

Relatório Octave Tarefa 2

Nome: Thiago Heron Albano de Ávila

1. Introdução

1.1 Definição da Tarefa

Utilizando matrizes, laços de repetição, estruturas de controle e conceitos estudados nas aulas anteriores, crie uma função Matlab/Octave que:

- Receba o nome de uma imagem em escala de cinza
- Receba dois números inteiros que identificam quantas vezes a imagem deve ser redimensionada em cada uma das dimensões (x e y)
- Receba o tipo de interpolação escolhida pelo usuário
 - 1 – Interpolação por Vizinho Mais Próximo
 - 2 – Interpolação Bilinear
- Escreva um novo arquivo com a imagem redimensionada de acordo com o tipo de interpolação escolhido pelo usuário
- Retorne como resultado o nome do novo arquivo gerado.

1.2 Imagem Utilizada

Para a resolução desse problema, será utilizado a seguinte imagem:



2. Executando o Código

Foi criado um arquivo **main.m**, que tem como objetivo:

- Carregar a Imagem Padrão
- Ler o número de redimensionamentos no Eixo X
- Ler o número de redimensionamentos no Eixo Y
- Ler a Interpolação a ser utilizada
 - 1 - Interpolação Vizinho Mais Próximo
 - 2 - Interpolação Bilinear
- Exibir a Imagem Redimensionada
- Salvar a imagem como "**lena_cinza_resized.png**"

Além disso, foi adicionado um arquivo **tests.m**, que contém um lote de execuções de redimensionamento utilizando diferentes parâmetros, que auxiliaram para validar as interpolações, no total são **24 testes**:

- 1 Teste com mesmos tamanhos do **Vizinho Mais Próximo**,
- 8 Testes Aumentando os Eixos X e/ou Y no **Vizinho Mais Próximo**.
- 3 Testes Diminuindo os Eixos X e/ou Y no **Vizinho Mais Próximo**.
- 6 Testes Aumentando os Eixos X e/ou Y na **Bilinear**.
- 3 Testes Diminuindo os Eixos X e/ou Y na **Bilinear**
- 2 Testes Aumentando e Diminuindo Eixos X e/ou Y na **Bilinear**
- 1 Teste com os mesmos tamanhos na **Bilinear**.

Caso seja de interesse executar esses testes, dentro do arquivo **test.m**, foram adicionados comentários especificando o nome de cada figura que será exibida, facilitando a sua visualização e comparação com os parâmetros.

3. Resolução com Interpolação por Vizinho Mais Próximo

A minha solução da Interpolação Linear, primeiramente copiei os pixels da matriz original para a matriz redimensionada nas suas novas posições, ou seja, encontrando as novas posições de acordo com os fatores de redimensionamentos (**num1 e num2**), para isso foi utilizando a sintaxe dos dois pontos.

Após isso, utiliza quatro laços **for** aninhados, onde os dois primeiros percorrem as colunas e linhas da matriz redimensionada, enquanto os outros dois **for** utilizam o fator de expansão para preenchem a matriz com a cor do pixel do vizinho mais próximo.

Esses últimos dois **for** foram controlados com base nos valores de **num1** e **num2** que são os valores de redimensionamento dos eixos x e y, respectivamente.

Em relação a reduzir a imagem com a opção por Vizinho Mais Próximo, a abordagem que foi utilizada, é para cada pixel da matriz original, divide pelo fator de redimensionamento num1 e num2, com base nos resultados, será a nova posição do pixel.

3. Código da Interpolação Bilinear

A minha solução da Interpolação Bilinear, primeiramente adicionei mais uma coluna e uma linha na matriz principal, com base na última linha e coluna, para possibilitar os cálculos das extremidades da matriz.

Após isso, utilizei a função módulo das posições da matriz redimensionada com o fator de redimensionamento, assim, caso o módulo seja zero, o pixel da matriz principal era copiada para a posição nova na matriz redimensionada, caso o módulo fosse maior que zero, era realizado o cálculo da distância entre os quatros pixels para encontrar tx e ty, consequentemente Q1, Q2 e a posição u,v final, conforme os slides da aula.

4. Não Implementado

Infelizmente, somente na Interpolação por Vizinho Mais Próximo, esse código não funcionará caso em um eixo aumente, enquanto no outro eixo esteja diminuindo o tamanho da imagem, ou vice-versa, em outras, a Interpolação por Vizinho Mais Próximo ou só aumenta, ou só diminui, não ambos ao mesmo tempo.

As minhas considerações sobre isso, acredito que não seria difícil modificar, pois tive o entendimento da lógica que do Vizinho Mais Próximo, teria que repartir as estruturas condicionais implementadas, possibilitando que fizesse ambos, ou seja, aumentar e diminuir eixos ao mesmo tempo.

Dado o tempo que demorei para implementar a Bilinear, acabei deixando esse caso citado. Porém, a Interpolação Bilinear funciona para todos os casos.

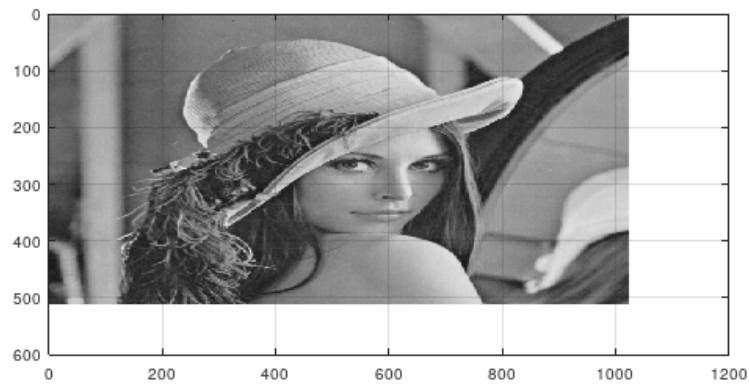
5. Imagens dos Resultados

A seguir, será demonstrado as imagens geradas de acordo com cada parametrização, tanto para Interpolação por Vizinho Mais Próximo, quanto Interpolação Bilinear.

5.1 Resultados da Interpolação por Vizinho Mais Próximo

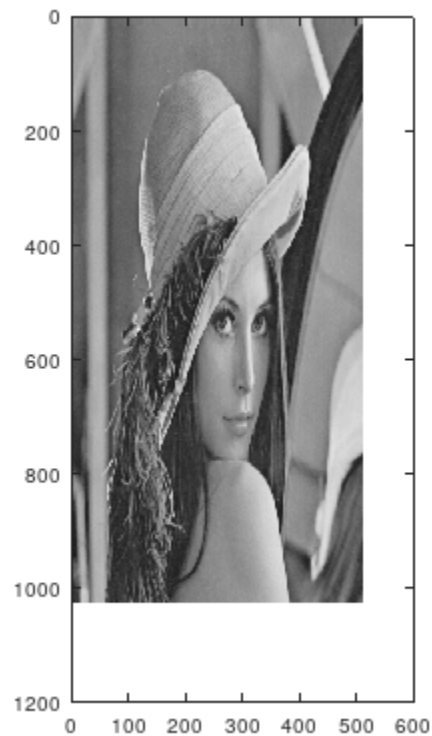
5.1.1 Aumentando a Imagem apenas no Eixo X

```
redimensionar("lena_cinza.bmp", 1, 2, 1);
```



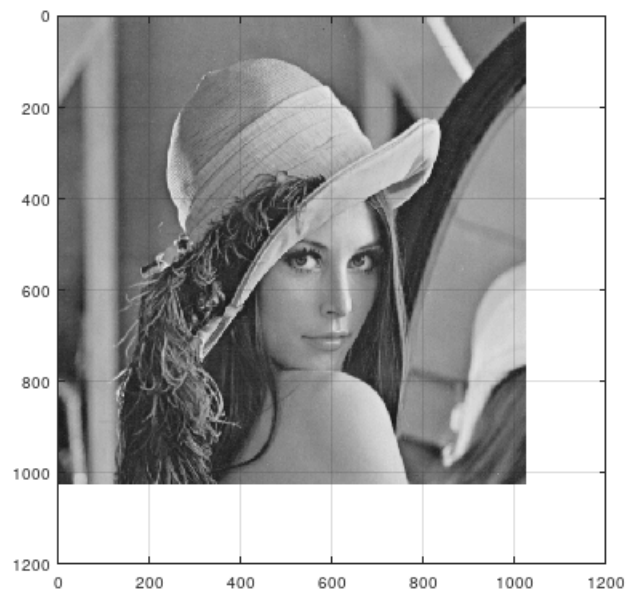
5.1.2 Aumentando a Imagem apenas no Eixo Y

```
image_resized = redimensionar("lena_cinza.bmp", 1, 1, 2);
```



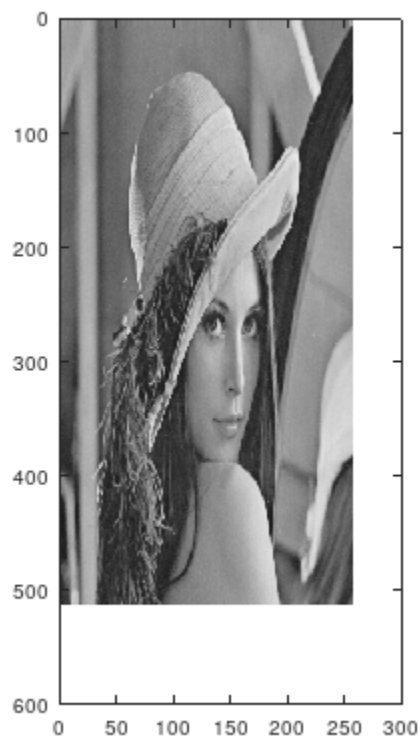
5.1.3 Aumentando em Ambos Eixos, X e Y

```
image_resized = redimensionar("lena_cinza.bmp", 1, 2, 2);
```



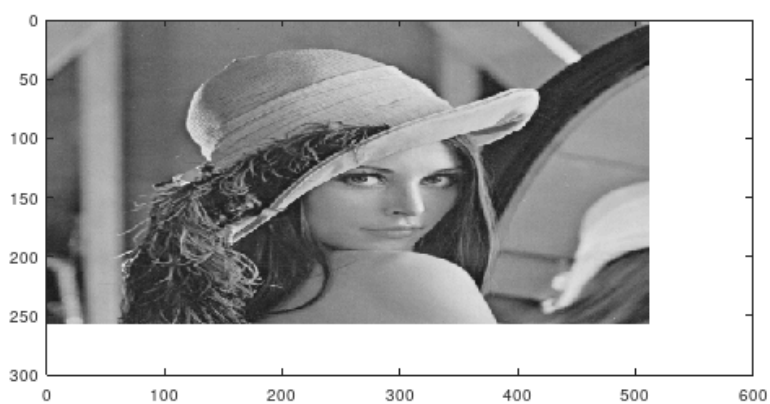
5.1.4 Diminuindo a Imagem no Eixo X

```
image_resized = redimensionar("lena_cinza.bmp", 1, 0.5, 1);
```



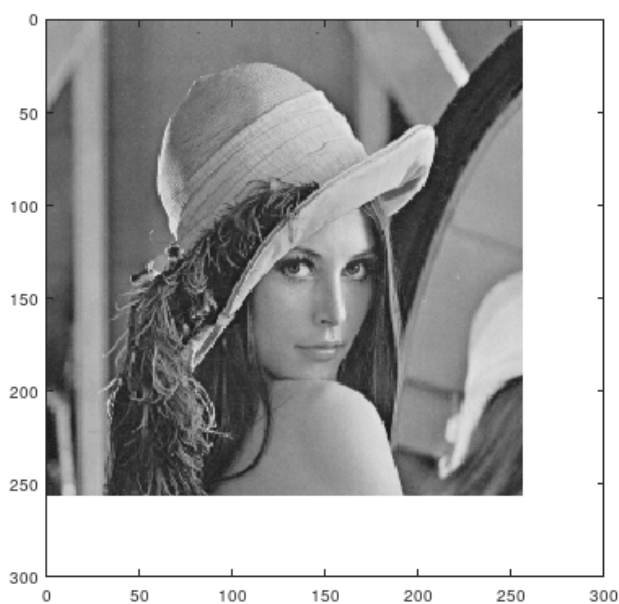
5.1.5 Diminuindo a Imagem no Eixo Y

```
image_resized = redimensionar("lena_cinza.bmp", 1, 1, 0.5);
```



5.1.6 Diminuindo em Ambos Eixos, X e Y

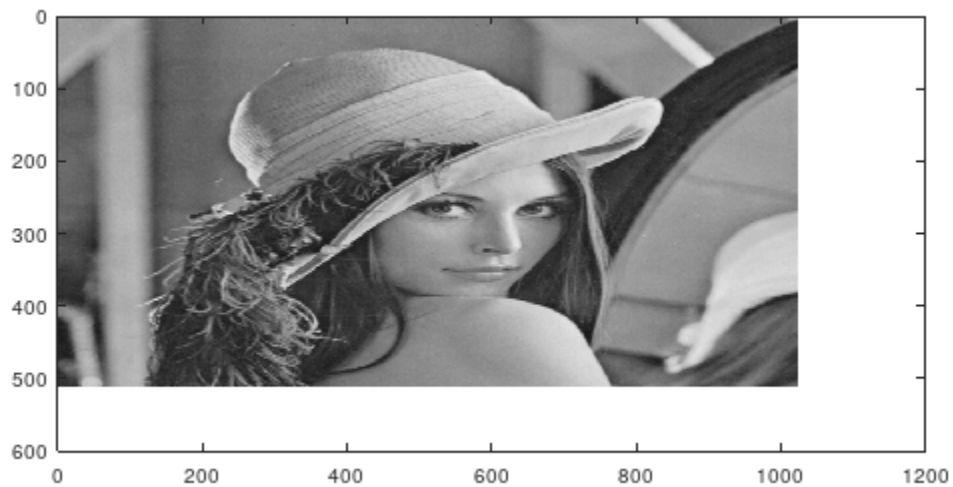
```
image_resized = redimensionar("lena_cinza.bmp", 1, 0.5, 0.5);
```



5.2 Resultados da Interpolação Bilinear

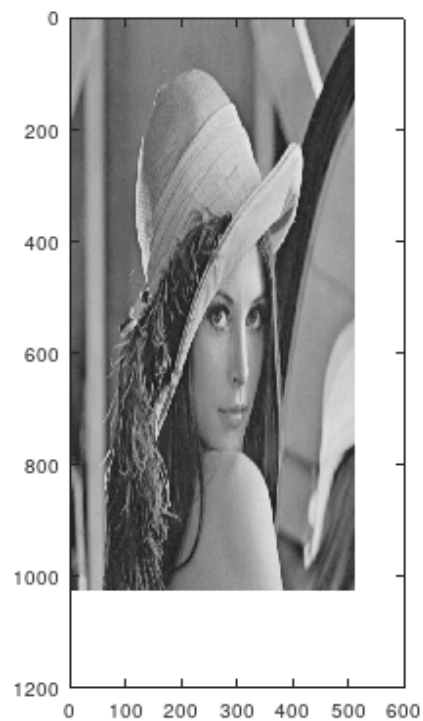
5.1.1 Aumentando a Imagem apenas no Eixo X

```
redimensionar("lena_cinza.bmp", 2, 2, 1);
```



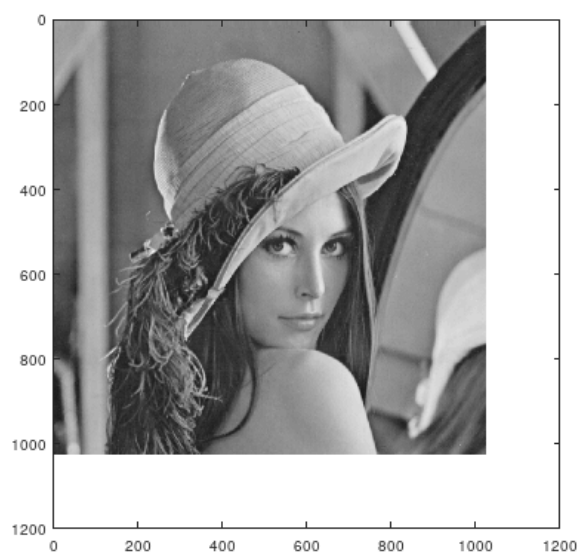
5.1.2 Aumentando a Imagem apenas no Eixo Y

```
redimensionar("lena_cinza.bmp", 2, 1, 2);
```



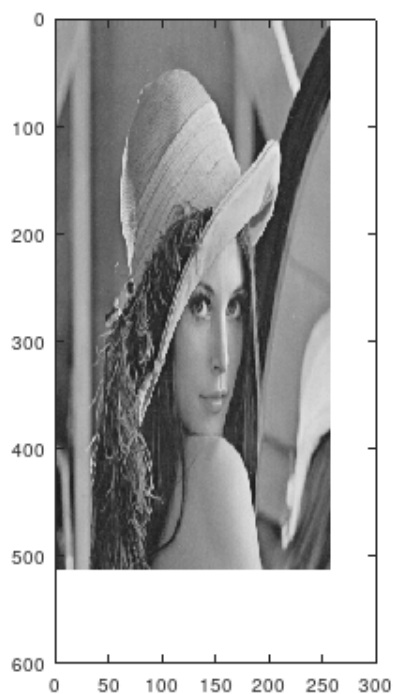
5.1.3 Aumentando em Ambos Eixos, X e Y

```
redimensionar("lena_cinza.bmp", 2, 2, 2);
```



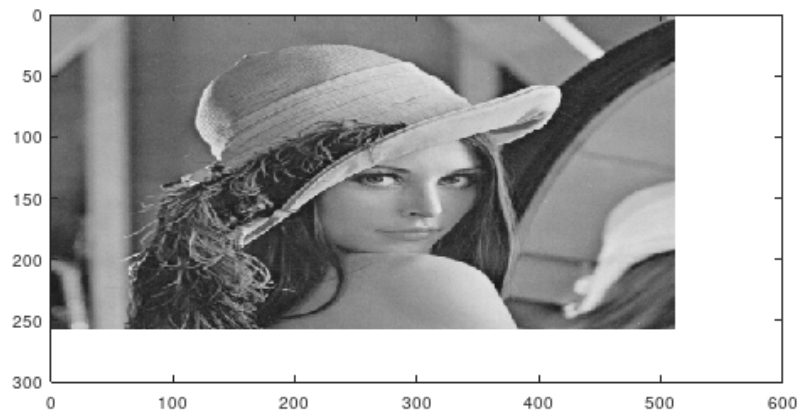
5.1.4 Diminuindo a Imagem apenas no Eixo X

```
redimensionar("lena_cinza.bmp", 2, 0.5, 1);
```



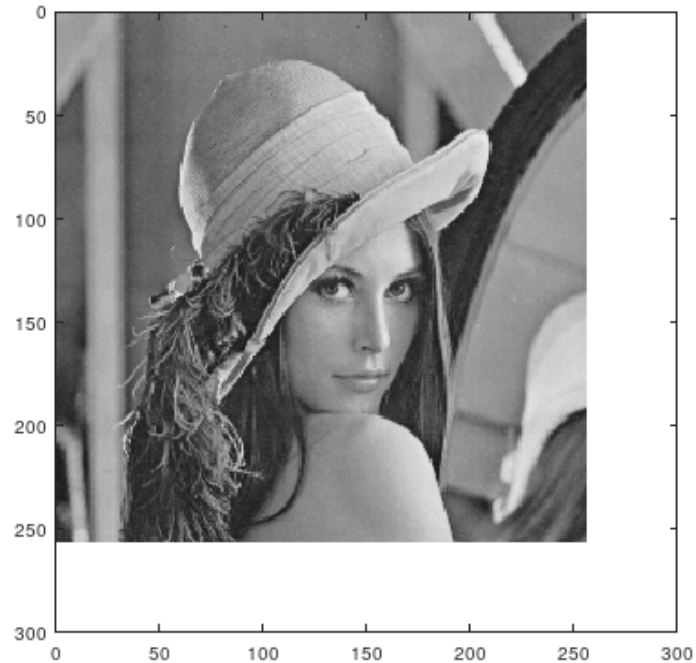
5.1.4 Diminuindo a Imagem apenas no Eixo Y

```
redimensionar("lena_cinza.bmp", 2, 1, 0.5);
```



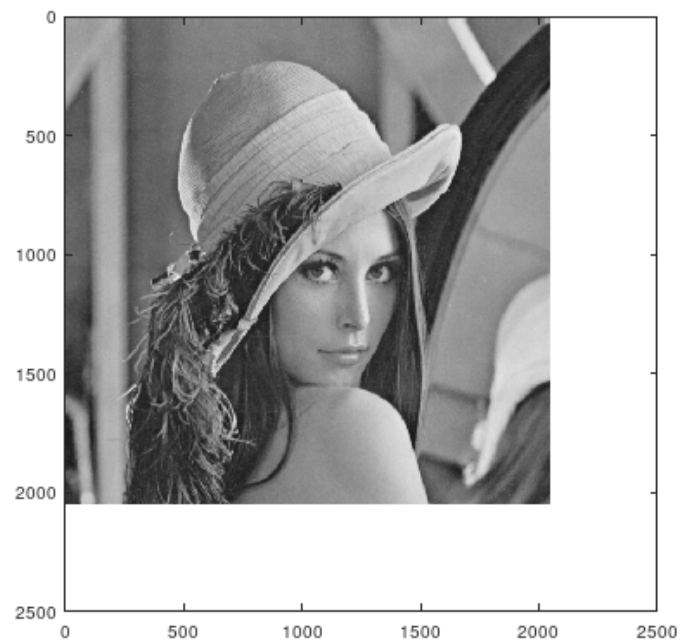
5.1.4 Diminuindo a Imagem em Ambos Eixos, X e Y.

```
redimensionar("lena_cinza.bmp", 2, 0.5, 0.5);
```

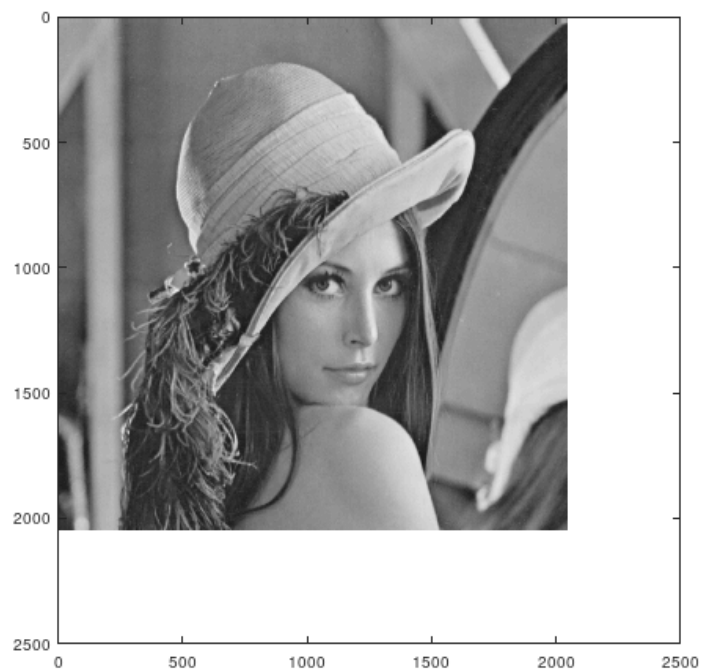


6. Comparações entre Vizinho Mais Próximo e Bilinear

Interpolação Vizinho Mais Próximo - redimensionar("lena_cinza.bmp", 1, 4, 4);



Interpolação Bilinear - redimensionar("lena_cinza.bmp", 2, 4, 4);



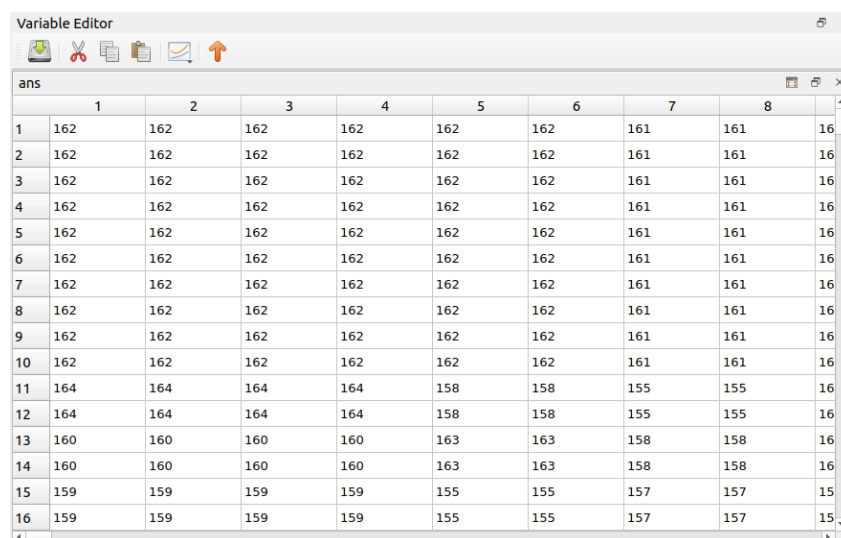
8. Conclusões

Observei que, a Interpolação por Vizinheiro Mais Próximo dá um aspecto da imagem mais “pixelizado”, enquanto na Interpolação Bilinear aparenta ficar mais “suave” os tons de cinza.

Em geral, primeira vez que tenho contato com esse tipo de conhecimento, fiquei muito surpreso pela lógica por trás dos redimensionamentos. Notei também, uma certa demora para redimensionar imagens com uma parametrização mais elevada.

9. Extra: Recurso do Octave “Variable Editor”

Para a resolução desses exercícios, utilizei a aba do Octave “Variable Editor” que possibilita verificar quais os valores que são armazenados durante a execução do programa, assim me possibilitou entender se os valores que foram preenchidos na matriz estavam corretos ou de acordo com o que era esperado. Esse recurso eu ainda não conhecia ainda na Tarefa 1, resolvi acrescentar.



	1	2	3	4	5	6	7	8	
1	162	162	162	162	162	162	161	161	16
2	162	162	162	162	162	162	161	161	16
3	162	162	162	162	162	162	161	161	16
4	162	162	162	162	162	162	161	161	16
5	162	162	162	162	162	162	161	161	16
6	162	162	162	162	162	162	161	161	16
7	162	162	162	162	162	162	161	161	16
8	162	162	162	162	162	162	161	161	16
9	162	162	162	162	162	162	161	161	16
10	162	162	162	162	162	162	161	161	16
11	164	164	164	164	158	158	155	155	16
12	164	164	164	164	158	158	155	155	16
13	160	160	160	160	163	163	158	158	16
14	160	160	160	160	163	163	158	158	16
15	159	159	159	159	155	155	157	157	15
16	159	159	159	159	155	155	157	157	15