**Image and Video Processing**

EEE412

Lab 1 - Report

Name: Cong Yin

ID Number: 1715471

**Introduction**

The color map is a real matrix is made by m\*3, the scope of real number is from 0 to 0.1, each of rows is defined to a RGB vector by each color.

**Task 1**

**1.1 Cool (256)**

Run the following code

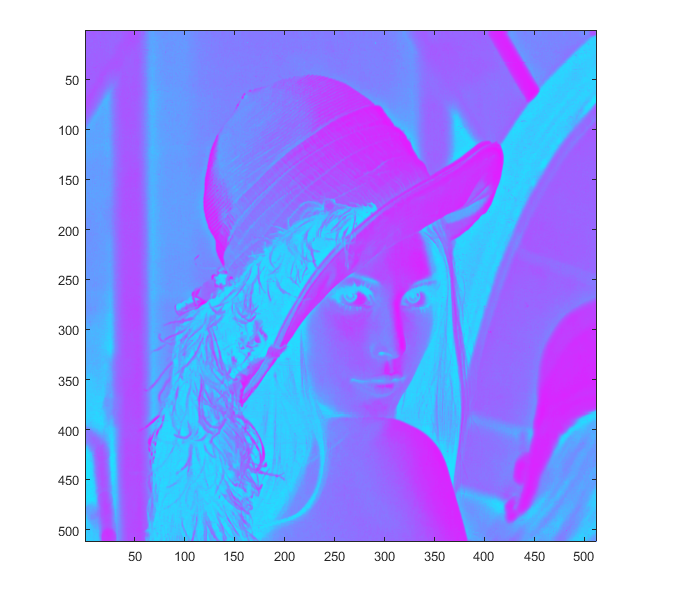
[A,map]=imread('lenna512.bmp');%Read the image from the file

image(A);%Display the image

colormap cool(256);% Set the colormap, which is from the question

truesize

It can get the following picture



**Figure 1-1 set the colormap to the cool (256)**

**1.2 Hot (255)**

Run the following code

[A,map]=imread('lenna512.bmp');%Read the image from the file

image(A);%Display image

colormap hot(255);%Set the colormap, whcih is from the question

truesize

It can get the following picture



**Figure 1-2 set the colormap to the Hot (255)**

**1.3 Gray (128)**

Run the following code

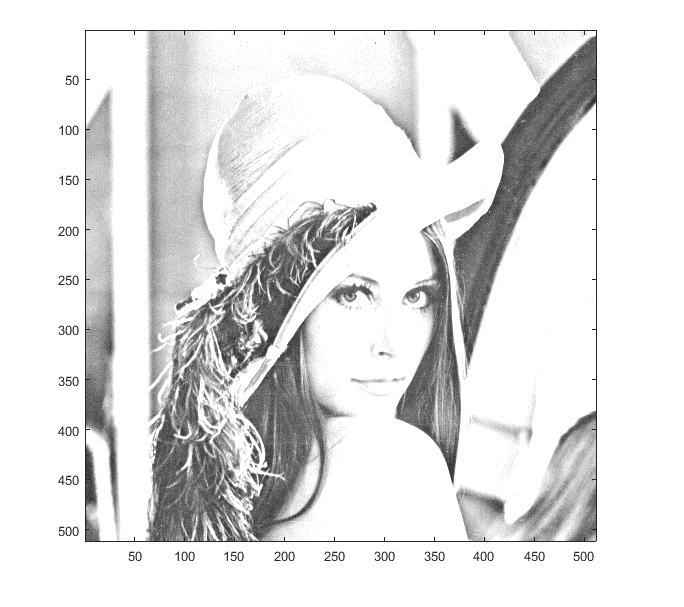
It can get the following picture

[A,map]=imread('lenna512.bmp');%Read the image from the file

image(A);%Display the image

colormap gray(128);%Set the colormap, whcih is from the question

truesize

****

**Figure 1-3 set the colormap to the Gray (128)**

**1.4 Gray (64)**

Run the following code

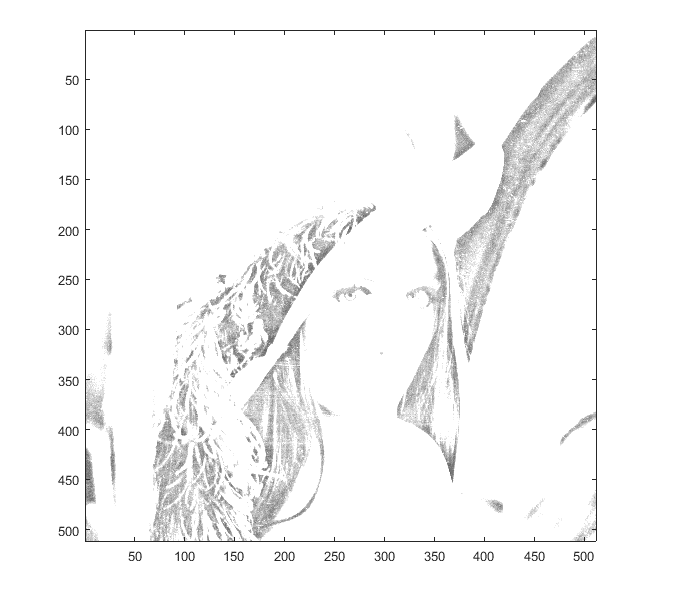
[A,map]=imread('lenna512.bmp');%Read the image from the file

image(A); %Display the image

colormap gray(64);%Set the colormap, whcih is from the question

truesize

It can get the following picture



**Figure 1-4 set the colormap to the Gray (64)**

In the MATLAB, figures have and only have one colormap. The ‘map’ means a matrix, which is made by m\*3, the scale of three values for each row is from 0 to 1 that means RGB value. For example, [1 0 0] means red, [0 1 0] means green and [0 0 1] means purple, the ‘colormap hot(64)’ can create a matrix, which is made by 64\*3. Stated another way, ‘hot(64)’ means create a hot colormap that have 64 kinds of colors.

**1.5 Negative**

Run the following code

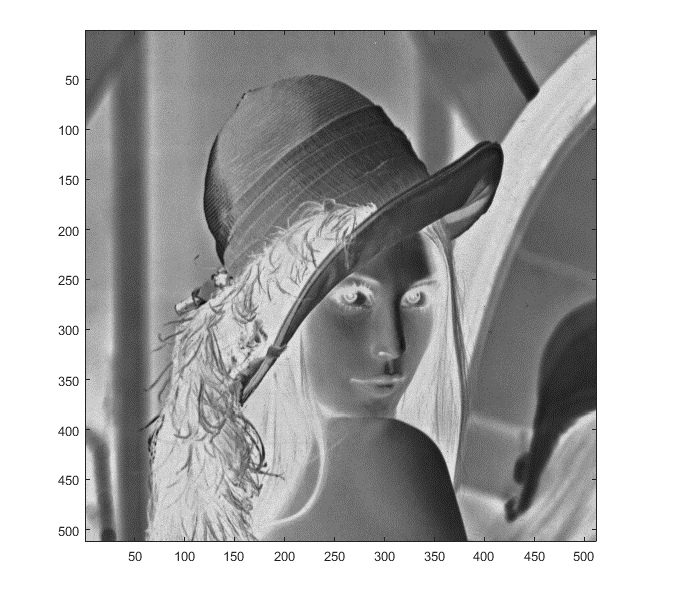
[A,map]=imread('lenna512.bmp');%Read the image from the file

image(A);%Display the image

colormap (1-map);

truesize

It can get the following picture



**Figure 1-4 the image as a negative**

In the matrix, the scale of three values for each row is from 0 to 1 that means RGB value, so use 1 to minus each numbers from the original image can display the image as a negative.

**Task 2**

**2.1 Data cursor**

Use the data cursor sign a point and run the following code

I = imread('lenna512.bmp');%Read the image from the file

I2 = imcrop(I,[195 168 200 270]);

subplot(1,2,1)

image(I)%Display the image

title('Original Image')

subplot(1,2,2)

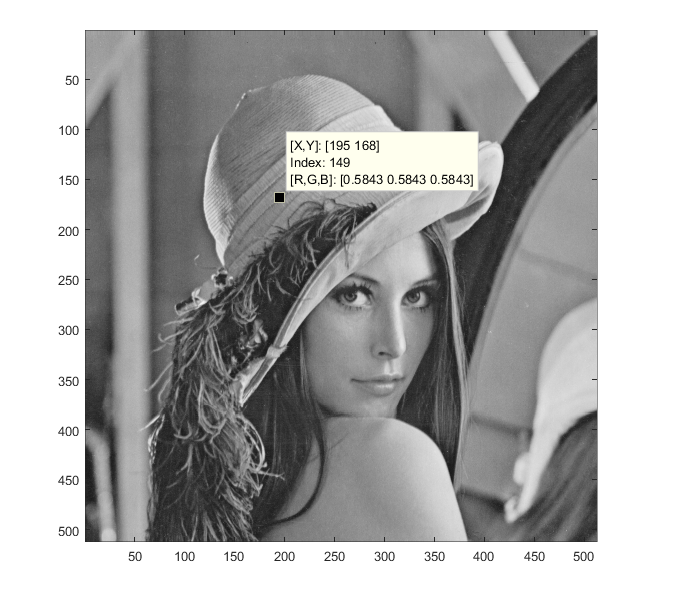
image(I2)%Display the image

title('Cropped Image')

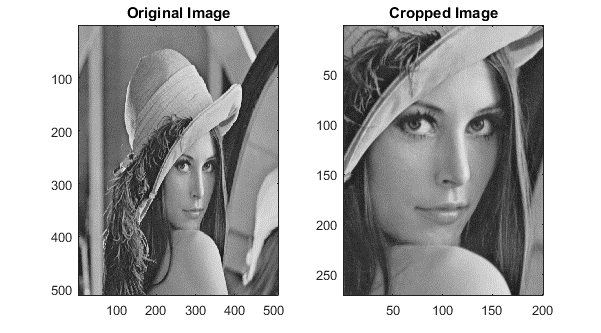
colormap(map)%Set the colormap, which is from question

truesize

It can get the following picture



**Figure 2-1 signs a point**



**Figure 2-2 result**

The‘[...] = imcrop(x,y,a,b)’means a cut out a rectangular area, the ‘x’ and ‘y’ means starting point coordinate, the ‘a’ and ‘b’ means a scale of length and width

**2.2 Down-sampling and up-sampling**

(a) Run the following code

I0=imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'nearest');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using nearest neighbor interpolation.

subplot(1,2,1)

image(I0)%Display the image

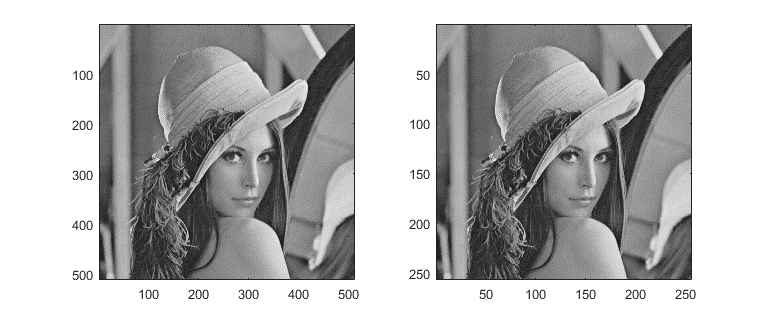
subplot(1,2,2)

image(I1)%Display the image

colormap(map)

truesize

It can get the following picture



**Figure 2-3 down-sampling to I1 with 1/2 size of I0 (using nearest neighbor)**

(b) Run the following code

[I0,map] = imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'nearest');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using nearest neighbor interpolation.

I1\_1=imresize(I1,2,'nearest');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using nearest neighbor interpolation.

subplot(1,2,1)

image(I0)%Display the image

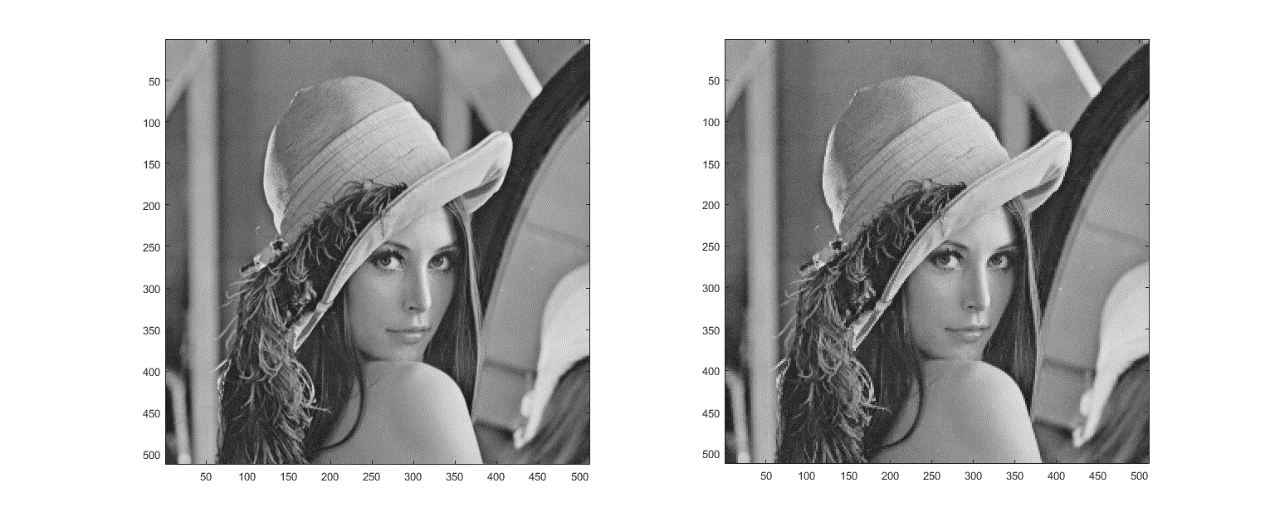
subplot(1,2,2)

image(I1\_1)%Display the image

colormap(map)

truesize

It can get the following picture

**Figure 2-4 up-sampling to I1\_1 with 2 size of I1 (using nearest neighbor)**

The nearest neighbor interpolation is a basic way to enlarged or reduce the image that is displayed by the voxel matrix. It use the function srcX=dstX\* (srcWidth/dstWidth)

srcY = dstY \* (srcHeight/dstHeight)

To calculate each voxel where needed to fill the new voxel matrix. However the pixel value must be an integer, so the last calculation needs rounding that can make the image full of the mosaic which was enlarged, at the same time, it also can make the image distorted which was reduced.

The new image, which was reduced and enlarged, is different from the original image

(c) Run the following code

I0 = imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'bilinear');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using bilinear

interpolation

subplot(1,2,1)

image(I0)%Display the image

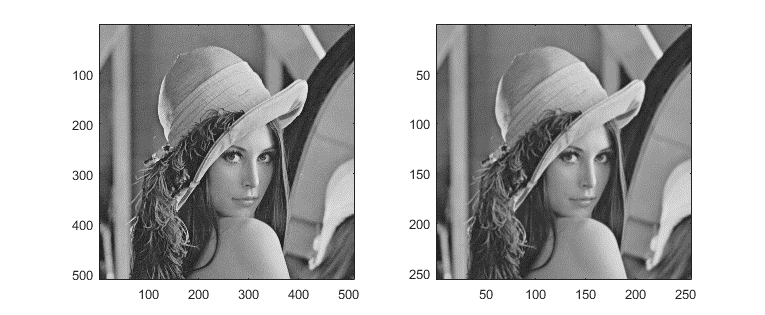
subplot(1,2,2)

image(I1)%Display the image

colormap(map)

truesize

It can get the following picture



**Figure 2-5 down-sampling to I1 with 1/2 size of I0 (using bilinear interpolation)**

Run the following code

[I0,map] = imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'bilinear');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using bilinear

interpolation

I1\_1=imresize(I1,2,'bilinear');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using bilinear

interpolation

subplot(1,2,1)

image(I0)%Display the image

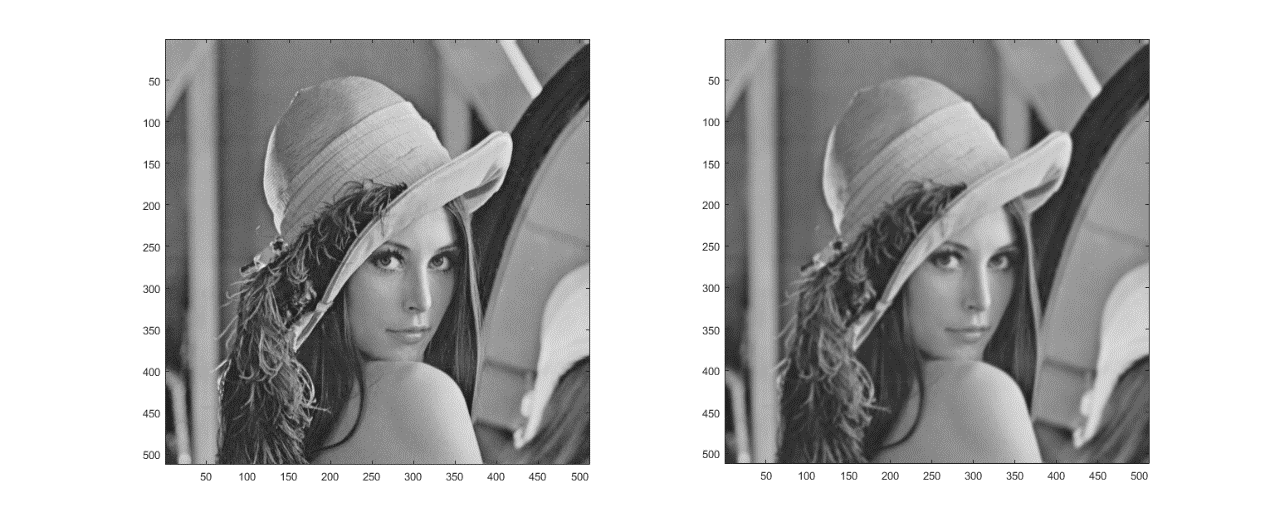
subplot(1,2,2)

image(I1\_1)%Display the image

colormap(map)

truesize

It can get the following picture



**Figure 2-6 Figure 2-5 up-sampling to I1\_1 with 2 size of I1 (using bilinear interpolation)**

For a destination pixels, set the coordinate of floating point to (i+u, j+v ), which is acquired by original coordinate through reverse. The ‘u’ and ‘v’ is a decimal part for coordinate of floating point, so the function

f(i+u,j+v) = (1-u)(1-v)f(i,j) + (1-u)vf(i,j+1) + u(1-v)f(i+1,j) + uvf(i+1,j+1)

It can display the new pixel value.

(d) Run the following code:

[I0,map]=imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'nearest');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using nearest neighbor interpolation.

I1\_1=imresize(I1,2,'nearest');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using nearest neighbor interpolation.

[peaksnr, snr] = psnr(I1\_1, I0);

fprintf('\n The Peak-SNR value is %0.4f', peaksnr);

The Peak-SNR value is 28.2912

And

[I0,map]=imread('lenna512.bmp'); %Read the image from the file

I1=imresize(I0,1/2,'bilinear');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using bilinear

interpolation

I1\_1=imresize(I1,2,'bilinear');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using bilinear

interpolation

[peaksnr, snr] = psnr(I1\_1, I0);

fprintf('\n The Peak-SNR value is %0.4f', peaksnr);

The Peak-SNR value is 31.3847

Peak Signal to Noise Ratio is objective criterion to evaluate the image, the function is PSNR=10\*log10((2^n-1)^2/MSE). The value of PSNR is smaller means distortion is fewer. So using bilinear interpolation is better.

Run the following code to compare the two pictures, which has same size.

[I0,map] = imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'nearest');

subplot(1,2,1)

image(I1)%Display the image

I2=imresize(I0,1/2,'bilinear');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using bilinear

interpolation

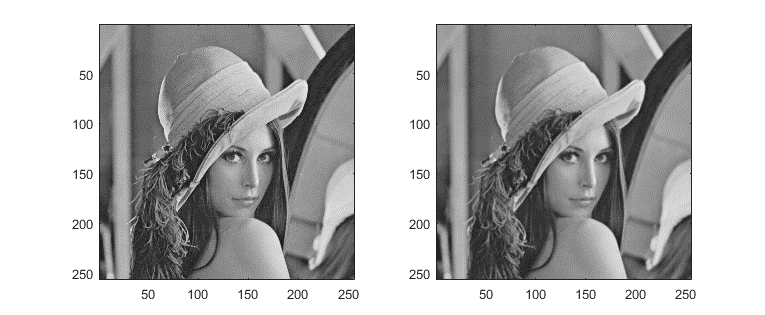
subplot(1,2,2)

image(I2)%Display the image

colormap(map)

truesize

It can get the following picture



**Figure 2-7 compare two ways to reduce the image**

[I0,map] = imread('lenna512.bmp');%Read the image from the file

I1=imresize(I0,1/2,'bilinear'); %down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using bilinear

interpolation

I1\_1=imresize(I1,2,'bilinear');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using bilinear

interpolation

I2=imresize(I0,1/2,'nearest');%down-sampling to I1 with 1/2 size of I0 (both horizontally and vertically) using nearest neighbor interpolation.

I1\_2=imresize(I2,2,'nearest');%up-sampling to I1\_1 with 2 size of I1 (both horizontally and vertically) using nearest neighbor interpolation.

subplot(1,2,1)

image(I1\_1)%Display the image

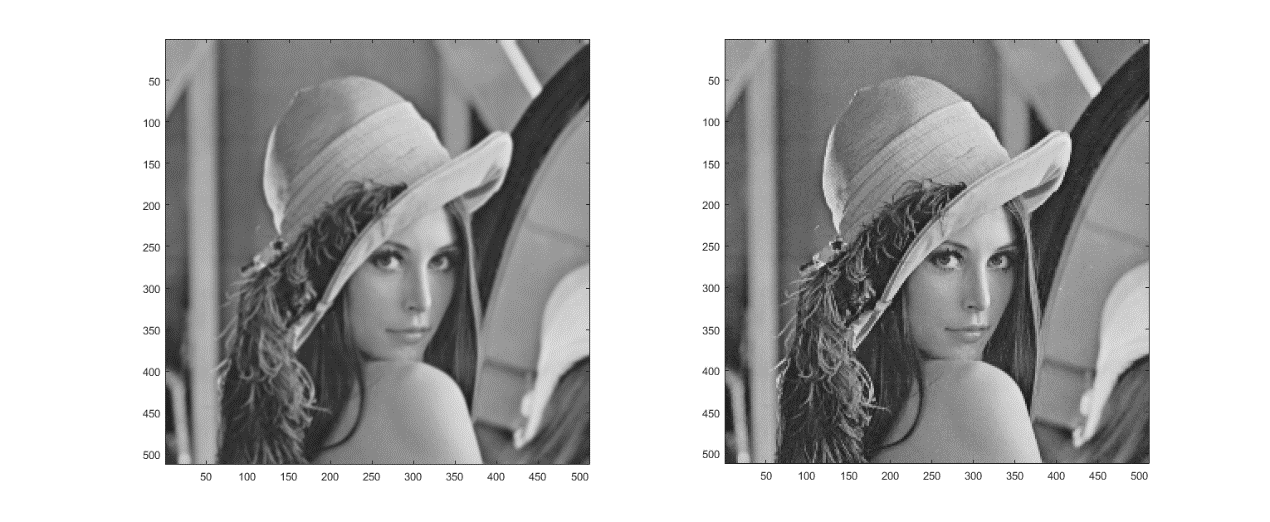
subplot(1,2,2)

image(I1\_2)%Display the image

colormap(map)

truesize

It can get the following picture



**Figure 2-8 compare two ways to enlarged the image**

For two same size images, we can also see the effect of bilinear interpolation is better than nearest neighbor interpolation.

**2.3 Reduce the picture to 16 values by quantization**

Gray = imread('lenna512.bmp'); %Read the image from the file

[M,N] = size(Gray); %acquire the line number and row number

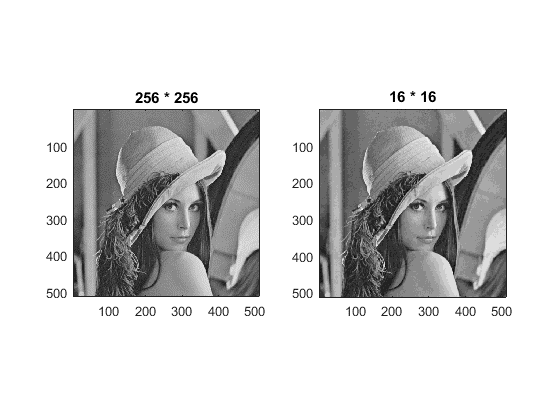
Gary1 = zeros(M, N); %M-by-N matrix of zeros

Gray1 = ( Gray - rem(Gray, 16) ) / 16; % Calculate the each corresponding pixel from 256 to 16

figure;

subplot(1, 2, 1); subimage(Gray,gray(256)); title('256 \* 256');

subplot(1, 2, 2); subimage(Gray1,gray(16)); title('16 \* 16');

****

**Figure 2-9 reduce the picture to 16 values by quantization**

The level of gray is larger that means the scale of color is more large, so it will have rich color.