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

Notas de aula e prática Octave #05

Arquivos necessários

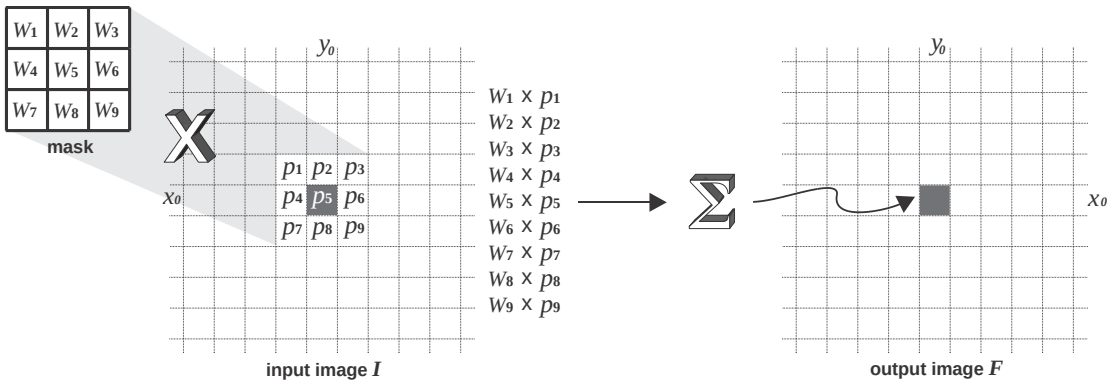
1. Lenna256g.png [Adaptada de https://en.wikipedia.org/wiki/Lenna]
2. b5s.40.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
3. b5s.100.bmp [Gerada em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]

5a) Operações baseadas em vizinhança

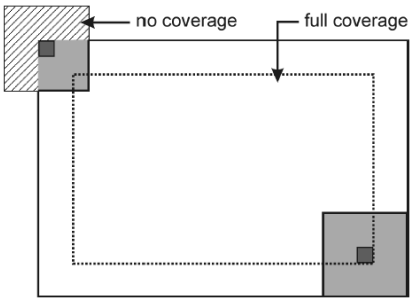
Nas operações ponto-a-ponto (point operations) apenas um único pixel da imagem de entrada é considerado de cada vez para o cálculo da imagem de saída. Por exemplo, para ajustar o contraste de uma imagem utilizando a função gamma, para cada pixel *x* da imagem de entrada calcula-se x^{gamma}. Já as operações 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.

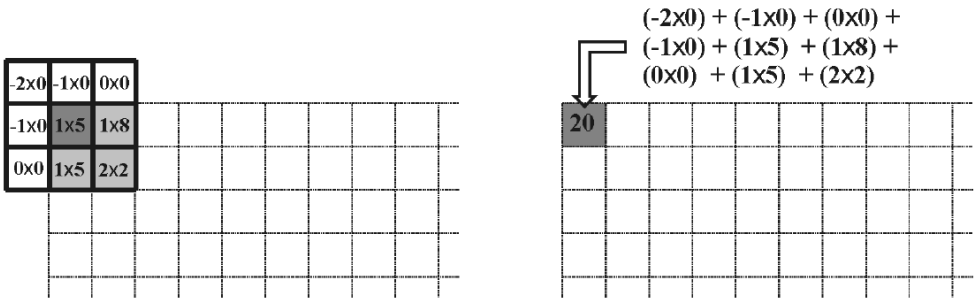


E quando a máscara é posicionada sobre um pixel dos extremos da imagem de entrada que não possui pixels vizinhos para preencher a máscara, como na situação *no coverage* da figura a seguir? [[OM], Tópico 10.2.4, Figura 10.4]



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 Octave 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 Octave que faz a convolução ou correlação de uma máscara com uma imagem é a **imfilter** [https://octave.sourceforge.io/image/function/imfilter.html].
"
J = imfilter(I, f, options, …)
Computes the linear filtering of the image I and the filter f…

The function also accepts a number of optional arguments that control the details of the filtering. The following options is currently accepted:

's' If a scalar input argument is given, the image is padded with this scalar as part of the filtering. The default value is 0.
'symmetric' The image is padded symmetrically.
'replicate' The image is padded using the border of the image.
'circular' The image is padded by circular repeating of the image elements.
'same' The size of the output image is the same as the input image. This is the default behaviour.
'full' Returns the full filtering result.
'corr' The filtering is performed using correlation. This is the default behaviour.
'conv' The filtering is performed using convolution.
"

A função **imfilter** geralmente trabalha em conjunto com a função **fspecial** [https://octave.sourceforge.io/image/function/fspecial.html], que gera as máscaras de convolução (parâmetro f descrito no help da função imfilter) mais comuns em PDI.
"
fspecial (type, …)
Create spatial filters for image processing. type is a string specifying the filter name. The input arguments that follow are type specific. The return value is a correlation kernel, often to be used by imfilter. Alguns kernels:

fspecial ('average', lengths) Rectangular averaging filter. The optional argument lengths controls the size of the filter. If lengths is an integer N, a N by N filter is created. If it is a two-vector with elements N and M, the resulting filter will be N by M. By default a 3 by 3 filter is created.
fspecial ('gaussian', lengths, sigma) Create Gaussian filter. Returns a N dimensional Gaussian distribution with standard deviation sigma and centred in an array of size lengths. lengths defaults to [3 3] and sigma to 0.5. If lengths is a scalar, it returns a square matrix of side lengths, .i.e., its value defines both the number of rows and columns.
fspecial ('laplacian', alpha) 3x3 approximation of the laplacian. The filter is approximated as
(4/(alpha+1)) * [alpha/4 (1-alpha)/4 alpha/4
(1-alpha)/4 -1 (1-alpha)/4
alpha/4 (1-alpha)/4 alpha/4];
where alpha is a number between 0 and 1. By default it is 0.2.
"

5.1) 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 a05_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

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

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

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

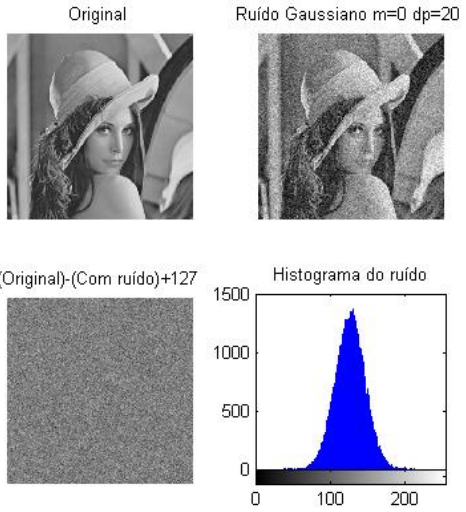
E

-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

5b) 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/].

```
clear all, 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(:)))
```



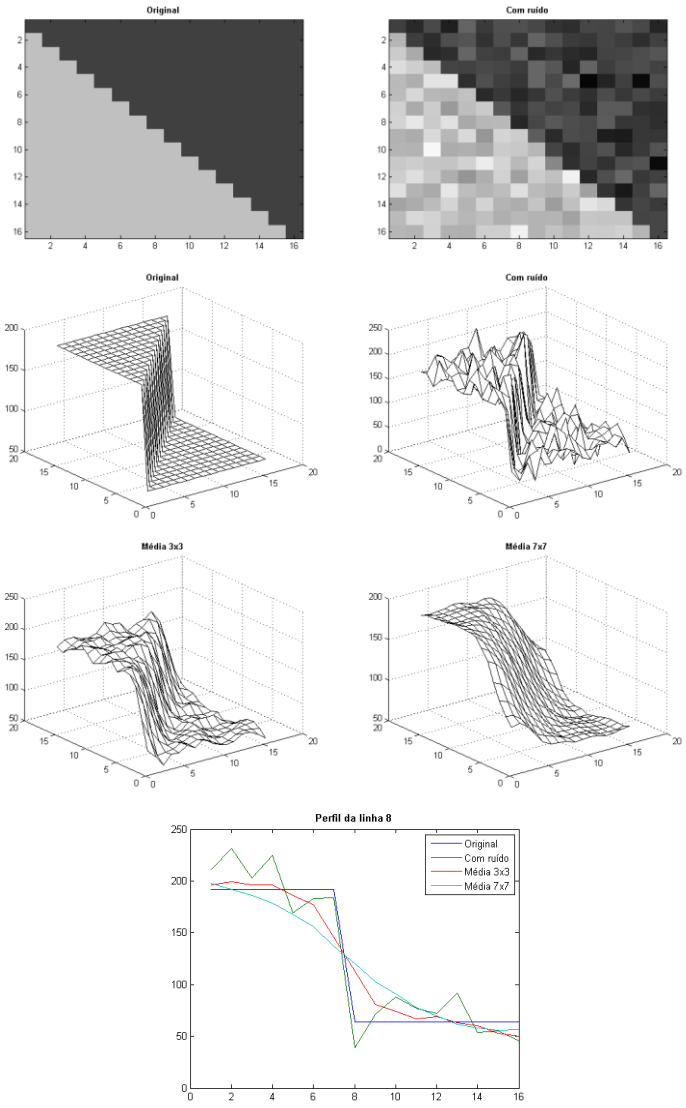
Em uma imagem, os valores das componentes de frequência são proporcionais às variações dos níveis de cinza com a distância. De uma forma simplificada, pode-se dizer que:

- 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.

```
clear all, close all, clc
%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')
```



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 Octave 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.

5c) Filtro espacial passa-baixas Gaussiano

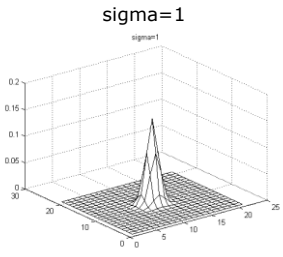
Outra função utilizada para a suavização de uma imagem é a Gaussiana. A função Gaussiana 2D é definida pela equação a seguir.

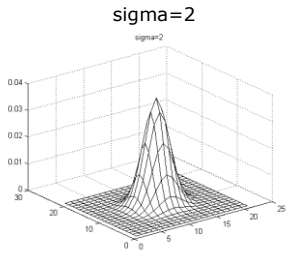
$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

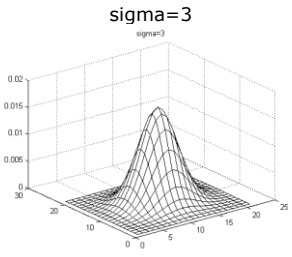
O principal parâmetro da função Gaussiana é o desvio padrão, sigma, que controla a "abertura" da curva. É importante observar que, se o desvio padrão for muito alto e as dimensões da máscaras pequenas, a função Gaussiana pode "não caber na máscara". Assim, o tamanho da máscara é determinado pelo desvio 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.

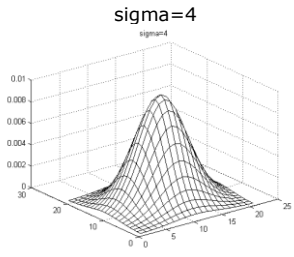
```
clear all, close all
[X,Y] = meshgrid(-10:10, -10:10);

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')
```

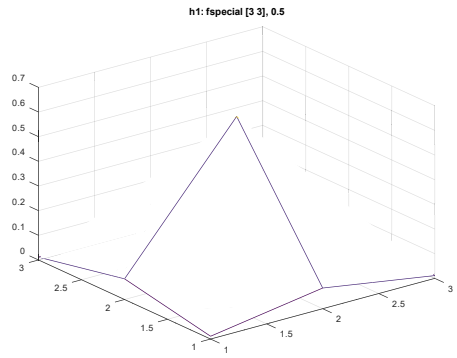


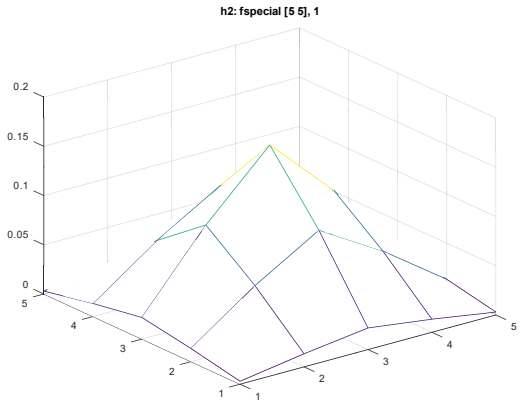


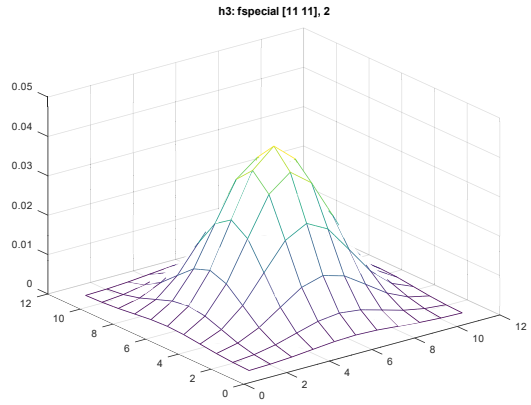





```
clear all, close all
%Parâmetros: 'gaussian', [dimensões],
variância
h1 = fspecial('gaussian', [3 3], 0.5);
sum(h1(:)) %(=1)só pra notar normaliz.
h2 = fspecial('gaussian', [5 5], 1);
sum(h2(:)) %(=1)só pra notar normaliz.
h3 = fspecial('gaussian', [11 11], 2);
sum(h3(:)) %(=1)só pra notar normaliz.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

5.2) 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 a05_02.m

5.3) 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 a05_03.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.