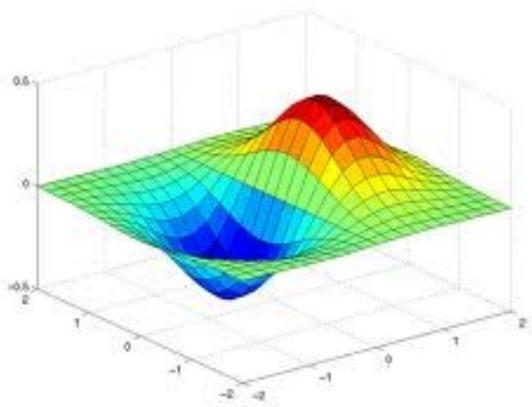


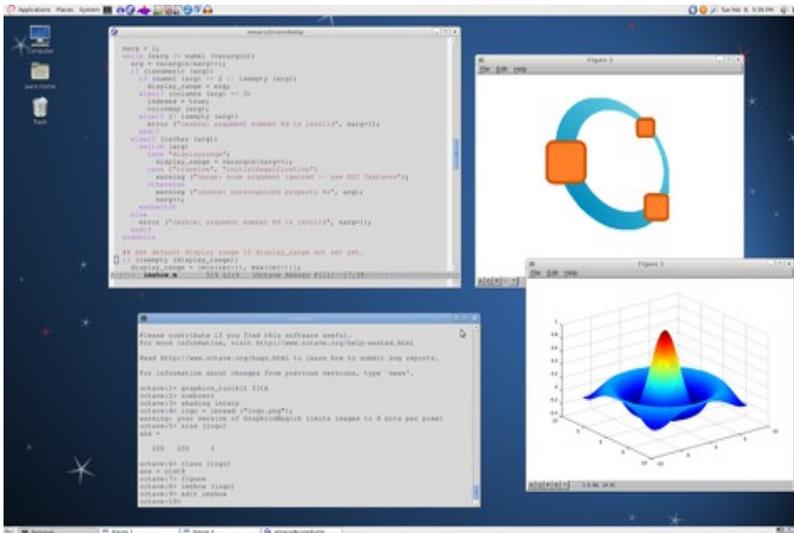
# Introdução ao processamento de imagens e OCTAVE



*Julio C. S. Jacques Junior*  
*juliojj@gmail.com*

# Octave

[www.gnu.org/software/octave/](http://www.gnu.org/software/octave/)



Linguagem Interpretada (similar ao MATLAB... portabilidade)

Voltada para Computação Numérica

Suporte à visualização e manipulação de dados

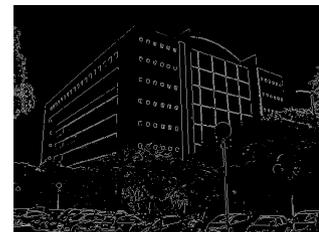
Interface por linha de comando (ou script)

# Motivação

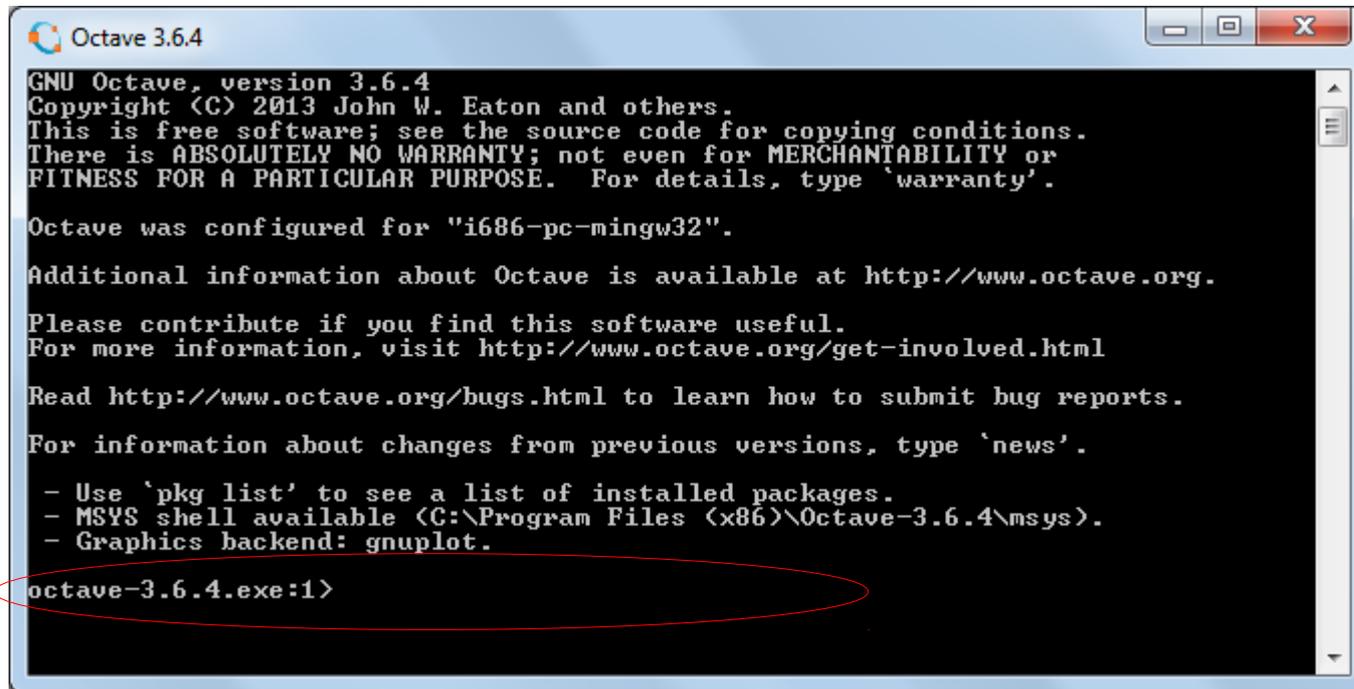
- Linguagem interpretada
  - Prototipação rápida
- Toolboxes
  - Processamento de imagens (octave 3.8.0) ← linux
  - Processamento de sinais
  - Estatística, etc

<http://octave.sourceforge.net/packages.php>

```
I = imread('facin.jpg');  
G = rgb2ntsc(I);  
E = edge(G(:,:,1),'sobel');  
imshow(E);
```



# Interface



```
GNU Octave, version 3.6.4
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-mingw32".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

- Use `pkg list' to see a list of installed packages.
- MSYS shell available (C:\Program Files (x86)\Octave-3.6.4\msys).
- Graphics backend: gnuplot.

octave-3.6.4.exe:1>
```

↖ Prompt de comando

# Help

```
>> help rgb2ntsc
```

```
octave-3.6.4.exe:14> help rgb2ntsc
'rgb2ntsc' is a function from the file C:\Program Files (x86)\Octave-3.6.4\share
\octave\3.6.4\m\image\rgb2ntsc.m

-- Function File:  rgb2ntsc (RGB)
   Transform a colormap or image from RGB to NTSC.

   See also:  ntsc2rgb

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at http://www.octave.org and via the help@octave.org
mailing list.
octave-3.6.4.exe:15> _
```

# Definição de variáveis, operações básicas e funções

- Variáveis não precisam ser declaradas

- Variáveis

- Ex.: atribuição e multiplicação

```
X = 5;  
Y = X * 2;
```

- Vetores

- Ex.: módulo do vetor

```
V = [1, 2, 3, 4];  
N = norm(V);
```

- Matrizes (2D)

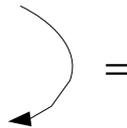
- Ex.: somatório

```
M = [3, 4, 6; 5, 6, 9; 12, 3, 1]  
S = sum(M(:));
```

# “Macetes”

- Matrizes (2D)

```
octave-3.6.4.exe:17>
octave-3.6.4.exe:17> M = [1 2 3;4 5 6]
M =
     1     2     3
     4     5     6
```

$$M = [1, 2, 3;4, 5, 6]$$
$$M = [1 2 3;4 5 6]$$


*Virgulas ou espaços em branco separam as colunas e ponto-e-vírgula as linhas (matriz 3x2)*

`sum (M)`

```
octave-3.6.4.exe:18> sum(M)
ans =
     5     7     9
```

`sum (M' )`

```
octave-3.6.4.exe:19> sum(M')
ans =
     6    15
```

`sum (sum (M) )`

```
octave-3.6.4.exe:20> sum(sum(M))
ans = 21
```

`sum (M (:))`

```
octave-3.6.4.exe:21> sum(M(:))
ans = 21
```

# Sumário

- Ler imagem RGB e exibir
- Verificar tamanho da imagem lida
- Exibir canais de cores individuais
- Transformar para escala de cinza
- Transformadas básicas (escala, rotação, crop)
- Filtragem espacial (suavização)
- Detecção de bordas
- Histograma
- Binarização

# Ler uma imagem do disco e exibir

- Diretório de trabalho

```
octave-3.6.4.exe:1> cd c:\temp
```

- Lendo uma image

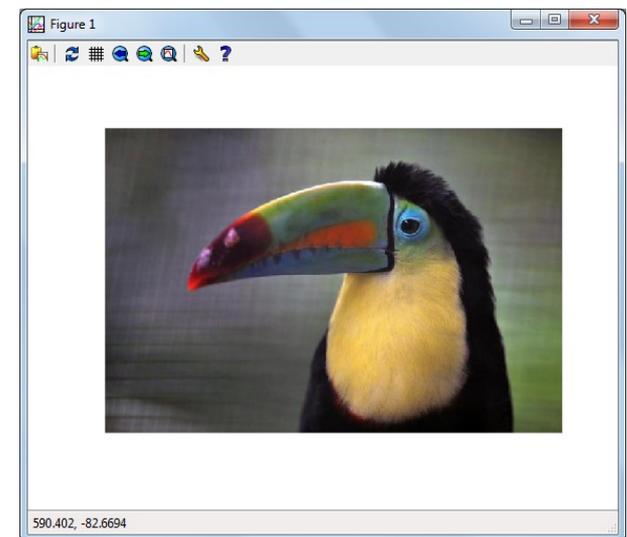
```
I = imread = ('tucano.jpg');
```

- Visualizando imagem lida:

```
imshow(I);
```

Ou

```
figure, imshow(I);
```



# Exibindo informações da imagem

```
Octave 3.6.4
For information about changes from previous versions, type 'news'.
- Use 'pkg list' to see a list of installed packages.
- MSYS shell available (C:\Program Files (x86)\Octave-3.6.4\msys).
- Graphics backend: gnuplot.

octave-3.6.4.exe:1> cd c:\temp
octave-3.6.4.exe:2> I = imread('tucano.jpg');
warning: your version of GraphicsMagick limits images to 16 bits per pixel
octave-3.6.4.exe:3> imshow(I)
octave-3.6.4.exe:4> whos
Variables in the current scope:

Attr Name      Size      Bytes  Class
==== =====
   I      412x620x3 766320  uint8

Total is 766320 elements using 766320 bytes

octave-3.6.4.exe:5> size(I)
ans =
   412   620     3

octave-3.6.4.exe:6>
```

octave-3.6.4.exe:1> whos

octave-3.6.4.exe:1> size(I)

octave-3.6.4.exe:1> s = size(I)

# Dimensões da imagem

```
% ver tamanho da imagem
```

```
>> size(I)
```

```
ans =
```

```
412    620     3 ← (Linhas Colunas Camadas)
```

```
% ver apenas num de linhas
```

```
>> size(I,1)
```

```
ans =
```

```
412
```

```
% ver apenas num de colunas
```

```
>> size(I,2)
```

```
ans =
```

```
620
```

Origem no canto superior esquerdo, na posição (1,1)



L  
I  
N  
H  
A  
S

COLUNAS

# Exibir canais de cores



```
% exibir o primeiro canal (Red)
```

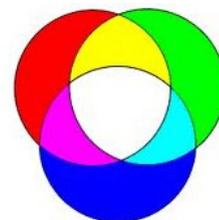
```
>> figure, imshow(I(:,:,1));
```

```
% atribuir o canal R à outra matriz
```

```
% e exibir o resultado
```

```
>> R = I(:,:,1);
```

```
>> figure, imshow(R);
```



```
% exibir o segundo canal (Green)
```

```
>> figure, imshow(I(:,:,2));
```

```
% exibir o terceiro canal (Blue)
```

```
>> figure, imshow(I(:,:,3));
```



R



G



B

# RGB para Escala de cinza

```
% função do octave (RGB → NTSC)
>> G = rgb2hsv(I);
% capturando o primeiro canal (V do HSV)
>> G = G(:, :, 3);
>> figure, imshow(G);

% média dos valores
>> G = (I(:, :, 1) + I(:, :, 2) + I(:, :, 3))/3;
```



V (HSV)



Média

ATENÇÃO (rgb2hsv)

Imagem de saída do tipo  
double (normalizada)

```
>> G = G * 255;
```

# Transformadas básicas

- Escala (*image toolkit*)

```
% o segundo parâmetro da função imresize define  
% o fator de escala  
>> I2 = imresize(I,0.5);  
>> I3 = imresize(I,2.0);
```



I  
(original)



I2



I3

# Transformadas básicas

- Rotação

```
% rotaciona 90° uma imagem (2D), no sentido  
anti-horário.
```

```
>> I2 = rotate(I(:, :, 1));
```

```
>> imshow(I2);
```



I(:, :, 1)



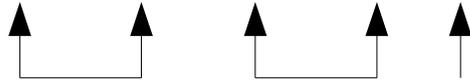
I2

# Transformadas básicas

- Crop

```
% recorta conteúdo da image I e armazena em I2
```

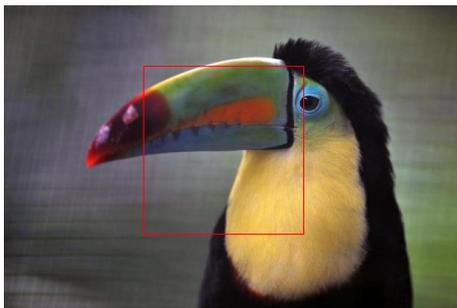
```
>> I2 = I(100:300,200:400,:);
```



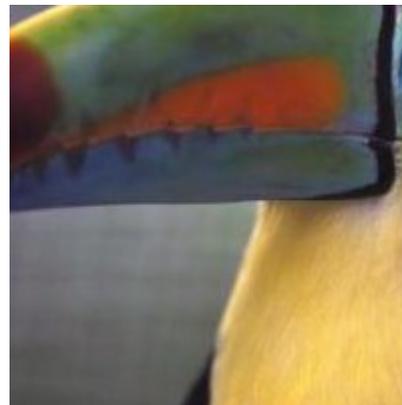
Da linha 100 até a linha 300,

Da coluna 200 até a coluna 400,

Para todos os canais (se a imagem tiver apenas um canal, o terceiro parâmetro pode ser omitido)



I  
(original)



I2  
(crop)

```
>> imshow(I2);
```

# Filtragem espacial

- Suavização (usando filtro da média)

```
% cria uma máscara com pesos  
% iguais, de tamanho 3x3 (filtro)  
% soma dos pesos deve ser = 1  
>> m = ones(3,3)/9;
```

```
0.1111    0.1111    0.1111  
0.1111    0.1111    0.1111  
0.1111    0.1111    0.1111
```

máscara de convolução

```
% convolve a imagem (gray) com a mascara (m)  
>> G2 = conv2(G,m);  
>> figure, imshow(G2);
```



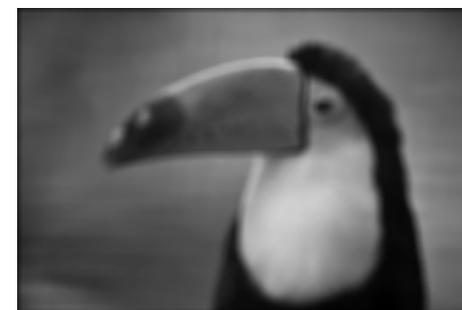
G  
(original)



máscara  
3x3



máscara  
5x5



máscara  
15x15

ATENÇÃO para o tamanho da imagem de saída (bordas adicionadas)

# Detecção de bordas

- Operador de Sobel

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

**mx**                      **my**

```
% criando as mascaras de convolução
```

```
>> mx = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
```

```
>> my = [1, 2, 1; 0, 0, 0; -1, -2, -1];
```

```
% calculando o gradiente (em x e y)
```

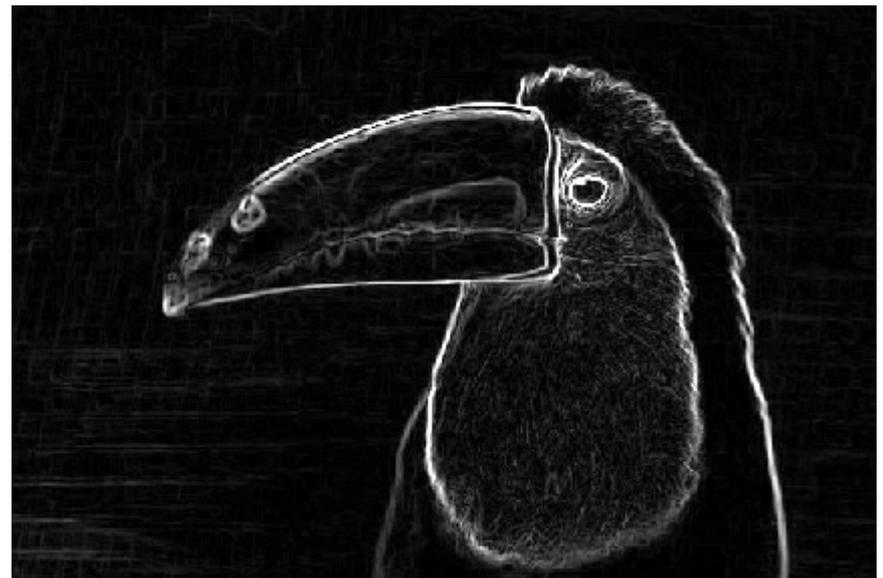
```
>> gx = conv2(g,mx);
```

```
>> gy = conv2(g,my);
```

```
% magnitude do gradiente
```

```
>> gxy = sqrt(gx.^2 + gy.^2);
```

```
>> figure, imshow(gxy);
```

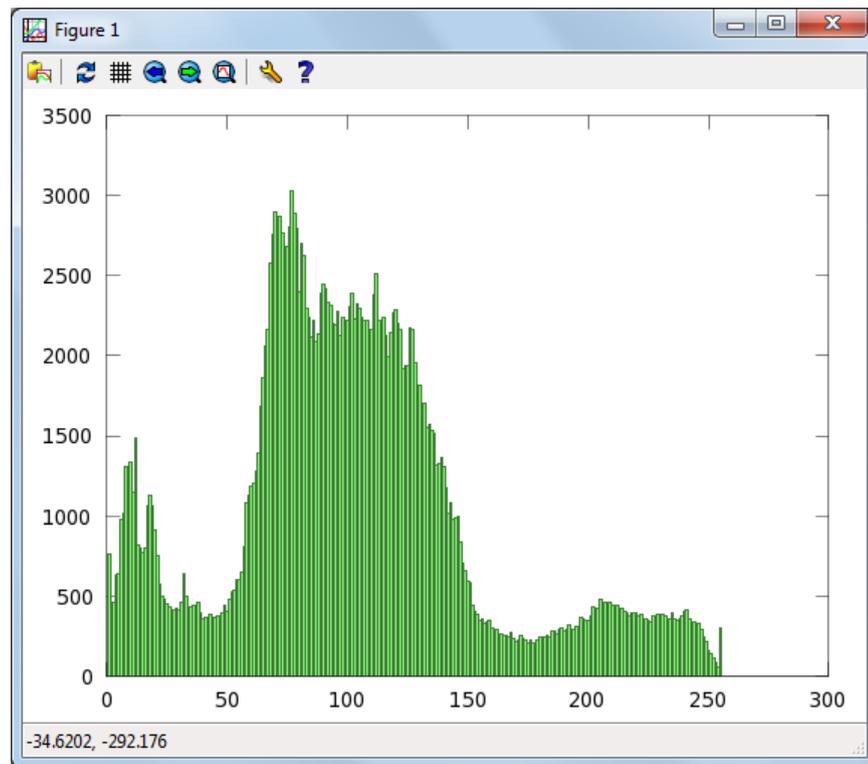


# Histograma

```
% distribuição de níveis de cinza  
>> hist(G(:),255);
```



G



histogama

# Binarização



G

% utiliza um limiar para dividir a imagem em dois grupos

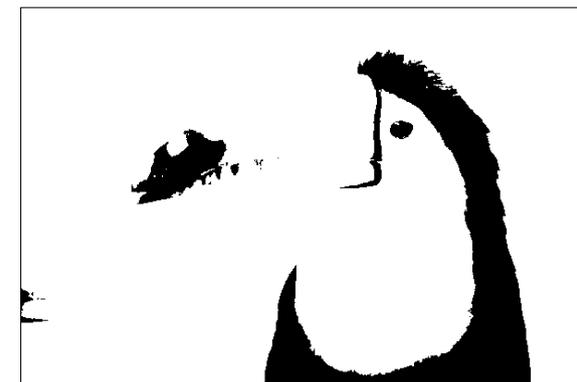
```
>> LIMIAR = 50;
```

% cria uma cópia de G em B

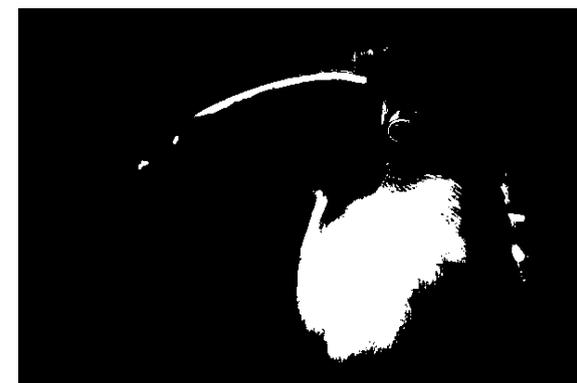
```
>> B = G;
```

```
>> B(B<=LIMIAR) = 0;
```

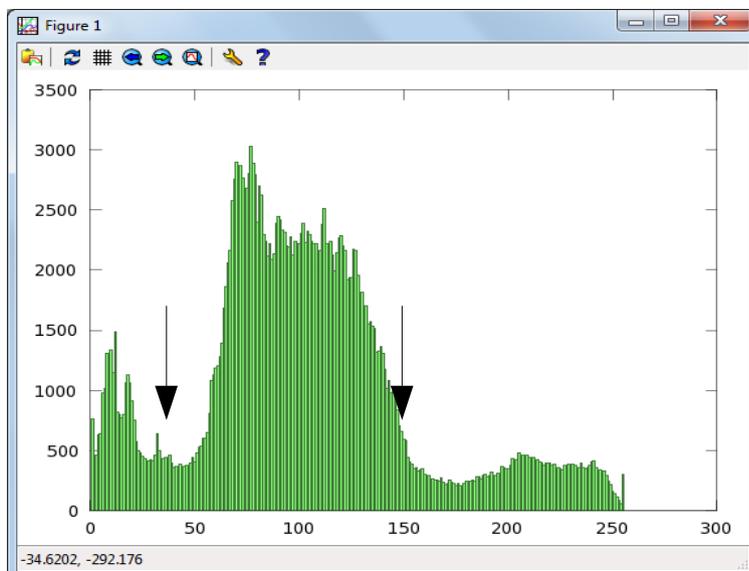
```
>> B(B>LIMIAR) = 255;
```

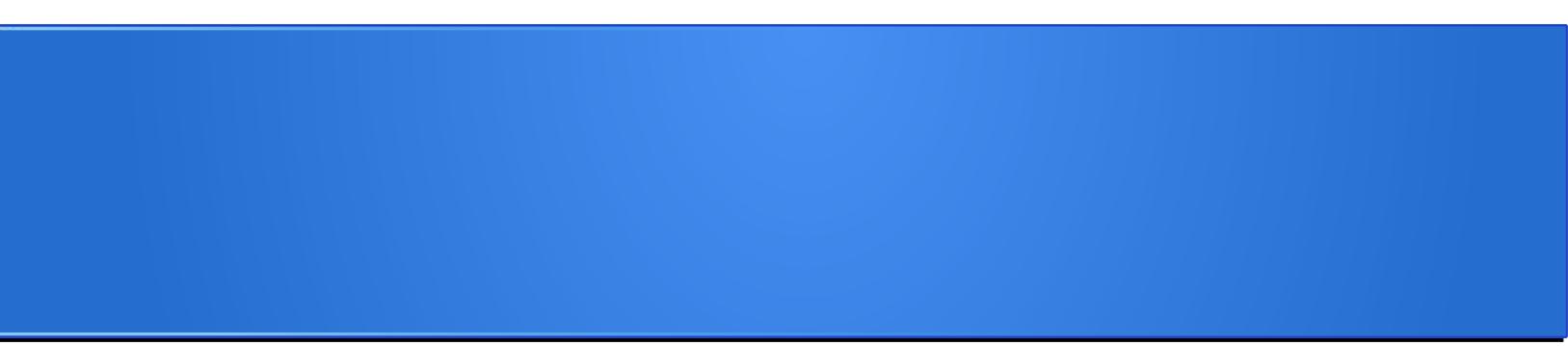


LIMIAR = 45



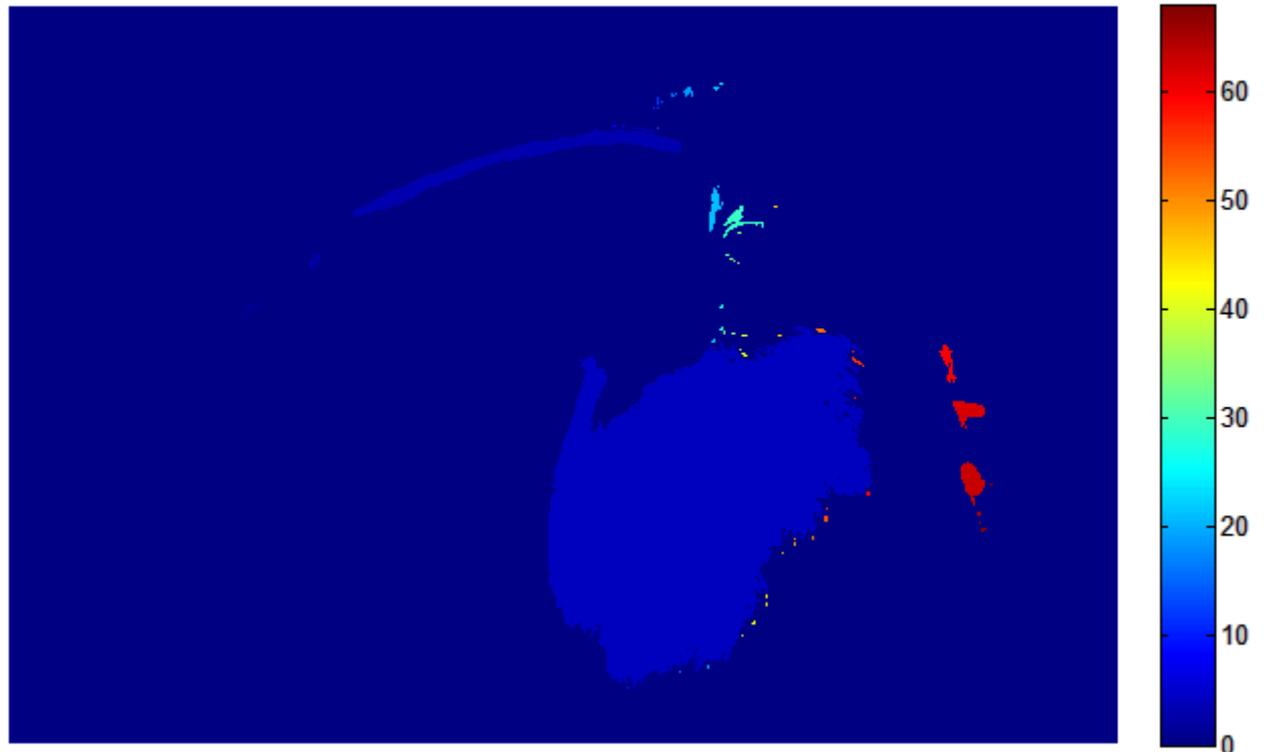
LIMIAR = 150





# Componentes conexos

```
% calcula os componentes conexos de uma imagem binária  
>> L = bwlabel(B);  
>> figure, imshow(L, []);  
>> colormap(jet), colorbar;
```



Cada componente conexo é representado por uma diferente cor.

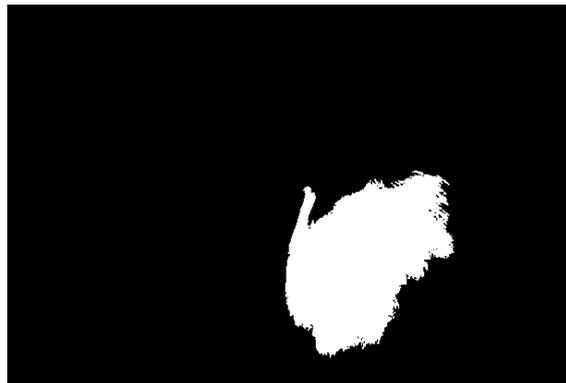
# Análise dos componentes conexos, laços e condicionais

- Mantém componente com maior área

```
% atribui o valor zeros para elementos com valor diferente do  
% componente de maior area  
L(L ~= indice) = 0;  
% atribui o valor 255 para o elementos com maior area  
L(L == indice) = 255;
```

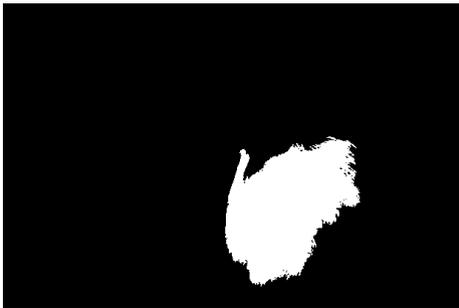


Binária



Maior componente conexo

# Edição de imagens



# Operações Morfológicas

- Dilatação
- Erosão
- Fechamento ( $D+E$ ) e Abertura ( $E+D$ )

