

Rio de Janeiro, 29 de setembro de 2017

Universidade do Estado do Rio de Janeiro

Instituto de Matemática e Estatística

**Computação Gráfica**

**Trabalho 3 – Segmentação de Imagens Digitais**

**Aluno:** Leonardo Lima Marinho

**Matrícula:** 2014.1.00506.11

**Exercício 1:**

**Calcule a magnitude e o ângulo do gradiente para cada pixel de uma imagem qualquer, utilizando o filtro de Sobel.**

**• Se a imagem escolhida for uma imagem colorida, transforme-a em  
monocromática antes de calcular o gradiente. Além disso, transforme a  
imagem monocromática para *double.***

>> sobel\_x = [-1 0 1; -2 0 2; -1 0 1];

>> sobel\_y = [-1 -2 -1; 0 0 0; 1 2 1];

>> ferrari = imread('D:/img/Ferrari.jpg');

>> ferrari = rgb2gray(ferrari);

>> ferrari = im2double(ferrari);

**• Limiarize a imagem da magnitude, para mostrar apenas pontos de borda  
“fortes”. Defina o limiar empiricamente.**

>> ferrari = imfilter(ferrari, fspecial('gaussian', [5 5]));

>> ferrari\_x = imfilter(ferrari, sobel\_x);

>> ferrari\_y = imfilter(ferrari, sobel\_y);

>> mag = abs(ferrari\_x) + abs(ferrari\_y);

>> imag = mag;

>> imag(find(mag < 0.3)) = 0;

**• Crie uma imagem indexada, de forma a colorir cada ponto de borda de acordo com a direção da borda:**

* **Amarelo: π/6 > ângulo > -π/6**
* **Verde: π/3 > ângulo > π/6**
* **Vermelho: -π/6 > ângulo > -π/3**
* **Azul: -π/3 > ângulo > π/3**

**Dica: use o comando *find* para encontrar os conjuntos de pixels que atendem  
às condições acima.**

>> ang = atan(ferrari\_y ./ ferrari\_x);

>> iang = zeros(size(ang, 1), size(ang, 2));

>> iang(find(ang < (pi/6) & ang > -(pi/6))) = 2; % 0 graus (amarelo)

>> iang(find(ang < (pi/3) & ang > (pi/6))) = 3; % 45 graus (verde)

>> iang(find(ang < -(pi/6) & ang > -(pi/3))) = 5; % 135 graus (vermelho)

>> iang(find(ang > (pi/3))) = 4; % 90 graus (azul)

>> iang(find(ang < -(pi/3))) = 4; % -270 graus (azul)

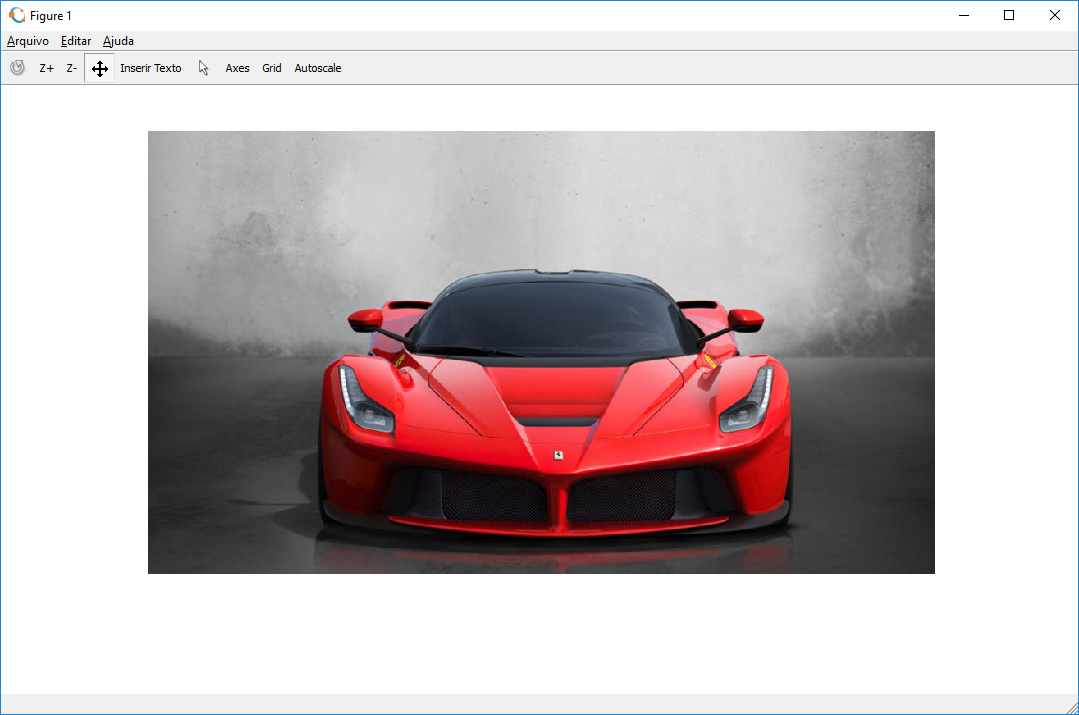
>> iang(find(mag < 0.3)) = 0;

>> map = [0 0 0; 1 1 0; 0 1 0; 0 0 1; 1 0 0];

**• Mostre em figuras diferentes a imagem original, a imagem com a magnitude do  
gradiente e a imagem indexada dos ângulos do gradiente**

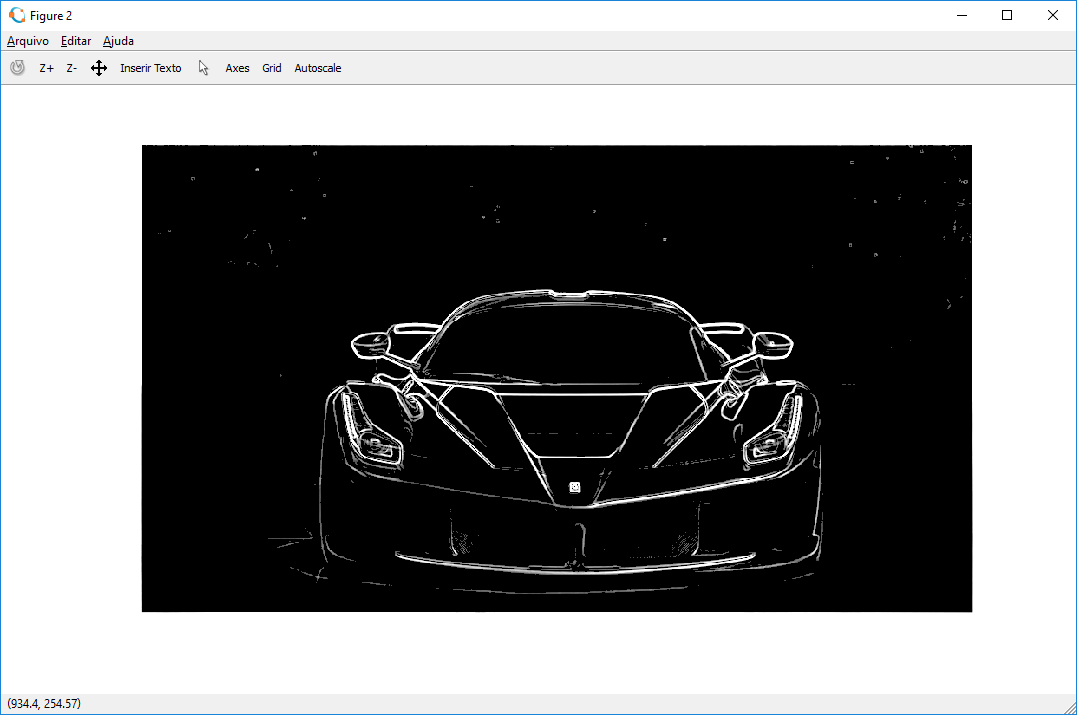
>> figure;

>> imshow(ferrari);



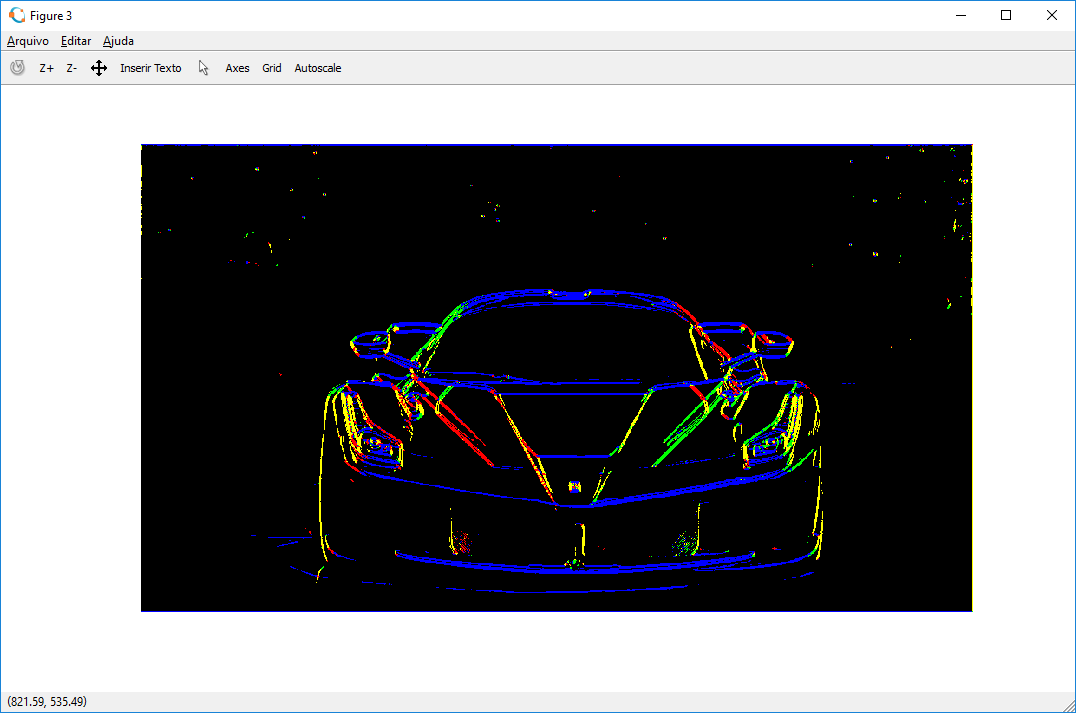
>> figure;

>> imshow(mag);



>> figure;

>> imshow(iang, map);



**Exercício 2:**

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

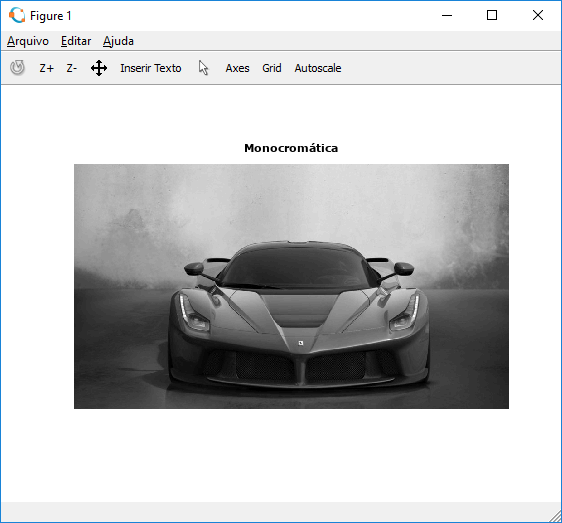
>> ferrari = imread('D:/img/ferrari.jpg');

>> fg = rgb2gray(ferrari);

>> figure;

>> imshow(fg);

>> title "Monocromática";

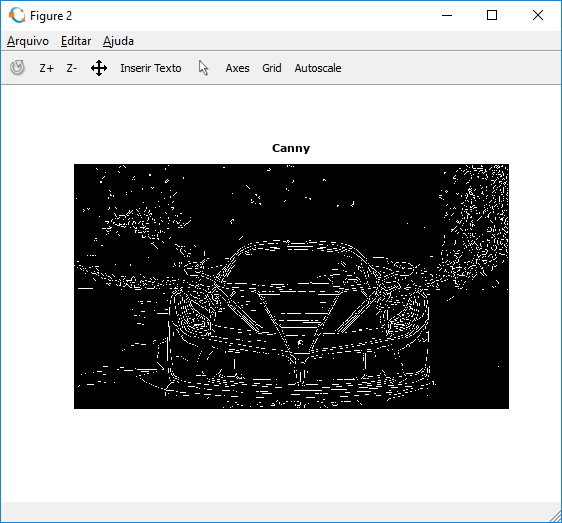


>> fCanny = edge(fg, "Canny");

>> figure;

>> imshow(fCanny);

>> title "Canny";



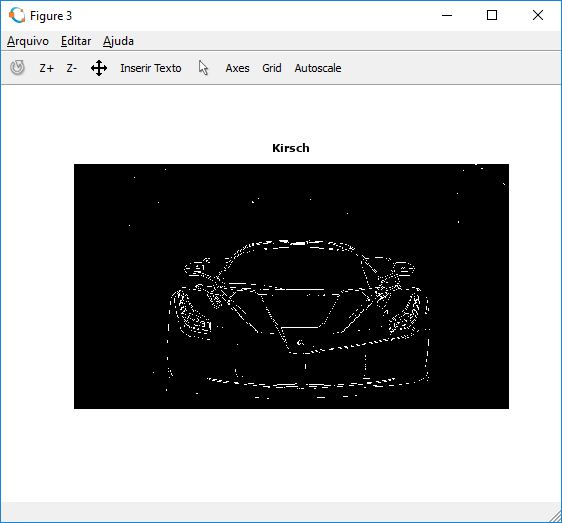
Percebe-se que o método de Canny, com os valores padrão do Octave, é bastante sensível às bordas em relação aos outros métodos disponíveis na função edge.

>> fKirsch = edge(fg, "Kirsch");

>> figure;

>> imshow(fKirsch);

>> title "Kirsch";



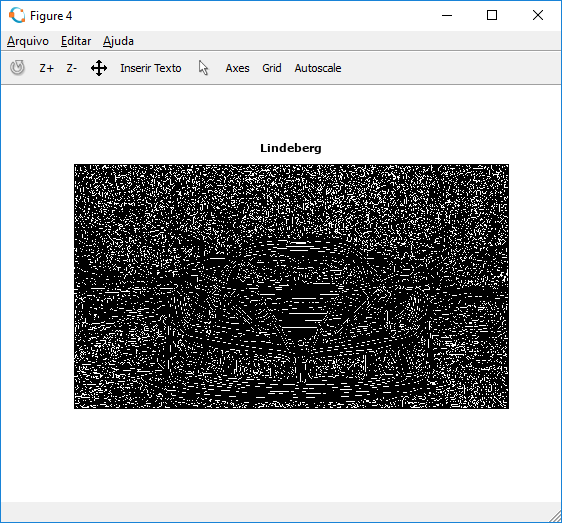
Percebe-se que o método de Kirsch, com os valores padrão do Octave, é um pouco menos sensível do que o ideal (para este caso). Ele se assemelha muito com os filtros de Prewitt e de Sobel, tendo apenas poucos pixels diferentes.

>> fLindeberg = edge(fg, "Lindeberg");

>> figure;

>> imshow(fLindeberg);

>> title "Lindeberg";



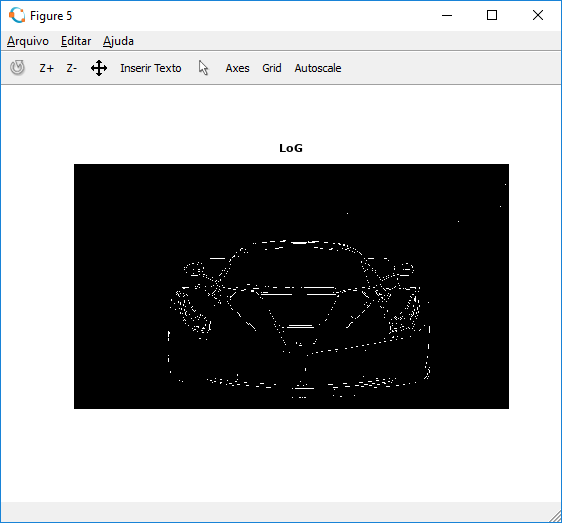
Percebe-se que o método de Lindeberg, com os valores padrão do Octave, é o mais sensível às bordas dentre os métodos disponíveis na função edge. Torna até mesmo difícil reconhecer a imagem original, neste caso.

>> fLoG = edge(fg, "LoG");

>> figure;

>> imshow(fLoG);

>> title "LoG";



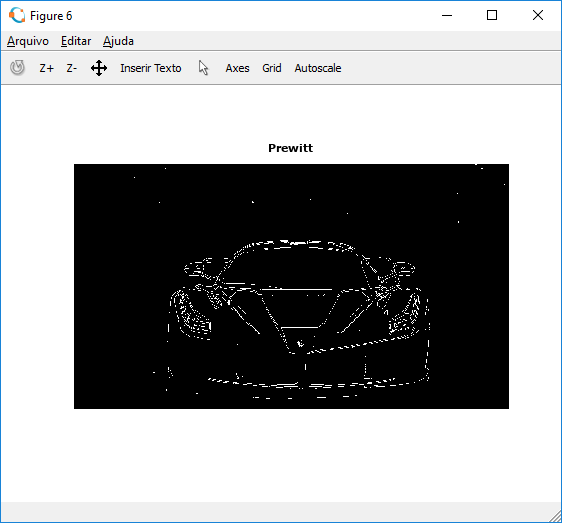
Percebe-se que o método de LoG, com os valores padrão do Octave, é o menos sensível às bordas dentre os métodos disponíveis na função edge.

>> fPrewitt = edge(fg, "Prewitt");

>> figure;

>> imshow(fPrewitt);

>> title "Prewitt";



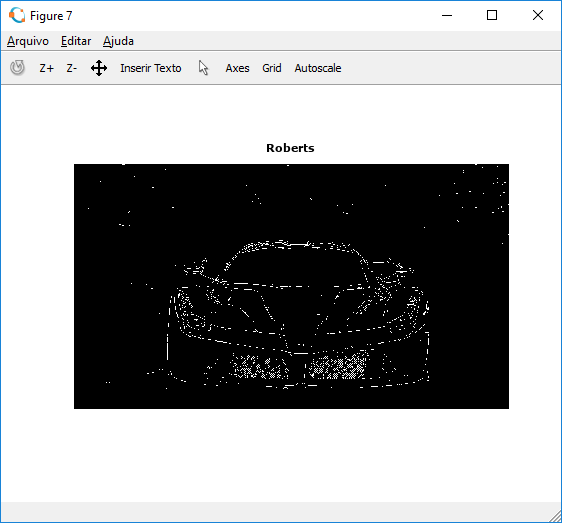
Percebe-se que o método de Prewitt, com os valores padrão do Octave, é um pouco menos sensível do que o ideal (para este caso). Ele se assemelha muito com os filtros de Kirsch e de Sobel, tendo apenas poucos pixels diferentes.

>> fRoberts = edge(fg, "Roberts");

>> figure;

>> imshow(fRoberts);

>> title "Roberts";



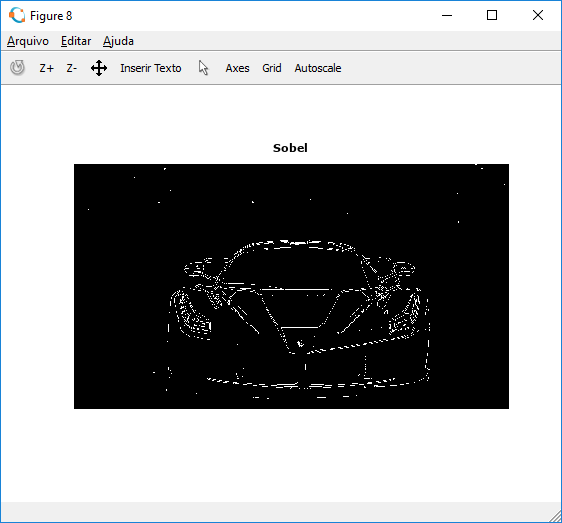
Percebe-se que o método de Roberts, com os valores padrão do Octave, é um pouco menos sensível à continuidade das bordas em relação aos outros métodos disponíveis na função edge. Assim como os filtros de Kirsch, Prewitt e Sobel, é um pouco menos sensível do que o ideal (para este caso).

>> fSobel = edge(fg, "Sobel");

>> figure;

>> imshow(fSobel);

>> title "Sobel";



Percebe-se que o método de Sobel, com os valores padrão do Octave, é um pouco menos sensível do que o ideal (para este caso). Ele se assemelha muito com os filtros de Kirsch e de Prewitt, tendo apenas poucos pixels diferentes.

**Exercício 3:**

**• 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*).**

>> gavea\_nir = imread('D:/img/gavea\_nir.png');

>> gavea\_r = imread('D:/img/gavea\_r.png');

>> gavea\_g = imread('D:/img/gavea\_g.png');

>> gavea\_b = imread('D:/img/gavea\_b.png');

>> gavea\_nir = rgb2gray(gavea\_nir);

>> gavea\_r = rgb2gray(gavea\_r);

>> gavea\_g = rgb2gray(gavea\_g);

>> gavea\_b = rgb2gray(gavea\_b);

**• Crie uma imagem cujos pixels equivalem ao NDVI (vide notas de aula). Lembre-se  
de, antes de calcular o NDVI, transformar as imagens das bandas para *double***

>> gavea\_nir = im2double(gavea\_nir);

>> gavea\_r = im2double(gavea\_r);

>> NDVI = (gavea\_nir - gavea\_r) ./ (gavea\_nir + gavea\_r);

**• Limiarize a imagem NDVI para selecionar os pixels com vegetação, criando uma  
imagem binária. Defina o limiar empiricamente.**

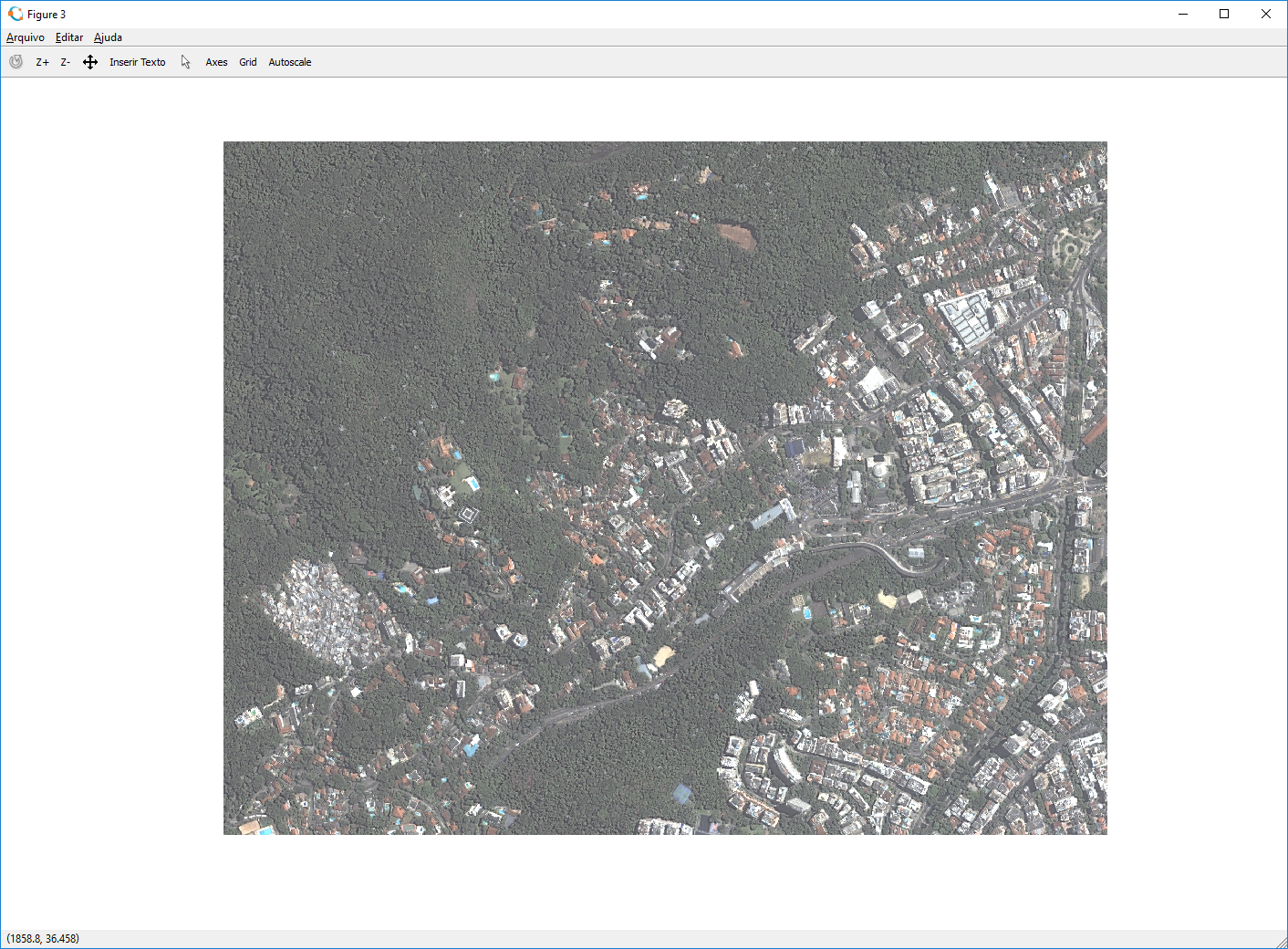
>> veg = NDVI > 0.15;

**• 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**

>> gavea\_original = cat(3, gavea\_r, gavea\_g, gavea\_b);

>> figure;

>> imshow(gavea\_original);



>> veg\_r = gavea\_r .\* veg;

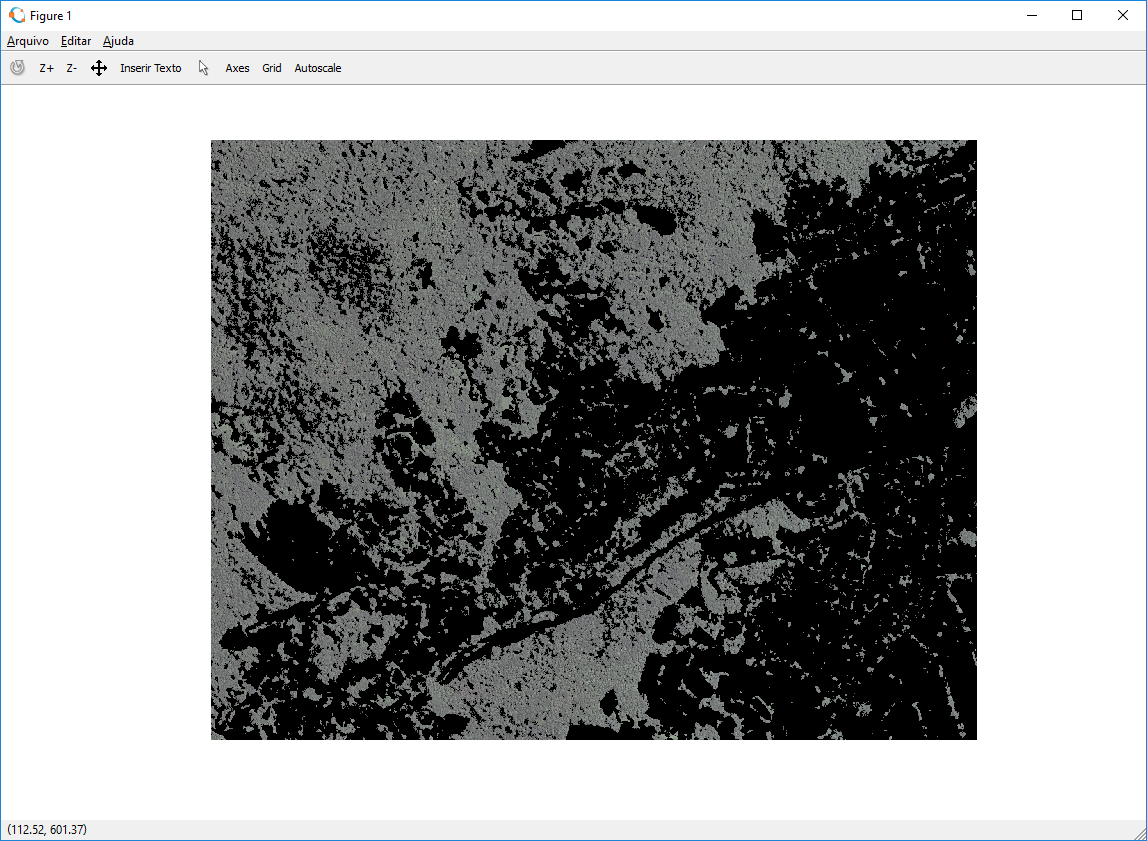
>> veg\_g = gavea\_g .\* veg;

>> veg\_b = gavea\_b .\* veg;

>> veg = cat(3, veg\_r, veg\_g, veg\_b);

>> figure;

>> imshow(veg);



>> naoVeg = NDVI <= 0.15;

>> naoVeg\_r = gavea\_r .\* naoVeg;

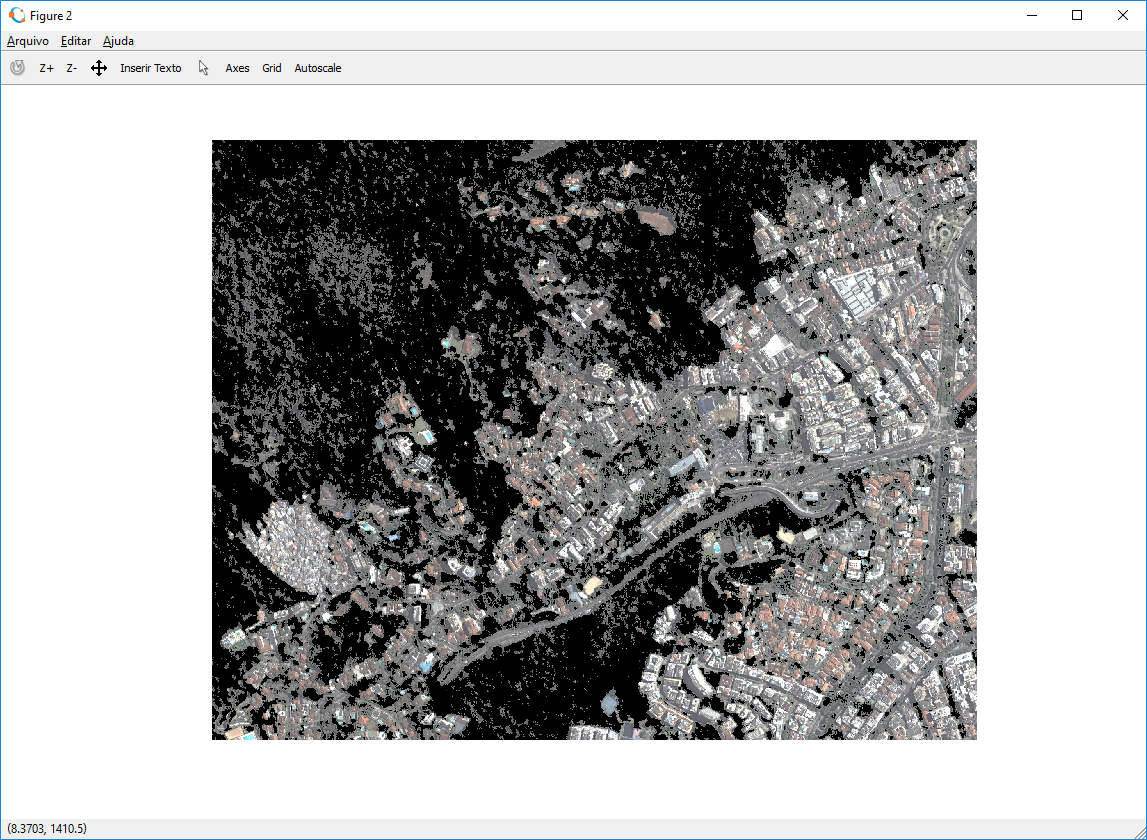
>> naoVeg\_g = gavea\_g .\* naoVeg;

>> naoVeg\_b = gavea\_b .\* naoVeg;

>> naoVeg = cat(3, naoVeg\_r, naoVeg\_g, naoVeg\_b);

>> figure;

>> imshow(naoVeg);



**Exercício 4:**

**• Baixe do AVA a imagem de uma digital.**

>> digital = imread('D:/img/digital.png');

**• Calcule um limiar global, para separar a digital do fundo da imagem, através do  
algoritmo descrito nas notas de aula.**

>> digital = rgb2gray(digital);

>> digital = im2double(digital);

>> limiar = limiarAnt = 0.2;

>> do

>> media\_baixo = mean(digital(find(digital < limiar)));

>> media\_alto = mean(digital(find(digital >= limiar)));

>> limiarAnt = limiar;

>> limiar = (media\_baixo + media\_alto) / 2;

>> until(limiar == limiarAnt)

>> limiar

limiar = 0.49390

**• Dica: use os comandos find e size para calcular as médias das intensidades  
dos pixels acima e abaixo do limiar.**

**• 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.**

>> subplot(1,3,1);

>> imshow(digital);

>> subplot(1,3,2);

>> imhist(digital);

>> str = cstrcat("Limiar = ", num2str(limiar));

>> title (str, "fontsize", 50);

>> subplot(1,3,3);

>> imshow(digital > limiar);

