Image super-resolution comparison among interpolation methods

Kaixing WU*

May 22, 2021

Abstract

In the image processing area, enhancing the quality of low-resolution images is a big issue. In this paper I will compare the processing time, results and difficulty of implementing different interpolation methods.

1 Introduction

Enhancing quality of low-resolution images is of importance for people to improve their experience when viewing images. This can be done by super-resolution methods. Currently there are 3 categories of approaches for this problem and they are interpolation approaches, reconstruction approaches and machine learning approaches [1]. In this paper, we mainly focus on the processing time, implement complexity and results of interpolation approaches. For representation purpose, I use the famous image Lenna for demonstration.

2 Governing equations

I have implemented three approaches, namely the nearest neighbor method, the bilinear interpolation method and the bicubic interpolation method. For the nearest neighbor method, the mathematical principle is the following equation.

$$g(x,y) = f(\left\lfloor \frac{x}{n} \right\rfloor, \left\lfloor \frac{y}{n} \right\rfloor), \tag{1}$$

where the index $x, y, n \in \mathbb{N}$ and n is the resize factor provided by user.In the equation, $g(x,y)$ is the high-resolution image generated and $f(x,y)$ is the low-resolution image provided.

As for the bilinear method, in order to enhance the image by a factor of 2 from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{2n\times 2n}$, the mathematical principle can be expressed as follows.

$$g(x,y) = \begin{cases} f(\frac{x}{2}, \frac{y}{2}) & \text{x,y are both even} \\ \frac{1}{2} \times [f(\left\lfloor \frac{x}{2} \right\rfloor, \frac{y}{2}) + f(\left\lfloor \frac{x}{2} \right\rfloor + 1, \frac{y}{2})] & \text{x is odd and y is even} \\ \frac{1}{2} \times [f(\frac{x}{2}, \left\lfloor \frac{y}{2} \right\rfloor) + f(\frac{x}{2}, \left\lfloor \frac{y}{2} \right\rfloor + 1)] & \text{x is even and y is odd} \\ \frac{1}{4} \times [f(\left\lfloor \frac{x}{2} \right\rfloor, \left\lfloor \frac{y}{2} \right\rfloor) + f(\left\lfloor \frac{x}{2} \right\rfloor, \left\lfloor \frac{y}{2} \right\rfloor + 1) \\ + f(\left\lfloor \frac{x}{2} \right\rfloor + 1, \left\lfloor \frac{y}{2} \right\rfloor) + f(\left\lfloor \frac{x}{2} \right\rfloor + 1, \left\lfloor \frac{y}{2} \right\rfloor + 1)] & \text{x is odd and y is odd} \end{cases} \tag{2}$$

*Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **kwual@connect.ust.hk**

When it comes to the bicubic interpolation method, the equation is much more complex. The original cubic function in 1 dimension can be expressed as follows[2].

$$g(x) = \begin{cases} \frac{3}{2}|x|^3 - \frac{5}{2}|x|^2 + 1, & |x| < 1 \\ -\frac{1}{2}|x|^3 + \frac{5}{2}|x|^2 - 4|x| + 2, & 1 < |x| < 2 \\ 0, & |x| > 2 \end{cases} \tag{3}$$

The matrix form of the bicubic equation in 2 dimensions can be expressed as follows[**?**].

$$p(t) = \frac{1}{2}\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} f_{-1} \\ f_0 \\ f_1 \\ f_2 \end{bmatrix} \tag{4}$$

In the equation, f_{-1}, f_0, f_1, f_2 are the nearest four points, two on the left and two on the right. t represents the distance from the indexed point 0 to the inquiry point. For the bicubic case in 2 dimensions, we need four more equations as follows[**?**].

$$b_{-1} = p(t_x, f_{-1,-1}, f_{-1,0}, f_{-1,1}, f_{-1,2}) \tag{5}$$

$$b_0 = p(t_x, f_{0,-1}, f_{0,0}, f_{0,1}, f_{0,2}) \tag{6}$$

$$b_1 = p(t_x, f_{1,-1}, f_{1,0}, f_{1,1}, f_{1,2}) \tag{7}$$

$$b_2 = p(t_x, f_{2,-1}, f_{2,0}, f_{2,1}, f_{2,2}) \tag{8}$$

$$p(x, y) = p(t_y, b_{-1}, b_0, b_1, b_2) \tag{9}$$

Except for the equations of the interpolation methods, I also use the concept of the average variation per pixel to measure the difference between the generated picture $g(x, y)$ with the original one $f(x, y)$of size $\mathbb{R}^{N\times N}$. The equation is given below.

$$AV = \frac{1}{N^2}\sum_{x=1}^{N}\sum_{y=1}^{N}|g(x, y) - f(x, y)| \tag{10}$$

3 Results

In the project, I am trying to apply the interpolation methods on the image of 256-by-256 pixels in Figure 1. The results and the ground truth image are all 512-by-512 pixels.

Figure 1: This is the input image and its size is 256-by-256 pixels. We are trying to make it 512-by-512 pixels.

As shown in Figure 2, it is obvious that although the mathematical principle and the implementation of the nearest neighbor approach is easy, the result is far from perfect. The strong

(a)  (b) 

Figure 2: Figure (a) is the result of using nearest neighbor approach. Figure (b) is the ground truth.

aliasing effect makes the image too sharp. To produce Figure 1, Eq. (1) is implemented in $MATLAB$ and applied on Figure 1. The AV of this method is 48.5316.

As for the bilinear approach, the result is shown in Figure 3. The image generated by bilinear interpolation is of better quality than using nearest neighbor approach. However, the blurring effect becomes an issue. The edges in the image are not as clear as the ground truth. Also, the colors are less vivid than the ground truth. The AV of this method is 48.3310.

(a)  (b) 

Figure 3: Figure (a) is the result of using bilinear approach. Figure (b) is the ground truth.

As for the bicubic approach, the result is shown in Figure 4. Th image generated by bicubic interpolation is almost of the same quality as the bilinear interpolation except that it preserves the edges better. The AV of this method is 48.7256.

(a)  (b) 

Figure 4: Figure (a) is the result of using bicubic approach. Figure (b) is the ground truth.

4 Conclusions

Among the three interpolation approaches, the bicubic approach is the most difficult to conduct the mathematical equations and to implement a well functioning program while the nearest neighbor approach is the easiest one. The level of difficulty of implementing the bilinear approach is between these two methods.

However, the processing time is the reverse way. Bicubic approach takes longest time, 1.2734 seconds, to complete. Nearest neighbor approach is the fastest among the three, taking only 0.3632 seconds. The bilinear approach lies in between, taking 0.5294 seconds. The processing time difference could be explained by the calculation involved each step. Nearest neighbor approach only considers 1 pixel each time while bilinear approach has to consider 4 neighbor pixels each time. As for the bicubic approach, it has to take the nearest 16 pixels into calculation at each step!

Considering the processing quality, although the average variation per pixel defined above cannot show clearly the difference among the pictures generated by the three methods, the nearest neighbor approach produced the lowest quality definitely. The picture generated by nearest neighbor approach is not smooth enough and has strong aliasing effect. Both the bilinear approach and the bicubic approach are far better than it. Between the later two methods, the bicubic approach is slightly better than bilinear one by better preserving the edge features while the bilinear approach smooths the edges too much.

Considering all the issues above, if the processing time is not an issue, the bicubic interpolation approach is always recommended since it produces pictures of the highest quality. However, if we have to consider time issue, the bilinear approach is recommended. This is because the processing time of the bilinear interpolation approach is not of orders of magnitudes higher than the most time-saving method, the nearest neighbor approach, and at the same time the bilinear approach produces pictures with far better quality than the nearest neighbor approach.

Acknowledgments

I would like to thank Prof. Leung for helping me figure out what and how to choose a project for this course.

References

[1] J. Sun, & Z. Xu, & H.Y. Shum, *Image super-resolution using gradient profile prior, 2008*

[2] R. Keys, *Cubic convolution interpolation for digital image processing, 1981*

[3] *Bicubic Interpolation. URL: https://en.wikipedia.org/wiki/Bicubic_interpolation.*