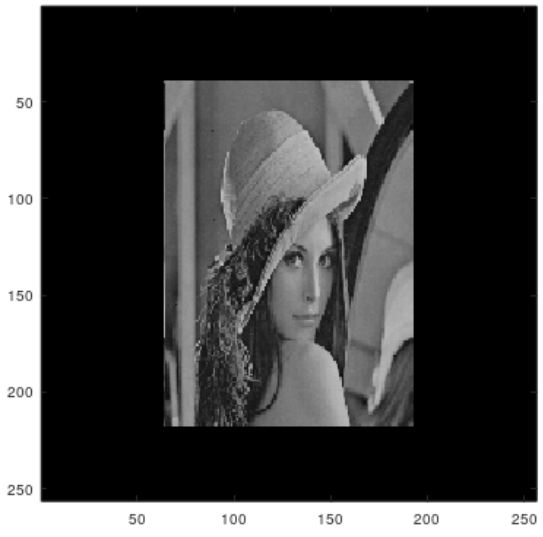
Escalamento

```
function nimg = ecala(img, sx, sy)

  [lin, col, ~] = size(img); %
  nimg = zeros(lin, col); %
  matT1 = [1 0 -lin/2 ; 0 1 -col/2 ;  0 0 1]; %
  matR = [sx 0 0; 0 sy 0 ; 0 0 1]; % Muda
  matT2 = [1 0 lin/2 ; 0 1 col/2 ;  0 0 1]; %
  mat = matT2 * matR * matT1;

  for i = 1 : lin
    for j = 1 : col
       ncoord = mat * [i; j; 1];
       ni = floor(ncoord(1));
       nj = floor(ncoord(2));
       if  ni > 0 && ni <= lin && nj > 0 && nj <=col
         nimg(ni, nj) = img(i, j);
       endif
    endfor
  endfor
  nimg = uint8(nimg);
```



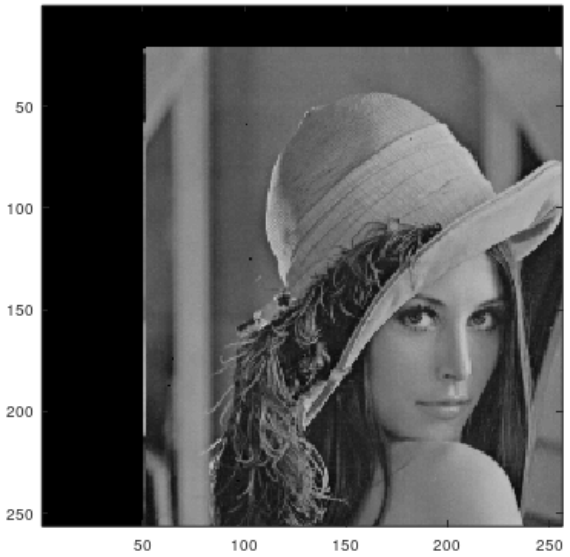


Translação:

```
function nimg = transladar(img, tx, ty)

  [lin, col, ~] = size(img); %
  nimg = zeros(lin, col); %
  matT1 = [1 0 -lin/2 ; 0 1 -col/2 ;  0 0 1]; %
  matR = [1 0 tx; 0 1 ty ; 0 0 1]; % Muda
  matT2 = [1 0 lin/2 ; 0 1 col/2 ;  0 0 1]; %
  mat = matT2 * matR * matT1;
```

```
for i = 1 : lin
  for j = 1 : col
     ncoord = mat * [i; j; 1];
     ni = floor(ncoord(1));
     nj = floor(ncoord(2));
     if  ni > 0 && ni <= lin && nj > 0 && nj <=col
       nimg(ni, nj) = img(i, j);
     endif
  endfor
endfor
nimg = uint8(nimg);
```



Cisalhamento

```
function nimg = cisalhamento(img, shear)

  [lin, col, ~] = size(img);
  nimg = zeros(lin, col);
  matT1 = [1 0 -lin/2 ; 0 1 -col/2 ;  0 0 1];
  matR = [1 shear 0; 0 1 0 ; 0 0 1]; % <-
  matT2 = [1 0 lin/2 ; 0 1 col/2 ;  0 0 1];
  mat = matT2 * matR * matT1;

  for i = 1 : lin
    for j = 1 : col
       ncoord = mat * [i; j; 1];
       ni = floor(ncoord(1));
       nj = floor(ncoord(2));
       if  ni > 0 && ni <= lin && nj > 0 && nj <=col
         nimg(ni, nj) = uint8(img(i, j));
```

```
    endif
  endfor
endfor
nimg = uint8(nimg);
```

Questão 03) Modificar a resolução de uma imagem, reduzindo à metade (não utilizar a função imresize). Isto é, se o tamanho da imagem é de 512 × 512, a dimensão da nova imagem será de 256 × 256. Logo, duplicar o tamanho da nova imagem, de forma tal, que ela tenha novamente 512 × 512 pixels.

```
function nimg = resize(img, scale)
  [lin, col, ~] = size(img);
  width = round(lin*scale);
  height = round(col*scale);
  nimg = zeros(width, height);

  for i = 1 : width
    for j = 1 : height
      nimg(i,j) = img(round(i/scale), round(j/scale));
    endfor
  endfor

  nimg = uint8(nimg);
```

1) Imagem original
2) Imagem reduzida pela metade
3) imagem aumentada em dobro o tamanho, utilizando a imagem reduzida pela metade

Questão 04) Testar as funções rgb2gray(), rgb2ind(), im2double(), im2bw() e salvar as images com imwrite()
Imagens salvas na pasta zip

Questão 05) Dada uma imagem, primeiro espelhar a imagem na vertical. Depois, a partir da imagem espelhada, espelhar novamente a mesma, mas na horizontal. Não utilizar as funções fliplr() e flipud(). Faça uso de vetores para acessar os índices

Espelhar Horizontalmente:

```
function nimg = mirror_vertical(img)
  img = double(img);
  [lin, col, n] = size(img);
    nimg = zeros(lin, col, n);
  nimg(:,1:col,:) = img(:,col:-1:1,:);

  nimg = uint8(nimg);
```

Espelhar Horizontalmente:

```
function nimg = mirror_horizontal(img)
  img = double(img);
  [lin, col, n] = size(img);
  nimg = zeros(lin, col, n);
  nimg(1:lin,:,:) = img(lin:-1:1,:,:);
   nimg = uint8(nimg);
```