Introdução ao Processamento Digital de Imagens
Prof. Leonardo
Trabalho 1
Data de entrega: 06/03/2017

1. Desenvolva um sistema para abrir, exibir, manipular e salvar imagens RGB com 24 bits/pixel (8 bits/componente/pixel) e "grayscale" (8 bits/pixel). O sistema deve ter a seguinte funcionalidade:
   1.1. Conversão RGB-YIQ-RGB (cuidado com os limites de R, G e B!)
   1.2. Exibição de bandas individuais (R, G e B) como imagens monocromáticas ou coloridas (em tons de R, G ou B, respectivamente)
   1.3. Negativo (intensidade na saída = 255 – intensidade na entrada)
   1.4. Controle de brilho aditivo (valor do pixel resultante = valor do pixel original + c, c inteiro) (cuidado com os limites de R, G e B!)
   1.5. Controle de brilho multiplicativo (valor do pixel resultante = valor do pixel original * c, c real não negativo) (cuidado com os limites de R, G e B!)
   1.6. Limiarização aplicada sobre Y, com limiar m e duas opções: a) m escolhido pelo usuário; b) m = média de valores da banda Y;
   1.7. Filtros de Média e Mediana de ordem escolhida pelo usuário
   1.8. Filtros de detecção de bordas Sobel e Laplaciano
   1.9. Aplicar os seguintes filtros (descreva o resultado no relatório):

0	-1	0
-1	5	-1
0	-1	0

0	0	0
0	1	0
0	0	-1

*O sistema deve ser desenvolvido em MATLAB e em outra linguagem de programação de sua escolha (tal como Java, C, C++, Python).

**Para os itens 1.3 a 1.5, duas formas de aplicação devem ser testadas: em RGB (banda a banda) e na banda Y, com posterior conversão para RGB. Discuta os resultados obtidos.

***As operações de filtragem (1.7 e 1.8) no MATLAB podem utilizar funções apropriadas para essas operações - exemplo: `filter2(fspecial('average', ksize), img)`.

****Neste trabalho, a biblioteca OpenCV só poderá ser usada para carregar, exibir e salvar imagens. Todo o processamento deverá ser feito diretamente nas matrizes.

2. Reproduza, utilizando o MATLAB e outra linguagem de programação de sua escolha, o exemplo apresentado na página 3 deste documento (ImageProcessing Toolbox User'sGuide - UsingMedianFiltering). Repita, no MATLAB, para ruído 'gaussian' e 'speckle', explicando o que são estes tipos de ruído.Discuta o experimento no relatório.

Observações:

1. O trabalhopode ser feito em grupo, com até três componentes.
2. Para integralização das notas, o trabalho deve ser apresentado na data e horário marcados, juntamente com um relatório impresso, contendo pelo menos as seguintes seções: introdução (contextualização e apresentação do tema, fundamentação teórica, objetivos), materiais e métodos (descrição detalhada das atividades desenvolvidas e das ferramentas e conhecimentos utilizados) resultados, discussão (problemas e dificuldades encontradas, comentários críticos sobre os resultados) e conclusão. Cada componente do grupo deve estar familiarizado com o trabalho desenvolvido pelos demais componentes do seu grupo, e todos devem comparecer à apresentação dos trabalhos.

Image Processing Toolbox User's Guide-Using Median Filtering

Median filtering is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. The medfilt2 function implements median filtering. Median filtering is a specific case of order-statistic filtering, also known as rank filtering. For information about order-statistic filtering, see the reference page for the ordfilt2 function.

The following example compares using an averaging filter and medfilt2 to remove salt and pepper noise. This type of noise consists of random pixels' being set to black or white (the extremes of the data range). In both cases the size of the neighborhood used for filtering is 3-by-3.

1. Read in the image and display it.

```
I = imread('eight.tif');
imshow(I)
```

2. Add noise to it.

```
J = imnoise(I,'salt& pepper',0.02);
figure, imshow(J)
```

3. Filter the noisy image with an averaging filter and display the results.

```
K = filter2(fspecial('average',3),J)/255;
figure, imshow(K)
```

4. Now use a median filter to filter the noisy image and display the results. Notice that medfilt2 does a better job of removing noise, with less blurring of edges.

```
L = medfilt2(J,[3 3]);
figure, imshow(L)
```