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

Atividade #05
Vale nota, individual, observar prazo e instruções de entrega no moodle

Arquivos necessários

1. Lenna256g.png [Adaptada de https://en.wikipedia.org/wiki/Lenna]
2. b5s.40.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
3. b5s.100.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
4. salt-and-pepper1.tif [Geoff Dougherty, Digital image processing for medical applications, Cambridge University Press, 2009, Figure 8.9. Imagem disponível em www.cambridge.org/dougherty -> Resources -> Figures for activities]
5. flowervaseg.png [http://www.digitalcamerainfo.com/content/Samsung-WB150F-Digital-Camera-Review/Sample-Photos.htm]

5a) Filtros passa-baixas

5.1) Filtro da média

Obtenha uma máscara de convolução do filtro da média 3x3 (box filter) sem usar a função `fspecial` e mostre-a em um gráfico 3D utilizando a função `mesh`.

Nome do .m: atv05_01.m

5.2) Filtro da média (funções imfilter e fspecial)

Usar as funções `imfilter` e `fspecial` com o filtro da média (box filter) 5x5 e 7x7 para suavizar o ruído das imagens de RM *b5s.40.bmp* e *b5s.100.bmp.*

Nome do .m: atv05_02.m

5.3) Filtro Gaussiano

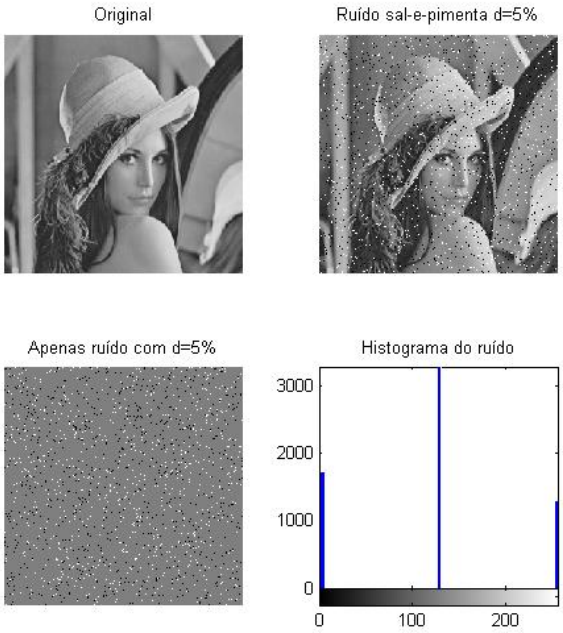
Obtenha uma máscara de convolução do filtro Gaussiano 5x5 e sigma 1 sem usar a função `fspecial` e mostre-a em um gráfico 3D utilizando a função `mesh`. Utilize essa máscara com a funções `imfilter` para suavizar o ruído das imagens de RM *b5s.40.bmp* e *b5s.100.bmp*.

Nome do .m: atv05_03.m

5b) Ruído sal-e-pimenta e filtro de estatística de ordem

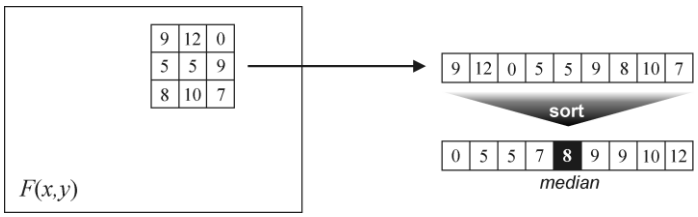
O ruído do tipo *sal-e-pimenta* (*salt-and-pepper*) é classificado como *impulsivo* (o Gaussiano é aditivo). É composto por pixels brancos (255 para imagens de 8 bits) e pixels pretos (0) distribuídos uniformemente sobre a imagem. Portanto, o nome sal e pimeta faz sentido: sal para os pixels de ruído claros e pimenta para os escuros.

```
clear all, close all
g = imread('Lenna256g.png');
%Parâmetro:densidade de pixels com ruído
gsp = imnoise(g, 'salt & pepper', 0.05);
%Apenas ruído, só pra visualizar
z = uint8(zeros(size(g))+127);
sp = imnoise(z, 'salt & pepper', 0.05);
unique(sp)%para notar que sal=255,pim=0
%Display
figure
subplot(2,2,1)
imshow(g), title('Original')
subplot(2,2,2)
imshow(gsp)
title('Ruído sal-e-pimenta d=5%')
subplot(2,2,3)
imshow(sp)
title('Apenas ruído com d=5%')
subplot(2,2,4)
imhist(sp), ylim([0 numel(sp)*0.05])
title('Histograma do ruído')
```

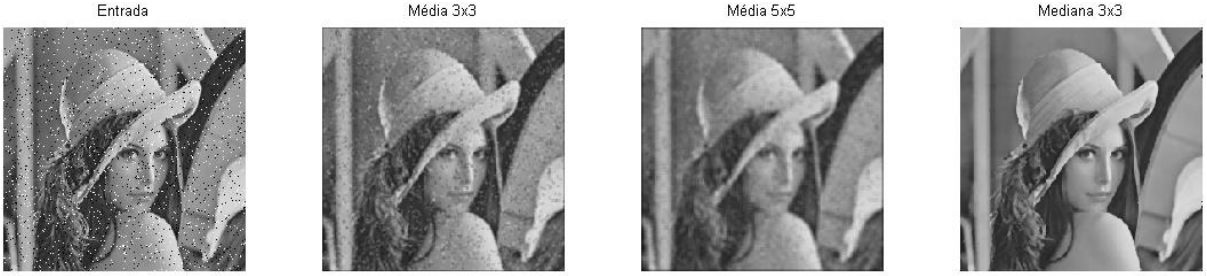


O filtro mais utilizado para a remoção do ruído sal-e-pimenta é o *filtro da mediana*. O filtro da mediana pertence a uma família denominada de filtros de *estatística de ordem*, pois para a obtenção da mediana os pixels da imagem de entrada pertencentes à janela devem estar ordenados, isto é, organizados em ordem crescente. Por isso, também são chamados de *rank filters* ou *rank-order filters*. É importante lembrar que o filtro da mediana não utiliza a operação de convolução. Por isso, o termo *janela* é mais adequado que o termo *máscara*. O termo máscara costuma ser mais utilizado no contexto da convolução

Como mostrado na figura a seguir [[OM], Tópico 10.3.4, Figura 10.8], para obter a mediana dos pixels pertencentes à janela posicionada sobre um pixel qualquer da imagem de entrada, deve-se primeiramente ordenar os pixels em um vetor. A mediana é então o elemento central do vetor. Se o vetor for de comprimento par, a mediana é obtida calculando-se a média dos dois elementos mais centrais. São estes valores de mediana obtidos a partir da varredura pixel a pixel da imagem de entrada que irão compor a imagem de saída.



Na figura a seguir, observe que o filtro da média não é adequado para o tratamento de imagens com ruído sal-e-pimenta, pois além de deteriorar as bordas, o ruído ainda fica visível. Já o filtro da mediana remove o ruído e as bordas presentes na imagem são pouco afetadas em termos de suavização.



A função Octave para aplicar o filtro da mediana em uma imagem é a **`medfilt2`** [https://octave.sourceforge.io/image/function/medfilt2.html]. Exemplos de opções da **`medfilt2:`**

"

```
medfilt2 (A, nhood)
medfilt2 (A, [M N])
```

Two dimensional median filtering.
Replaces elements of A with the median of their neighbours as defined by the true elements of logical matrix nhood or by a matrix of size M by N. The default nhood is a 3 by 3 matrix of true elements.
"

5.4) Filtro da mediana (função medfilt2)

Utilize o filtro da mediana para remover o ruído sal-e-pimenta da imagem *salt-and-pepper1.tif*. Compare com a saída de um filtro da média 3x3 e um filtro da média 5x5.

Nome do .m: atv05_04.m

5.5) Filtro da mediana

Implementar o filtro da mediana na unha. Use laços *for* à vontade. Pode ser pra uma janela de dimensões hard coded (fixas, que não permitem configurações). Não precisa tratar as bordas.

Nome do .m: atv05_05.m

5c) Realce de imagem usando o Laplaciano

Relembrando: em uma imagem, os valores das componentes de frequência são proporcionais às variações dos níveis de cinza com a distância. De uma forma simplificada, pode-se dizer que:

- Regiões homogêneas da imagem, nas quais os níveis de cinza apresentam poucas variações com a distância, correspondem a frequências baixas.
- Variações abruptas nos níveis de cinza, como em bordas agudas (sharp edges) e ruído, correspondem a frequências altas.

É por isso que o filtro que suaviza (reduz) o ruído e as bordas é o passa-baixas, aquele que atenua as componentes de alta frequência da imagem. Então, se quisermos a operação oposta à da suavização, chamada de *realce* de imagem (*image sharpening*), a estratégia é intensificar as componentes de alta frequência, utilizando filtros passa-altas. Como a suavização (passa-baixas) é obtida a partir da intergral (soma), é natural que o realce (passa-altas) seja obtido a partir da derivada. Ainda, o uso da derivada justifica-se se lembrarmos do próprio propósito de uma derivada, que é capturar variações em um sinal.

Um método bastante utilizado para o realce de imagens é o que utiliza uma máscara do tipo *Laplaciano*. A máscara do Laplaciano é aproximação para sinais discretos da derivada de segunda ordem. Em outras palavras, pode-se dizer que o Laplaciano é a implementação da derivada de segunda ordem em imagens. Existem diferentes versões da máscara Laplaciano, conforme descrito a seguir.

Laplaciano direto: obtido diretamente a partir da equação da aproximação da derivada de segunda ordem para sinais discretos [[GW], Tópico 3.6.2; [OM], Tópico 10.4.1; [GD], Tópico 6.4.3]. Considera as direções vertical e horizontal, o que proporciona ao Laplaciano a seguinte característica: isotrópico para rotações de 90º [[GD], Tópico 6.4.2]. Isotrópico é a qualidade de um "meio cujas propriedades físicas são iguais, qualquer que seja a direção considerada" [http://aulete.uol.com.br/nossoaulete/isotrópico]. Na prática, "um filtro isotrópico é invariante à rotação, no sentido de que rotacionar a imagem e depois aplicar o filtro fornece o mesmo resultado que aplicar o filtro e depois rotacionar o resultado" [[GW], Tópico 3.6.2]. Também existe a versão invertida [[GW], Tópico 3.6.2; [GWm], Exemplos 3.9 e 3.10]. A maioria dos autores considera que o termo 'Laplaciano' (sem especificações adicionais) refere-se à máscara *Laplaciano direto* a seguir:

0	-1	0
-1	4	-1
0	-1	0

Laplaciano direto

0	1	0
1	-4	1
0	1	0

Laplaciano invertido

Laplaciano extendido (*extended Laplacian*) [[GW], Tópico 3.6.2; [OM], Tópico 10.4.1; [GD], Tópico 6.4.3]: considera as direções vertical, horizontal e as diagonais, por isso é isotrópico para rotações de 45º [[GD], Tópico 6.4.2]. O realce utilizando o Laplaciano extendido é mais intenso que o obtido através do Laplaciano. Também existe a versão invertida [[GW], Tópico 3.6.2; [GWm], Exemplos 3.9 e 3.10]. A modificação em relação ao anterior está na inclusão de coeficientes 1 nos cantos da máscara:

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplaciano direto extendido

1	1	1
1	-8	1
1	1	1

Laplaciano invertido extendido

Ao aplicar o Laplaciano na imagem original, o que se obtém é uma imagem contendo as componentes de alta frequência da imagem original, e não a imagem original já realçada. Assim, para obter a imagem realçada final, é necessário combinar a imagem original com a saída do Laplaciano. Caso seja usada uma máscara do *Laplaciano direto*, deve-se fazer a soma da imagem original com a saída do Laplaciano. Caso seja usada uma máscara do *Laplaciano invertido*, deve-se subtrair a saída do Laplaciano da imagem original (original – Laplaciano invertido), já que os valores do Laplaciano invertido são negativos.

5.6) Função fspecial, máscara 'laplacian'

Qual é a máscara gerada pela função `fspecial('laplacian', 0)` ?

Nome do .m: atv05_06.m

5.7) Laplaciano direto e Laplaciano invertido

Como a máscara do Laplaciano contém coeficientes negativos e o coeficiente 4 (tomando como exemplo o Laplaciano direto), o resultado da convolução pode apresentar valores negativos e maiores que o valor máximo possível na imagem. Lembrando ainda que, para a obtenção da imagem realçada, deve-se combinar a imagem original com o resultado da convolução com o Laplaciano (para o Laplaciano direto, soma). Na operação de realce de imagens, o mais comum é truncar os valores que aparecem fora da faixa da imagem, pois neste caso uma normalização (função *mat2gray* ou equivalente) pode alterar significativamente o nível de cinza médio da imagem. Lembrando: truncar o pixel *p* de uma imagem de 8 bits por pixel significa fazer qualquer $p<0 \rightarrow 0$ e qualquer $p>255 \rightarrow 255$.

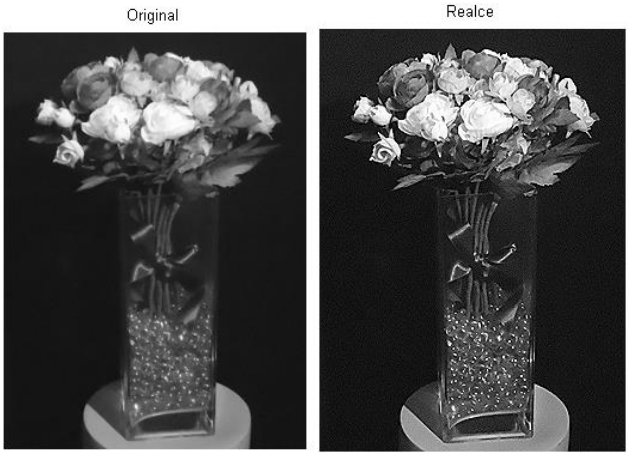
```
clear all
close all

g = imread('flowervaseg.png');

%Imfilter retorna imagem da mesma clase da
%de entrada. Se g fosse uint8 o imfilter
%truncaria os valores de saída e a
%visualização da convolução
%seria comprometida. Por isso:
gd = im2double(g);

h = fspecial('laplacian', 0);
gdL = imfilter(gd, h, 'replicate');
gdLs = gd - gdL;
gdLsu = im2uint8(gdLs); %trunca

%Display
figure, imshow(g)
title('Original')
%mat2gray apenas para a
%visualização do Laplaciano
gdLn = mat2gray(gdL);
figure, imshow(gdLn)
title('Laplaciano')
figure, imshow(gdLsu)
title('Realce')
```



Obetenha a imagem *flowervaseg.png* realçada utilizando a máscara do Laplaciano direto (elemento central positivo). O resultado deve ser similar ao do exemplo neste exercício, que foi obtido usando o Laplaciano invertido (elemento central negativo). Mostre também a saída normalizada (autocontraste *mat2gray*) ao invés de truncada para verificar se o nível de cinza médio é alterado.

Nome do .m: atv05_07.m

5.8) Laplaciano direto extendido e Laplaciano invertido extendido

A opção '`laplacian`' da função *fspecial* gera a máscara do Laplaciano utilizando a equação a seguir [https://octave.sourceforge.io/image/function/fspecial.html]:

$$\frac{4}{\alpha+1}\begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

A idéia é que a máscara seja mais genérica, permitindo pequenos ajustes no resultado do realce. A opção default é `alfa = 0.2`. Fazendo `alfa = 0` gera-se a máscara do Laplaciano invertido com os coeficientes que conhecemos e usamos no exercício anterior: [0 1 0; 1 -4 1; 0 1 0]. Não é possível gerar a máscara do Laplaciano direto extendido através da função `fspecial`, mas pode-se criá-la da mesma forma que uma matriz comum (`h = [...]`).

Obetenha a imagem *flowervaseg.png* realçada utilizando a máscara do Laplaciano invertido extendido ([1 1 1; 1 -8 1; 1 1 1]). Compare com a imagem realçada utilizando a máscara do Laplaciano invertido ([0 1 0; 1 -4 1; 0 1 0]). Qual a diferença? (Resposta: o realce com o Laplaciano invertido extendido é mais intenso. Faça e veja!).

Nome do .m: atv05_08.m

5d) Realce de imagem usando o método unsharp

Na implementação mais comum do método unsharp, uma versão suavizada da imagem original é subtraída da imagem original (original - suavizada), resultando na chamada *imagem máscara unsharp*. Esta imagem é então somada à imagem original.

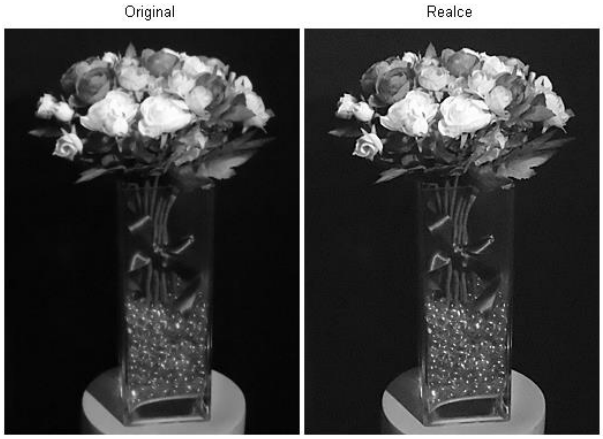
```
clear all, close all

g = imread('flowervaseg.png'); %(3)
g = double(g); %double, uint8(1)
%g = im2double(g);%im2double,im2uint8(2)

h = fspecial('gaussian', [5 5], 1);
gg = imfilter(g, h, 'replicate');

unshmask = g - gg;
gunsharp = g + unshmask; %(3)
gunsharp = uint8(gunsharp);%double,uint8(1)
%gunsharp = ...,
%im2uint8(gunsharp); %im2double,im2uint8(2)

%Display
figure, imshow(g,[])
title('Original')
figure, imshow(unshmask,[])
title('Imagem máscara unsharp')
figure, imshow(gunsharp)
title('Realce')
```



O código exemplo deste exercício é uma boa oportunidade para relembrar como utilizar corretamente as classes `uint8` e `double` para a manipulação de imagens no Octave. No exemplo, utilizamos a função `double` para converter a imagem de entrada de `uint8` para `double`. Fizemos isto porque sabemos que a operação de subtração que gera `unshmask` pode gerar números negativos e a soma que gera `gunsharp` pode gerar também números maiores que 255. Números negativos e maiores que 255 não podem ser representados na classe `uint8`, números negativos seriam truncados em zero e números maiores que 255 seriam truncados em 255. Depois, usamos a função `uint8` para converter a imagem `gunsharp` para `uint8` para a visualização e eventual armazenamento em arquivo. Sabemos que, ao usar a função `uint8` para passar a imagem de double para `uint8`, os valores menores que 0 e maiores que 255 são truncados, mas é isso que desejamos. No realce de uma imagem pelo método unsharp, assim como pelo Laplaciano, é mais apropriado truncar a saída do que normalizar, para não alterar o nível de cinza médio da imagem.

Também é possível usar a combinação das linhas de código comentadas identificadas por '(2)'. Nesse caso, a imagem de entrada também é convertida para double, mas agora pela função `im2double`, que representa a imagem usando valores entre 0 e 1, ao invés de entre 0 e 255. Assim, a imagem `gunsharp` é depois transformada para `uint8` usando a função `im2uint8`, que considera que a entrada está entre 0 e 1 e transforma estes valores para a classe `uint8` na faixa 0 a 255. Sabemos que, ao usar a função `im2uint8` para passar a imagem de double para `uint8`, os valores menores que 0 são truncado em 0 e os maiores que 1 são truncados em 255, mas é isso desejamos.

Ainda, é possível usar a combinação das linhas de código comentadas identificadas por '(3)'. Nesse caso, não fazemos nenhuma conversão de classe entre `double` e `uint8`. Sabemos que o resultado das operações que geram `unshmask` e `gunsharp` podem gerar valores negativos e maiores que 255, que serão perdidos (truncados em 0 e 255), mas tudo bem, já que desejamos uma saída truncada. Porém, este método não é indicado, porque perde-se precisão nas operações matemáticas.

5.9) Parâmetros do unsharp ('amplificação' da máscara unsharp)

É possível multiplicar a imagem máscara unsharp por uma constante de amplificação (número maior que 1) antes de somá-la à imagem original. Com isso, pode-se intensificar o realce. Faça testes com a imagem *flowervaseg.png* e mostre os resultados.

Nome do .m: atv05_09.m

5.10) Parâmetros do unsharp (geração da máscara unsharp)

O resultado do unsharp também apresenta alteração caso altere-se o filtro passa-baixas utilizado. Faça testes com a imagem *flowervaseg.png* e mostre os resultados.

Nome do .m: atv05_10.m

Referências

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

[GWm] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, Digital image processing using MATLAB, Pearson Prentice Hall, 2004.

[GW] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, Digital image processing, Pearson Prentice Hall, 3rd ed, 2008.

[GD] Geoff Dougherty, Digital image processing for medical applications, Cambridge University Press, 2009.
Imagens disponíveis em www.cambridge.org/dougherty