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

CC BY NC ND

Notas de aula e prática Octave #07

Arquivos necessários [disponíveis no zip]

1. cameraman.tif [https://homepages.cae.wisc.edu/~ece533/images/]
2. Usado nos slides: 09_01_s13.png
3. Usado nos slides: mycoins02.png
4. Usado nos slides: 42049.png [https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/images/gray/42049.html]

7a) Detecção de bordas por gradiente (primeira derivada)

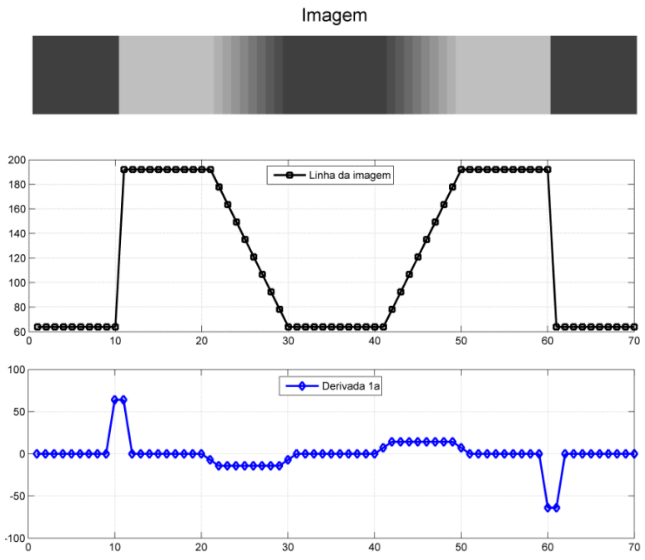
Uma borda é uma descontinuidade na intensidade dos níveis de cinza em uma determinada direção. O objetivo da detecção de bordas é encontrar a fronteira entre os objetos, ou entre os objetos e o fundo da imagem. Os métodos de detecção de bordas baseiam-se na análise das diferenças de intensidade entre pixels vizinhos da imagem. Nos métodos mais simples, isto é implementado utilizando-se máscaras de convolução que calculam uma aproximação discreta da derivada de primeira ordem em cada pixel da imagem. Em uma imagem, a derivada de primeira ordem também é chamada de *gradiente da imagem*, por isso, estas máscaras também são denominadas de *filtros de gradiente* [BB], *operadores de gradiente discretos* [[AB] Tópico 19.2.2]] ou *operadores de primeira ordem* [NA]. Também é possível utilizar a derivada de segunda ordem em cada pixel da imagem, utilizando o operador Laplaciano (usado no sharpening) ou outros com características parecidas.

Uma das definições[1] da derivada de primeira ordem e a sua aproximação para sinais discretos é [[AM], Tópico 8.2.1, Página 147]:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} = \frac{f(x+1) - f(x-1)}{2}$$

No exemplo a seguir é mostrada a derivada de primeira ordem para uma linha da imagem sintética que contém bordas do tipo degrau (step) e rampa (ramp). As bordas do tipo degrau são abruptas e, as do tipo rampa, mais suaves. Na saída da derivada de primeira ordem, valores diferentes de zero correspondem às bordas da imagem original. Quanto mais abrupta é a borda, maior é o valor da derivada.

```
% a07teo_01 [script]
clear all, close all, clc

g1 = ones(1,10)*64;
g2 = ones(1,10)*192;
g3 = linspace(192,64,10);
% Imagem sintética com bordas
% do tipo degrau e rampa
g = [g1 g2 g3 g1,...
     fliplr(g3) g2 g1];
g = repmat(g,9,1);
ncol = size(g,2);
% Amostra linha da imagem
row = g(1,1:ncol);
% Aloca derivada prim.
d = zeros(1,ncol);
% Aloca derivada seg.
dd = zeros(1,ncol);
```



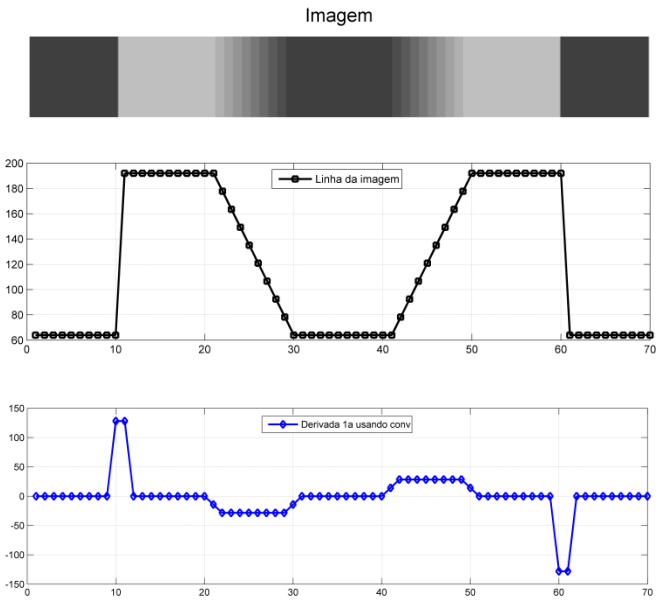
[1] "There are many possible derivative-approximation filters for use in gradient estimation. …Two simple approximation schemes for the derivative in one dimension are: i) **First difference:** $f(n) - f(n\text{-}1)$ ii) **Central difference:** $\frac{1}{2}[f(n+1) - f(n\text{-}1)]$. " [[AB] Chapter 19 - Gradient and Laplacian Edge Detection, Topic 19.9.2. Discrete Gradient Operators, pp. 504]

```
% Derivada de primeira ordem
% segundo definição
for k = 2 : ncol-1
    d(k) = row(k+1) - row(k-1);
end
d = d/2;

%Display
figure, image(g)
colormap(gray(256));
title('Imagem');
axis off
% Relação de aspecto para
% mostrar pixels quadrados
axis image
figure
x = 1:ncol;
plot(x,row,'-ks','LineWidth',2)
grid
legend('Linha da imagem')
figure
plot(x,d,'-bd','LineWidth',2)
grid
legend('Derivada 1a')
```

Lembrando:
- A derivada de uma constante (não tem variação) é zero.
- A derivada de um degrau é um impulso.
- A derivada de uma reta (inclinação constante) é uma constante.

Note que a mesma operação de derivada realizada através da equação, pode também ser realizada através de uma convolução com uma máscara de uma linha e três colunas com os valores [-1 0 1], como no exemplo a seguir (sem o fator de escala 1/2).

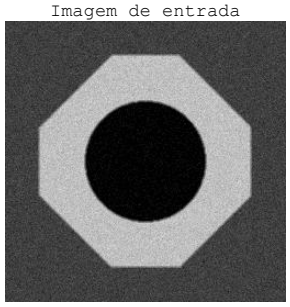
```
% a07teo_02 [script]
clear all, close all, clc

g1 = ones(1,10)*64;
g2 = ones(1,10)*192;
g3 = linspace(192,64,10);
% Imagem sintética com bordas
% do tipo degrau e rampa
g = [g1 g2 g3 g1 fliplr(g3),...
    g2 g1];
g = repmat(g,9,1);
ncol = size(g,2);
% Amostra linha da imagem
row = g(1,1:ncol);
% Máscara para derivada prim.
% por convolução para
% reproduzir f(x+1)-f(x-1).
% Portanto, temos que fazer convolução
% com [1 0 -1] e não com [-1 0 1], como
% seria de se esperar. Isso porque a
% convolução espelha a máscara em x e y.
h = [1 0 -1];
d = conv(row,h,'valid');
% Apenas para os graficos
% ficarem com o mesmo numero
% de pontos no eixo x:
d = [0 d 0];
%Display
figure, image(g)
colormap(gray(256));
title('Imagem');
axis off
axis image
figure
x = 1:ncol;
plot(x,row,'-ks','LineWidth',2)
grid
legend('Linha da imagem')
figure
plot(x,d,'-bd','LineWidth',2)
grid
legend('Derivada 1a usando conv')
```



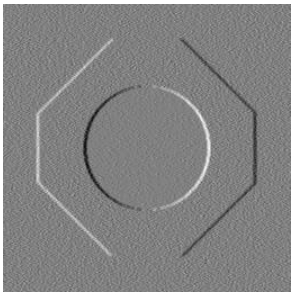
A máscara [-1 0 1] detecta apenas as bordas verticais da imagem. Para detectar as bordas horizontais podemos utilizar a mesma máscara rotacionada de 90º. O exemplo a seguir mostra a convolução de uma imagem sintética com a máscara que detecta as bordas verticais ([-1 0 1]) e com a máscara que detecta as bordas horizontais ([-1 0 1]'). Observe que cada máscara responde também aos ângulos intermediáros entre 0º e 90º em todos os quadrantes.

```
% a07teo_03 [script]
clear all, close all, clc

% Cria imagem sintética g
w = 256;
objt = 192; fundo = 64; rnd = 10;
g = makeImSynthHex(w,objt,fundo, rnd);

g = im2double(g);
hv = [-1 0 1];
gv = imfilter(g, hv, 'replicate');%correlat.
hh = hv';
gh = imfilter(g, hh, 'replicate');%correlat.

% Normaliza, pois existem
% valores negativos
gv = mat2gray(gv);
gh = mat2gray(gh);

% Display
figure, imshow(g)
title('Imagem de entrada')
figure, imshow(gv)
title('Resultado da máscara vertical')
figure, imshow(gh)
title('Resultado da máscara horizontal')
```

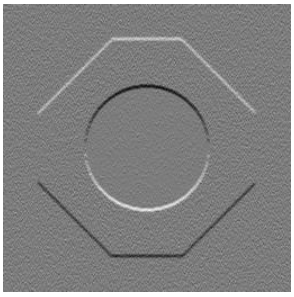
Imagem de entrada



Resultado da máscara vertical



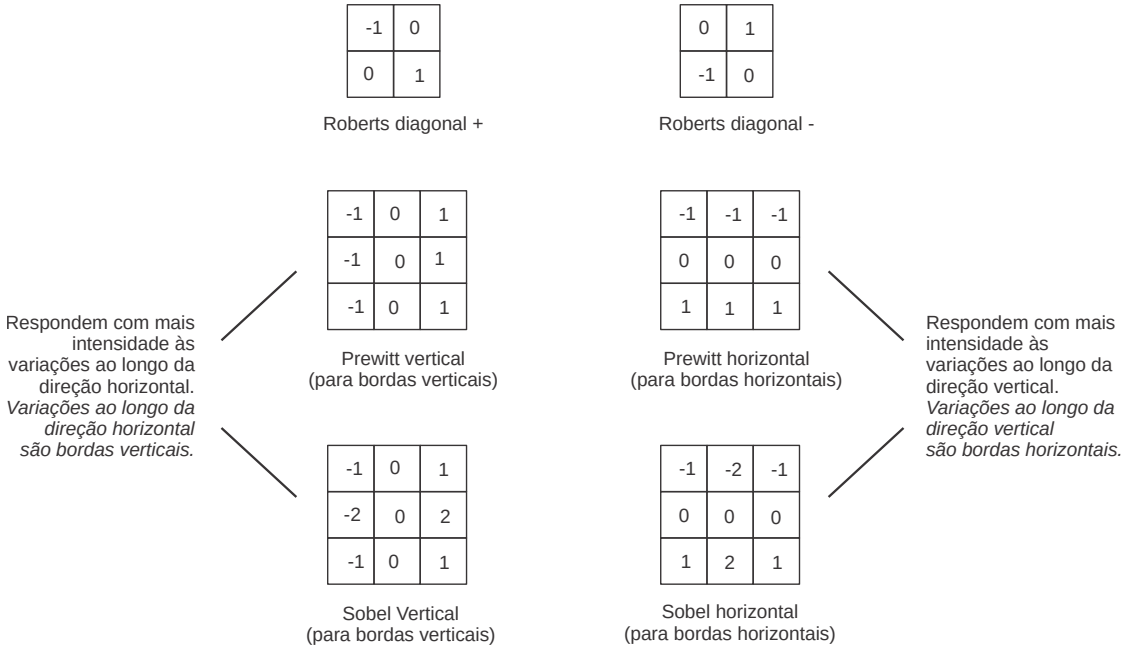
Resultado da mascara horizontal



```
function  s = makeImSynthHex(M, obj, bck, sd)
% Cria uma imagem sintética uint8 de dimensões
% M linhas e M colunas. O objeto tem nível de
% cinza obj e o fundo tem nível de cinza bck.
% O centro tem nível de cinza zero.
% Adiciona ruído Gaussiano de desvio padrão sd.

nrhs = floor(M/2);
nchs = floor(M/2);

a = triu(ones(nrhs,nchs))*bck;
b = tril(ones(nrhs,nchs),-1)*obj;
g1 = uint8(a+b);
g2 = fliplr(g1);
g3 = flipud(g2);
g4 = flipud(g1);
s = [g2 g1; g3 g4];
[r c] = size(s);
g5 = ones(r,c);
g5(nrhs/4:end-(nrhs/4),nchs/4:end-(nchs/4)) = 0;
idx = g5 == 1;
s(idx) = bck;

circ = fspecial('gaussian', [r c], r/10);
circ = circ < max(circ(:))/10;
s = s .* uint8(circ);

% Suaviza, para o degrau não ser ideal
h = fspecial('average', [3 3]);
s = imfilter(s, h, 'replicate');
% Um ruidinho de média zero e desvio padrão sd
s = imnoise(s,'gaussian',...
(0/255),(sd/255)^2);
```

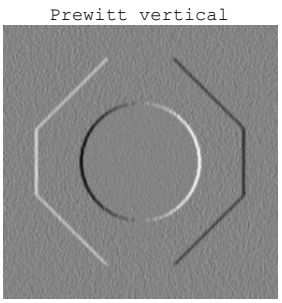
Função *makeImSynthHex*

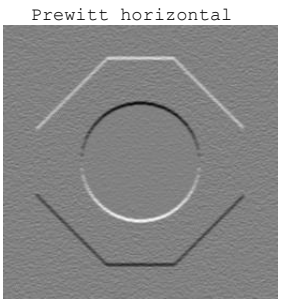
As máscaras mais conhecidas para a detecção de bordas utilizando o gradiente (primeira derivada) são variações da definição da primeira derivada para sinais discretos ([-1 0 1] e rotacionada, mostradas anteriormente). Elas são conhecidas pelos nomes dos seus autores: Roberts (1965), Prewitt (1966) e Sobel (1970) e mostradas a seguir.

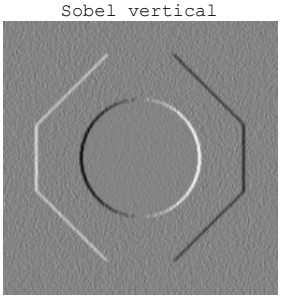


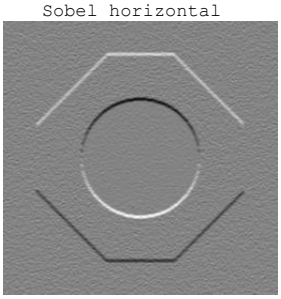
A detecção de bordas pode ser comprometida por ruído eventualmente presente na imagem. A mudança da máscara de Prewitt em relação à de Roberts tem a intenção de atenuar o ruído [[BB], Tópico 7.3.1 explicam em detalhes porque a Prewitt atenua ruído, mostrando como é a Prewitt na forma separável]. A máscara de Sobel tem o mesmo objetivo. O coeficiente central de valor 2 dá mais importância para o pixel central e portanto captura menos as variações devidas aos outros pixels e ao ruído. A máscara de Sobel é a mais popular e a mais utilizada para a detecção de bordas baseada exclusivamente no gradiente [[NA] Tópicos 4.2.2, 4.2.3 e 4.2.4].

```
% a07teo_04 [script]
clear all, close all, clc

% Cria imagem sintética g
w = 256;
objt = 192; fundo = 64; rdn = 10;
g = makeImSynthHex(w,objt,fundo,rdn);

g = im2double(g);
Ph = fspecial('prewitt');
gPh = imfilter(g,Ph,'replicate','conv');
Pv = Ph';
gPv = imfilter(g,Pv,'replicate','conv');
Sh = fspecial('sobel');
gSh = imfilter(g,Sh,'replicate','conv');
Sv = Sh';
gSv = imfilter(g,Sv,'replicate','conv');

% Normaliza, pois existem
% valores negativos
gPh = mat2gray(gPh);
gPv = mat2gray(gPv);
gSh = mat2gray(gSh);
gSv = mat2gray(gSv);

% Display
figure, imshow(g)
title('Imagem de entrada')
figure, subplot(1,2,1), imshow(gPv)
title('Prewitt vertical')
subplot(1,2,2), imshow(gPh)
title('Prewitt horizontal')
figure, subplot(1,2,1), imshow(gSv),
title('Sobel vertical')
subplot(1,2,2), imshow(gSh)
title('Sobel horizontal')
```







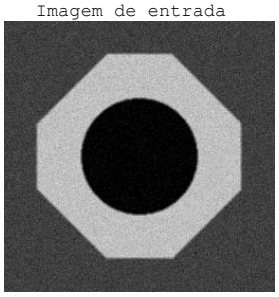


A imagem resultante da convolução da imagem de entrada com a máscara de detecção de bordas vertical é denominada *imagem do gradiente vertical ($G_v(i,j)$)*. A imagem resultante da convolução da imagem de entrada com a máscara de detecção de bordas horizontal é denominada *imagem do gradiente horizontal ($G_h(i,j)$)*. Os valores de um pixel $p \in G_v(i,j)$ e do mesmo pixel $p \in G_h(i,j)$ podem ser interpretados como as magnitudes do ponto *p* da borda da imagem original, em relação ao eixo *x* e ao eixo *y,* respectivamente. Assim, o valor final da magnitude da borda no ponto *p* é a soma vetorial de $p \in G_v(i,j)$ e $p \in G_h(i,j)$. O cálculo deste valor final de magnitude para todos os pixels dá origem à chamada *imagem de magnitude do gradiente (M(i,j)),* ou imagem de *magnitude das bordas* ou *amplitude das bordas*, conforme a equação a seguir.

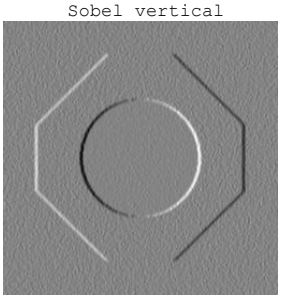
$$M(i,j) = \sqrt{G_v(i,j)^2 + G_h(i,j)^2}$$

```
% a06teo_05 [script]
clear all, close all, clc

% Cria imagem sintética g
w = 256;
objt = 192; fundo = 64; rdn = 10;
g = makeImSynthHex(w,objt,fundo,rdn);

g = im2double(g);
Sh = fspecial('sobel');
gSh = imfilter(g,Sh,'replicate','conv');
Sv = Sh';
gSv = imfilter(g,Sv,'replicate','conv');

% Imagem de magnitude do gradiente
S = sqrt(gSv.^2 + gSh.^2);
% Normaliza
gSh = mat2gray(gSh);
gSv = mat2gray(gSv);
S = mat2gray(S);

% Display
figure, imshow(g)
title('Imagem de entrada')
figure, subplot(1,2,1), imshow(gSv)
title('Sobel vertical')
subplot(1,2,2), imshow(gSh)
title('Sobel horizontal')
figure, imshow(S),
title('Magnitude do gradiente')
```

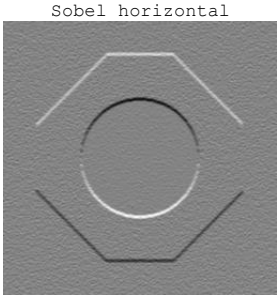
Imagem de entrada



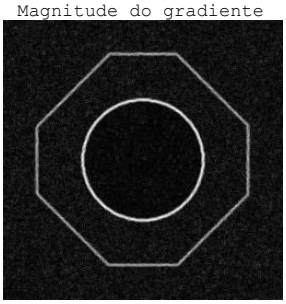
Sobel vertical



Sobel horizontal



Magnitude do gradiente

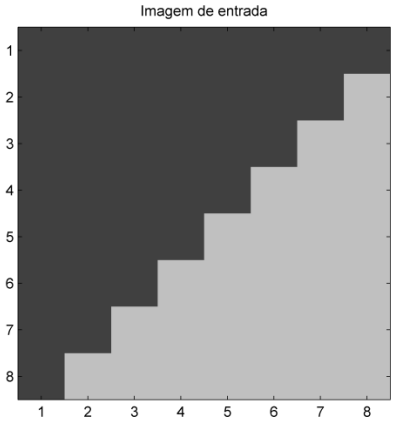


A partir das imagens $G_v(i,j)$ e $G_h(i,j)$ também é possível obter a orientação (ângulo) do vetor magnitude do gradiente, corforme a equação a seguir.

$$\theta(i,j) = tan^{-1}\left(\frac{G_h(i,j)}{G_v(i,j)}\right)$$

Este ângulo é em relação ao eixo vertical e não é o ângulo da borda, mas sim do gradiente. A borda é ortogonal à este ângulo [[GW] Tópico 10.2.5, 'The image gradient and its properties'].

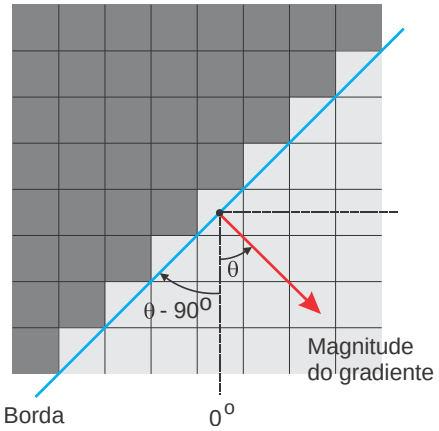
```
% a07teo_06 [script]
clear all, close all, clc

% Cria imagem sintética g
a = triu(ones(8,8))*64;
b = tril(ones(8,8),-1)*192;
g8 = fliplr(uint8(a+b));

g = im2double(g8);
Sh = fspecial('sobel');
gSh = imfilter(g,Sh,'replicate','conv');
Sv = Sh';
gSv = imfilter(g,Sv,'replicate','conv');

% Orientação do gradiente em graus.
% gSh contém as variações ao longo
% da direção vertical, isto é, ao longo
% do eixo y.
% gSv contém as variações ao longo
% da direção horizontal, isto é, ao longo
% do eixo x.
St = atand(gSh./gSv);

% Display
figure, image(g8)
colormap(gray(256));
title('Imagem de entrada');
axis image
```



Orientação

NaN	NaN	NaN	NaN	NaN	NaN	45.00	71.57
NaN	NaN	NaN	NaN	NaN	45.00	45.00	63.43
NaN	NaN	NaN	NaN	45.00	45.00	45.00	45.00
NaN	NaN	NaN	45.00	45.00	45.00	45.00	NaN
NaN	NaN	45.00	45.00	45.00	45.00	NaN	NaN
NaN	45.00	45.00	45.00	45.00	NaN	NaN	NaN
45.00	45.00	45.00	45.00	NaN	NaN	NaN	NaN
18.43	26.57	45.00	NaN	NaN	NaN	NaN	NaN

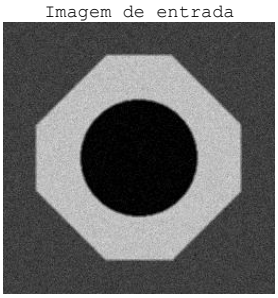
Para interpretar a orientação do gradiente obtida no código anterior, vamos observar apenas o pixel $p(5,5)$ da imagem de entrada. O valor obtido para a orientação do gradiente é 45º. (A figura a seguir foi baseada em [[GW] Figure 10.12]).

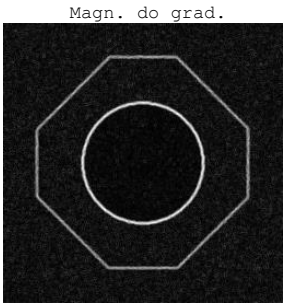


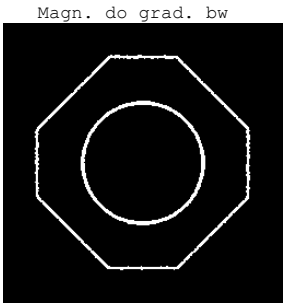
1. A referência é o eixo vertical (0°)
2. O ângulo obtido para a magnitude do gradiente é θ = 45°
3. A borda é ortogonal à magnitude do gradiente
4. O ângulo da borda é dado por θ - 90° = 45° - 90° = -45°

Como o produto final esperado de um detector de bordas é geralmente uma imagem binária na qual os pixels '1' correspondem às bordas da imagem original, deve-se limiarizar a imagem de magnitude do gradiente. Esta limiarização pode ser manual ou automática, utilizando alguma técnica adequada para a aplicação [[WP], Tópico 15.2.3].

```
% a07teo_07 [script]
clear all, close all,clc
% Cria imagem sintética g
w = 256;
objt = 192; fundo = 64; rdn = 10;
g = makeImSynthHex(w,objt,fundo,rdn);
g = im2double(g);
Sh = fspecial('sobel');
gSh = imfilter(g,Sh,'replicate','conv');
Sv = Sh';
gSv = imfilter(g,Sv,'replicate','conv');
% Imagem de magnitude do gradiente
S = sqrt(gSv.^2 + gSh.^2);
% Normaliza
gSh = mat2gray(gSh);
gSv = mat2gray(gSv);
S = mat2gray(S);
% Limiariza
Sbw = im2bw(S,graythresh(S));
% Display
figure, imshow(g)
title('Imagem de entrada')
figure, imshow(S),
title('Magn. do grad.')
figure, imshow(Sbw)
title('Magn. do grad. bw')
```







No Octave, as máscaras Prewitt e Sobel podem ser aplicadas utilizando-se as funções `fspecial` e `imfilter`, como mostrado nos exemplos anteriores. Existe também a função `edge`, bem mais completa, que já apresenta na saída uma imagem binária (classe `logical`):

Detecção de bordas com a função `edge`
[https://octave.sourceforge.io/image/function/edge.html]

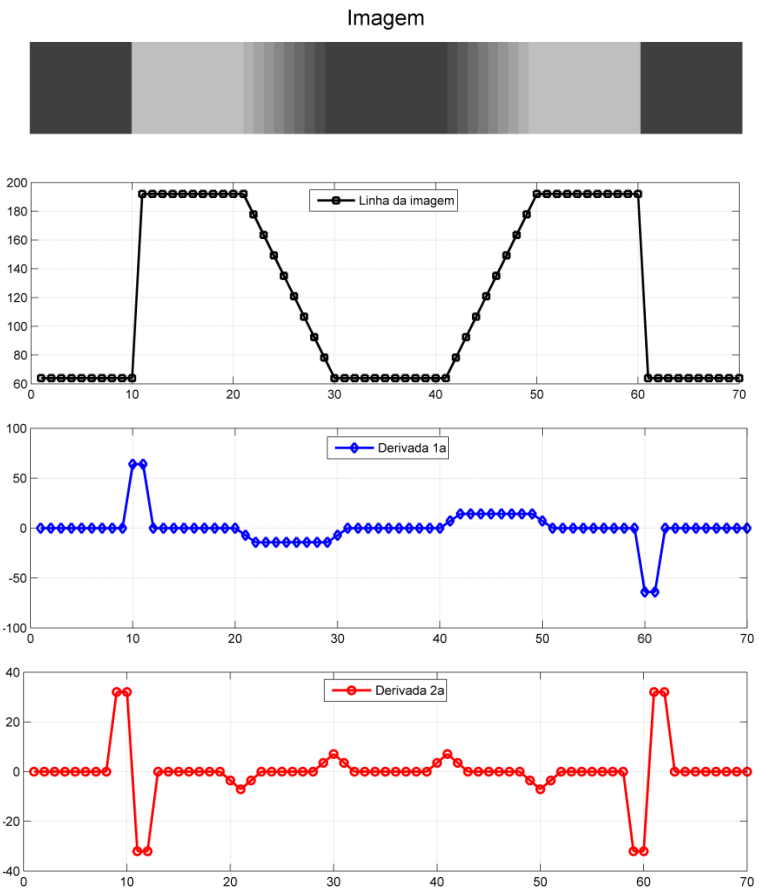
`[bw, thresh] = edge (im, method, …)`
Find edges using various methods.
The image im must be 2 dimensional and grayscale. The method must be a string with the string name. The other input arguments are dependent on method.
bw is a binary image with the identified edges. thresh is the threshold value used to identify those edges. Note that thresh is used on a filtered image and not directly on im.

Nas opções default, para os métodos Sobel, Prewitt e Roberts, a função `edge` aplica as máscaras, calcula a imagem magnitude do gradiente, limiariza automaticamente esta imagem e aplica um afinamento na imagem limiarizada, para que as bordas na imagem binária tenham largura de um pixel. Que métodos o Octave utiliza para limiarizar e afinar? Só vendo o código da função. Se for fazer isso, cuidado para não alterá-la!

7b) Detecção de bordas usando o Laplaciano do Gaussiano (LoG)

No exemplo a seguir são mostradas as derivadas de primeira e segunda ordem para uma linha da imagem sintética que contém bordas do tipo degrau (step) e rampa (ramp). Na saída da derivada de segunda ordem, as passagens por zero (zero-crossing) correspondem aos centros das bordas da imagem original. Quanto mais abrupta é a borda, maior é o valor da derivada.

```
% a07teo_08 [script]
clear all, close all,clc


g1 = ones(1,10)*64;
g2 = ones(1,10)*192;
g3 = linspace(192,64,10);
% Imagem sintética com bordas
% do tipo degrau e rampa
g = [g1 g2 g3 g1,...
     fliplr(g3) g2 g1];
g = repmat(g,9,1);
ncol = size(g,2);
% Amostra linha da imagem
row = g(1,1:ncol);
% Aloca derivada prim.
d = zeros(1,ncol);
% Aloca derivada seg.
dd = zeros(1,ncol);

% Derivada de primeira ordem
% segundo definição
for k = 2 : ncol-1
    d(k) = row(k+1) - row(k-1);
end
d = d/2;
% Derivada de segunda ordem
for k = 2 : ncol-1
    dd(k) = d(k+1) - d(k-1);
end
dd = dd/2;
%Display
figure, image(g)
colormap(gray(256));
title('Imagem');
axis off
% Relação de aspecto para
% mostrar pixels quadrados
axis image
figure
x = 1:ncol;
plot(x,row,'-ks','LineWidth',2)
grid
legend('Linha da imagem')
figure
plot(x,d,'-bd','LineWidth',2)
grid
legend('Derivada 1a')
figure
plot(x,dd,'-ro','LineWidth',2)
grid
legend('Derivada 2a')
```



Lembrando:
- A derivada de uma constante (não tem variação) é zero.
- A derivada de um degrau é um impulso.
- A derivada de uma reta (inclinação constante) é uma constante.

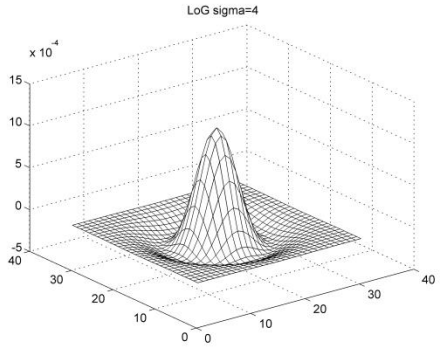
Lembrando: a derivada de segunda ordem de uma imagem é obtida utilizando-se o operador *Laplaciano* (o Laplaciano é uma máscara de convolução 3x3 que implementa uma aproximação discreta da derivada segunda). A seguir é mostrada a máscara com os coeficientes do Laplaciano [[NA] Figura 4.24].

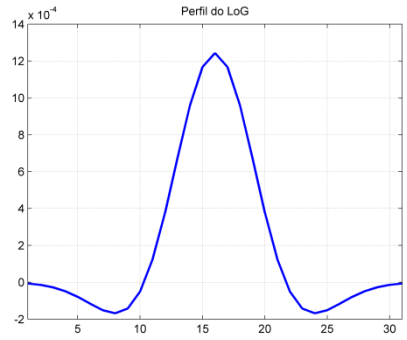
0	1	0
1	-4	1
0	1	0

Laplaciano

Porém, a derivada de segunda ordem não costuma ser utilizada diretamente para a detecção de bordas porque é muito sensível ao ruído. O operador baseado na derivada segunda utilizado para a detecção de bordas é o *Laplaciano do Gaussiano* (LoG), que incorpora no operador Laplaciano (derivada segunda) uma suavização através de uma função Gaussiana. Em outras palavras, o LoG pode ser entendido como um Gaussiano (passa-baixas) associado a um Laplaciano, isto é, realiza uma suavização *e* a derivada segunda. Lembrando: o parâmetro que regula a 'abertura' da Gaussiana é o σ. Quanto maior o σ, maior a suavização. Isso vale também para a função LoG. Portanto, valores altos de σ resultam em uma detecção de bordas menos susceptível ao ruído, e consequentemente também menos sensível às potenciais bordas (é um compromisso [trade-off]).

```
% a07teo_09 [script]
clear all, close all,clc

% Gera um LoG de sigma=4 em uma janela 31x31
% É 31x31 pois o objetivo é apenas visualizar
% uma LoG e não convoluir com uma imagem
logf = fspecial('log', [31 31], 4);
p = logf(16,:); %linha central

%Display
figure
mesh(-logf, 'EdgeColor', 'black')
title('LoG sigma=4')
figure
plot(1:31,-p,'LineWidth',2)
xlim([1 31])
grid
title('Perfil do LoG')
```



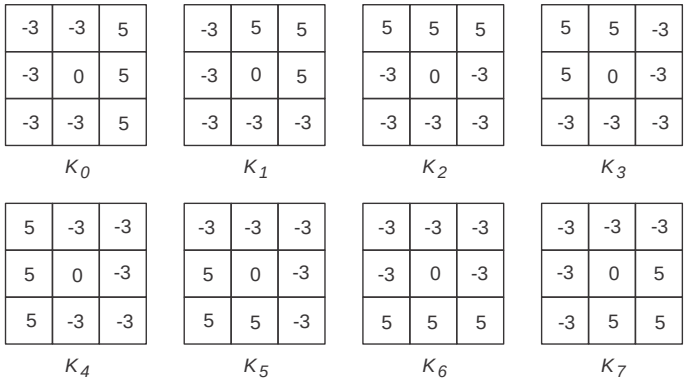


Detecção de bordas com a função edge**, método LoG**
[https://octave.sourceforge.io/image/function/edge.html]

[bw, thresh] = edge (im, "LoG", thresh, sigma)
Find edges by convolving with the Laplacian of Gaussian (LoG) filter, and finding zero crossings.
Only zero crossings where the filter response is larger than thresh are retained. thresh is automatically computed as 0.75*M, where M is the mean of absolute value of LoG filter response.
sigma sets the spread of the LoG filter. Default to 2.

7c) Detecção de bordas por compass operator

Outro método de detecção de bordas que também utiliza máscaras de convolução é o denominado *compass operator* [[BB] Tópico 7.7.3]. O objetivo das *compass masks* [OM Tópico 14.3] é capturar apenas variações de intensidade nos pixels em direções específicas. Como são mais discriminativas, a idéia é que proporcionem uma detecção de bordas menos sensível ao ruído [[BB] Tópico 7.3.3]. Um dos conjuntos de compass masks mais difundidos é o de Kirsh, mostrado a seguir.



A saída do detector de bordas de Kirsh é o máximo das saídas das oito máscaras. A orientação é a da própria máscara que forneceu o máximo (múltiplos de 45º).

Na prática, o método de Kirsh fornece resultados bastante parecidos aos dos filtros de gradiente. Por isso, para aplicações em geral, o método de Sobel é mais indicado que os baseados em compass masks [[BB] Tópico 7.3.7, [JRP] Tópico 'Template-based Edge Detection'].

7d) Detector de bordas Canny

Extraído de [[OM] Tópico 14.5]:
"
The Canny edge detector [JC] is one of the most popular, powerful, and effective edge detection operators available today. Its algorithm can be described as follows:

1. The input image is smoothed using a Gaussian lowpass filter (LPF) (Section 10.3.3), with a specified value of σ: large values of σ will suppress much of the noise at the expense of weakening potentially relevant edges.

2. The local gradient (intensity and direction) is computed for each point in the smoothed image.

3. The edge points at the output of step 2 result in wide ridges*. The algorithm thins those ridges, leaving only the pixels at the top of the ridge, in a process known as *non-maximal suppression*.

4. The ridge pixels are then thresholded using two thresholds, T_{low} and T_{high}: ridge pixels with value greater than T_{high} are considered *strong* edge pixels; ridge pixels with values between T_{low} and T_{high} are said to be *weak* pixels. This process is known as *hysteresis thresholding*.

5. The algorithm performs edge linking, aggregating weak pixels that are 8-connected to the strong pixels.
"

*ridge: cume (parte mais alta de uma montanha).

Em [JRP], capítulo 1, tem explicação bem detalhada e código C do Canny. Em [GW], capítulo 10 seção 10.2.6 também tem explicação detalhada.

Referências

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

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

[BB] Wilhelm Burger, Mark Burge, Digital image processing – an algorithmic introduction using Java, Springer, 2010.

[AM] Alasdair McAndrew, An Introduction to Digital Image Processing with Matlab, Notes for SCM2511 Image Processing 1, September 2004, School of Computer Science and Mathematics, Victoria University of Technology. Disponível em http://visl.technion.ac.il/labs/anat/An%20Introduction%20To%20Digital%20Image%20Processing%20With%20Matlab.pdf. Provavelmente é uma versão draft do livro http://www.amazon.com/Introduction-Digital-Image-Processing-MATLAB/dp/0534400116

[WP] William K. Pratt, Digital Image Processing: PIKS Inside, 3rd Ed, Wiley, 2001.

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

[JRP] J. R. Parker, Algorithms for image processing and computer vision, Wiley, 1997.

[JC] John Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[AB] Al Bovik, The Essential Guide to Image Processing, Academic Press, 2009.