Rio de Janeiro, 22 de setembro de 2017

Universidade do Estado do Rio de Janeiro

Instituto de Matemática e Estatística

Computação Gráfica

Trabalho 2 – Manipulação de Imagens Digitais

Aluno: Leonardo Lima Marinho

Matrícula: 2014.1.00506.11

Exercício 1:

- **Leia a ajuda para as funções *imadjust* e *imhist***

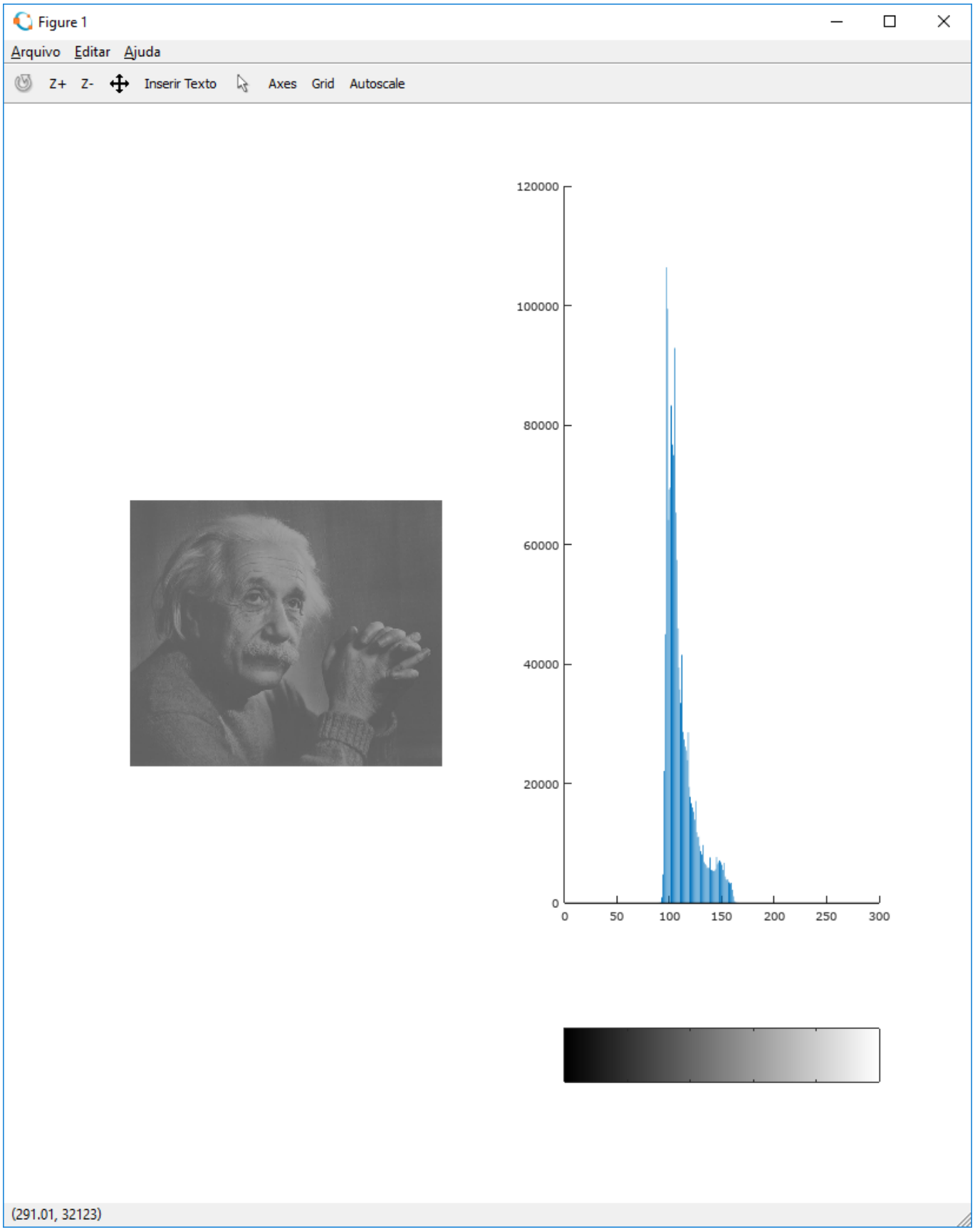
```
>> help imadjust
>> help imhist
```

- **Leia as imagens *Einstein_low_contrast.png*, *Einstein_med_contrast.png* e *Einstein_high_contrast.png***

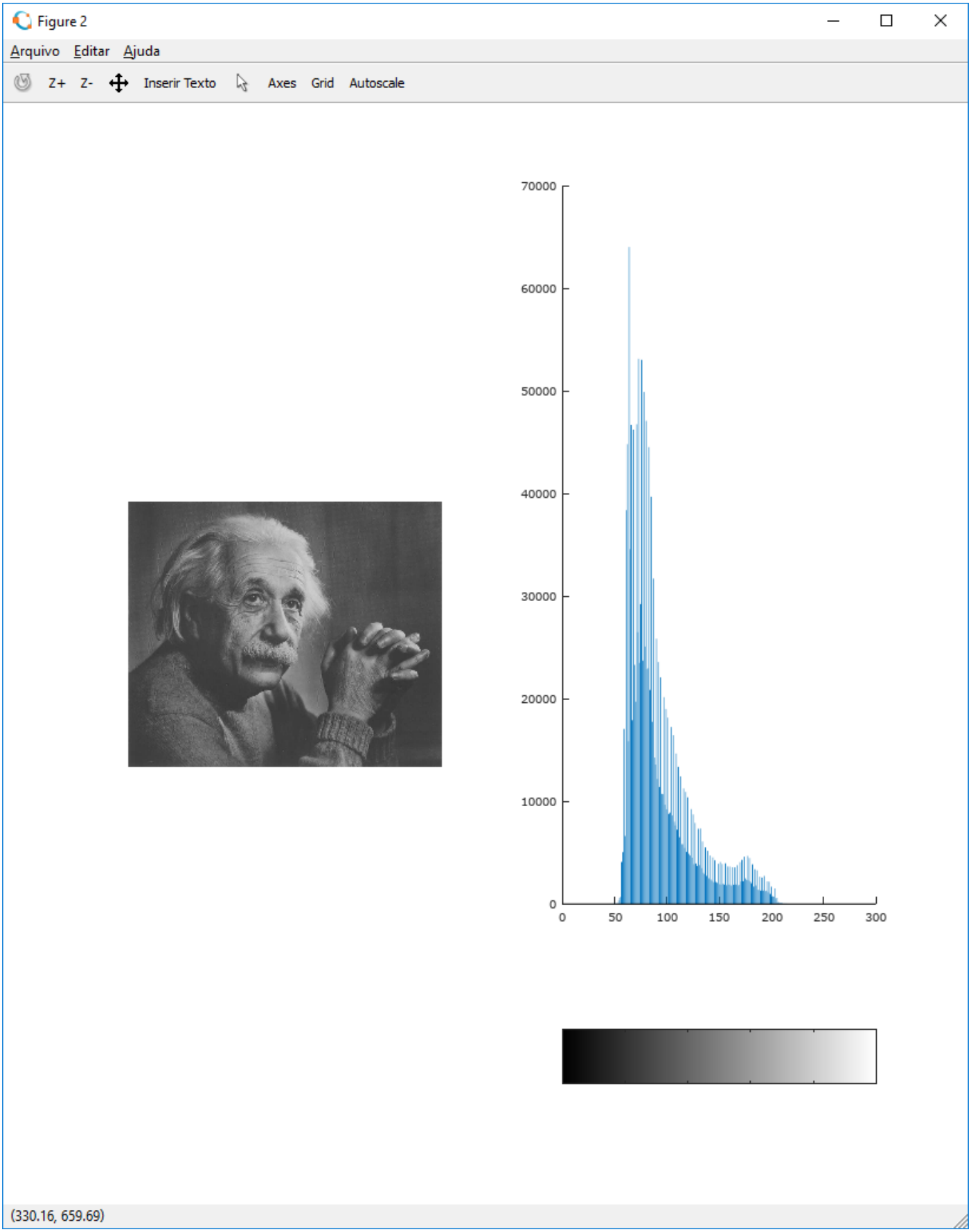
```
>> elc = imread("D:/img/Einstein_low_contrast.png");
>> emc = imread("D:/img/Einstein_med_contrast.png");
>> ehc = imread("D:/img/Einstein_high_contrast.png");
```

- **Apresente cada imagem e seu respectivo histograma em uma figura diferente. Compare os histogramas.**

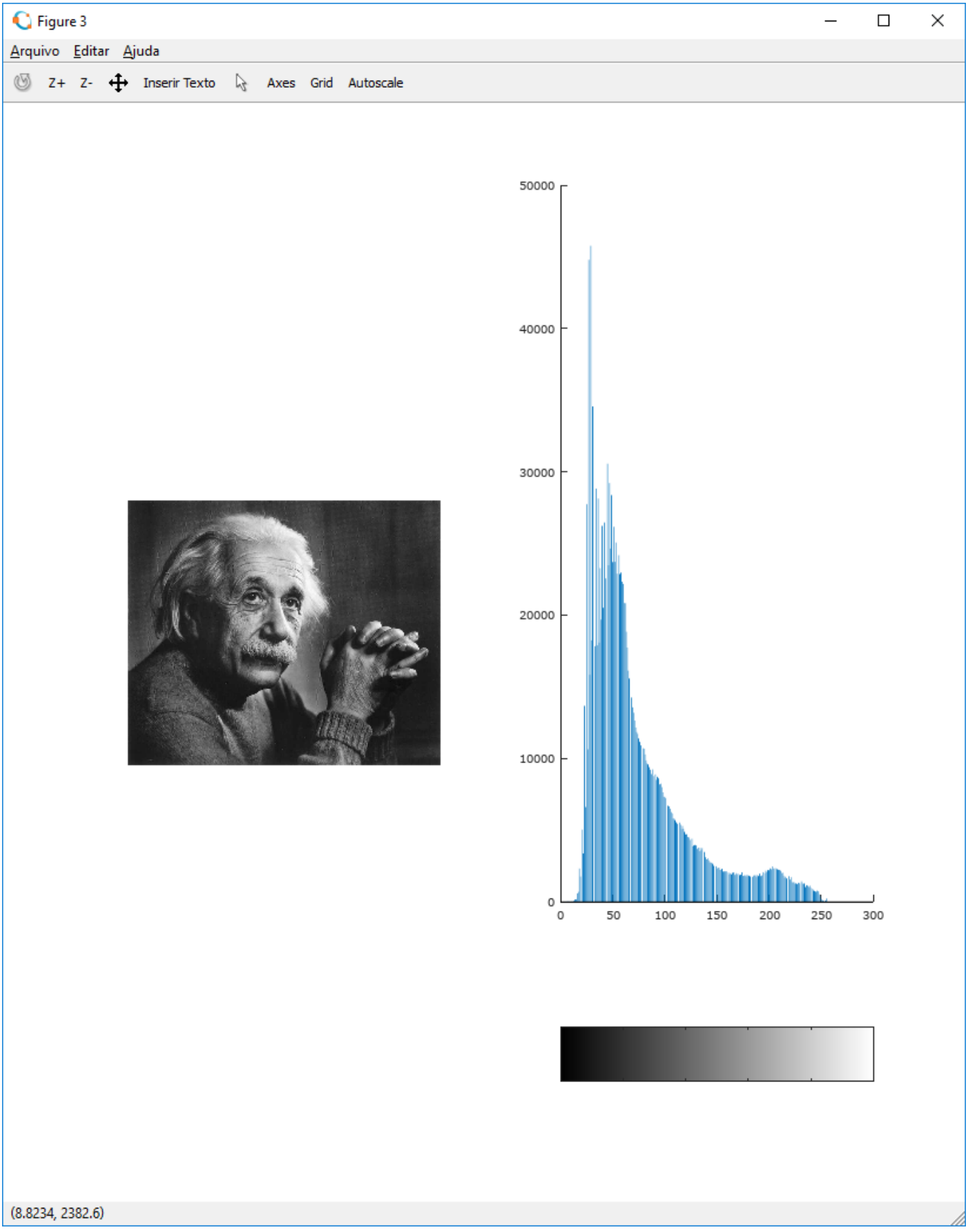
```
>> subplot(1, 2, 1);
>> imshow(elc);
>> subplot(1, 2, 2);
>> imhist(elc);
```



```
>> figure
>> subplot(1, 2, 1);
>> imshow(emc);
>> subplot(1, 2, 2);
>> imhist(emc);
```

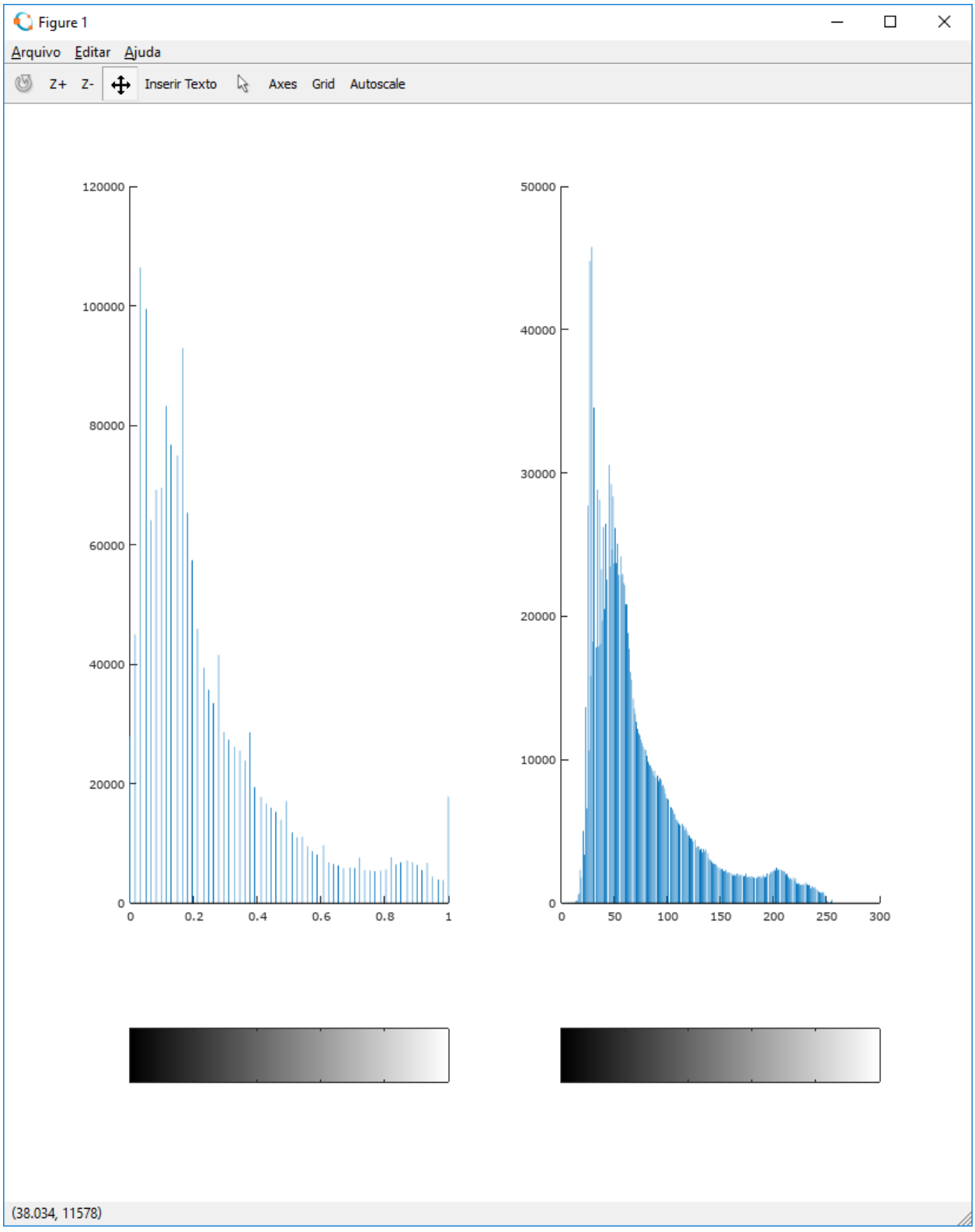


```
>> figure
>> subplot(1, 2, 1);
>> imshow(ehc);
>> subplot(1, 2, 2);
>> imhist(ehc);
```



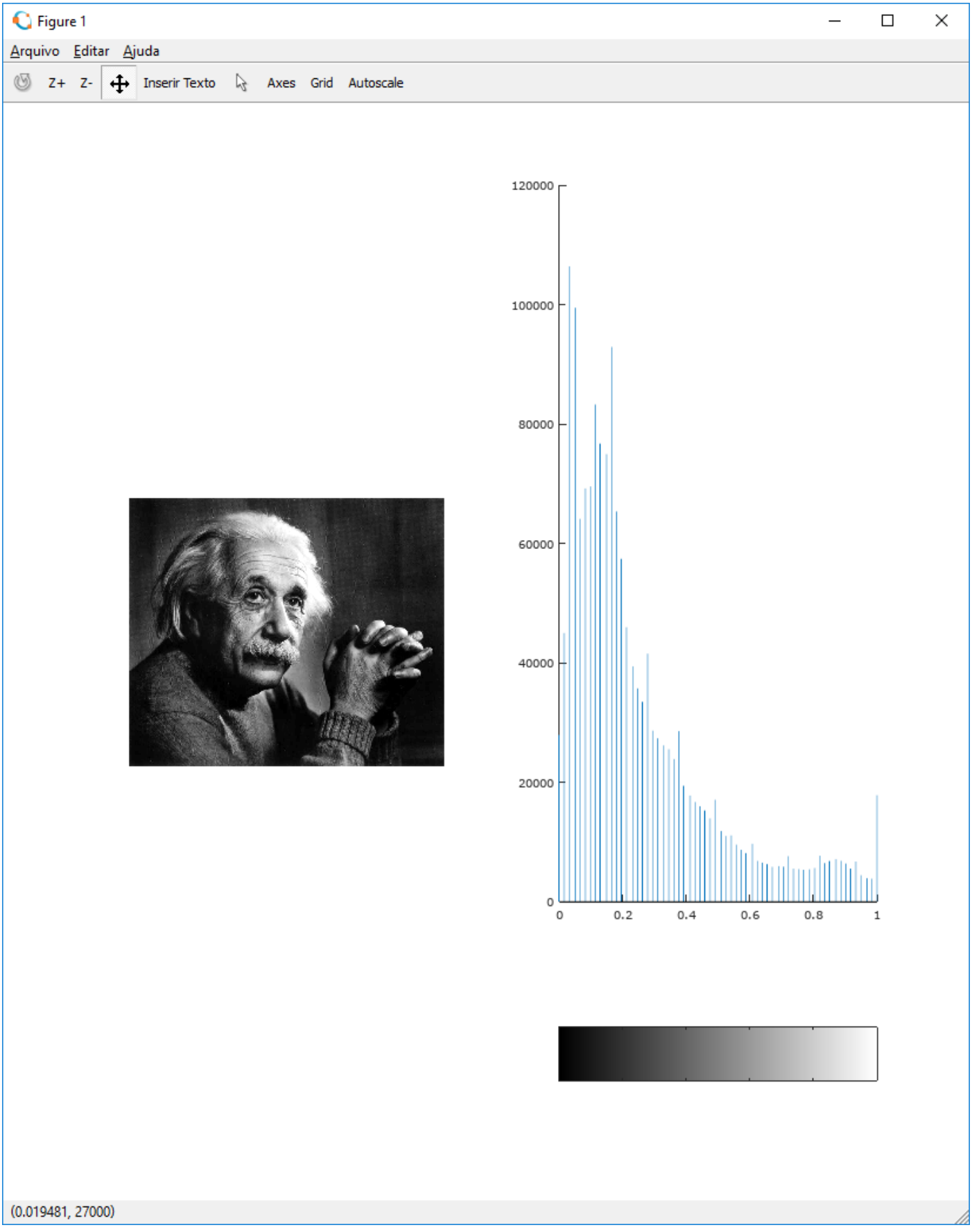
- **Utilize a função *imadjust* para melhorar o contraste da imagem *Einstein_low_contrast.png*, de forma a que o histograma da imagem resultante seja parecido com o da imagem *Einstein_high_contrast.png*.**

```
>> elc_double = im2double(elc);
>> elc_adjusted = imadjust(elc_double);
>> subplot(1,2,1);
>> imhist(elc_adjusted);
>> subplot(1,2,2);
>> imhist(ehc);
```



- **Apresente a nova imagem e seu histograma em uma única figura.**

```
>> subplot(1, 2, 1);
>> imshow(elc_adjusted);
>> subplot(1, 2, 2);
>> imhist(elc_adjusted);
```



Exercício 2:

- **Leia a imagem *leme.bmp***

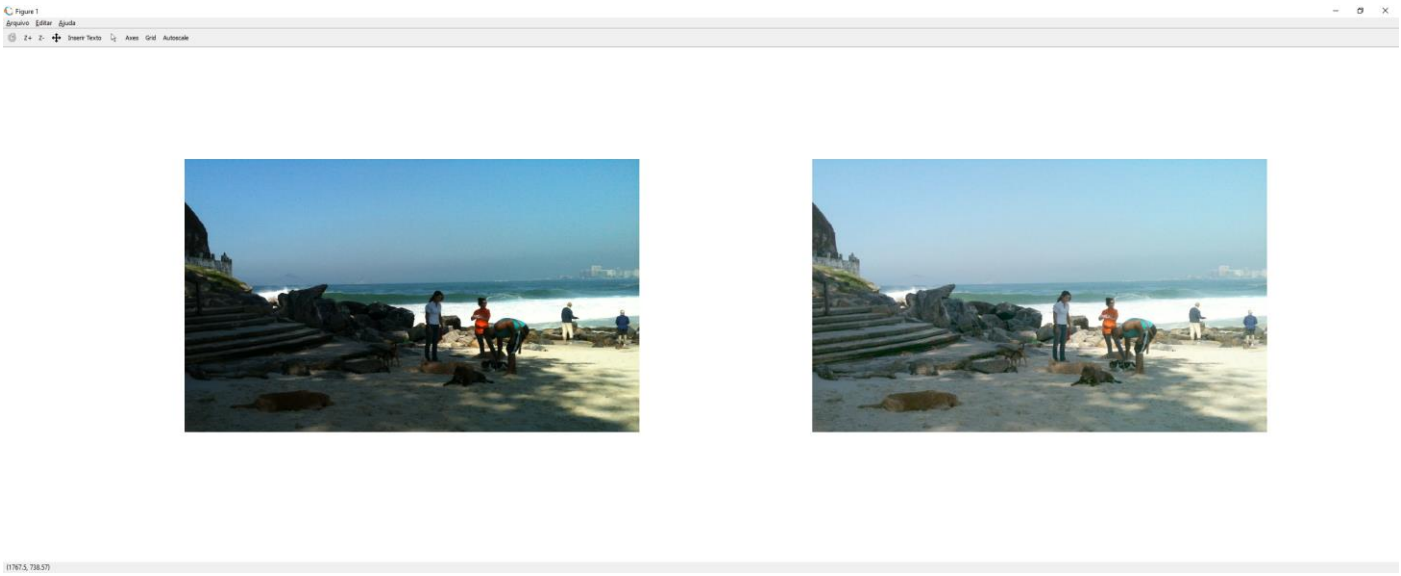
```
>> leme = imread("D:/img/leme.bmp");
```

- **Utilizando a função *imadjust*, crie uma nova imagem colorida em que as áreas escuras da imagem original fiquem mais claras, mas as áreas claras da imagem original não mudem muito de intensidade na nova imagem.**

```
>> leme_double = im2double(leme);
>> leme_adjusted = imadjust(leme_double, [0;1], [0;1], 0.5);
```

- **Apresente a imagem original e a nova imagem em uma única figura.**

```
>> subplot(1,2,1);
>> imshow(leme);
>> subplot(1,2,2);
>> imshow(leme_adjusted);
```



Exercício 3:

- **Leia a ajuda para a função *imfilter*.**

```
>> help imfilter
```

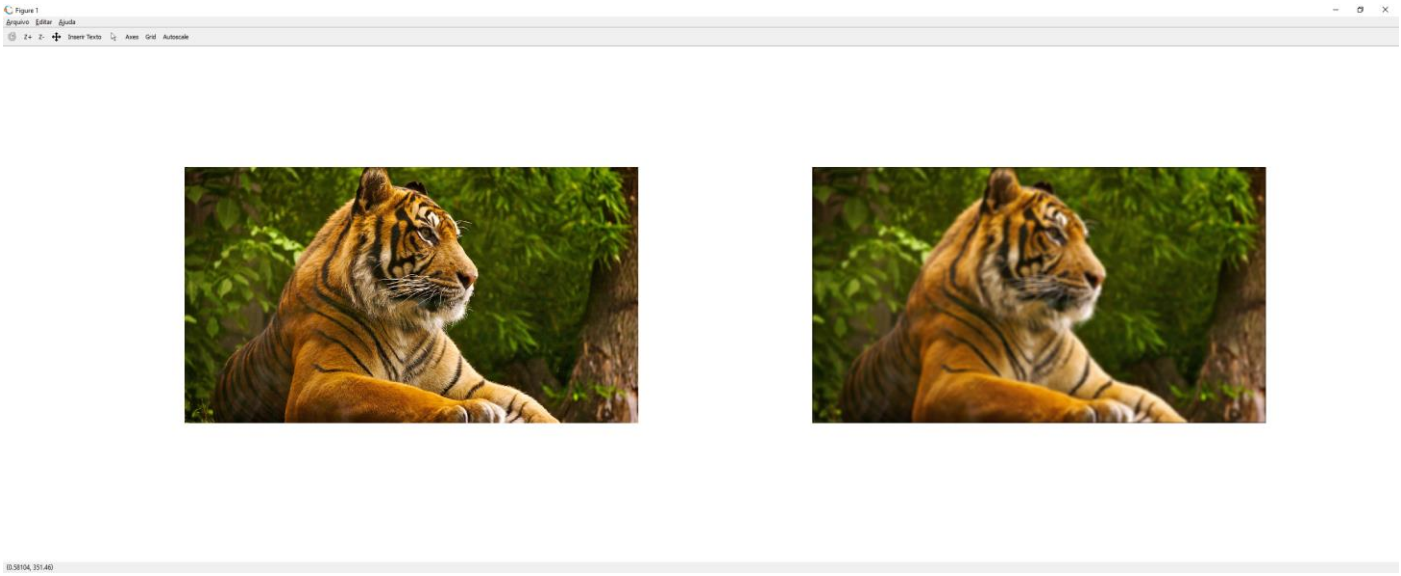
- **Escrever um script que:**

a) Lê uma imagem RGB.
b) Cria uma máscara espacial para suavização linear.
c) Filtra a imagem de entrada (todos os planos) com esta máscara.
d) Mostra as imagem original e filtrada numa única figura.

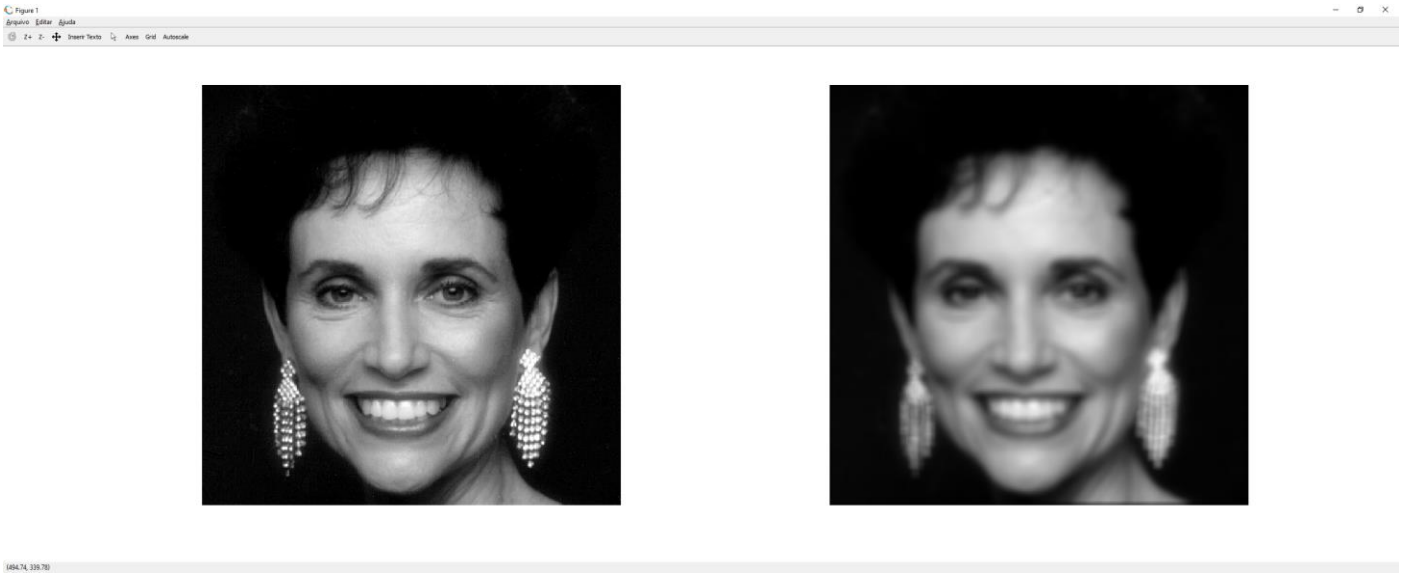
```
pkg load image;

caminhoImagem = input("Digite o caminho da imagem a ser filtrada:\n", 's');
tamanhoMascara = input("Digite o tamanho da mascara a ser utilizada:\n");

imagem = imread(caminhoImagem);

mascara = ones(tamanhoMascara) .* (1/(tamanhoMascara * tamanhoMascara));

imagem_filtrada = imfilter(imagem, mascara);

figure
subplot(1,2,1);
imshow(imagem);
subplot(1,2,2);
imshow(imagem_filtrada);
```

- **Execute o script para pelo menos duas imagens e apresente os resultados**

Com máscara de tamanho 15x15:



Com máscara de tamanho 10x10:



Exercício 4:

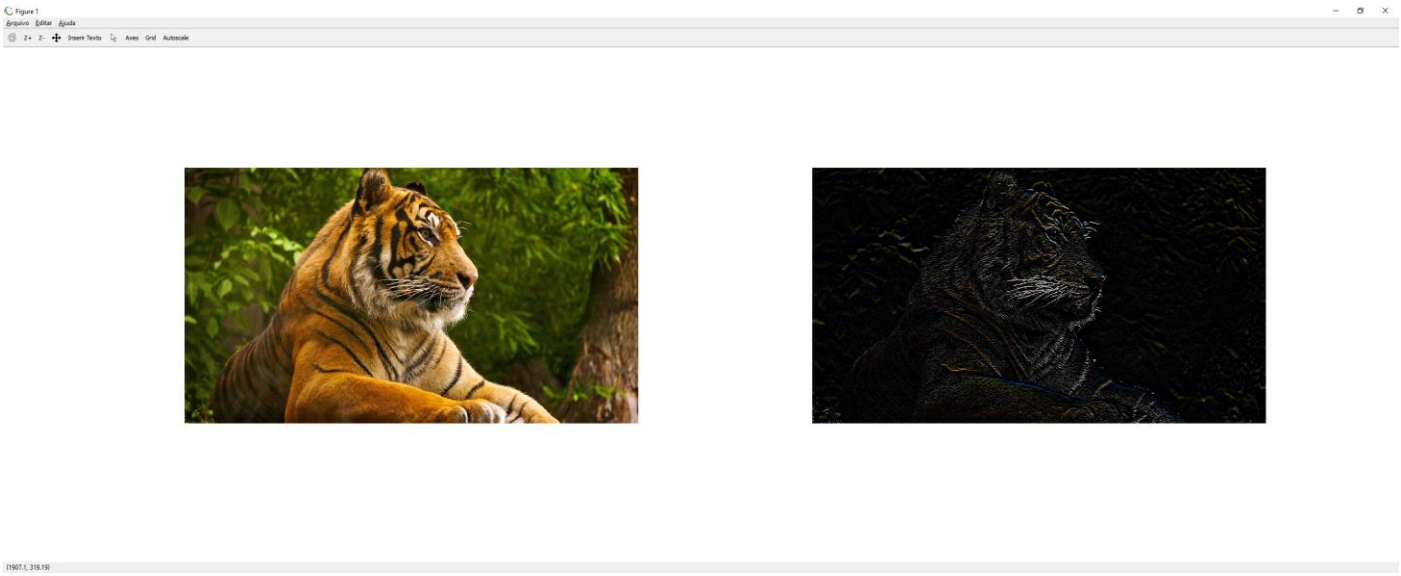
- **Leia a ajuda da função *fspecial* as máscaras disponíveis para filtragem linear.**

```
>> help fspecial
```

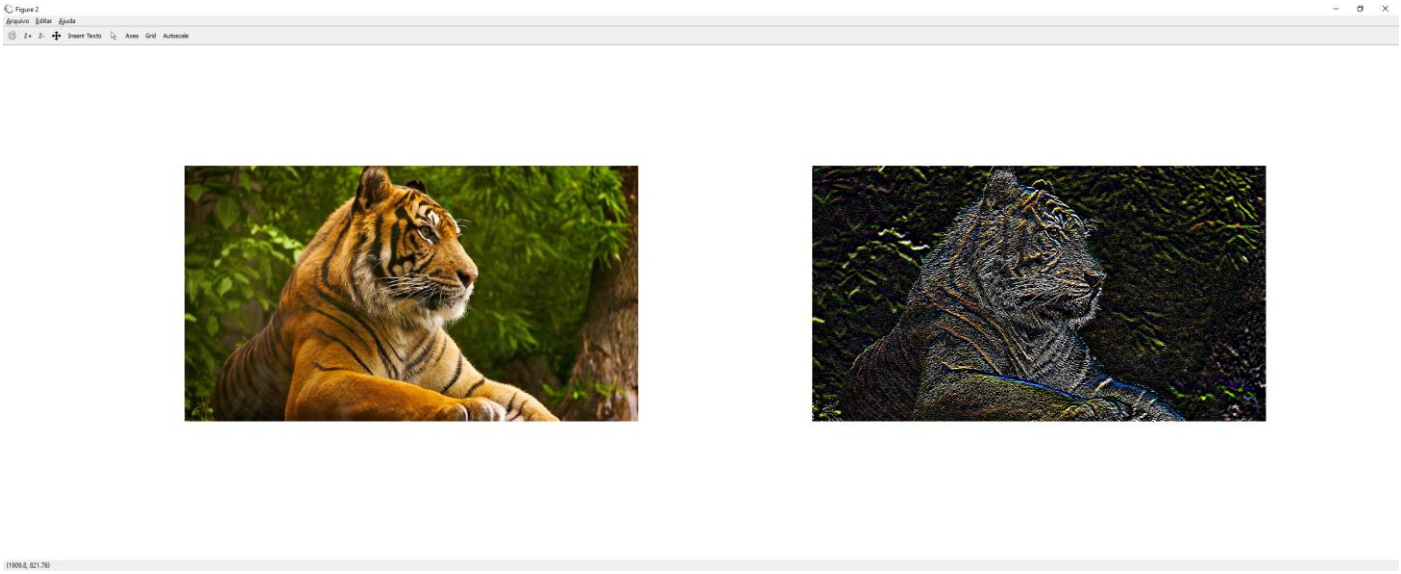
- **Repita o exercício anterior, usando algumas (ao menos 2) das máscaras disponíveis a partir de *fspecial***

```
pkg load image;

caminhoImagem = input("Digite o caminho da imagem a ser filtrada:\n", 's');
disp("\nMascaras disponiveis:\n average, disk, gaussian, log, laplacian, unsharp, motion, sobel, prewitt, kirsch\n")
tipoMascara = input("Digite o tipo da mascara a ser utilizada:\n", 's');

imagem = imread(caminhoImagem);

imagem_filtrada = imfilter(imagem, fspecial(tipoMascara));

figure
subplot(1,2,1);
imshow(imagem);
subplot(1,2,2);
imshow(imagem_filtrada);
```

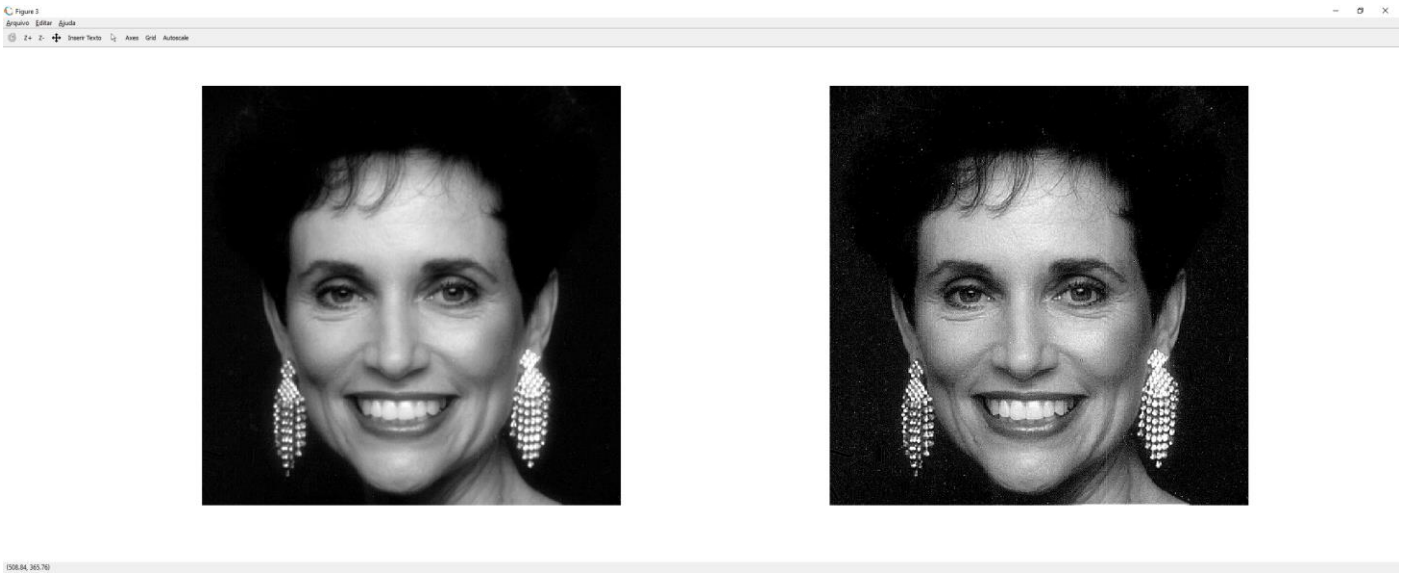
Com máscara de prewitt:



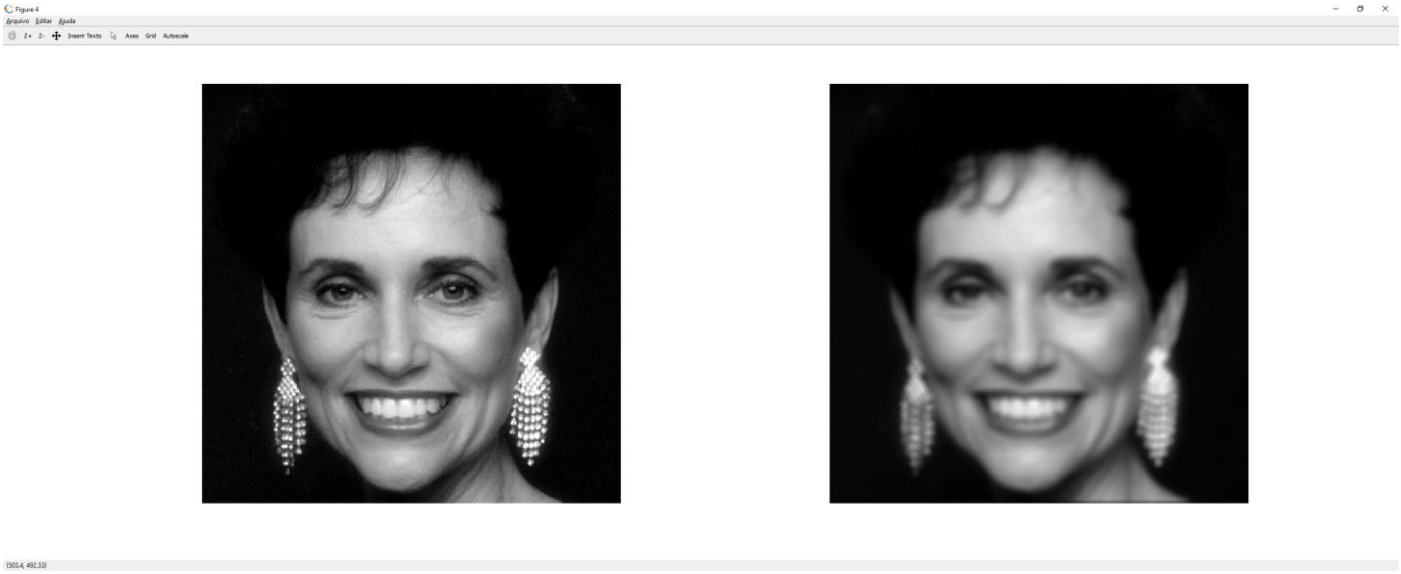
Com máscara de kirsch:



Com máscara de unsharp:



Com máscara de disk:



Exercício 5:

- **Leia a imagem *Coins.png***

```
>> coins = imread("D:/img/coins.png");
```

- **Com ajuda da função imhist identifique um limiar (de intensidade) que melhor separa as moedas do fundo da imagem (*background*).**

```
>> imhist(coins);
```

- **Com o limiar definido, crie uma imagem binária em que os pixels do background das moedas tenham valores diferentes.**

```
>> coins_bin = coins > 85;
```

- **Aplique esta máscara binária à imagem original de forma a mudar a cor do *background* na imagem resultante.**

```
>> coins_out = coins .* coins_bin;
>> imshow(coins_out);
```

- **Apresente a imagem original e a resultante em uma única figura.**

```
>> figure;
>> subplot(1, 2, 1);
>> imshow(coins);
>> subplot(1, 2, 2);
>> imshow(coins_out);
```