Faster Image Zooming using Cubic Spline Interpolation Method

Shabana Parveen
M.Tech scholar
Marudhar Engineering College
Bikaner, Rajasthan
saboparveen@gmail.com

Ms. Rajbala Tokas
Assistant Professor
Marudhar Engineering College
Bikaner, Rajasthan
rajtokas26@gmail.com

***Abstract*: -** This paper proposes an efficient image interpolation algorithm using cubic spline interpolation method. The proposed interpolation algorithm is done in two steps. In the first step, the area of the image is enlarged by inserting zeros in between every two columns and rows according to the zooming intensity as input by the user. Second step involves the estimation of correct values of those zeros so that after zooming the image does not ruptures with absurd values of the pixel. The cubic spline method is efficiently programmed in MATLAB 7.12 and results were according to the requirement.

Keywords:- *Grey Scale Image, Zooming, cubic spline, interpolation*

*****

1. Introduction

Interpolation in image processing is a method to increase the number of pixels in a digital image. Interpolation has been widely used in many image processing applications such as facial reconstruction, multiple description coding, and super resolution. Interpolation based super resolution has been used for a long time, and many interpolation techniques have been developed to increase the quality of this task [1]. Xin Li et al, proposed a hybrid approach for interpolation using switching between bilinear interpolation local covariance based adaptive interpolation. In this method local covariance was first calculated for the low resolution image. These covariance values were then used for interpolating the low resolution images using the property of geometric duality between the low resolution covariance and the high resolution covariance [2].
Zhou Dengwen et al, proposed a technique for color image interpolation. In this technique the edge pixels were distinguished from the smooth pixels based on a preset threshold value and then the pixels were interpolated using extended bi-cubic convolution algorithm [3].
Jinglun Shi proposed an algorithm for image interpolation using variation based approach. In this algorithm the pixels of low resolution image were projected using various angular orientations and the sum of absolute difference at various orientations were calculated. The pixels were then interpolated along the direction of minimum variation [4].
Computation and measurement can be performed to determine a relation between two variables. The result is given as a set of discrete data points in a plane. Knowing that the relation can be represented by a smooth curve, then a curve fitting used to set of data points so that it will pass through all the points. Manual drawing is the most primitive method for this purpose and results in a reasonable curve if it is done by a well-trained scientist or engineer. But, since it is very tedious and time consuming a computer is used to draw the curve. The computer must then be provided with necessary instructions for mathematically interpolating additional points between the given data points [5]. The common difficulty is that the resultant curve sometimes shows unnatural wiggles. This seems inevitable if we make any assumption concerning the functional form for the whole set of given data points other than the continuity and the smoothness of the curve.

2. Interpolation Overview

The word "interpolation" originates from the Latin verb *interpolare*, a contraction of "inter," meaning "between," and "polare," meaning "to polish." That is to say, to smooth in between given pieces of information. It seems that the word was introduced in the English literature for the first time around 1612 and was then used in the sense of "to alter or enlarge [texts] by insertion of new matter". The original Latin word appears to have been used first in a mathematical sense by Wallis in his 1655 book on infinitesimal arithmetic [5]. In antiquity, astronomy was all about time keeping and making predictions concerning astronomical events. This served important practical needs: farmers, e.g., would base their planting strategies on these predictions. To this end, it was of great importance to keep up lists—so-called ephemerides—of the positions of the sun, moon, and the known planets for regular time intervals. Obviously, these lists would contain gaps, due to either atmospheric conditions hampering observation or the fact that celestial bodies may not be visible during certain periods. From his study of ephemerides found on ancient astronomical cuneiform tablets originating from Uruk and Babylon in the Seleucid period (the last three centuries BC), the historian-mathematician Neugebauer concluded that interpolation was used in order to fill these gaps. Apart from linear interpolation, the tablets also revealed the use of more

complex interpolation methods. Precise formulations of the latter methods have not survived, however. By the beginning of the 20th century, the problem of interpolation by finite or divided differences had been studied by astronomers, mathematicians, statisticians, and actuaries and most of the now well-known variants of Newton's original formulae had been worked out. This is not to say, however, that there are no more advanced developments to report on. Quite to the contrary, already in 1821, Cauchy studied interpolation by means of a ratio of two polynomials and showed that the solution to this problem is unique, the Waring–Lagrange formula being the special case for the second polynomial equal to one.
Generalizations for solving the problem of multivariate interpolation in the case of fairly arbitrary point configurations began to appear in the second half of the 19th century, in the works of Borchardt and Kronecker.

3, The Interpolation Problem
Typically, you will have a sampled data system representing your image. With a two dimensional array of samples usually linearly spaced in the x (horizontal) and y (vertical) directions. For one reason or another, you will require a group of new sample points intermediate to your input pixels. Typically, one new pixel will have four nearest neighbors on a rectangular grid.

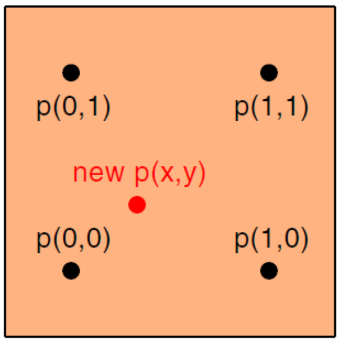


Fig.1 New pixel insertion

It is often simplest to assume a unit square between the four nearest pixels. Our new and sought after pixel will then be some x fraction and y fraction into this unit square. Your interpolation problem then consists of finding suitable values for these fractions. Your sought after new pixel position should be the result of some pixel -by- pixel calculation seeking to change the size or distortion of the image. Both the new x and y results will have an integer portion that decides which initial group of four pixels to use. Plus a residue or fractional remainder portion that positions you within the selected four sample square. With the Nearest Neighbor scheme, you just grab the nearest pixel and use it. One simple way to do this is to round your x value and add it to a rounded and doubled y value. This will give you four integers 0, 1, 2, and 3 that can use table lookup or case commands to read one of the four corner pixels. We might start with this sample data set array and use it to compare the different interpolation methods [6].

4. Bicubic Interpolation Method
Bicubic interpolation is an extension of cubic spline interpolation on a two dimensional regular grid. Bilinear interpolation, which only takes 4 pixels (2x2) into account, bicubic interpolation considers 16 pixels (4x4). The fundamental idea behind cubic spline interpolation is based on the engineer's tool used to draw smooth curves through a number of points. Cubic method determines the grey level from the weighted average of the 16 closest pixels to the specified input coordinates, and assigns that value to the output coordinates. Cubic method calculates a distance weighted average of a block of 16 pixels from the original image which surround the new output pixel location. As with bilinear interpolation this method results in completely new pixel values. A flexible strip is then bent across each of weights, resulting in a pleasingly smooth curve. Bicubic requires about 10 times the computation time required by the nearest neighbor method. The image is theoretically slightly sharper than that produced by bilinear interpolation and it does not have the disjointed appearance produced by nearest neighbor interpolation [7].

5. Pseudo code of bicubic:
Step: 1 Input original image and convert into matrix form.
Step: 2 Insert new pixels according to zooming number in columns.Fig.2

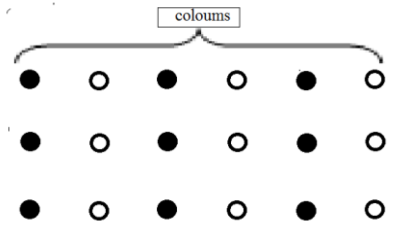


Fig. 2 Column Insertion

Step: 3 Insert new pixels according to zooming number in rows. Fig. 3

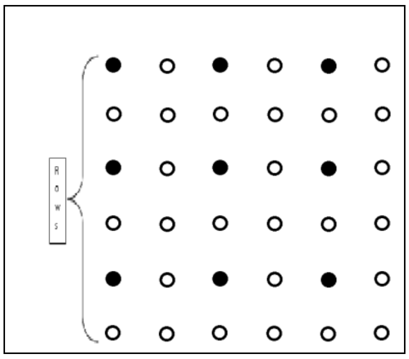


Fig.3 Row Insertion

Step: 4 Delete last row and column. Fig.4

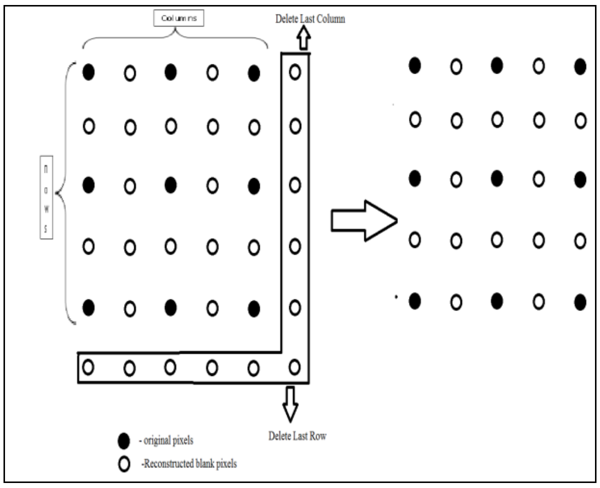


Fig .4 Deletions last Row and Colum

Step: 5 Create a matrix corresponding by cubic spline method.

$$\begin{bmatrix} 6 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix}$$

Step: 6 calculate the values of M in terms of y using matrix of cubic spline.

$M_n = 2M_{n-1} – M_{n-2}$

Step: 7 Repeated previous step until whole image does not covered.

A Cubic Spline Third Degree Polynomial is defined by –

$$s(x_i) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

$$a_i = \frac{M_{i-1} - M_i}{6h}$$

$$b_i = \frac{M_i}{2}$$

$$c_i = \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} - 2M_i}{6}\right) h$$

$$d_i = y_i$$

Where $s_i''(x_i) = M_i$

Which lead to the matrix equation as follows:

$$\begin{bmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \\ M_n \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix}$$

It assigns *M*1 to be 2*M*2 - *M*3 and *Mn* to be 2*Mn*-1 – *Mn*-2. This causes the curve to degrade to a single cubic.

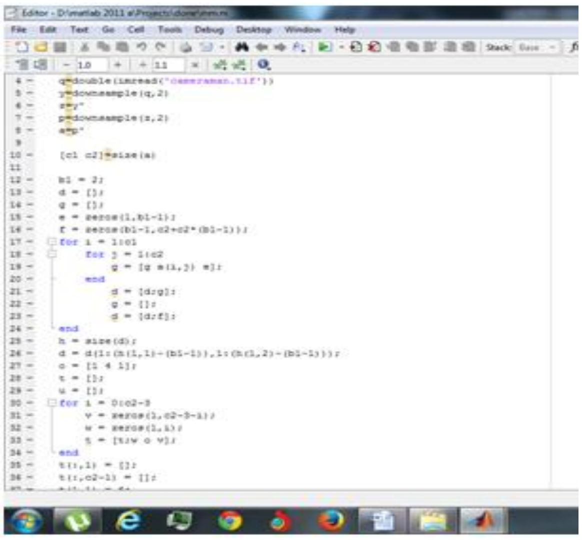
$$\begin{bmatrix} 6 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix}$$

This is the final Matrix for cubic spline interpolation Technique. In this matrix, y1 to yn is related to original image coordinates and h is total image matrix size like that if an image have 3*3 sizes then h is equals to the 9.

Cubic Spline is another method used for reconstructed image. A cubic spline function is particularly used for curve fitting between the existing points. Cubic spline is most widely used to fit a smooth continuous function through discrete data. This function plays an important role in image processing and image interpolation. Cubic spline is useful in implementing high quality image zooming. If we joined several point in a common curve then we uses cubic spline technique because it uses derivatives at each end points.

6. Procedure For Zoomed Image Through Matlab-

1. Open MATLAB Software.
2. Find the editor window.
   File >> New >> Script
3. Write Down the Program in Editor Window.


Editor - D:\matlab 2011 a\Projects\done\mm.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1

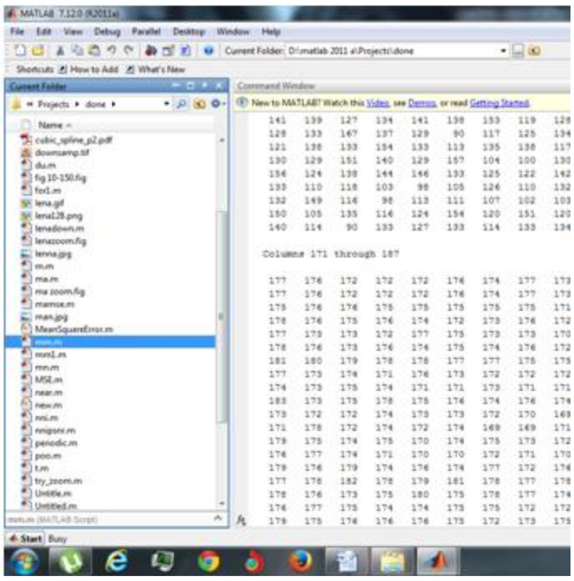
```
q=double(imread('cameraman.tif'))
y=downsample(q,2)
z=y'
p=downsample(z,2)
a=p'

[c1 c2]=size(a)

b1 = 2;
d = [];
g = [];
e = zeros(1,b1-1);
f = zeros(b1-1,c2+c2*(b1-1));
for i = 1:c1
    for j = 1:c2
        g = [g a(i,j) e];
    end
        d = [d;g];
        g = [];
        d = [d;f];
end
h = size(d);
d = d(1:(h(1,1)-(b1-1)),1:(h(1,2)-(b1-1)));
o = [1 4 1];
t = [];
u = [];
for i = 0:c2-3
    v = zeros(1,c2-3-i);
    w = zeros(1,i);
    t = [t;w o v];
end
t(:,1) = [];
t(:,c2-1) = [];
```

Fig. 5 Layout of Editor Window

4. After completion the program, click the RUN button.
5. A new window is opened with the columns and Rows matrix, shown in Fig. 6



```
   141   139   127   134   141   138   133   119   12
   128   133   167   137   129    90   117   125   13
   121   138   133   154   133   113   135   138   11
   130   129   151   140   129   157   104   100   13
   156   124   138   144   146   133   125   122   14
   133   110   118   103    98   105   126   110   13
   132   149   116    98   113   111   107   102   10
   150   105   135   116   124   154   120   151   12
   140   114    90   133   127   133   114   133   13

  Columns 171 through 187

   177   176   172   172   172   176   174   177   17
   177   176   172   172   172   176   174   177   17
   175   176   176   175   175   175   175   175   17
   178   176   175   176   174   172   173   176   17
   177   173   173   172   177   175   173   173   17
   178   176   173   176   174   175   174   176   17
   181   180   179   178   178   177   177   175   17
   177   173   174   171   176   173   172   172   17
   174   173   175   174   171   171   173   171   17
   183   173   175   178   175   176   174   176   17
   173   172   172   174   173   173   172   170   16
   171   178   172   174   172   174   169   169   17
   179   175   174   175   170   174   175   173   17
   176   177   174   171   170   170   172   171   17
   179   176   179   174   176   174   177   172   17
   177   178   182   178   179   181   178   177   17
   178   176   173   175   180   175   178   177   17
   176   177   175   174   174   175   175   172   17
   179   175   176   176   176   175   172   173   17
```

Fig. 6 Layout of MATLAB to Calculation Window

6. Original image(cameraman) with MATLAB.

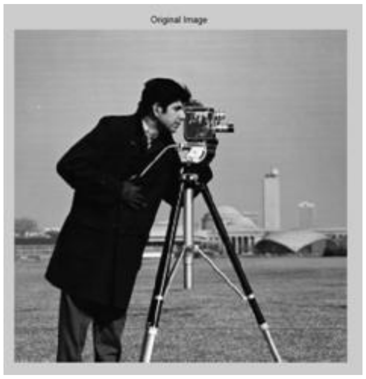


Fig. 7 Layout of Original Image (cameraman)

7. Downsampled Image

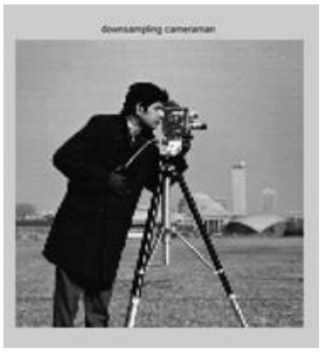


Fig. 8 Layout of down sampled Image

8. Zoomed Image (2*X)

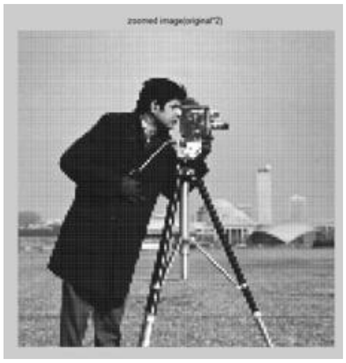


Fig. 9 Layout of Zoomed Image of cameraman

9. Find PSNR between original image and zoomed image

Mathematical Formula of PSNR

Now, The PSNR ratio is defined by

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{\sqrt{MSE}}\right)$$

$$PSNR = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Here MAXI – maximum possible pixel value of the image
When Pixel are represented using 8 bits per sample, then
$MAX_I = 2^B - 1$
B=Bits per sample
$MAX_I = 2^8 - 1=255$
Here, the mean Square error is calculated by

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

Where m, n –axis points
I – original image
K – Reconstructed image
The experimental result shown in following Table.1

Image	PSNR(db)
Cameraman	50.4751

Table 1.Experimental Result of image PSNR

REFERENCES

[1] R. Jansi, Asnath Victy Phamila, and R. Amutha, "Color Image Interpolation using Optimal Edge Detector", International Journal of Innovation and Applied Studies ISSN 2028-9324 Vol. 3, No. 3, pp. 813-819, July 2013

[2] Xin Li and Michael T. Orchard, "New Edge-Directed Interpolation," *IEEE Transactions on Image Processing,* vol. 10, no. 1, pp. 1521-1527, October 2001.

[3] Zhou Dengwen and Shen Xiaoliu, "An Effective Color Image Interpolation Algorithm," *International Congress on Image and Signal Processing*, Vol. 2, pp. 984-988, 2011.

[4] Jinglun Shi and Zhilong Shan, "Image Interpolation Using a Variation-Based Approach," *IEEE Proceedings*, 2011.

[5] Ahmed N. H. Alnuaimy, Mahamod Ismail, Mohd A. M. Ali and Kashmiran Jumari, "An Improved Method for Successive Data Schism and Interpolation", WSEAS Transactions On Mathematics, Issue 12, Volume 8, p.p. 712 – 722, December 2009

[6] Don Lancaster, " A Review of Some Image Pixel Interpolation Algorithms", Synergetics.

[7] Don Lancaster, "A Review of Some Image Pixel Interpolation Algorithms", Synergetic, Box 809, Thatcher, AZ 85552, 2007 as Guru Gram http:// www.tinaja.com (928) 428-4073.