Table of Contents

Introduction

Interpolation is the process of estimating the values of a continuous function from discrete samples. Image processing applications of interpolation include image magnification or reduction, subpixel image registration, to correct spatial distortions, and image decompression, as well as others. Of the many image interpolation techniques available, nearest neighbour, bilinear and cubic convolution are the most common, and will be talked about here. Sinc Interpolation provides a perfect reconstruction of a continuous function, provided that the data was obtained by uniform sampling at or above the Nyquist rate. Sinc Interpolation does not give good results within an image processing environment, since image data is generally acquired at a much lower sampling rate. The mapping between the unknown high-resolution image and the low-resolution image is not invertible, and thus a unique solution to the inverse problem cannot be computed. One of the essential aspects of interpolation is efficiency since the amount of data associated with digital images is large.

Image Zooming Methods

The general form for an interpolation function is:

$$g(x) = \Sigma_k \, c_k \, u(\text{distance}_k) \qquad \text{Equation 1}$$

where g() is the interpolation function, u() is the interpolation kernel, distance_k is the distance from the point under consideration, x, to a grid point, x_k, and c_k are the interpolation coefficients. The c_k's are chosen such that $g(x_k) = f(x_k)$ for all x_k. This means that the grid point values should not change in the interpolated image.

Nearest Neighbour Interpolation

Nearest Neighbour Interpolation, the simplest method, determines the grey level value from the closest pixel to the specified input coordinates, and assigns that value to the output coordinates. It should be noted that this method does not really interpolate values, it just copies existing values. Since it does not alter values, it is preferred if subtle variations in the grey level values need to be retained.

For one-dimension Nearest Neighbour Interpolation, the number of grid points needed to evaluate the interpolation function is two. For two-dimension Nearest Neighbour Interpolation, the number of grid points needed to evaluate the interpolation function is four.

For nearest neighbour interpolation, the interpolation kernel for each direction is:

$$u(s) = \begin{cases} 0 & |s| > 0.5 \\ 1 & |s| < 0.5 \end{cases} \qquad \text{Equation 2}$$

where s is the distance between the point to be interpolated and the grid point being considered. The interpolation coefficients $c_k = f(x_k)$.

Bilinear Interpolation

Bilinear Interpolation determines the grey level value from the weighted average of the four closest pixels to the specified input coordinates, and assigns that value to the output coordinates.

First, two linear interpolations are performed in one direction (horizontally in this paper) and then one more linear interpolation is performed in the perpendicular direction (vertically in this paper).

For one-dimension Linear Interpolation, the number of grid points needed to evaluate the interpolation function is two. For Bilinear Interpolation (linear interpolation in two dimensions), the number of grid points needed to evaluate the interpolation function is four.

For linear interpolation, the interpolation kernel is:

$$u(s) = \begin{cases} 0 & |s| > 1 \\ 1 - |s| & |s| < 1 \end{cases} \qquad \text{Equation 3}$$

where s is the distance between the point to be interpolated and the grid point being considered. The interpolation coefficients $c_k = f(x_k)$.

Cubic Convolution Interpolation

Cubic Convolution Interpolation determines the grey level value from the weighted average of the 16 closest pixels to the specified input coordinates, and assigns that value to the output coordinates. The image is slightly sharper than that produced by Bilinear Interpolation, and it does not have the disjointed appearance produced by Nearest Neighbour Interpolation.

First, four one-dimension cubic convolutions are performed in one direction (horizontally in this paper) and then one more one-dimension cubic convolution is performed in the perpendicular direction (vertically in this paper). This means that to implement a two-dimension cubic convolution, a one-dimension cubic convolution is all that is needed.

For one-dimension Cubic Convolution Interpolation, the number of grid points needed to evaluate the interpolation function is four, two grid points on either side of the point under consideration. For Bicubic Interpolation (cubic convolution interpolation in two dimensions), the number of grid points needed to evaluate the interpolation function is

16, two grid points on either side of the point under consideration for both horizontal and vertical directions. The grid points needed in one-dimension and two-dimension cubic convolution interpolation are shown in Figure 1.

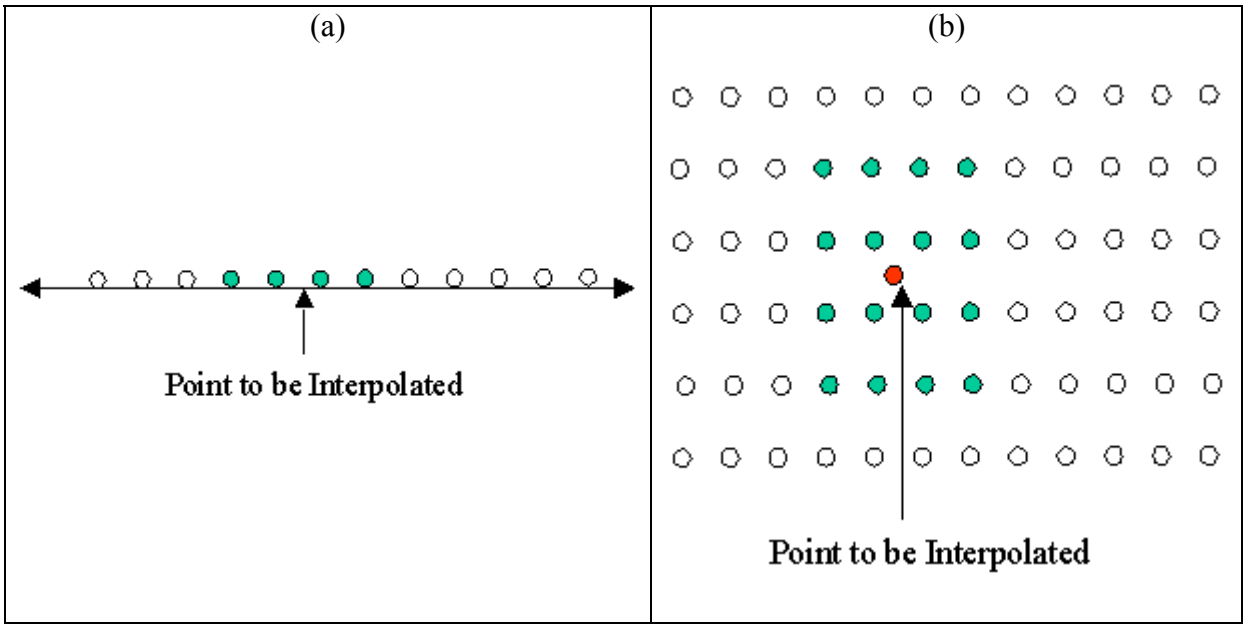


Figure 1. Grid points need in (a) one-dimension and (b) two-dimension cubic convolution interpolation.

The one-dimension cubic convolution interpolation kernel is:

$$u(s) = \begin{cases} 3/2|s|^3 - 5/2|s|^2 + 1 & 0 \le |s| < 1 \\ -1/2|s|^3 + 5/2|s|^2 - 4|s| + 2 & 1 \le |s| < 2 \\ 0 & 2 < |s| \end{cases} \qquad \text{Equation 4}$$

where s is the distance between the point to be interpolated and the grid point being considered. A plot of the one-dimension cubic convolution interpolation kernel vs s is shown in Figure 2.

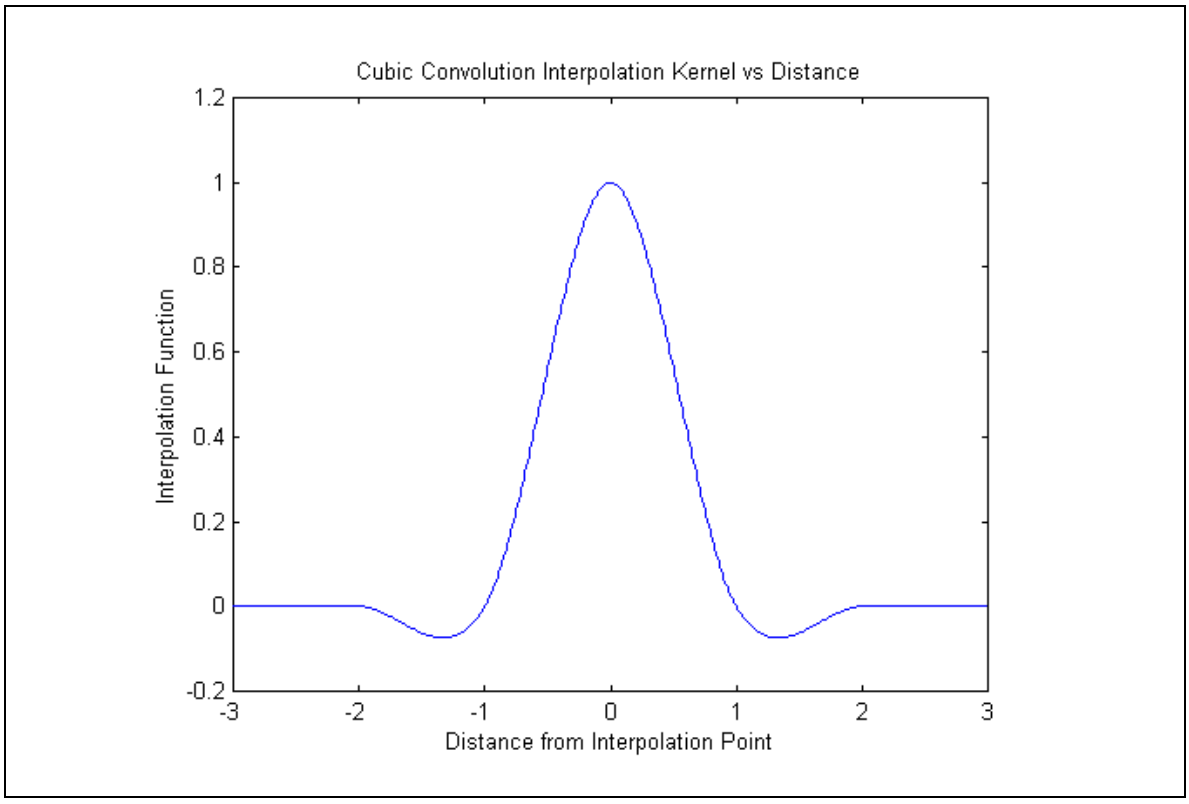


Figure 2. Cubic Convolution Interpolation Kernel vs s.

For two-dimensional interpolation, the one-dimensional interpolation function is applied in both directions. It is a separable extension of the one-dimensional interpolation function. Given a point (x, y) to interpolate, where $x_k < x < x_{k+1}$ and $y_k < y < y_{k+1}$, the two-dimensional cubic convolution interpolation function is:

$$g(x,y) = \sum_{l=-1}^{2}\sum_{m=-1}^{2} c_{j+l,k+m}\, u(dis\tan ce_x)\, u(dis\tan ce_y) \qquad \text{Equation 5}$$

where u() is the interpolation function of Equation 4, and $distance_x$ and $distance_y$ are the x and y distances from the four grid points in each direction. For non-boundary points, the interpolation coefficients, c_{jk}'s are given by $c_{jk} = f(x_j, y_k)$.

Comparison of Image Zooming Methods

The nearest-neighbour and bilinear interpolation methods are very practical and easy to apply, due to their simplicity. However, their accuracy is limited and may be inadequate for interpolating high-frequency signals. There is a trade-off between computational complexity and accuracy.

Computational Complexity/Computation Time

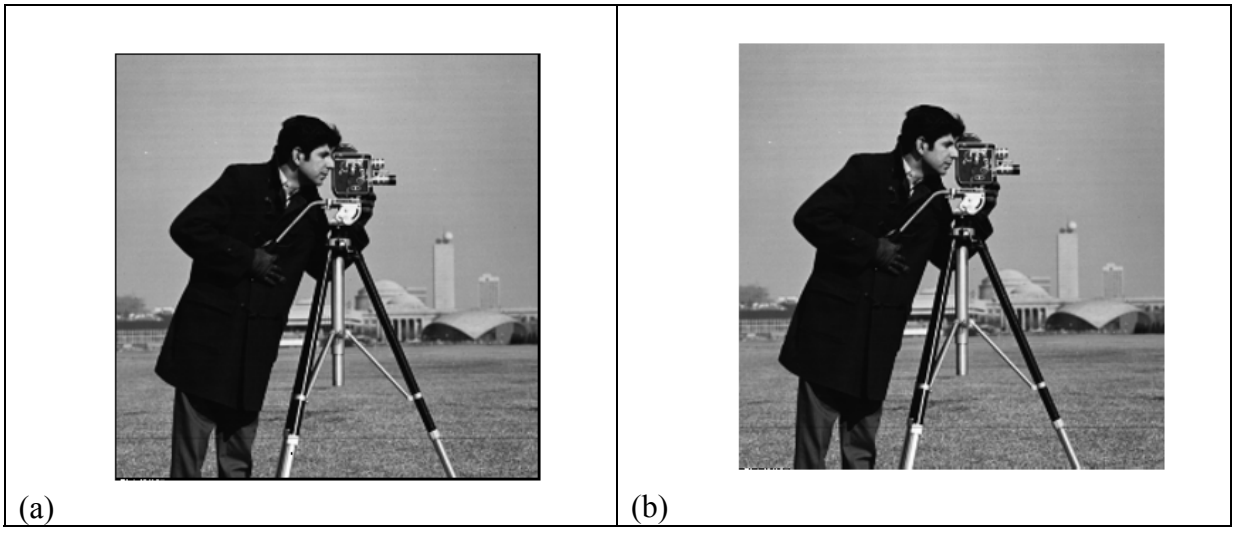
Nearest Neighbour Interpolation is the most efficient in terms of computation time. Bilinear Interpolation requires 3 to 4 times the computation time of Nearest Neighbour Interpolation. Cubic Convolution Interpolation requires about 10 times the computation time of Nearest Neighbour Interpolation.

Accuracy

Nearest Neighbour Interpolation generally performs poorly. This image may be spatially offset by up to ½ a pixel, causing a jagged or blocky appearance. Bilinear Interpolation generates an image of smoother appearance than nearest neighbour interpolation, but the grey levels are altered in the process, resulting in blurring or loss of image resolution.

Matlab's Implementation of Bicubic Interpolation

A comparison of the above implementation of bicubic convolution interpolation with Matlab's implementation shows some interesting results. The interpolated images are shown in Figure 3, while the difference in the images are shown in Figure 4.



(a) (b)

Figure 3. (a) Keys Implementation of Bicubic Convolution Interpolation, and (b) Matlab's Implementation of Bicubic Convolution Interpolation.

Figure 4. Difference between Keys Implementation of Bicubic Convolution Interpolation and Matlab's. Darker means less difference.

Both methods are almost identical, except at image boundaries. This isn't surprising since boundary conditions for Keys implementation were not added.

Conclusion

There are a number of techniques that can be used to enlarge an image. The three most common were presented here. All try to emulate as close as possible to an ideal Low Pass Filter. Keys implementation of Bicubic Convolution Interpolation gave the best results in terms of image quality, but took the greatest amount of processing time. Finally, Keys implementation was shown to be equivalent to that of Matlab.

References

[1] Keys, R., "Cubic Convolution Interpolation for Digital Image Processing", IEEE Trans on ASSP, vol ASSP-29, No. 6, Dec 1981

[2] Gleicher, M., "A Brief Tutorial on Interpolation for Image Scaling", Dec 1999

[3] Bourke, P., "Bicubic Interpolation for Image Scaling", May 2001

[4] Carlson, B., "Image Interpolation and Filtering", March 8, 2000

[5] "Appendix A: Bicubic Interpolation",
http://www.npac.syr.edu/projects/nasa/MILOJE/final/node36.html

[6] "Image Processing Fundamentals – Convolution-based Operations",
http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Convolut-2.html

[7] "Examples for Interpolators",
http://www.geo.tudelft.nl/fmr/research/insar/dig/bibliography/papers/interp/node3.html

[8] Schultz, R., Stevenson, R., "Extraction of High-Resolution Frames from Video Sequences", Accepted to appear in IEEE Trans on Image Processing

[9] "Chapter 4: Image Pre-processing",
http://www.eng.iastate.edu/ee528/sonkamaterial/chapter_4.html

[10] "Interpolation Theory",
http://sepwww.stanford.edu/public/docs/sep107/paper_html/node20.html

[11] "Introduction to Map Algebra",
http://www.quantdec.com/SYSEN597/GTKAV/section9/map_algebra.htm

Appendix I: Interpolation Source Code