

## A3 - Computação Gráfica

**Professor:** Euzébio Souza

**Alunos:** Isaac Penaforte, Lucas Aquino, Júlia de Souza Aguiar

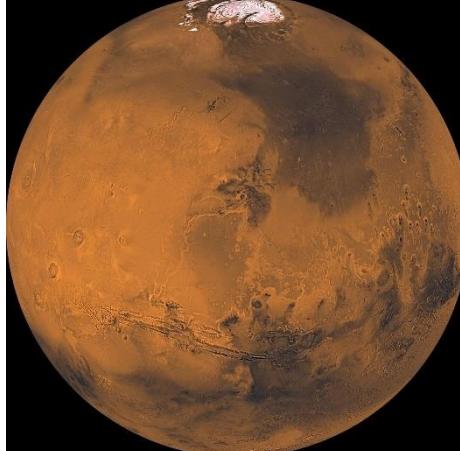
### Apresentação Canvas

**Tema do projeto:** Cálculo do diâmetro de planetas a partir de análise e processamento de imagens

O trabalho tem como objetivo calcular o diâmetro de um planeta a partir de imagens processadas usando a tecnologia do SciLab.

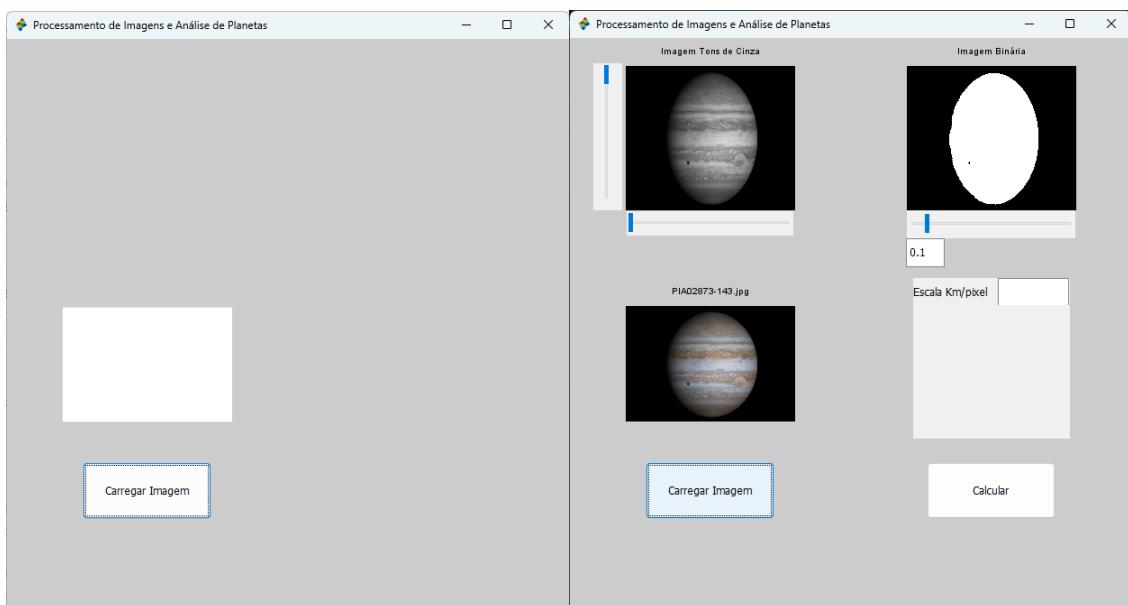
Nesse trabalho foram utilizadas três imagens dos planetas: Saturno, Marte, Júpiter

### **Imagens utilizadas e suas fontes:**

| Planetas  | Fonte | Escala       | Resolução   |
|---|-------|--------------|-------------|
|   | NASA  | 70 km/pixel  | 4613 x 2233 |
|  | NASA  | 143 km/pixel | 1920 x 1080 |
|  | NASA  | 1 km/pixel   | 6787 x 6787 |

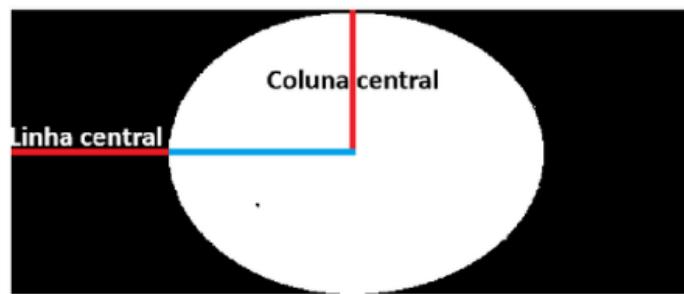
# Objetivo

A aplicação tem a função de processar e analisar imagens tiradas por sondas em missões de reconhecimento e análise de planetas. A partir dessas imagens, o sistema calcula o raio, diâmetro, o volume e área da superfície de um planeta, com base na quantidade de pixels e na escala de km/pixel fornecida.



O programa oferece a opção de seleção de imagem. Assim que escolhida, ela é transformada e exibida em escala de cinza, sendo também possível realizar o corte da imagem para conseguir maior precisão na análise e remover ruídos ou objetos indesejados. A partir da imagem em tons de cinza, uma versão binária da imagem é criada, com possibilidade de alternar o valor do limiar da geração. Essa transformação realça o planeta, facilitando a análise computacional.

Depois de ajustada a imagem, garantindo que o planeta fique centralizado na janela da interface, uma contagem dos pixels brancos é realizada sobre a matriz da imagem binária cortada. Essa contagem percorre a linha central da imagem, do início até posição da coluna central.



Ao final da contagem de pixels brancos, a escala em km/pixel, previamente inserida pelo usuário, é multiplicada pela quantidade de pixels contabilizados. Isso permite o cálculo do raio aproximado do planeta. Tendo em mãos o raio, é possível calcular:

- Diâmetro:  $d = 2Raio$
- Área de sua superfície:  $A = 4\pi R^2$
- Volume:  $V = (4\pi \cdot R^3)/3$

|                     |                           |
|---------------------|---------------------------|
| Escala Km/pixel     | 70                        |
| Contagem de pixels: | 849                       |
| Raio:               | 59430 quilômetros         |
| Diâmetro:           | 118860 quilômetros        |
| Área superfície:    | 4.438D+10 km <sup>2</sup> |
| Volume:             | 8.792D+14 km <sup>3</sup> |

# Escalas e Margem de Erro

Para garantir a maior precisão possível, assim como na leitura de mapas, a imagem escolhida mantenha sua resolução original e que a escala fornecida seja a mesma da fonte onde foi gerada.

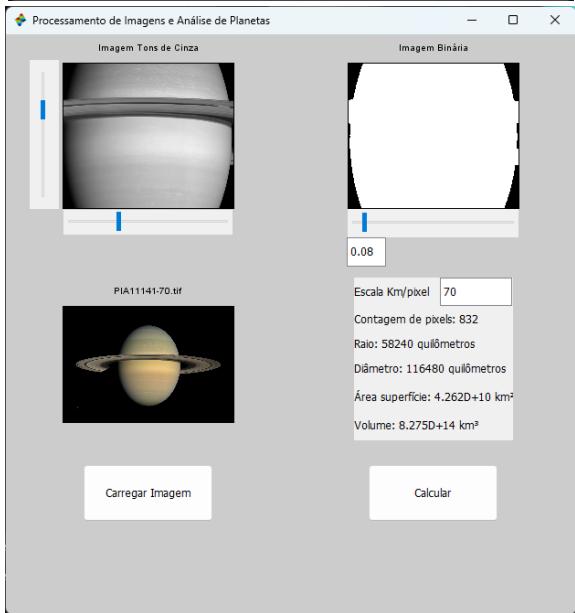
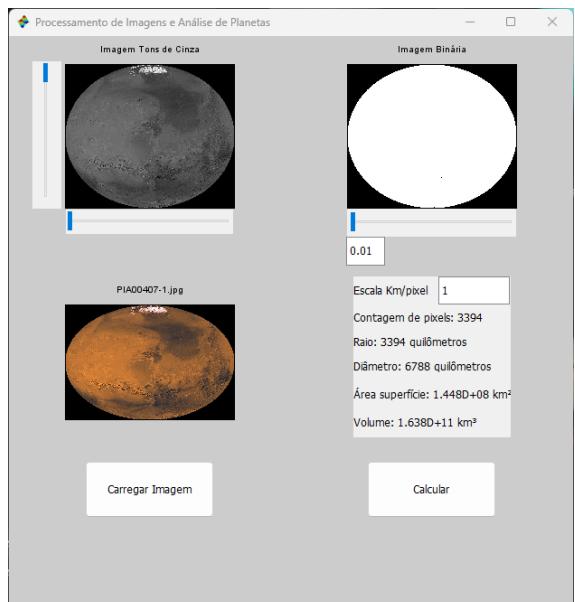
Sem essas condições não é possível chegar a um resultado satisfatório e com a menor margem de erro possível.

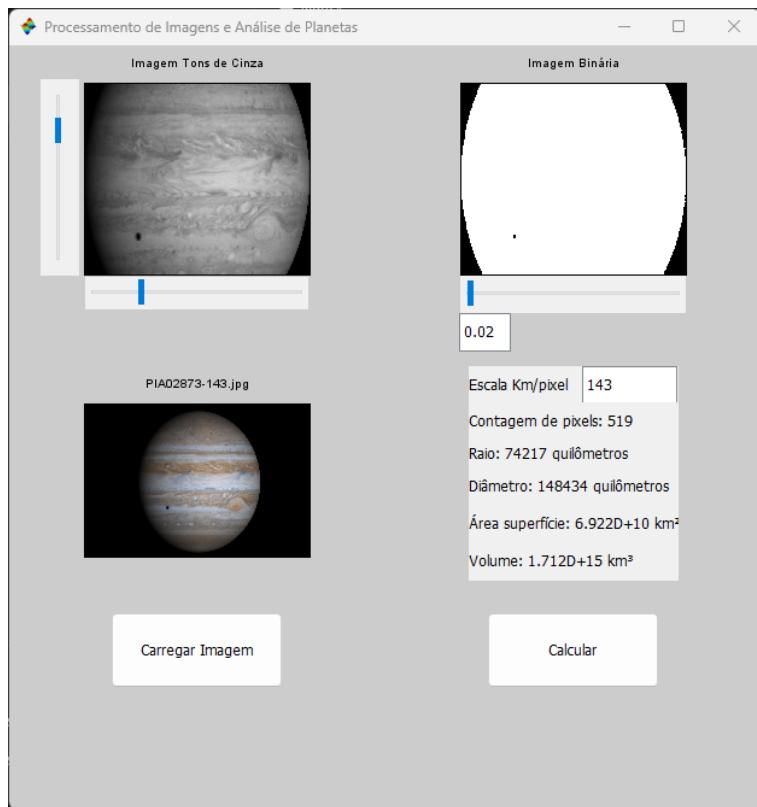
Os dados dos campos “**oficial**” foram retirados do site da NASA.

Como estamos lidando com planetas, o raio obtido pelo programa é um valor aproximado. Existe uma margem de erro influenciada por fatores como:

- Corte manual da imagem
- Resolução e precisão da escala usada
- Iluminação e contraste do planeta na foto

Apesar desses pontos, os resultados obtidos conseguem se manter próximos dos valores oficiais divulgados pela Nasa.





# Resultados

## Marte

| Medidas         | Oficial                                      | SciLab                              | Margem de Erro % |
|-----------------|--|-------------------------------------|------------------|
| Diâmetro        | $\approx 6.779 \text{ km}$                   | 6.788 km                            | 0.13%            |
| Área Superfície | $\approx 1,448 \times 10^8 \text{ km}^2$     | $1,448 \times 10^8 \text{ km}^2$    | 0.00%            |
| Volume          | $\approx 1,6318 \times 10^{11} \text{ km}^3$ | $1,638 \times 10^{11} \text{ km}^3$ | 0.38%            |

## Júpiter

| Medidas         | Oficial                                     | SciLab                              | Margem de Erro % |
|-----------------|---|-------------------------------------|------------------|
| Diâmetro        | $\approx 142.984 \text{ km}$                | 148.148 km                          | 3.61%            |
| Área Superfície | $\approx 6,14 \times 10^{10} \text{ km}^2$  | $6,895 \times 10^{10} \text{ km}^2$ | 12.30%           |
| Volume          | $\approx 1,431 \times 10^{15} \text{ km}^3$ | $1,702 \times 10^{15} \text{ km}^3$ | 18.90%           |

## Saturno

| Medidas         | Oficial                                      | SciLab                              | Margem de Erro % |
|-----------------|--|-------------------------------------|------------------|
| Diâmetro        | $\approx 120.536 \text{ km}$                 | 116.480 km                          | -3.36%           |
| Área Superfície | $\approx 4,27 \times 10^{10} \text{ km}^2$   | $4,262 \times 10^{10} \text{ km}^2$ | -0.19%           |
| Volume          | $\approx 8,2713 \times 10^{14} \text{ km}^3$ | $8.275 \times 10^{14} \text{ km}^3$ | 0.04%            |

## # Fontes

### Fotos e consultas

Saturno - <https://photojournal.jpl.nasa.gov/catalog/PIA11141>

Jupiter - <https://images.nasa.gov/details/PIA02873>

Marte - <https://photojournal.jpl.nasa.gov/catalog/PIA00407>

Instituto de Física da UFRGS - <https://www.if.ufrgs.br/if/>

Nasa - <https://www.nasa.gov>

Nasa Photo Journal - <https://photojournal.jpl.nasa.gov>

## Tamanhos e medidas dos planetas

Nasa Saturno - <https://science.nasa.gov/saturn/facts/>

Nasa Júpiter - <https://www.space.com/18392-how-big-is-jupiter.html>

Nasa Marte - <https://science.nasa.gov/mars/facts/>

## # Código Scilab

```
pratica.sce //////////////////////////////////////////////////////////////////
34 // Disciplina: Computação Gráfica
35 // Aula prática - 2025_1
36 // Programa: Interface GUI BUILDER - Processamento de Imagens
37 // Turma: UNA - Cristiano Machado - Manhã - 14/05
38 // Integrantes: Isaac Penaforte
39 // ..... Júlia Aguiar
40 // ..... Lucas Ferreira
41 //////////////////////////////////////////////////////////////////
42
43 clc; // Limpa console
44
45 f.userData = handles; // Permite armazenar imagens do eixo "handles"
46 // em UserData para uso em funções...
47
48
49 // Define variáveis como globais para visualização no navegador de variáveis
50 global Im1 Im2 im_binaria_im_cortada;
51
52 function btn1_callback(handles)
53 //Write your callback for btn1 here
54 global Im1 Im2; // Define variáveis como global para visualização no navegador de variáveis
55
56 limpar_frames(handles); // Limpa os Axes de imagens anteriores
57 visibilidade(handles,"off"); // Torna os campos e Axes invisíveis até selecionar uma imagem válida
58 f = gcf(); // Recebe manipulador da interface
59 handles = f.userData; // Recupera imagens armazenadas em UserData
60
61 [ImNome,caminho] = uigetfile(); // Recebe uma imagem do navegador de arquivos
62 Im1 = imread(fullfile(caminho,ImNome)); // Lê e armazena uma imagem
63 handles.nome_original = ImNome; // Armazena o nome da imagem
64
65 Im1 = imread(fullfile(caminho,ImNome)); // Lê e armazena uma imagem
66 handles.nome_original = ImNome; // Armazena o nome da imagem
67
68 handles.nome_original = ImNome;
69 visibilidade(handles,"on"); // Mostra os Axes e campos
70
71 handles.ImOriginal = Im1; // Armazena a imagem original
72 sca(handles.Axes_Original); // Seleciona um Axes para plot
73 imshow(Im1);replot(); // Apresenta imagem e a redimensiona para encaixar no Axes
74 title(handles.nome_original); // Título da imagem
75
76 sca(handles.Axes_Cinza); // Seleciona um Axes para plot
77 Im2 = rgb2gray(Im1); // Transforma imagem genérica para escala-de-cinza
78 handles.Im2 = Im2; // Armazena Im2 no handles
79 imshow(Im2);replot(); // Apresenta imagem e a redimensiona para encaixar no Axes
80 title("Imagen Tons de Cinza"); // Título imagem
81
82 f.userData = handles; // Armazena dados/imagens em UserData
83 sld_bin_callback(handles);
84
85 endfunction
86
87 function btn_diametro_callback(handles)
88 //Write your callback for btn_diametro here
89
90
91 escala = strtod(handles.edt_escala.String); // Recebe o campo da escala de KM/pixel do usuário como valor numérico
92 if isnan	escala then // Se não for um valor válido exibe erro e para a função
93 messagebox("Preencha o campo de escala com um valor válido!","Aviso","warning");
94 return;
95 end
96 handles.txt.frame.String = 'Resultado: Carregando.'; // Apresenta a mensagem na interface
```

```

8     --> emu
9     handles.txt_frame.String = 'Resultado: Carregando.'; // Apresenta a mensagem na interface
10    handles.txt_result.String = ''; // Deixa vazio o campo resultado
11    editaveis(handles,'off'); // Desativa recursos da interface
12
13    f = gcf(); // Recebe manipulador da interface
14    handles = f.UserData; // Recupera imagens armazenadas em UserData
15    im_binaria = handles.im_binaria; // Aponta para a imagem binária no handles
16
17    [rows, cols] = size(im_binaria); // Recebe quantidade de linhas e colunas da imagem
18
19    pos_central_coluna = round(cols/2); // Divide colunas para ter o valor central da imagem
20    pos_central_linha = round(rows/2); // Divide linhas para ter o valor central da imagem
21    contador = 0; // Quantidade de pixels brancos contabilizados
22
23    for j = 1:pos_central_coluna // Percorre a linha até o centro da imagem
24
25        pixel = im_binaria(pos_central_linha, j); // Define pixel a posição atual na matriz
26
27        if pixel == %T then // Se pixel for branco o contador aumenta
28            contador=contador+1; // Contador de pixels brancos
29        end
30
31    end
32
33    if contador ~= 0 then // Se detectar pixels brancos, calcula raio e diametro
34        raio=contador;
35    else // Se não encontrar, envia um aviso e retorna
36        messagebox("Nenhum pixel detectado!","Aviso","warning");
37        editaveis(handles,'on'); // Permite o uso da interface novamente
38
39    end
40
41
42    raio = raio * escala;// Calcula o valor do raio multiplicando o contador pela escala
43    diametro= raio * 2;// Calcula o diametro
44    area_superficie = 4*pi*raio^2; // Calcula a área da superfície
45    volume = (4*pi*raio^3)/3; // Calcula o volume
46
47    msg_contador = strcat(['Contagem de pixels: ',string(contador)]); // Define a mensagem que representa o contador de pixels
48    msg_raio = strcat(['Raio: ',string(raio),' quilômetros']); // Define a mensagem que representa o raio
49    msg_diametro = strcat(['Diâmetro: ',string(diametro),' quilômetros']); // Define a mensagem que representa o diametro
50    msg_area = strcat(['Área superfície: ',string(area_superficie),' km²']); // Define a mensagem que representa o diametro
51    msg_volume = strcat(['Volume: ',string(volume),' km³']); // Define a mensagem que representa o volume
52
53    handles.txt_frame.String = msg_raio; // Apresenta o raio do planeta
54    handles.txt_result.String = msg_diametro; // Apresenta o diametro na interface
55    handles.txt_resultinfo.String = msg_area; // Apresenta a área da superfície do planeta
56    handles.txt_volume.String = msg_volume; // Apresenta o volume do planeta
57    handles.txt_contador.String = msg_contador; // Apresenta o contador na interface
58
59
60    editaveis(handles,'on'); // Permite o uso da interface novamente
61
62    f.UserData = handles; // Armazena dados/imagens em UserData
63
64 endfunction
65
66
67 editaveis(handles,'on'); // Permite o uso da interface novamente
68 return
69 end
70
71
72 raio = raio * escala;// Calcula o valor do raio multiplicando o contador pela escala
73 diametro= raio * 2;// Calcula o diametro
74 area_superficie = 4*pi*raio^2; // Calcula a área da superfície
75 volume = (4*pi*raio^3)/3; // Calcula o volume
76
77 msg_contador = strcat(['Contagem de pixels: ',string(contador)]); // Define a mensagem que representa o contador de pixels
78 msg_raio = strcat(['Raio: ',string(raio),' quilômetros']); // Define a mensagem que representa o raio
79 msg_diametro = strcat(['Diâmetro: ',string(diametro),' quilômetros']); // Define a mensagem que representa o diametro
80 msg_area = strcat(['Área superfície: ',string(area_superficie),' km²']); // Define a mensagem que representa o diametro
81 msg_volume = strcat(['Volume: ',string(volume),' km³']); // Define a mensagem que representa o volume
82
83 handles.txt_frame.String = msg_raio; // Apresenta o raio do planeta
84 handles.txt_result.String = msg_diametro; // Apresenta o diametro na interface
85 handles.txt_resultinfo.String = msg_area; // Apresenta a área da superfície do planeta
86 handles.txt_volume.String = msg_volume; // Apresenta o volume do planeta
87 handles.txt_contador.String = msg_contador; // Apresenta o contador na interface
88
89
90 editaveis(handles,'on'); // Permite o uso da interface novamente
91
92 f.UserData = handles; // Armazena dados/imagens em UserData
93
94 endfunction
95
96
97 function atualizar_imbin(handles) // Gera nova imagem binária

```

```

147 function atualizar_imbin(handles) // Gera nova imagem binária
1 global im_binaria; // Define variável como global para visualização no navegador de variáveis
2 f = gcf(); // Recebe manipulador da interface
3 handles = f UserData; // Recupera imagens armazenadas em UserData
4
5
6 delete(handles.Axes_Binario.children); // Remove a imagem binária anterior
7 im_binaria = im2bw(handles.Im2,handles.bin_threshold); // Gera imagem binária da imagem em escala de cinza
8 handles.im_binaria = im_binaria; // Armazena imagem binária em handles
9 sca(handles.Axes_Binario); // Seleciona um Axes para plot
10 imshow(im_binaria);replot(); // Apresenta imagem e a redimensiona para encaixar no Axes
11 title("Imagen Binária"); // Da título a imagem
12
13 f.UserData = handles; // Armazena dados/imagens em UserData
14 endfunction
162
1 function atualizar_corte(handles) // Corta a imagem em escala de cinza
2 global im_cortada; // Define variável como global para visualização no navegador de variáveis
3 f = gcf(); // Recebe manipulador da interface
4 handles = f.UserData; // Recupera imagens armazenadas em UserData
5
6 im = handles.ImOriginal; // Recebe a imagem original
7 [altura, largura] = size(im); // Armazena as linhas e colunas da imagem
8
9 // Normaliza os sliders entre 1 até altura/largura
10 x1 = round(handles.sld_x.value * (altura - 1)) + 1;
11 y1 = round(handles.sld_x.value * (largura - 1)) + 1;
12
13 x2 = round(handles.sld_y.value * (altura - 1)) + 1;
14 y2 = round(handles.sld_y.value * (largura - 1)) + 1;
15
16 // Corrige o corte da imagem em caso de inversão de valores/sliders
17 // para em im(x1:x2,y1:y2) x1 ou y1 não seja maior que x2 ou y2 respectivamente
18 if x1 > x2 then
19 temp = x1; // Armazena x1 em temp
20 x1 = x2; // x1 recebe o valor menor
21 x2 = temp; // x2 recebe o valor maior em temp
22 end
23 if y1 > y2 then
24 temp = y1; // Armazena y1 em temp
25 y1 = y2; // y1 recebe o valor menor
26 y2 = temp; // y2 recebe o valor maior em temp
27 end
28
29 // Corta a imagem e a retorna como Im2
30 im_cortada = im(x1:x2, y1:y2);
31 handles.Im2 = im_cortada;
32
33 delete(handles.Axes_Cinza.children); // Limpa a imagem antiga
34 sca(handles.Axes_Cinza); // Recebe o Axes1 como referencia
35 imshow(im_cortada);replot(); // Plota a imagem cortada e a reajusta no Axes
36
37 f.UserData = handles; // Armazena dados/imagens em UserData
38 sld_bin_callback(handles); // Atualiza e gera nova imagem binária com a imagem recortada
39
40 endfunction
203
203
1 function sld_x_callback(handles) // Callback do slider de corte de imagem
2 //Write your callback for sld_x here
3 atualizar_corte(handles);
4 endfunction
208
1 function sld_y_callback(handles) // Callback do slider de corte de imagem
2 //Write your callback for sld_x2 here
3 atualizar_corte(handles);
4 endfunction
213
214
1 function sld_bin_callback(handles) // Callback do slider do valor limite binário
2 //Write your callback for sld_bin here
3 f = gcf(); // Recebe manipulador da interface
4 handles = f.UserData; // Recupera imagens armazenadas em UserData
5
6 handles.edt_bin.String = string(handles.sld_bin.value); // Campo editável recebe valor do slider
7 handles.bin_threshold = handles.sld_bin.value; // Atualiza novo valor limite binário
8
9 f.UserData = handles;
10 atualizar_imbin(handles); // Atualiza imagem binária com novo valor limite
11
12 endfunction
227
1 function edt_bin_callback(handles) // Callback do campo de texto do valor limite binário
2 f = gcf(); // Recebe manipulador da interface
3 handles = f.UserData; // Recupera imagens armazenadas em UserData
4 if isempty(handles.edt_bin.String) then

```



```

21     handles.txt_escala.Visible = modo;
22
23
24 endfunction
291 // Desativa sliders, botões e campos
1 function editaveis(handles,modo)
2     // modo = 'on' ou 'off'
3
4     handles.sld_x.Enable = modo;
5     handles.sld_y.Enable = modo;
6     handles.sld_bin.Enable = modo;
7     handles.edt_escala.Enable = modo;
8     handles.edt_bin.Enable = modo;
9     handles.btn_diametro.Enable = modo;
10    handles.btnl.Enable = modo;
11
12 endfunction
305 visibilidade(handles,"off"); // Inicia o programa com os campos e Axes invisiveis
307
308 // Anotações -----
309 //
310 // Para o calculo do diametro correto, o planeta deve permanecer no centro do campo Axes,
311 // se necessário, cortando a imagem utilizando os sliders para evitar objetos/areas que não
312 // fazem parte da area de interesse.
313 // A escala deve ser definida antes de realizar o calculo, conferir fonte.
314 //
315 // f(userData = handles;

313 // A escala deve ser definida antes de realizar o calculo, conferir fonte.
314 //
315 // f.userData = handles;
316 // Permite armazenar imagens do eixo "handles".
317 // em UserData para uso em funções...
318 //
319 // -rgb2gray: Transforma uma imagem para escala de cinza
320 // -uigetfile: Recebe imagem do explorador de arquivos
321 // -sca: Seleciona um Axe específico para plot
322 //
323 // Cada elemento na interface possui atributos como visible e enable que
324 // permitem a alteração de sua aparência e uso na interface, devem ser modificados
325 // para evitar interações indevidas e bugs utilizando as funções "editaveis" e "visibilidade"
326 //
327 // delete: remove do Axes selecionado a imagem ocupada
328 /////////////
329
330
331

```