PRÁTICA 5

Prof. Dr. Evandro Luis Linhari Rodrigues

Além da solução usando Matlab, todos os exercícios deverão ser implementados também usando Python e OpenCV.

Segmentação de Imagens – Parte 1.

1) Detecção de Pontos isolados.

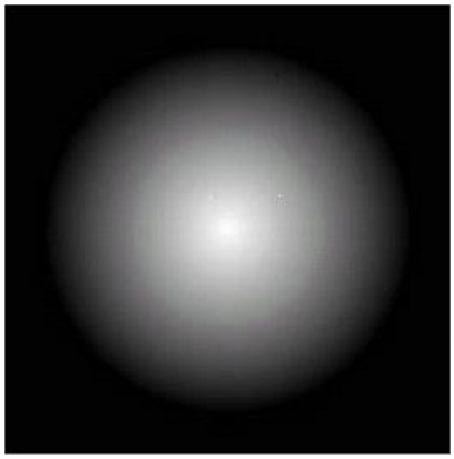


Figura 1 - points.tif

A imagem da Figura 1 apresenta 3 pontos brancos isolados quase imperceptíveis que podem ser detectados com um filtro do tipo:

−1	−1	−1
−1	8	−1
−1	−1	−1

T_1: Detectar os pontos isolados da Imagem da Figura 1:

```
f = imread('point.tif');
fg = rgb2gray(f)
w = [-1 -1 -1; -1  8 -1; -1 -1 -1];
g = imfilter(fg,w);
BW = im2bw(g);
imview(fg)
imview(BW)
```

2) Detecção de linhas.

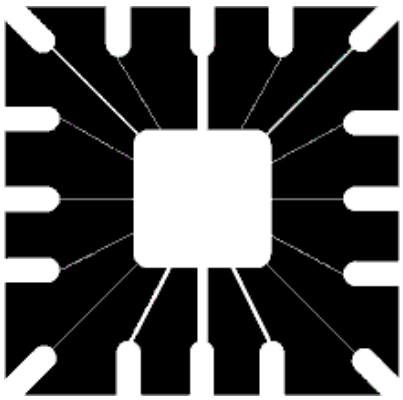


Figura 2 - wirebond_mask.tif

Linhas Verticais, Horizontais ou em ±45º podem ser detectadas através da convolução da imagem com templates do tipo:

−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
Horizontal			+45°			Vertical			−45°		

T_2: Detectar linhas em –45º da Imagem da Figura 2:

```
f1 = imread('wirebond_mask.tif');
w = [ 2 -1 -1; -1  2 -1; -1 -1  2];
g1 = imfilter(double(f1),w);
T = max(g1(:));
g1 = g1>=T-40
imview(f1, [])
imview(g1)
```

E_1: Variar o valor do Threshold e observar o que acontece com as linhas detectadas. Detectar linhas em 45º, Verticais e Horizontais da Imagem. Em cada caso variar o Threshold e concluir a respeito.

3) Detecção de Bordas.

O Toolbox de Processamento de Imagens do MatLab possui uma função que realiza a detecção de bordas. A sintaxe desta função é:

[g, t] = edge(f, 'method', parameters)

Onde ***f*** é a Imagem de entrada, ***g*** é a Imagem de saída (arranjo lógico com 1 nas posições de bordas detectadas e 0 nas demais) e ***'method'*** corresponde ao tipo de detector utilizado, conforme mostra a Tabela 1. O valor de ***t*** é opcional e corresponde ao Threshold que a função utilizará para definir uma borda.

Tabela 1- Detectores de Borda da função *edge*

Edge Detector	Basic Properties
Sobel	Finds edges using the Sobel approximation to the derivatives shown in Fig. 10.5(b).
Prewitt	Finds edges using the Prewitt approximation to the derivatives shown in Fig. 10.5(c).
Roberts	Finds edges using the Roberts approximation to the derivatives shown in Fig. 10.5(d).
Laplacian of a Gaussian (LoG)	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a Gaussian filter.
Zero crossings	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a user-specified filter.
Canny	Finds edges by looking for local maxima of the gradient of $f(x, y)$. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. Therefore, this method is more likely to detect true weak edges.

A sintaxe para o Detector de **Sobel** usando a função ***edge*** é:

[g, t] = edge(f, 'sobel', T, dir)

onde ***T*** é um Threshold especificado e ***dir*** dá a direção preferencial das bordas detectadas (***vertical, horizontal, ou both*** (default)). Se ***T*** é especificado então ***t = T***. Se T não for especificado, a função ***edge*** usa um Threshold automático e retorna seu valor em ***t***.

T_3: Aplicar o Detector de Bordas de Sobel na imagem da Figura 3 e obter o valor de *t*.

```
f2 = imread('igreja.tif');
[g2,t]=edge(f2,'sobel');
imview(f2), imview(g2)
```

Figura 3 - igreja.tif

E_2: A partir do valor de *t* obtido, variar o valor do Threshold e observar o que acontece com as bordas detectadas.

E_3: Comparar o uso da função *edge* com a função *imfilter*.

As sintaxes da função ***edge***, para os Detectores de **Roberts** e de **Prewitt** , são idênticas à sintaxe do Detector de **Sobel.**

E_4: Repetir o exercício E_2, agora com os detectores de Roberts e de Prewitt. Comparar os resultados com o Detector de Sobel.

O detector de bordas **LoG** (Laplaciano da Gaussiana) tem a seguinte sintaxe usando a função ***edge***:

[g, t] = edge(f, 'log', T, sigma)

onde ***sigma*** é o desvio padrão relacionado à suavização realizada pela Gaussiana. O valor default para ***sigma*** é 2. ***T*** é um Threshold especificado. Fazer ***T = 0*** produz bordas em contornos fechados, característica da metodologia LoG.

E_5: Repetir o exercício E_2, agora com o detector de bordas LoG. Variar também o valor de *sigma*. Comparar os resultados com os detectores anteriores.

A sintaxe para o Detector de Bordas de **Canny** através da função ***edge***, é:

[g, t] = edge(f, 'canny', T, sigma)

onde ***T*** é um vetor ***T = [T1, T2]*** contendo os dois valores de Threshold e ***sigma*** é o desvio padrão do filtro de suavização. O default para ***sigma*** é 1.

E_6: Repetir o exercício E_2, agora com o detector de bordas de Canny. Obter os valores de T1 e T2 gerados pela função. Variar os valores de Threshold respeitando-se T1<T2.Variar também o valor de *sigma*. Analisar e comparar os resultados com os detectores anteriores.

4) Transformada de Hough (HT).

Não existe no Toolbox de Processamento de Imagens do MatLab uma função que implemente diretamente a HT. O livro "*Gonzalez, R.C.; Woods,R.E.; Eddins,S.L. Digital Image Processing Using MATLAB*" publicou funções para realizar a HT para retas, facilitando inclusive a localização dos segmentos de retas nas imagens gradiente. As funções ***hough.p, houghpeaks.p, houghpixels.p*** e ***houghlines.p*** devem ser copiadas para seu diretório de trabalho.

A função ***hough*** tem a seguinte sintaxe:

[H, theta, rho] = hough(f, dtheta, drho)

Computa a HT da imagem ***f*** gerando o Arranjo Acumulador ***H*** com espaçamento entre as células dado por ***dtheta*** e ***drho***. Se omitidos ***dtheta = 1*** e ***drho = 1***.

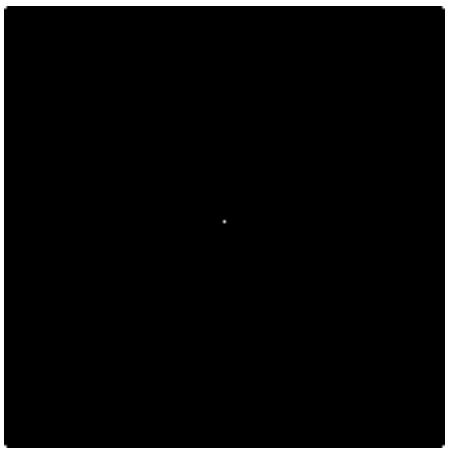


Figura 4 - pontos.tif

T_4: Calcular a HT na imagem da Figura 4 obtendo as senóides geradas no plano de parâmetros.

```
f2 = imread('pontos.tif');
figure, imshow(f2)
[H, theta, rho] = hough(f2);
imshow(theta, rho, H,  'notruesize')
axis on, axis normal
xlabel('\theta'), ylabel('\rho')
```

A função que detecta os picos no Arranjo Acumulador gerado pela HT é a ***houghpeaks*** que tem a seguinte sintaxe:

[r, c, hnew] = houghpeaks(H, numpeaks, threshold, nhood)

que detecta picos na matriz ***H*** (Arranjo Acumulador) . ***numpeaks*** especifica o máximo número de localização de picos (o default é 1). Valores de ***H*** abaixo do ***threshold*** não serão considerados picos. ***nhood*** é um vetor de 2 elementos especificando o tamanho da vizinhança de supressão, ou seja, o tamanho da vizinhança do pico encontrado que será zerada após a localização de cada pico.
hnew é o Arranjo Acumulador com as vizinhanças de pico suprimidas.
r e ***c*** são as coordenadas de linha(***r***) e coluna(***c***) dos picos identificados.

T_5: Marcar os picos do Arranjo Acumulador da HT realizada na imagem da Figura 4.

```
f2 = imread('pontos.tif');
figure, imshow(f2)
[H, theta, rho] = hough(f2);
imshow(theta, rho, H,  'notruesize')
axis on, axis normal
xlabel('\theta'), ylabel('\rho')
[r, c] = houghpeaks(H,6,3);
hold on
plot(theta(c), rho(r), 'linestyle','none','marker','s','color','b')
```

Como a HT determina as retas que passam pelos pontos determinados no Arranjo Acumulador, duas funções são utilizadas para calcular os segmentos de retas entre os pontos. Estas funções são a ***hougpixels*** e a ***houghlines.***

A sintaxe da função ***houghlines*** (que usa a ***houghpixels***) é:

lines = houghlines(f, theta, rho, rr, cc, fillgap, minlength)

onde ***f*** é a imagem gradiente original, ***theta*** e ***rho*** são os vetores retornados pela função ***hough***. ***rr*** e ***cc*** são vetores que especificam as linhas e colunas do Arranjo Acumulador para se procurar por segmentos de linhas. ***fillgap*** é o intervalo entre as células do Arranjo Acumulador associados com a mesma reta e serão fundidas em um mesmo segmento de reta (default é 20). Segmentos de linhas fundidos menores do que ***minlength*** (default = 40 pixels) são descartados.

lines é um arranjo de estrutura cujo comprimento é igual ao número de segmentos fundidos encontrados. Cada elemento da estrutura tem os seguintes campos:

point1 → Ponto-final do segmento de linha; vetor de dois elementos
point2 → Ponto-final do segmento de linha; vetor de dois elementos
length → Distância entre o point1 e o point2

theta → Ângulo em graus da célula da HT
rho → Posição de rho na célula da HT

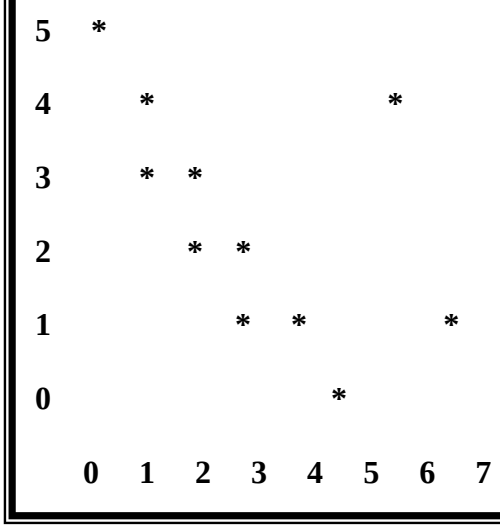
T_6: Detectar os segmentos de retas mais significativos na imagem da Figura 5 através da Transformada de Hough (HT) implementadas pelas funções fornecidas.

```
f2 = imread('madeira.tif');
g2=edge(f2,'canny',[0.06 0.5], 1.5);
imview(g2)
[H, theta, rho] = hough(g2);
imshow(theta, rho, H,  'notruesize')
axis on, axis normal
xlabel('\theta'), ylabel('\rho')
[r, c] = houghpeaks(H,10,10);
hold on
plot(theta(c), rho(r), 'linestyle','none','marker','s','color','b')
lines = houghlines(g2, theta, rho, r, c);
 figure,imshow(f2)
hold on
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,2), xy(:,1), 'LineWidth', 2,'color', 'b');
end
```

Figura 5 - madeira.tif

E_7: Aplicar a HT na imagem da Figura 3 e variar os parâmetros, tanto do detector de bordas como da HT, para conseguir o maior número de linhas significativas possível (segmentos). Concluir a respeito da HT para retas.

E_8: Realize a Transformada de Hough no arranjo de pixels abaixo usando o espaço de parâmetros (m,c) e (ρ,θ) . (Não é necessário usar o MatLab)



E_9: (Não é necessário usar o Matlab)
Dados os seguintes pontos de uma imagem:
(280,252) , (233,249) , (165,248) , (325,254) , (394,257)
(283,147) , (281,212) , (280,252) , (278,290) , (278,357)
usar a Transformada de Hough com:

$$\Delta\rho = \frac{512.\sqrt{2}}{200}$$

$$\Delta\theta = \frac{2.\pi}{200}$$

Determinar:
a) O número de segmentos de linha compreendidos por estes pontos
b) Os parâmetros destas linhas.
c) Mostrar como determinar a interseção de duas linhas quaisquer, através do exemplo dado.