Laboratory 2 Image Interpolation

11810506 JIA Jiyuan
2021/01/18

Objective:

1. Learning the background and principle of nearest neighbor interpolation, bilinear interpolation and bicubic interpolation.
2. Applying the methods above to process the gray scale image “rice.tif”.
3. For bicubic interpolation, “interp2d” can help a lot.

Keywords: nearest neighbor interpolation, bilinear interpolation, bicubic interpolation

Introduction:

Meaning:

Image interpolation is a process of generating high-resolution images from low-resolution images in a model-based framework, which is used to recover lost information in the image. For example, to advertise, art photos need to be made into large posters, to detect cases without witnesses, need to do the clearness processing of surveillance videos, which need to use image interpolation technology, to enlarge the original low-resolution image or blurred image.

Image interpolation is to use the known gray values of adjacent pixels (or trichromatic values in RGB images) to generate the gray values of unknown pixels, so that the original image can be regenerated with a higher resolution image. Interpolation is a method of redistributing the pixels of the original image to change the number of pixels. In the process of image magnification, pixel also increases accordingly, increasing the process is the process of "interpolation" works, "interpolation" program automatically choose information better pixel as increase, make up the pixels of blank space, not only use the neighboring pixels, so when magnified image, the image will look smoother.

It should be noted, however, that interpolation does not add information to the image.

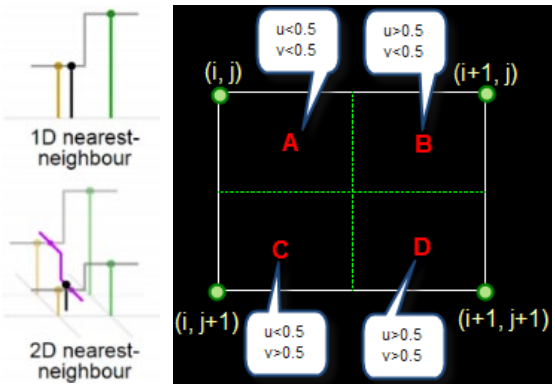
Methodology:

Image interpolation methods involve nearest neighbor interpolation, bilinear interpolation, double square interpolation, double cubic interpolation, and other higher order methods. In this experiment, nearest neighbor interpolation, bilinear interpolation and bicubic interpolation will be applied.

Nearest neighbor interpolation:

Principle:

The positions of new grayscale image’s pixels are obtained after scaling is mapped to the original image, and the value of each pixel position of the new grayscale image is assigned to it the value of the pixel of the original image closest to the pixel of the new grayscale image.

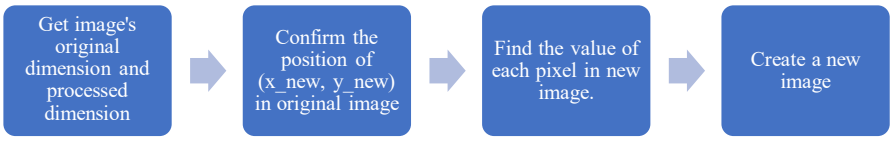


Question Formulation:

$$[x_{new}, y_{new}] = \frac{Processed\ dimension}{Original\ dimension} * [x_{origin}, y_{origin}]$$

If (i + u , j + v) falls in area A, that is, u < 0.5, v < 0.5, then the gray value of the upper left pixel is assigned to the pixel to be solved. Similarly, if it falls in area B, the gray value of the upper right pixel is assigned; if it falls in area C, the gray value of the lower left pixel is assigned; if it falls in area D, the gray value of the lower right pixel is assigned.

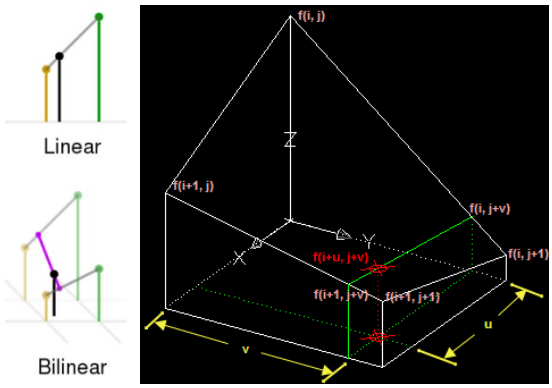
Ps: $[x_{new}, y_{new}]$ is the coordinates of pixels in new greyscale image, $[x_{origin}, y_{origin}]$ is the coordinates of pixels in original greyscale image



Bilinear neighbor interpolation:

Principle:

The bilinear interpolation method uses the values of four adjacent points and gives different weights according to their distance from the interpolation points to carry out linear interpolation.



Question Formulation:

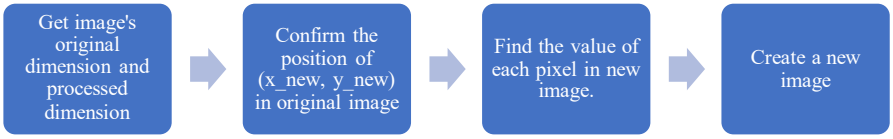
For (i , j + v), the change of gray level is linear between f(i , j) and f(i , j + 1).

Then: f(i , j + v) = [f(i , j + 1) - f(i , j)] * v + f(i , j)

Similarly, f(i + 1, j + v) = [f(i + 1 , j + 1) - f(i + 1 , j)] * v + f(i + 1 , j)

The change of gray level from f(i , j + v) to f(i + 1 , j + v) is also linear. Thus, the calculation formula of the gray level of the pixel to be solved can be deduced as following:

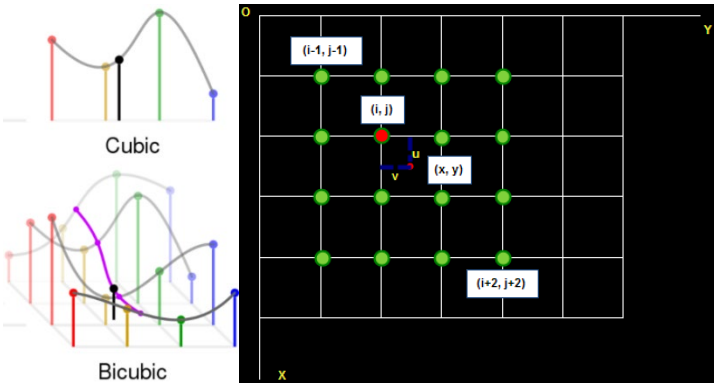
f(i + u, j + v) = (1 - u) * (1 - v) * f(i , j) + (1 - u) * v * f(i , j + 1) + u * (1 - v) * f(i + 1 , j) + u * v * f(i + 1 , j + 1)



Bicubic interpolation

Principle:

Bicubic interpolation is also called cubic convolution interpolation. Cubic convolution interpolation is a more complex interpolation method. In this algorithm, the gray values of 16 points around the sampling point are used for cubic interpolation, which takes into account not only the gray values of four direct adjacent points, but also the change rate of gray values among adjacent points. Three operations can produce magnification that is closer to a high-resolution image, but it also leads to a dramatic increase in the amount of computation.



Question Formulation:

"interp2d" will be used in this experiment.

The gray value of the pixel (x, y) is obtained by weighted interpolation of 16 gray values around it.

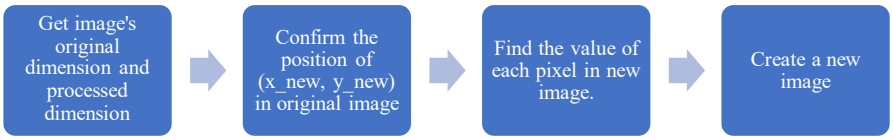
$$s(x) = \begin{cases} |x|^3 - 2|x|^2 + 1 & for\ |x| \le 1 \\ -|x|^3 + 5|x|^2 - 8|x| + 4 & for\ 1 < |x| < 2 \\ 0 & otherwise \end{cases}$$

$$A = \begin{pmatrix} S(1+v) \\ S(v) \\ S(1-v) \\ S(2-v) \end{pmatrix}^T$$

$$B = \begin{pmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) & f(i-1,j+2) \\ f(i,j-1) & f(i,j) & f(i,j+1) & f(i,j+2) \\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) & f(i+1,j+2) \\ f(i+2,j-1) & f(i+2,j) & f(i+2,j+1) & f(i+2,j+2) \end{pmatrix}$$

$$C = \begin{pmatrix} S(1+u) \\ S(u) \\ S(1-u) \\ S(2-u) \end{pmatrix}$$

f(x , y) = f(i + u , j + v) = ABC



Experiment:

Nearest neighbor interpolation:

The parameter of INTERP1D function is set as Nearest can realize the nearest interpolation algorithm.

```
def Nearest_11810506(input_file, dim):
Load image
    in_image = io.imread(input_file)
Dimension of output image
    out_width = dim[0]
    out_height = dim[1]
Dimension of input image
    in_width = in_image.shape[0]
    in_height = in_image.shape[1]
Initial the output image
    out_image = np.zeros(dim, dtype=np.uint8)

Perform Exchange

Applying the Question Formulation:

    for i in range(out_height):
        for j in range(out_width):
Mapping
            m = round(i / (out_height-1) * (in_height-1))
            n = round(j / (out_width-1) * (in_width-1))
No out of bounds
Interpolation
            out_image[i, j] = in_image[m, n]

Save image
    if (out_width > in_width):
        io.imsave("Enlarged_Nearest_11810506.tif", out_image)
    else:
        io.imsave("Shrunken_Nearest_11810506.tif", out_image)
```

Bilinear neighbor interpolation:

Perform Exchange

Applying the Question Formulation:

```
for i in range(out_height):
    for j in range(out_width):

        x = i / (out_height-1) * (in_height-1)
        y = j / (out_width-1) * (in_width-1)

        x1 = int(x)
        y1 = int(y)

        x2 = x1
        y2 = y1 + 1

        x3 = x1 + 1
        y3 = y1

        x4 = x1 + 1
        y4 = y1 + 1

        u = x - x1
        v = y - y1
```

No out of bounds

```
        if x4 >= in_height:
            x4 = in_height - 2
            x1 = x1
            x2 = x2
            x3 = x4
        if y4 >= in_width:
            y4 = in_width - 2
            y1 = y1
            y2 = y4
            y3 = y3
```

Interpolation

```
        out_image[i, j] = (1 - u) * (1 - v) * (in_image[x1, y1]) + (1 -
u) * v * (in_image[x2, y2]) + u * (1 - v) * (in_image[x3, y3]) + u * v *
(in_image[x4, y4])
```

Bicubic interpolation:

Perform Exchange

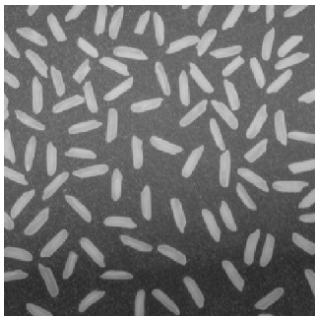
Applying the Question Formulation:

```
out_x = np.linspace(0, in_height - 1, num=out_height)
out_y = np.linspace(0, in_width - 1, num=out_width)
f = interp2d(range(in_height), range(in_width), in_image, kind='cubic')
out_image = f(out_x, out_y)
```

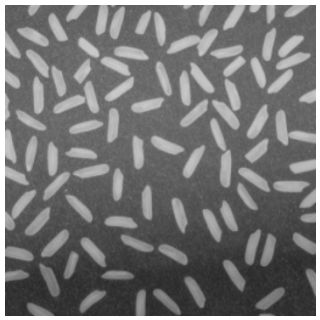
Results and conclusion:

Results:

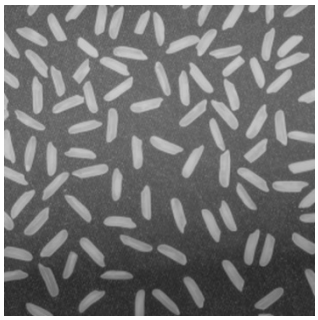
Enlarged_Nearest_11810506:



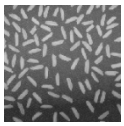
Enlarged_Bilinear_11810506:



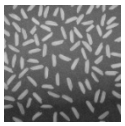
Enlarged_Bicubic_11810506:



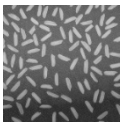
Shrunken_Nearest_11810506:



Shrunken_Bilinear_11810506:



Shrunken_Bicubic_11810506:



Nearest neighbor interpolation:

The nearest neighbor element method requires less computation, but it may cause discontinuity in the gray level of the image generated by interpolation, and obvious zigzag may appear where the gray level changes. This distortion is caused by the fact that the value of each pixel in the output image is only assigned to the nearest neighbor. In other words, when extrapolating from the coordinate of the output image, if the coordinate in the original image is a floating-point number, it is unscientific to simply use the rounding method to directly take the pixel value closest to the coordinate of the floating-point number.

Bilinear neighbor interpolation:

The calculation of bilinear interpolation method is more complicated than the nearest neighbor method, but it does not have the disadvantage of gray discontinuity, and the result is basically satisfactory. It has the characteristics of low pass filtering, which makes the high frequency component of the image damaged, and the image contour may be a little blurred. Compared with the most recent interpolation method, the bilinear interpolation method shows no discontinuity of pixel values. The value of each pixel in the output image is calculated by first mapping the pixel to the original image and then substituting the values of the four original pixels around the coordinates into the formula. Despite the computational complexity, the image quality after scaling is higher than that of the nearest neighbor interpolation. When the image is scaled using bilinear interpolation, it looks roughly the same as the original image.

Bicubic interpolation:

The image obtained by bicubic interpolation scaling is very sharp because the image is obtained by bicubic interpolation, and the value of each pixel of the output image is calculated using the 16 coordinates of the original image, as compared to the bilinear interpolation method. Bicubic interpolation is a more complex interpolation method than bilinear interpolation, which produces smoother edges. Bicubic interpolation algorithm is usually used for image or video scaling, and it can maintain better detail quality than the mainstream bilinear filtering algorithm.

Conclusion:

In the nearest neighbor interpolation experiment, if the width of the output image is the same as the high scaling factor, in addition to the method I mentioned above, it can also be calculated directly through matrix operation. Although the above method is slower in the calculation speed in this experiment, it is more adaptable. This method can also be used when the width of the output image is different from the high scaling factor.

After this experiment, I have a deeper understanding of the principle and specific operation process of the nearest neighbor interpolation and bilinear interpolation. I can analyze the reasons for the actual effect of the pictures obtained by these two methods. At the same time, the principle and operation flow of bicubic interpolation are also understood.

Nearest neighbor interpolation maps each pixel of the output image to the original image, and then get the value of closest coordinate point of the original image to it. With the closest interpolation, the enlarged image will appear jagged at the edges of the pattern, and the compressed image will appear significant distortion. This is because when the pixel of the output image is mapped to the original image, if the coordinate is a floating-point number, the value of the nearest coordinate point is directly assigned to it, thus losing precision.

Bilinear interpolation maps each pixel of the output image to the original image and uses the values of the four nearest points on that point to get the result. Compared with the nearest interpolation method, the scale results of bilinear interpolation have no obvious distortion. Therefore, its effect is superior to that of recent interpolation methods.

For bicubic interpolation, compared to bilinear interpolation, it maps each pixel of the output image to the original image and uses the values of the closest 16 coordinate points, so the result is more closely related to the original image and is better.

In the bilinear interpolation experiments in our laboratory, special attention should be paid to pixel position matching, otherwise obvious distortion will occur.