A Comprehensive Study of 1D and 2D Image Interpolation Techniques

V. Diana Earshia and M. Sumathi

Abstract Image interpolation plays an important role in converting a low resolution image into a high resolution image. This paper provides a comprehensive study of perdurable image interpolation techniques, such as nearest neighbor, bilinear, bicubic, cubic spline, and iterative linear interpolation. The usage of a Lagrange polynomial and a piecewise polynomial gives a better fitting curve for interpolated pixel values. The parameters of interest are the signal-to-noise ratio, peak signal-to-noise ratio, mean square error and processing time. Experiment results are used to analyze the performance of interpolation algorithms. These results help us to choose an appropriate algorithm for better usage.

Keywords Image interpolation · Image scaling · Image magnification
Image interpolation algorithms

1 Introduction

Image interpolation [1, 2] is widely used in many areas of modern electronics, such as digital cameras and computer graphics etc. Image interpolation is a method of estimating new data points with the range of discrete set of known data points. It is a process of finding a value between two points on a line or curve.

The objective of interpolation is to retain the qualitative characteristics of a reproduced image from artifacts such as blurring, checkerboard effects, edge discontinuities, and jagging etc. Interpolation techniques were developed to convert low resolution images to high resolution images, with low computational complexity and high accuracy. The various interpolation techniques that were

V. Diana Earshia (✉) · M. Sumathi
Department of Electronics and Communication Engineering,
Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India
e-mail: earshy@gmail.com

M. Sumathi
e-mail: sumagopi206@gmail.com

A. Kumar and S. Mozar (eds.), *ICCCE 2018*,
Lecture Notes in Electrical Engineering 500,
https://doi.org/10.1007/978-981-13-0212-1_40

developed are nearest neighbor interpolation [3, 4], bilinear interpolation [5, 6], bicubic interpolation [7, 8], cubic spine interpolation [9, 10] and iterative linear interpolation [11, 12].

Interpolation is the problem of approximating the value of a function for a non-given point in a space when given the value of that function at points around that point. In computer graphics and digital imaging, image scaling refers to the resizing of a digital image. In video technology, the magnification of digital material is known as upscaling or resolution enhancement [1, 2]. Photo interpolation is the process by which the number of pixels comprising an image is increased to allow printing enlargements that are of higher quality than photos that are not interpolated. Interpolation is commonly used to make quality large prints from digital photos and film-scanned images.

2 Interpolation Techniques

The various interpolation techniques considered for comprehensive study are as follows.

2.1 Nearest Neighbor Interpolation (NNA)

In NNA [3, 4] neighboring pixel points are chosen to calculate the value of the interpolated pixel. Now consider the interpolated pixel as $A = f(x, y)$. Let the neighboring pixels be at (i, j), $(i, j + 1)$, $(i + 1, j)$, $(i + 1, j + 1)$. Its pixel value is manipulated based on the algorithm described below.

2.1.1 Methodology

If $(y - j) < ((j + 1) - y)$ then the value of the interpolated pixel will be one of the top two pixel values, else any one of the bottom two pixel values will be chosen.

If $(x - i) < ((i + 1) - x)$ then the value of the interpolated pixel will be one of the left two pixel values, else any one of the bottom two pixels will be chosen.

Figure 1b depicts how the interpolated pixel value $A(x \cdot y)$ is obtained from the actual image shown in Fig. 1a. The actual image is divided into, say, four pixels. The scaled image has sixteen pixels. The interpolated image should have similar feature characteristics to the actual image.

This method is simple and easy to implement and has flexible features choices. The major drawbacks are its speed and poor interpolated image generation with aliasing and blurring effects.

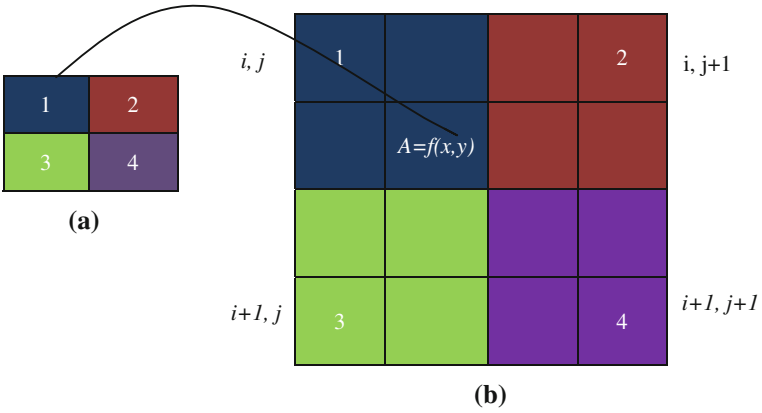

Fig. 1 **a** Actual image **b** image block describing the interpolated pixel $A(x, y)$ using NNA [4]

2.2 *Bilinear Interpolation (BLI)*

In BLI [5, 6] four adjacent pixel points on a rectilinear 2D grid are used for calculating the weighted value of the interpolated point in a scaled image. Linear calculation functions are used in two directions, namely horizontal and vertical, to find the interpolated pixel value.

2.2.1 Methodology

To find the value of P:

$$f(i, j+y) = [f(i,j+1) - f(i,j)]y + f(i,j) \tag{1}$$

To find the value of Q:

$$f(i+1, j+1) = [f(i+1,j+1) - f(i+1,j)]y + f(i+1,j) \tag{2}$$

To find the value of A:

$$\begin{aligned} f(i+x, j+y) &= (1-x)(1-y)f(i \cdot j) - (1-x)yf(I, j+1) + x(1-y) \\ & f(i+1,j) + xyf(i+1,j+1) \end{aligned} \tag{3}$$

Figure 2 shows the pictorial representation of the process of obtaining the weighted value of the interpolated pixel using BLI. Equations (1), (2), and (3) help to obtain the weighted value of $f(x, y)$. It uses a weighted average of the four nearest cell centers. The closer an input cell center is to the output cell center, the higher the influence of its value is on the output cell value.

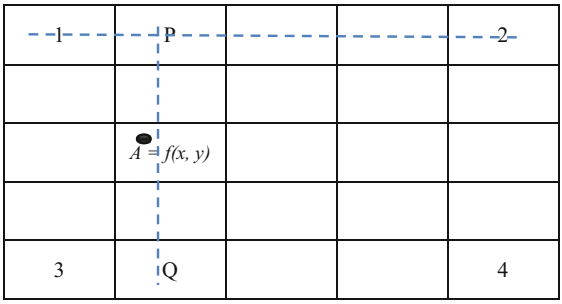

Fig. 2 Image depicting the prediction of weighted value for the interpolated pixel using BLI [5]

This means that the output value could be different to the nearest input, but is always within the same range of values as the input. Since the values can change, BLI is not recommended for use with categorical data. Instead, it should be used for continuous data, such as elevation and raw slope values [13].

The major advantage is that the visual distortion caused by fractional zoom is reduced. The block uses the weighted average of two translated pixel values for each output pixel value. The local gradients for the test images can be investigated [14] by applying an iterative procedure. A perceptually optimum value for each interpolated pixel can be obtained from the local mean value of the inverse gradients.

2.3 Bicubic Interpolation (BCI)

In BCI [7, 8] Lagrange polynomials are used for calculating the weight of the interpolated point A = f(x, y) in a scaled image. A 4 × 4, i.e. 16, adjacent pixels are used. For every single dimension calculation, the values of four pixels positioned at i, j, i − 1, and j + 1 are used. Figure 3 depicts the image for predicting the interpolated pixel value using BCI.

2.3.1 Methodology

The weighted sum of the interpolated pixel can be found by manipulating Eq. (4):

$$f(x,y) = \frac{1}{16} \sum_{l=-1}^{2} \sum_{m=-1}^{2} f(i+l, j+m) u(dx) u(dy) \tag{4}$$

where, $f(i+l, j+m)$ gives the gray value of that pixel, $u(dx)$ gives the variation on the x axis, and $u(dy)$ gives the variation on the y axis.

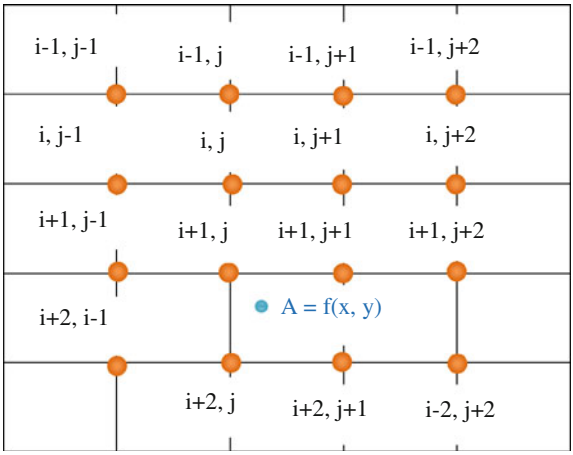

Fig. 3 Image depicting the prediction of the weighted value for the interpolated pixel using BCI [8]

The above figure shows how the 16 pixel values are considered for calculating the weighted average of the final interpolated image. It gives sharper images than the methods described above, but requires more computational time.

2.4 Cubic Spline Interpolation (CSI)

This [11] interpolation technique uses a special type of piecewise polynomial called a spline. A cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set of control points. The second derivative of each polynomial is commonly set to zero at the endpoints, since this provides a boundary condition that completes the system of equations. The interpolation curves are produced as a smooth curve.

CSI is widely used in curve fitting because of its ability to work with both low and high degree polynomials [15]. The depth of the interpolation is variable, and can be set to depend on an absolute or relative error tolerance. It is meant to be easy to interpolate expensive functions that take a lot of time.

2.4.1 Methodology

Given a function $f(x)$ defined on [a, b] and a set of nodes [12],

$$a = x_0 < x_1 < \cdots < x_n = b \tag{5}$$

Equation (5) describes the set of nodes. Considering $f(x)$ is a piecewise cubic polynomial, a cubic spline interpolant, S, is given in Eq. (6),

$$S(x) = \begin{cases} a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3 & \text{if } x_0 \le x \le x_1 \\ a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & \text{if } x_1 \le x \le x_2 \\ . & \\ . & \\ . & \\ a_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3 & \text{if } x_{n-1} \le x \le x_n \end{cases} \tag{6}$$

2.5 *Iterative Linear Interpolation (ILI)*

ILI [4] adopts the fuzzy gradient model to estimate gradients of the target point according to its neighbor sample points in different directions by weighing the gradients using fuzzy membership grades. It estimates the difference between the target point and its neighboring sample points and finally obtains the target point. In 1D signal reconstructions, it uses only three multipliers. Bidirectional interpolation is composed of multiple 1D interpolations. To approximate 2D signal, five 1D ILIs are used, which cost only eight multipliers to obtain similar peak signal-to-noise ratios (PSNR). Further exploiting, multiple 1D interpolation has moderate PSNR performance but better robustness.

2.5.1 Methodology

Fuzzy sets and membership functions are chosen first. The pictorial description of membership functions is shown in Fig. 4.

Consider uniform sampling is done. The below equations show that only two adders and two multipliers are required for calculating the value of interpolated pixel. The left side and right side gradient values are defined in Eqs. (7) and (8).

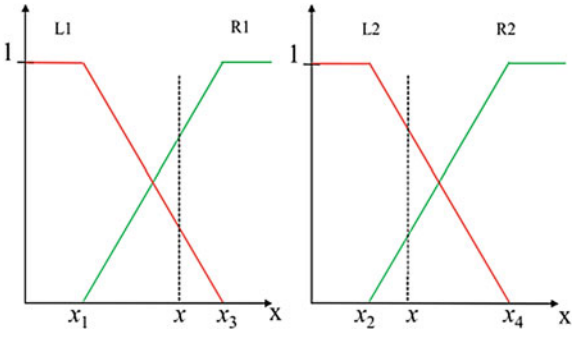

Fig. 4 Image depicting the fuzzy sets and membership functions [4]

$$\hat{f}_L(x) = f(x_2) + (x - x_2) \times q_L$$
$$\hat{f}_R(x) = f(x_3) + (x - x_3) \times q_R. \tag{7}$$

where,

q_L describes the left to right gradient functioinal value.
q_R describes the right to left gradient functional value.

$$q_L = \frac{q_{12} \times L_1 + q_{23} \times R_1}{L_1 + R_1}$$
$$q_R = \frac{q_{23} \times L_2 + q_{34} \times R_2}{L_2 + R_2}. \tag{8}$$

3 Results and Discussion

The following are the pictures considered for comprehensive analysis. These pictures are captured using a Nokia 310 (flower), Samsung GT-S5360 (teddy), and Samsung GT-18552 (vase and bird).

The performances of the algorithms discussed above are compared based on signal-to-noise ratio (SNR), peak signal-to-noise ratio (PSNR), mean squared error (MSE), and processing time. The results were obtained with the help of Matlab R2011a and are tabulated in Table 1.

From Fig. 5, we can see the SNR of various interpolation algorithms, when acted on interpolating images of sizes from 256 × 256 to 1024 × 1024 as tabulated in Table 2.

From Fig. 6, we can observe the PSNR of various interpolation algorithms, when acted on interpolating images of sizes from 256 × 256 to 1024 × 1024 as tabulated in Table 3.

From Fig. 7, we can see the processing time of interpolating techniques and their mean squared error for image sizes from 256 × 256 to 1024 × 1024.

Table 1 Comparison of interpolation techniques based on SNR (dB) for image sizes from 256 × 256 to 1024 × 1024

Interpolation type	Teddy	Flower	Vase	Bird
Nearest neighbor	18.05	21.61	19.12	19.15
Bilinear	22.32	26.37	20.23	22.13
Bicubic	22.56	25.66	20.98	22.24
Cubic spline	22.74	24.35	18.65	21.37
Iterative linear	22.68	25.59	20.11	22.58

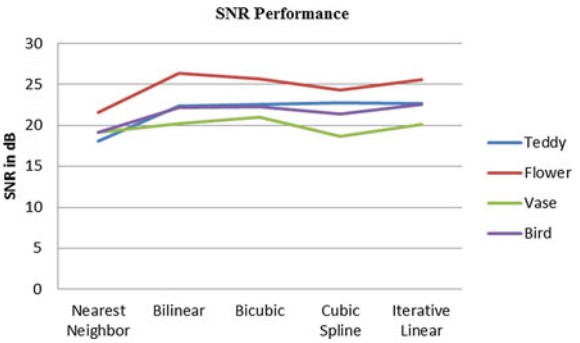

Fig. 5 SNR performance of interpolation techniques

Table 2 Comparison of interpolation techniques based on PSNR (peak signal-to-noise ratio) for image sizes from 256 × 256 to 1024 × 1024

Interpolation type	Teddy	Flower	Vase	Bird
Nearest neighbor	33.50	29.16	23.77	25.18
Bilinear	33.47	29.17	23.84	25.15
Bicubic	32.57	28.84	23.23	23.92
Cubic spline	30.75	29.78	23.89	25.26
Iterative linear	27.96	29.25	23.71	24.65

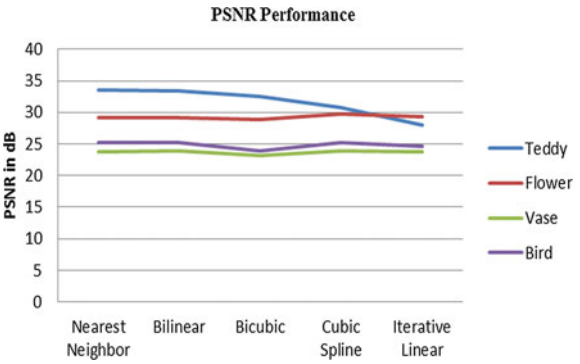

Fig. 6 PSNR performance of various interpolation techniques

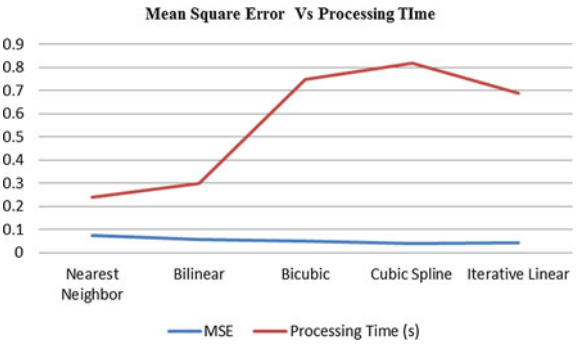

Fig. 7 Mean square error versus processing time for image sizes from 256 × 256 to 1024 × 1024

Table 3 Comparison of interpolation techniques based on MSE (mean square error) and processing time for images of sizes from 256 × 256 to 1024 × 1024

Interpolation type	MSE	Processing time (s)
Nearest neighbor	0.0761	0.24
Bilinear	0.0556	0.3
Bicubic	0.0484	0.75
Cubic spline	0.0395	0.82
Iterative linear	0.0412	0.69

4 Conclusion

In this paper, various interpolation techniques were discussed for their better usage in the field of image interpolation. This survey will assist us in developing a new technique for interpolating images with improved results. As a result future algorithms developed in the future will be optimized for any sort of imaging.

References

1. Gonzalez RC, Woods RE (2008) Digital image processing. Nueva Jersey, p 976
2. Pratt WK (2001) Processing digital image processing, vol 5, no 11
3. Parker JA, Kenyon RV, Troxel DE (1983) Comparison of interpolating methods for image resampling. IEEE Trans Med Imaging MI-2:31–39
4. Hanssen R, Bamler R (1999) Evaluation of interpolation kernels for SAR interferometry. IEEE Trans Geosci Remote Sens Part 1 37(1):318–321
5. Lehmann TM, Gonner C, Spitzer K (1999) Survey: interpolation methods in medical image processing. IEEE Trans Med Imaging 18(11):1049–1075
6. Han D (2013) Comparison of commonly used image interpolation methods. In: Proceedings of the 2nd international conference on computer science and electronic engineering (ICCSEE 2013), pp 1556–1559
7. Haifeng Z, Yongfei Z, Ziqiang H (2010) Comparison of image amplifying method. Mod Electron Tech (24):33–36
8. Amanatiadis A, Andreadis I (2008) Performance evaluation techniques for image scaling algorithms. In: Proceedings of the IEEE international workshop on imaging systems and techniques, pp 114–118 (2008)
9. McKinley S, Levine M (1998) Cubic spline interpolation. Coll Redw 45(1):1049–1060
10. Hou HS, Andrews HC (1978) Cubic splines for image interpolation and digital filtering. IEEE Trans Acoust Speech Signal Process ASSP-26(6):508–517
11. Chen C, Lai C (2014) Iterative linear interpolation based on fuzzy gradient model for low-cost VLSI implementation. 22(7):1526–1538
12. Muresan D, Parks T (2004) Adaptively quadratic image interpolation. IEEE Trans Image Process 690–698
13. Sen Wang, Kejian Yang (2008) An image scaling algorithm based on bilinear interpolation with VC++. J Tech Autom Appl 27(7):44–45
14. Hwang JW, Lee HS (2004) Adaptive image interpolation based on local gradient features. IEEE Signal Process Lett 11(3):359–362
15. Huang JJ, Siu WC, Liu TR (2015) Fast image interpolation via random forests. IEEE Trans Image Process 24(10):3232–3245