Trab 3 - Octave

Anny Caroline Correa Chagas
Ciência da Computação, UERJ

21 de Julho de 2019

Exercício 1

1. Calcule a magnitude e o ângulo do gradiente para cada pixel de uma imagem qualquer, utilizando o filtro de Sobel.
2. Se a imagem escolhida for uma imagem colorida, transforme-a em monocromática antes de calcular o gradiente.
3. Limiarize a imagem da magnitude, para mostrar apenas pontos de borda "fortes". Defina o limiar empiricamente.
4. Crie uma imagem indexada, de forma a colorir cada ponto de borda de acordo com a direção da borda:
   (a) Amarelo: $\pi/6 >$ ângulo $> -\pi/6$
   (b) Verde: $\pi/3 >$ ângulo $< \pi/6$
   (c) Vermelho: $-\pi/6 >$ ângulo $> -\pi/3$
   (d) Azul: $-\pi/3 >$ ângulo $> \pi/3$
5. Mostre em figuras diferentes a imagem original, a imagem com a magnitude do gradiente e a imagem indexada dos ângulos do gradiente.

Resposta:

1. Primeiro lemos a imagem e a transformamos em escala de cinza e em `double`. Na linha 4 foi aplicado um filtro gaussiano para suavizar a imagem e melhorar os resultados.

```
1 FERRARI = imread("Ferrari.jpg");
2 FERRARI = rgb2gray(FERRARI);
3 FERRARI = im2double(FERRARI);
4 FERRARI = imfilter(FERRARI, fspecial('gaussian',[5 5]));
```

2. Depois se calculou as componentes x e y do gradiente e a magnitude. Para definir as bordas foi usado um `LIMIAR=0.3`.

```
1  LIMIAR = 0.3;
2  SOBEL_X = [-1 0 1; -2 0 2; -1 0 1];
3  SOBEL_Y = [-1 -2 -1; 0 0 0; 1 2 1];
4
5  magx = imfilter(FERRARI, SOBEL_X);
6  magy = imfilter(FERRARI, SOBEL_Y);
7
8  #calcula magnitude
9  mag = abs(magx) + abs(magy);
10
11 #liamiariza magnitude
12 imag = mag >= LIMIAR;
```

3. O resultado da magnitude pode ser visto na Figura 1

```
figure;
subplot(2,2,1);
imshow(magx);
subplot(2,2,2);
imshow(magy);
subplot(2,2,3);
imshow(mag);
subplot(2,2,4);
imshow(imag);
```

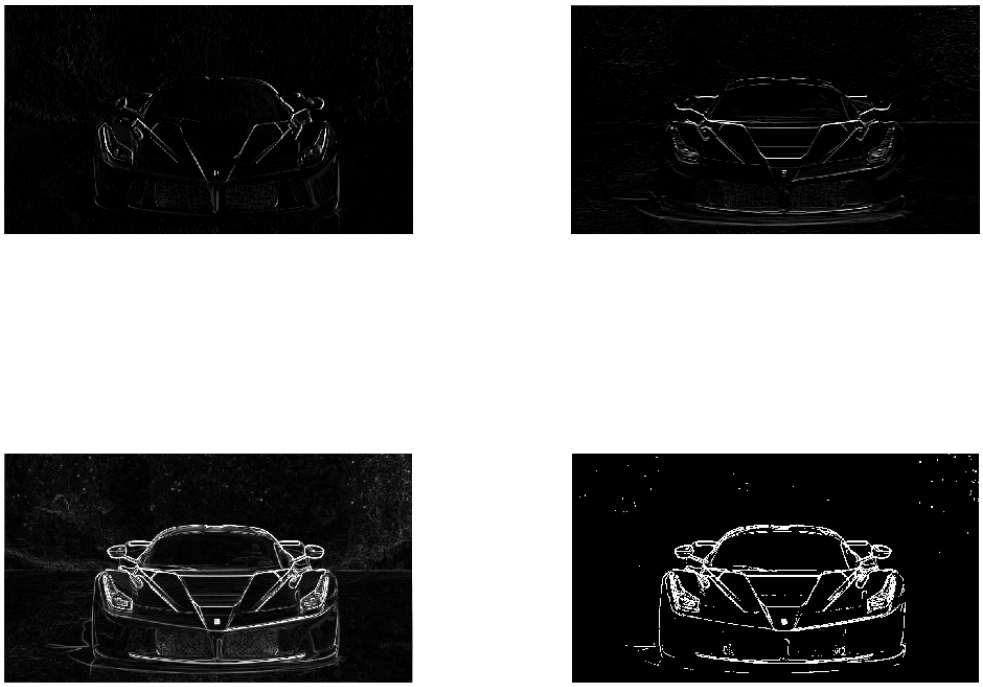

Figura 1: Componente x, componente y, magnitude e magnitude limiarizada

4. Depois foi criada uma matriz de ângulos e uma matriz zerada do mesmo tamanho da matriz de ângulos.

```
#calcula ângulos
ang = atan(magy ./ magx); #definido entre -pi/2 e pi/2

#cria matriz vazia do tamanho da matriz "ang"
iang = zeros(size(ang,1),size(ang,2));
```

5. Essa matriz zerada terá alguns de seus valores setados com base no ângulo que representam.

```
#Aplica cor
iang(find(ang < (pi/6) & ang >-(pi/6))) = 2; # Amarelo: 0 graus
iang(find(ang < (pi/3) & ang > (pi/6))) = 3; # Verde: 45 graus
iang(find(ang <-(pi/6) & ang >-(pi/3))) = 5; # Vermelho: 135 graus
iang(find(ang > (pi/3))) = 4; # Azul: 90 graus
iang(find(ang <-(pi/3))) = 4; # Azul: -270 graus
```

6. Cada um desses valores (entre 2 e 4) representa uma cor. Todos os valores não setados continuam com o valor 0. Por padrão, o valor zero em uma matriz da imagem indexada será renderizado com a cor da primeira posição do mapa de cores, por isso nosso `COLOR_MAP` começa com a cor preto.

```
COLOR_MAP = [0 0 0; 1 1 0; 0 1 0; 0 0 1; 1 0 0];
```

7. O resultado dessa matriz `iang` pode ser conferido na Figura 2

```
figure;
imshow(iang, COLOR_MAP);
```

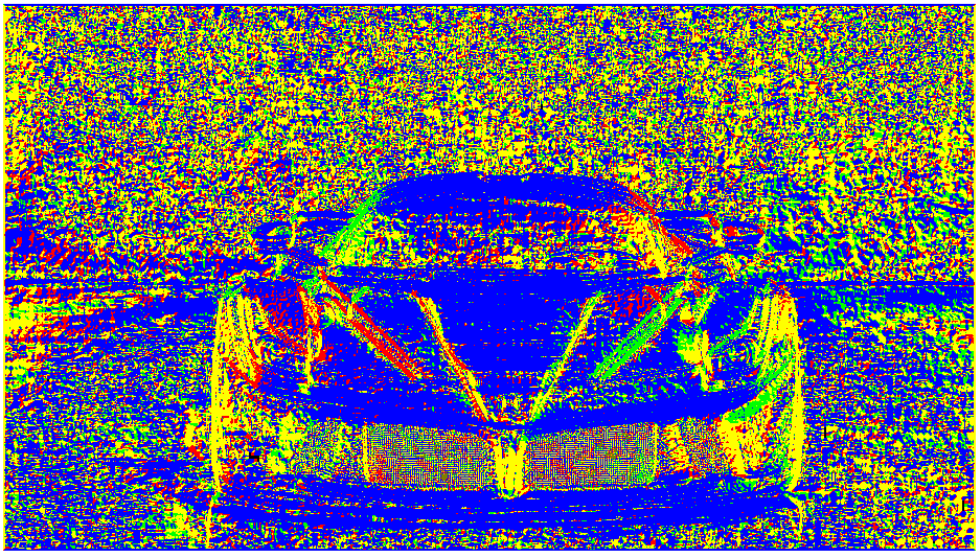

Figura 2: iang - matriz de ângulos colorida

8. Por fim, basta “apagar” os pixels da matriz `iang` que não façam parte da borda.

```
# Pintar de preto todos os pixels que estão fora da borda (definida pelo limiar)
iang(find(mag < LIMIAR)) = 0;
figure;
imshow(iang, COLOR_MAP);
```

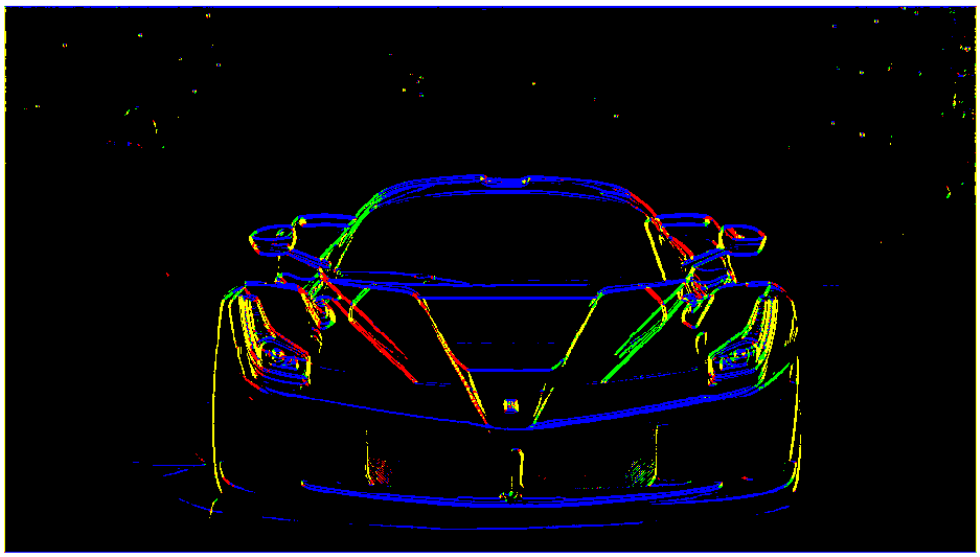

Figura 3: Imagem final

Exercício 2

Detecte as bordas em uma imagem monocromática usando as várias possibilidades da função `edge` do Octave. Discuta os resultados.

1. **Carregar a imagem original:**

```
IMG = imread("Woman.png");
IMG = rgb2gray(IMG);
IMG = im2double(IMG);
figure,imshow(IMG);
```

Figura 4: Imagem original

2. **Canny:**

```
IMG_CANNY = edge(IMG, "Canny");
figure,imshow(IMG_CANNY);
```

3. **Lindeberg:**

```
IMG_LINDEBERG = edge(IMG, "Lindeberg");
figure,imshow(IMG_LINDEBERG);
```

4. **Sobel:**

```
IMG_SOBEL = edge(IMG, "Sobel");
figure,imshow(IMG_SOBEL);
```

5. **Andy:**

```
IMG_ANDY = edge(IMG, "Andy");
figure,imshow(IMG_ANDY);
```

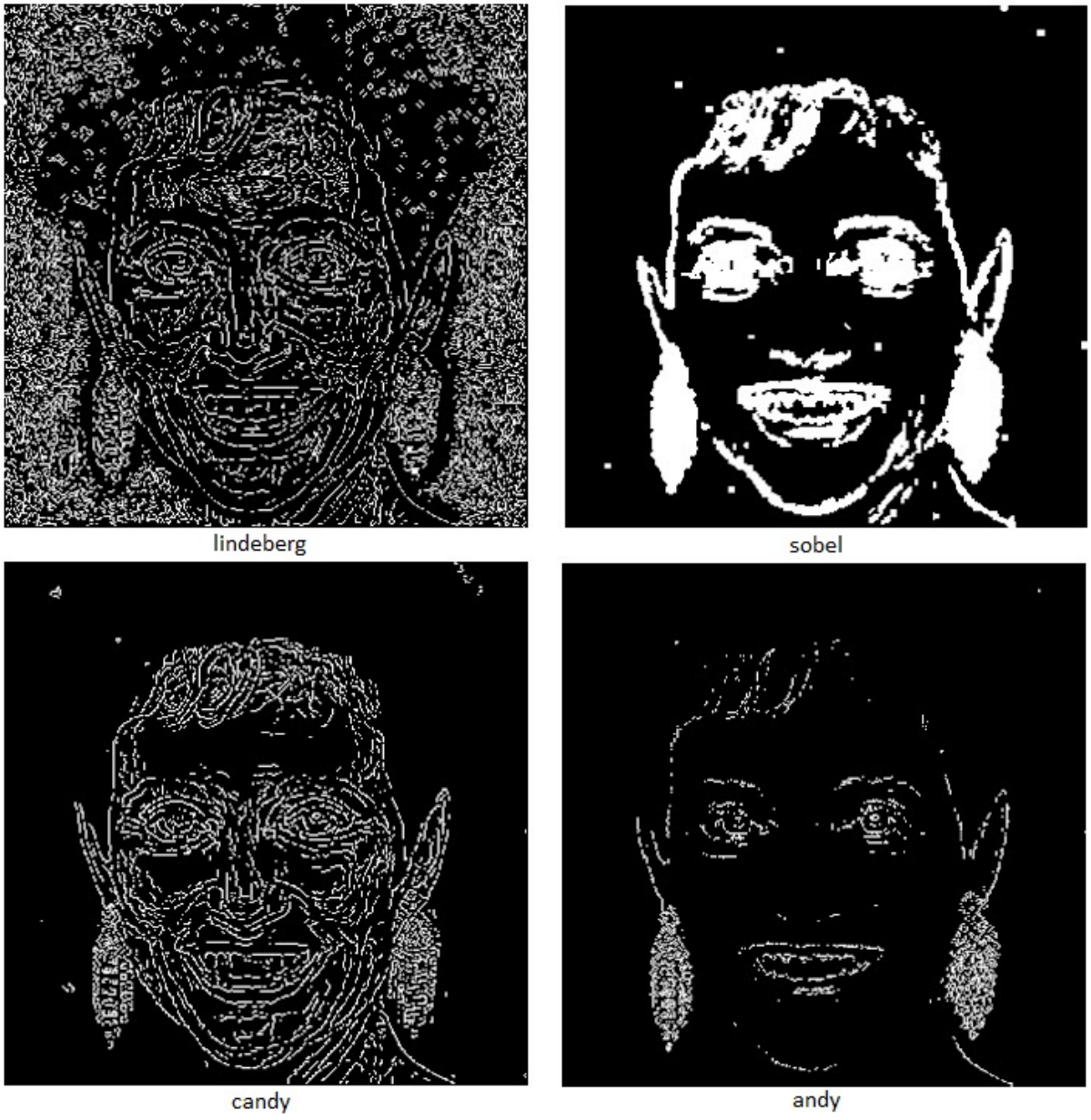

Figura 5: Resultado

Como pode ser visto na Figura 5, o método `andy` é o menos sensível as bordas. Ele obtém somente as bordas mais marcantes do rosto da mulher, como contorno dos olhos, boca e brincos. Nesse ponto ele é muito parecido com o método `sobel`, mas esse último demarca bem mais esses contornos. O método `candy` já obtém muitos mais contornos: a franja já é mais marcada, detalhes do nariz e da pele da mulher, mas assim como nos outros métodos o fundo ainda não é levado em consideração, somente alguns ruídos são marcados. Por fim, o *lindeberg* é o que apresenta mais bordas marcadas.

Exercício 3

1. Baixe do AVA as bandas de uma imagem de sensoriamento remoto: `laranjeiras_X.tif` ou `gavea_X.png`. O sufixo no nome do arquivo identifica a banda: azul (X=b); verde (X=g); vermelha(X=r); infra-vermelha (X=nir).
2. Crie uma imagem cujos pixels equivalem ao NDVI (vide notas de aula). Dica: transforme as imagens das bandas para double antes de calcular o NDVI.
3. Limiarize a imagem NDVI para selecionar os pixels com vegetação, criando uma imagem binária. Defina o limiar empiricamente.
4. Mostre em figuras diferentes a imagem original (composição colorida das bandas r, g, b), uma imagem só com a vegetação (com os outros pixels pretos) e uma imagem só com as áreas que não tem vegetação.

```
#separar vegetação do que não é vegetação usando ndvi

#exibindo a img rgb
gr = imread("imgs/gavea_b.png");
gg = imread("imgs/gavea_g.png");
gb = imread("imgs/gavea_b.png");
gnir = imread("imgs/gavea_nir.png");

gr = rgb2gray(gr);
gg = rgb2gray(gg);
gb = rgb2gray(gb);
gnir = rgb2gray(gnir);

grgb = cat(3,gr,gg,gb);

figure;
imshow(grgb);
title("Imagem RGB");

#Calculando o NDVI
gr = im2double(gr);
gg = im2double(gg);
gb = im2double(gb);
gnir = im2double(gnir);

NDVI = (gnir - gr) ./ (gnir + gr);

figure;
imshow(NDVI);
title("Imagem NDVI");

#Montando a img de vegetação
VEG = NDVI > 0.1;
figure;
imshow(VEG);
title("Imagem VEG");

#grafico
PRETO = VEG*0;
grgb = cat(3, PRETO, VEG, PRETO);
figure;

```

```
subplot(2,2,1);

subplot(2,2,3);
imshow(grgb);

vegetacao = {'Vegetação', ''}
quantidadeDePixels = [sum(sum(VEG)), sum(sum(~VEG))];

subplot(2,2,4);

p = pie(quantidadeDePixels, vegetacao);
set(p(1),'facecolor',[0 1 0]);
set(p(3),'facecolor',[0 0 0]);
```

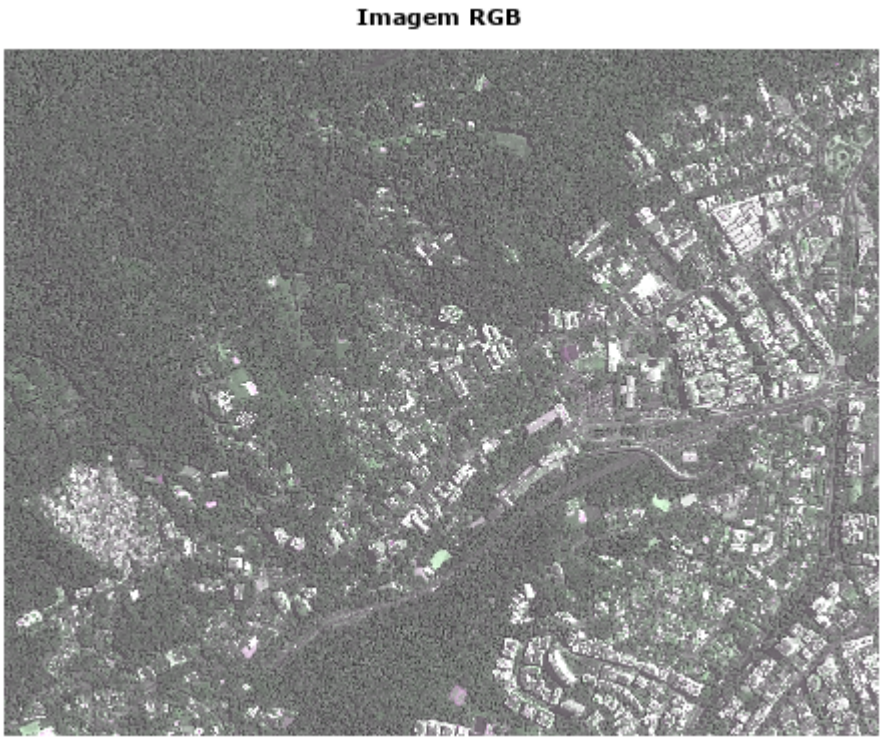

Figura 6: Imagem RGB

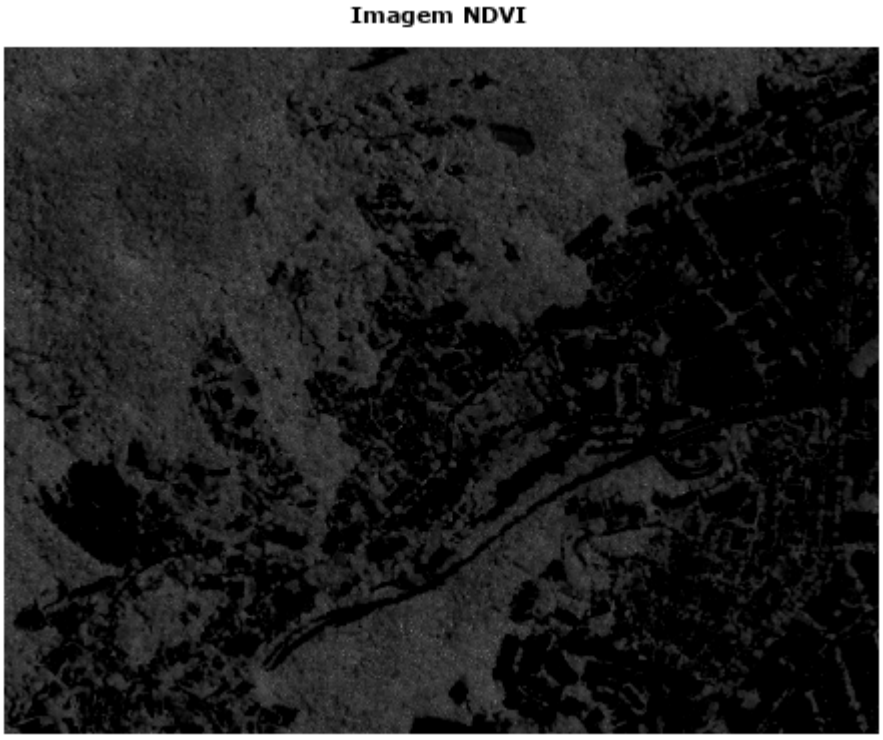

Figura 7: Imagem NDVI

Figura 8: Imagem VEG

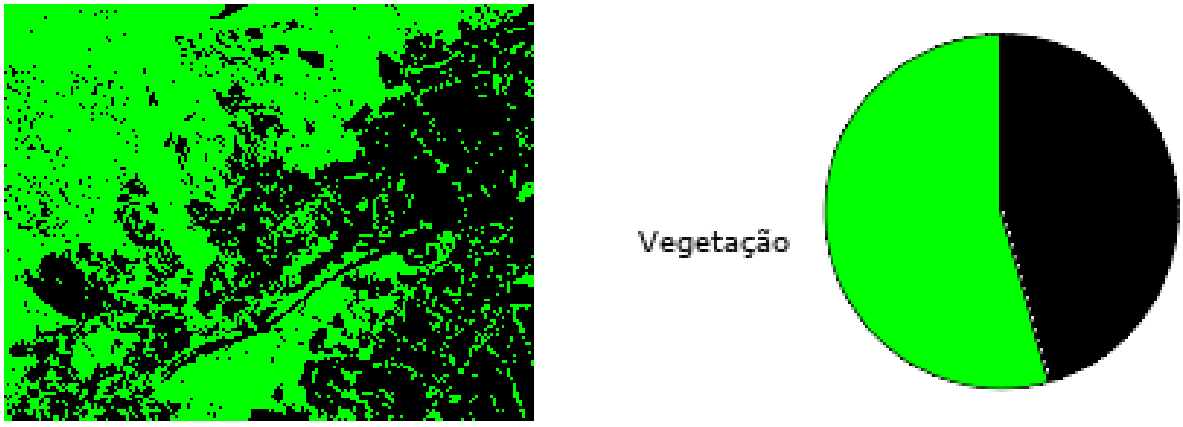

Figura 9: Porcentagem de vegetação numa imagem

Exercício 4

1. Baixe do AVA a imagem de uma digital.
2. Calcule um limiar global, para separar a digital do fundo da imagem, através do algoritmo descrito nas notas de aula. Dica: use os comandos `find` e `size` para calcular as médias das intensidades dos pixels acima e abaixo do limiar.

```
IMG = imread("imgs/digital.png");
IMG = rgb2gray(IMG);
IMG = im2double(IMG);

#Selecione um limiar inicial T
T = 0.1;

do
  #{
  2. Limiarize a imagem usando T:  Vai gerar dois grupos de pixels:
  G1(pixels com valores < T ) e G2(pixels com valores >= T )
  #}
  G1 = IMG(find(IMG <  T));
  G2 = IMG(find(IMG >= T));

  #3. Calcule a intensidade media de cada grupo (u1 e u2)
  U1 = mean(G1);
  U2 = mean(G2);

  #4. Calcule um novo limiar:
  Tant = T;
  T = (1/2) * (U1+U2);
until(T == Tant)


subplot(2,1,2);
imhist(IMG);
legend(cstrcat("Limiar: ", num2str(T)));
subplot(2,2,1);
imshow(IMG);
subplot(2,2,2);
imshow(IMG>T);
```

3. Apresente numa mesma figura a imagem original, o histograma da imagem original (identificando o valor do limiar em uma legenda), e a imagem com o fundo em branco e os pixels da digital pretos.

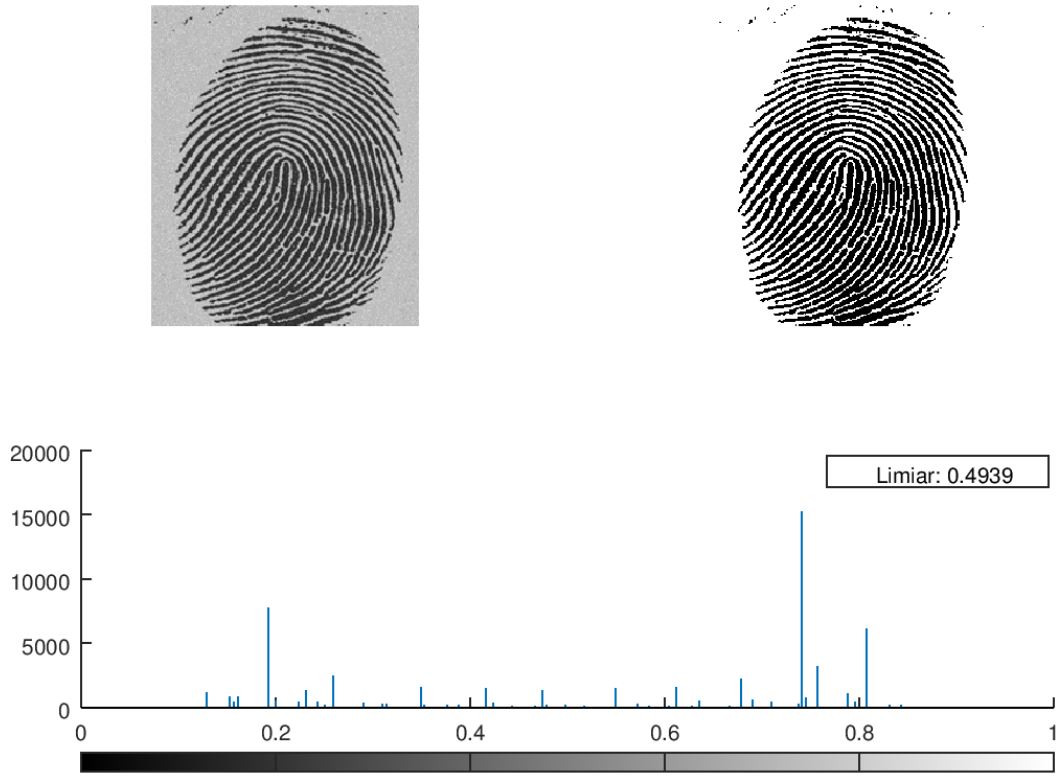

Figura 10: Imagem original, histograma da imagem original e a imagem resultante da limiarização