5)

5.1) Metodologia

Para a realização da questão 5 foram criados seis script (funções) em MATLAB. Os quatro primeiros scripts realizam o escalamento da imagem por fatores de 2 e de 3, utilizando as técnicas de interpolação de ordem zero (Nearest Neighbor) e de primeira ordem (Bilinear). Os dois últimos scripts realizam a rotação de uma imagem, novamente, utilizando os métodos de interpolação já mencionados.

Ao ser feito o processo de escala da imagem 'Lenna', nota-se uma leve otimização da resolução da imagem resultante pelo processo de interpolação Bilinear, frente ao algoritmo de interpolação Nearest Neighbor. Também constatamos que as imagens geradas pelo processo Nearest Neighbor ocupam menos espaço no dispositivo de armazenamento secundário, quando comparadas com as imagens geradas pelo algoritmo Bilinear. Estes resultados já eram esperados, uma vez que o método Bilinear é mais complexo, demanda mais operações matemáticas para ser realizado e gera menos redundâncias na imagem final. Quanto aos testes de rotações, é possível ver uma introdução de aliasing em nível mais elevado ao utilizarmos a rotação com interpolação Nearest Neighbot do que quando realizado a rotação com interpolação Bilinear.

A realização desta questão nos possibilitou pôr em prática e fixar diversos conceitos visto em nossas aulas, que foram: programação em MATLAB, escala de imagens, métodos de interpolações (0 e 1ªordem), translações e rotações de imagens.

5.2) Códigos

5.2.1) nearest_neighbor_scale2(I)

```
function Iout = nearest_neighbor_scale2(I)
[rows, columns, numberOfColorChannels] = size(I);
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Amplia o tamanho dos três canais de cores
Ir_out = ones(2*rows, 2*columns);
Ig_out = ones(2*rows, 2*columns);
Ib_out = ones(2*rows, 2*columns);
% Mapeamento da imagem de saída para a imagem de entrada
% Rows
x0_rows = 1;
y0_rows = 1;
m_rows = (rows - 1) / (2*rows - 1);
function y = f_rows(x)
y = y0_rows + (x - x0_rows)*m_rows;
end
% Columns
x0_columns = 1;
y0_columns = 1;
m_columns = (columns - 1) / (2*columns - 1);
function y = f_columns(x)
y = y0_columns + (x - x0_columns)*m_columns;
end
```

```
% Interpolação de ordem zero
for i=1:(2*rows)
for j=1:(2*columns)
y = round(f_rows(i));
x = round(f_columns(j));
Ir_out(i, j) = Ir(y, x);
Ig_out(i, j) = Ig(y, x);
Ib_out(i, j) = Ib(y, x);
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.2) nearest_neighbor_scale3(I)

```
function Iout = nearest_neighbor_scale3(I)
[rows, columns, numberOfColorChannels] = size(I);
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Amplia o tamanho dos três canais de cores
Ir_out = ones(3*rows, 3*columns);
Ig_out = ones(3*rows, 3*columns);
Ib_out = ones(3*rows, 3*columns);
% Mapeamento da imagem de saída para a imagem de entrada
% Rows
x0_rows = 1;
y0_rows = 1;
m_rows = (rows - 1) / (3*rows - 1);
function y = f_rows(x)
y = y0_rows + (x - x0_rows)*m_rows;
end
% Columns
x0_columns = 1;
y0_columns = 1;
m_columns = (columns - 1) / (3*columns - 1);
function y = f_columns(x)
y = y0_columns + (x - x0_columns)*m_columns;
end
% Interpolação de ordem zero
for i=1:(3*rows)
for j=1:(3*columns)
y = round(f_rows(i));
x = round(f_columns(j));
Ir_out(i, j) = Ir(y, x);
Ig_out(i, j) = Ig(y, x);
Ib_out(i, j) = Ib(y, x);
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.3) bilinear_scale2(I)

```
function Iout = bilinear_scale2(I)
[rows, columns, numberOfColorChannels] = size(I);
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Amplia o tamanho dos três canais de cores
Ir_out = ones(2*rows, 2*columns);
Ig_out = ones(2*rows, 2*columns);
Ib_out = ones(2*rows, 2*columns);
% Mapeamento da imagem de saída para a imagem de entrada
% Rows
x0_rows = 1;
y0_rows = 1;
m_rows = (rows - 1) / (2*rows - 1);
function y = f_rows(x)
y = y0_rows + (x - x0_rows)*m_rows;
end
% Columns
x0_columns = 1;
y0_columns = 1;
m_columns = (columns - 1) / (2*columns - 1);
function y = f_columns(x)
y = y0_columns + (x - x0_columns)*m_columns;
end
% Interpola os valores originais para a nova imagem
for i=1:(2*rows)
for j=1:(2*columns)
y = f_rows(i);
x = f_columns(j);
a = y - floor(y);
b = x - floor(x);
Ir_out(i, j) = (1-a)*((1-b)*Ir(floor(y), floor(x)) + b*Ir(floor(y),
ceil(x))) + a*((1-b)*Ir(ceil(y), floor(x)) + b*Ir(ceil(y), ceil(x)));
Ig_out(i, j) = (1-a)*((1-b)*Ig(floor(y), floor(x)) + b*Ig(floor(y),
ceil(x))) + a*((1-b)*Ig(ceil(y), floor(x)) + b*Ig(ceil(y), ceil(x)));
Ib_out(i, j) = (1-a)*((1-b)*Ib(floor(y), floor(x)) + b*Ib(floor(y),
ceil(x))) + a*((1-b)*Ib(ceil(y), floor(x)) + b*Ib(ceil(y), ceil(x)));
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.4) bilinear_scale3(I)

```
function Iout = bilinear_scale3(I);
[rows, columns, numberOfColorChannels] = size(I);
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Amplia o tamanho dos três canais de cores
Ir_out = ones(3*rows, 3*columns);
Ig_out = ones(3*rows, 3*columns);
Ib_out = ones(3*rows, 3*columns);
% Mapeamento da imagem de saída para a imagem de entrada
% Rows
x0_rows = 1;
y0_rows = 1;
m_rows = (rows - 1) / (3*rows - 1);
function y = f_rows(x)
y = y0_rows + (x - x0_rows)*m_rows;
end
% Columns
x0_columns = 1;
y0_columns = 1;
m_columns = (columns - 1) / (3*columns - 1);
function y = f_columns(x)
y = y0_columns + (x - x0_columns)*m_columns;
end
% Copia os valores originais para a nova imagem
for i=1:(3*rows)
for j=1:(3*columns)
y = f_rows(i);
x = f_columns(j);
a = y - floor(y);
b = x - floor(x);
Ir_out(i, j) = (1-a)*((1-b)*Ir(floor(y), floor(x)) + b*Ir(floor(y),
ceil(x))) + a*((1-b)*Ir(ceil(y), floor(x)) + b*Ir(ceil(y), ceil(x)));
Ig_out(i, j) = (1-a)*((1-b)*Ig(floor(y), floor(x)) + b*Ig(floor(y),
ceil(x))) + a*((1-b)*Ig(ceil(y), floor(x)) + b*Ig(ceil(y), ceil(x)));
Ib_out(i, j) = (1-a)*((1-b)*Ib(floor(y), floor(x)) + b*Ib(floor(y),
ceil(x))) + a*((1-b)*Ib(ceil(y), floor(x)) + b*Ib(ceil(y), ceil(x)));
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.5) nearest_neighbor_rotation

```
function Iout = nearest_neighbor_rotation(I, angle)
[rows, columns, numberOfColorChannels] = size(I);
angle = -angle;
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Recriação dos três canais de cores
Ir_out = zeros(rows, columns);
Ig_out = zeros(rows, columns);
Ib_out = zeros(rows, columns);
% Mapeamento da imagem de saída para a imagem de entrada
s = sin(angle);
c = cos(angle);
% Matriz de Rotação
R = [c -s;
s c];
% Matriz de Translação
T = [-columns/2 -rows/2];
% Função que realiza as mudanças de coordenadas e a rotação
function W = f(x, y)
X = [y x];
X = X + T;
W = R'*X';
W = W - T';
end
% Interpolação
for i=1:(rows)
for j=1:(columns)
POSICAO = round(f(i, j));
if POSICAO(1) <= 0 || POSICAO(1) > (columns) || POSICAO(2) <= 0 ||
POSICAO(2) > (rows)
Ir_out(i, j) = 0;
Ig_out(i, j) = 0;
Ib_out(i, j) = 0;
else
Ir_out(i, j) = Ir(POSICAO(2), POSICAO(1));
Ig_out(i, j) = Ig(POSICAO(2), POSICAO(1));
Ib_out(i, j) = Ib(POSICAO(2), POSICAO(1));
end
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.6) bilinear_rotation

```
function Iout = bilinear_rotation(I, angle)
[rows, columns, numberOfColorChannels] = size(I);
angle = -angle;
Ir = I(:, :, 1);
Ig = I(:, :, 2);
Ib = I(:, :, 3);
% Recriação dos três canais de cores
Ir_out = zeros(rows, columns);
Ig_out = zeros(rows, columns);
Ib_out = zeros(rows, columns);
% Mapeamento da imagem de saída para a imagem de entrada
s = sin(angle);
c = cos(angle);
% Matriz de Rotação
R = [c -s;
s c];
% Matriz de Translação
T = [-columns/2 -rows/2];
% Função que realiza as mudanças de coordenadas e a rotação
function W = f(x, y)
X = [y x];
X = X + T;
W = R'*X';
W = W - T';
end
% Interpolação
for i=1:(rows)
for j=1:(columns)
POSICAO = f(i, j);
a = POSICAO(2) - floor(POSICAO(2));
b = POSICAO(1) - floor(POSICAO(1));
if POSICAO(1) < 1 || POSICAO(1) > (columns) || POSICAO(2) < 1 || POSICAO(2)
> (rows)
Ir_out(i, j) = 0;
Ig_out(i, j) = 0;
Ib_out(i, j) = 0;
else
Ir_out(i, j) = (1-a)*((1-b)*Ir(floor(POSICAO(2)), floor(POSICAO(1))) +
b*Ir(floor(POSICAO(2)), ceil(POSICAO(1)))) + a*((1-b)*Ir(ceil(POSICAO(2)),
floor(POSICAO(1))) + b*Ir(ceil(POSICAO(2)), ceil(POSICAO(1))));
Ig_out(i, j) = (1-a)*((1-b)*Ig(floor(POSICAO(2)), floor(POSICAO(1))) +
b*Ig(floor(POSICAO(2)), ceil(POSICAO(1)))) + a*((1-b)*Ig(ceil(POSICAO(2)),
floor(POSICAO(1))) + b*Ig(ceil(POSICAO(2)), ceil(POSICAO(1))));
Ib_out(i, j) = (1-a)*((1-b)*Ib(floor(POSICAO(2)), floor(POSICAO(1))) +
b*Ib(floor(POSICAO(2)), ceil(POSICAO(1)))) + a*((1-b)*Ib(ceil(POSICAO(2)),
floor(POSICAO(1))) + b*Ib(ceil(POSICAO(2)), ceil(POSICAO(1))));
end
end
end
Iout(:,:,1) = Ir_out;
Iout(:,:,2) = Ig_out;
Iout(:,:,3) = Ib_out;
end
```

5.2.7) Q5

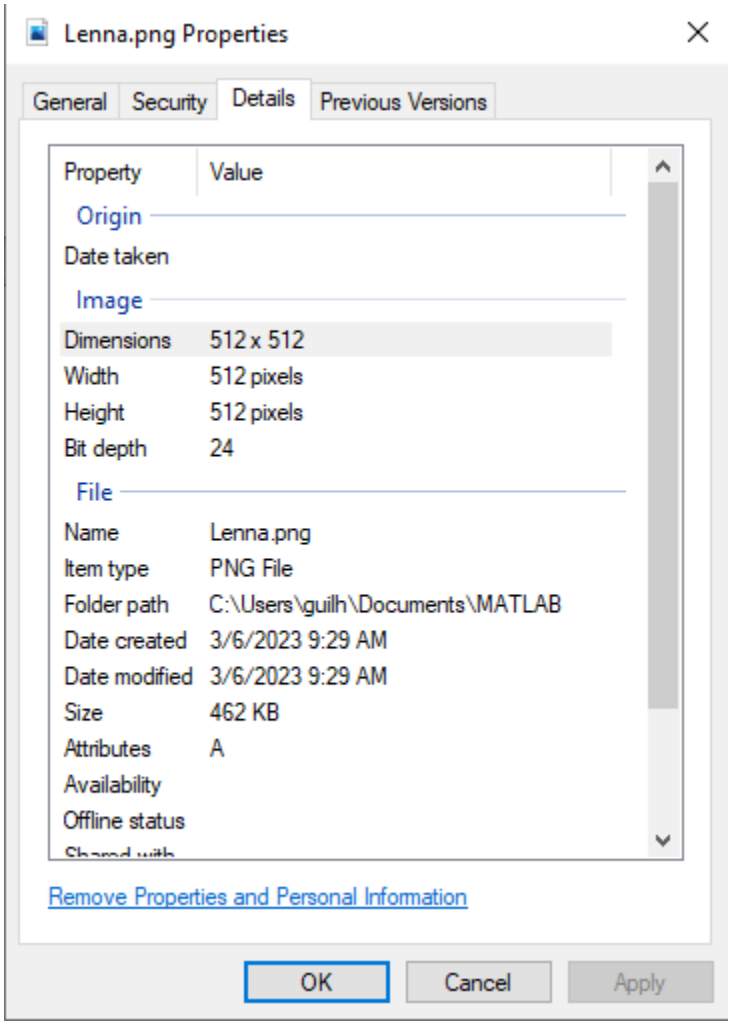
(Script para visualizar os testes feitos. Descomente os comandos na parte inferior do código, onde diz “% Testes”)

```
% Script para Questão 5
I = imread('autumn.tif');
%I = imread('Lenna.png');
% Escalas
I2_nearest = nearest_neighbor_scale2(I);
I2_bilinear = bilinear_scale2(I);
I3_nearest = nearest_neighbor_scale3(I);
I3_bilinear = bilinear_scale3(I);
% Rotações
I2_nearest_rotated5 = bilinear_rotation(I2_nearest, pi/36);
I2_nearest_rotated45 = bilinear_rotation(I2_nearest, pi/4);
I2_bilinear_rotated5 = bilinear_rotation(I2_bilinear, pi/36);
I2_bilinear_rotated45 = bilinear_rotation(I2_bilinear, pi/4);
I3_nearest_rotated5 = bilinear_rotation(I3_nearest, pi/36);
I3_nearest_rotated45 = bilinear_rotation(I3_nearest, pi/4);
I3_bilinear_rotated5 = bilinear_rotation(I3_bilinear, pi/36);
I3_bilinear_rotated45 = bilinear_rotation(I3_bilinear, pi/4);
I2_nearest_rotated5_nearest = nearest_neighbor_rotation(I2_nearest, pi/36);
I2_nearest_rotated45_nearest = nearest_neighbor_rotation(I2_nearest, pi/4);
I2_bilinear_rotated5_nearest = nearest_neighbor_rotation(I2_bilinear, pi/36);
I2_bilinear_rotated45_nearest = nearest_neighbor_rotation(I2_bilinear, pi/4);
I3_nearest_rotated5_nearest = nearest_neighbor_rotation(I3_nearest, pi/36);
I3_nearest_rotated45_nearest = nearest_neighbor_rotation(I3_nearest, pi/4);
I3_bilinear_rotated5_nearest = nearest_neighbor_rotation(I3_bilinear, pi/36);
I3_bilinear_rotated45_nearest = nearest_neighbor_rotation(I3_bilinear, pi/4);
% Testes
% figure, imshow(I)
%
% Escala 2X
%
% figure, imshow(uint8(I2_nearest))
% figure, imshow(uint8(I2_bilinear))
%
% Rotação Bilinear
%
% figure, imshow(uint8(I2_bilinear_rotated45))
% figure, imshow(uint8(I2_bilinear_rotated5))
% figure, imshow(uint8(I2_nearest_rotated45))
% figure, imshow(uint8(I2_nearest_rotated5))
%
% Rotação Nearest
%
% figure, imshow(uint8(I2_bilinear_rotated45_nearest))
% figure, imshow(uint8(I2_bilinear_rotated5_nearest))
% figure, imshow(uint8(I2_nearest_rotated45_nearest))
% figure, imshow(uint8(I2_nearest_rotated5_nearest))
%
%
% 3X
%
% figure, imshow(uint8(I3_bilinear))
% figure, imshow(uint8(I3_nearest))
%
```

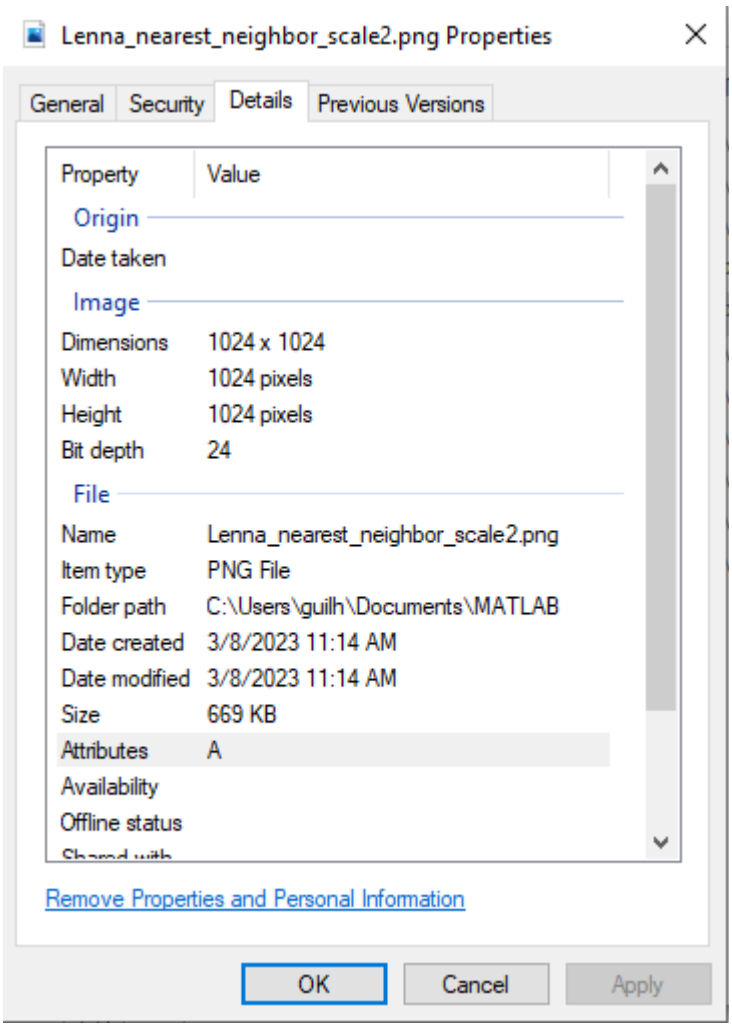
```
% Rotação Bilinear
%
% figure, imshow(uint8(I3_bilinear_rotated45))
% figure, imshow(uint8(I3_bilinear_rotated5))
% figure, imshow(uint8(I3_nearest_rotated45))
% figure, imshow(uint8(I3_nearest_rotated5))
%
% Rotação Nearest
%
% figure, imshow(uint8(I3_bilinear_rotated45_nearest))
% figure, imshow(uint8(I3_bilinear_rotated5_nearest))
% figure, imshow(uint8(I3_nearest_rotated45_nearest))
% figure, imshow(uint8(I3_nearest_rotated5_nearest))
```

5.3) Resultados

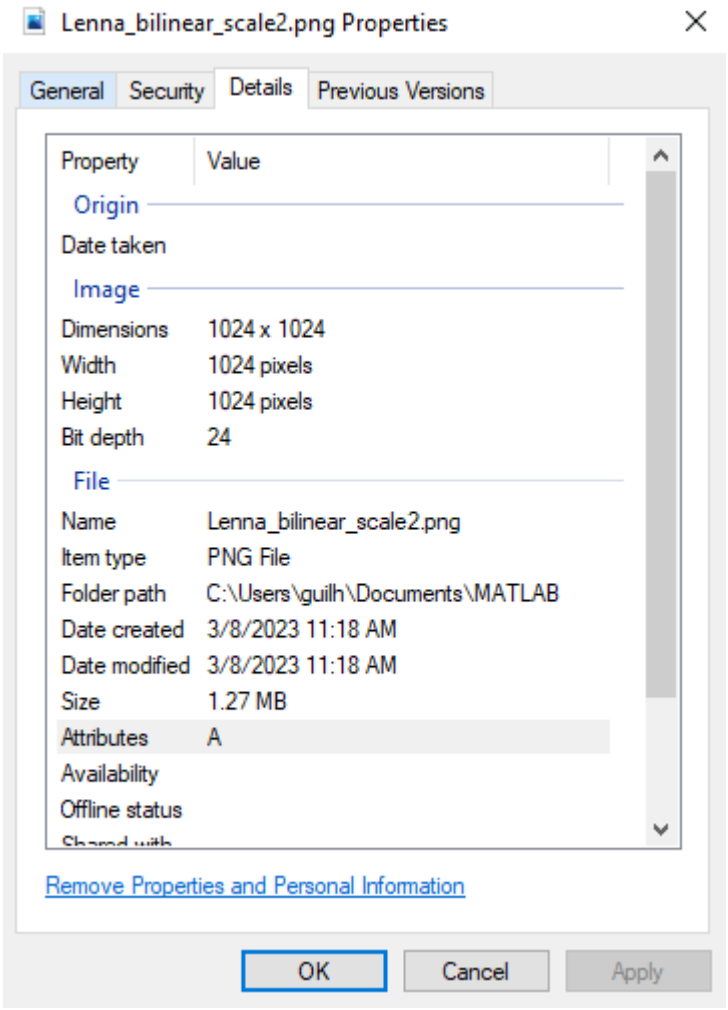
5.3.1) Lenna.png (Imagem Original)


Lenna.png Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions
512 x 512
Width
512 pixels
Height
512 pixels
Bit depth
24
File
Name
Lenna.png
Item type
PNG File
Folder path
C:\Users\guilh\Documents\MATLAB
Date created
3/6/2023 9:29 AM
Date modified
3/6/2023 9:29 AM
Size
462 KB
Attributes
A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply

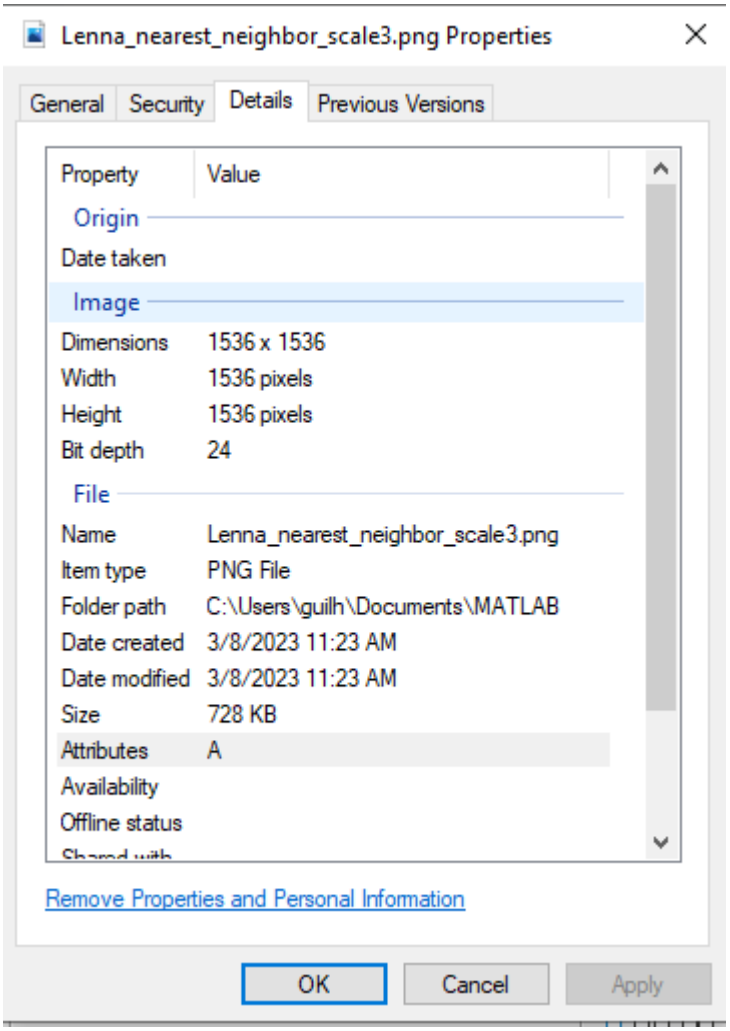
5.3.2) Lenna_nearest_neighbor_scale2.png


Lenna_nearest_neighbor_scale2.png Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions 1024 x 1024
Width 1024 pixels
Height 1024 pixels
Bit depth 24
File
Name Lenna_nearest_neighbor_scale2.png
Item type PNG File
Folder path C:\Users\guilh\Documents\MATLAB
Date created 3/8/2023 11:14 AM
Date modified 3/8/2023 11:14 AM
Size 669 KB
Attributes A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply

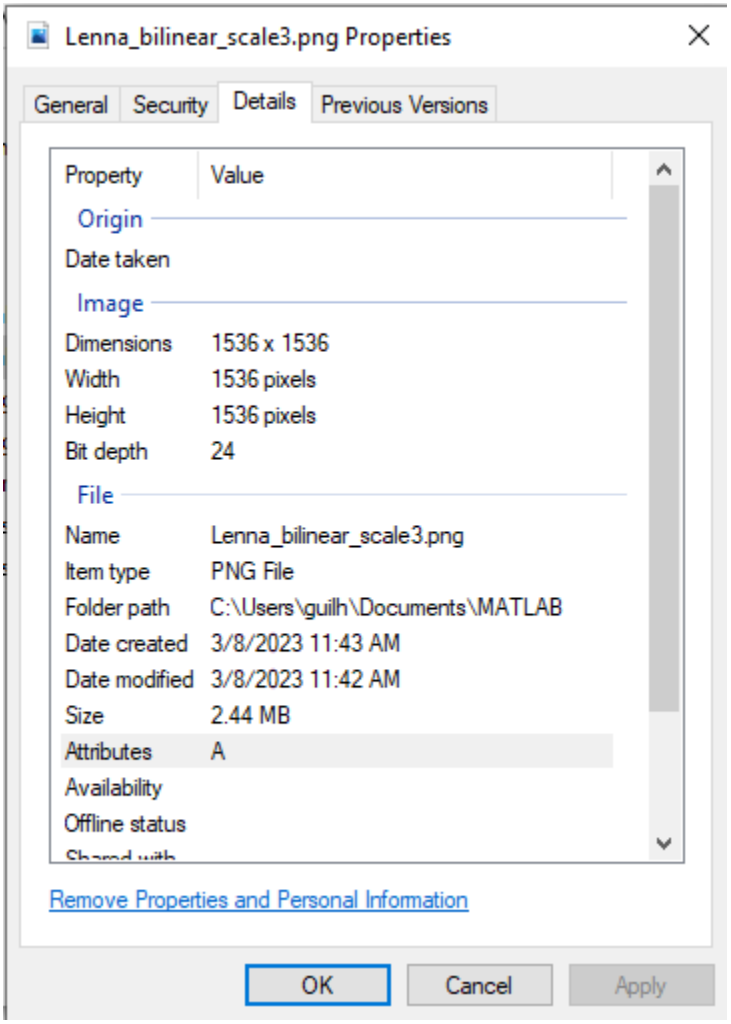
5.3.3) Lenna_bilinear_scale2.png


Lenna_bilinear_scale2.png Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions 1024 x 1024
Width 1024 pixels
Height 1024 pixels
Bit depth 24
File
Name Lenna_bilinear_scale2.png
Item type PNG File
Folder path C:\Users\guilh\Documents\MATLAB
Date created 3/8/2023 11:18 AM
Date modified 3/8/2023 11:18 AM
Size 1.27 MB
Attributes A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply

5.3.4) Lenna_nearest_neighbor_scale3.png


Lenna_nearest_neighbor_scale3.png Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions
1536 x 1536
Width
1536 pixels
Height
1536 pixels
Bit depth
24
File
Name
Lenna_nearest_neighbor_scale3.png
Item type
PNG File
Folder path
C:\Users\guilh\Documents\MATLAB
Date created
3/8/2023 11:23 AM
Date modified
3/8/2023 11:23 AM
Size
728 KB
Attributes
A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply

5.3.5) Lenna_bilinear_scale3.png


Lenna_bilinear_scale3.png Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions
1536 x 1536
Width
1536 pixels
Height
1536 pixels
Bit depth
24
File
Name
Lenna_bilinear_scale3.png
Item type
PNG File
Folder path
C:\Users\guilh\Documents\MATLAB
Date created
3/8/2023 11:43 AM
Date modified
3/8/2023 11:42 AM
Size
2.44 MB
Attributes
A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply

5.3.6) Lenna_bilinear_scale2_nearest_neighbor_rotation5

5.3.7) Lenna_bilinear_scale2_bilinear_rotation5

5.3.8) Lenna_bilinear_scale2_nearest_neighbor_rotation45

5.3.9) Lenna_bilinear_scale2_bilinear_rotation45

5.3.10) Rotações com as imagens escaladas por 3

As imagens escaladas por um fator de 3 são muito grandes e não cabem nas dimensões desse documento. Isto faz com que o software de edição de texto redimensione as imagens e os resultados das rotações aparentam ser iguais aos casos em escala de fator 2. Com isso em mente, recomendamos que utilize o script MATLAB “Q5” deixado na seção “5.2.7) Q5” para a visualização correta destes testes. Não se esqueça de descomentar os testes no final deste script para poder analisar os resultados obtidos.