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

Notas de aula e prática MATLAB #04

Arquivos necessários

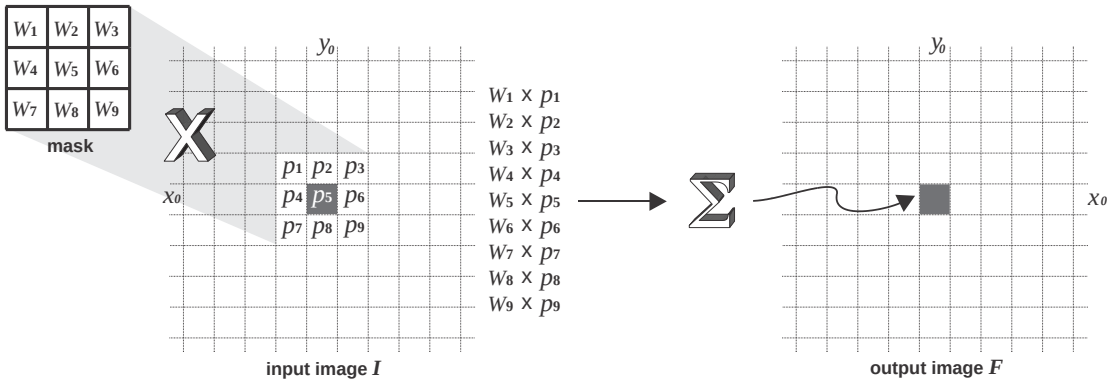
1. Lenna256g.png [Adaptada de https://en.wikipedia.org/wiki/Lenna]
2. b5s.40.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
3. b5s.100.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
4. flowervaseg.png [http://www.digitalcamerainfo.com/content/Samsung-WB150F-Digital-Camera-Review/Sample-Photos.htm]

4a) Operações baseadas em vizinhança

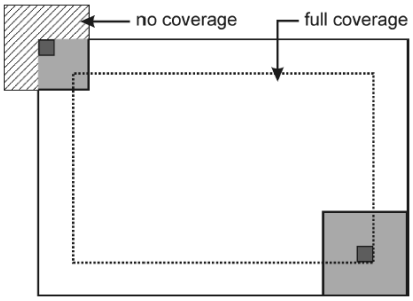
Nas operações ponto-a-ponto (point operations) apenas um único pixel da imagem de entrada é considerado de cada vez para o cálculo da imagem de saída. Por exemplo, para ajustar o contraste de uma imagem utilizando a função gamma, para cada pixel x da imagem de entrada calcula-se x^{gamma}. Já as operações baseadas em vizinhança (neighborhood operations) levam em consideração o pixel x e seus pixels vizinhos. Por exemplo, para suavizar uma imagem utilizando o filtro da *média móvel*, também chamado de *box filter*, calcula-se a média dentro de uma máscara quadrada que tem x como elemento central. Para uma máscara 3-por-3, por exemplo, o pixel de saída correspondente ao pixel central x da imagem de entrada é igual à média dos 9 pixels pertencentes à máscara. Estas operações podem ser descritas matematicamente através da operação de convolução com uma máscara, que define a vizinhança do pixel x e os coeficientes usados no cálculo da convolução. Geralmente, a máscara é quadrada, de dimensões 3-por-3 pixels, 5-por-5 pixels, 7-por-7 pixels e assim por diante, para proporcionar simetria às operações. O pixel x da imagem original que está sendo processado é o pixel central da máscara.

A figura a seguir [[OM], Tópico 10.2, Figura 10.1] descreve a operação de convolução do pixel de coordenadas x_0,y_0 da imagem de entrada com uma máscara de convolução 3-por-3, de coeficientes $W_1...W_9$. Esta operação é repetida para cada pixel da imagem de entrada, isto é, a máscara varre a imagem de entrada pixel por pixel, calculando cada pixel correspondente da imagem de saída. Textualmente, a convolução de uma máscara de coeficientes Wn com um pixel qualquer $p(i,j)$ da imagem de entrada, pode ser descrita assim:

1º. Posiciona-se o elemento central da máscara sobre o pixel $p(i,j)$ da imagem de entrada.
2º. Multiplica-se cada coeficiente Wn da máscara pelo pixel correspondente da imagem.
3º. Soma-se todos os resultados destas multiplicações.
4º. O resultado desta soma é o valor do pixel $p'(i,j)$ da imagem de saída.

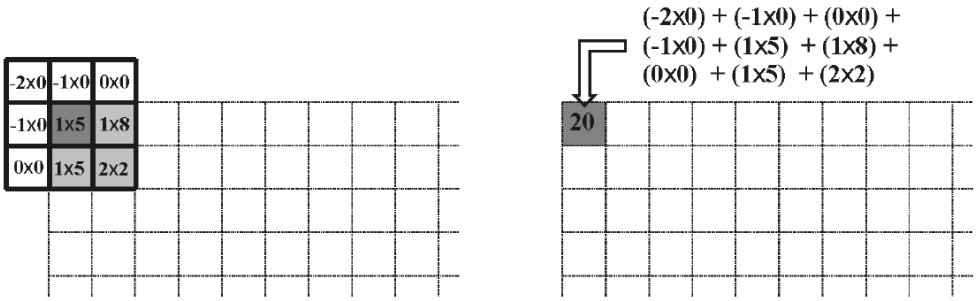


E quando a máscara é posicionada sobre um pixel dos extremos da imagem de entrada que não possui pixels vizinhos para preencher a máscara, como na situação *no coverage* da figura a seguir? [[OM], Tópico 10.2.4, Figura 10.4]



Nestes casos, o mais comum é considerar que os pixels da máscara que ficaram "para fora" da imagem de entrada estão convoluindo com pixels de valor zero (pois não existem pixels de imagem correspondentes a estes ceficientes da máscara). Esta técnica é chamada de *zero-padding* (como se os quatro extremos da imagem fossem ampliados e preenchidos com zero). A figura a seguir ilustra o zero-padding quando a máscara exemplo a seguir é convoluída com o pixel de coordenadas *p(*1,1*)* de uma imagem de entrada [[OM], Tópico 10.2.2, Figura 10.2].

2	**1**	**0**
1	**1**	**-1**
0	**-1**	**-2**



Após analisar a figura, você deve ter pensado: "Puts! A máscara deveria ser [2 1 0; 1 1 -1; 0 -1 -2], como na figura da máscara, e a máscara que estou vendo convoluir com a imagem é [-2 -1 0, -1 1 1, 0 1 2]. Ela foi espelhada em *x* e em *y* antes de convoluir". É isso mesmo. A definição matemática da operação de convolução estabelece que a máscara deve ser espelhada em relação ao eixo *x* e ao eixo *y* antes de ser aplicada (fazer as multiplicações e depois acumular). A operação *correlação* é a parente da convolução que dispensa esse espelhamento. Na maioria das aplicações práticas de PDI, as máscaras são simétricas. Com máscaras simétricas, convolução e correlação são equivalentes, já que após espelhar a máscara em *x* e *y* ela continua igual. É por isso que, em alguns textos sobre PDI, esta diferença entre a correlação e a convolução muitas vezes não é comentada.

A função `imfilter` do MATLAB utiliza como default a correlação. Veja o help no box a seguir.

As operações baseadas em vizinhança com máscaras de convolução também são chamadas de *filtros* ou ainda *filtros espaciais*. A máscara de convolução (*mask*) também recebe o nome de *kernel*, janela (*window*) ou *template*.

A função do MATLAB que faz a convolução ou correlação de uma máscara com uma imagem é a **`imfilter`** [https://www.mathworks.com/help/images/ref/imfilter.html].

"

```
B = imfilter(A,h) filters the multidimensional array A with the
                  multidimensional filter h and returns the result in B.
B = imfilter(A,h,options,...) performs multidimensional filtering according
                              to one or more specified options.
```

By default, imfilter uses correlation because the toolbox filter design functions produce correlation kernels. Use the optional parameter to use convolution.

Options that control the filtering operation, specified as a character vector, string scalar, or numeric scalar. The following table lists all supported options.

'numeric scalar, X' Input array values outside the bounds of the array are assigned the value X. When no padding option is specified, the default is 0.
'symmetric' Input array values outside the bounds of the array are computed by mirror-reflecting the array across the array border.
'replicate' Input array values outside the bounds of the array are assumed to equal the nearest array border value.
'circular' Input array values outside the bounds of the array are computed by implicitly assuming the input array is periodic.
'same' The output array is the same size as the input array. This is the default behavior when no output size options are specified.
'full' The output array is the full filtered result, and so is larger than the input array.
'corr' imfilter performs multidimensional filtering using correlation, which is the same way that filter2 performs filtering. When no correlation or convolution option is specified, imfilter uses correlation.
'conv' imfilter performs multidimensional filtering using convolution.

"

A função **`imfilter`** geralmente trabalha em conjunto com a função **`fspecial`** [https://www.mathworks.com/help/images/ref/fspecial.html], que gera as máscaras de convolução (parâmetro h descrito no help da função imfilter) mais comuns em PDI.

"

```
h = fspecial(type)
h = fspecial('average',hsize)
h = fspecial('disk',radius)
h = fspecial('gaussian',hsize,sigma)
h = fspecial('laplacian',alpha)
h = fspecial('log',hsize,sigma)
h = fspecial('motion',len,theta)
h = fspecial('prewitt')
h = fspecial('sobel')
```

"

4.1) Correlação vs convolução, opções de tratamento das bordas e dimensões da saída (função imfilter)

Use a função `imfilter` para gerar as saídas mostradas a seguir. 'A' é a entrada dada e 'H' é a máscara de convolução dada. Não alterar 'A' e 'H' antes de passá-los para a `imfilter` (exceto para obter a saída 'C2'), isto é, obter as saídas desejadas atuando apenas nos parâmetros da função `imfilter`. Resposta disponível em a04_01.m.

A

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

H

5	5	-3
5	0	-3
-3	-3	-3

B

-9	-7	-7	-7	-1
-7	0	0	0	9
-7	0	0	0	9
-7	0	0	0	9
-1	9	9	9	15

C

15	9	9	9	-1
9	0	0	0	-7
9	0	0	0	-7
9	0	0	0	-7
-1	-7	-7	-7	-9

C2
[crie H2 a partir de H e use `imfilter(A,H2)`]

15	9	9	9	-1
9	0	0	0	-7
9	0	0	0	-7
9	0	0	0	-7
-1	-7	-7	-7	-9

D

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

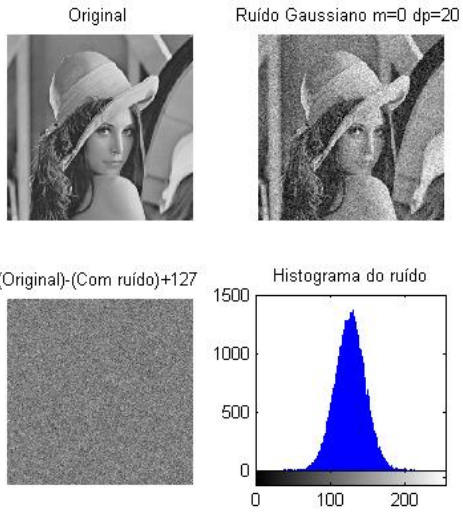
E

-3	-6	-9	-9	-9	-6	-3
-6	-9	-7	-7	-7	-1	2
-9	-7	0	0	0	9	7
-9	-7	0	0	0	9	7
-9	-7	0	0	0	9	7
-6	-1	9	9	9	15	10
-3	2	7	7	7	10	5

4b) Ruído em imagens e filtro espacial passa-baixas da média

O ruído Gaussiano é do tipo *aditivo*, isto é, incorpora-se à imagem original através da soma. A distribuição estatística Gaussiana é a que possui a conhecida função densidade de probabilidade (FDP) em forma de sino (bell-shabed curve). Falamos em FDP porque um ruído é uma variável aleatória e portanto é caracterizado pela sua FDP (distribuição estatística). A seguir podem ser vistas a imagem Lenna [http://en.wikipedia.org/wiki/Lenna] sem ruído e com ruído aditivo do tipo Gaussiano, apenas o ruído e seu histograma. Pode-se dizer que o ruído Gaussiano é o tipo mais comum em imagens adquiridas com luz visível, já que é gerado no processo de aquisição da imagem pelos sensores do tipo CCD ou CMOS. Em imagens de ressonância magnética (RM), por exemplo, o ruído é visualmente similar ao Gaussiano, mas a distribuição é do tipo Rician [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2254141/].

```
% noise_g [script]
clc, clear, close all
g = imread('Lenna256g.png');
%m=0 e desvio padrão = 20
gg = imnoise(g,'gaussian',(0/255),(20/255)^2);
%Apenas ruído
s = imsubtract(double(g), double(gg));
%Para visualizar o ruído
sviz = uint8(s + 127);
%Display
figure
imshow(g), title('Original')
figure,imshow(gg)
title('Ruído Gaussiano m=0 dp=20')
figure, imshow(sviz)
title('(Original)-(Com ruído)+127')
figure, imhist(sviz)
ylim('auto'), title('Histograma do ruído')
%Só conferindo o dp do ruído:
dp = std(double(s(:)))
```



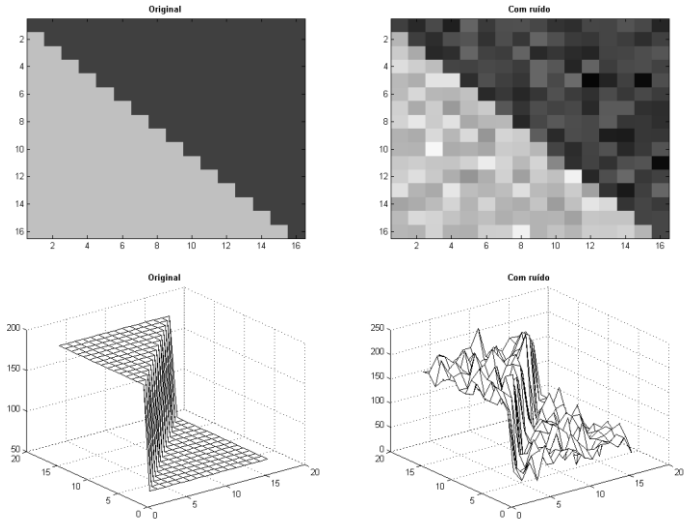
Em uma imagem, os valores das componentes de frequência são proporcionais às variações dos níveis de cinza com a distância. De uma forma simplificada, pode-se dizer que:

- 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), correspondem a frequências altas. É nesta faixa de frequências espaciais que se encontra também o ruído. Portanto, remover o ruído de uma imagem sem deteriorar as bordas é uma tarefa difícil.

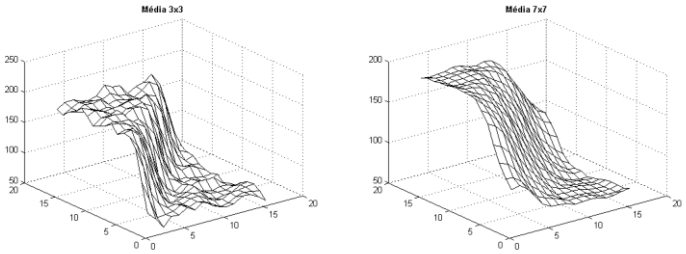
O tipo de filtro utilizado para remover o ruído Gaussiano de uma imagem é o *passa-baixas*. Possui este nome porque deixa passar as frequências baixas e corta as frequências altas (ruído) presentes em uma imagem. Este tipo de filtro suaviza a imagem como um todo, tanto o ruído quanto as bordas. Ao suavizar as bordas, a imagem fica com o efeito de *imagem borrada* (*blurring*), o que não é desejável no contexto da recuperação de imagens (image restoration).

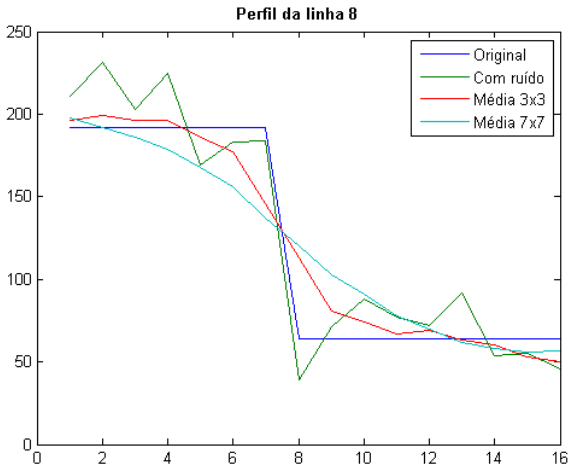
O termos 'suavizar o ruído' é utilizado aqui como sinônimo de 'reduzir o ruído'. A seguir são mostradas uma imagem sintética original, a mesma imagem sintética contaminada com ruído *uniforme* e depois processada com o filtro da média. As imagens foram plotadas como uma superfície para auxiliar na visualização do conceito de suavização do ruído e das bordas. Observar os valores dos pixels de uma linha da imagem na forma de um gráfico convencional também auxilia na visualização do conceito de suavização do ruído e das bordas. Este tipo de gráfico é chamado de *perfil* de uma linha da imagem.

```
% edge_smooth [script]
clc, clear, close all
%Cria imagem
nr = 16; nc = 16;
a = triu(ones(nr,nc))*64;
b = tril(ones(nr,nc),-1)*192;
g = uint8(a+b);
%Média zero e desvio padrão 20
gg = imnoise(g,'gaussian',...
    (0/255),(20/255)^2);
%Filtro da média
h = fspecial('average', [3 3]);
ggm1 = imfilter(gg, h, 'symmetric');
h = fspecial('average', [7 7]);
ggm2 = imfilter(gg, h, 'symmetric');
%Display
figure, image(g), colormap(gray(256))
title('Original','FontWeight','bold')
figure, image(gg), colormap(gray(256))
title('Com ruído','FontWeight','bold')
figure,
mesh(double(g),'EdgeColor','black')
title('Original','FontWeight','bold')
```



```
figure,
mesh(double(gg),'EdgeColor','black')
title('Com ruído','FontWeight','bold')
figure,
mesh(double(ggm1),'EdgeColor','black')
title('Média 3x3','FontWeight','bold')
figure,
mesh(double(ggm2),'EdgeColor','black')
title('Média 7x7','FontWeight','bold')
x = 1:size(g,2);
i = ceil(size(g,1)/2);
figure
plot(x,g(i,:),x,gg(i,:),...
    x,ggm1(i,:),x,ggm2(i,:)),
legend('Original','Com ruído',...
    'Média 3x3', 'Média 7x7')
title(['Perfil da linha ',...
    num2str(i)],'FontWeight','bold')

title('Com ruído','FontWeight','bold')
figure,
mesh(double(ggm1),'EdgeColor','black')
title('Média 3x3','FontWeight','bold')
figure,
mesh(double(ggm2),'EdgeColor','black')
title('Média 7x7','FontWeight','bold')
x = 1:size(g,2);
i = ceil(size(g,1)/2);
figure
plot(x,g(i,:),x,gg(i,:),...
    x,ggm1(i,:),x,ggm2(i,:)),
legend('Original','Com ruído',...
    'Média 3x3', 'Média 7x7')
title(['Perfil da linha ',...
    num2str(i)],'FontWeight','bold')
```





As imagens abaixo apresentam as saídas do filtro passa-baixas da média (*fspecial* 'average') de máscaras 3-por-3, 5-por-5 e 7-por-7, aplicado à imagem Lenna com ruído Gaussino de média 0 e desvio padrão 20 (código MATLAB anterior). Assim como na imagem sintética, observe que quanto maior é a dimensão da máscara, maior é a suavização do ruído. Porém, o efeito de blurring também é maior.

4.2) Filtro da média (funções imfilter e fspecial)

Usar as funções `imfilter` e `fspecial` para aplicar o filtro da média (box filter) na imagem de RM *b5s.40.bmp* e *b5s.100.bmp*. Utilize duas dimensões diferentes de janela à sua escolha, e observe como o tamanho da janela atua na intensidade da suavização. Mostre as imagens de entrada e de saída (in vs. out). Resposta disponível em a04_02.m.

4c) Filtro espacial passa-baixas Gaussiano

Outra função utilizada para a suavização de uma imagem é a Gaussiana. A função Gaussiana 2D é definida pela equação a seguir.

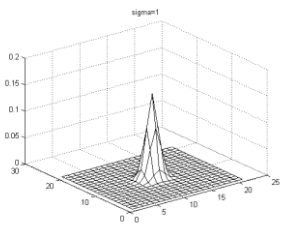
$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

O principal parâmetro da função Gaussiana é o desvio padrão, sigma, que controla a "abertura" da curva. É importante observar que, se o desvio padrão for muito alto e as dimensões da máscaras pequenas, a função Gaussiana pode "não caber na máscara". Assim, o tamanho da máscara é determinado pelo desvio padrão. Deve-se utilizar uma combinação de tamanho de máscara e desvio padrão que permita que os coeficientes da máscara caiam para próximo de zero nas bordas da máscara. Uma combinação bastante utilizada é a de **máscara 5-por-5 com desvio padrão igual a 1** [[NA], Tópico 3.4.4]. O termo que multiplica a exponencial serve para a normalização da curva, de maneira que a soma de todos os coeficientes da máscara é sempre igual à 1, para qualquer valor de sigma. Com isso, a máscara não altera o valor médio da imagem. A função `fspecial` (opção '`gaussian`') retorna uma máscara com os coeficientes normalizados, independente do tamanho da máscara, isto é, mesmo que a Gaussiana não caiba na máscara a soma dos coeficientes é igual a 1.

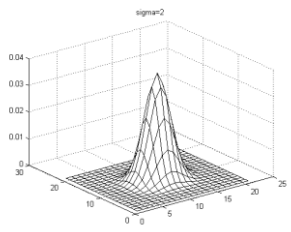
```
% gaussian_equation [script]
clc, clear, close all

[X,Y] = meshgrid(-10:10, -10:10);

sigma = 1; %desvio padrão
v = sigma^2; %variância
gauss1 = (1/(2*pi*v))*exp(-(X.^2+Y.^2)/(2*v));

sigma = 2; %desvio padrão
v = sigma^2; %variância
gauss2 = (1/(2*pi*v))*exp(-(X.^2+Y.^2)/(2*v));

sigma = 3; %desvio padrão
v = sigma^2; %variância
gauss3 = (1/(2*pi*v))*exp(-(X.^2+Y.^2)/(2*v));

sigma = 4; %desvio padrão
v = sigma^2; %variância
gauss4 = (1/(2*pi*v))*exp(-(X.^2+Y.^2)/(2*v));
```

```
%Display
figure,
mesh(gauss1, 'EdgeColor', 'black')
title('sigma=1')

figure,
mesh(gauss2, 'EdgeColor', 'black')
title('sigma=2')

figure,
mesh(gauss3, 'EdgeColor', 'black')
title('sigma=3')

figure,
mesh(gauss4, 'EdgeColor', 'black')
title('sigma=4')
```

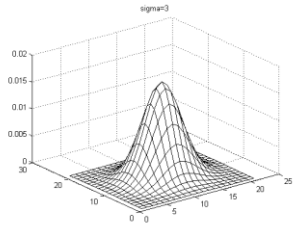
sigma=1



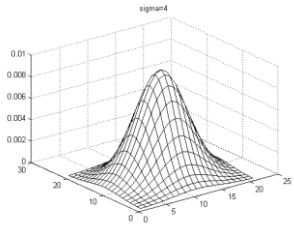
sigma=2



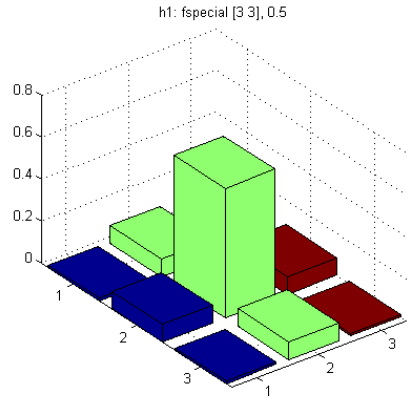
sigma=3

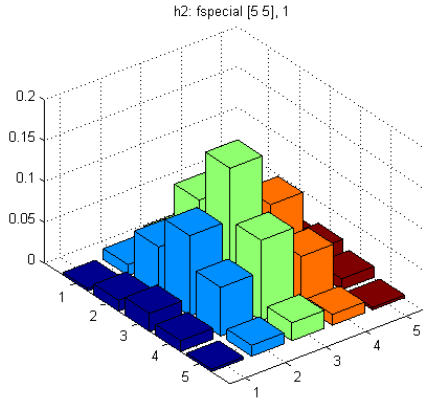


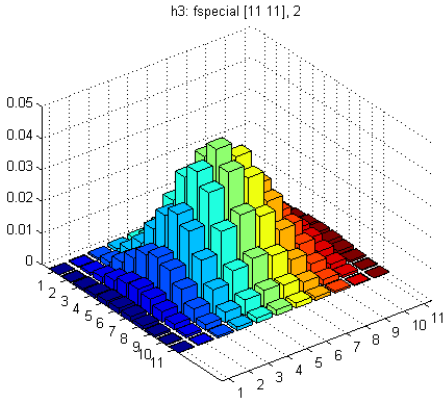
sigma=4



```
% gaussian_fspecial [script]
clc, clear, close all
%Parâmetros: 'gaussian', [dimensões],
variância
h1 = fspecial('gaussian', [3 3], 0.5);
sum(h1(:)) %(=1)só pra notar normaliz.
h2 = fspecial('gaussian', [5 5], 1);
sum(h2(:)) %(=1)só pra notar normaliz.
h3 = fspecial('gaussian', [11 11], 2);
sum(h3(:)) %(=1)só pra notar normaliz.
%Display
figure, bar3(h1)
title('h1: fspecial [3 3], 0.5')
figure, bar3(h2)
title('h2: fspecial [5 5], 1')
figure, bar3(h3)
title('h3: fspecial [11 11], 2')
```







A maioria dos autores considera que o filtro passa-baixas Gaussiano fornece imagens com uma suavização mais natural para o observador humano, portanto mais adequada do que a obtida com o filtro da média. Além disso, o filtro Gaussiano suaviza o ruído deteriorando menos as bordas, se comparado ao filtro da média [em [NA], Tópico 3.4.4 há uma boa explicação disso, em função do comportamento na frequência de cada filtro]. As imagens a seguir apresentam as saídas do filtro passa-baixas Gaussiano (`fspecial 'gaussian'`) para as máscaras geradas anteriormente. Observe que quanto maior é o sigma, maior é a suavização do ruído, mas também maior é o efeito de blurring.

4.3) Filtros espaciais passa-baixas (filtro Gaussiano)

Crie um script para avaliar a afirmação "O principal parâmetro da função Gaussiana é o desvio padrão, sigma, que controla a "abertura" da curva. É importante observar que, se o desvio padrão for muito alto e as dimensões da máscaras pequenas, a função Gaussiana pode "não caber na máscara". Assim, o tamanho da máscara é determinado pelo desvio padrão. Resposta disponível em a04_03.m

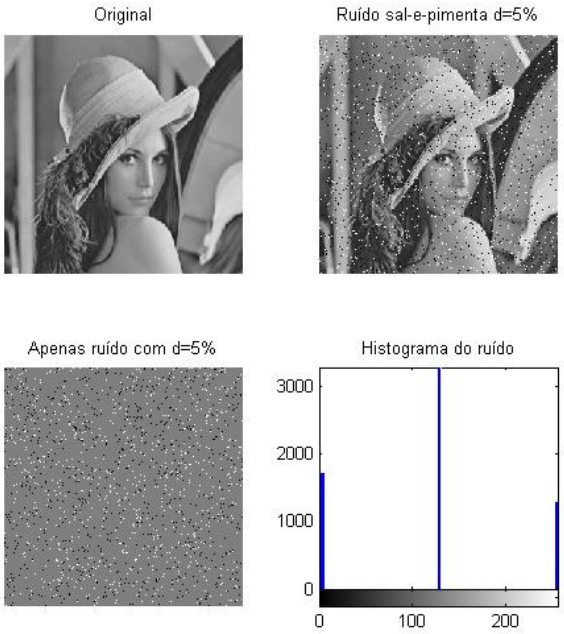
4.4) Filtro Gaussiano (funções imfilter e fspecial)

Usar as funções `imfilter` e `fspecial` com o filtro Gaussiano de janela 5-por-5 e sigma 1 para suavizar o ruído das imagens de RM *b5s.40.bmp* e *b5s.100.bmp*. Compare a saída deste filtro Gaussiano com a saída do filtro da média 5x5, para verificar a validade da afirmação: "a maioria dos autores considera que o filtro passa-baixas Gaussiano fornece imagens com uma suavização mais natural para o observador humano, portanto mais adequada do que a obtida com o filtro da média. Além disso, o filtro Gaussiano suaviza o ruído deteriorando menos as bordas, se comparado ao filtro da média de mesmas dimensões". Resposta disponível em a04_04.m

4b) Ruído sal-e-pimenta e filtro de estatística de ordem

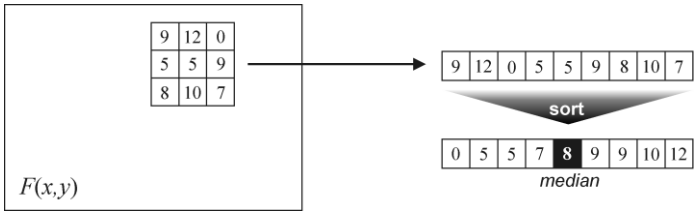
O ruído do tipo *sal-e-pimenta* (*salt-and-pepper*) é classificado como *impulsivo* (o Gaussiano é aditivo). É composto por pixels brancos (255 para imagens de 8 bits) e pixels pretos (0) distribuídos uniformemente sobre a imagem. Portanto, o nome sal e pimeta faz sentido: sal para os pixels de ruído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

```
% noise_sp [script]
clc, clear, 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')
```

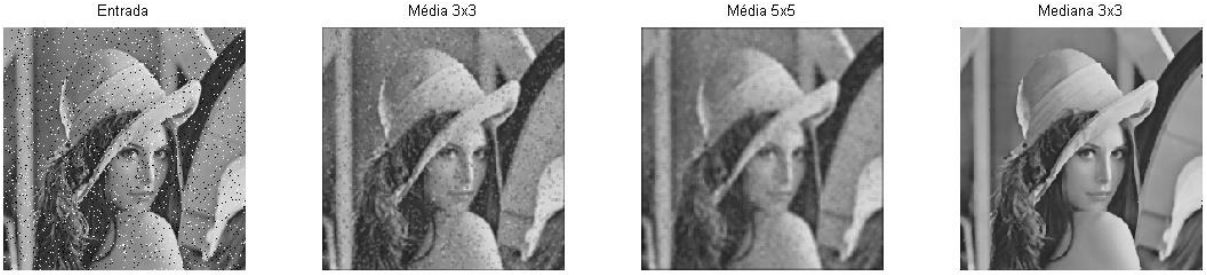


O filtro mais utilizado para a remoção do ruído sal-e-pimenta é o *filtro da mediana*. O filtro da mediana pertence a uma família denominada de filtros de *estatística de ordem*, pois para a obtenção da mediana os pixels da imagem de entrada pertencentes à janela devem estar ordenados, isto é, organizados 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 MATLAB para aplicar o filtro da mediana em uma imagem é a **`medfilt2`** [https://www.mathworks.com/help/images/ref/medfilt2.html]:

"

`J = medfilt2(I)` performs median filtering of the image I in two dimensions. Each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image.

`J = medfilt2(I,[m n])` performs median filtering, where each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image.

`J = medfilt2(___,padopt)` controls how medfilt2 pads the image boundaries.

"

4.5) Filtro da mediana (função medfilt2)

Utilize o filtro da mediana com a função `medfilt2` com uma janela 3x3 para remover o ruído sal-e-pimenta da imagem *salt-and-pepper1.tif*. Mostre as imagens de entrada e de saída (in vs. out) e compare com a saída de um filtro da média 3x3 e um filtro da média 5x5. Resposta disponível em a04_05.m

4c) 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), faz sentido afirmar que o realce (passa-altas) é 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

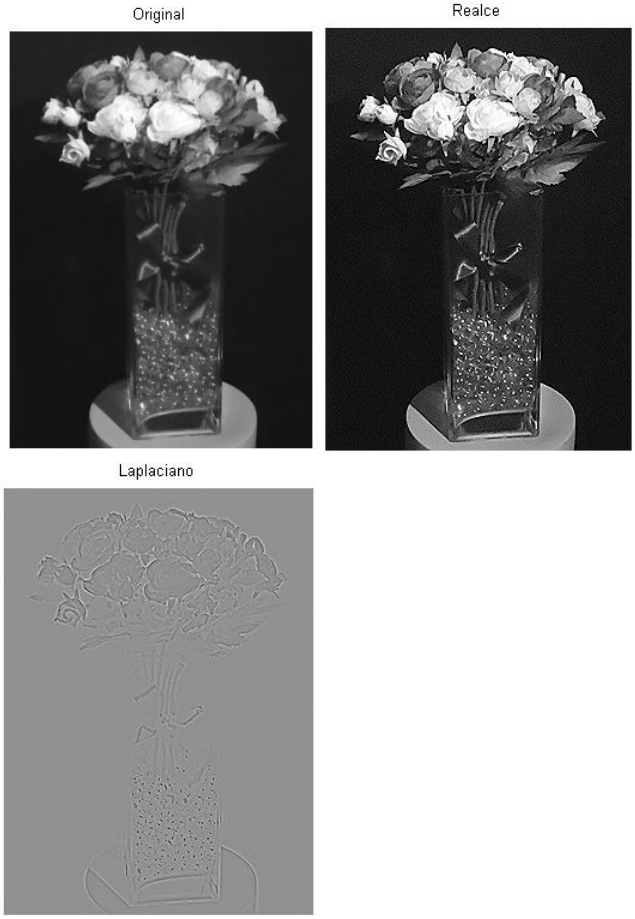
Laplaciano invertido extendido

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.

4.6) 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 $p>255 \rightarrow 255$.

```
% sharpening_lap_neg_center [script]
clc, clear, 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')
```



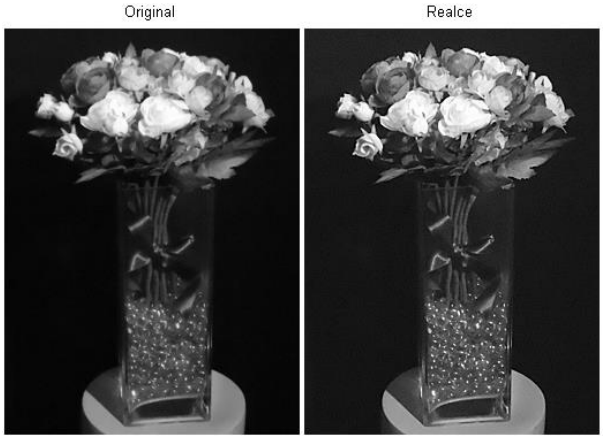
Obetenha a imagem *flowervaseg.png* realçada utilizando a máscara do Laplaciano direto (elemento central positivo). O resultado deve ser similar ao do exemplo acima, que foi obtido usando o Laplaciano invertido (elemento central negativo). Resposta disponível em a04_06.m

4d) 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.

```
% sharpening_unsharp [script]
clc, clear, close all

g = imread('flowervaseg.png');
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;
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 acima é uma boa oportunidade para relembrar como utilizar corretamente as classes `uint8` e `double` para a manipulação de imagens no MATLAB. 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.

4.7) 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. Resposta disponível em a04_07.m

4.8) 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. Resposta disponível em a04_08.m

Referências

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

[NA] Mark Nixon, Alberto Aguado, Feature extraction and image processing, Academic Press, 2008.

[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