Assignment 1 -- Interpolation

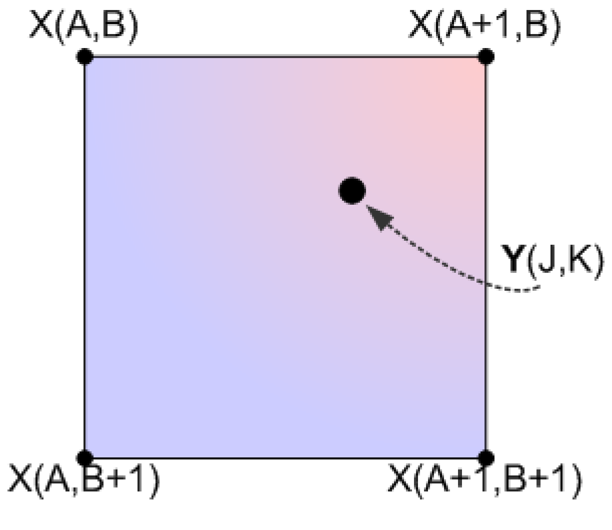
- ## Objective of the laboratory and the basic principle of the algorithms
  - **Objective**

    This assignment is to use nearest neighbor interpolation and bilinear interpolation to interpolate a gray scale image. Implement the interpolation as a function and output the enlarged as well as shrinked picture using interpolation respectively.
  - **the basic principle of the algorithms**
    - Nearest-neighbor interpolation

      it is a simple method of multivariate interpolation in one or more dimensions.

      the block used the value of **nearby** translated pixel values for output pixel values
    - bilinear interpolation

      The key idea is to perform linear interpolation first in one direction, and then again in the direction. For bilinear interpolation, the block uses the weighted average of two translated pixel values for each output pixel value.
- ## Pseudo code
  - **Nearest-neighbor interpolation**
    - To visualize nearest neighbor interpolation, consider the diagram below. The data points in the set X represent pixels from the original source image, while the data points in the set Y represent pixels in our target output image.



- So, for each pixel in the output image Y, we must calculate the nearest neighboring pixel in our source image X. Furthermore, we should only need to rely on 4 specific data points: X(A,B), X(A+1,B), X(A,B+1), and X(A+1,B+1). We can handle this decision very quickly with a few IF statements, as outlined in the pseudo-code below.

```
START
IF ( K-B < B+1-K)
     Pixel is one of the top two
ELSE
     Pixel is one of the bottom two
ENDIF
IF ( J-A < A+1-J)
     Pixel is one of the "left" two
ELSE
     Pixel is one of the "right" two
ENDIF
DONE
```

- For an image Bigger than 2*2 Pixels, we consider a small image which is m pixels wide by n pixels high, which we want to re-size to p pixels wide by q pixels high. Now we need two scaling constants:

$$s1 = \frac{p}{m}, s2 = \frac{q}{n}$$

- we simply loop through all the pixels in the target/output image, addressing the source pixels to copy from by scaling our control variables by s1 and s2, and rounding the resulting scaled index values.
    - **Bilinear interpolation**
        - Suppose that we want to find the value of the unknown function *f* at the point (*x*, *y*). It is assumed that we know the value of *f* at the four points *Q*11 = (*x*1, *y*1), *Q*12 = (*x*1, *y*2), *Q*21 = (*x*2, *y*1), and *Q*22 = (*x*2, *y*2). We first do linear interpolation in the *x*-direction. This yields

$$\begin{aligned} f(x, y_1) &\approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \\ f(x, y_2) &\approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}). \end{aligned}$$

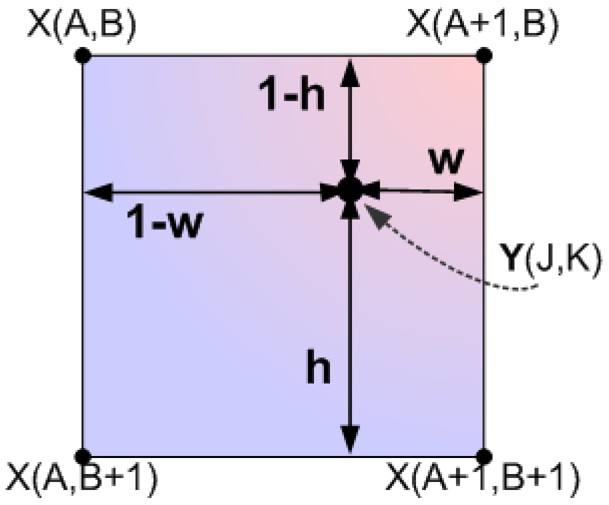
        - We proceed by interpolating in the *y* direction to obtain the desired estimate:

$$\begin{aligned} f(x,y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})\right) + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})\right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)}\left(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1)\right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)}\begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix}\begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix}\begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}. \end{aligned}$$

        - Note that we will arrive at the same result if the interpolation is done first along the *y* direction and then along the *x* direction.

If we choose a coordinate system in which the four points where *f* is known are (0, 0), (0, 1), (1, 0), and (1, 1), then the interpolation formula simplifies to

$$f(x,y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy,$$



- **Matlab codes**
    - **Nearest-neighbor interpolation**

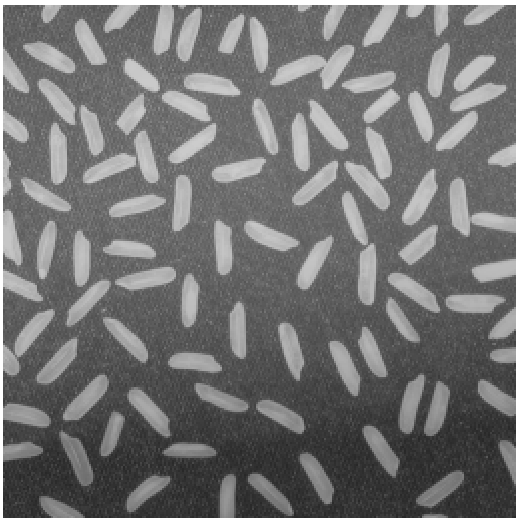
```
function []=Nearest_11712116(input_file,dim)
img=imread(input_file); % read image
[x0,y0]=size(img);% find the size of image and storage to x0,y0 respectively

figure;
imshow(img);% show original image

x=x0/dim(1);
y=y0/dim(2);% computing the ratio of origin size to target image
re_img=zeros(round(dim(1)),round(dim(2)));% new image

for i=1:round(dim(1))
    for j=1:round(dim(2))
        tx=round(i*x);
        ty=round(j*y);
        if(tx<1)
            tx=1;
        end
        if(tx>x0)
            tx=x0;
        end
        if(ty<1)
            ty=1;
        end
        if(ty>y0)
            ty=y0;
        end
        % avoid tx,ty out of bound
        re_img(i,j)=img(tx,ty);
        % interpolation
        end
end

figure;
```

```
re_img=uint8(re_img); %storage gray image to 8 bit matrix
imshow(re_img);

imwrite(re_img,'Shrinked_Nearest_11712116.tif');  %output
```

- **Bilinear interpolation**

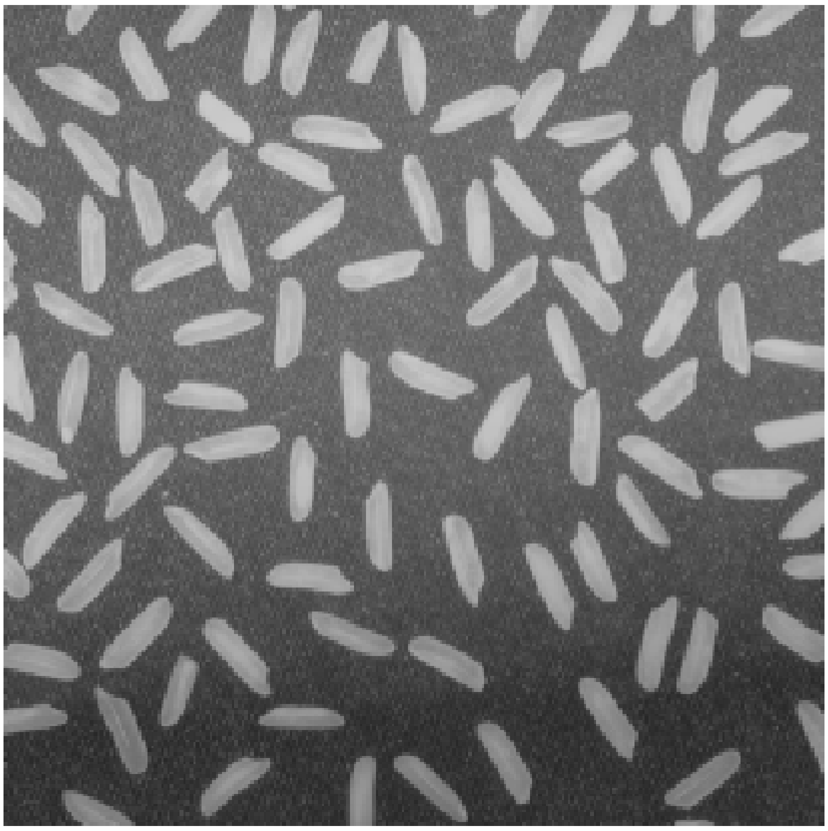
```
function []=Bilinear_11712116(input_file,dim)
img=imread(input_file);% read image
[x0,y0]=size(img);% find the size of image and storage to x0,y0 respectively

figure;
imshow(img);% show original image

x=x0/dim(1);
y=y0/dim(2);% computing the ratio of origin size to target image
re_img=zeros(round(dim(1)),round(dim(2)));% new image

for i=1:round(dim(1))
    x1=abs(i*x-floor(i*x));
    x2=floor(i*x);
    if x2>=x0
        x2=x0-1;
    end
    if x2<1
        x2=1;
    end
    % first do linear interpolation in the x-direction
    for j=1:round(dim(2))
        y1=abs(j*y-floor(j*y));
        y2=floor(j*y);
        if y2>=y0
            y2=y0-1;
        end
        if y2<1
            y2=1;
        end
       % interpolating in the y direction
    re_img(i,j)=(1-x1)*(1-y1)*img(x2,y2)+x1*(1-y1)*img(x2+1,y2)+(1-
x1)*y1*img(x2,y2+1)+x1*y1*img(x2+1,y2+1);
    % interpolation
    end
end

figure;
re_img=uint8(re_img);%storage gray image to 8 bit matrix
imshow(re_img);

imwrite(re_img,'Shrinked_Bilinear_11712116.tif');%output
```

- **Results**
  - Original Image rice.tif **(256 * 256)**



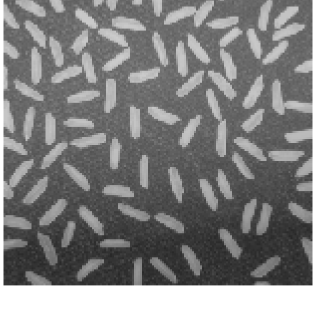
- Enlarged_Nearest_11712116.tif **(410 * 410)**

  > nearest_11712116("rice.tif",[256 * 1.6;256 * 1.6])



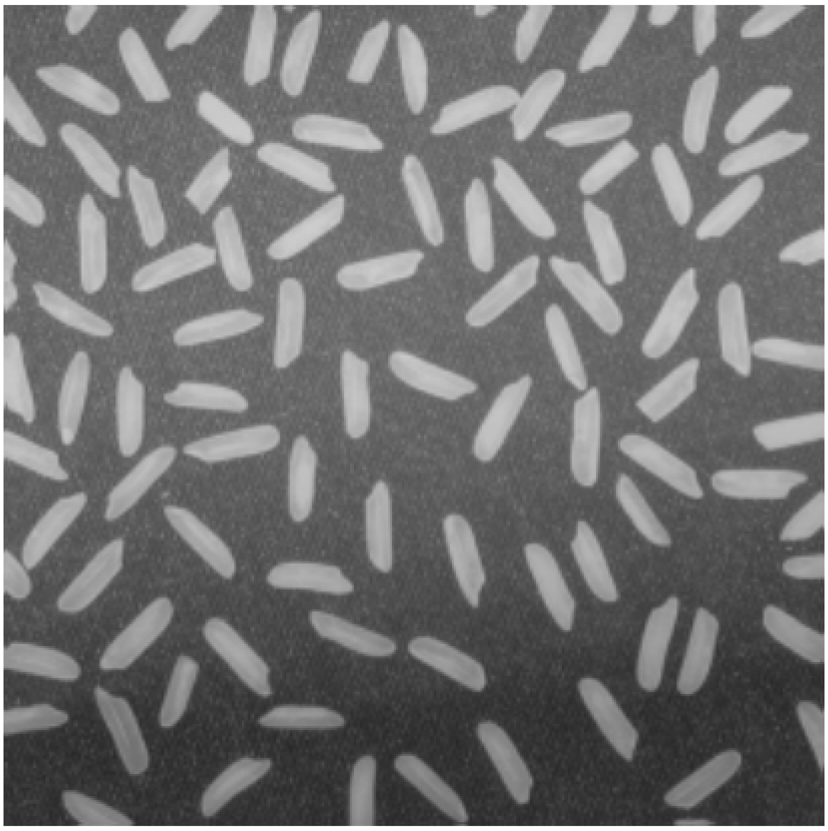
- Shrinked_Nearest_11712116.tif **(154 * 154)**

  > nearest_11712116("rice.tif",[256 * 0.6;256 * 0.6])



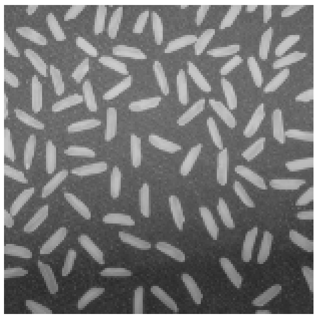
- Enlarged_Bilinear_11712116.tif **(410 * 410)**

  > Bilinear_11712116("rice.tif",[256 * 1.6;256 * 1.6])



- Shrinked_Bilinear_11712116.tif **(154 * 154)**

  > Bilinear_11712116("rice.tif",[256 * 0.6;256 * 0.6])



- **Analysis and conclusion**
  - Although I just picked this course, I am enthusiastic in digital image processing. It's really a magical approach for image.
  - I studied by myself for the knowledge that I haven't listened, it's challenging.
  - The assignment work is about nearest neighbor interpolation and bilinear interpolation to interpolate a gray scale image. They are both significant interpolation ways for image interpolation.
  - nearest neighbor interpolation is **easier** but also **not effective** than bilinear interpolation to interpolate a gray scale image